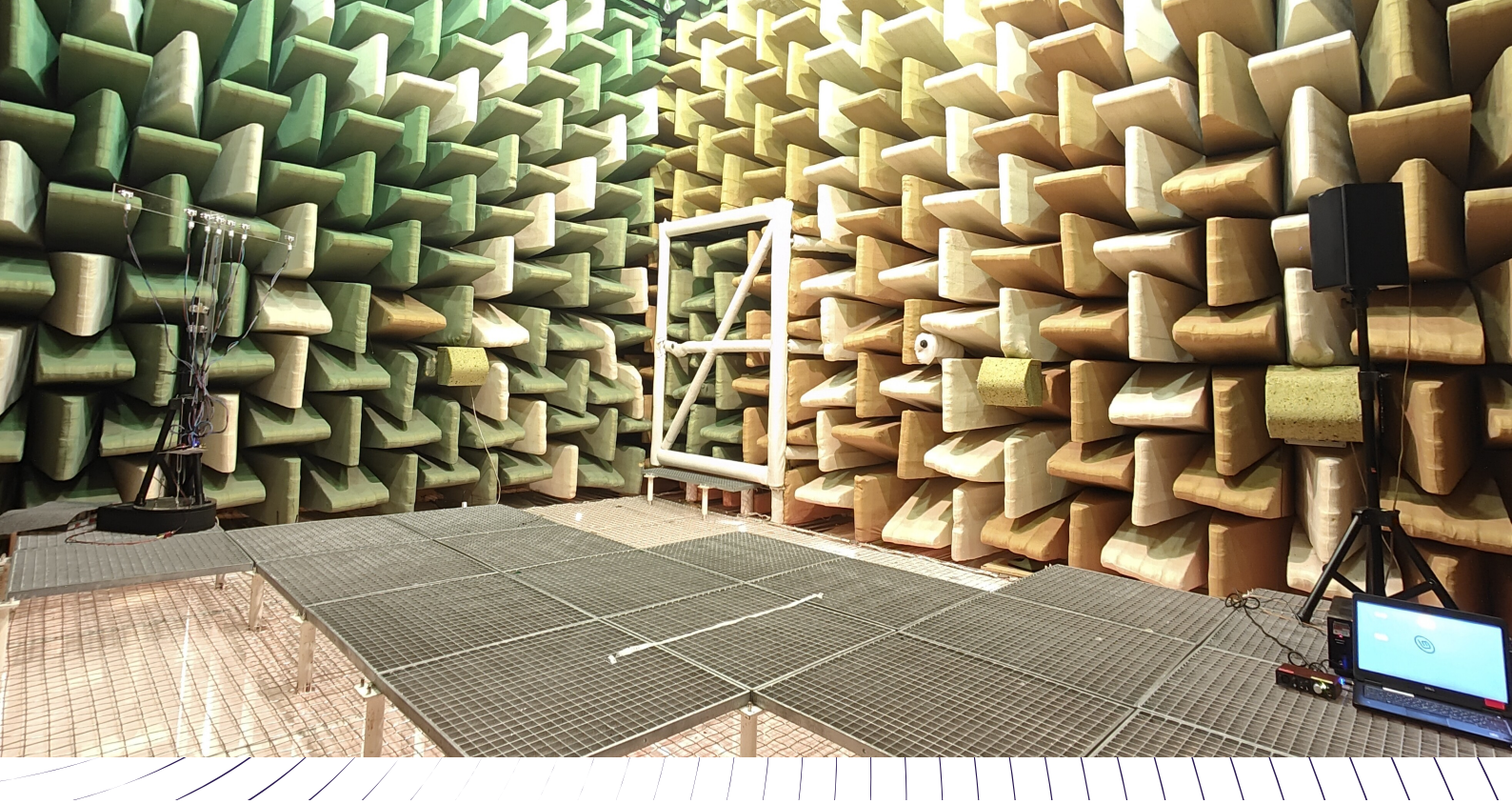
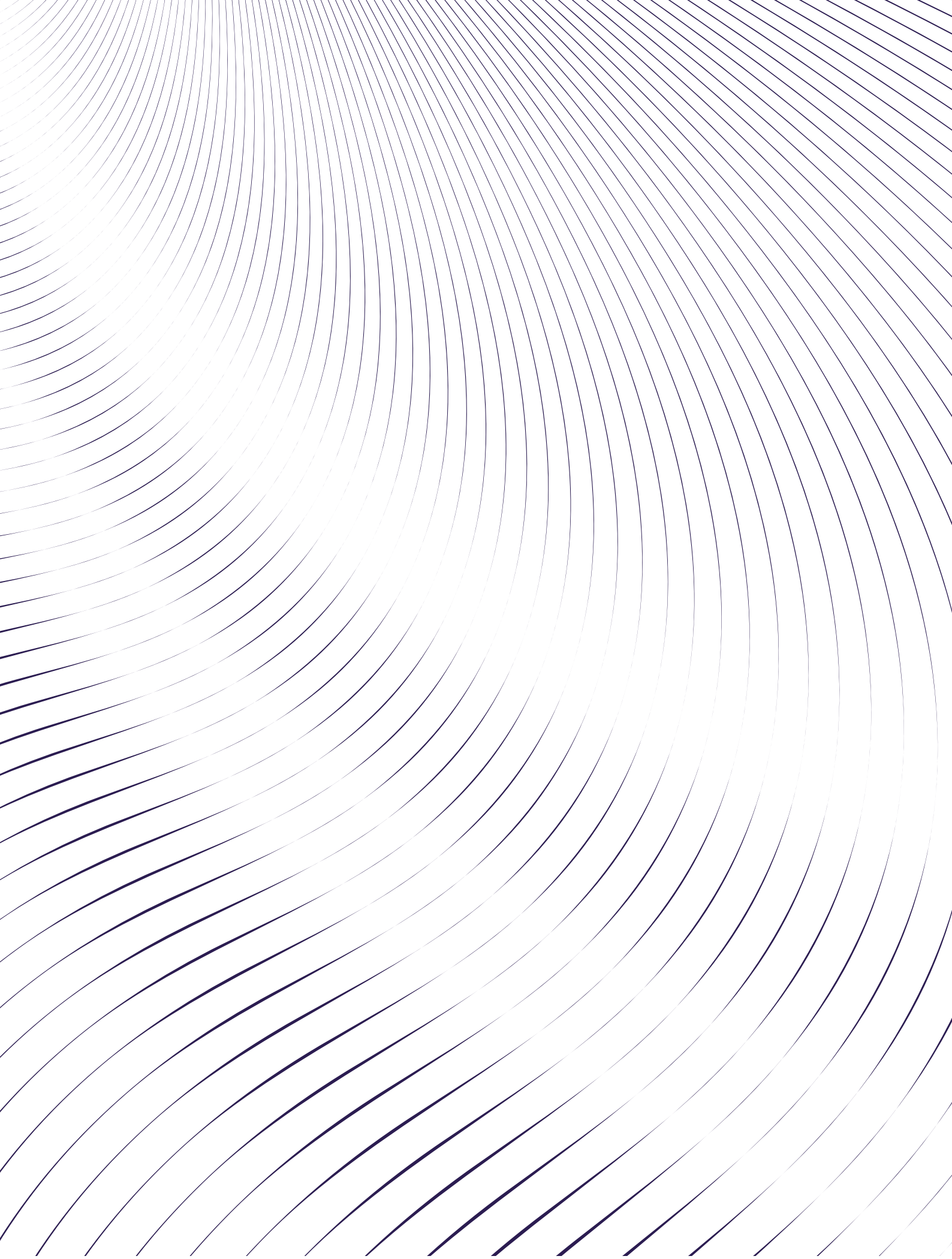


Autonomous Splash Detection System for Data Collection: A Subaqueous Multi-DAS Beamforming Approach

Bjørn Højmose Grevenkop-Castenskiold & Jonas Emil Nielsen

Electronic Systems, ES10, 05-2024

10th Semester Project



10th semester

Electronic Systems
Fredrik Bajers Vej 7A
9200 Aalborg Øst
http://www.tech.aau.dk

Title:

Autonomous Splash Detection System for Data Collection: A Subaqueous Multi-DAS Beamforming Approach

Project:

Master's Thesis

Project period:

January 2024 - May 2024

Participants:

Bjørn Højmose Grevenkop-Castenskiold
Jonas Emil Nielsen

Supervisors:

Flemming Christensen

Pages: 123
Appendices: 3
End date: 31-05-2024

Abstract:

This report describes the development of a system capable of autonomously detecting, recording, and offloading splash events caused by people at bathing locations with the intent of creating a dataset to be used for training algorithms capable of confidently detecting potential drowning accidents in harbours. A system was designed, consisting of initial splash detection, calculation of direction of arrival (DOA) using time difference of arrival (TDOA), delay-and-sum (DAS) beamforming, confirmation of splash, recording and offloading data to a server. A subaqueous DAS beamformer was implemented on an ESP32-S3, consisting of three uniform linear arrays (ULAs), covering a frequency band each, to ensure suppression in the entire frequency band, receiving audio from 7 24-bit I2S microphones, impedance matched to water using nitrile gloves, over SPI from a PSoC 5LP, after which it was offloaded to a server using Wi-Fi. An automatic test system was implemented and used to verify the beampattern of the beamformer in air, by playing and recording an exponential sine sweep (ESS) and single tones at a time (STT) for each angle in an anechoic chamber. The measured beampatterns were then compared to simulations, where likeness was observed as well as some noise sources. The system was not evaluated in a watery setting, but is estimated to be ready for deployment with a few further developments and permit.

Prefatory Note

This project report was written by group ES10-1020 from *Electronic Systems* in spring 2024. Throughout the report, references to scripts can be found through hyperlinks in Appendix A, which will lead to a GitHub repository.

Reading Guide

The source references in this report will be shown by numbers in brackets [x], and the same number is used to show the corresponding source in the bibliography seen at the back of the report. The source references follow the standard DS/ISO 690, as a guideline for the numbering of sources. If the source is a book, then it is specified with author, title, version, and publisher. Websites are specified with author, title, and date. Figures and tables are enumerated in relation to which chapter it is located in, the first figure in chapter one would then be referred to as Figure 1.1 and the next would be referred to as 1.2, etc. To each figure, a caption is added, which can be seen under the given figure. Lastly, the report follows the standard ISO 80000 as a guideline for all written equations.

The report is best read in full colour due to colour utilisation in figures and plots throughout the report.

The plots of beampatterns are mostly shown in both 2- and 3D throughout the report. The 2D plot shows the beampattern at specific frequencies denoted in the legend of the plot, whereas the 3D plot shows the broadband beampattern. The 2D plots are faced upwards on the page, while the 3D plots face downwards. As the beampatterns are symmetrical about the array's axis, meaning the facing direction is irrelevant, the steering angle in the 2D plots is denoted between 0-180°, and between 180-360° for the 3D beampatterns. The steering angle is always included in the caption as "80/170°" for instance, where the former is the steering angle for the 2D plot and the latter is for the 3D plot. They are, however, identical if flipped due to the symmetry. The beampattern plots are illustrated using *cylindricalPlot.mlx* and *circPlot.mlx* in Appendix A.

The authors would like to thank Assistant Engineer Claus Vestergaard for assistance with equipment in the anechoic chamber.

Contents

1 Introduction

In 2023 a report from the Danish National Institute of Public Health regarding deaths by drowning came out, presenting drowning statistics from 1970 to 2021. In this period, 8 407 people died by drowning, whereof 3 682 were accidents (43.8 %), 3 926 (46.7 %) were suicides and 36 (0.4 %) were murders [1]. In the same time period (1970-2021), the amount of drowning deaths per year has trended downwards as seen on Figure 1.1, but a significant amount of drowning is still happening every year [1].

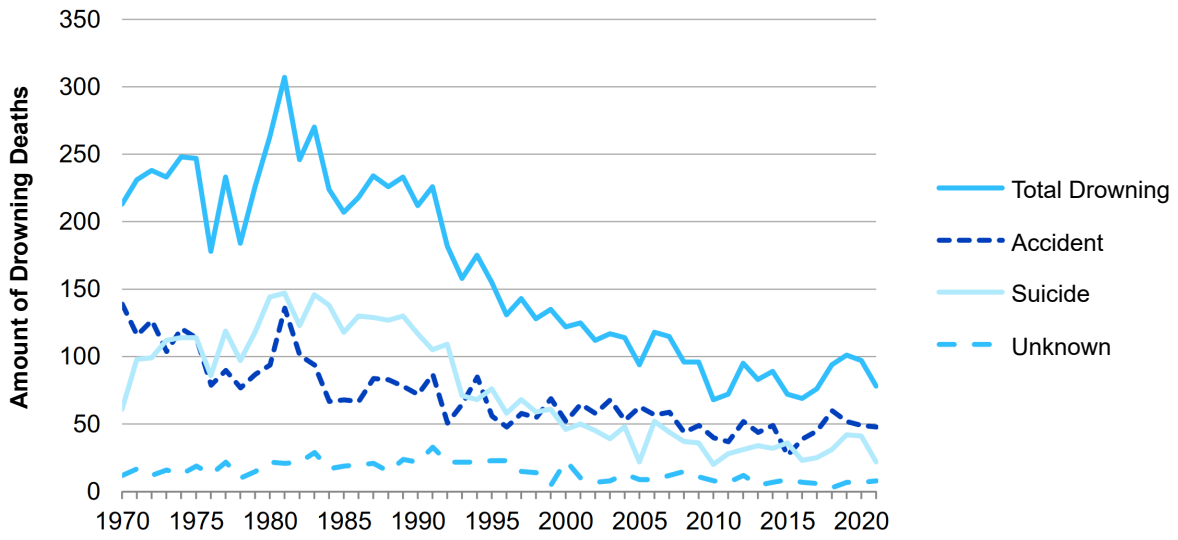


Figure 1.1. Fatal drowning incidents of involving Danish people in the period from 1970-2021 grouped by incident type [1].

The body of water of fatal drowning incidents are shown for the 10-year average (2011-2020) and specific numbers from 2021 on Figure 1.2. While most drown along the coast, this is also where the most work is done to prevent it: coastguards, beach flags, and so on. The second largest danger zone is the harbour, where little supervision is done.

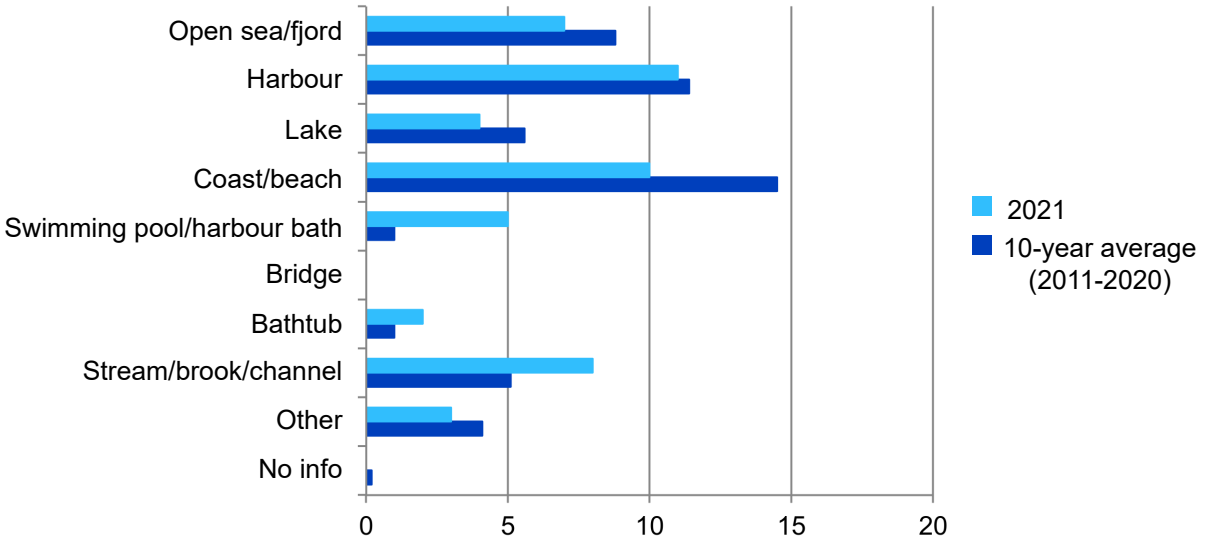


Figure 1.2. Fatal drowning incidents in Denmark by body of water types [1].

Efforts have been made in later years to mitigate drowning in harbours, such as the thermal cameras installed in Aalborg Harbour in 2015. These cameras monitor ∼200 meters of Aalborg Waterfront from Limfjordsbroen along Honnørkajen and have saved two people in the testing period (as of feb 2022). There are however around two kilometres of critical waterfront, with heightened risk of drowning. Securing all of this would take around ten thermal cameras at a cost of around two million DKK along with a need for more people monitoring them. [2]

A Master's thesis by Frederik Sidenius Dam explored the feasibility of utilising hydrophones to detect a fall into water [3]. The thesis examined the performance of cheap piezoelectric and electret transducers encased in a nitrile rubber glove. Both solutions performed nearly as well as commercial hydrophones and provided promising measurements in regards to distinguish the falls from noise.

Furthermore, a project by Axel Villads Burford Toft, Kaj Mørk, Lukas Bisgaard Kristensen, Máté Tallósi, and Tudor Razvan-Tatar attempted to design an algorithm to detect the sound of a person falling into water through machine learning [4]. However, the group suffered from a lack of training data, as they had to perform and record the jumps themselves.

The splash from a person might vary heavily depending on the energy and contact area of the impact and the individual's clothing. Additionally, attempting to imitate an accidental or inebriated fall deliberately might also not be exactly representative. However, mimicking vast amounts of falls with sufficient variation is tedious/unfeasible. Therefore, the initial problem statement is as follows:

How is a vast amount of representative data gathered with the purpose to develop an algorithm to detect a person falling into water?

System Analysis 2

The purpose of this chapter is to inspect and analyse the aspects of subaqueous acoustics and splash characteristics. Rather than assembling subjects with the intent to record the audio of their falls into water, another approach is desired. Instead, the possibility to passively surveil a bathing location where jumps into water naturally occur is explored. The shortcomings of this approach is the unknown difference between a deliberate fall and an accidental fall, and the lack of day to day clothing on the subjects. An illustration of the case is shown in Figure 2.1, where a system is setup to monitor a bathing location with a microphone with the intent to record the subaqueous audio of people jumping into the water.

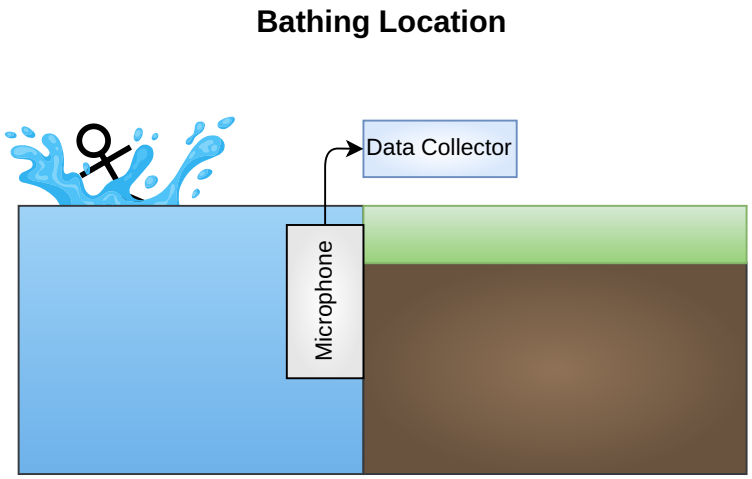


Figure 2.1. Figure depicting a person jumping into water at a bathing location with a data collector nearby.

The general acoustic characteristics of the area should imitate those of the end use-case, meaning noise sources such as boat engines, seagulls, and so on should be present at the recording location. Ideally, the location has an elevated edge such the height of the jumps matches a harbour. The system should be operational in both fresh- and saltwater.

As the strategy is to passively surveil the bathing location, some event detection should be developed in order to know when to start the data recording and avoid recording when people are not jumping into the water.

2.1 Splash Characteristics

In order to establish an idea of how the audio from a splash will appear, measurements from hydrophones are examined. The spectrograms of a person jumping into water measured by

hydrophones placed one and 11.5 meters away are seen in Figure 2.2 and 2.3, respectively. The data is taken from one of the previous projects ([4]) and both are recordings of the same jump into water. The audio clips consists of 10 seconds of background noise, the splash, and subsequent swimming.

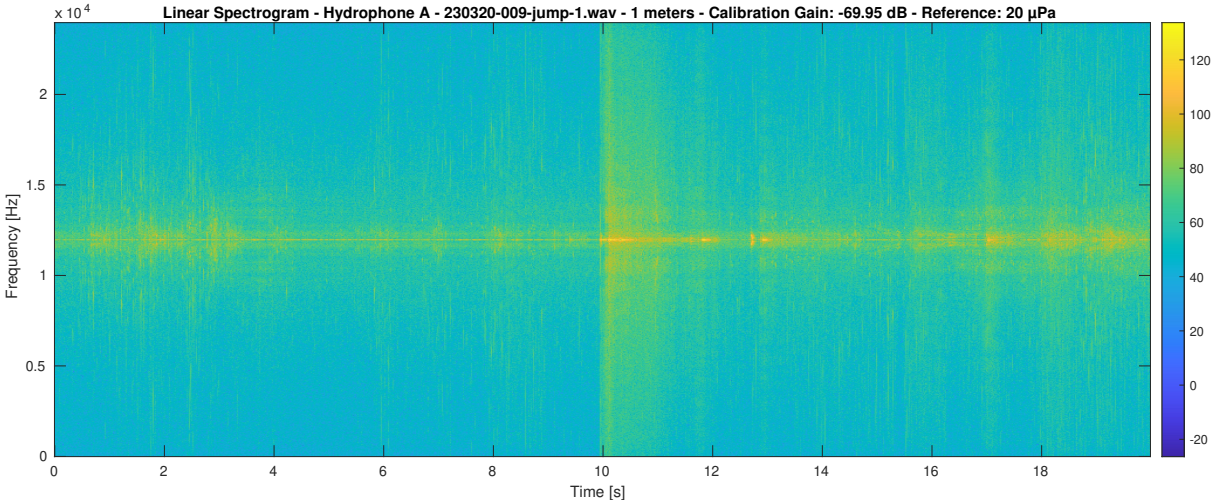


Figure 2.2. Spectrogram of a person jumping into water measured by a hydrophone placed one metre away from the impact [4]. The time of impact starts at 10 seconds.

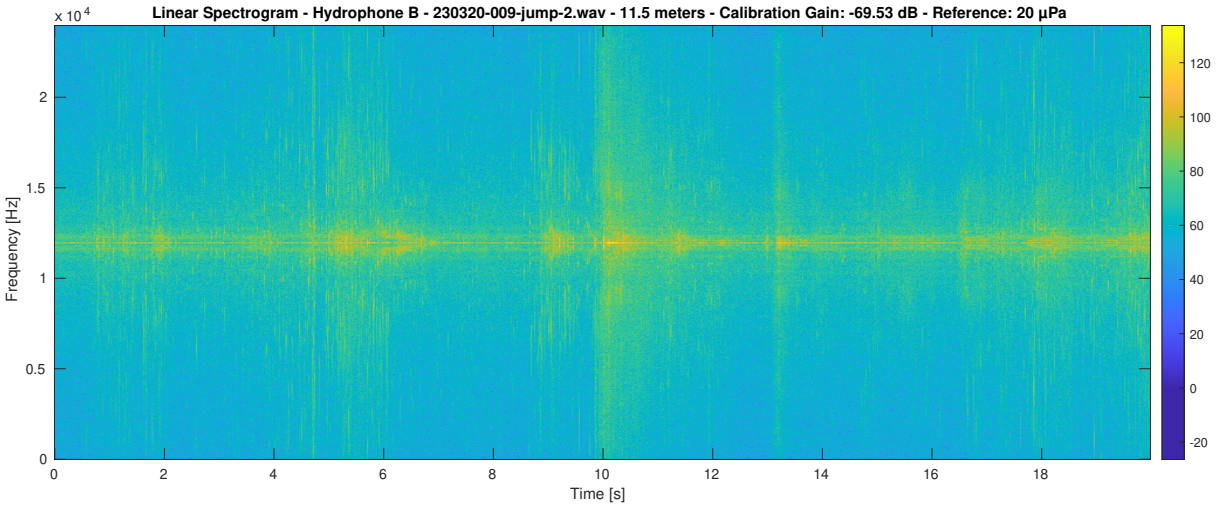


Figure 2.3. Spectrogram of a person jumping into water measured by a hydrophone placed 11.5 metres away from the impact [4]. The time of impact starts at 10 seconds.

The splash event is distinguishable in the spectrogram for one meter, but at 11.5 meters the background noise is too significant to easily depict the splash from the spectrogram. Therefore, somehow increasing the intensity of the signal is desired in order to increase the SNR and thereby extend the range of the system.

The audio recordings were sampled at 48 kHz, which means the range of the spectrogram is up to 24 kHz. However, it appears the splash generates frequencies across the entire spectrogram, meaning the sample rate of the system should be at least 48 kHz.

2.2 Subaqueous Acoustics

The acoustical properties in water vary greatly depending on the salinity, temperature, and depth. Therefore, an analysis of underwater acoustical propagation is conducted.

2.2.1 Speed of Sound

The speed of sound in water is dependent on temperature, salinity, and pressure. Numerous equations exist for calculating the speed of sound in water, e.g. Mackenzie, Coppens, UNESCO, Del Grosso, and NPL [5]. The range of validity differs for each of the equations and some are not applicable to freshwater. An equation that has a wide range of validity for temperature and salinity is Coppens' equation. The range of validity for Coppens is 0 to 35 °C, 0 to 45 ‰, and 0 to 4000 m depth. Coppens' equation is as follows [5].

$$c(0,S,t) = 1449.05 + 45.7t - 5.21t^2 + 0.23t^3 + (1.333 - 0.126t + 0.009t^2)(S-35) \tag{2.1}$$

$$\begin{aligned} c(z,S,t) = {} & c(0,S,t) + (16.23 + 0.253t)z + (0.213 - 0.1t)z^2 \\ & + [0.016 + 0.0002(S-35)](S-35)tz \end{aligned} \tag{2.2}$$

$c(z,S,t)$	Speed of sound at depth (z)	$[\frac{\text{m}}{\text{s}}]$
$c(0,S,t)$	Speed of sound at surface	$[\frac{\text{m}}{\text{s}}]$
z	Depth	[km]
S	Salinity of the water	[‰]
t	Water temperature $\cdot \frac{1}{10}$	[°C]

2.2.2 Absorption

The attenuation caused by absorption in the water is examined, as the absorption influences the feasible range of the system and the influence of noise. A simplified equation for calculating the attenuation factor is as follows [6]:

$$f_1 = 0.78 \left(\frac{S}{23}\right)^{1/2} e^{T/26} \tag{2.3}$$

$$f_2 = 42e^{T/17} \tag{2.4}$$

$$\alpha = 0.106 \frac{f_1 f}{f^2 + f_1^2} e^{(pH-8)/0.56} + 0.52 \left(1 + \frac{T}{43}\right) \left(\frac{S}{23}\right) \frac{f_2 f^2}{f^2 + f_2^2} e^{-z/6} + 0.00049 f^2 e^{-(T/27+z/17)} \tag{2.5}$$

f_1	Relaxation frequency for boron	[kHz]
S	Salinity of the water	[‰]
T	Water temperature	[°C]
f_2	Relaxation frequency for magnesium	[kHz]
α	Attenuation of signal pr. km	$[\frac{\text{dB}}{\text{km}}]$
f	Frequency of signal	[kHz]
pH	Potential of hydrogen of the water	$[\cdot]$
z	Depth	[km]

The attenuation caused by the inverse square law is as seen in Eq. (2.6), or more commonly known as -6 dB per doubling in distance.

$$I = \frac{P}{4\pi r^2} \tag{2.6}$$

I	Sound intensity	$[\frac{\mathrm{W}}{\mathrm{m}^2}]$
P	total power radiated from a point source	[W]
r	Radius of radiation sphere	[m]

2.2.3 Acoustic Impedance

The acoustic impedance is crucial for the design or choice of transducer, as the transducer must be impedance-matched with the surrounding water to optimise energy transfer. The acoustic impedance is approximative through the medium's density and the speed of sound.

$$Z = \rho c \tag{2.7}$$

Z	Acoustic impedance of medium	$[\frac{\mathrm{kg}}{\mathrm{m}^2\mathrm{s}}]$
ρ	Density of medium	$[\frac{\mathrm{kg}}{\mathrm{m}^3}]$

2.3 Power Considerations

Public bathing areas usually lack access to electricity near the water, so in order to power a surveillance solution, an alternative power source must be found such as solar, wind energy, a battery, or a combination. Both solar and wind will provide some amount of energy some amount of time each day, so a battery is needed nonetheless. Additionally, having sufficient storage at the location might be impractical, so the system should have some way to offload the data wirelessly.

In order to size a battery for use without a charging source Eq. (2.8) can be used, based on power draw and wished battery run time.

$$B = M \cdot P_d \cdot T_r \tag{2.8}$$

P_d	Power draw	[W]
M	Safety factor	$[\cdot]$
B	Battery size	[Wh]
T_r	Battery run time	[h]

Several strategies exist for sizing a solar panel or a wind turbine, whereof the one suitable for a device that must never die, is called the battery strategy as seen on Eq. (2.9) [7].

$$P_{PV} = M \cdot \left(\frac{B}{T_c}\right) \tag{2.9}$$

P_W	Wattage of source	[W]
T_c	Charge time	[h]

The battery strategy determines the wattage of the solar panel or wind turbine, based on charge-time of the battery. So by specifying battery size, i.e. 1 days worth of power, and minimum sun/wind-hours on a worst-case day, a solar panel size or wind turbine size can be determined.

2.4 Problem Statement

How is a system designed and implemented to autonomously detect and record splashes at a bathing location on a low power budget and with cheap transducers with the intent to create a large dataset for training of algorithms?

2.5 Requirements

This section describes the desired functionality of the system in order to fulfil the problem statement and formalises the technical requirements of the system.

2.5.1 Functionality of System

The key functionalities of the system are defined and described in this subsection. Multiple solutions for implementing the functionalities might exist, but these are explored in the next chapter/design stage.

F.1 Detect the splash from a person jumping into the water. This functionality should require minimal power, as the system should be in this state until a splash is detected. The system should deploy some technique in an attempt to maximise the SNR in order to increase the system's confidence in the detection.

F.2 Record the submerged audio of the splash and some subsequent swimming and save it in memory. The recordings must contain the entire splash, meaning that the measurements must begin before the initial impact.

F.3 Offload the recorded data wirelessly to a server and clear the recordings from the system memory once the data has been transferred.

F.4 The system must function on a low power budget that can be maintained by a small battery package with a potential renewable power source.

As the functional requirements are not solution-specific, deriving specific technical requirements before further exploration is unfeasible, since specifics are yet to be uncovered. The technical requirements are extracted from this chapter when possible, but specifics for developed modules will be discussed in the design and implementation phase, when relevant decisions have been made.

2.5.2 Technical Requirements

The technical requirements are established on the basis of the preceding analysis and functional requirements.

Table 2.1. Register of technical requirements for the transducers.

ID	Description	Requirement	Traceability
T.1	Equal sensitivity within frequency range 0 Hz – 24 kHz	±3 dB	Section 2.1
T.3	Operational within salinity range	0 – 40 ‰	Chapter 2
T.4	Operational within temperature range	−2 – 30 °C	Chapter 2
T.5	Acoustic impedance of the transducer must be matched to the water's	$1525 \cdot 10^3 \ \frac{\text{kg}}{\text{m}^2\text{s}}$	Subsection 2.2.3

Table 2.2. Register of technical requirements for the processing system.

ID	Description	Requirement	Traceability
P.1	The sample rate of the audio recordings	48 kHz	T.1
P.2	The system must be able to enter some low-power/sleep mode	-	F.1

2.6 Case Specification

A candidate for such a bathing location in Aalborg is Vestre Fjordpark. A satellite view of Vestre Fjordpark is seen on Figure 2.4. The lido has two enclosed sections: one that is primarily used for water polo and swimming and one mainly for dives. The diving section is equipped with 1 and 3 meter springboards, a diving tower that has diving platforms at 5 and 10 meters, and a climbing wall. The surrounding edge of the sections is around 0.5 m high.

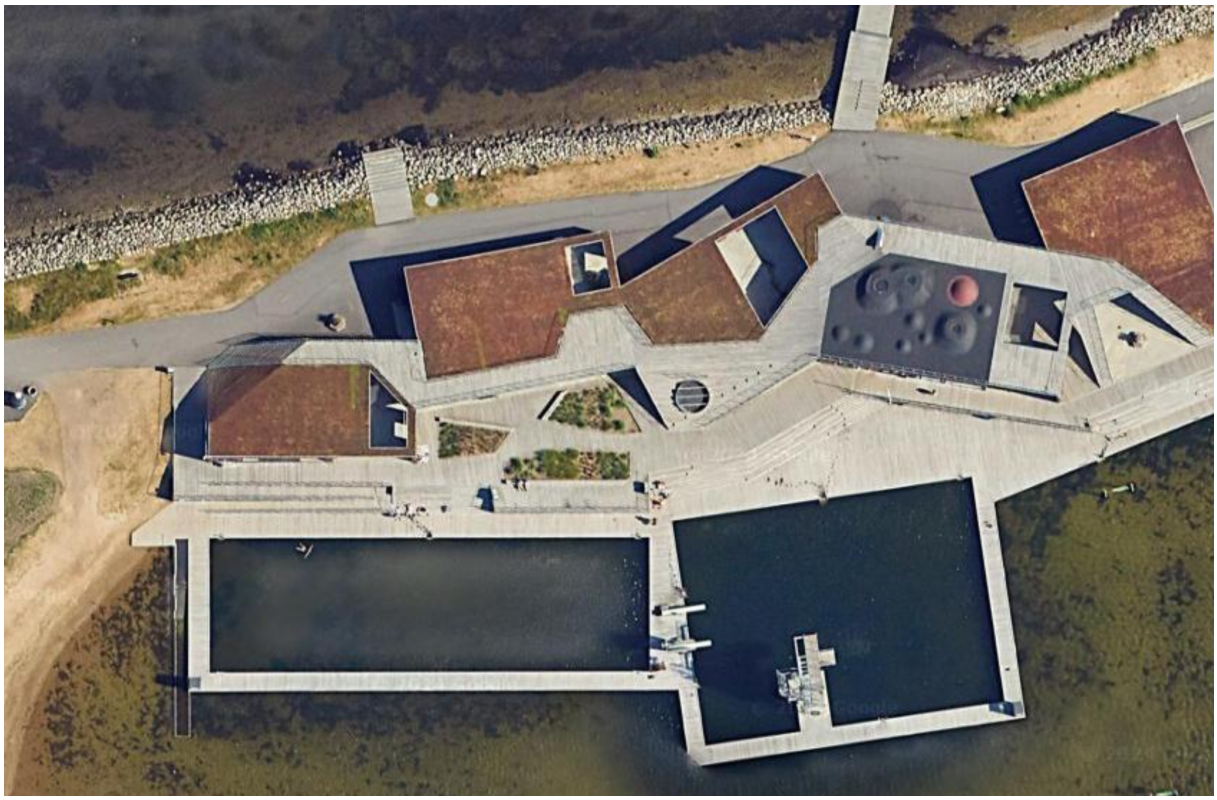


Figure 2.4. Satellite view of Vestre Fjordpark located in Aalborg [8]. The lido has two enclosed sections and an open area.

The eastmost section of the lido is the most obvious candidate for surveillance, as this section likely has the most amount of jumps and with sufficient variety due to the jumping platforms.

The water temperature in Vestre Fjordpark varies between 0 and 22 °C with an annual average of 10.2 °C [9]. The salinity of the water is around 22-30 ‰ [10], and the annual pH-balance is around 8.1 [11].

Using the average water temperature and salinity values from Vestre Fjordpark and a depth of 20 cm, the average theoretical speed of sound in the water is 1496 $\frac{\mathrm{m}}{\mathrm{s}}$ in Vestre Fjordpark, cf. Coppens' equation (Eq. (2.1)). At the water's lowest temperature and salinity, the speed of sound is 1448 $\frac{\mathrm{m}}{\mathrm{s}}$ and 1538 $\frac{\mathrm{m}}{\mathrm{s}}$ using the maximum values. Which means the possible variation of speed of sound in the water at Vestre Fjordpark is 89.6 $\frac{\mathrm{m}}{\mathrm{s}}$ which might influence the system's calculations.

The calculated attenuation caused by absorption in the water for a 10 kHz signal in Vestre Fjordpark is 0.9805 $\frac{\mathrm{dB}}{\mathrm{km}}$. Under similar conditions, the absorption in air is around 159 $\frac{\mathrm{dB}}{\mathrm{km}}$ [12]. The inverse square law applies for underwater signals in the same way as in air, but since the absorption in water is much less significant, the influence of noise sources will be more severe.

The density of the seawater of Vestre Fjordpark is approximately 1020 $\frac{\mathrm{kg}}{\mathrm{m}^3}$ using the temperature and salinity mentioned previously [13]. The acoustic impedance of the water in the lido is thus calculated to $1525 \cdot 10^3$ $\frac{\mathrm{kg}}{\mathrm{m}^2\mathrm{s}}$.

The transducer can be placed anywhere in the section, but it is undesired to have it directly underneath the diving platforms, as the power of the majority of the impacts might be too high.

A suiting placement of the transducer could be at the ladder, as it is centred at one of the sides and it is sufficiently distanced to the diving platforms. Furthermore, the least probable diving place of the section might be directly in front of the ladder, which means eventual near-field complications of the transducer are suppressed. The suggested placement of the transducer together with distance markings are illustrated in Figure 2.5.

Figure 2.5. Visualisation of dimensions in one of the two enclosed jumping sections at Vestre Fjordpark and the potential transducer placement at the ladder of the section [8].

With the presented requirements and case specification, the project will be deemed successful in the case where these are complied with and the problem assessment is fulfilled to satisfaction.

Design Specification 3

This chapter presents proposals to methods and hardware selections for fulfilling the functional requirements listed in Section 2.5. The most suitable candidates for each functionality are chosen at the end of the chapter.

3.1 Splash Recognition

The aim is not to create a perfect recogniser, as the goal of the project is to provide data to create a perfect recogniser. So, the trigger to record should be sensitive, but not so sensitive that it triggers on every small wave, every passing boat, or activities of people on land close to the area of interest. The trigger can be based on any sensor that in any sense can be setup to react on a person falling into water. This section will explore the most promising trigger types, which can be split into three categories i.e. **Object Detection**, **Basic Pattern Detection** and **Spectral Density**.

3.1.1 Object Detection

An example of an object detector is the infrared sensor. Infrared sensors could be used to monitor the surface of the water and detect if something is about to break the surface, since people emit infrared light that can be detected. An alternative to infrared sensors could be time-of-flight sensors that are based on either ultrasound or light. These are however more vulnerable to triggers on inanimate objects that do not emit heat.

3.1.2 Basic Pattern Detection

Another solution is to analyse specific patterns that might exist in a splash, such as the timing of the following splashes caused by the water that is shot into the air or similar. However, this approach resembles machine learning and requires access to a variety of data in order to find general patterns in the data.

3.1.3 Spectral Density

Another option is to record using omnidirectional hydrophones and base the trigger mechanism directly on the recordings. This would result in a more simple physical system, with a more complex processing solution. This could be done by inspecting the energy in specific frequency bands by use of the PSD, the FFT, or by filtering the signal into larger frequency bins and finding the power of those frequency ranges.

From Figure 2.2 it is seen that a splash has the same frequency content as an impulse, as it is of almost the same power across the recorded spectrum. Since either filtered bins, the PSD,

or FFT would be used to look for a broad spectrum signal, it could be vulnerable to noise, as noise sources from different angles might combine to a broad spectrum signal depending on the size and amount of bins/areas of inspection. This problem could however be mitigated by use of beamforming. An example of this is seen on Figure 3.1.

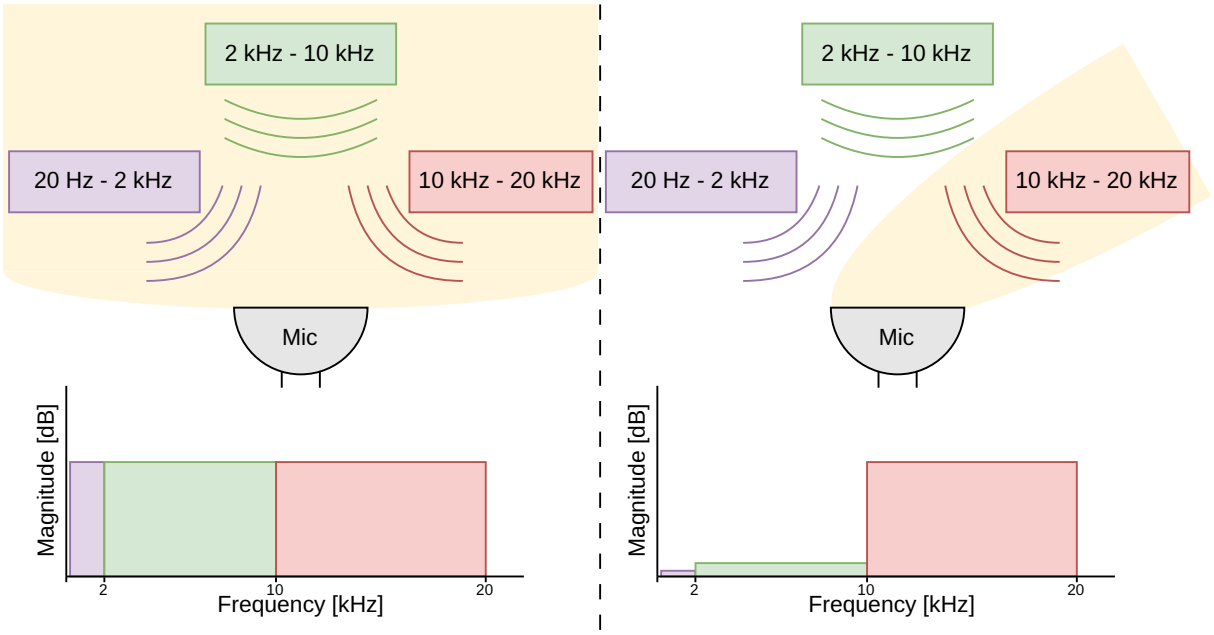


Figure 3.1. Visualisation of the influence of beamforming on the frequency spectrum. On the left an omnidirectional microphone is recording everything, resulting in no way of discerning acoustic sources. On the right a beamformer is steering in the direction of the 10 kHz to 20 kHz source, which makes it stand out as significantly more powerful in the frequency spectrum.

On the left, it is seen that all of the acoustic sources combine to a flat frequency response with the omnidirectional hydrophone, wheras the beamformer on the right makes sure the frequency spectrum primarily consists of what happens in the steered direction.

3.1.4 Summary

Basic Pattern Detection is likely a too complex solution for use as a trigger, while both **Object Detection** and **Spectral Density** are good candidates for a trigger solution. The trade-off primarily lies in the complexity of the physical system versus complexity of the digital system.

Since better SNR would likely result in a better recogniser, beamforming would already be beneficial for such a system, regardless of trigger mechanism. Therefore, using beamforming for eliminating the possibility of false broad spectrum signals generated by multiple noise sources, by focusing in a direction thought to have a splash event would be an efficient solution.

3.2 Beamforming

Beamforming functions through the principle of narrowing or steering the directivity pattern by combining the outputs of multiple microphones in an array configuration. An omnidirectional microphone is equally sensitive to all directions as seen in the directivity pattern illustrated in

Figure 3.2. As a consequence, the recorded power level of some signal is independent of its incident angle.

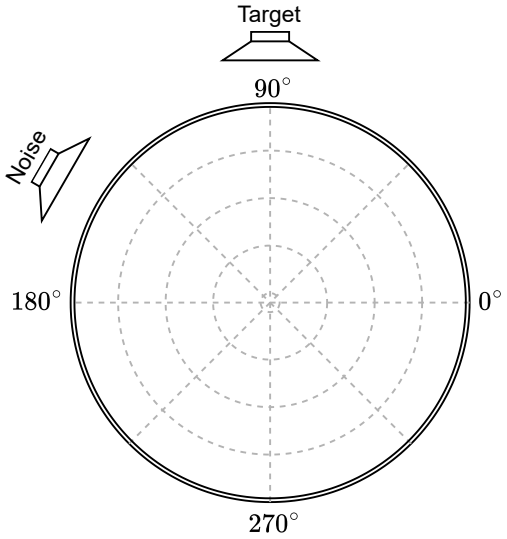


Figure 3.2. Illustration of the directivity pattern of an omnidirectional microphone. The *target* and *noise* sources originate from separate directions but are measured equally.

However, if the target and noise sources have separate angles of incident to the microphone array, the beam can be advantageously steered towards the *target* by combining the outputs of multiple microphones. An illustration of the desired directivity pattern is seen in Figure 3.3.

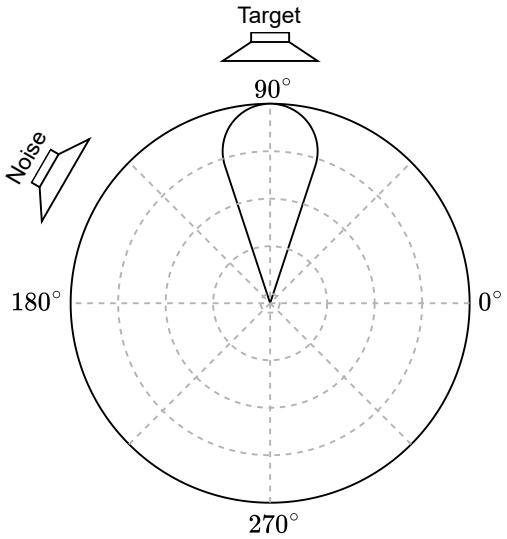


Figure 3.3. Illustration of the desired directivity pattern where a beam is focused towards the *target*'s angle of incidence and thereby increasing the SNR.

The microphones can be arranged in any configuration to generate the desired beam pattern, but uniform linear arrays (ULA) are usually used as the basis of the development. An illustration of a planar wavefront impacting a ULA is seen in Figure 3.4.

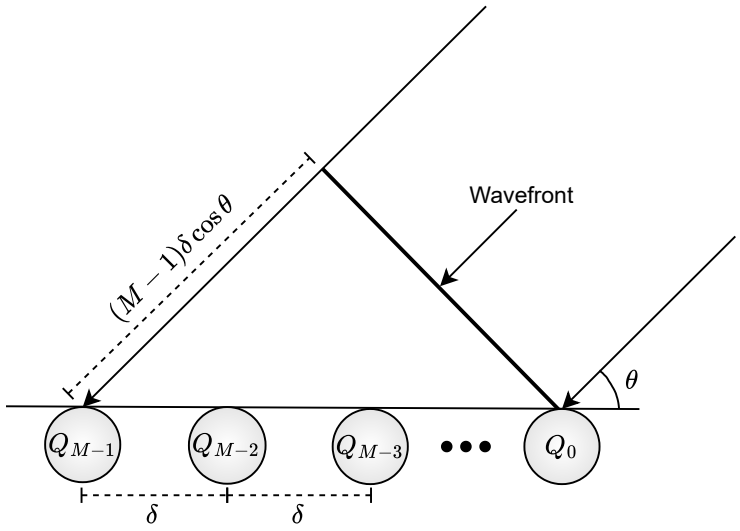


Figure 3.4. Illustration of M microphones in a ULA being impacted by a planar wavefront. The angle of incidence of the wavefront is θ and the microphones (Q) are equally spaced with a distance of δ.

The corresponding phase vector of the ULA in Figure 3.4 is defined as follows [14].

$$\bar{d}(\omega, \cos\theta) \triangleq \left[1 \; e^{\frac{-j\omega\delta\cos\theta}{c}} \; \cdots \; e^{\frac{-j(M-1)\omega\delta\cos\theta}{c}}\right]^{\mathrm{T}} \tag{3.1}$$

$\bar{d}$	Phase vector of length M	$[\cdot]$
ω	Angular frequency ($= 2\pi f$)	$[\frac{\mathrm{rad}}{\mathrm{s}}]$
c	Speed of sound in medium	$[\frac{\mathrm{m}}{\mathrm{s}}]$
f	Temporal frequency ($f > 0$)	[Hz]
δ	Microphone spacing	[m]
j	Imaginary unit	$[\cdot]$
θ	Incidence angle	$[^\circ]$
M	Total number of microphones	$[\cdot]$
T	Transpose operator	$[\cdot]$

Arranging the microphones in a ULA points the beam towards the broadside of the array-axis. The phenomenon is illustrated in Figure 3.5, where the output of three microphones ($M = 3$) in a ULA are summed for incident angles of 135°, 90° (broadside), and 45°. The wavefronts are assumed planar.

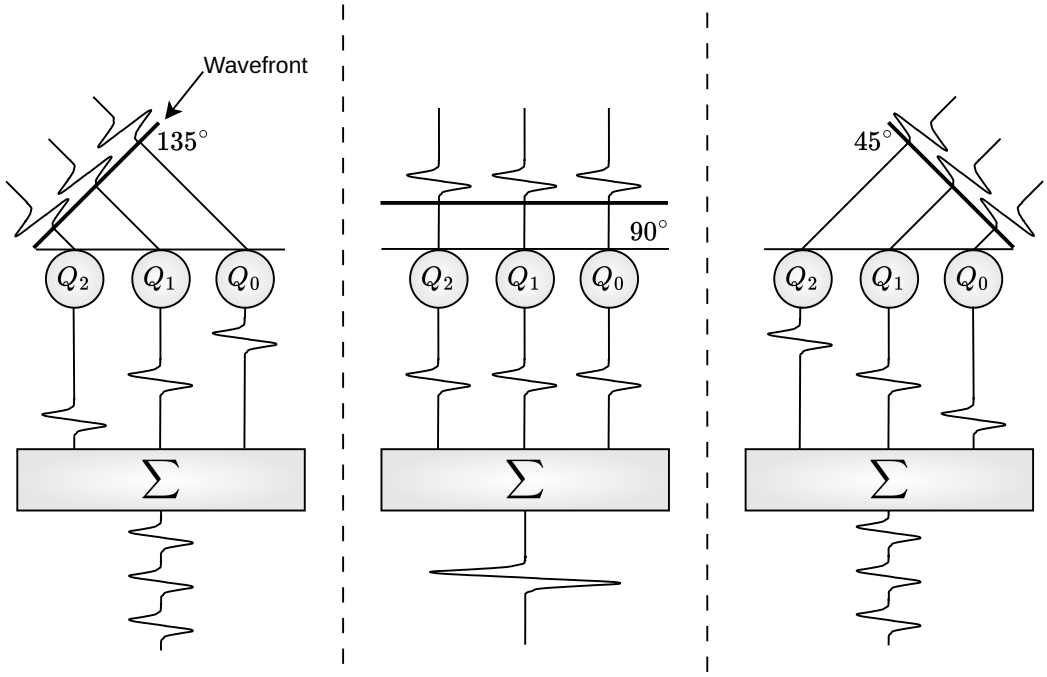


Figure 3.5. Illustration of the summed output of an $M = 3$ ULA stimulated with a planar wavefront from 135°, 90° (broadside), and 45°. The signal from the broadside hits the three microphones simultaneously and the signal is thereby amplified.

The output gain of the microphone array as a function of the signal's frequency and angle of incidence is calculated as follows.

$$BP_{das}(\omega, \cos\theta) = 20\log_{10}\left(\frac{1}{M}\sum_{m=0}^{M-1} d_m(\omega, \cos\theta))\right) \tag{3.2}$$

As stated earlier, the main lobe of the array is always at the array's broadside. However, it is desirable to be able to steer the main lobe if the *target* is non-static. Steerability can be achieved by individually delaying the output of each microphone. This technique is known as delay-and-sum (DAS) beamforming and is illustrated in Figure 3.6. The illustration is the same setup as Figure 3.5 but with added delays to steer the main lobe to 135°.

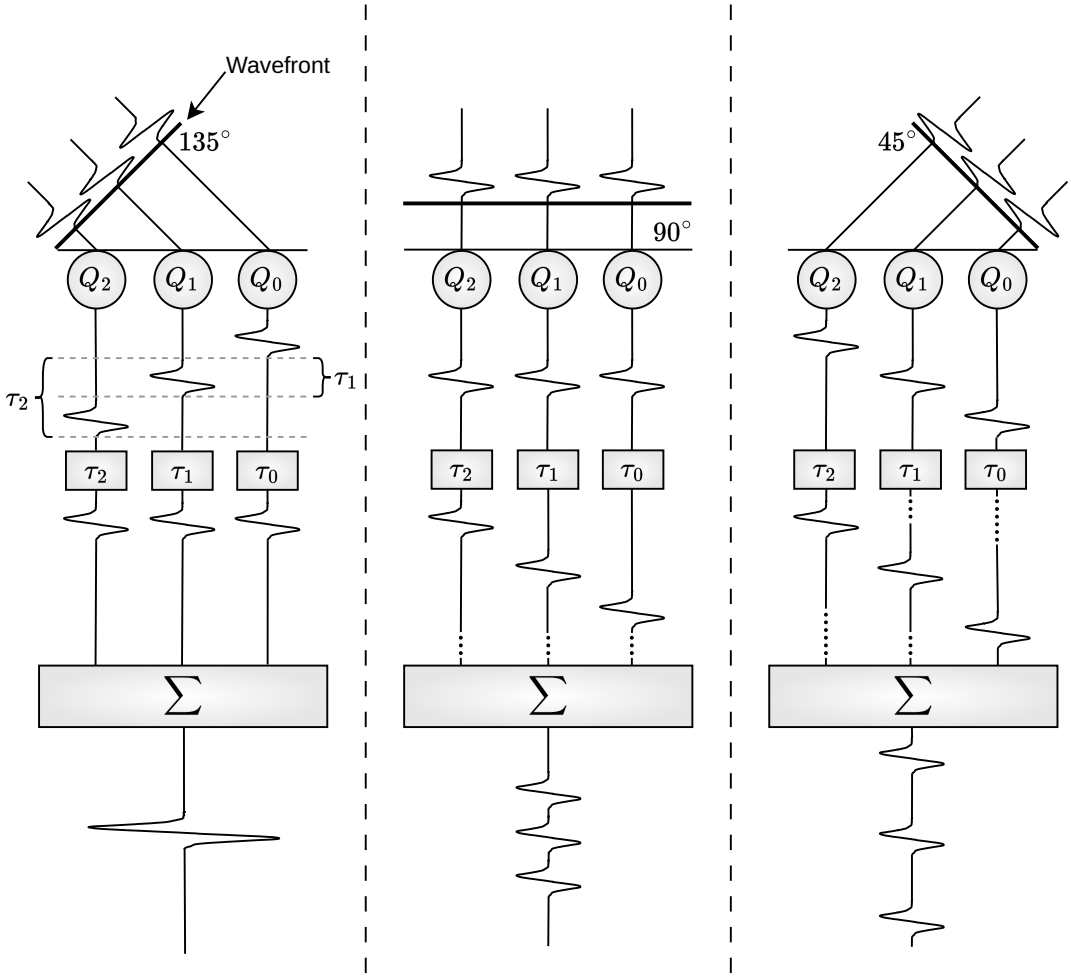


Figure 3.6. Illustration of delay-and-sum (DAS) beamforming for an $M = 3$ ULA stimulated with a planar wavefront from 135°, 90° (broadside), and 45°. The output of the microphones are delayed and summed in order to move the main lobe from 90° to 135°.

The delays are decided by the desired steering angle.The pattern is symmetric about the array's endfire direction so the delays for the microphone outputs of DAS beamforming are calculated as follows.

$$\tau_m = \frac{m\delta \cos\theta_s}{c}, \qquad \text{where } m = 0, 1, 2, \ldots, M \tag{3.3}$$

τ_m	Delay for the output of the mth microphone	[s]
θ_s	Steering angle of the beamformer	[°]

Simulation of DAS

The beampattern for the $M = 3$ ULA from Figure 3.5 with a microphone spacing of $\delta = 10$ cm and with the speed of sound in water at $c = 1496$ $\frac{\text{m}}{\text{s}}$ is seen in Figure 3.7. The directivity gain is calculated with Eq. (3.2). The upper plot shows the beampattern at 1 kHz, and the lower plot shows the broadband beampattern. As the beampatterns are symmetrical about the array's axis, the steering angle in the 2D plot is denoted between 0-180°, and between 180-360° for the 3D beampattern. They are, however, identical if flipped due to the symmetry.

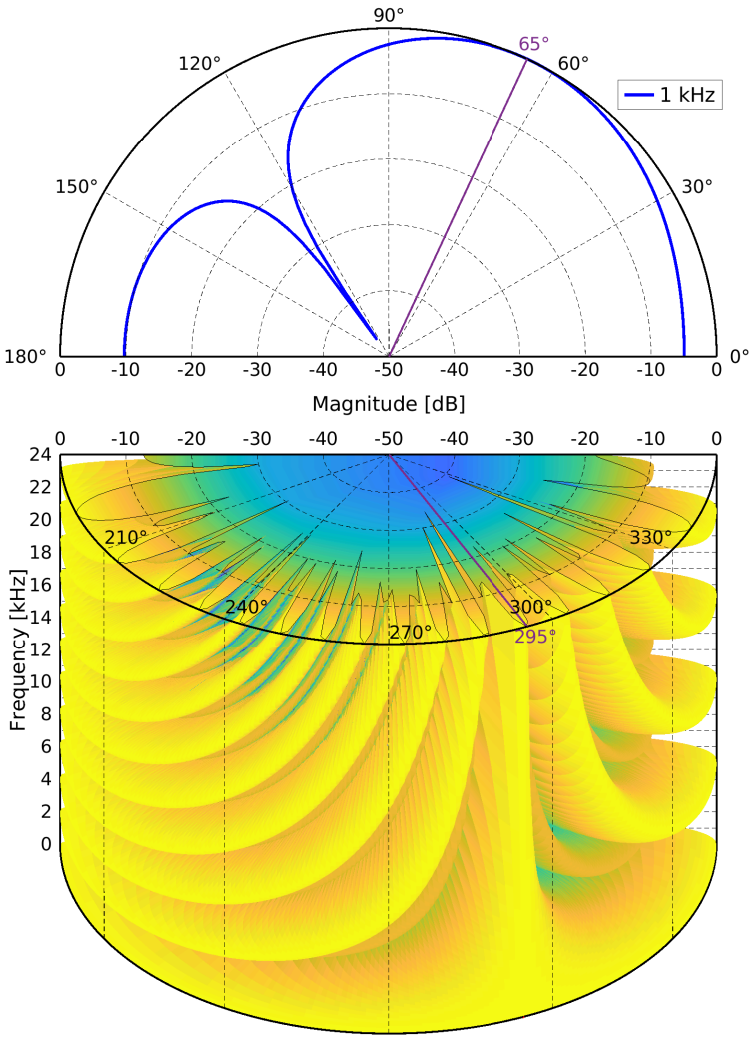


Figure 3.7. Simulated directivity pattern for an $M = 3$ ULA for 1 kHz and 20 Hz − 24 kHz. The beam is steered to 65/295°.

The simulated directivity pattern highlights one of the shortcomings of the DAS technique: the directivity is significantly frequency reliant. The specifications of the simulated array were chosen arbitrarily, but optimisations to the array can reduce the frequency dependency by increasing the number of microphones in the array and adjusting the spacing [15]. However, the technique is inherently frequency-dependent.

The DAS technique is categorised as additive, but another branch of beamforming is differential that is less frequency dependent [16].

3.2.1 Differential Beamforming

Differential beamforming functions on the principle of approximating the spatial derivative of the acoustic field across the array through the measured difference between the microphones [14]. Opposite to additive beamforming, where the main lobe is at the broadside of the array (90°), the main lobe of a differential beamformer is at the endfire (0°), as it functions on the derivative of the field.

Differential beamformers are hard to steer, but the paper *Steering Study of Linear Differential Microphone Arrays* presents a method for designing steerable linear differential microphone arrays (SLDMAs) [16]. The outline of SLDMAs is based on a scenario similar to Figure 3.4: a ULA being impacted by planar wavefronts. The phase vector of the microphone array is still as follows.

$$\bar{d}(\omega, \cos\theta) \triangleq \left[1 \; e^{\frac{-j\omega\delta\cos\theta}{c}} \; \cdots \; e^{\frac{-j(M-1)\omega\delta\cos\theta}{c}}\right]^{\mathrm{T}} \tag{3.4}$$

In order for the acoustic pressure differentials to be approximative by the microphones, the microphone spacing must be much smaller than the smallest wavelength in the signal: $\delta \ll \frac{c}{f_{max}}$. The observation signal vector is defined as follows.

$$\begin{aligned} \bar{y}(\omega) &\triangleq [Y_1(\omega) \; Y_2(\omega) \; \cdots \; Y_M(\omega)]^{\mathrm{T}} \\ &= \bar{d}(\omega, \theta_s)X(\omega) + \bar{v}(\omega) \end{aligned} \tag{3.5}$$

$\bar{y}$	Observation signal vector of length M	$[\cdot]$
X	Zero-mean source signal	$[\cdot]$
θ_s	Incidence angle of source signal	$[^\circ]$
$\bar{v}$	Zero-mean noise signal vector of length M	$[\cdot]$

The objective is to estimate the source signal ($X(\omega)$) by applying a spatial filter ($\bar{h}(\omega)$) to the observation signal vector ($\bar{y}(\omega)$).

$$Z(\omega) = \bar{h}^{\mathrm{H}}(\omega)\bar{y}(\omega) \tag{3.6}$$

The spatial filter is designed by establishing a frequency-independent ideal function that describes the response to a given incidence angle [14]. The number of coefficients decides the order of the beamformer.

$$\mathcal{B}_N(\theta) = \sum_{n=0}^{N} a_{N,n}\cos^n\theta \tag{3.7}$$

$\mathcal{B}_N$	Ideal beamforming function of order N	$[\cdot]$
$a_{N,n}$	Real coefficients	$[\cdot]$

The beamformer should comply with a distortionless constraint, meaning the gain should be 1 for the steering angle.

$$\mathcal{B}_N(\theta_s) = \bar{h}^{\mathrm{H}}(\omega)\bar{d}(\omega, \theta_s) = 1 \tag{3.8}$$

The steering angle and the N nulls of the beam pattern are assembled in a vector: $\bar{x} = [x_s \; x_1 \; \cdots \; x_N]^{\mathrm{T}}$, where $x_s = \cos\theta_s$, $x_n = \cos\theta_n$, and $n = 1, 2, ..., N$. From the distortionless constraint, the following linear system of equations can be formed.

$$\bar{D}(\omega, \bar{x})\bar{h}(\omega) = \bar{\mathrm{i}}_1, \qquad \text{where } \bar{\mathrm{i}}_1 = [1\ 0\ \cdots\ 0]^{\mathrm{T}} \tag{3.9}$$

However, for the beamformer to be steerable, the order of the beamformer must be greater than 1 and the coefficients of the ideal function must comply with the following constraint [16]:

$$\sum_{n=1}^{N} \frac{a_{N,n}}{a_{N,N}} n x_s^{n-1} = 0 \tag{3.10}$$

The nulls from x_1 to x_{N-1} are freely chosen except that they have to be distinct. The final null (x_N) has to be calculated in order to satisfy the constraint. If the nulls have multiplicity, a different system of equations should be used [16]. A list of the nulls as indeterminates of the ideal function is formed.

$$\bar{q}_N(x) = \left[1\ x^1\ \cdots\ x^N\right]^{\mathrm{T}}, \qquad \text{where } x = \{x_s, x_1, x_2, ..., x_N\} \tag{3.11}$$

The coefficients of the ideal function can then be found through the following linear system of equations [14]:

$$\bar{Q}(\bar{x})a_N = \bar{\mathrm{i}}_1 \tag{3.12}$$

Where the Q-matrix is given as shown in Eq. (3.13). The second entry in the matrix comes from setting the derivative equal to 0 at x_s.

$$\bar{Q}(\bar{x}) = \begin{bmatrix} \bar{q}_N^{\mathrm{T}}(x_s) \\ \bar{q}_N^{\mathrm{T}}(x_s)\bar{\Sigma}_N \\ \bar{q}_N^{\mathrm{T}}(x_1) \\ \vdots \\ \bar{q}_N^{\mathrm{T}}(x_{N-1}) \end{bmatrix}, \qquad \text{where } \bar{\Sigma}_N = \mathrm{diag}\,(0, 1, 2, ..., N) \tag{3.13}$$

Solving Eq. (3.12) for a_N yields the coefficients of the ideal function.

$$a_N = \bar{Q}^{-1}(x)\bar{\mathrm{i}}_1 \tag{3.14}$$

The final null (x_N) is then determined from the two last coefficients and the other nulls as follows [16].

$$x_N = -\frac{a_{N,N-1}}{a_{N,N}} - \sum_{n=1}^{N-1} x_n \tag{3.15}$$

With the last null determined, the filter weights can be calculated with Eq. (3.16) or (3.17) depending on if $M = N + 1$ or $M > N + 1$, respectively [16].

$$\bar{h}(\omega) = \bar{D}^{-1}(\omega, \bar{x})\bar{\mathrm{i}}_1 \tag{3.16}$$

$$\bar{h}(\omega) = \bar{D}^{\mathrm{H}}(\omega, \bar{x}) \left[\bar{D}(\omega, \bar{x})\bar{D}^{\mathrm{H}}(\omega, \bar{x})\right]^{-1} \bar{\mathrm{i}}_1 \tag{3.17}$$

The coefficients for the ideal function are invariant of ω, but the weights should be recalculated with Eq. (3.16) or (3.17) for each frequency of interest.

Three performance measures are used to evaluate the performance of the beamformer: beampattern, directivity factor (DF), and white noise gain (WNG) [16]. The beampattern is the beamformer's sensitivity to a given incident angle (similar to Figure 3.7) and is defined as follows.

$$\mathcal{B}_{\theta_s}[\bar{h}(\omega), \theta] \triangleq \bar{h}^{\mathrm{H}}(\omega)\bar{d}(\omega, \cos\theta) \tag{3.18}$$

The DF describes the ratio of the gain of the main lobe to the accumulated gain of the whole pattern, or in other words: how focused the beampattern is. The DF is defined as follows.

$$\mathcal{D}[\bar{h}(\omega)] \triangleq \frac{|\mathcal{B}_{\theta_s}[\bar{h}(\omega), \theta_s]|^2}{\frac{1}{2}\int_0^{\pi} |\mathcal{B}_{\theta_s}[\bar{h}(\omega), \theta]|^2 \sin\theta \mathrm{d}\theta} \tag{3.19}$$

The WNG describes the beamformer's amplification of white noise, e.g. noise from the microphone array's imperfections. The WNG is defined as follows.

$$\mathcal{W}[\bar{h}(\omega)] \triangleq \frac{|\bar{h}^{\mathrm{H}}(\omega)\bar{d}(\omega, \cos\theta_s)|^2}{\bar{h}^{\mathrm{H}}(\omega)\bar{h}(\omega)} \tag{3.20}$$

The DF can be increased by increasing the order of the beamformer (N), and the WNG can be improved by increasing the number of microphones (M) [16].

Simulation of Differential

The simulated beampattern for 1 kHz, broadband beam pattern, WNG, and DF for a second-order SLDMA with three microphones are seen in Figure 3.8. The configurations used to generate the beamformer and the patterns are listed in Table 3.1. The weights are recalculated for every 100 Hz from 20 Hz to 24 kHz with Eq. (3.17) in the *DMA.mlx* script (Appendix A).

Table 3.1. Parameters of the beamformer in Figure 3.8. The microphone spacing (δ) is set to one-tenth of the biggest wavelength.

N	M	x_s	x_1	x_N	c	δ
2	3	$\cos(65°)$	$\cos(155°)$	1.75	1496 $\frac{\mathrm{m}}{\mathrm{s}}$	0.62 cm

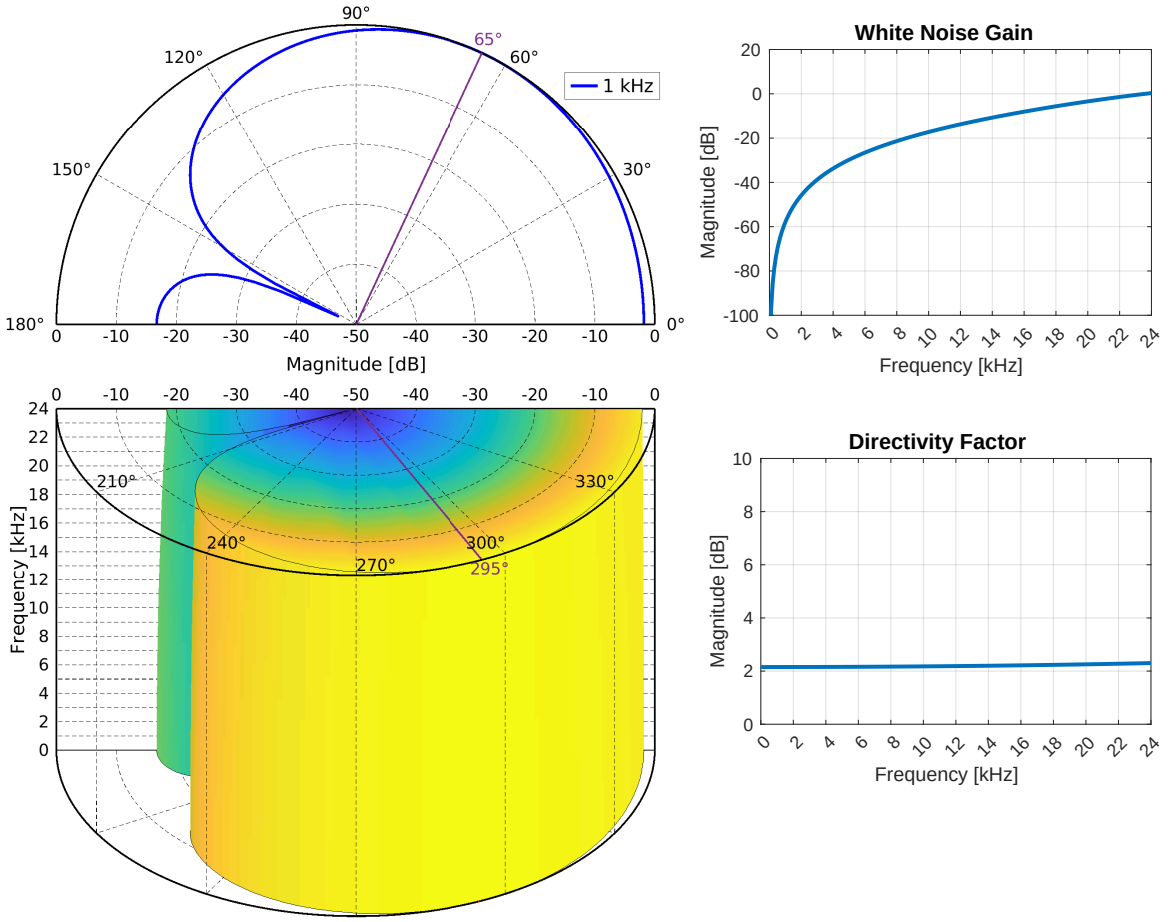


Figure 3.8. Simulated beampattern at 1 kHz and broadband and corresponding performance measures for a second-order SLDMA. The beamformer and beampattern were constructed using the specifications listed in Table 3.1. The WNG is calculated with Eq. (3.20) and the DF is calculated with Eq. (3.19) using the trapezoidal rule for numerical integration.

The beampattern clearly shows a null in 155/205° and a maximum at 65/295° as expected. The fact that 0° is not a maximum also indicates that the steering of the main lobe was successful. The WNG is not great for the lower frequencies, but the DF is constant which proves the frequency-invariance. However, the beam pattern is not particularly focused towards the steering angle, as the gain is almost 1 in the entire first quadrant. Adding more nulls, and thereby increasing the order of the beamformer, should increase the DF.

The simulation of a fourth-order SLDMA with five microphones is shown in Figure 3.9. The configurations used in the simulation are listed in Table 3.2.

Table 3.2. Parameters of the beamformer in Figure 3.9. The microphone spacing (δ) is set to one-tenth of the biggest wavelength.

N	M	x_s	x_1	x_2	x_3	x_N	c	δ
4	5	$\cos(65°)$	$\cos(22°)$	$\cos(110°)$	$\cos(155°)$	8.31	1496 $\frac{\text{m}}{\text{s}}$	0.62 cm

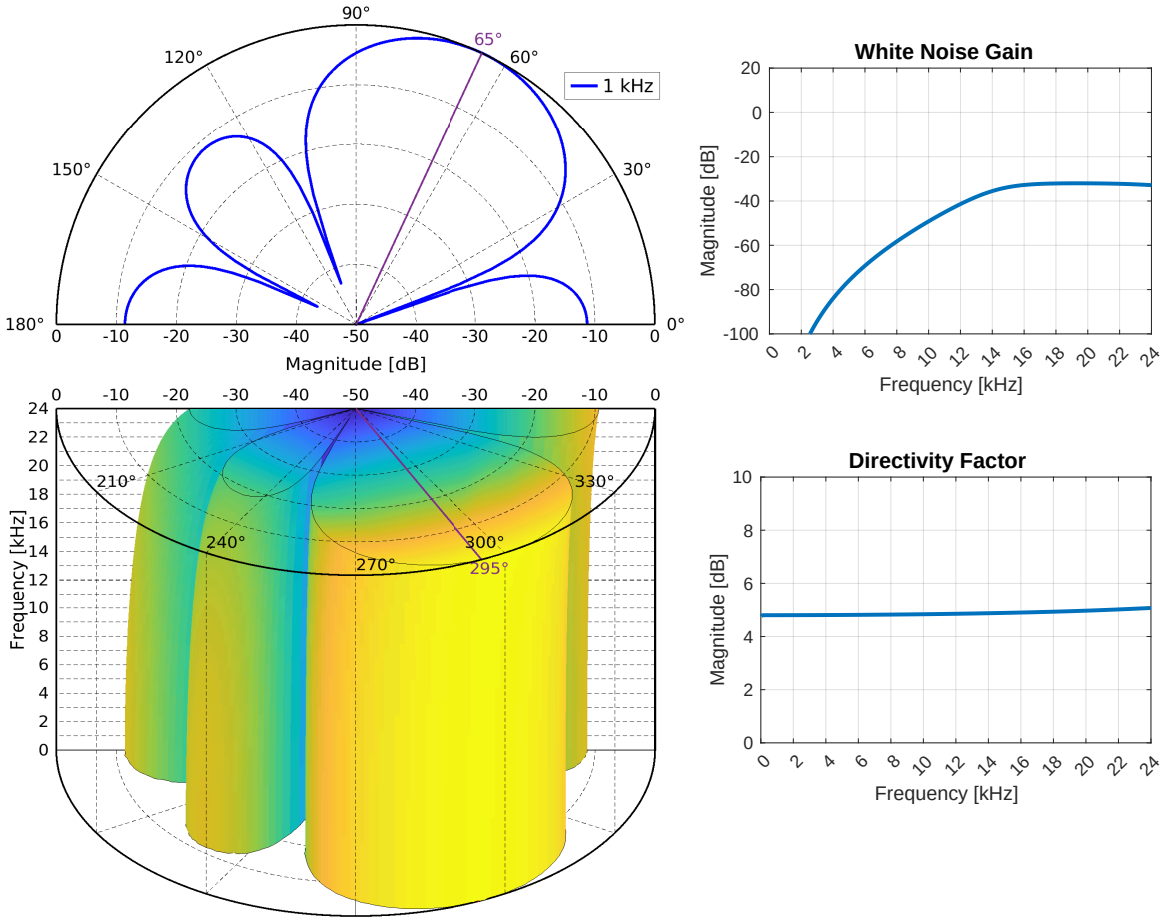


Figure 3.9. Simulated beam pattern at 1 kHz and broadband and corresponding performance measures for a fourth-order SLDMA. The beamformer and beam pattern were constructed using the specifications listed in Table 3.2. The WNG is calculated with Eq. (3.20) and the DF is calculated with Eq. (3.19) using the trapezoidal rule for numerical integration.

The DF of the beampattern has more than doubled from Figure 3.8, but the DF is no longer constant as it shows a slight increase in the higher frequencies. Inappropriately, the WNG has worsened. Adding more microphones to the beamformer should increase the WNG, as it helps dampen the imperfections of each microphone.

The beampattern at 18 kHz and broadband for a second-order SLDMA with 15 microphones together with the WNG and DF for the beamformer with a range of microphones between three and 15 are shown in Figure 3.10.

Table 3.3. Parameters of the beamformer in Figure 3.10. The number of microphones is a list between three and 15 with a spacing of two. The microphone spacing (δ) is set to one-tenth of the biggest wavelength.

N	M	x_s	x_1	x_N	c	δ
2	$(3:2:\mathbf{15})$	$\cos(65°)$	$\cos(155°)$	1.75	1496 $\frac{\text{m}}{\text{s}}$	0.62 cm

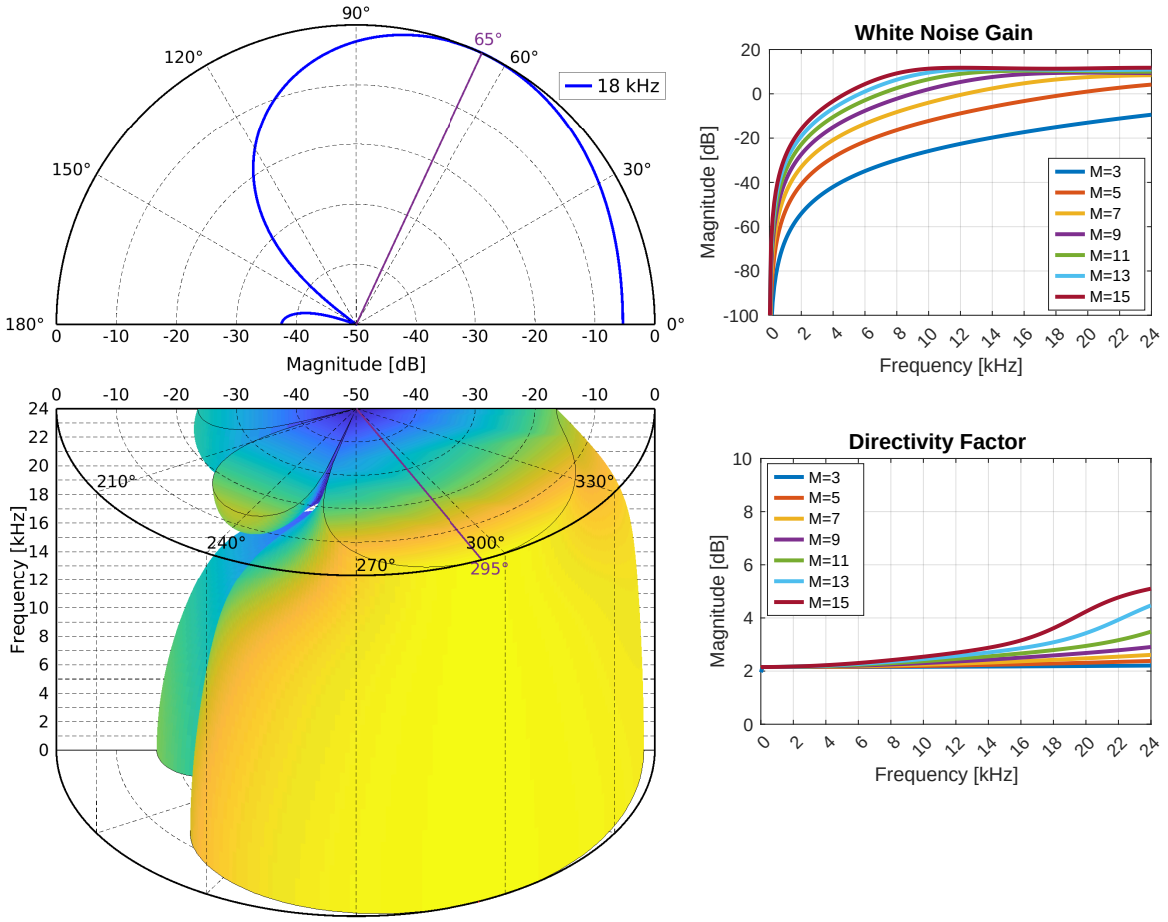


Figure 3.10. Simulated beam pattern at 18 kHz and broadband for a fourth-order SLDMA with 15 microphones. The corresponding performance measures are shown for a range of three and 15 microphones. The beamformer and beam pattern were constructed using the specifications listed in Table 3.3. The WNG is calculated with Eq. (3.20) and the DF is calculated with Eq. (3.19) using the trapezoidal rule for numerical integration.

The simulated beam pattern with 15 microphones is not frequency-invariant. Even though the beamformer is second-order, more nulls appear at the higher frequencies. This is due to Eq. (3.17) being the minimum-norm solution. However, the WNG graph clearly shows that adding more microphones increases the WNG as expected.

Thus, the design of SLDMAs primarily consists of finding a trade-off between DF and WNG by adjusting the order of the beamformer and number of microphones.

3.2.2 Summary of Beamforming

Two techniques for beamforming were presented: **DAS** and **SLDMA**. The former has the advantage of bigger possible microphone spacing, as the spacing in **SLDMA** must be much smaller than the smallest wavelength in the incoming signal. However, **SLDMA** has the property of a frequency-invariant beampattern which is appealing if the direction of arrival for noise can be determined such a null can be placed in that direction.

DAS functions through delays, meaning older samples must be stored in the system and the delays are fractional which complicates the implementation. The number of filter weights that has to be stored in the system for **SLDMA** depends on the frequency resolution and a balance between WNG and DF has to be found.

3.3 Direction of Arrival Estimation

The steering angle must be set in order to utilise beamforming. Multiple methods exist for estimating the direction of arrival (DOA) of a signal, and some of the most obvious candidates are presented here.

3.3.1 Time Difference of Arrival

A common method for determining the DOA of a signal is to calculate the time difference of arrival (TDOA) of the signal to the array. The time difference can be determined through peak detection, circular cross-correlation, or least squares. The DOA can then be estimated through the geometry of the array and the time difference.

3.3.2 Sensor System

The **Object Detection** method to detect a splash in Section 3.1 could be expanded upon such the sensor would not only alert of a splash but also estimate the DOA to the hydrophones. This could be implemented as a semicircle of infrared sensors or time-of-flight sensors.

3.3.3 Beamforming

It is also a possibility to utilise beamforming to estimate the DOA by rapidly switching the steering angle across the field of view and determining the direction with the highest power. However, this method requires the calculations to be done quickly in order to evaluate the same signal for all steering directions.

3.3.4 Summary of Direction of Arrival

Three methods are proposed for estimating the DOA of an incoming signal to a microphone array. **TDOA** requires no extra hardware and the complexity of the calculations can be adjusted through choice of algorithm. Adding a **Sensor System** has the disadvantage of extra hardware and thereby complexity of the system. **Beamforming** to estimate the DOA is practical if the beamformer is already implemented but the computational complexity depends on the implemented beamformer.

3.4 Processing Platform

A processing platform which fulfils the demands from Section 2.5 must be found. Specifically, the demands related to the processing platform are wireless offloading capabilities (F.3), low-power capabilities (F.4, P.2), temperature resistance (T.4), and I/O that allows for 48 kHz sampling (P.1). A system overview is seen on Figure 3.11.

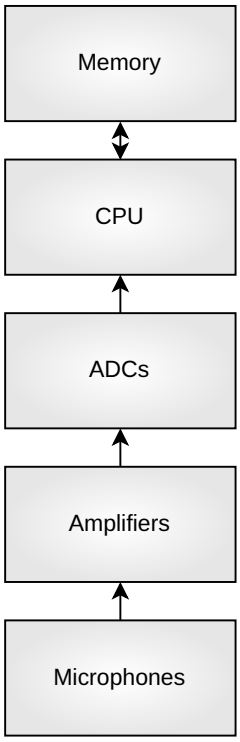


Figure 3.11. Initial system overview of a microphone sampling system with a CPU.

Since the scope of the project is to autonomously detect and record splashes on a low power budget, it is chosen to examine the well-known ESP32 series.

3.4.1 ESP32

The ESP32 is a popular series of microcontrollers from Espressif. They are compatible with the Arduino platform and PlatformIO for simple usage, but also comes with the Espressif IoT Development Framework (ESP-IDF), which is a mature software platform that includes a toolchain, a build system, debugging tools, and more [17]. For this project the ESP-IDF will be used, enabling full system control and eliminating the overhead needed for Arduino or PlatformIO. Espressif is a big contributor to the Arduino Project, which is a great indication of their dedication to usability. This, together with the low-power usage is a good balance between rapid development and complying with the demands from Section 2.5. The latest models are the ESP32-S3 and the ESP32-C3, where the C3 is an ultra-low-power single-core RISC-V microcontroller and the S3 is simply a low-power dual-core Xtensa LX7 microcontroller. Since the ultra-low-power RISC-V microcontroller might not be powerful enough for the use-case, the ESP32-S3 is selected without further investigation. For simplicity while developing, a development board will be used, specifically the ESP32-S3-DevKitM-1 as seen on Figure 3.12.

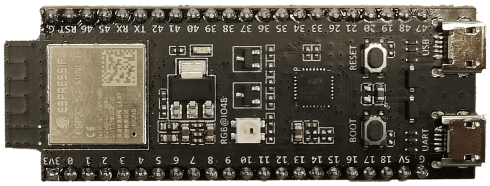


Figure 3.12. Image of the chosen ESP32-S3 development board.

For sampling, custom microphone sampling hardware with buffers is needed. Constant sampling and buffering is needed, such that a low-power trigger on the Ultra-Low-Power (ULP) coprocessor on the ESP32-S3 can wake the main processor, and continue calculations without loss of the original trigger data. Such a system is seen on Figure 3.13.

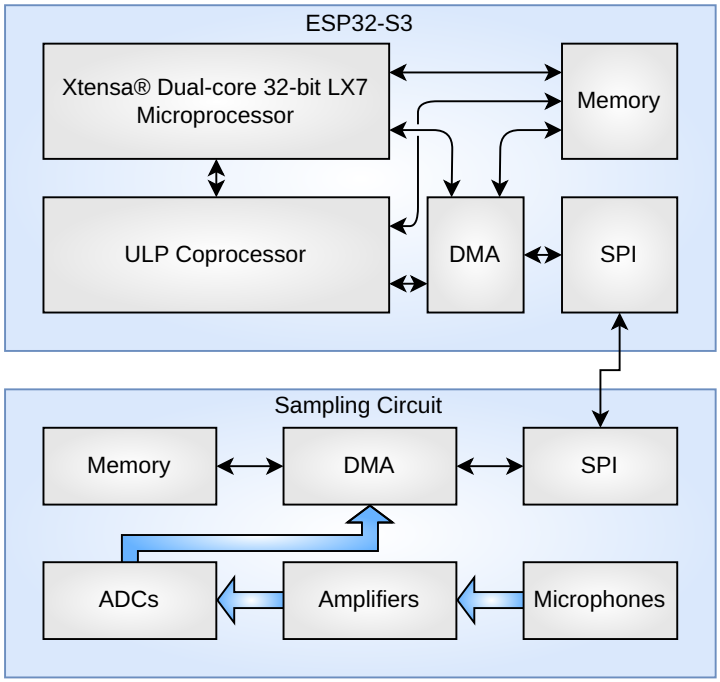


Figure 3.13. Proposed system overview utilising an ESP32-S3-DevKitM-1 development board as the compute unit.

The development of an ultra-low-power microphone amplification and sampling circuit is deemed out of scope, so the configurable hardware platform PSoC is chosen semi-arbitrarily instead. Using an FPGA is deemed overkill and too expensive for the use-case, so it was chosen to use a configurable hardware platform i.e. the PSoC 5LP. Specifically a CY8CKIT-059, which is a PSoC 5LP prototyping kit as seen on Figure 3.14.

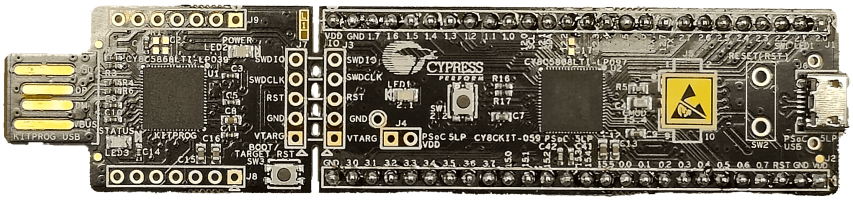


Figure 3.14. Image of the chosen PSoC 5LP development board.

3.4.2 PSoC 5LP

Programmable system on a chip, or PSoC for short, is a family of microcontroller integrated circuits from Infineon (Previously Cypress). The PSoC 5LP is equipped with 24 Universal Design Blocks (UDB), an Arm processor and a high-performance DMA controller which allows for rapid development of hardware systems. Infinion have pre-verified over 150 peripheral components for use with the UDBs, such that adding functionality such as I2S is done simply by dragging and dropping a block in their tool PSoC Creator and configuring it in C [18]. This will allow for

rapid prototyping of a microphone amplification and sampling circuit with buffering, whereas the low-power aspect is deemed out of scope for this block. As the PSoC 5LP only has 2 ADCs, it is chosen to use I2S Microphones with built-in sampling. Such a system is seen on Figure 3.15.

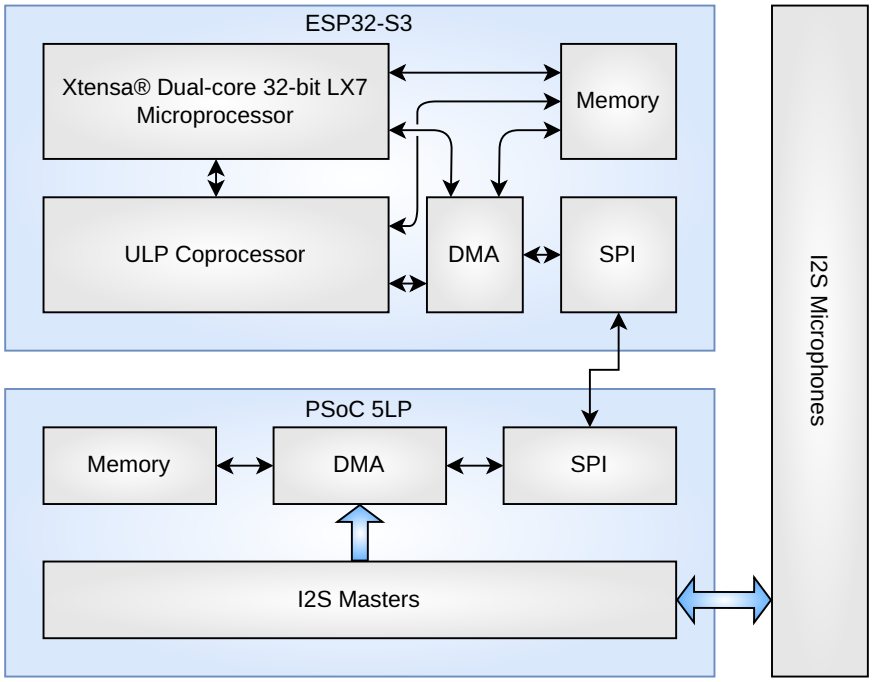


Figure 3.15. Proposed system overview utilising an ESP32-S3-DevKitM-1 development board as the compute unit and a PSoC 5LP prototyping kit with I2S microphones as the sampling circuit.

In order to find an estimate of how many microphones the PSoC supports, the available pins are counted. The PSoC has 43 pins usable for SPI and I2S. SPI needs at least three pins for one-way communication, leaving 40 pins [19]. The I2S blocks can share a clock, leaving 39 pins left for data and word select resulting in a theoretical limit of 19 I2S blocks. An I2S block uses about 8% of the UDB resources [20], wheras a SPI block uses about 12% [21]. This results in a maximum of 10 I2S blocks when using one SPI block.

3.4.3 Hydrophone

In a previous project by Frederik Sidenius Dam [3] an investigation comparing calibrated reference hydrophones to inexpensive custom hydrophones was conducted, in the use-case of detecting drowning accidents. The project found that inexpensive custom hydrophones are highly capable, which supports the decision to use I2S MEMS microphones. Specifically, a DMM-4026-B-I2S-EB-R is chosen, as it comes pre-configured with amplification, sampling and I2S on a small evaluation board, as seen on Figure 3.16.

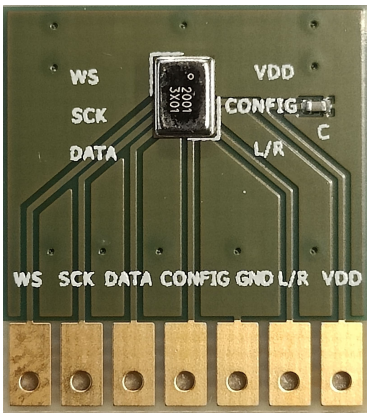


Figure 3.16. Image of the backside of the chosen transducer.

The specifications for the microphone are listed in Table 3.4. Since the word size is 32-bit i.e. 64-bit for stereo operation with two microphones, each microphone will need its own I2S block on the PSoC, as it supports a maximum of 32-bit words [20].

Table 3.4. Specifications of DMM-4026-B-I2S-EB-R MEMS microphone [22].

Parameter	Condition	Value	Unit
Directivity	Omnidirectional		
Data format	I^2S 24-bit data size with 18-bit precision, 32-bit word size		
Sensitivity	1 kHz @ 50cm with 94 dB source 0dB=1V/Pa	-26 ± 1	dB
Signal-to-Noise Ratio	1kHz, 94 dB input, A-weighted	64	dB
Frequency Range	20 to 20 000		Hz
Pass Band	Fs=48 kHz	18	kHz
Pass Band Attenuation	-	0.5	dB
Operating Temperature	-40 to +100		°C

The typical frequency response of the microphone is seen on Figure 3.17. The frequency response is measured 50 cm away from a 94 dB acoustic source and a consistency of ±1 dB from 20 to 20 000 Hz is guaranteed in the datasheet [22].

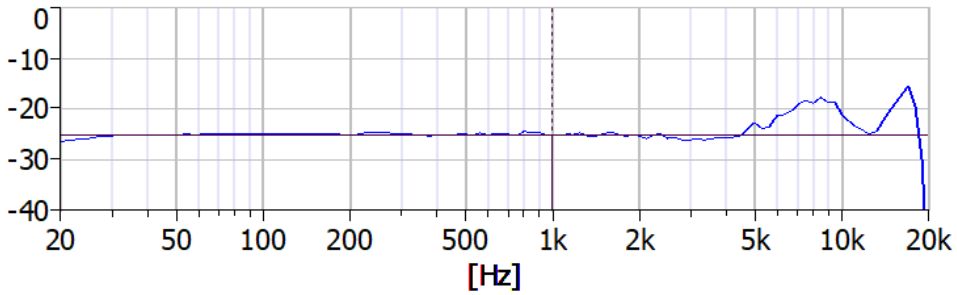


Figure 3.17. Typical frequency response of DMM-4026-B-I2S-EB-R when 50 cm from 94 dB acoustic source with a production consistency of ±1 dB from 20 to 20 000 Hz [22].

The technical requirements for the transducer are partly complied with. The temperature requirement c.f. T.4 is met. The impedance match with water c.f. T.5 and underwater operation in saline water c.f. T.3 is met, by inserting the microphones into a rubber glove as described by the previous project by Frederik Sidenius Dam [3]. The frequency range requirement c.f. T.1 is not met, since the frequency range is only 20 Hz to 20 kHz.

3.5 Design Composition

Selections of the methods/technologies proposed in this chapter are made in this section with the aim to compose a design that is in compliance with the requirements from Section 2.5.

It is decided to utilise **Spectral Density** to detect a splash in collaboration with **DAS** beamforming. By having multiple microphones in the system, the same splash can be recorded with variations and the amount of training data is increased. Beamforming is chosen as part of the splash detection, as it is an obvious addition with the introduction of more microphones. **DAS** beamforming is selected as it allows for greater microphone spacings than **SLDMA**. Furthermore, it is decided to estimate direction of arrival through the time difference of arrival of the signal to the microphones. The algorithm to determine the time difference is undecided here, as a choice might become more apparent in the implementation.

It is decided to use a microcontroller from the ESP32 series, as they are used extensively for many use-cases and are very inexpensive. The latest models are the ESP32-S3 and the ESP32-C3, where the C3 is an ultra-low-power single-core RISC-V microcontroller and the S3 is simply a low-power dual-core Xtensa LX7 microcontroller. Since the ultra-low-power RISC-V microcontroller might not be powerful enough for the use-case, the ESP32-S3 is selected without further investigation.

For the microphone amplification and sampling circuit, custom hardware is deemed out of scope. Instead, the configurable hardware platform PSoC is chosen, as using an FPGA is deemed overkill and too expensive for the use-case, so it was chosen to use a configurable hardware platform i.e. the PSoC 5LP instead.

The microphone (DMM-4026-B-I2S-EB-R) was chosen based on its small form factor, communication protocol, and price. Although the microphone does not comply with the technical requirement in Table T.1 (frequency sensitivity), it was the best option at the time.

It is decided to implement eight microphones in the system, as 10 are available to the project and it is desired to keep some as spares. The chosen hardware should be compatible with eight I2S microphones, as the maximum for the PSoC was estimated to 10 in Subsection 3.4.2. The microphones are placed on a plank with a maximum length of one metre, as anything larger than that is estimated to become too impractical to handle during development/prototyping.

Implementation 4

This chapter implements the chosen methods and technologies from Chapter 3 on the hardware selections presented in the same chapter with the goal of fulfilling the functional requirements. An illustration of the desired functionality from Chapter 2 is visualised on Figure 4.1, which this chapter will implement.

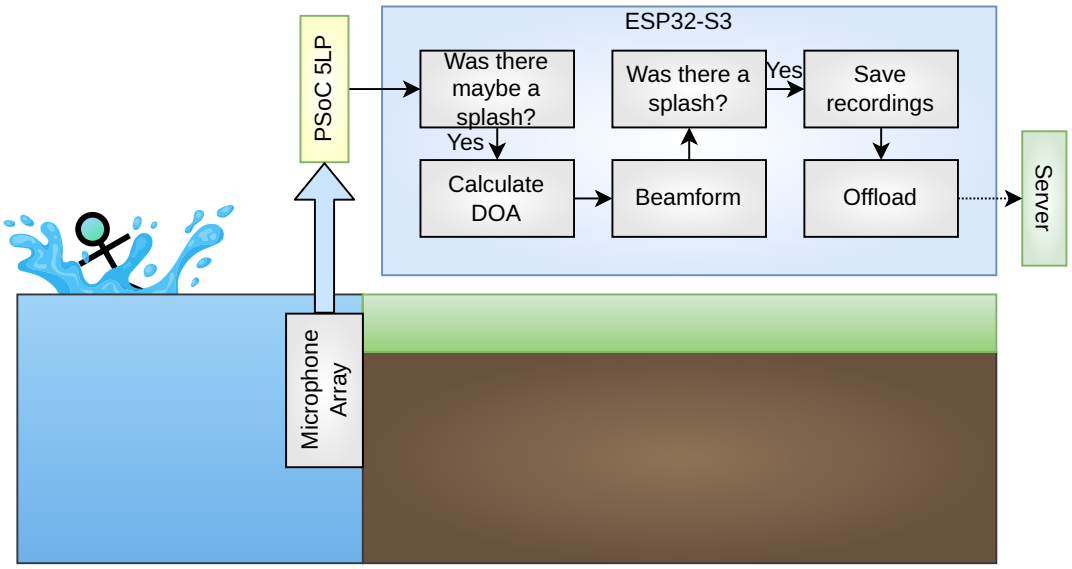


Figure 4.1. Visualisation of the functional demands from Chapter 2, where a person creates a splash, of which the system the system is detecting, recording, and offloading.

If the impact occurs too closely to the microphones, the assumption of planar wavefronts that is vital for the beamformer becomes more invalid, as the propagation becomes more spherical closer to the source. This could be accounted for by adding another splash-check that can bypass the beamformer, if, for instance, the signal power exceeds some very high threshold.

4.1 Design of Delay-and-Sum Beamformer

The purpose of this section is to decide the spacing of the microphones in order to implement a satisfactory DAS beamformer and investigate the effect of neglecting fractional delays. Firstly, a desired beampattern is derived by calculating the phase vector. Thereafter, the beampattern is simulated with a discrete input signal by comparing the power of the input signal to the output of the beamformer. Finally, if the influence of rounding off the fractional delays is too significant, the fractional delays will be implemented.

It was decided to utilise delay-and-sum (DAS) beamforming rather than differential, as DAS allowed for bigger microphone spacings. Since the objective is to gather training data from all microphones in the array, it is beneficial to have some distance between them in order to capture data with a higher variance.

The exposition of DAS beamforming is found in Section 3.2, but a summary of the principles and equations of DAS is included here. The uniform linear array (ULA) of microphones is impacted by planar wavefronts as seen in Figure 4.2, with the corresponding phase vector of length M (number of microphones) defined in Eq. (4.1).

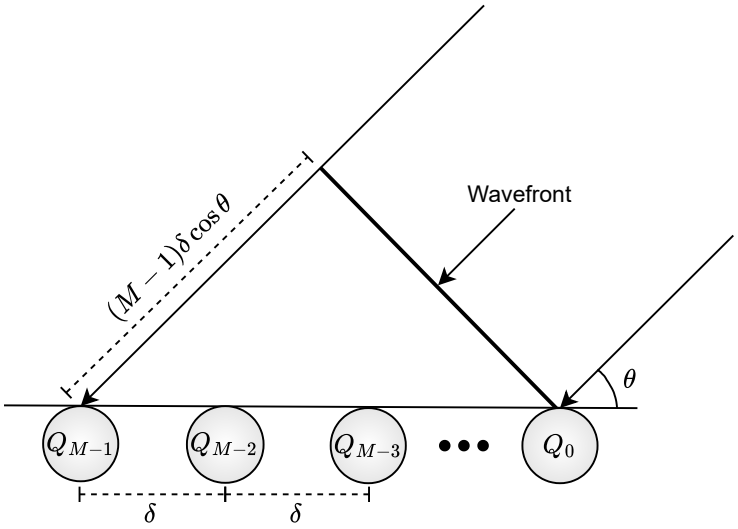


Figure 4.2. Illustration of M microphones in a ULA being impacted by a planar wavefront. The angle of incidence of the wavefront is θ and the microphones (Q) are equally spaced with a distance of δ.

$$\bar{d}(\omega, \cos\theta) \triangleq \left[1 \; e^{-j\omega\delta\cos\theta/c} \; \cdots \; e^{-j(M-1)\omega\delta\cos\theta/c}\right]^{\mathrm{T}} \tag{4.1}$$

$\bar{d}$	Phase vector of length M	$[\cdot]$
ω	Angular frequency ($= 2\pi f$)	$[\frac{\mathrm{rad}}{\mathrm{s}}]$
c	Speed of sound in medium	$[\frac{\mathrm{m}}{\mathrm{s}}]$
f	Temporal frequency ($f > 0$)	[Hz]
δ	Microphone spacing	[m]
j	Imaginary unit	$[\cdot]$
θ	Incidence angle	$[^\circ]$
M	Total number of microphones	$[\cdot]$
T	Transpose operator	$[\cdot]$

Delays are then added to the output of the individual microphones in order to steer the lobe of the DAS beamformer. The delays are calculated as follows based on the desired steering direction.

$$\tau_m = \frac{m\delta\cos\theta_s}{c}, \qquad \text{where } m = 0, 1, 2, \ldots, M-1 \tag{4.2}$$

τ_m	Delay for the output of the mth microphone	[s]
θ_s	Steering angle	[°]

With the addition of the delays, the phase vector for the ULA becomes as follows.

$$\bar{d}_{das}(\omega, \cos\theta) = \left[1 \; e^{-j\omega(\delta\cos\theta/c+\tau_m)} \; \cdots \; e^{-j\omega((M-1)\delta\cos\theta/c+\tau_m)}\right]^{\mathrm{T}} \tag{4.3}$$

The gain of the beamformer for a specific direction and frequency is then found by summing the delayed outputs of the microphones as seen in Eq. (4.4). By evaluating the equation for a range of incident angles and frequencies, the beampattern of the beamformer can be generated.

$$BP_{das}(\omega, \cos\theta) = 20\log_{10}\left(\frac{1}{M}\sum_{m=0}^{M-1} d_{\mathrm{das},m}(\omega, \cos\theta))\right) \tag{4.4}$$

4.1.1 Design Decisions

The beamformer's functionality is to suppress any signals arriving from incident angles other than the steering angle, as the idea for recognising splashes is to check the power spectrum of an incoming signal (cf. Section 3.1). This way, only signals from the steering direction will have a flat power spectrum. However, to allow some leeway for the angle of incidence determination, the distortionless requirement should apply to adjacent angles. The design philosophies for the beamformer are thus as follows.

- Distortionless in steering direction (and adjacent degrees) across frequency interval
- Varying distortion in side lobes across frequency interval

In Section 3.5 it was decided to constraint the maximum length of the microphone array to one metre and use eight microphones in the setup.

The directivity factors (DF) in this section are calculated with Eq. (3.19) using the trapezoidal rule for numerical integration, and a speed of sound of $c = 1496 \; \frac{\mathrm{m}}{\mathrm{s}}$.

DAS 1 (Single ULA)

The beampattern utilising all eight microphones uniformly spaced across one metre is seen in Figure 4.3. The figure shows the beampattern at 1 kHz and at broadband, the microphone placements, and the DF. The script for the simulation is found in *DAS8M.mlx* in Appendix A.

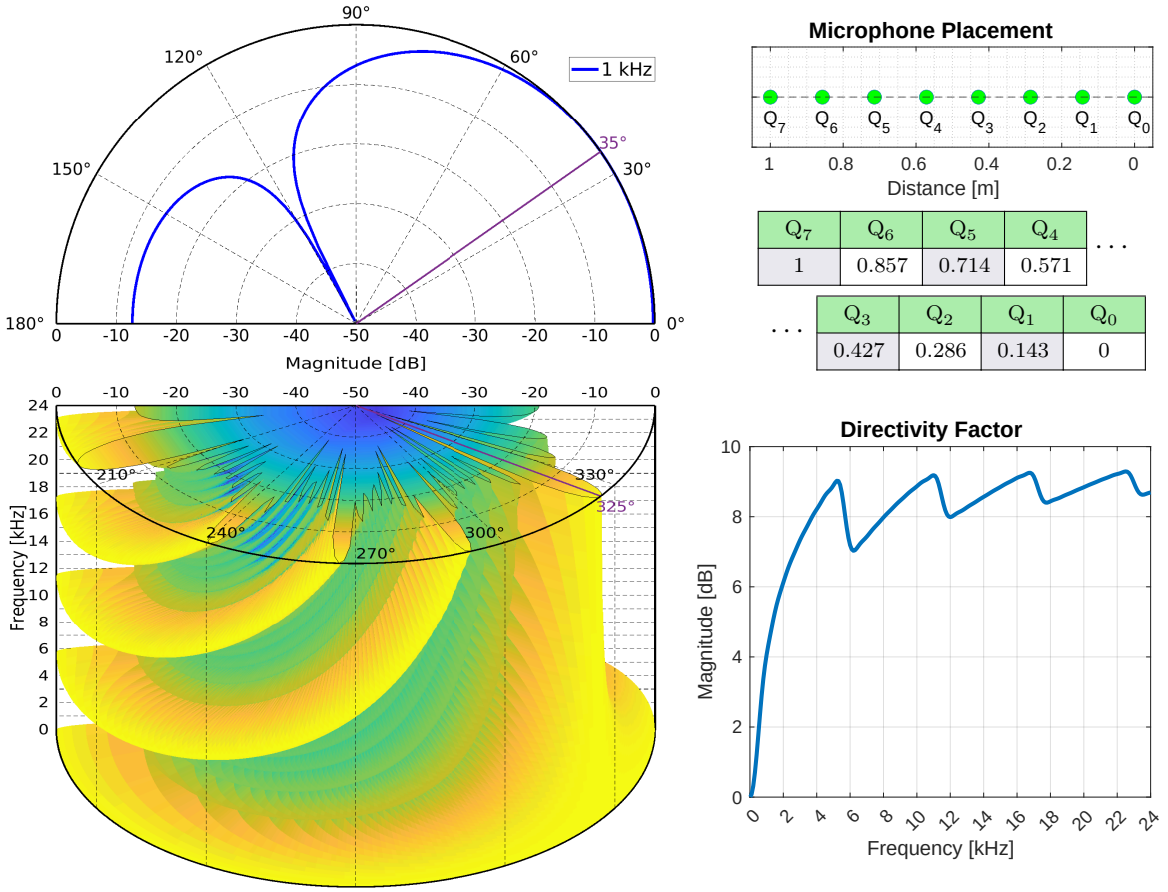


Figure 4.3. Simulated beampattern for an $M = 8$ delay-and-sum beamformer at 1 kHz and broadband, a list of the microphone placements, and directivity factor of the beampattern. The steering angle is set to 35/325°.

The angular width of the main lobe at the upper frequencies is too narrow, meaning slight inaccuracy in the angle of incidence determination results in the signal becoming distorted. The number of periods across the array for a 24 kHz signal is $\frac{1 \text{ m} \cdot 24 \text{ kHz}}{1496 \frac{\text{m}}{\text{s}}} = 16.05$, which explains the vast number of lobes in the upper frequencies.

However, decreasing the spacing would result in the DF worsening at the lower frequencies and the total length of the array becoming smaller. Therefore, rather than using all eight microphones in a single beamformer, the microphones are arranged such they comprise three ULAs each with three microphones, where each ULA is only active for a specific frequency region. The wavelength as a function of frequency in the signal interval is plotted in Figure 4.4 along with the chosen separation regions for the three ULAs. The frequency regions are chosen based on the slope of the function.

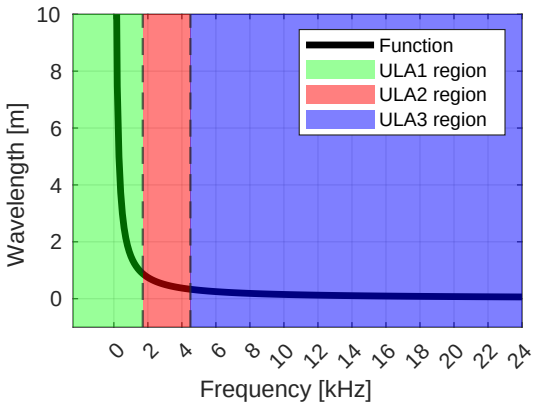


Figure 4.4. Wavelength as a function of frequency in the frequency interval and the chosen frequency regions for the three ULAs. The intervals are seen in Table 4.1.

Table 4.1. Frequency regions for the three ULAs plotted in Figure 4.4.

	ULA1	ULA2	ULA3
Frequency interval [Hz]	$[0, 1700)$	$[1700, 4500)$	$[4500, 24000]$
Avg. wavelength [m]	1.76	0.388	0.0907

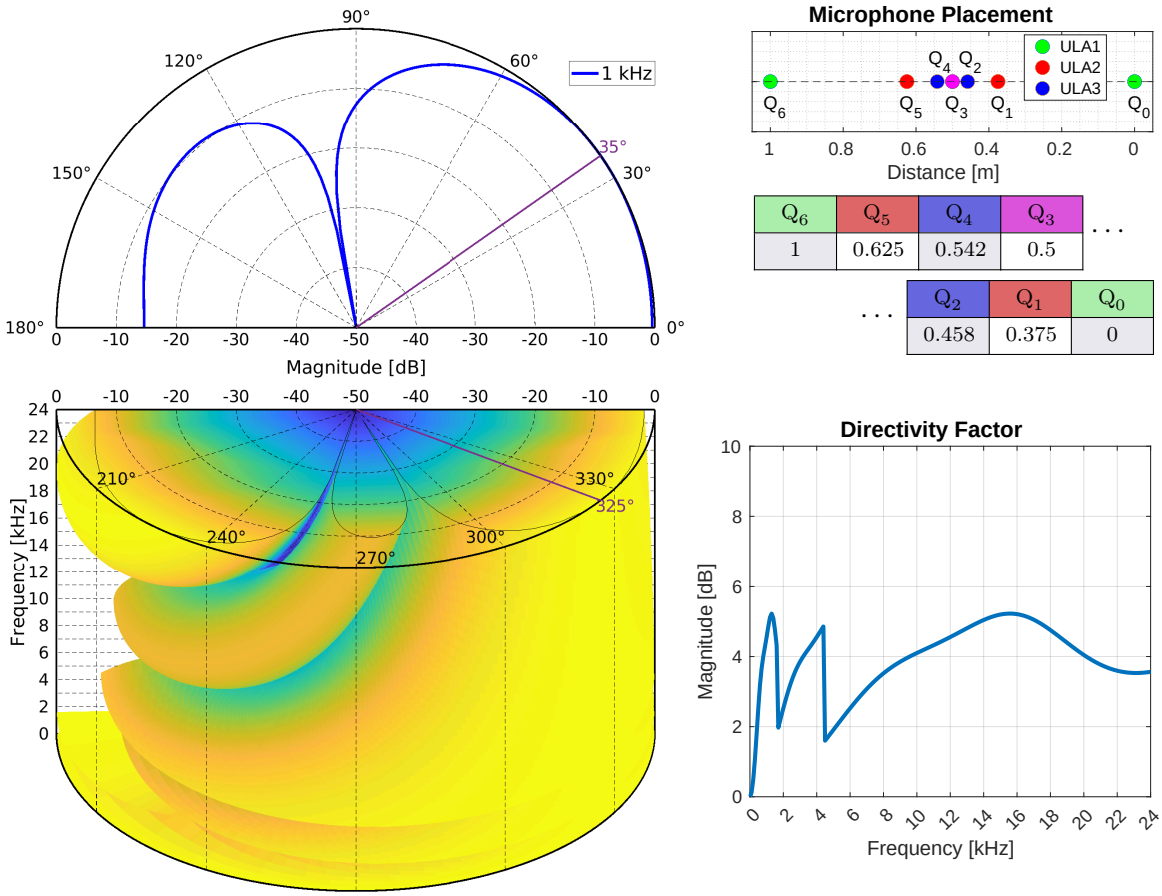
DAS 2 (Separate ULAs)

The number of microphones used in the beamformer is changed to seven, as it is more convenient to divide the ULAs. The array is arranged such the centre microphone is used in all three ULAs, and the remaining microphones are spaced about the centre microphone. The spacing for ULA1 is set to the maximum possible in the 1 metre array, whereas ULA2's spacing is set to a quarter of ULA1, as its avg. wavelength in the frequency region is roughly a quarter of ULA1. Following this procedure, ULA3's spacing is set to a quarter of ULA2's. The microphone spacings in the three ULAs for **DAS 2** are seen in Table 4.2.

Table 4.2. Spacings between the microphones for the three ULAs in the **DAS 2** beamformer. The spacings are denoted as fractions of the length of the microphone array.

δ_{ULA1}	δ_{ULA2}	δ_{ULA3}
$\frac{1}{2} \cdot 1$ m	$\frac{1}{8} \cdot 1$ m	$\frac{1}{24} \cdot 1$ m

The beampattern for **DAS 2** is seen in Figure 4.5 along with the microphone placements and DF. The active ULA is set by *if*-statements based on the frequency of the input signal, which results in sharp and sudden transitions between the ULA regions. Ideally, the inputs are filtered such the ULAs overlap, but the purpose of these simulations is primarily to see the effect of the individual ULAs. The script for the simulation is found in *DAS.mlx* Appendix A.



Q_6	Q_5	Q_4	Q_3	...
1	0.625	0.542	0.5	

...	Q_2	Q_1	Q_0
	0.458	0.375	0

Figure 4.5. Simulated combined beampattern of **DAS 2** consisting of three $M = 3$ delay-and-sum beamformers at 1 kHz and broadband, a list of the microphone placements, and directivity factor of the beampattern. The frequency regions of the ULAs are seen in Table 4.1. Q_3 is the centre microphone in all three ULAs, and the steering angle is set to 35/325°.

Simulations of **DAS 2** with lowered viewing angle and steering angles set to 90/270° and 140/220° are seen in Figure 4.6 and 4.7, respectively.

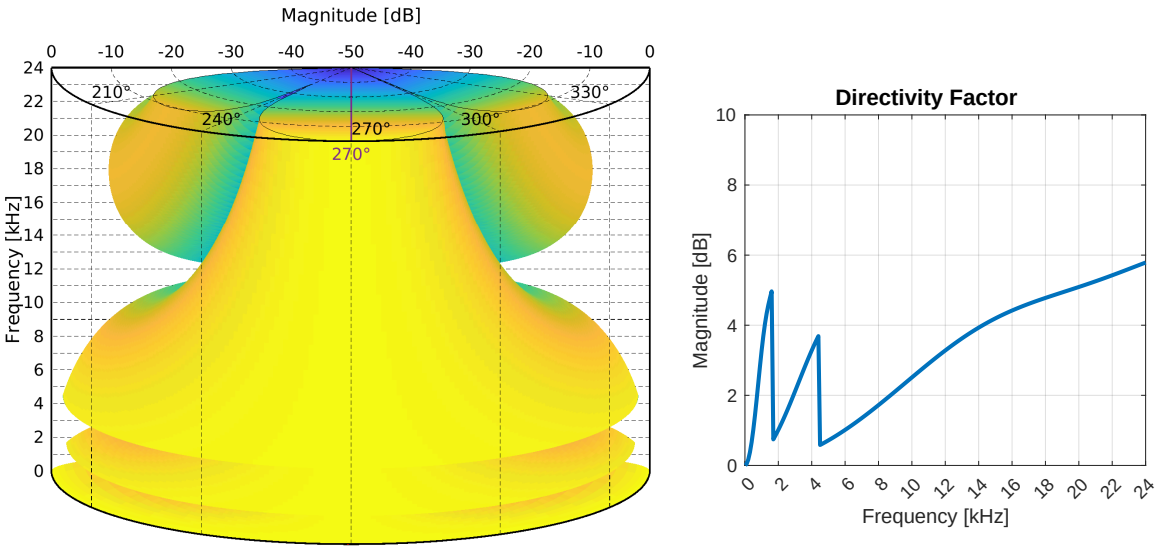


Figure 4.6. Simulated combined broadband beampattern of **DAS 2** (Figure 4.5) with the steering angle changed to 90/270° and lowered viewing angle.

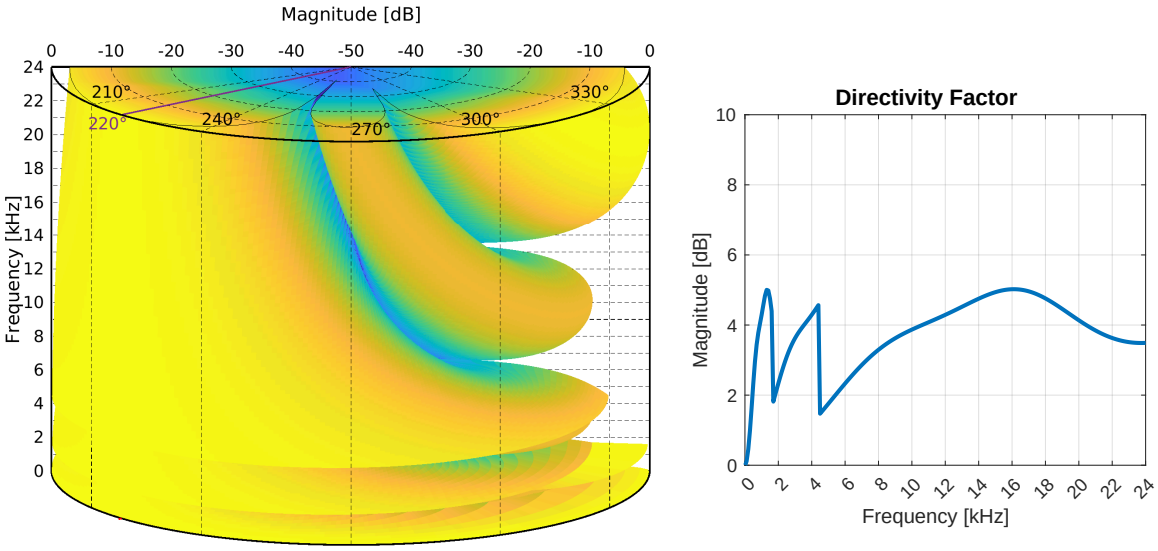


Figure 4.7. Simulated combined broadband beampattern of **DAS 2** (Figure 4.5) with the steering angle changed to 140/220° and lowered viewing angle.

The DF and magnitude variation across frequency in ULA2 and ULA3's regions in **DAS 2** are too low, meaning the spacings should be increased, as this means the number of periods across the ULA is increased.

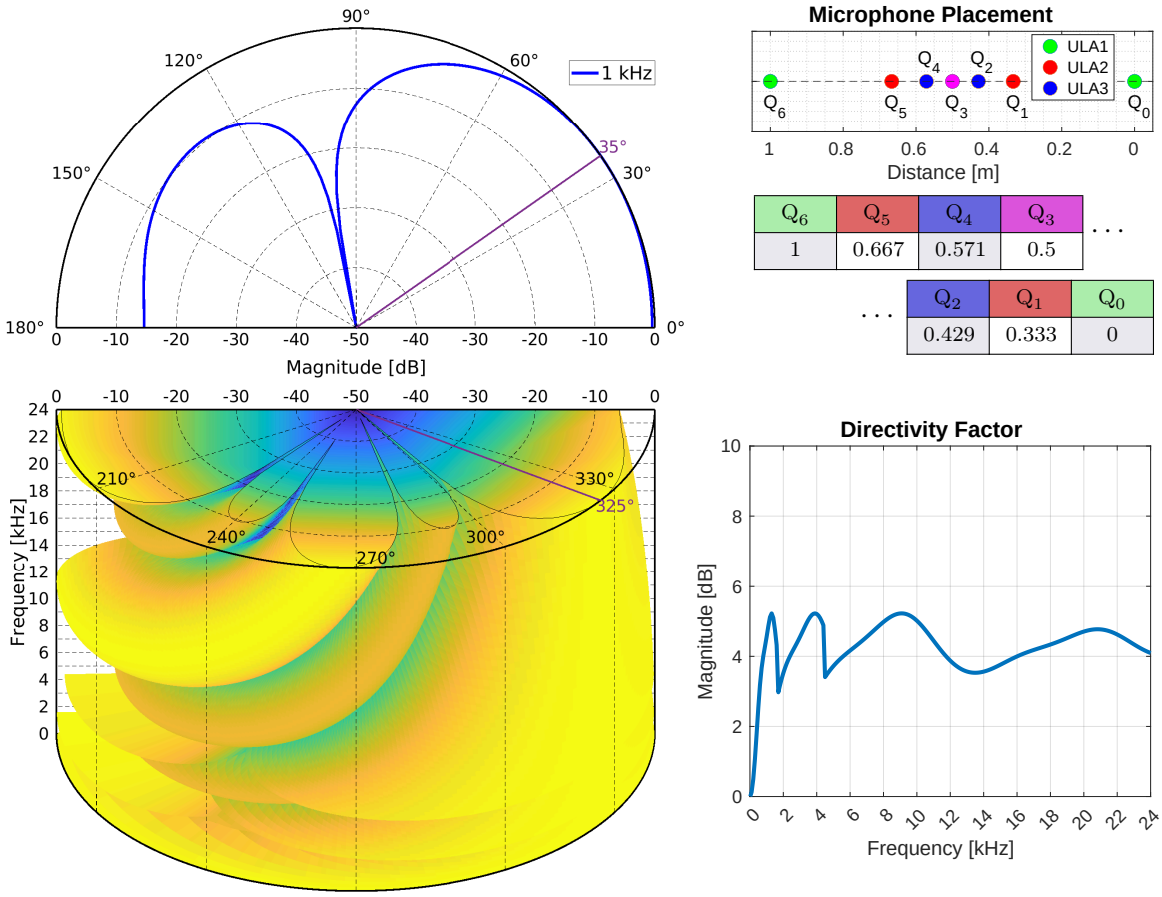
4.1.2 DAS 3 (Separate ULAs)

The microphone spacings are increased for ULA2 and ULA3 in order to increase the variation in suppression for the frequencies above 1700 Hz. The spacings for the three ULAs in **DAS 3** are seen in Table 4.3.

Table 4.3. Spacings between the microphones for the three ULAs in the **DAS 3** beamformer.

δ_{ULA1}	δ_{ULA2}	δ_{ULA3}
$\frac{1}{2} \cdot 1$ m	$\frac{1}{6} \cdot 1$ m	$\frac{1}{14} \cdot 1$ m

The beampattern for **DAS 3** is seen in Figure 4.8 along with the microphone placements and DF. The active ULA is still chosen based on *if*-statements, meaning the transitions between the regions appear rough. The script for the simulation is found in *DAS.mlx* Appendix A.



Q_6	Q_5	Q_4	Q_3	...
1	0.667	0.571	0.5	

...	Q_2	Q_1	Q_0
	0.429	0.333	0

Figure 4.8. Simulated combined beampattern of **DAS 3** consisting of three $M = 3$ delay-and-sum beamformers at 1 kHz and broadband, a list of the microphone placements, and directivity factor of the beampattern. The frequency regions of the ULAs are seen in Table 4.1. Q_3 is the centre microphone in all three ULAs, and the steering angle is set to 35/325°.

Simulations of **DAS 3** with lowered viewing angle and steering angles set to 90°/290° and 140°/220° are seen in Figure 4.9 and 4.10, respectively.

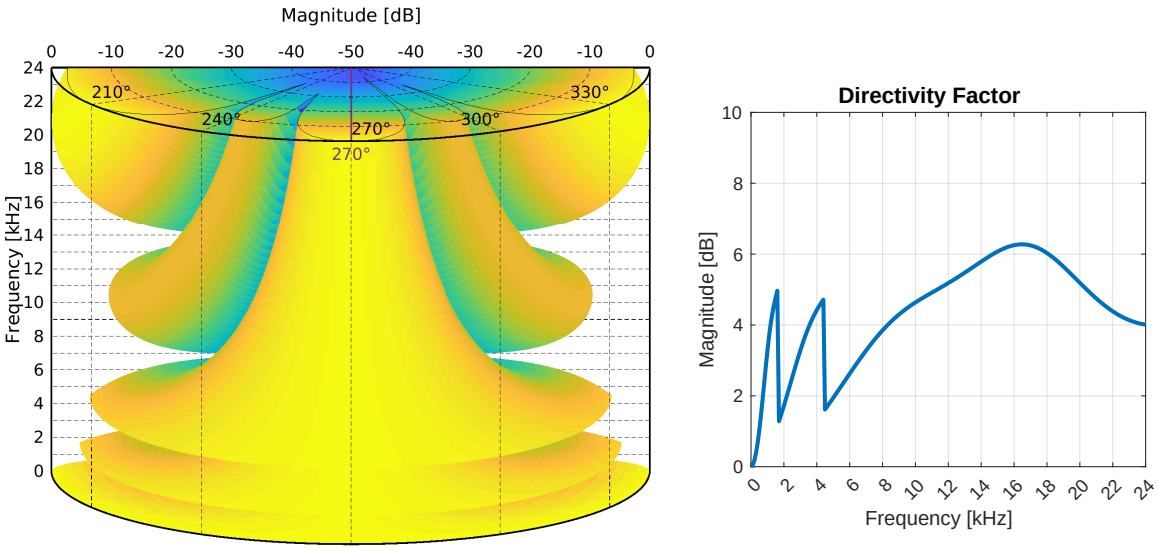


Figure 4.9. Simulated combined broadband beampattern of **DAS 3** (Figure 4.8) with the steering angle changed to 90/270° and lowered viewing angle.

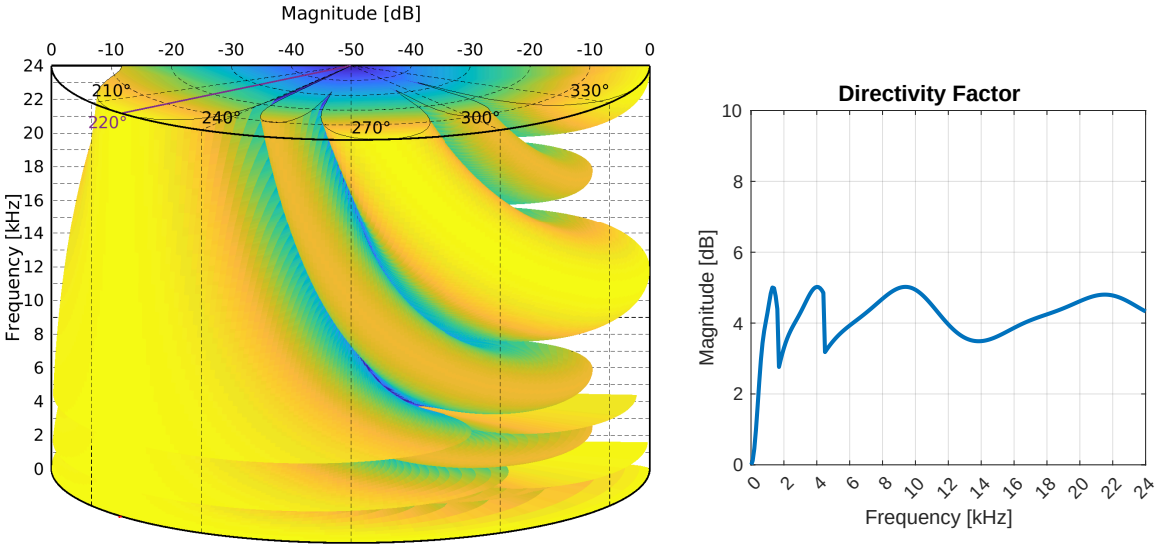


Figure 4.10. Simulated combined broadband beampattern of **DAS 3** (Figure 4.8) with the steering angle changed to 140/220° and lowered viewing angle.

The beampattern of **DAS 3** is deemed sufficient in relation to the design philosophy. However, in order to ensure the beamformer functions as expected and to check the effect of neglecting fractional delays, the beamformer is simulated with discrete signals.

4.1.3 Simulation with Discrete Signal

The preceding simulations of the beamformers were conducted by calculating the gain through the phase vector (Eq. (4.4)), whereas these simulations calculate the gain by comparing the

power of some discrete input signal to the output of the beamformer. In order to separate the input signal into the frequency regions, some filters must be developed.

Filter Design for Region Division

It is important that the filters have linear phase in order to maintain the timing alignment across frequencies. Furthermore, the filters should be designed with shared cut-off frequencies at -3 dB and with similar slopes to maintain a magnitude of 0 dB when summing the outputs in parallel. Based on this, it is decided to use FIR filters. The type of filter and corresponding cut-off frequencies are listed in Table 4.4. The script for generating and plotting the filters is found in *genFIRFilters.mlx* in Appendix A.

Table 4.4. Cut-off frequencies for the three FIR filters used to separate the input signal into the three ULA frequency regions.

	FIR_{ULA1}	FIR_{ULA2}	FIR_{ULA3}
Type	*Lowpass*	*Bandpass*	*Highpass*
Cut-off [Hz]	1700	1700, 4500	4500

The filters are designed using MATLAB's *fir1*-function using the cut-off frequencies listed in Table 4.4. The order of the filters cannot be too high, as this would require significant compute resources, that may not be available. However, choosing a too low order, might result in an inadequate filter response. An order of 60 is chosen for each of them as a middle ground, but is a place for optimisation if the implementation is running too slow.

However, as the *fir1*-function sets the cut-off frequencies to -6 dB, the filters are redone using the frequency at -3 dB as the new cut-off frequency. The frequency and phase response of the filters are seen in Figure 4.11 and 4.12.

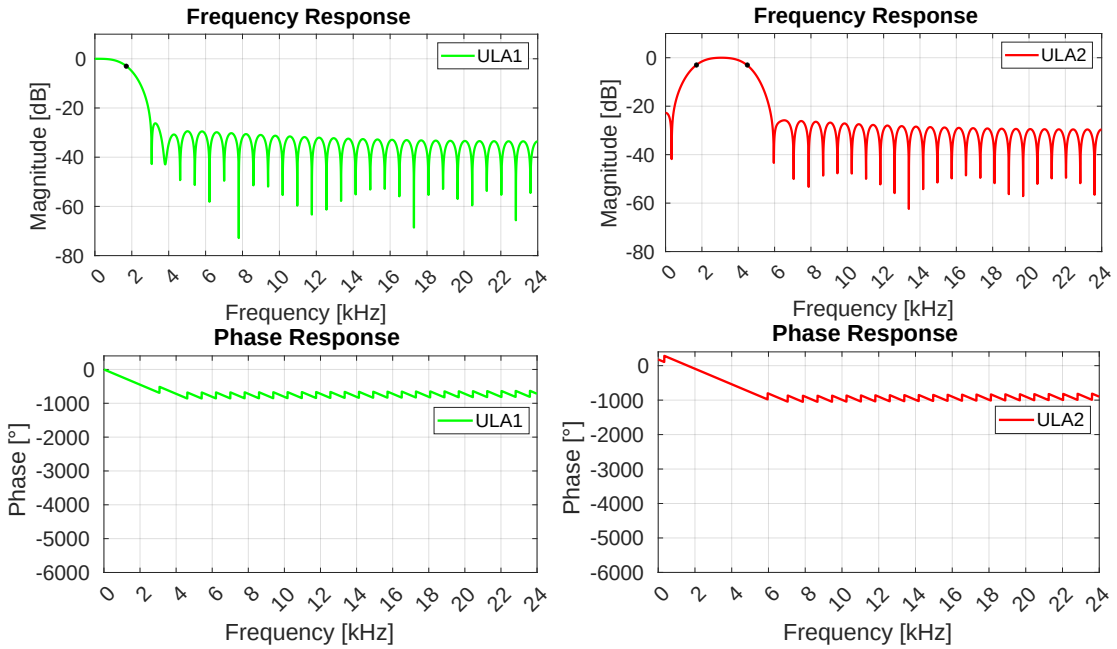


Figure 4.11. Frequency and phase response of FIR_{ULA1} and FIR_{ULA2}. The black dots mark the cut-off frequency at -3 db.

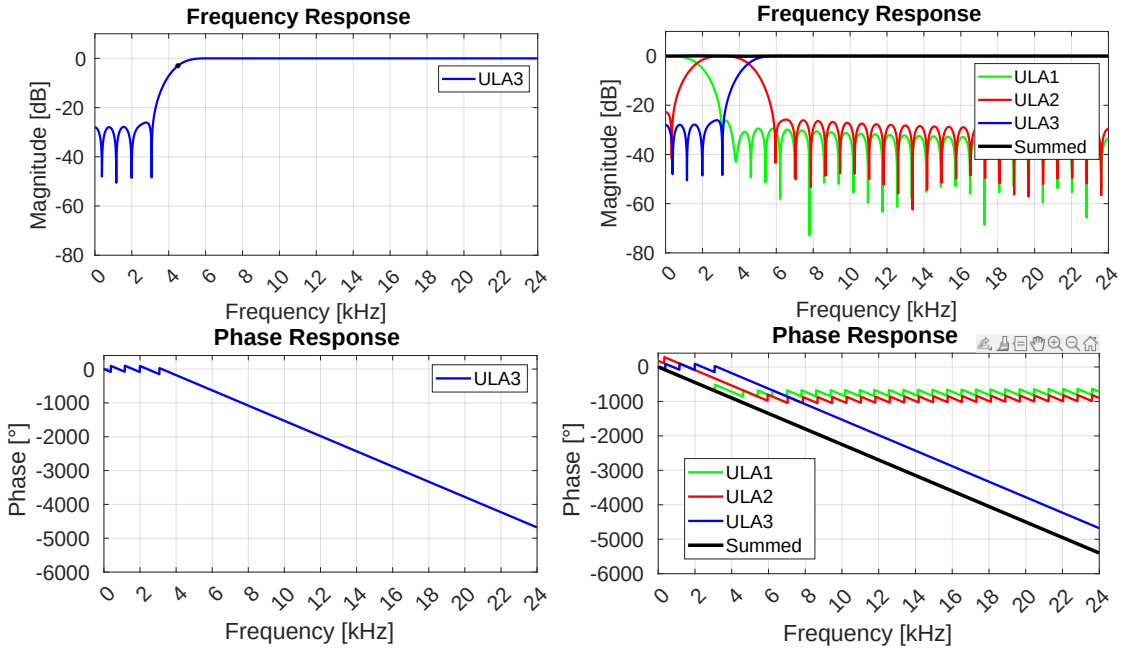


Figure 4.12. Frequency and phase response of FIR_{ULA3} and the sum of the three filters. The black dot marks the cut-off frequency at -3 db.

Simulation of DAS 3

A diagram of the signal flow in the **DAS 3** simulation with a sampled signal is seen in Figure 4.13. The input signal is a single sine wave, and the propagation delays are calculated using the microphone spacings (Figure 4.8), speed of sound, and the signal's incidence angle and is implemented as a phase offset. The filters are applied using MATLAB's *filter*-function. The microphone delays (τ) are calculated with Eq. (4.2), but are implemented as rounded index offsets in the output of the filters. Finally, the delayed outputs are summed in the individual ULAs and divided by three.

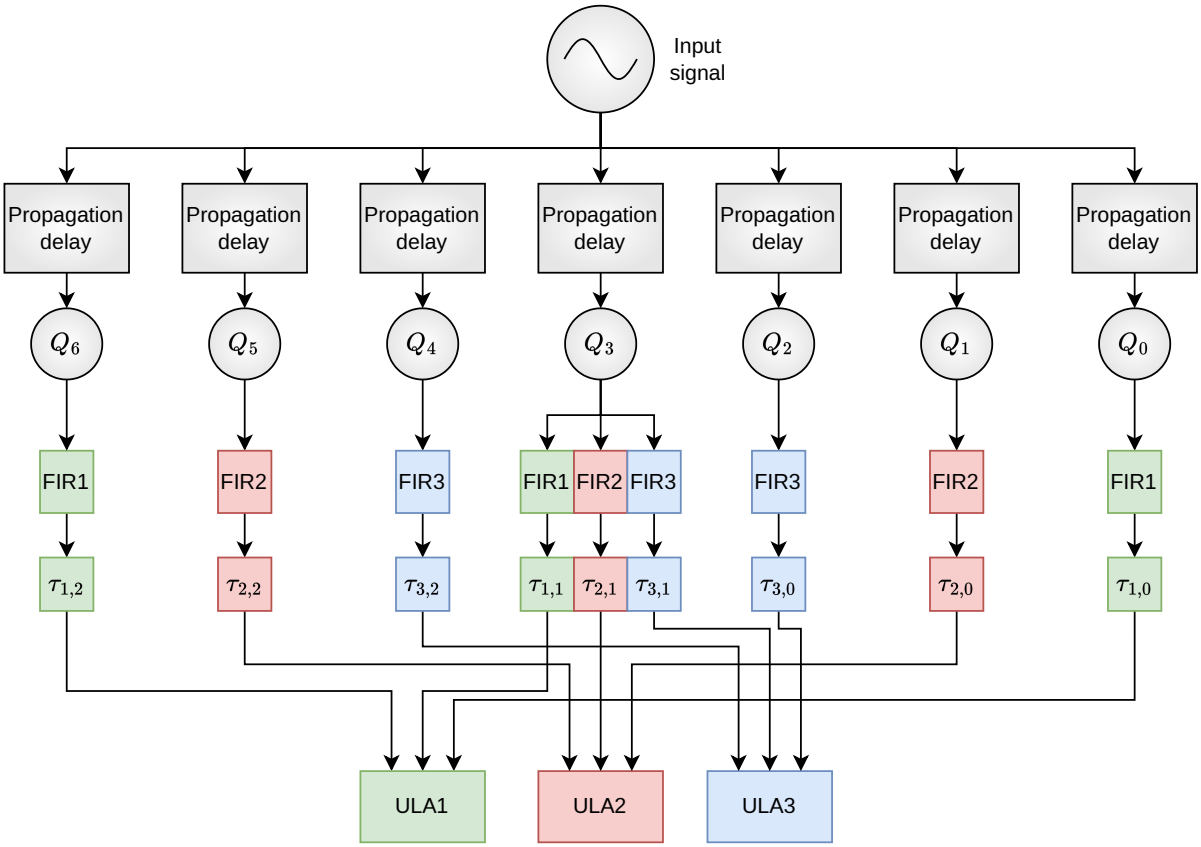


Figure 4.13. Setup of the simulation of **DAS 3**. The propagation delays to the microphones (Q) are implemented as phase offsets in the input signal, whereas the microphone delays (τ) are implemented as rounded index offsets. The delayed outputs are summed and divided by three in the *ULA* blocks.

In order to generate beampatterns, the frequency and incidence angle of the input signal are varied. The input signal frequency range is 0-24 kHz and increments with 100 Hz, and the incidence angle is evaluated from 0-180° with increments of 1°. The resulting broadband beampatterns for steering angles of 35°, 90°, and 140° are seen in Figure 4.14, 4.15, and 4.16, respectively, along with the magnitude at the steering angle across frequencies. The script for simulation is found in *systemSim.mlx* in Appendix A.

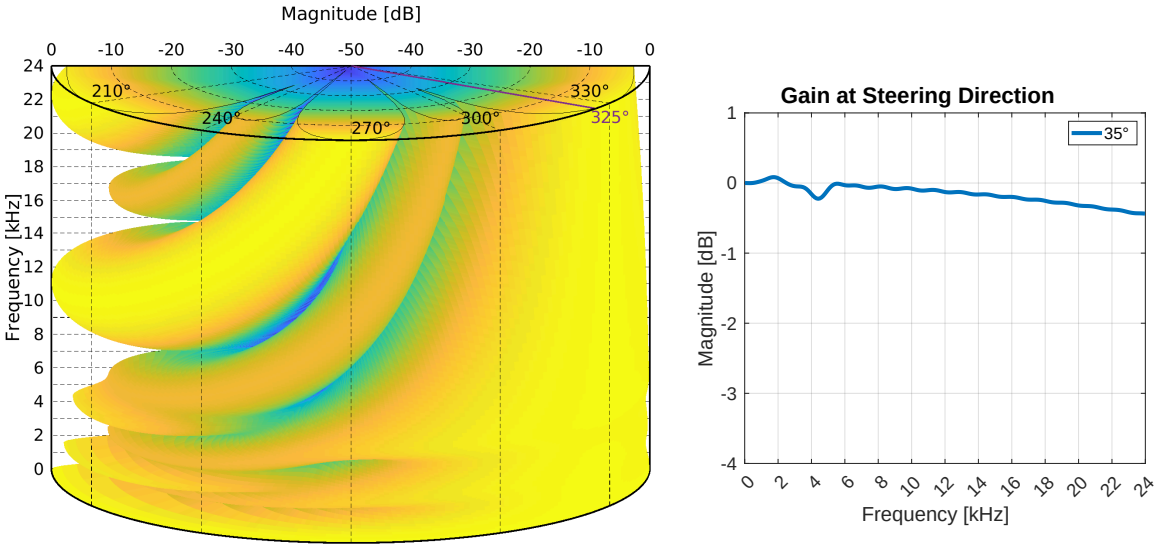


Figure 4.14. Combined broadband beampattern of the **DAS 3** simulation with a discrete input signal and a steering angle of 35/325°. The rightmost graph shows the magnitude at the steering angle.

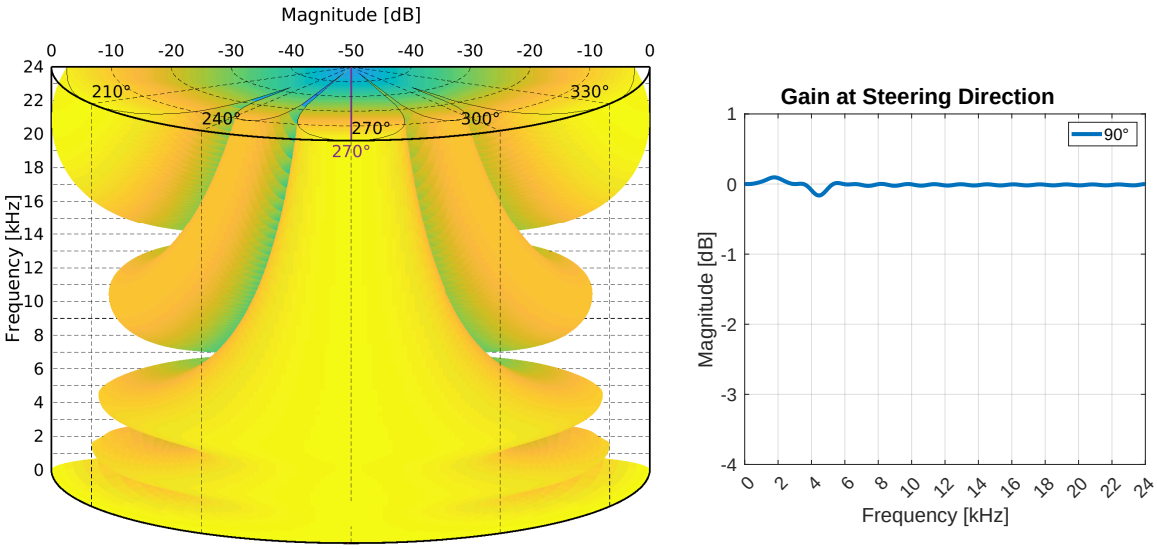


Figure 4.15. Combined broadband beampattern of the **DAS 3** simulation with a discrete input signal and a steering angle of 90/270°. The rightmost graph shows the magnitude at the steering angle.

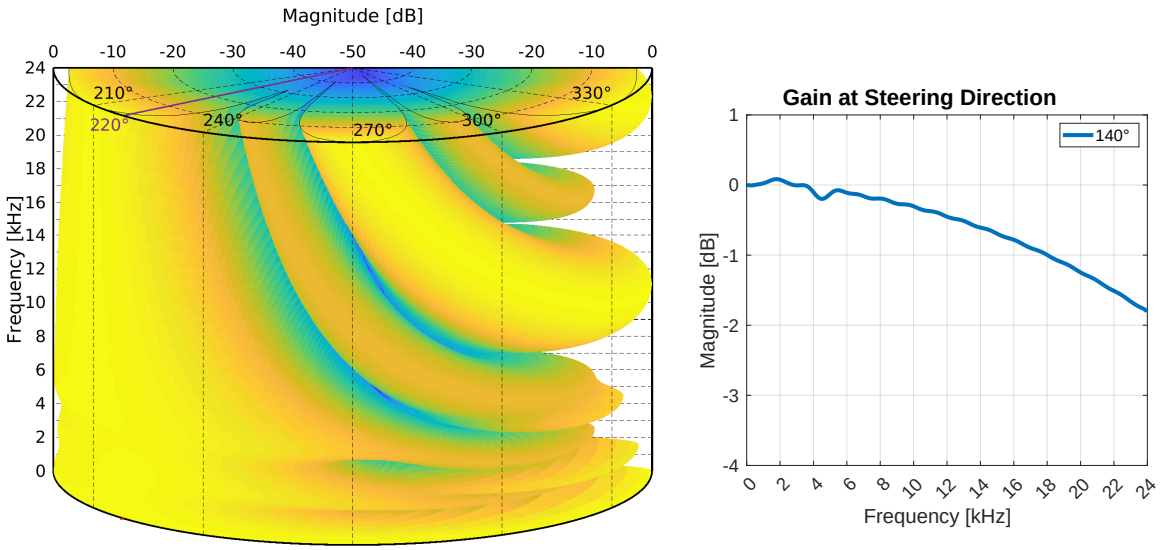


Figure 4.16. Combined broadband beampattern of the **DAS 3** simulation with a discrete input signal and a steering angle of 140/220°. The rightmost graph shows the magnitude at the steering angle.

The beampatterns appear very similar to the simulations using the phase vector, but the transitions are now smoothed due to the FIR filters. However, the magnitude is not consistently 0 dB at the steering angle. The downward trend of the magnitude in the steering angle (primarily noticeable in Figure 4.16) is due to the rounding of the microphone delays. The microphone delays (τ) are smaller at higher frequencies which means the fractional part of the delay is more significant.

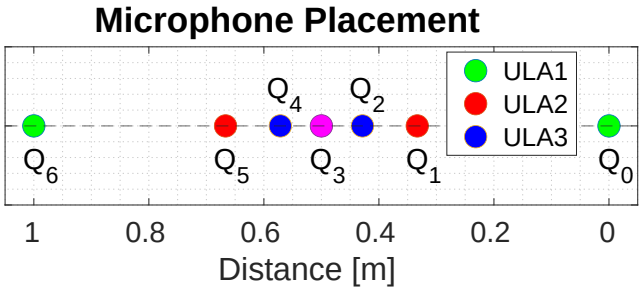
The small distortions at 1700 and 4500 Hz are most likely caused by an imperfect match of cut-off frequency in the FIR filters, as the adjusted cut-off frequencies for -3 dB are derived from a frequency response with a finite frequency resolution.

However, the slight distortions in the steering direction are deemed acceptable for now. Though, if it proves that maintaining absolute integrity of the signal is critical, fractional delays will be implemented.

4.1.4 Conclusion

Three delay-and-sum beamformers were evaluated: one using eight microphones uniformly spaced (**DAS 1**) and two using seven microphones separated into three uniform arrays (**DAS 2** and **3**). **DAS 1** had too many and too narrow lobes at the higher frequencies, whereas **DAS 2**'s directivity factor was too low. Therefore, modifications were made to **DAS 2** and **DAS 3** was developed and its pattern deemed satisfactory.

In order to verify the beamformer's function on a discrete time signal and review the effect of rounding the microphone delays, further simulations were conducted. The simulations showed a slight distortion in the steering direction, but was accepted for now. However, if the distortion interferes with recognising the input impulse, fractional delays will be implemented. The final microphone placements for the beamformer **DAS 3** are seen in Figure 4.17.



Q_6	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0
1	0.667	0.571	0.5	0.429	0.333	0

Figure 4.17. Determined microphone placements for the beamformer (**DAS 3**). The centre microphone (Q_3) is used in all three ULAs.

4.2 Physical Design

The designed beamformer only utilises seven microphones, although it was decided to have eight in the system in Section 3.5. This means that the positions of the seven microphones used in the beamformer are already decided (Figure 4.17), but the position of the eighth microphone is not, and can be placed anywhere. However, it is decided to discard the eighth microphone for now, such the arrangement of microphones can be symmetrical in the array. The eighth microphone might be added at a later time at a distance further away from the beamformer array in order to further increase the variety of the captured data.

In Subsection 3.4.3 it was decided that the microphones will be turned into hydrophones by insertion into nitrile gloves. This results in a mounting problem, since screws or bolts would need holes. Therefore a clamping system is designed as seen on Figure 4.18. The microphone can then be slid into the middle piece, after which it can be inserted into a nitrile glove and clamped to a flat surface such as plexiglass.

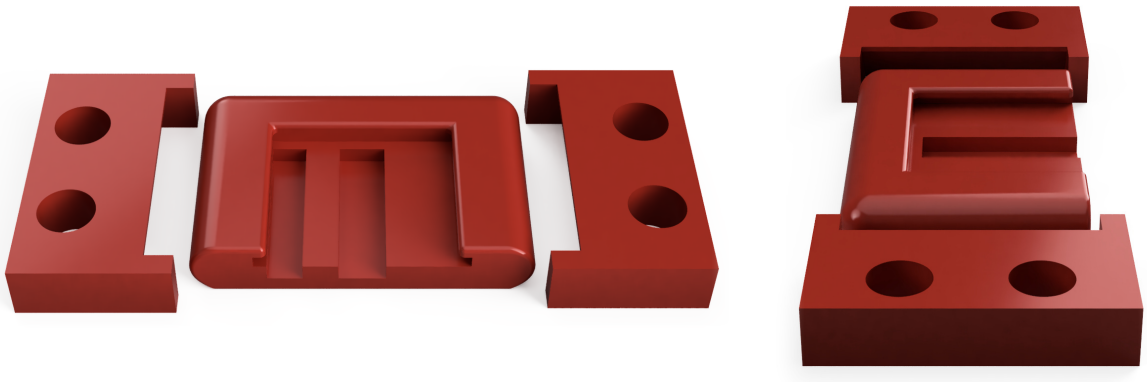


Figure 4.18. Clamping system, where microphone can be inserted into middle piece, which can then be inserted into a nitrile glove, which can then be fastened to a flat side by clamping.

The end of the nitrile glove then needs to be closed off in a way that allows the wiring to pass through. This is decided to be done by use of an inner tube of hard plastic and an outer tube of

thermoplastic polyurethane, which is a elastic plastic as seen on Figure 4.19.

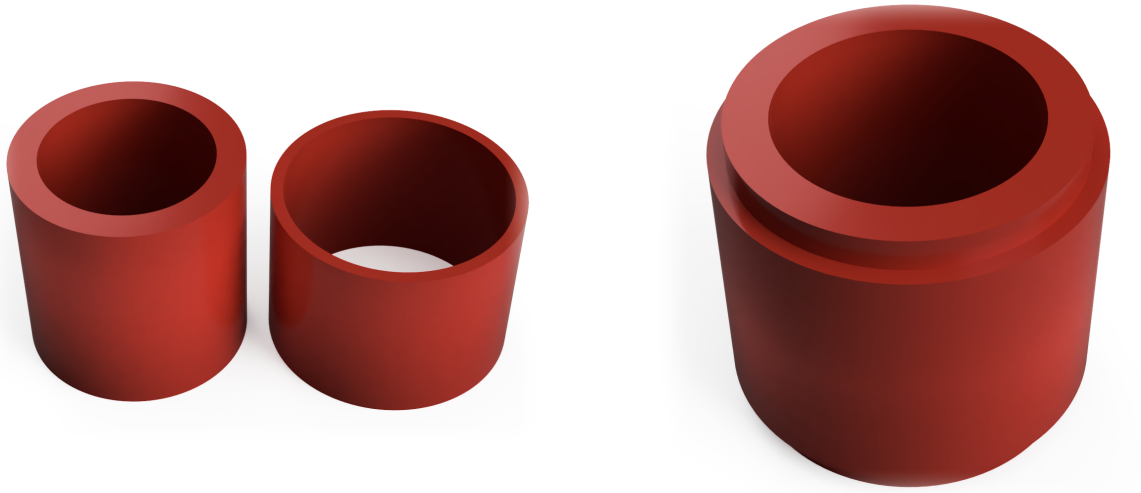


Figure 4.19. Nitrile glove clamping method, consisting of an inner tube of hard plastic and an outer tube of soft plastic, wherein the nitrile glove will go in the middle and be waterproofed by clamping force as seen on the right.

The nitrile glove will then go in between the soft and hard plastic which will then be waterproof by the clamping force of the elastic plastic. The wiring is then pulled through the inner tube, which is then waterproofed by filling it with silicone.

With the waterproofing determined, a mounting surface has to be built. Since wood floats, it is deemed best to use plexiglass to build a structure. The structure has to be at least 1 m long to hold the microphones, so 110 cm is chosen. A structure for mounting the plexiglass to a platform on e.g. a harbour also has to be added. The structure is seen on Figure 4.20, where the red mounts is the clamping mechanism from Figure 4.18.

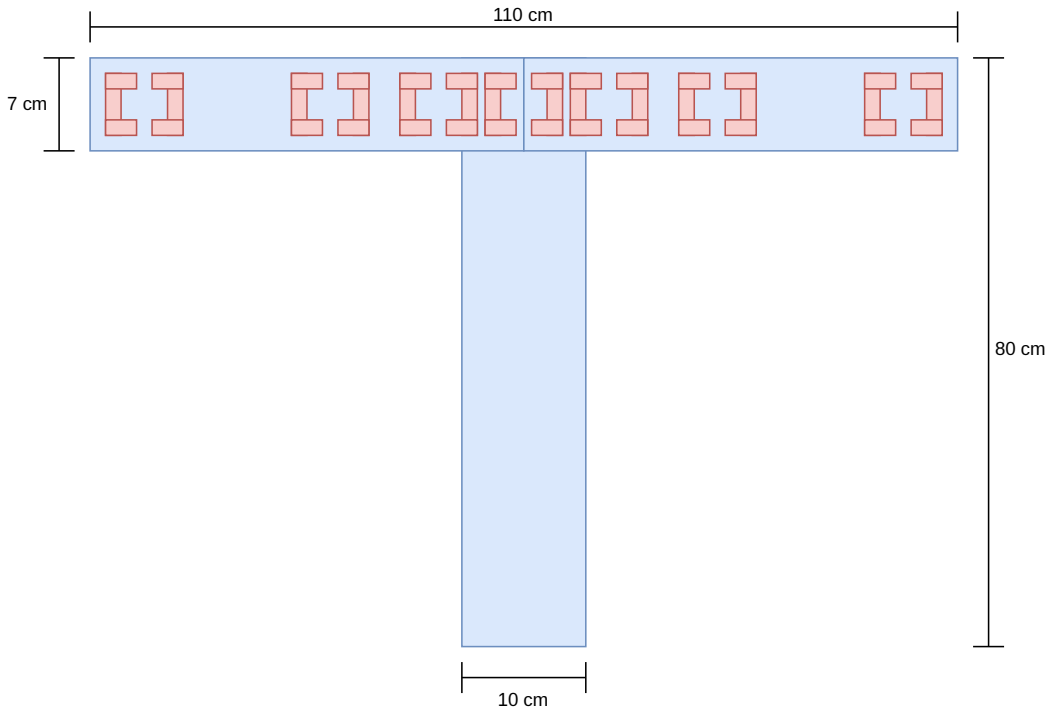


Figure 4.20. Drawing of the plexiglass mount for the microphones.

With a drawing of the mount, construction can commence. The plexiglass is cut with a jigsaw with a ruler for guidance afterwhich it is bolted together. Mounting holes for the clamps are carefully measured out and drilled. The finished mount with the microphones installed, however without the nitrile gloves, is seen on Figure 4.21.

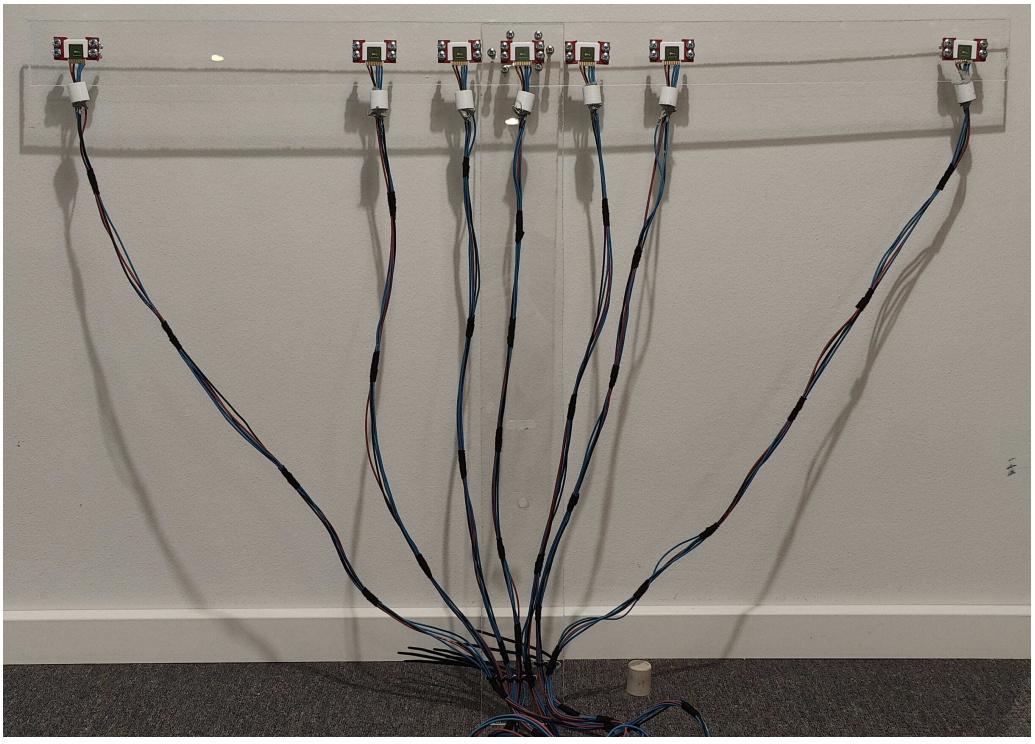


Figure 4.21. Photo of the finished mount with microphones installed, but no nitrile gloves.

A closeup of the nitrile glove waterproofing mechanism installed is seen on Figure 4.22. The left side shows the glove inserted, while the right side shows how the inner tube is waterproofed using silicone.

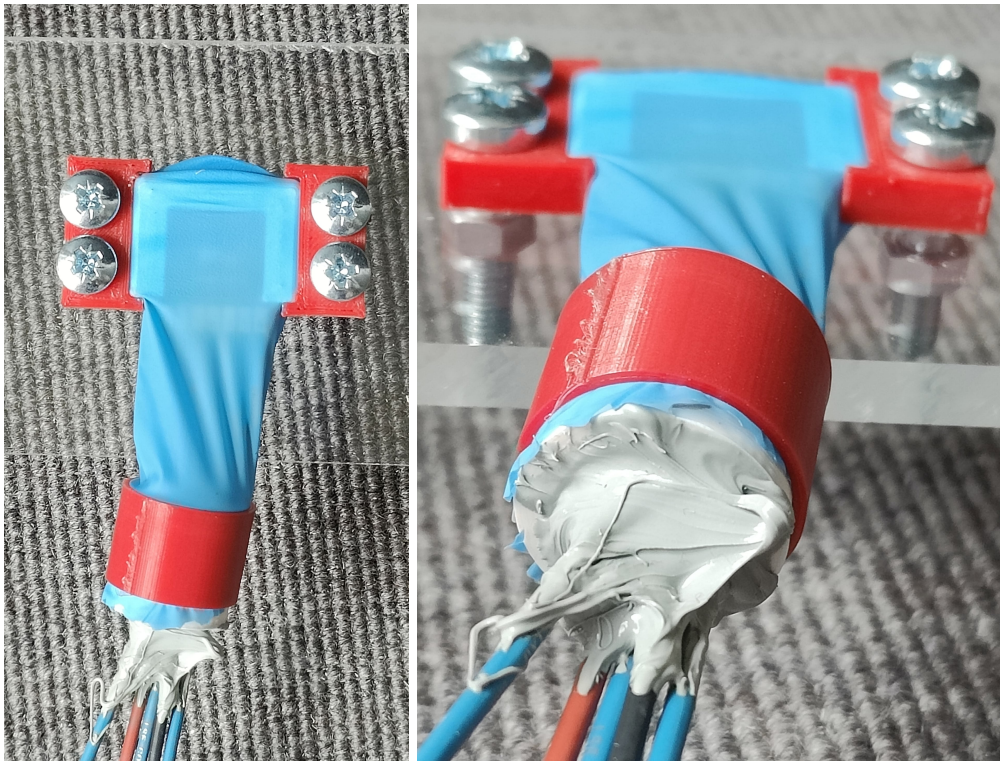


Figure 4.22. Photo of the finished waterproofing mechanism. The middlepart of the clamping mechanism has the microphone inserted and is placed in a nitrile glove. The glove then goes around the inner tube and has the outer tube clamp it tight.

With the physical construction done, the digital implementation can commence. The nitrile gloves can be taken on or off, allowing for both in water and in air testing.

4.3 Processing Platform

In Section 3.4 it was chosen to use a PSoC 5LP and I2S microphones as a prototype sampling circuit, which is offloaded to an ESP32-S3 using SPI for surveillance and then further offloaded to a socket-server via Wi-Fi whenever a splash is detected.

The system overviews from Section 3.4 were made based on the assumption that individual peripherals on the ESP32-S3 could be set in *Deep-Sleep* mode for power saving. This assumption was false, as the *Deep-Sleep* mode turns off the main CPU and almost all peripherals including SPI. Therefore use of the *ULP Coprocessor* is dropped.

In order to implement the entire system, several subsystems can be defined for compartmentalisation. The system is seen in Figure 4.23, whereof the implementation order will be **I2S Microphone Sampling Using DMA**, **SPI Communication**, **Offloading** and **Data processing**.

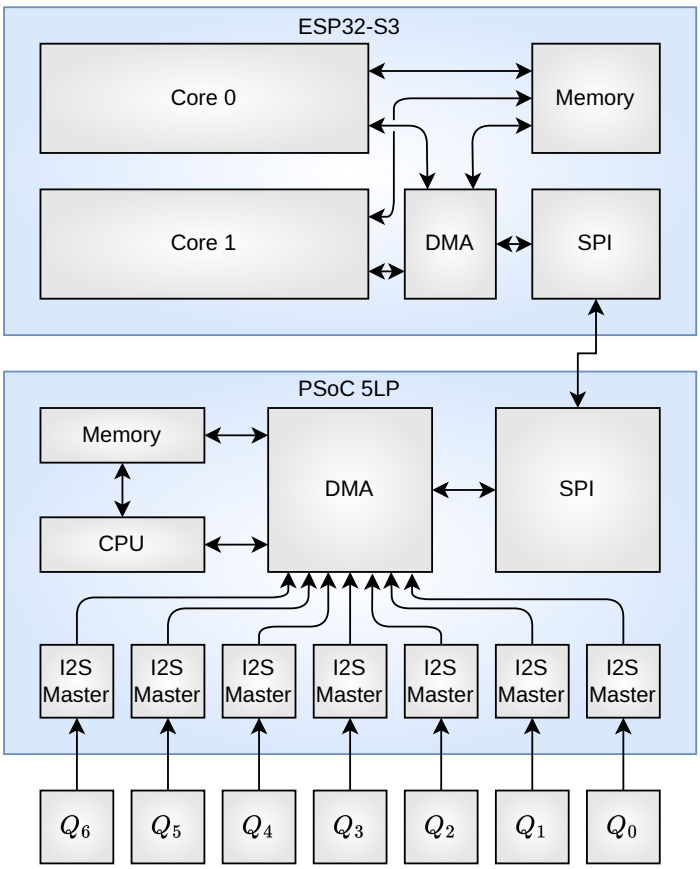


Figure 4.23. Overview of the hardware that needs to be configured.

Before implementation can commence, a common logic level has to be found, since the PSoC 5LP development board runs at 5 V, while the ESP32-S3 is designed for 3.3 V. According to the documentation of the PSoC 5LP [19], the programmer runs at 5 V, while the chip itself can be tied to an external reference from 3.3 V to 5.5 V by removing diode D1 from the programmer and connecting an external reference to the VDD net. A photo of the microcontrollers interconnected on a stripboard is seen on Figure 4.24, with connectors for the I2S microphones.

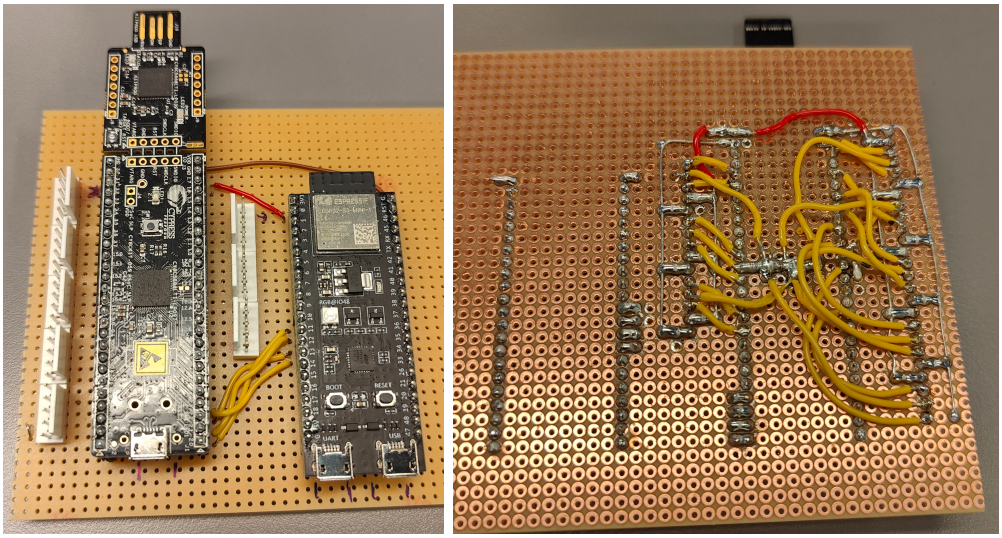


Figure 4.24. Front and back of the stripboard, with the ESP32-S3 and PSoC 5LP interconnected and ready for the I2S microphones via the onboard connectors.

The subsystems will be individually developed and interconnected as they are verified. For development on the PSoC 5LP, the development suite PSoC Creator 4.4 is used. For ESP32-S3 development the ESP-IDF is used in Visual Studio Code.

Development in PSoC Creator consists of three parts: Design Wide Resources (pin routing, clock configuration, etc.), Top Design (Component and signal routing), and source files (Code on CPU). First, the needed components are placed in the Top Design. Pins are also placed here, as well as needed clocks, whereafter they can be configured and routed physically in the Design Wide Resources area. Finally the source files i.e. code in the C language is used for final configuration of the components such as DMA configuration using their API.

Development using the ESP-IDF is more straightforward, since it is a more traditional microcontroller, where everything is configured in the source files. The only exception is the installation of external libraries, such as the ESP-DSP library, for efficient use of hardware and configuration of settings for e.g. the serial monitor in the SDK configuration editor.

4.3.1 Implementation of I2S Microphone Sampling Using DMA

In order to receive audio from the I2S MEMS microphones on the PSoC 5LP, an I2S configuration that is supported by both has to be found. In Subsection 3.4.3 it was determined that each microphone will need its own I2S master with a 32-bit word size. An I2S master is seen on Figure 4.25, specifically the one for the first microphone. As also mentioned in Subsection 3.4.3, the I2S masters can share the external clock pin, when the input clock is also shared, so `sck` is left unconnected on the rest of the I2S masters and `sck_I2S_one` is used for the rest instead. This will result in synchronised sampling, which means the I2S buffers will fill synchronised and be full at the same time and hence be ready for processing at the same time, with the same time data.

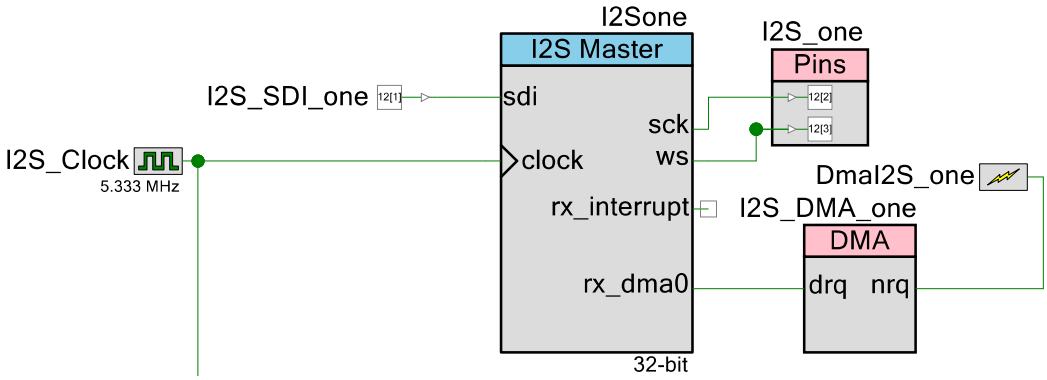


Figure 4.25. PSoc 5LP I2S TopDesign zoomed in on the I2S Master for microphone one.

The wished sampling frequency is 48 000 Hz, but as seen in Table 3.4, the microphone's frequency range is 20 to 20.000 Hz. This means the sampling frequency should just be higher than 40.000 Hz. The sampling rate is chosen by selecting the clock by use of Eq. (4.5) [20].

$$f_{clock} = 2 \cdot f_s \cdot t_{ws} \tag{4.5}$$

f_{clock}	Clock frequency	[Hz]
f_s	Sampling frequency	[Hz]
t_{ws}	Word select period	[·]

The standard sample rates for audio are 44.1 kHz and 48 kHz [23], which results in a f_{clock} of 5.6448 MHz or 6.144 MHz. Clock generation on the PSoC 5LP is done by division of one of the system clocks. Neither 5.6448 MHz or 6.144 MHz are precisely feasible with the available system clocks, resulting in a non-standard sample rate. This is incompatible with the desire to create a standard dataset for splash detection, but it is deemed a non-issue for the prototype. The master clock defaults to 24 MHz [24], leaving a division factor of 5 as the closest, resulting in `I2S_Clock` being 6 MHz i.e. a sample rate of 46 875 Hz.

Each I2S master is configured with a static bit resolution of 32 data bits with a word select period of 64 in rx only mode. It is configured to *Mono left*, which means the L/R pin on the microphone should be grounded so the configurations match [22]. Furthermore rx dma requests are enabled and routed to a DMA controller, which is routed further to a interrupt component which bonds a hardware event to software. The physically routed pins are `sck` (clock), `ws` (word select) and `sdi` (serial data in), where the clock is shared, so only data and word select are uniquely routed for each master. This is seen on Table 4.5.

Table 4.5. Overview of I2S pinout.

#	1	2	3	4	5	6	7
sck	P12[2]	-	-	-	-	-	-
ws	P12[3]	P3[1]	P1[4]	P15[0]	P3[5]	P15[2]	P0[1]
sdi	P12[2]	P3[0]	P1[5]	P15[1]	P3[6]	P3[3]	P15[3]

With everything routed, it can be initialised and started from the code. This is seen on Figure 4.26 which explains the I2S setup from *SamplingCircuit.cydsn/main.c* from Appendix A. The flowchart depicts the setup of I2Sone, which is the I2S master for microphone one.

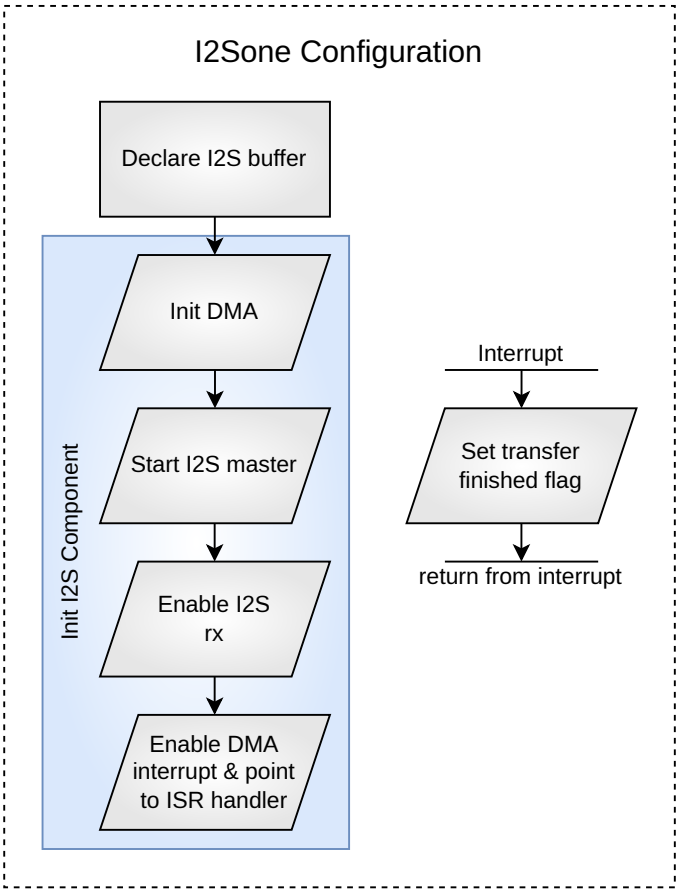


Figure 4.26. Diagram of I2S setup and test code for I2Sone, which is the I2S master for microphone one. This is copied for all seven microphones.

The *Init DMA* block consists of several steps. First the DMA block from Figure 4.25 is initialised on a DMA channel, while specifying 1 *byte per request*, 1 *request per burst*, *the address space of the source* and *the address space of the destination*. Then an instance of a TD (Transaction Descriptor) is created and configured with the transfer count i.e. buffer length, it is configured to increment the destination address after each burst and to create a DMA done event when finished. Furthermore it is set to point to itself, when it is finished, so continuous transfer is done. The specific start source and destination addresses are then specified and finally the TD is bound to the channel, so it can be enabled. The configured DMA flow is visualised on Figure 4.27.

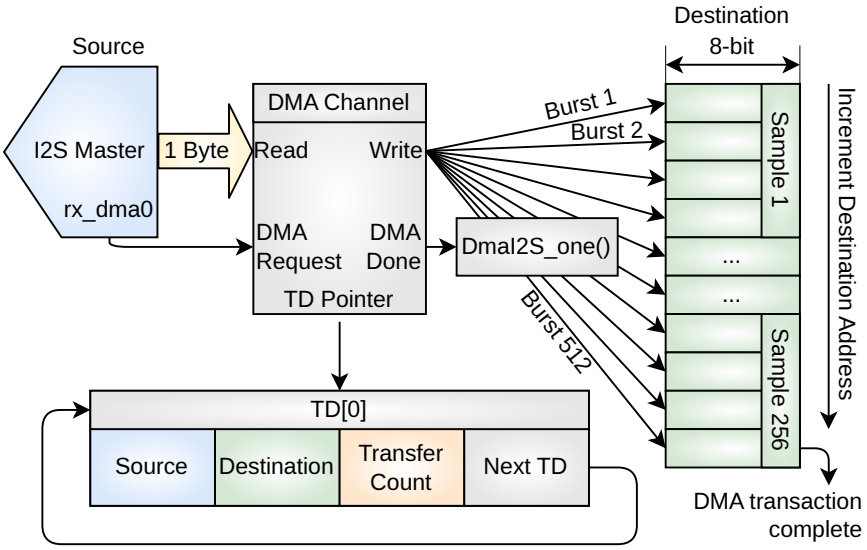


Figure 4.27. Diagram of the I2S DMA configuration.

With everything configured, the setup needs to be verified. The PSoC 5LP does however have no debug console, meaning that facilitating print statements would need to be implemented manually. This could be done by use of the `USBUART` block, but this was made incompatible when the voltage reference was changed to 3.3 V [21]. Since a communication protocol for communication between the PSoC 5LP and the ESP32-S3 has to be found anyway, this line will also be used for debugging and verification of the PSoC 5LP after it is implemented.

4.3.2 Implementation of SPI Communication

In order to stream the seven microphones in real time to the ESP32-S3, SPI was chosen. The PSoC supports I2C, SPI, UART, USB, CAN and LIN [19] while the ESP32 supports Wi-Fi, Bluetooth, SPI, I2C, UART and CAN [25]. The overlapping protocols are SPI, UART, I2C and CAN, whereas SPI was chosen, since it supports the highest throughput. In order to transmit microphone samples as soon as all buffers are full, the PSoC is elected SPI master. Then the ESP32 can focus on processing data instead of having to request it as well.

Each microphone is producing 46 875 32-bit samples per second, resulting in 1 312 500 bytes per second or 10 500 000 bits per second. The throughput of the SPI master on the PSoC thus has to be higher than 10.5 Mbps, which is within the specified 18 Mbps in high-speed mode but higher than the 9 Mbps in normal operation [21] thus high-speed mode is required on the PSoC. The ESP32-S3 SPI slaves are designed to operate at up to 60 MHz, so a data rate of 30 Mbit [26], meaning the PSoC is the limiting factor. Before software can be configured, the microphones, the PSoC, and the ESP32 are wired as seen on Figure 4.28.

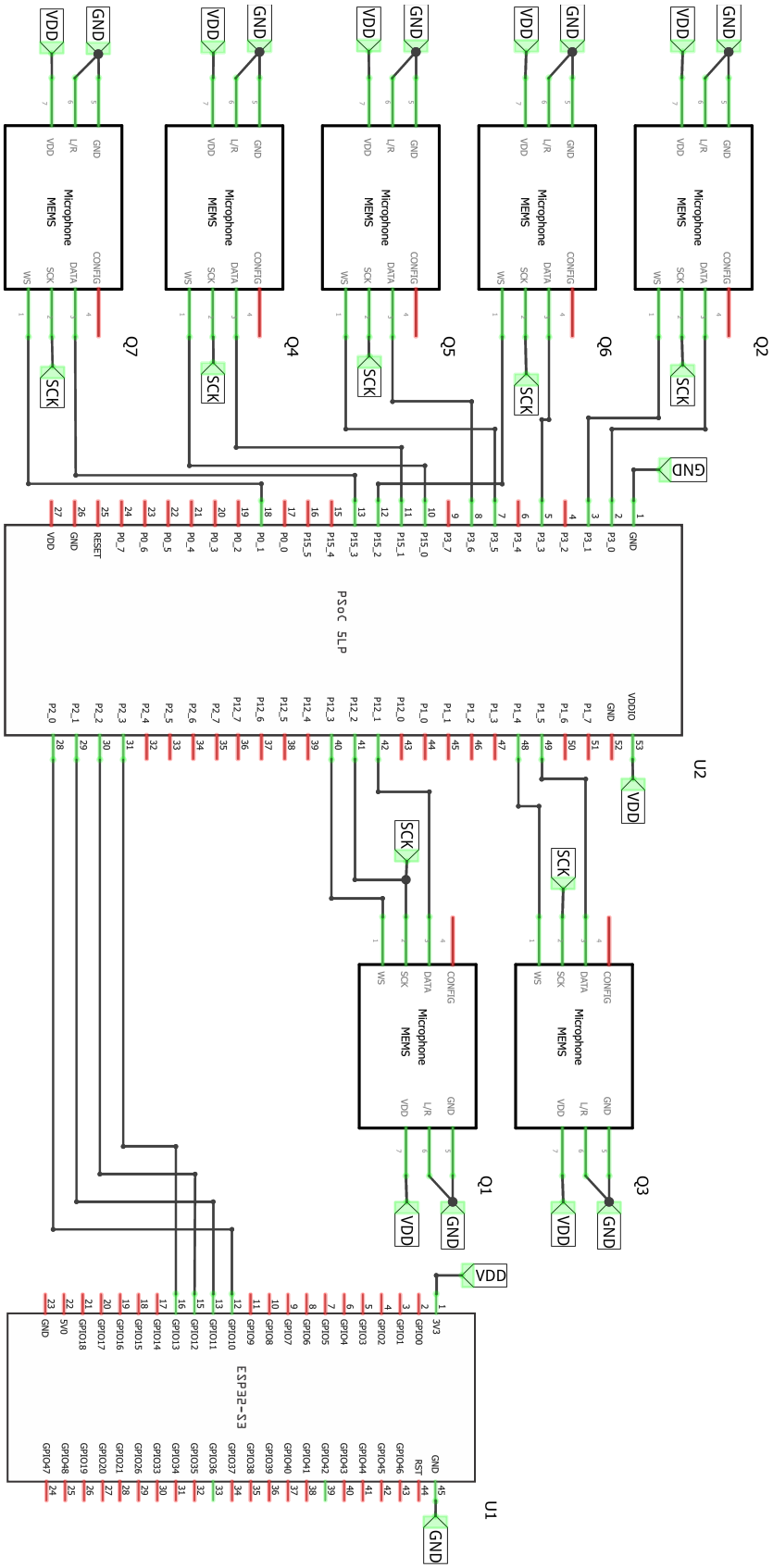


Figure 4.28. Schematic of wiring for microphones, PSoC 5LP, and ESP32-S3.

SPI on the PSoC 5LP

The microphone setup on the PSoC 5LP was described in Subsection 4.3.1. A SPI master is added with both MOSI and MISO in `CPHA = 0, CPOL = 0` mode. It is configured with 8 data bits and a shift direction of MSB first. Interrupt on Tx FIFO Empty is enabled, high-speed mode is enabled and the clock is routed to `SPI_Clock` which is set based on division of `MASTER_CLK` resulting in a data rate of one-half `SPI_Clock`. Since the `SPI_Clock` needs to be divisible by the `MASTER_CLK` and higher than 21 MHz, the `SPI_Clock` is set to 24 MHz and the `MASTER_CLK` is set to 48 MHz. Since the `I2S_Clock` is generated from the `MASTER_CLK`, the `I2S_Clock` is changed to 5.333 MHz, resulting in a sample rate of 41 666 kHz instead of 46 875 kHz as anything above 40 kHz is enough and smaller division steps are available with a higher `MASTER_CLK`. The SPI master is seen on Figure 4.29.

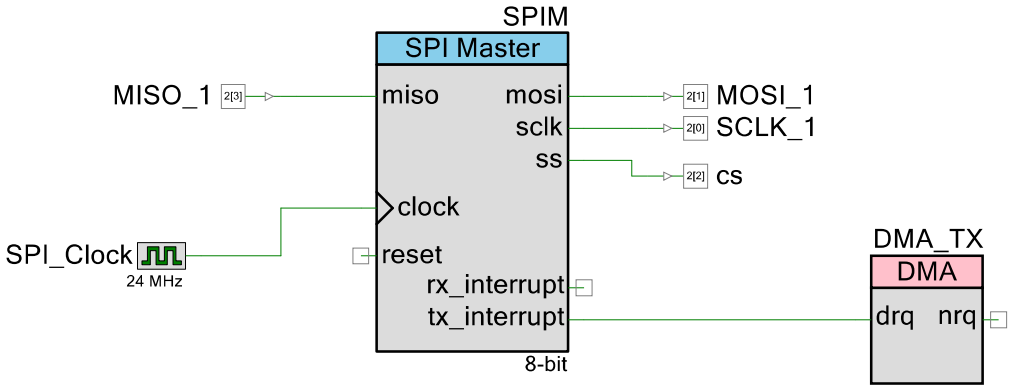


Figure 4.29. PSoC 5LP SPI master TopDesign zoomed in on the SPI Master.

The I2S modules are initialised as in Figure 4.26. The SPI module is initialised by calling `SPIM_Start` and initialising DMA. A DMA channel is configured, while specifying 1 *byte per request*, 1 *request per burst*, *the address space of the source* and *the address space of the destination*. In contrast to the I2S DMA configuration, two instances of a TD is created, where the first one is configured similarly to I2S with address increments after each burst, it is however configured to switch to the second TD when done. The specific start source and destination addresses are then specified and the first TD is bound to the DMA channel, so it can be enabled. The second TD is configured to watch a variable called `InterruptControl`, which is a copy of `SPIM_TX_STATUS_MASK_REG`. Since the TD is watching a valid DMA status mask, it can then be triggered done by doing a logic OR with `SPIM_INT_ON_TX_EMPTY`. This is usually done by the DMA channel, but by not attaching it to a DMA channel it can be exploited for manual control. This way the SPI DMA transfer can be triggered manually, when all I2S buffers are full. The configured DMA flow is visualised on Figure 4.30.

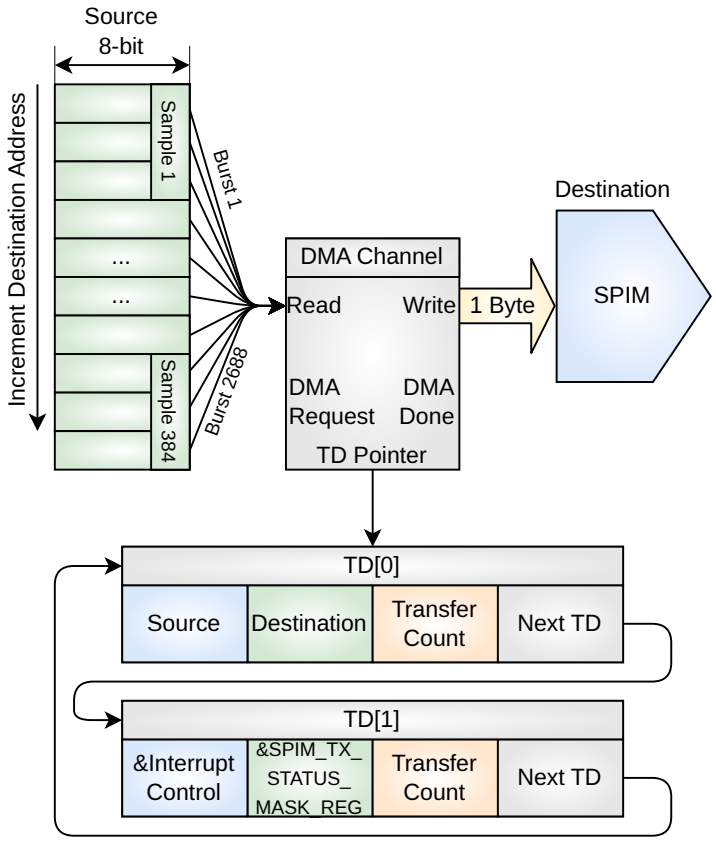


Figure 4.30. Diagram of the SPI DMA configuration.

Figure 4.30 shows samples as 3 bytes, which is because the I2S microphones generate 24-bit samples, even though it uses a 32-bit word length. This is because the last 8-bits of every sample is zeroes, meaning they can be discarded when moving the samples to the SPI buffer. Since the I2S DMA channels are continuously fetching data and writing to the buffer, a second buffer is added to each I2S DMA channel. The new program flow on the PSoC is seen on Figure 4.31 or in *main.c* in Appendix A.

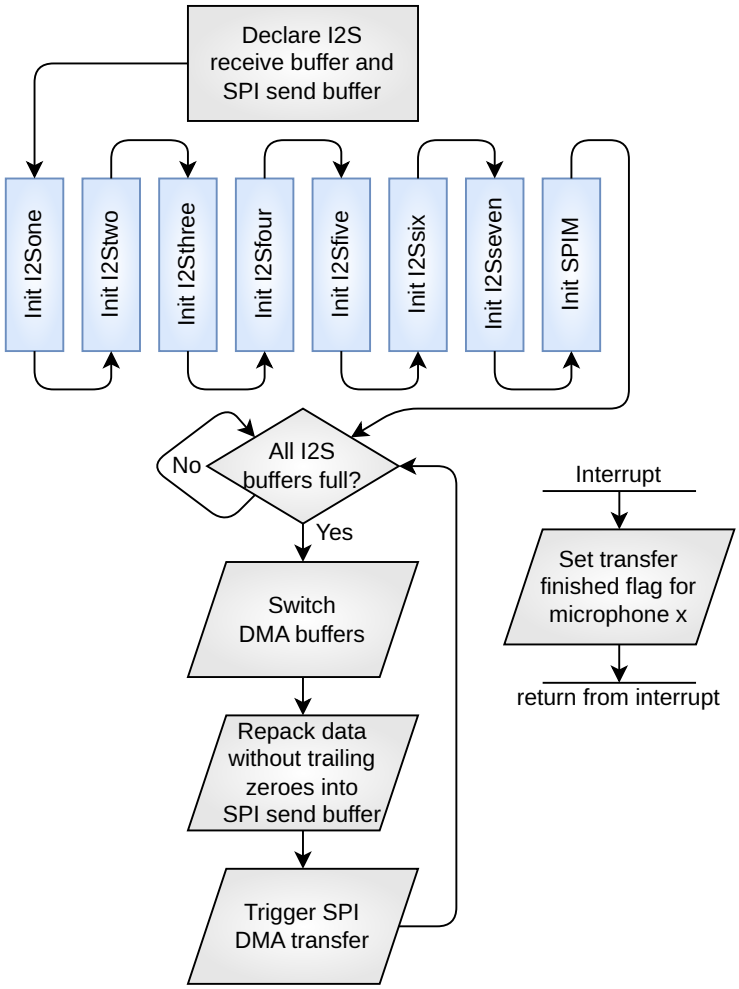


Figure 4.31. Diagram of the PSoC code as seen in *SamplingCircuit.cydsn/main.c* in Appendix A.

The second buffer ensures enough time for removing zeros when moving the samples to the SPI buffer. This is done by swapping the TD destination address for every I2S TD whenever the transfer finished flag is set by its interrupt handler.

SPI on the ESP32-S3

The ESP-IDF comes with a SPI slave driver, aiding the configuration. First the bus has to be configured, by selecting the pins for MOSI, MISO and SCLK. Then the SPI slave interface is configured in mode 0, corresponding to the `CPHA = 0, CPOL = 0` mode on the PSoC. Here, the pin for slave select is also configured as well as post setup and post transaction interrupts. Then the SPI slave driver is initialised selecting the `SPI2_HOST`, which is not used internally, and the DMA channel is set to auto. A SPI transaction can then be triggered with a pointer to the desired receive buffer and a desired transaction length in bits as seen on Figure 4.32.

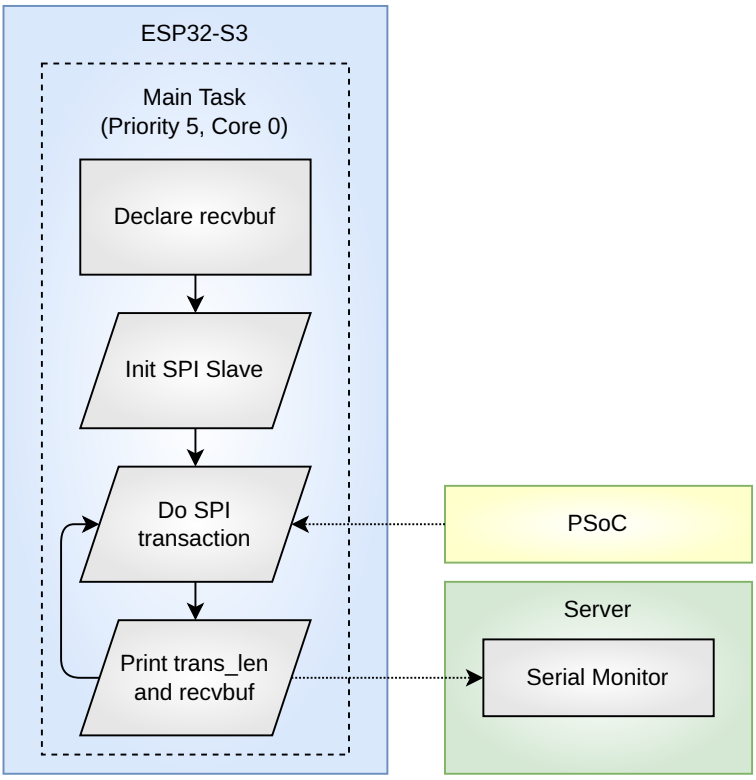


Figure 4.32. Diagram of the test code for the ESP32-S3 showing the FreeRTOS task for SPI and that the data and transaction length is printed to serial monitor on the server.

In order to test the SPI connection between the PSoC 5LP and the ESP32-S3, the SPI buffer on the PSoC is filled with an incrementing for loop, so the data is easily verifiable in the serial monitor on the ESP32. This is seen on Figure 4.33.

```
Transaction length in bits: 21504
First 128 bytes:
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031
323334353637383939a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f60616263
6465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f
Last 128 bytes:
ff000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f30
313233343536373839a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f606162
636465666768696a6b6c6d6e6f707172737475767778797a7b7c7d7e7f
```

Figure 4.33. Screenshot of the serial monitor, showing the amount of bits received, the first 128 bytes and the last 128 bytes received.

The captured measurements show that the SPI setup between the PSoC 5LP and ESP32-S3 is correct. The expected number of bits is received and the output, which is printed as a 32-bit hex string that counts up as expected.

4.4 Implementation of Offloading

In order to make sure the audio is passed through the PSoC 5LP correctly from the I2S MEMS microphones to the ESP32-S3, the data has to be further offloaded, since the data rate is too high for serial monitor debugging. Furthermore, in the end, the microphones have to be offloaded

from the ESP32-S3 for later training of a machine learning model. This was decided to be done using Wi-Fi, since it allows for the server to be some distance away from the water, where the data collection will be happening. The only modification done to the PSoC 5LP software setup is that the SPI send buffer is filled with samples from the I2S microphones instead of the counter, discarding the 8 zero bits in the end of each sample.

Offloading on the ESP32-S3

The software setup for the ESP32-S3 has to be modified significantly, since Wi-Fi capabilities will be enabled. The ESP32-S3 has two cores and great support for a modified version of FreeRTOS through the ESP-IDF, allowing for utilisation of both cores using FreeRTOS [27]. As little blocking as possible is desired on the ESP32, so UDP is chosen over TCP. This makes the system vulnerable to packets being received in the wrong order, if there are switches with multiple routes between the ESP32 and the server. Since it is assumed that they are connected to Wi-Fi from the same access point, this is deemed not a problem. Packet loss is however still a possibility, based on noise sources in the area and the distance between the ESP32 and the server. This can however be mitigated by being mindful about it on installation. A header on each packet, with a packet number on it, or even a checksum, could make the system identify and communicate problems automatically. This is however deemed unnecessary for the scope of the project. The ESP32-S3 code is described on Figure 4.34.

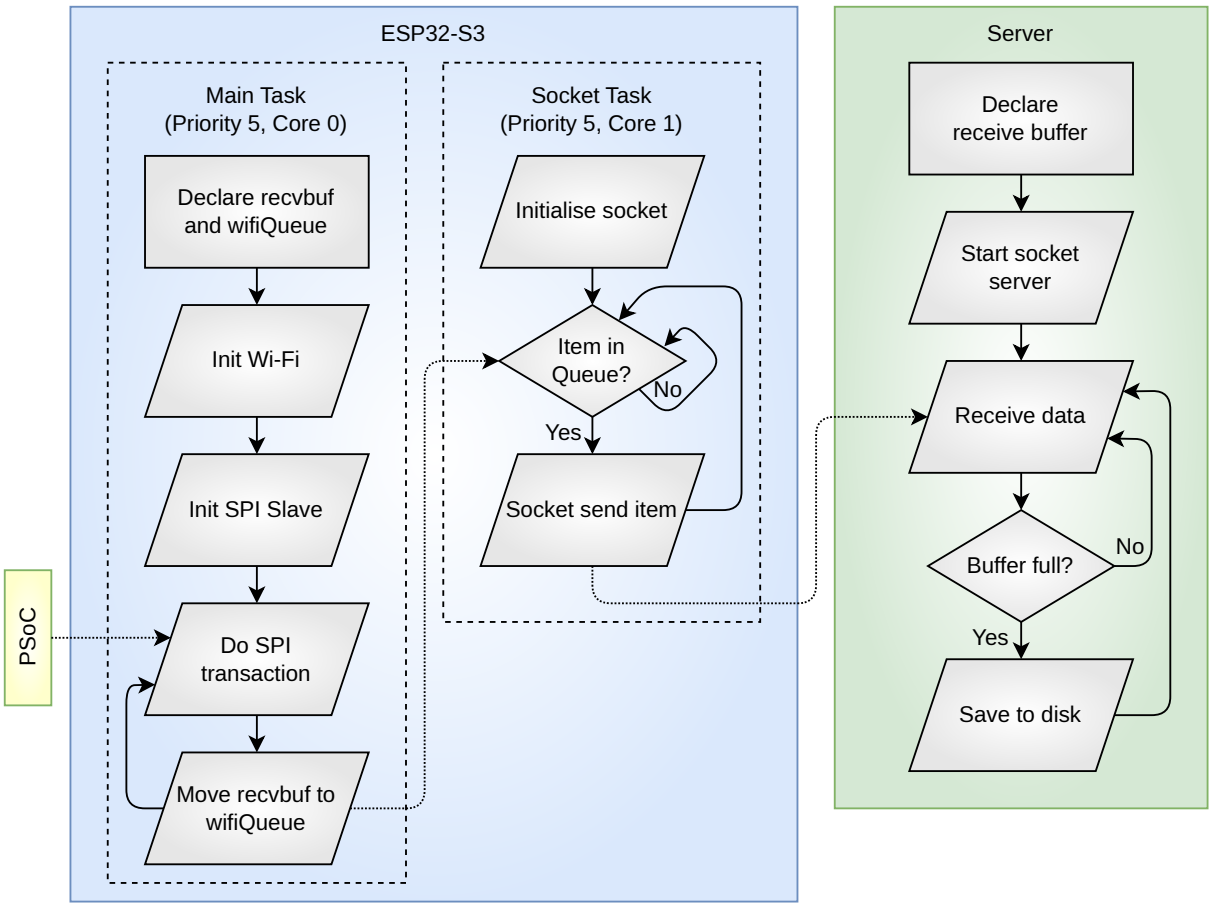


Figure 4.34. Diagram of the test code for the ESP32-S3 showing the FreeRTOS tasks for SPI, sockets for offloading to the server.

In order to test the whole signal path from microphone to Wi-Fi server, a known sine is played from a speaker into the microphones in the anechoic chamber on Fredrik Bajers Vej 7b room 4-111. A journal documenting the measurements is found in Appendix B, but the results are presented here. The anechoic chamber is chosen, in order to eliminate external sources of noise. The recordings can then be visually inspected in the time domain and the frequency domain after the data is saved on the server. The microphones are configured with the distances described in Figure 4.17 from Chapter 4 with the speaker two meters away from the centre microphone (Q_3). A picture of the lineup in the anechoic chamber is seen on Figure 4.35.

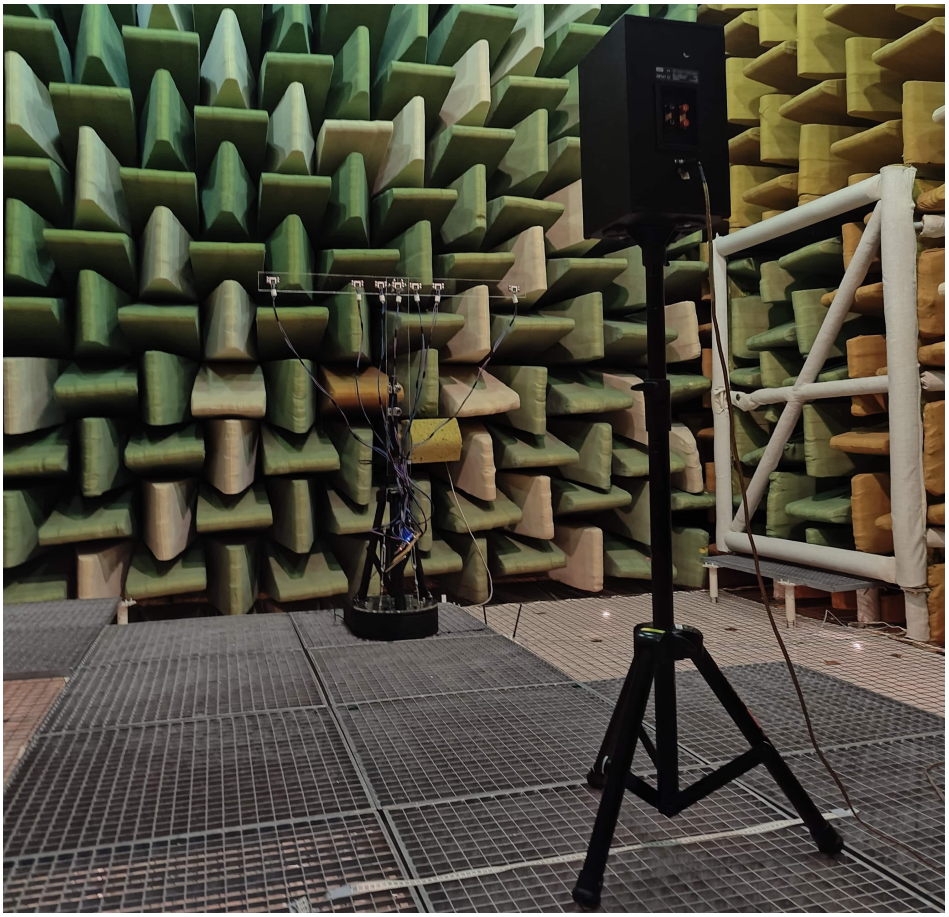


Figure 4.35. The photo shows the microphone array turned to 90 degrees and a speaker directly in front of Q_3, two meters away.

It was immediately observed that all microphones would intermittently adopt capacitor unloading-like behaviour as seen on Figure 4.36 when a pure tone was played.

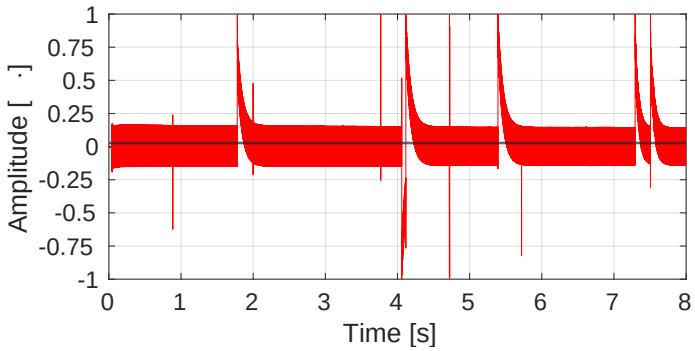


Figure 4.36. Captured audio on Q_0 when a pure 1 kHz tone was played 2 meters away directly in front of it, showing capacitor unloading like behaviour.

A correlation with volume was quickly observed. The distortion looks analogue, so the only analogue part of the signal chain i.e. the I2S microphones was deemed to contain the problem. Since it is unlikely that all purchased microphones were factory defect, it was decided to swap out the power supply as seen on Figure 4.37.

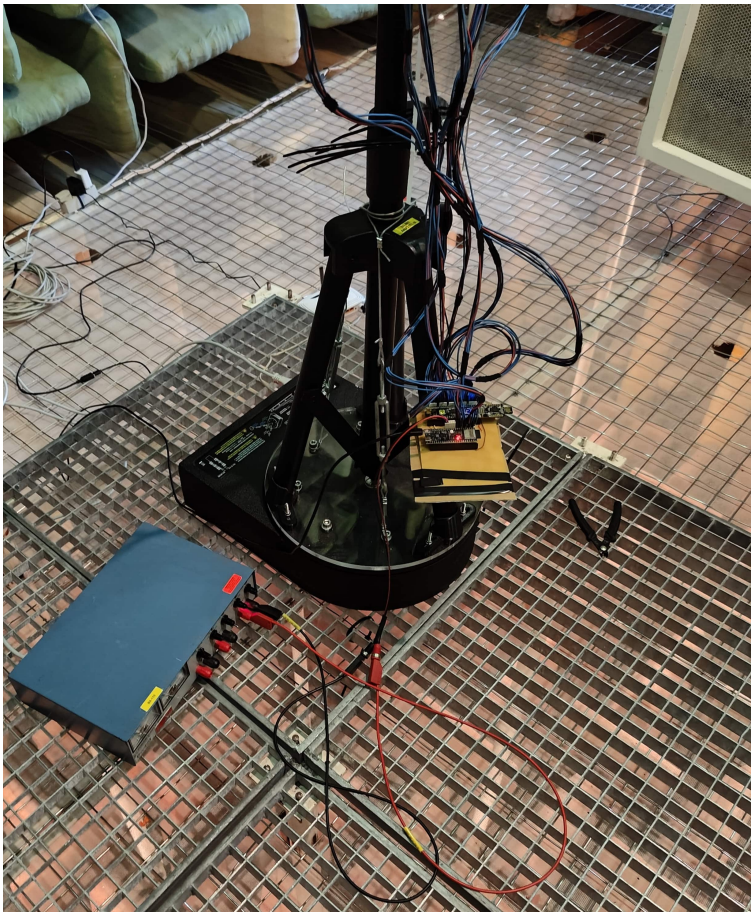


Figure 4.37. Photo of new B&O power supply, connected to the PSoC 5LP, which is supplying the microphone array.

The ESP32-S3 was supplying everything before, so a B&O power supply was used for the PSoC part instead. This fixed the issue seen on Figure 4.38.

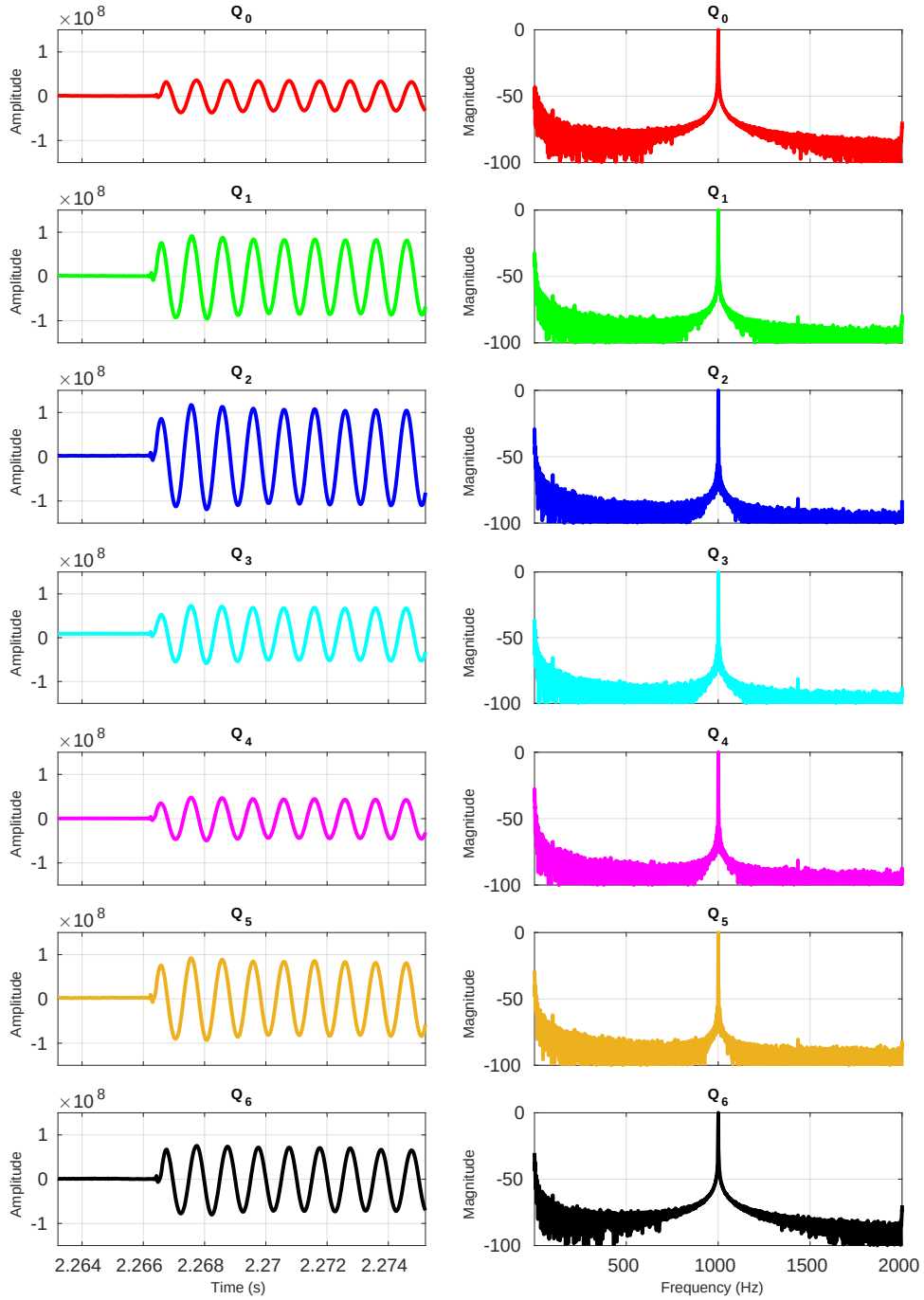


Figure 4.38. Plots of measured audio when a 1 kHz sine is played directly in front of Q_3 two meters away in both time and frequency domain.

Figure 4.38 shows the measured audio when a 1 kHz sine is played directly in front of Q_3 two meters away in both the time and the frequency domain. It is seen that the sines have no artefacts and the frequency spectrum's have a peak at 1 kHz. In Table 4.6 it is seen that the sine arrives first to the three centre microphones and then progressively to the further away microphones.

Table 4.6. Time of peak of first sine period for each microphone subtracted the time of peak for Q_3 in samples.

Microphone	Arrival of Signal [samples]	Expected AOS [continous samples]
Q_0	7	7.86
Q_1	1	0.89
Q_2	0	0.16
Q_3	0	0
Q_4	0	0.16
Q_5	1	0.89
Q_6	7	7.86

The captured measurements show that the SPI setup between the PSoC 5LP and ESP32-S3 is correct, and that the audio quality is good, when the power supply is switched out. It also shows that the wavefront is received at slightly different intervals, which is expected since the microphones are at slightly different lengths away from the speaker. It is however noted, that the distance between the microphones and the speaker should be raised in subsequent testing, such that the wavefront will appear planar.

4.5 Implementation of Delay-and-Sum on ESP32

The implementation of the DAS beamformer designed in Section 4.1 into the ESP32 is described in this section. The beamformer's function is to process the microphone inputs transmitted through SPI from the PSoC. The ESP32 receives 190 24-bit samples from each of the seven microphones every 24 μs which the beamformer implementation needs to filter, delay, and sum. The designed beamformer consists of three beamformers as uniform linear arrays (ULAs) with different microphone spacing in order to have sufficient suppression across the frequency spectrum. As a consequence, the input data must be filtered such only the desired frequencies enter the designated ULA with significant magnitude. The three $M = 3$ ULAs and their respective microphones are seen in Table 4.7.

Table 4.7. List of the microphones for each of the three ULAs used in the DAS beamformer.

	Mic 1	Mic 2	Mic 3
ULA1	Q_6	Q_3	Q_0
ULA2	Q_5	Q_3	Q_1
ULA3	Q_4	Q_3	Q_2

The initial signal path of the microphone data is seen in Figure 4.39. Firstly, the FIR filters are applied to separate the frequencies. Thereafter, the filtered output of each microphone is delayed using a value based on the set steering angle. Finally, the delayed outputs are summed in each of the three beamformers.

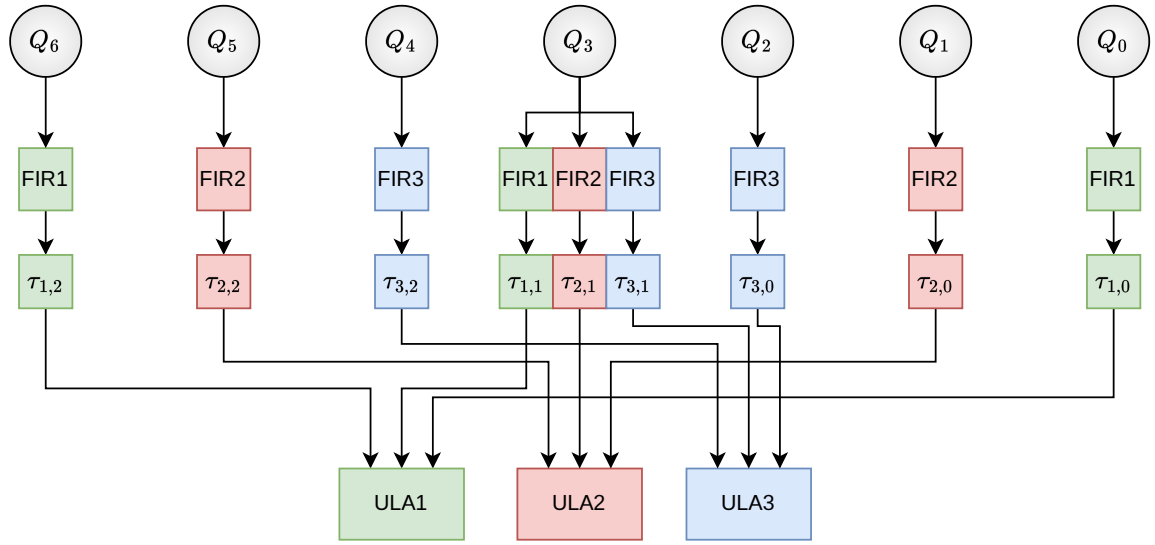


Figure 4.39. Initial signal diagram for applying the beamformer to the microphone signals. The *FIR*-boxes are filters to separate the data in frequency bins, and τ are delays.

The value of the delays are calculated using Eq. (4.6), where the steering angle (θ_s) is set to the desired angle and the time delays are converted into sample delays through the sampling frequency and rounded to nearest integer. The delays (τ) are then implemented by offsetting the index in the data packets. In Section 4.1 it was estimated that neglecting the fractional part of the delays was adequate for this application.

$$\tau_{1,m} = \frac{m\delta_{ULA1}\cos\theta_s}{c}, \qquad \text{where } m = 0, 1, 2 \tag{4.6}$$

$$\tau_{2,m} = \frac{m\delta_{ULA2}\cos\theta_s}{c}, \qquad \text{where } m = 0, 1, 2 \tag{4.7}$$

$$\tau_{3,m} = \frac{m\delta_{ULA3}\cos\theta_s}{c}, \qquad \text{where } m = 0, 1, 2 \tag{4.8}$$

The spacings for the developed beamformer are seen in Table 4.8.

Table 4.8. Spacings between the microphones for the three ULAs in the designed beamformer.

δ_{ULA1}	δ_{ULA2}	δ_{ULA3}
0.500 m	0.333 m	0.071 m

The delays are all relative to Q_0, as it is positioned in the null-position of the microphone array. This means that there is never any offset applied to Q_0 and the sign of the offset of the remaining microphones depends on the steering angle. The correlation between the steering angle and the sign of the offset is shown in Figure 4.40. A positive offset means the index in the microphone's data packet is increased and thereby forwarded in time and oppositely for negative offsets.

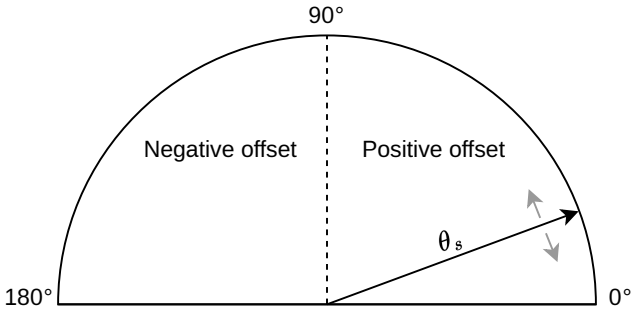


Figure 4.40. Illustration of the correlation between the sign of the index offset in the buffer and the steering angle.

As any offset other than zero will cause the index value to exceed the indices at some point for a data packet, it is required to store some old and new data. Therefore, a circulating buffer is implemented that stores three complete packets from the PSoC at a time. The principle of the buffer and the packets are seen in Figure 4.41. Each of the three segments store seven data packets (one from each microphone), where each of the microphone packets contain 190 samples. Whenever the PSoC transmits new packets, the pointers are incremented and the *new* segment is overwritten with the latest data.

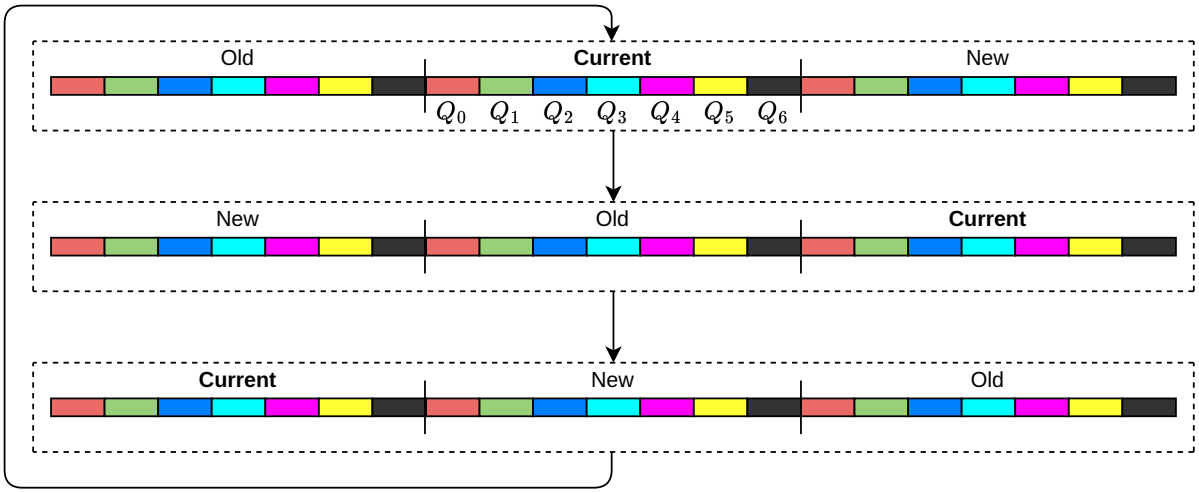


Figure 4.41. Illustration of the circular buffer logic used to always have a new and old packet available to the beamformer. The pointers to the segments are updated on every new packet from the PSoC such the *new* segment is always updated with new data. Each colour represents the 190 samples from each of the microphones.

The logic for managing offsets is seen in Figure 4.42. If the index attempts to move below or beyond the number of samples stored in the *current* segment due to the offset, the index jumps to the *old* or *new* segment of the buffer, respectively.

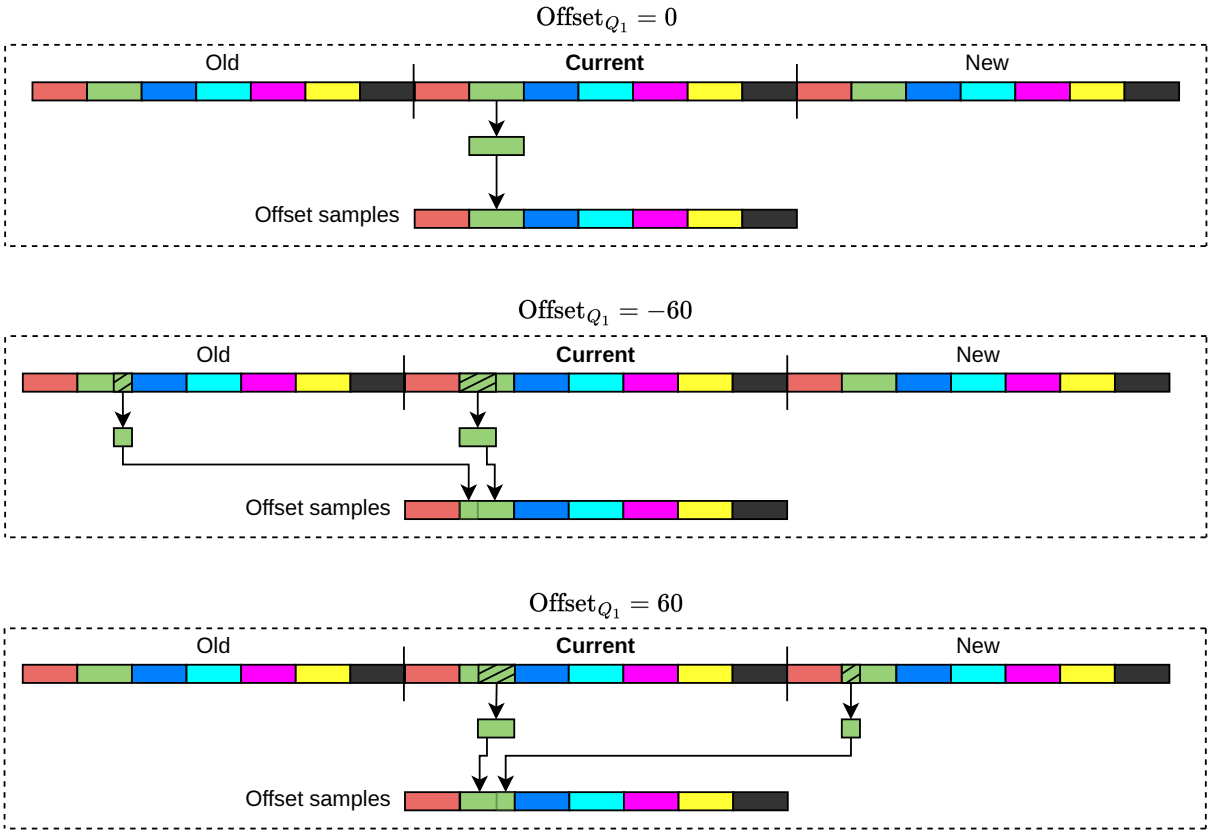


Figure 4.42. Illustration of how the index offsets are implemented into the circular buffer depicted in Figure 4.41. If the offset for the microphone is negative the early samples are taken from the *old* segment of the bufer, whereas a positive offset means the latest samples are taken from the *new* segment.

Each of the 24-bit microphone samples from the PSoC arrives as 3x8-bit (cf. Figure 4.29). It is desired to use the ESP-DSP Library [28] that utilises the Processor Instruction Extensions (PIE) featured in the ESP32-S3 in order to maximise performance. The library supports 16-bit integers for fixed point arithmetic and 32-bit floats for floating point arithmetic [28]. Since using 16-bit integers would require a third of each sample be thrown away i.e. significantly reducing the SNR, of which the goal is to improve, it is chosen to use 32-bit floats. If the end-user, who will use the data for model training, later deem 16-bits of resolution desirable, resolution can always be thrown away later. Since 32-bit floats are chosen, the 3x8 bits are stored as floats with the 8 LSBs set to zeroes.

The filter weights for the FIR filters are calculated in MATLAB using *genFirFilters.mlx* from Appendix A and automatically exported to a header file containing the variable declarations.

The distributive property of convolution is taken advantage of in order to reduce the number of filter applications from nine to three. The updated signal diagram for the beamformer is seen in Figure 4.43.

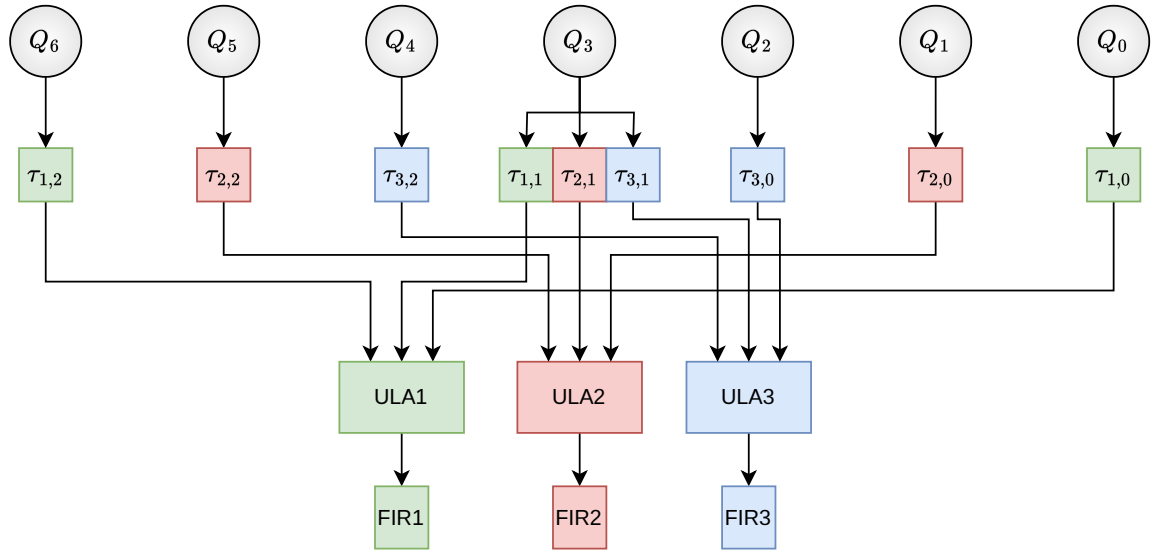


Figure 4.43. Updated signal diagram for the beamformer. The FIR filters are moved to the other side of the ULAs in order to reduce the number of operations.

With the theoretical design done, implementation on the ESP32 can commence. The ESP-DSP library provides three 32-bit floating point FIR filter implementations called `dsps_fir_f32` with extensions `_ansi`, `_ae32` and `_aes3`. The `_ansi` extension uses ANSI C and could be compiled and run on any platform, whereas `_ae32` is optimised for the base model ESP32 and `_aes3` is optimised for the ESP32-S3 [28]. Before using the filter functions, the FIR structure has to be initialised using `dsps_fir_init_f32` with pointers to a pre-allocated FIR filter structure of type `fir_f32_t`, the N coefficients, a delay line of length N and finally the FIR filter length N, which specifically for the ESP32-S3 has to be divisible by 4 and aligned to 16. Aligning to 16 is done by setting `__attribute__((aligned(16)))` when initialising the delay line, making sure the optimised SIMD (Single Instruction, Multiple Data) instructions will function properly, since crossing memory boundaries will cause issues, as 16 bytes are accessed at the same time [29].

The `_ansi` and `_aes3` extensions are tested using `dsp_get_cpu_cycle_count` in order to verify the alleged performance gains from using `_aes3`. When using `_ansi` for a filter with 60 taps on 190 samples, 215 029 CPU clocks were used. For the same coefficients and samples `aes3` uses only 28 058 CPU clocks, which is a significant reduction and on par with the benchmarks from the documentation [28]. The two cores on the ESP32-S3 both run at 240 MHz [25]. The microphones have a sample rate of 41 667 Hz, while 190 samples per microphone are sent at a time from the PSoC to the ESP32. This results in 219 transmissions per second, or just over 1 million clocks per core per transmission when not factoring overhead. Since the 28 058 or 215 029 clocks are per filter, it would likely not have been feasible to use the non optimised implementation of the FIR filter on the ESP32. The software implementation of Delay-and-Sum is seen on Figure 4.44.

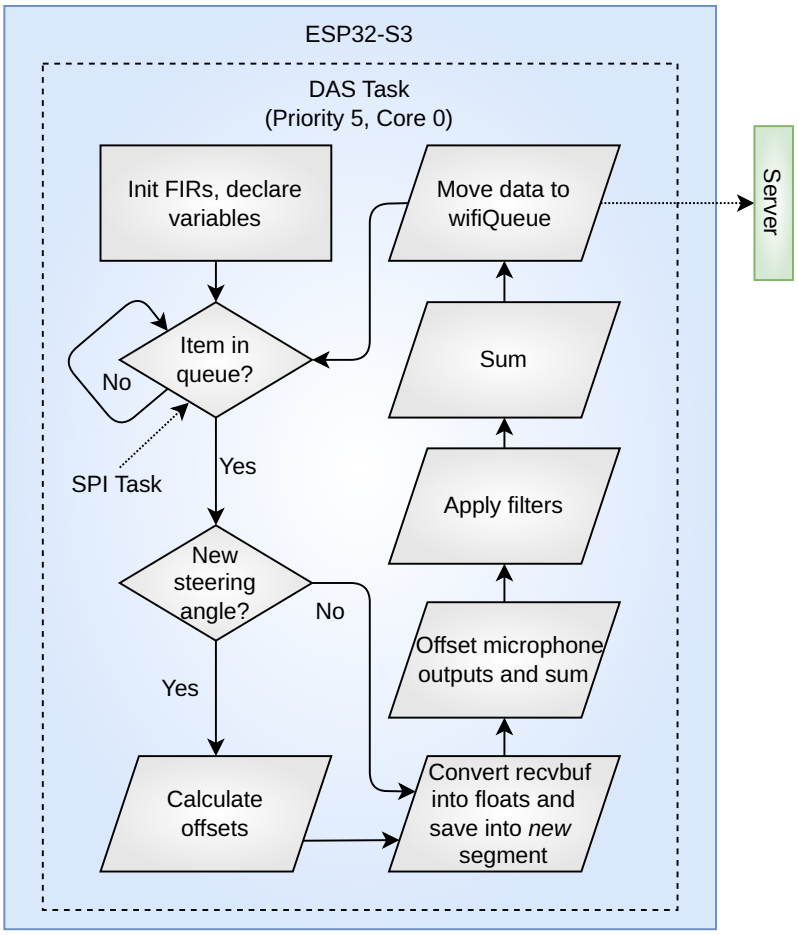


Figure 4.44. Flow diagram of the Delay-and-Sum software running on the ESP32.

With all tasks configured, the final program is found in */SplashDetect/main/** in Appendix A and depicted on Figure 4.45. Since the *SPI Slave Task* and the *DAS Task* will never run at the same time, they are allocated to the same core. Since it cannot be guaranteed that the *Socket Task* is finished offloading when the next SPI transaction is ready, it is allocated a core of its own.

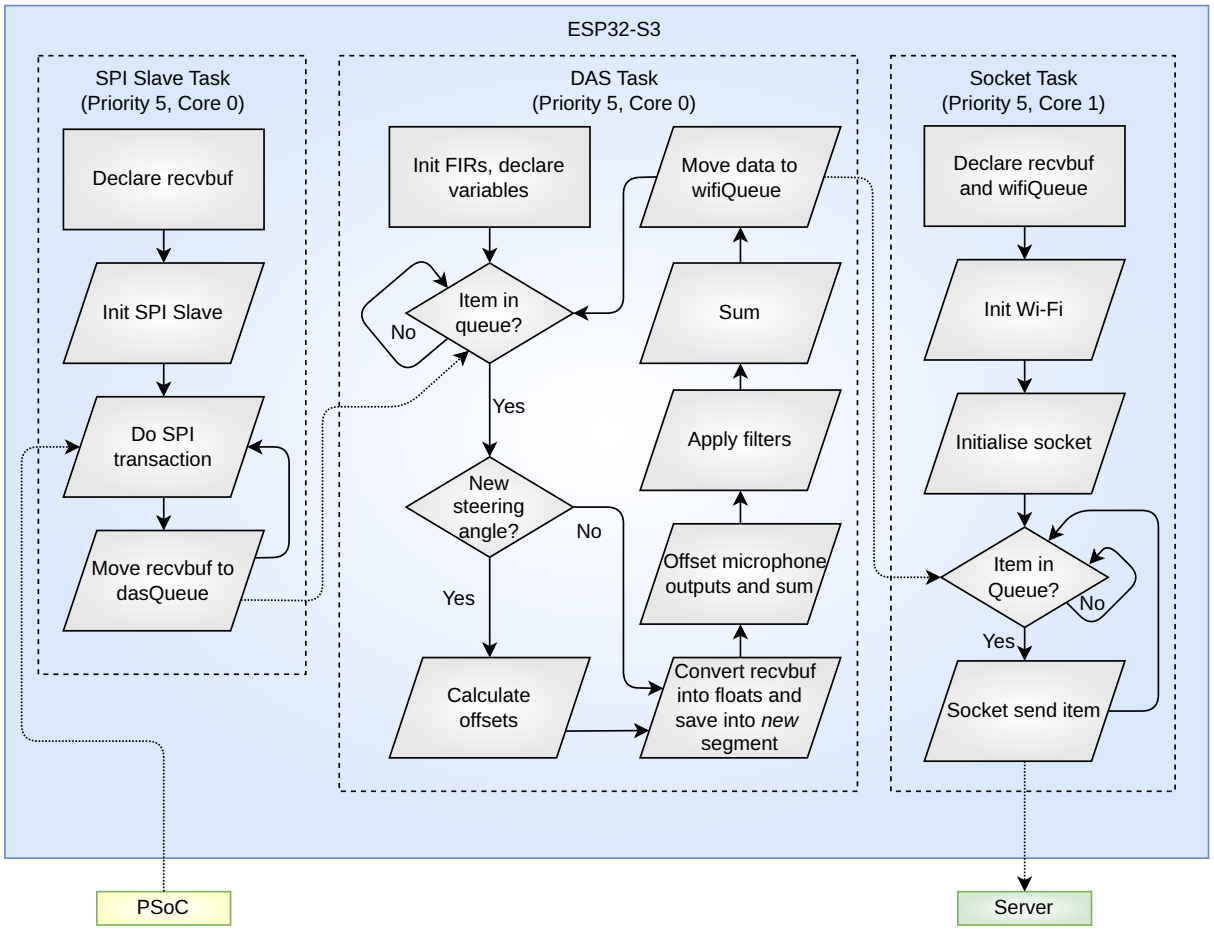


Figure 4.45. Flow diagram of all three tasks running on the ESP32-S3 using the ESP-IDF modified version of FreeRTOS. Corresponding code is found in *SplashDetect/main/** in Appendix A.

The implementation of the beamformer is finished, but has to be validated.

4.6 Validation of Beamformer Implementation

The journal of the validation of the implementation of delay-and-sum is found in Appendix C. However, a walkthrough of the journal is included here. The purpose of the journal is to compare the beampattern of the implementation to a simulated equivalent and thereby validate the implementation. The journal contains two parts: **Microphone Calibration** and **Beamformer Measurements**.

The beamformer designed in Section 4.1 was designed with underwater usage in mind, meaning the speed of sound was set to 1496 $\frac{\text{m}}{\text{s}}$ in the calculations. However, the tests in Appendix C are conducted in air, as this significantly reduces the complexity of the setup and no underwater anechoic chamber was available at the time of measuring. Therefore, in order to have a comparison for the implementation, the simulations from Section 4.1 are redone with the speed of sound in air (343 $\frac{\text{m}}{\text{s}}$).

In order to extract the beampattern of the beamformer, the system's response to an impulse at a range of incidence angles must be measured. However, generating an auditory impulse that is always identical is a challenge.

4.6.1 Excitation Signal

A method for exciting the system and extracting the frequency (or impulse) response must be chosen. Multiple methods for finding the impulse of a linear time-invariant system is presented.

Maximum Length Sequence (MLS)

The **MLS**-method stimulates the system with a pseudo-random binary sequence. The impulse response is then extracted by performing Fast Hadamard Transform (FHT) on the output and analysing the circular cross-correlation between the in- and output. The procedure for **MLS** is illustrated in Figure 4.46.

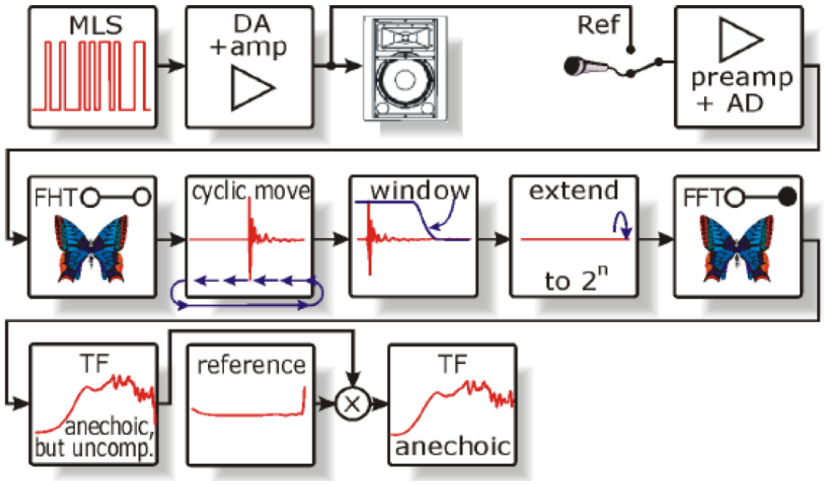


Figure 4.46. Illustration of the steps in the **MLS**-method. The output of the system is transformed with the Fast Hadamard Transform (FHT), moved, windowed, extended, FFTed, and multiplied with a reference [30].

Exponential Swept Sine (ESS)

Another option for estimating the impulse response is **ESS**. The excitation signal of the **ESS** is a logarithmic frequency sweep with more energy in the lower frequencies. Due to this energy discrepancy, the output of the system is adjusted by dividing the FFT of the output with the FFT of the excitation signal. The procedure for **ESS** is shown in Figure 4.47.

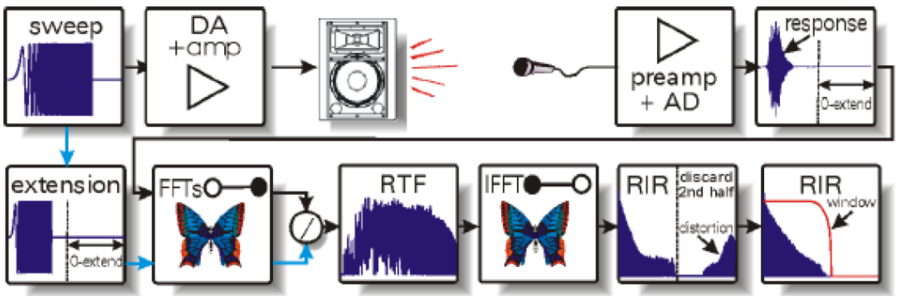


Figure 4.47. Illustration of the steps in the ESS method. The output is FFTed and divided by the FFT of the exictation signal in order to account for the energy discrepancy of the input signal [30].

Impulse by Object (IO)

Another option is to generate an impulse with an object such as a movie clipboard or a simple clap from a person. However, replicating the exact same sound is challenging without a sophisticated setup or dedicated tool.

Single Tones at a Time (STT)

A more naive approach is to play a single tone or frequency range at a time, measure the output, and calculate the power. By measuring the power for some list of chosen frequencies and input angles, the frequency response of the system can be estimated.

Choice of Excitation Signal

It is decided to generate the beampattern through the ESS-method, as the number of steps in the procedure is the smallest. Only the frequency response is of interest, so the last three steps in the ESS-method can be disregarded (IFFT, discard, and window).

If the beampattern from **ESS** is unrecognisable or too noisy, the **STT**-method is conducted in an attempt to generate a *prettier* plot, as noise on an **ESS** measurement will result in noise across the frequency spectrum, whereas noise on an **STT** only results in noise in the single measurement.

Test Setup

The wiring diagram of the equipment used in the measurements is illustrated in Figure 4.48.

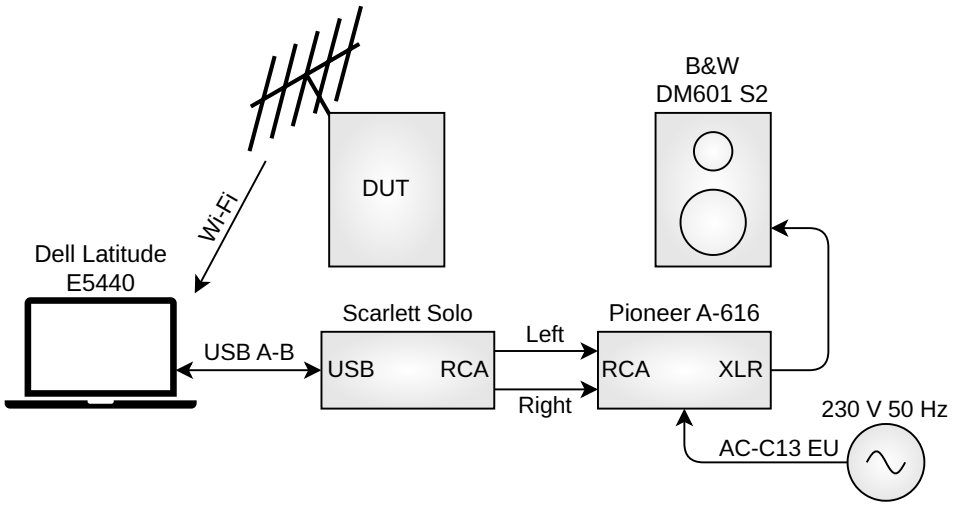


Figure 4.48. The figure shows the wiring of the equipment for the implementation tests.

A picture of the lineup is shown in Figure 4.49. The microphone array is placed upside down in the turntable, meaning the steering angle is rotated around the pointing direction. The array is pointed towards the speaker with a distance of 5.35 m to the centre of the array to the centre of the speaker. The microphones are positioned 1.42 m from the grates and so is the vertical centre of the speaker. The distance between the speaker and microphones are made as large as possible to make the wavefronts as planar as possible.

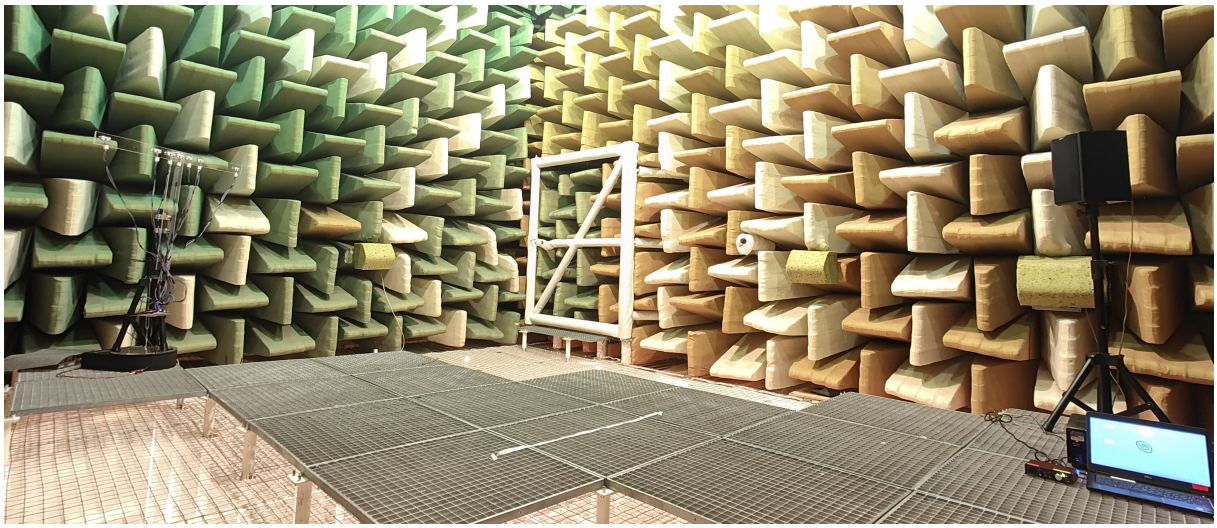


Figure 4.49. The photo shows how the microphones and speaker are setup in the anechoic chamber. A Rockwool sheet was placed against the amplifier and laptop during tests in order to minimise noise influence.

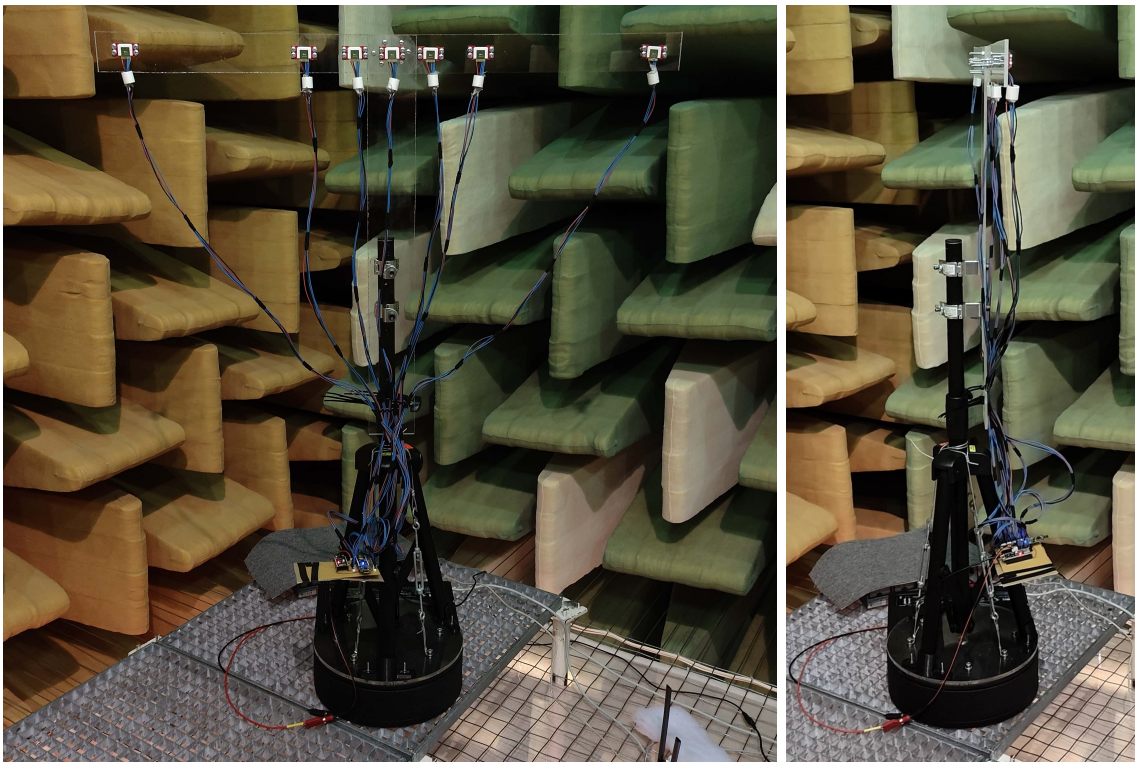


Figure 4.50. The photo shows the microphone array turned to 90 degrees on the left and 180 degrees on the right. Both are taken in the anechoic chamber.

4.6.2 Procedures

The procedures for performing the calibration of the microphones and the beampattern measurements are listed in this subsection together with flowcharts for the data capture algorithms.

Microphone Calibration

In order for the beamformer to function as intended, the sensitivity of the microphones in the array must be similar. The software procedure for data capture of the calibration is shown in Figure 4.51.

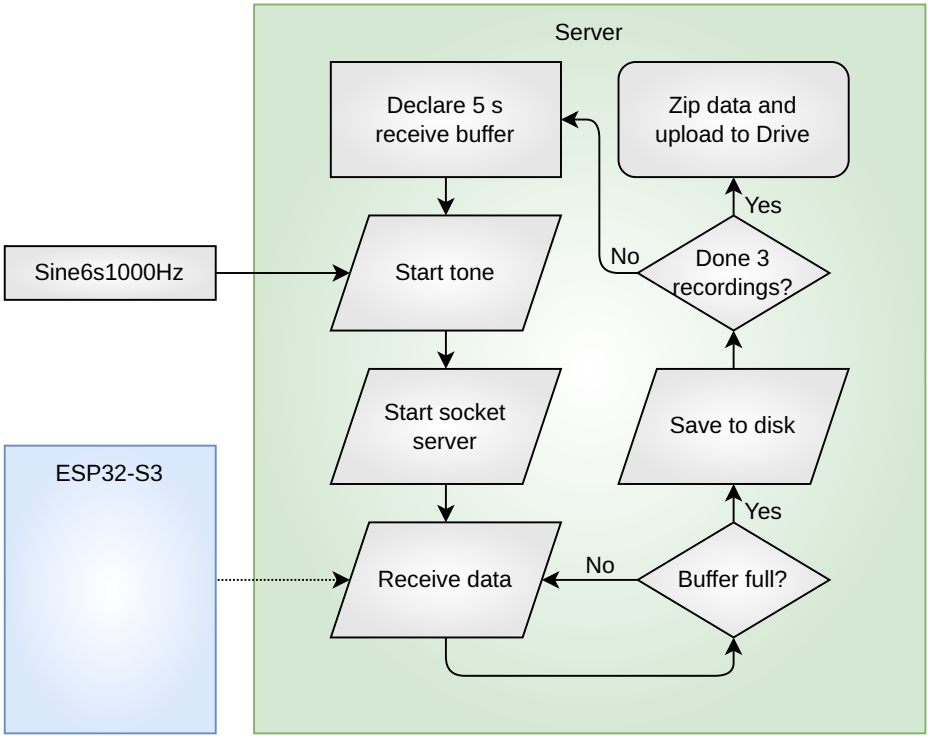


Figure 4.51. Flowchart of the data capture for microphone calibration, where a 6 second 1000 Hz tone is recorded three times.

The received data is then processed to find the mean of the data and the power. From the power, gain factors for each of the microphones are determined in order to equalise the gain of the microphones. The procedure in Figure 4.51 is then repeated in order to verify the calibration.

Beamformer Measurements - ESS

Once the microphones have been calibrated to equal gain, the beampattern can be estimated using the **ESS**-method. The excitation signal is generated using the *sweeptone*-function in MATLAB. The signal sweeps from 20 Hz to 20.8 kHz over 20 seconds starting at 2 seconds. The maximum frequency has changed from 24 kHz to 20.8 kHz due to change of sampling rate in Section 4.3. Additionally, a single period of a 1 kHz signal is inserted at 1 second for synchronisation purposes in post-processing. The excitation signal for the **ESS**-method is seen in Figure 4.52.

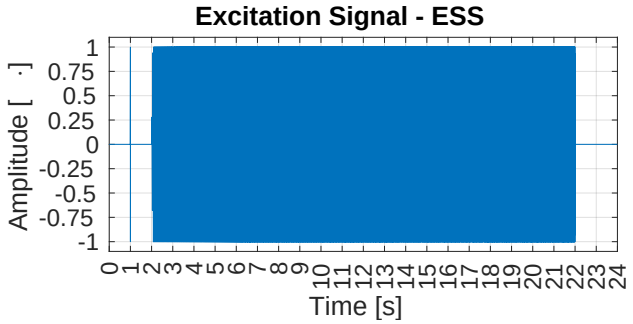


Figure 4.52. Excitation signal for the **ESS**-method. The signal sweeps logarithmically from 20 Hz to 20.8 kHz over 20 seconds, but contains 2 seconds of silence in the beginning and end and a single period of a 1 kHz signal for synchronisation at 1 second.

The procedure for the data capture for the **ESS**-method is shown in Figure 4.53. Three sweeps are done in order to suppress noise.

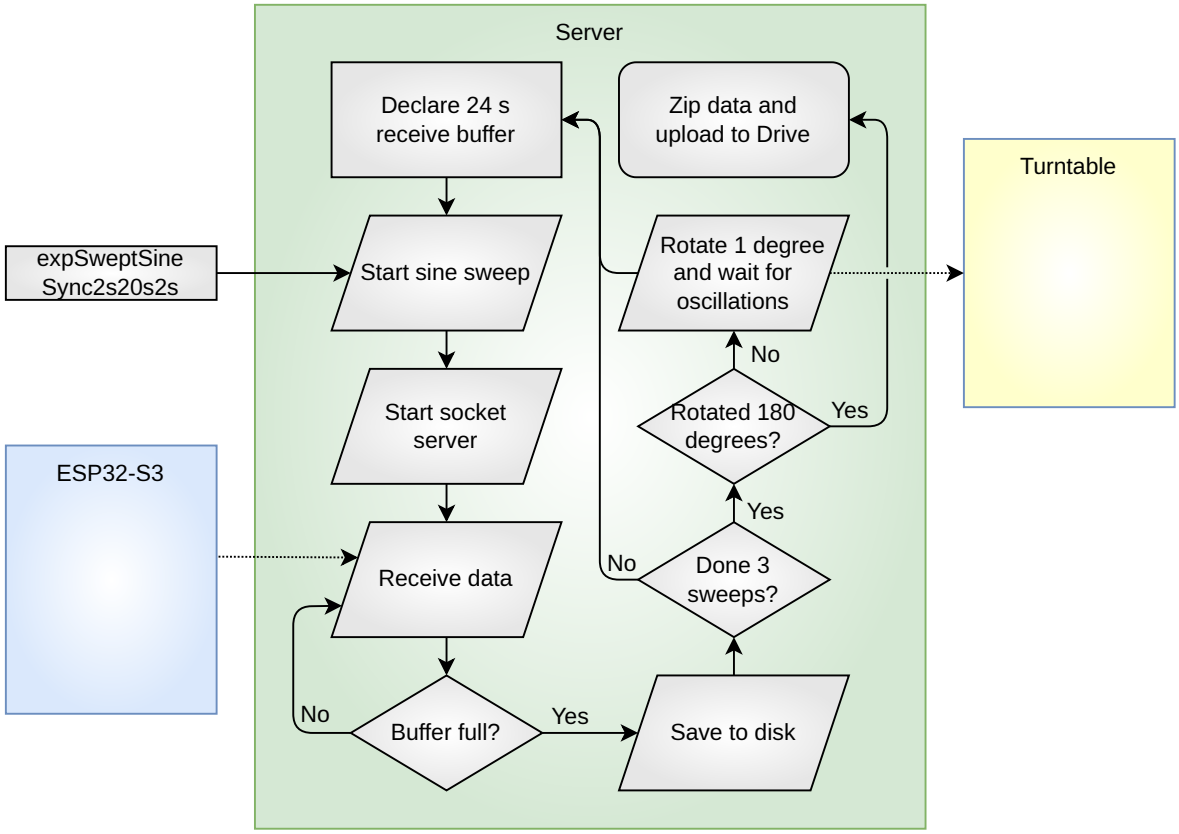


Figure 4.53. Flowchart of the data capture for the **ESS**-method, where a 24 second sine sweep is recorded. The turntable is rotated 1 degree following the sweep with a delay to allow oscillations to cease.

The output data is then processed cf. Figure 4.47 to estimate the frequency response of the system.

Beamformer Measurements - STT

For the **STT**-method, a list of single-tone signals are generated with a resolution of 200 Hz in the span of 50 Hz to 20.8 kHz and a length of 4 seconds. The procedure for the data capture is illustrated in Figure 4.54. Three recordings are done in order to suppress noise.

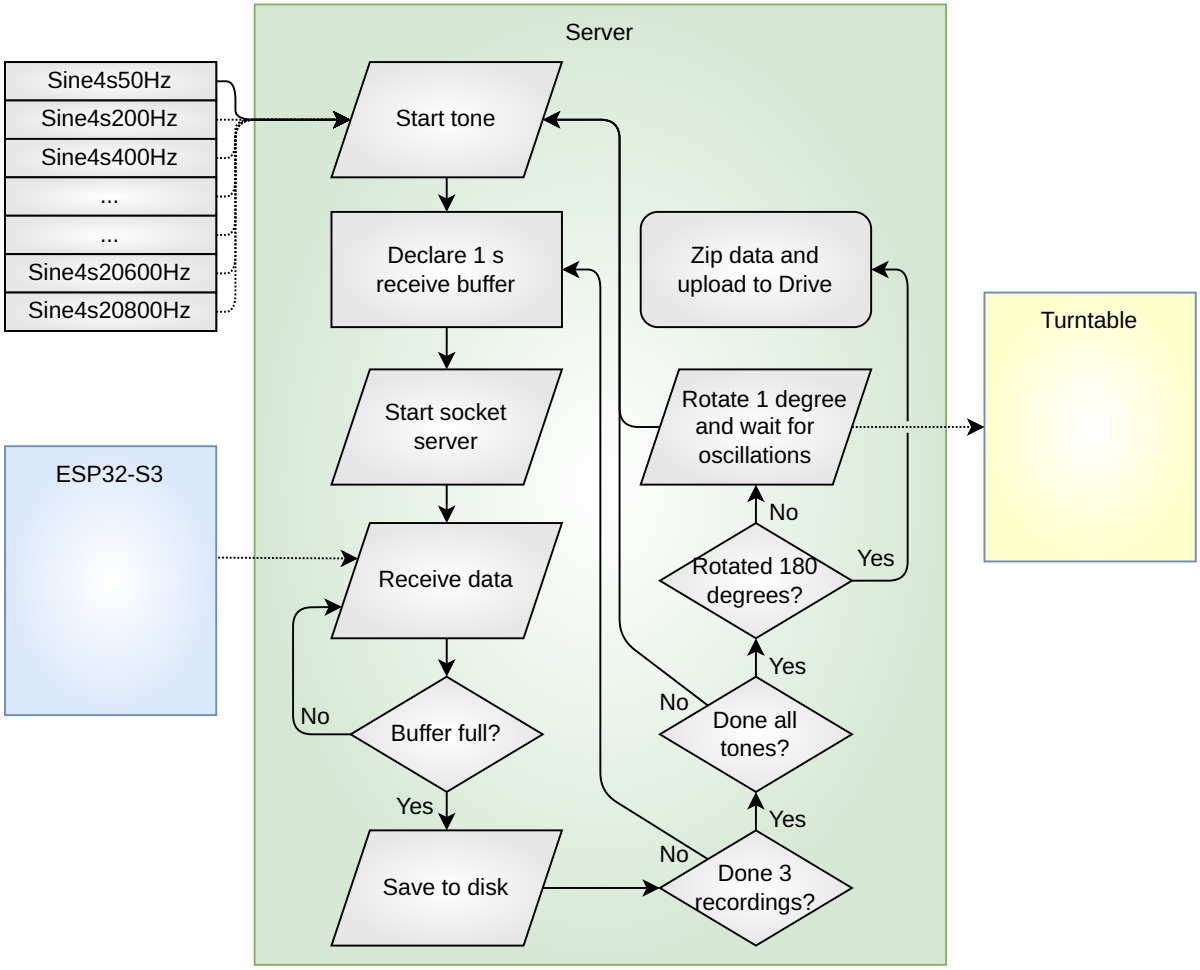


Figure 4.54. Flowchart of the data capture for the **STT**-method, where three 1 s recordings are made for each tone. The turntable is rotated 1 degree following the rotation with a delay to allow oscillations to cease.

The power and mean of each of the recordings are calculated and logged.

4.6.3 Results

The results of the measurements are presented and discussed in this subsection.

Microphone Calibration

The three raw measurements for Q_4 during the first part of calibration are plotted in Figure 4.55.

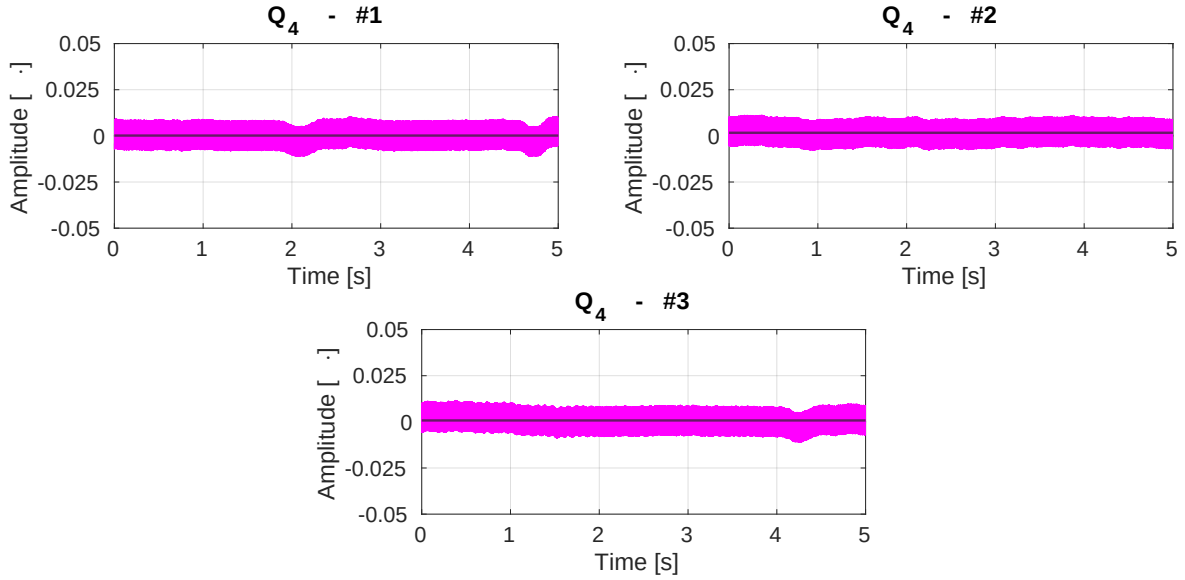


Figure 4.55. Plot of the measured data before calibration for Q_4. The line shows the mean value of the data.

The expected shape of the plots is a rectangle, as the amplitude and frequency of the input signal are constant. However, variations in the amplitude occur for many of the microphones that look like low-frequency noise. The exact reason for the variations is unknown but could be due to an oscillatory effect happening within the microphones, as it exposed to a single tone.

The measured mean and power of the microphone data for each of the three runs are seen in Table C.4 (in the journal) together with an average of the runs and a gain factor in order to equalise the power to the microphone to the highest power (Q_2). As the means of the measurements are not constant and not too significant, it is decided to disregard the slight DC-offset. The same procedure is repeated but with the gain factor applied to the microphones within the ESP32. The measurements are shown below.

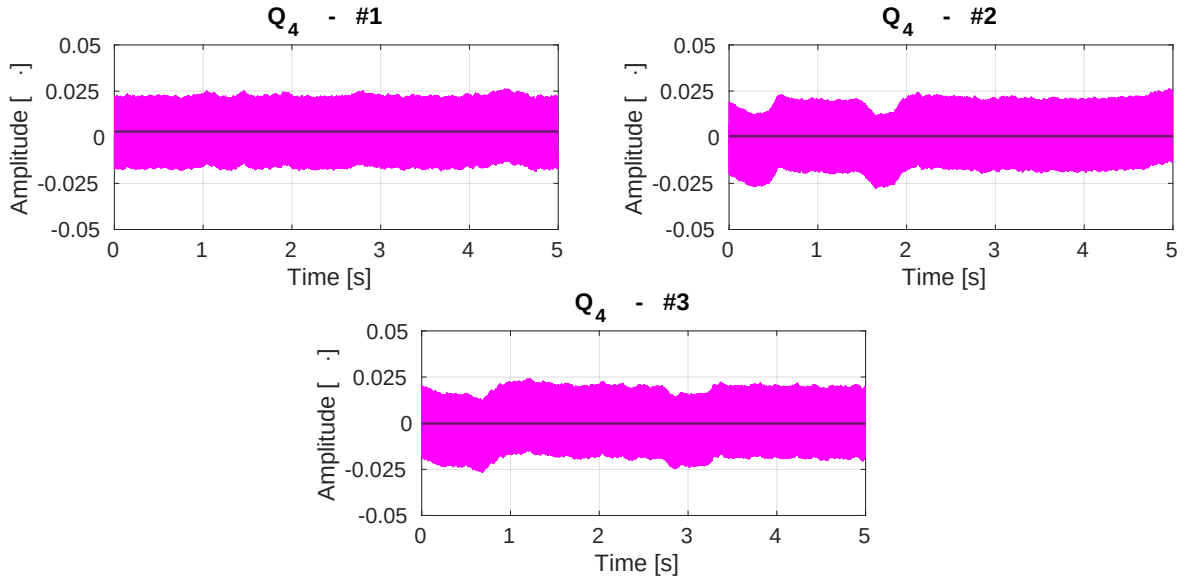


Figure 4.56. Plot of the measured data post-calibration to match Q_2's gain for Q_4. The line shows the mean value of the data.

The power of the outputs of the microphones are now more similar, as desired.

Beamformer Measurements - ESS

The measurements of the ESS-method are processed with *systemIImpulseResponse.mlx* that estimates the frequency response for each of the input angles. Firstly, cross-correlation is performed on the excitation signal and the output on the data between 0.5 and 1.5 seconds in order to find an alignment such the exact beginning of the sweep in the output signal can be found. Thereafter, the frequency response is estimated with the procedure denoted in Figure 4.47. Finally, beampattern plots for chosen frequencies and broadband are generated from the average of the three measurements together with a plot showing the magnitude in the steering angle.

Additionally, equivalent (speed of sound) simulations are generated with *systemSim.mlx* in order to have a comparison for the measurements. The resulting beampatterns from a measurement through the ESS-method and a steering angle set to 90/270° are seen in Figure 4.57.

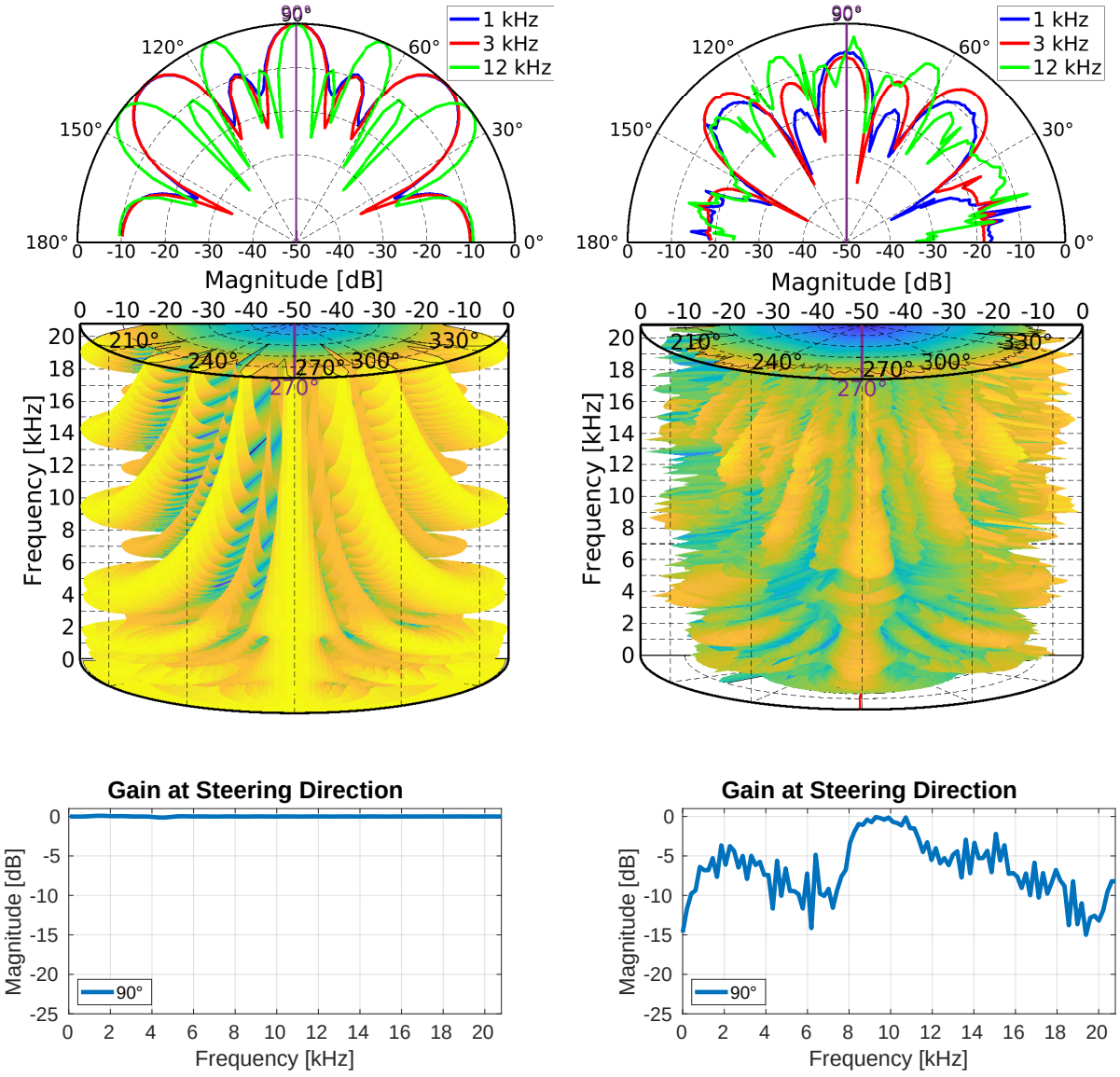


Figure 4.57. Comparison plots for the results of the ESS-method for the DAS implementation (right) and a simulated equivalent (left). The steering angle is set to 90/270°.

The beampattern clearly bears resemblance to the simulated but with significant noise and varying suppression at the steering direction. The same measurement is conducted but with a steering angle set to 130/230° in order to examine if the beamformer can steer the main lobe. The processed results are seen in Figure 4.58.

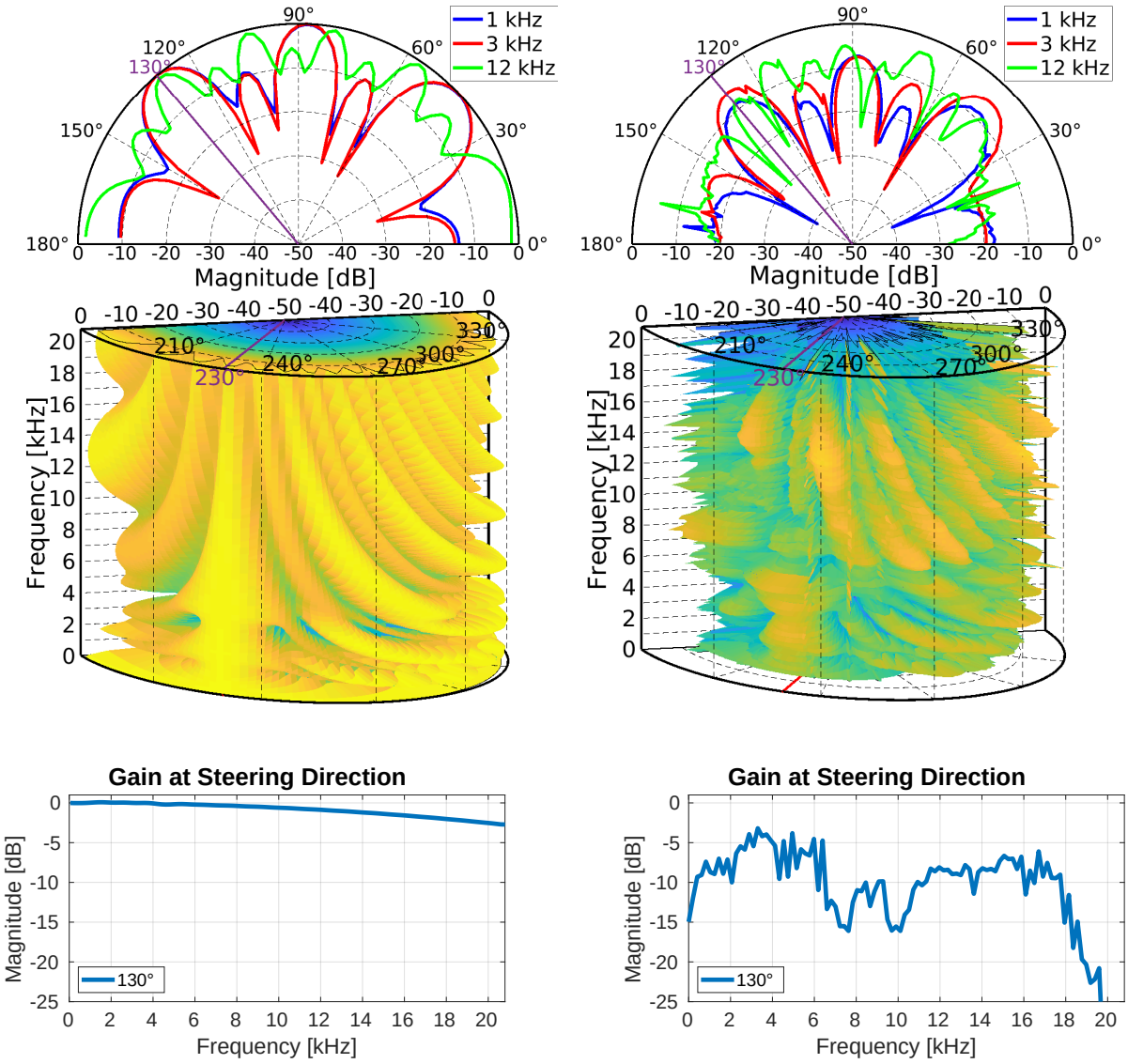


Figure 4.58. Comparison plots for the results of the ESS-method for the DAS implementation (right) and a simulated equivalent (left). The steering angle is set to 130/230°.

The beamformer can clearly steer the main lobe in the same manner as the simulated with the same shape. However, the noise is still significant. Therefore, the STT-method is attempted, as it should be less susceptible to noise. The STT consists of many more smaller measurements to make the frequency sweep rather than a single measurement.

Beamformer Measurements - STT

The measurements of the STT-method are processed with *naiveSweepFrequencyResponse.mlx* that calculates the average power for each of the measured frequencies and input angles. The same plots as ESS are then generated.

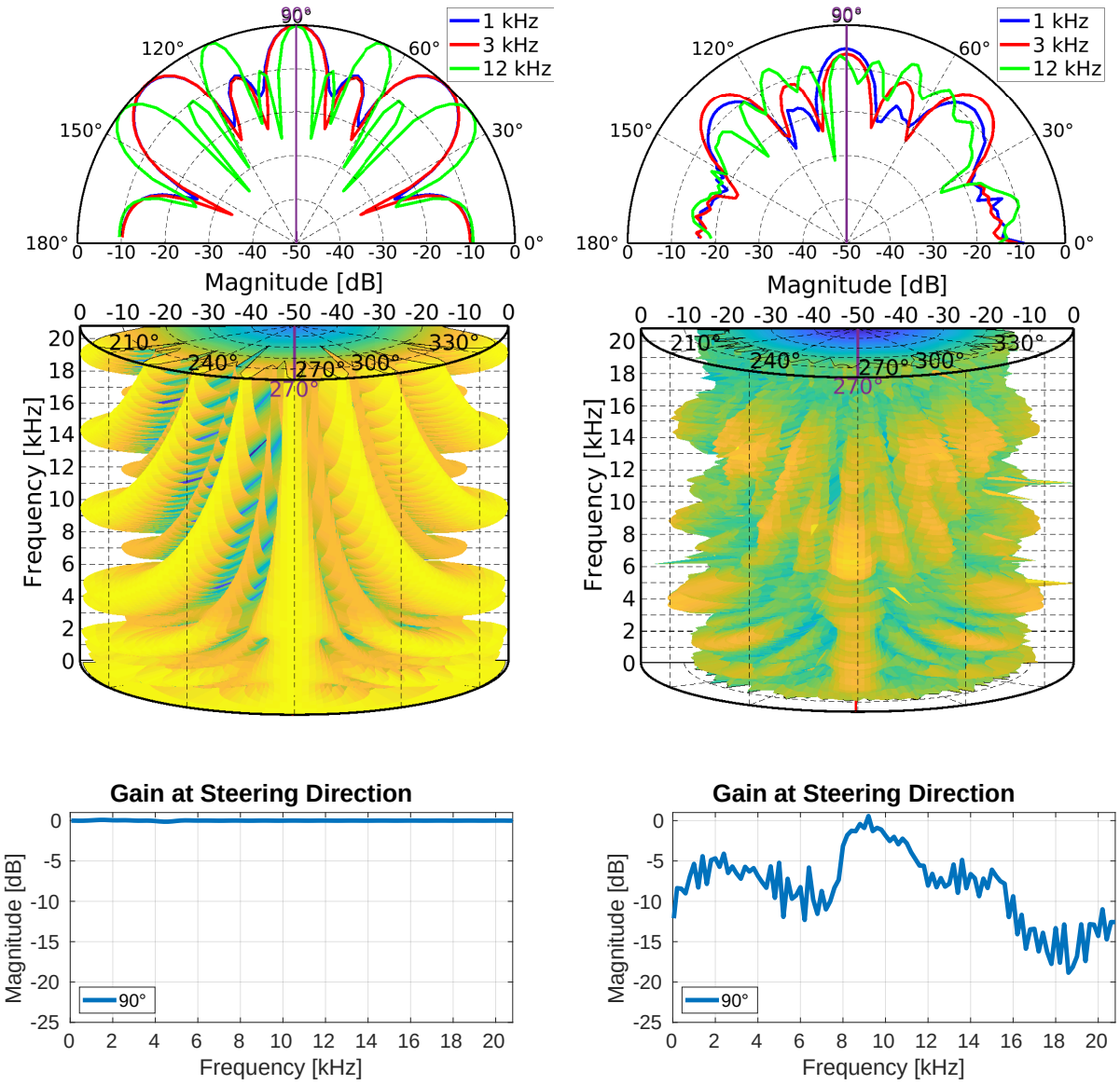
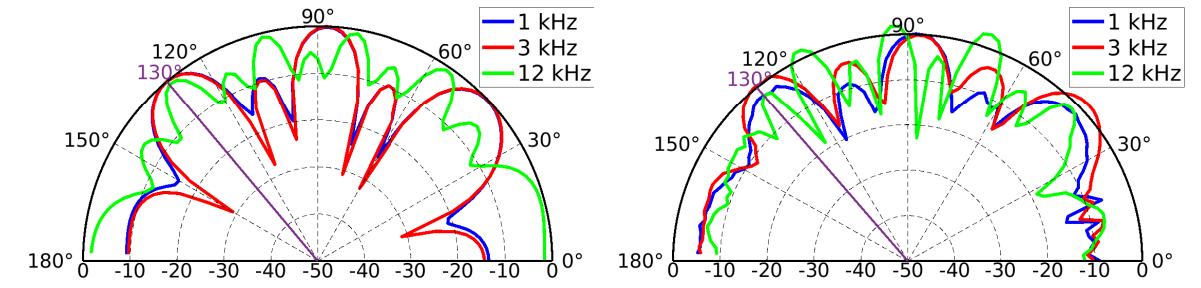


Figure 4.59. Comparison plots for the results of the STT-method for the DAS implementation (right) and a simulated equivalent (left). The steering angle is set to 90/270°.

The significance of the noise has been reduced, but spikes are still noticeable and the magnitude in the steering angle is still varying which might be a combination of the frequency response of the speaker and microphones. The same measurement is conducted but with a steering angle set to 130/230°. The results of the measurements are seen in Figure 4.58.



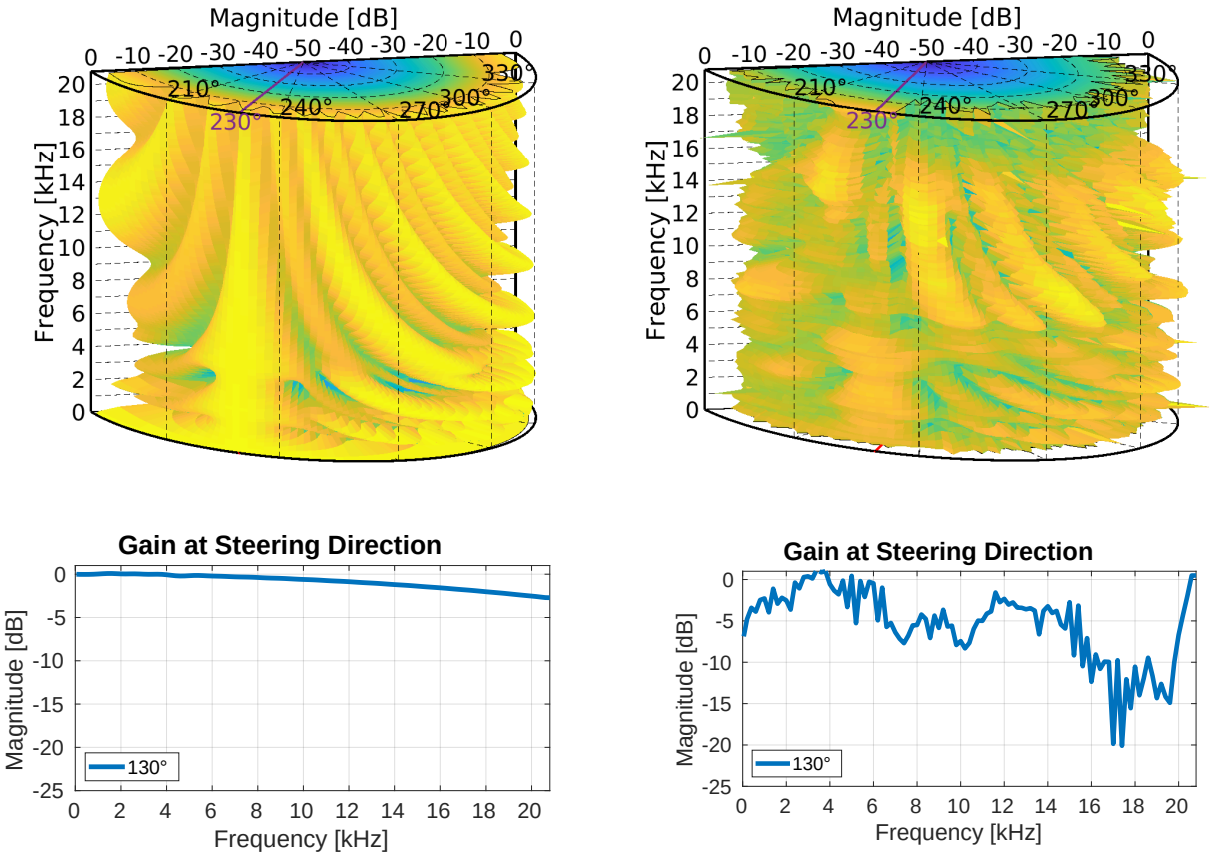


Figure 4.60. Comparison plots for the results of the STT-method for the DAS implementation (right) and a simulated equivalent (left). The steering angle is set to 130/230°.

The frequency response of the setup, meaning the combined frequency response of the signal chain from the PC to the centre microphone (Q_3) is measured in order to compare it to the gain at the main lobe. The measured frequency response of the DUT and test equipment compared to the gain at the steering angle from Figure 4.57 is seen in Figure 4.61.

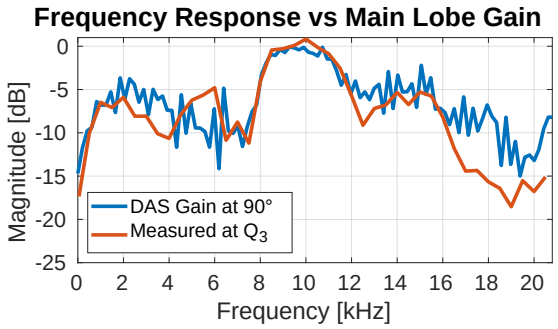


Figure 4.61. Measured frequency response for the signal chain (laptop to microphone) measured with Q_3 vs. the gain at the steering direction from Figure 4.57.

The frequency responses clearly share the same trend, meaning the varying gain in the steering direction and difference to the simulated are most likely due to the fact that the speaker and microphones were not frequency equalised.

Conclusion

The microphones were successfully calibrated to equal sensitivity at 1 kHz. However, some low-frequency noise was evident in the measurements that remained unexplained.

The beampattern of the system was first estimated using an exponential sine sweep with steering angles set to 90/270° and 130/230°. The beampatterns estimated through the measurements turned out similar to the simulated, but influenced by noise and the frequency response of the speaker and microphones.

The same beampatterns were generated by playing a single tone at a time and measuring the power rather than performing a sweep. This reduced the influence of the noise by a margin and made the beampatterns a bit more identical to the simulated.

However, the gain at the steering direction was not constant, as the frequency response of the speaker and microphones were unaccounted for. But the implementation of the delay-and-sum beamformer is validated successfully.

4.7 Direction of Arrival and Trigger

In Section 3.3 it was decided to estimate the direction of arrival (DOA) through time difference of arrival (TDOA) and estimate whether the beamformed signal is impulse-like through the PSD or power check in frequency bins. However, due to prioritisation of time, the aforementioned are not implemented into the system.

Discussion 5

The purpose of this chapter is to evaluate and discuss the implementation choices and findings of the preceding chapters. Firstly, the project direction, requirements, and use-case derived from the analysis are discussed. Thereafter, the choice of development platform is assessed. The decisions revolving beamforming techniques are then discussed, whereafter the methods and measured beampatterns are evaluated.

The analysis in Chapter 2 culminated in a list of functional and technical requirements, where the functional encompass the desired functionality of the system and the technical encompass specifications for the components of the system. The functional requirements provided a template for the design and implementation, but the technical requirements were primarily only relevant for hardware selections, as they were set before any design decisions had been proposed. It would have been preferable if another set of requirements for the methods had been established containing requirements such as tolerance for direction of arrival estimation, maximum magnitude variation in beamformer's steering angle, suppression in every other direction than the steering, transmission data rates, etc. This would ease some design decisions and validation if specifications with corresponding acceptance tests were predetermined. However, design philosophies were introduced and explained during the design process sharing the same principle.

The choice to use the PSoC 5LP as a prototype for sampling hardware implicitly chooses I2S microphones, since it, just as the ESP32-S3, does not have the required ADCs for direct sampling. There is inherently nothing wrong with I2S microphones, but they significantly reduce the available choices of microphones and result in some locked choices, that then have to be worked around. The chosen I2S microphones were locked to a 24-bit data size with 18-bit precision, but with a 32-bit word size. This meant that the data had to be repacked in order to not transmit informationless data to the ESP32-S3. Other than this, problems with the PSoC clock configuration and the intricacies of DMA on the PSoC resulted in significant development time, which could just as well have been spent on the development of custom sampling hardware.

The ESP32-S3 turned out to be a fine choice, with some limitations. It is a power efficient development board, even without the use of *Deep-Sleep* mode and the processing power was sufficient for the use-case. There was a wish to use the *ULP Coprocessor* for extra power efficiency, but there was no obvious way of doing this. The *ULP Coprocessor* has access to RTC GPIO, RTC ADC and RTC I2C in *Deep-Sleep*, so it might be possible to forward low resolution audio signals to it, by use of custom hardware. The *ULP* would then be able to wake the main CPU for beamforming, but this is merely speculation. Two cores allowed for true concurrency as Figure 4.45 shows. This ended up being very useful, as a single core struggled to do everything on its own.

It was decided to implement beamforming as three beamformers in parallel, which successfully ensured significant suppression in any directions other than the steering direction across the entire frequency band. However, the necessary frequency division in the implementation adds a lot of extra computation that could be saved if a single beamformer was used for all frequencies. Additionally, the decision of three independent beamformers locks the position of the seven microphones and thereby limiting design freedom.

Even after the switch from the 3.3 V ESP32-S3 supply to the B&O supply, crackling showed up at random intervals on all microphones with no discernible time pattern. There was fairly long between cracklings, so debugging it was not prioritised. While investigating what ended up being a noisy power supply, the connectors for the microphones were soldered in place, so bad connections were likely not the culprit. During calibration and testing three recordings for each angle and sweep or tone combination were made, so the mean could be used to mitigate the impact of possible crackling.

The maxmimum length sequence (MLS) was presented as another method for estimating the impulse reponse of the system in order to extract the beampattern of the implemented system. The method was however not attempted, so its ability to work alongside the apparent noise in the measurements is unknown. Though, the method would most likely suffer from the same complication as exponential swept sine (ESS), where the likelihood of a crackle in the measurement is bigger due to the extended measuring time.

The implementation was designed for underwater usage, but the validation was performed in air. The primary consequence of this was the shape of the beampattern, as the wavelength of a signal in air relative to water is about a quarter. In retrospect, it would have been ideal to firstly design the beamformer for air, as this would make the validation of the implementation easier. Additionally, the microphones and speaker should have been equalised before the measurements in order to be able to verify whether the gain was constant in the steering direction, as decided in the design philosophy. This would also further increase the similitude of the simulated and measured beampatterns.

The physical construction was handcrafted using a jigsaw and a drill. Measurements were made by hand with a ruler, so an imprecision of a few millimetres is to be expected. This imprecision also has an impact on the similitude between the simulated and measured beampatterns. Determining the effect of the imperfect microphone placements could be explored in simulations.

Conclusion 6

The objective of the project was to solve the problem of lack of training data for acoustic detecting of drowning accidents as a continuation of two earlier University projects. The system aimed to autonomously collect audio recordings of people jumping into water at a bathing location, so the end-users can train algorithms to alert the proper authorities when a drowning event is detected.

Based on the initial analysis and the use-case, the following problem statement emerged:

How is a system designed and implemented to autonomously detect and record splashes at a bathing location on a low power budget and with cheap transducers with the intent to create a large dataset for training of algorithms?

To successfully develop such a system, a set of requirements were established, categorised as functional and technical. The functional requirements were defined to ensure that the overall functionality of the system adhered to the problem statement. The technical requirements aimed to make sure a final system would be operational in the designated environment, while capturing subaqueous audio of the quality described in the previous projects.

Hardware selections were made based on the technical requirements. It was decided to utilise a PSoC 5LP for sampling 7 24-bit I2S microphones and an ESP32-S3 for processing. A physical system was constructed with the 7 microphones mounted on a plexiglass mount. The previous project by Frederik determined that nitrile gloves could be used for impedance matching traditional microphones to water [3]. Placing a microphone into a nitrile glove makes traditional mounting practices impossible, so a non permanent clamping system was developed, so conversion to a water safe system was non-permanent.

The autonomous detection of the splashes was decided to be implemented as a combination of delay-and-sum beamforming and spectral density analysis. From measurements conducted by the authors of the preceding projects on the same topic it was discovered that a splash has the characteristics of an impulse. Based on this, it was decided that the system should determine the direction of arrival of any incoming signal and use beamforming to concentrate the beam of the microphone array towards the signal. If the signal matches the required energy spectra and length, the signal would be detected as a splash.

In order to perform tests efficiently, an automatic test system was developed based around a remote controlled turntable. The microphone array was mounted on the turntable and calibrated in relation to the speaker. A server, which was also configured to receive data from the ESP32-S3 and connected to the speaker, could then start a tone or sweep, start a recording, wait for a desired period of time, stop and save the recording and rotate the turntable to the next position.

This automation made it possible to perform both short and long running tests without human intervention, resulting in both time saved and better repeatability. When each test was finished, all data was automatically zipped and uploaded to the cloud.

The beampattern of the designed beamformer was measured with an exponential sine sweep (ESS) that clearly resembled the simulated equivalent, but was influenced heavily by sporadic crackles in the microphones, as each sweep lasted 24 seconds. This was circumvented by switching measuring method to a more naive approach with single tones at a time (STT) only measuring one second at time. This allowed for a more noise-free beampattern but the combined measuring time increased by a magnitude. In order to properly compare the measured to the simulated beampatterns the source of the sporadic crackles should be found and resolved.

The design criteria for the delay-and-sum beamformer was that the magnitude in the steering direction and adjacent degrees should be constant. As the test setup and microphones were not calibrated to a flat frequency response before the measurements, it is hard to verify whether the criteria is met. However, as the magnitude variation in the steering direction is consistent with the measured frequency response at the centre microphone, it appears to be in compliance if the frequency sensitivity is equalised.

The compliance with the four functional requirements are as follows.

F.1 Detect the splash from a person jumping into the water.
Since beamforming was considered the largest hurdle in the designed system, the implementation was focused around the same. **F.1** was determined to be doable by inspection of spectral density, where beamforming minimises potential sources of error, resulting in the implemented system being deemed a partial success in relation to **F.1**.

F.2 Record the submerged audio of the splash.
All subsystems needed for fulfilling **F.2** were developed fully, however the functionality was never tested. Since the subsystems worked in air, it is assumed to work under water as well, based on the success of the previous project by Frederik Sidenius Dam.

F.3 Offload the recorded data wirelessly.
This functionality was used meticulously during measurements, as the results were continuously uploaded to a laptop.

F.4 The system must function on a low power budget.
Considerations and suggestions for low-power design were made throughout the report, but the power utilisation was not a top priority for the prototype.

The technical requirements primarily concern hardware specifications, meaning the compliance was decided through the choice of hardware. The selected microphones fail to comply with **T.1** due to their frequency response and maximum frequency, but they were the most suitable candidate at the time and it is possible to equalise the output of the microphones to equal sensitivity through filtering. The microphones have a built in band-pass filter, limiting the frequency range to 20 Hz to 20 kHz, which is undesired.

Furthermore, the sample rate (**P.1**) was also not complied with, but since it was a consequence of the microphone choice, it can be fixed by replacing them, also fixing the limited frequency range.

The system needs slightly more development before deployment and subsequent data collection. The developed system and the part that was implemented is however deemed a success, since it is assessed that finishing the implementation is straightforward with what was accomplished in this project.

Further Work 7

A complete implementation of the recording system described in the use-case would allow the end-user to create a robust drowning alert system based on the data collected by the system, potentially saving lives in the harbours of the world. This project ended up primarily focusing on an implementation of a DAS beamformer for subaqueous use, whereas more work has to be done on the actual detection of splashes. Investigative work was however done on the entirety of the data collection system, so a continuation of the project should be straightforward.

The next step in the design phase is direction of arrival (DOA), where time-difference of arrival (TDOA) was selected in this project. It was determined to use spectral density, but a continuation of the project should include an analysis of the specifics involved, as bin sizes and the most important frequency ranges are unknown. It would likely be relevant to analyse the frequency areas of the most common noise sources in harbours, as to make sure the system is resilient towards them.

It was determined that equalising the microphones would likely improve the system significantly, since the frequency response of the signal chain from laptop to Q_3 was eerily similar to that of the main lobe frequency response in the beamformer. A final version of the system should reprimand this error, either by use of microphones with a flatter frequency response, or by correcting it by applying the inverse filter either directly on the custom sampling hardware or by applying the inverse filter of the average microphone frequency response on the beamformed signal.

With the validation completed in air and DOA and TDOA implemented, the array is ready to be waterproofed such the system can be tested in a watery setting. The design work for waterproofing is done in this project, but its effectiveness should be formally verified before actual testing. Firstly, an appropriate threshold for the spectral density should be determined. Thereafter, the system's ability to successfully trigger on splashes from subjects can be evaluated and optimised.

A final installation of the data collector would need a deliberate power budget, since there is a general lack of outlets near relevant bathing locations. A balance would need to be struck between power usage, battery size and solar panel/wind turbine size. The ESP32 series might be a good enough choice for a final product, but the sampling circuit can most certainly be improved upon power usage wise. Very low power components exist, meaning it should be feasible to save significant power by designing a custom circuit for the use-case.

In Section 2.6 the average theoretical speed of sound of the water in Vestre Fjordpark is calculated to 1496 $\frac{\mathrm{m}}{\mathrm{s}}$ but with theoretical variations of up to 90 $\frac{\mathrm{m}}{\mathrm{s}}$. The speed of sound is critical for the beampattern application as the delays are based on the expected propagation delay. Therefore, the system should be able to adjust the speed of sound according to the conditions. This can be

implemented through sensors in the system, but as the system is already developed with active Wi-Fi communication, the parameters could be fetched from some database.

Based on the desired data collection location it might be relevant to collect a permit to record in public. Some countries, such as Denmark, disallow recordings of conversations, unless the persons recording partake in the conversations themselves.

7.1 Alerter-System

The system developed in this project was designed for autonomously collecting data for training an algorithm to detect a splash. However, when an algorithm has been developed, some system must be designed that deploys the algorithm. The **Alerter** system potentially shares a lot of similarities to the developed **Collector** system, as both's functionality is to passively surveil an underwater area and wait for a splash. The main distinction of the two systems is the purpose, as the purpose of the **Collector** is to record the splash event with spaced microphones, whereas the **Alerter** is supposed to alert emergency services on a splash event. As a consequence, the **Alerter** must be significantly more confident in a detection than the **Collector**.

Differential beamforming could be an ideal implementation in the **Alerter** as an effort to increase SNR and thereby confidence of detection. The case for using delay-and-sum beamforming in the **Collector** was to have big spacings between the microphones, whereas the opposite might be preferable for the **Alerter** to minimise its form factor. If the system has the ability to determine the direction of arrival for noise sources, differential beamforming's frequency invariance can be exploited to its fullest by placing a null in that direction.

Additionally, the ESP32-C3 might be a better fit for the **Alerter**, as this version of the ESP32 is intended for ultra-low power use that can further minimise the power consumption of the system. A suitable location for the **Alerter** system could be on something like Trygfonden's rescue ladders that are equipped with solar panels.

Bibliography

[1] Hannah Ahrensberg, Lau Caspar Thygesen, and Maria Holst Algren. Druknedødsfald i Danmark i 2021 – og udviklingen 1970-2021, July 2023.

[2] Line Ettinger Julsgaard. Termiske kameraer har reddet to mennesker fra at drukne - her er prisen for at sikre resten af havnefronten, February 2022. [Online; accessed 20. Feb. 2024]. URL: https://nordjyske.dk/nyheder/aalborg/kameraer-har-reddet-to-mennesker-i-aalborg-havn-lige-uden-for-deres-raekkevidde-faldt-oliver-i-vandet/c852956a-a75a-402b-8f3b-6ce0fb8f9abf.

[3] Frederik Sidenius Dam. Acoustic Detection for Warning of Drowning Accidents, June 2023.

[4] Axel Villads Burford Toft, Kaj Mørk, Lukas Bisgaard Kristensen, Máté Tallósi, and Tudor Razvan-Tatar. Acoustic Detection for Warning of Drowning Accidents, May 2023.

[5] Stephen Robinson and Ben Ford. Technical Guides - Speed of sound in sea water - Underlying Physics. [Online; accessed 19. Feb. 2024]. URL: http://resource.npl.co.uk/acoustics/techguides/soundseawater/underlying-phys.html.

[6] Michael A. Ainslie and James G. McColm. A simplified formula for viscous and chemical absorption in sea water. *The Journal of the Acoustical Society of America*, 103(3):1671–1672, March 1998. arXiv:https://pubs.aip.org/asa/jasa/article-pdf/103/3/1671/10678344/1671_1_online.pdf, doi:10.1121/1.421258.

[7] Samuel Jones. How to Size a Solar Panel for an IoT Device, November 2022. [Online; accessed 26. Feb. 2024]. URL: https://blog.powerfilmsolar.com/how-to-size-a-solar-panel-for-an-iot-device.

[8] Google Maps. Imagery ©2024 Aerodata International Surveys, Airbus, Maxar Technologies, Scankort, Map data ©2024.

[9] Water temperature in Aalborg, February 2024. [Online; accessed 20. Feb. 2024]. URL: https://seatemperature.net/current/denmark/aalborg-north-denmark-region-denmark-sea-temperature.

[10] Aalborg Kommune. Badevandsprofil: Limfjorden Vestre Fjordpark, 2021.

[11] Jacob Carstensen, Nikolaj R. Andersen, Jesper Christensen, and Christian Mohn. HAVETS pH-BALANCE - påvirkning fra klima og næringsstoffer. (429), February 2021. URL: https://dce2.au.dk/pub/SR429.pdf.

[12] Eberhard Sengpiel. Calculation method of absorption of sound by atmosphere air, February 2017. [Online; accessed 28. Feb. 2024]. URL: https://sengpielaudio.com/calculator-air.htm.

[13] Editor Engineeringtoolbox. Seawater - Properties, September 2023. [Online; accessed 26. Mar. 2024]. URL: https://www.engineeringtoolbox.com/sea-water-properties-d_840.html.

[14] Jacob Benesty and Jingdong Chen. *Study and Design of Differential Microphone Arrays*. January 2012. doi:10.1007/978-3-642-33753-6.

[15] Differential Beamforming Introduction Example, March 2024. [Online; accessed 19. Mar. 2024]. URL: https://se.mathworks.com/help/phased/ug/introduction-to-differential-beamforming.html.

[16] Jilu Jin, Gongping Huang, Xuehan Wang, Jingdong Chen, Jacob Benesty, and Israel Cohen. Steering Study of Linear Differential Microphone Arrays. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:158–170, 2021. doi:10.1109/TASLP.2020.3038566.

[17] Getting Started - - — Arduino ESP32 latest documentation, March 2024. [Online; accessed 28. Mar. 2024]. URL: https://docs.espressif.com/projects/arduino-esp32/en/latest/getting_started.html.

[18] Infineon Technologies A. G. 32-bit PSoC™ 5 LP Arm® Cortex®-M3 - Infineon Technologies, March 2024. [Online; accessed 28. Mar. 2024]. URL: https://www.infineon.com/cms/en/product/microcontroller/32-bit-psoc-arm-cortex-microcontroller/32-bit-psoc-5-lp-arm-cortex-m3.

[19] Cypress Semiconductor. PSoC® 5LP Prototyping Kit Guide, 2018. [Online; accessed 8. May 2024]. URL: https://www.infineon.com/dgdl/Infineon-CY8CKIT-059_PSoC_5LP_Prototyping_Kit_Guide-UserManual-v01_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ef981770f63.

[20] Cypress Semiconductor Corporation. Inter-IC Sound Bus (I2S) 2.70, November 2017.

[21] Cypress Semiconductor Corporation. Serial Peripheral Interface (SPI) Master 2.50, October 2017.

[22] Data Sheet DMM-4026-B-I2S-EB-R, 2020. [Online; accessed 13. May 2024]. URL: https://www.mouser.dk/datasheet/2/334/DMM_4026_B_I2S_EB_R-1853413.pdf.

[23] Sample Rates and Audio Sampling: A Guide for Beginners | Adobe, May 2024. [Online; accessed 18. May 2024]. URL: https://www.adobe.com/uk/creativecloud/video/discover/audio-sampling.html.

[24] Cypress Semiconductor Corporation. Clock 2.20, March 2018.

[25] ESP32-S3 Series Datasheet, 2023. [Online; accessed 13. May 2024]. URL: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf.

[26] SPI Slave Driver, 2024. [Online; accessed 12. May 2024]. URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/api-reference/peripherals/spi_slave.html.

[27] FreeRTOS Overview, 2024. [Online; accessed 12. May 2024]. URL: https://docs.espressif.com/projects/esp-idf/en/v5.2.1/esp32s3/api-reference/system/freertos.html.

[28] ESP-DSP Library - ESP32, July 2023. [Online; accessed 22. May 2024]. URL: https://docs.espressif.com/projects/esp-dsp/en/latest/esp32/index.html.

[29] ESP32-S3 Series Datasheet, 2024. [Online; accessed 24. May 2024]. URL: https://www.espressif.com/sites/default/files/documentation/esp32-s3_technical_reference_manual_en.pdf.

[30] Fundamentals of Acoustics and Sound (CE8-AVS, ES8), 2023. [Slides; accessed 26. May 2024].

Github Software A

The file tree of referenced folders and files is found below. Every entry is a hyperlink to Github when the report is viewed as a PDF. If you are reading a printed version, or the links are not working, everything is located at https://github.com/UnknownDK/ES10-Drowning.

- /
  - dirtree.py
  - ESP32/
    - audioStream.py
    - ParforsystemIImpulseResponse.mlx
    - audioStream.mlx
    - micCalibrationLongDist.mlx
    - micCalibration.mlx
    - micOutputsPlayer.mlx
    - systemIImpulseResponse.mlx
    - spidemo/
      - tcp_client.py
      - main/
        - app_main.c
    - tcp_client/
      - pytest_tcp_client.py
      - main/
        - tcp_client_v4.c
        - tcp_client_v6.c
        - tcp_client_main.c
    - SplashDetect/
      - main/
        - offloading.c
        - main.c
        - offloading.h
        - constants.h
        - DAS.c
        - DAS.h
        - FIRWeights.h
    - wifidemo/
      - main/
        - wifi_eap_fast_main.c
    - I2Sdemo/
    - cppstreamer/
      - pySocketStream.py
      - audiotest.py
      - server.exe

- zero.py
- plotaudio.py
- control.py
- UDP2TXT.py
- socketStream
- autoCalib.py
- autoTest.py
- socketStream.cpp
- onlyBeamform/
- bioncodetest/

PSoCSampler/
- PSoCSampler.cywrk
- SamplingCircuit.cydsn/
  - cyapicallbacks.h
  - main.c
  - TopDesign/
    - TopDesign.cysch

MATLAB/
- naiveSweepFrequencyResponse.mlx
- fractionalDelays.mlx
- DAS.mlx
- sineToneGen.mlx
- SystemSim.mlx
- Spectrogram.mlx
- PSD.mlx
- parfor_wait.m
- DMA.mlx
- cylindricalPlot.mlx
- checkESPOutput.mlx
- aosTest.mlx
- circPlot.mlx
- plotWav.mlx
- DAS8M.mlx
- diretivity.mlx
- DMALUT.mlx
- sweptSineGenerator.mlx
- genFIRFilters.mlx
- PSDottendPlot.mlx
- freqResponsePlots.mlx
- DASVerification.mlx

DAS/
- main/
  - main.c
  - FIRWeights.h

Journal: Test of Data Path from Microphone to Server

B

The purpose of this journal is to verify the implementation of the data path from microphones to the server. The measurements were performed May 11. 2024 on Fredrik Bajers Vej 7b, in the anechoic chamber in room B4-111.

B.1 Software Tools

A table containing the main software, their functionalities and their versions can be seen in Table B.1.

Table B.1. Software used to monitor the received SPI data on the ESP32-S3 and the received data on the UDP server. Name, functionality, and version are presented.

Name	Functionality	Version
Visual Studio Code	Code Editor	1.89.0
ESP-IDF	Development framework for ESP32-S3	v5.2.1
ESP-IDF Visual Studio Code Extension	VScode extension	v1.7.1
MATLAB	Data processing	R2023b
Python	UDP server	3.10.12

B.2 Equipment

A table containing the main equipment, their functionalities, and AAU-Nr. can be seen in Table B.2.

Table B.2. List of the components used in the measurement.

Device	Functionality	AAU-Nr.
PSoC 5LP	Microcontroller	-
ESP32-S3	Microcontroller	-
Lenovo Yoga Slim 7 Pro	ESP32 serial monitor UDP server Playback audio MATLAB	-
TP-Link TL-WR841N	Wi-Fi for offloading	-
B&W DM601 S2	Speaker	02144-00
Focusrite Scarlett Solo	Soundcard	2176-22
Pioneer A-616	Amplify audio signal	08341-00
DMM-4026-B-I2S-R Microphones	Capturing audio	-
B&O SN16A	Power supply	33130

Table B.3. List of the cables used in the measurement.

Type	Amount
AC-C13 EU	1
USB A-B	1
RCA	2
AC-C13 EU	1
XLR	1

B.3 Physical Setup

The microphones, the PSoC and the ESP32 are wired as seen on Figure B.1. A picture of the lineup in the anechoic chamber is seen on Figure B.2.

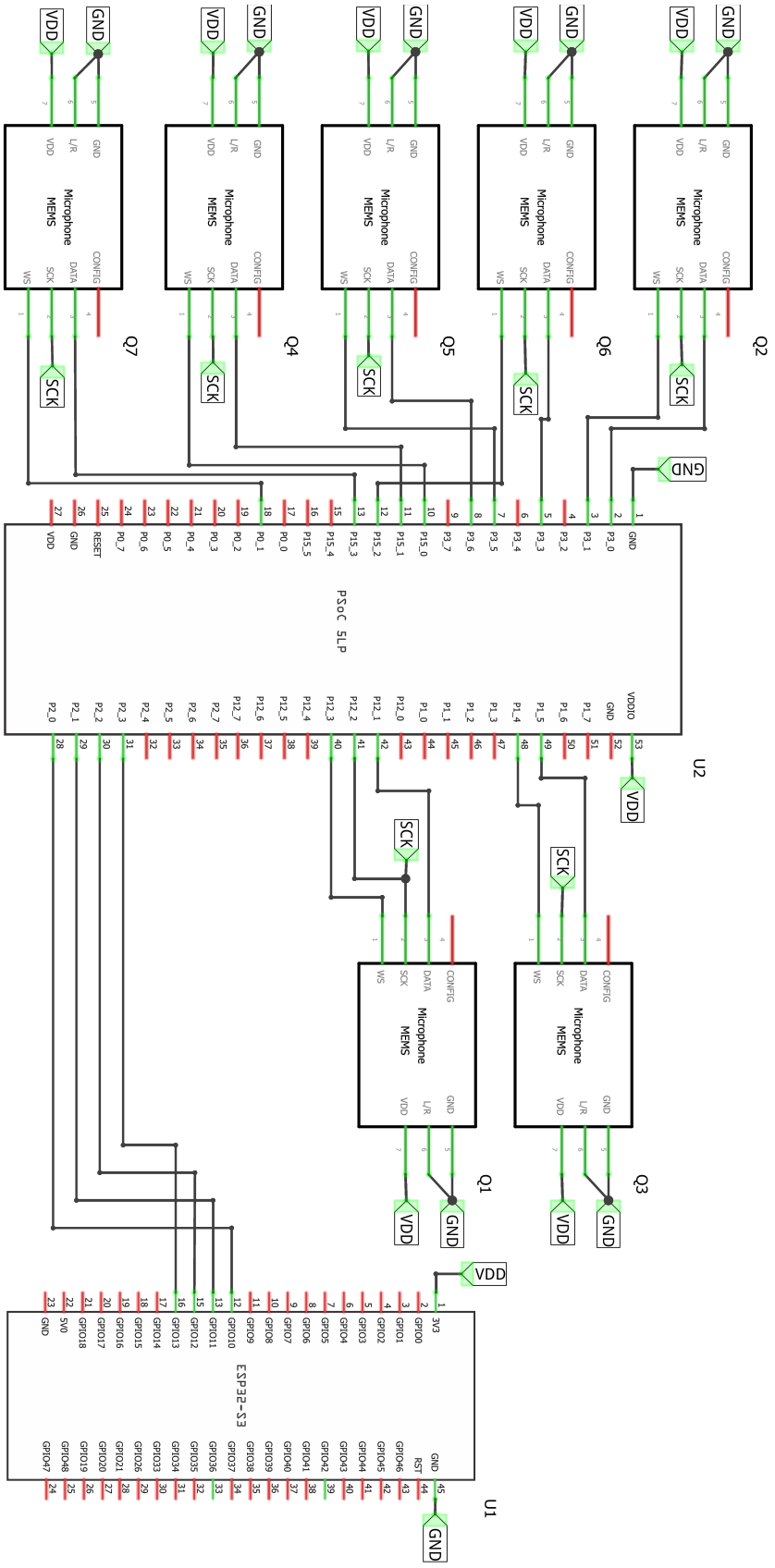


Figure B.1. Schematic of wiring for microphones, PSoC 5LP, and ESP32-S3.

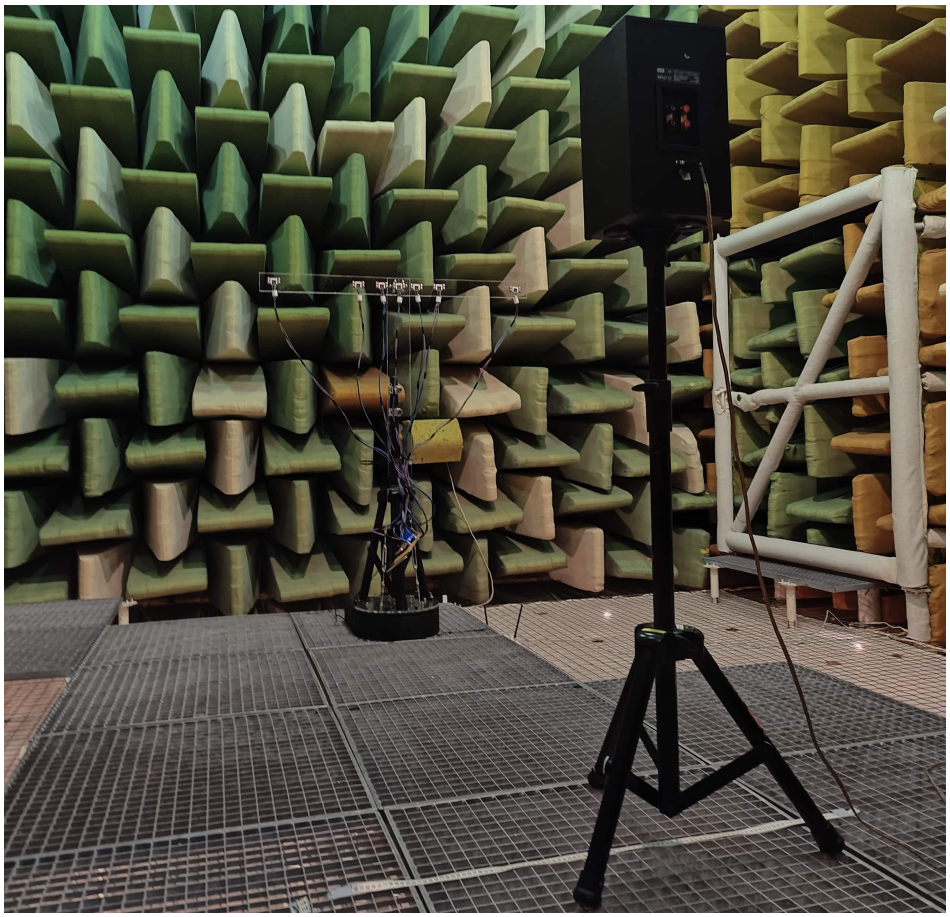


Figure B.2. The photo shows the microphone array turned to 90 degrees and a speaker directly in front of Q_3, two meters away.

B.3.1 Procedure/Course of Action

In order to test the whole signal path from microphone to Wi-Fi server, a known sine is played from a speaker into the microphones in the anechoic chamber on Fredrik Bajers Vej 7b room 4-111. This can then be visually inspected in the time domain and the frequency domain after the data is saved on the server. The microphones are configured with the distances described in Figure 4.17 from Chapter 4 with the speaker two meters away from the centre microphone (Q_3).

B.3.2 Measurement

The measuring procedure for the test is as follows.

1. Set up the equipment and software cf. Figure B.1.
2. Make sure Wi-Fi is available.
3. Power on the ESP32-S3, PSoC 5LP and server.
4. Play 1000 Hz sinus from speaker.
5. Start UDP server.
6. Wait 8 seconds.
7. Stop UDP server.

The software algorithm for data capture during the test is seen in Figure B.3.

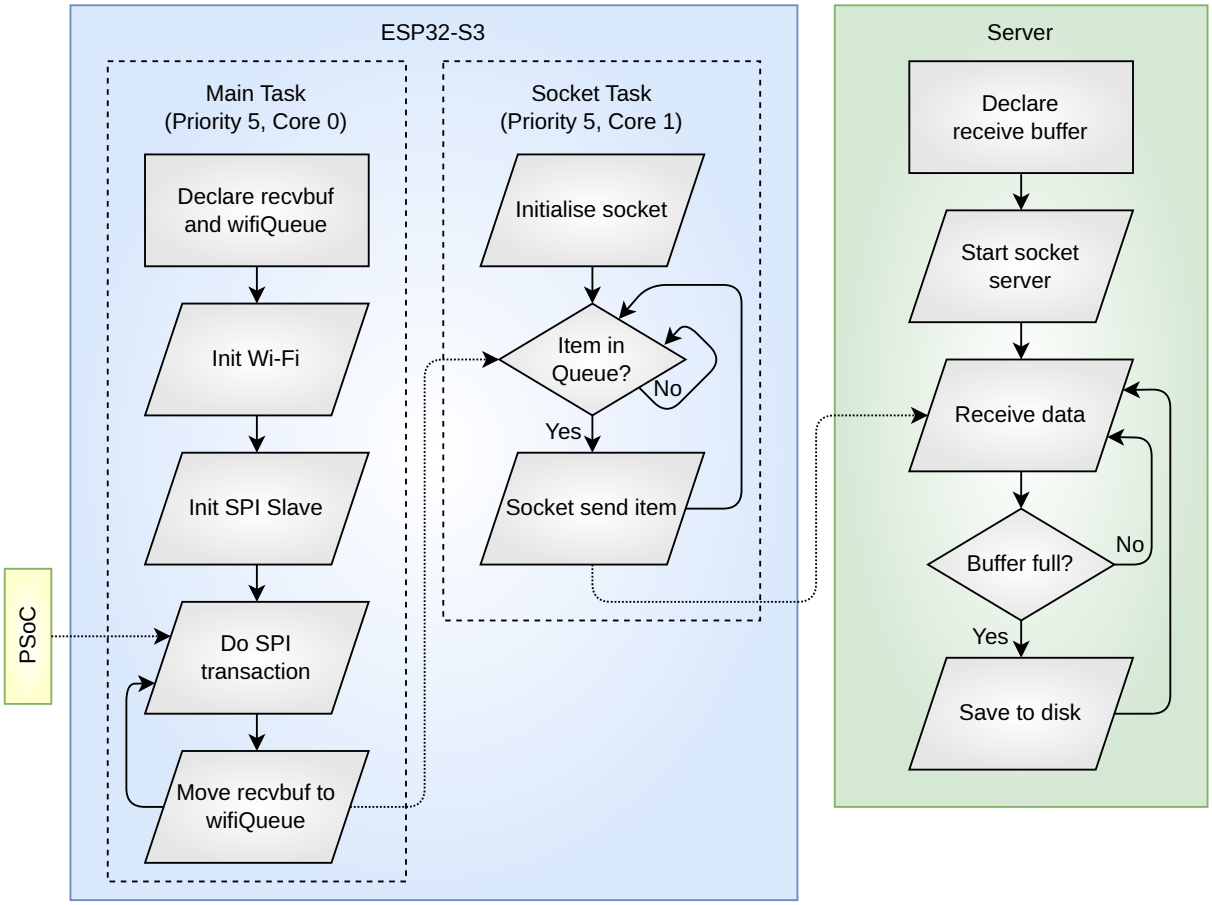


Figure B.3. Diagram of the test code for the ESP32-S3 showing the FreeRTOS tasks for SPI, sockets for offloading to the server.

B.3.3 Results

It was immediately observed that all microphones would intermittently adopt capacitor unloading like behaviour as seen on Figure B.4 when a pure tone was played.

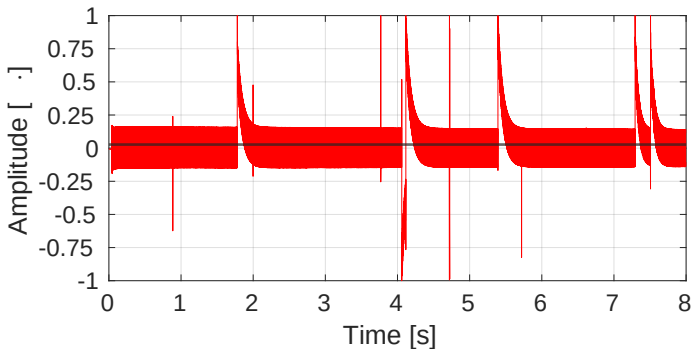


Figure B.4. Captured audio on Q_0 when a pure 1 kHz tone was played 2 meters away directly in front of it, showing capacitor unloading like behaviour.

A correlation with volume was quickly observed. The distortion looks analogue, so the only analogue part of the signal chain i.e. the I2S microphones was deemed to contain the problem.

Since it is unlikely that all purchased microphones were factory defect, it was decided to swap out the power supply as seen on Figure B.5.

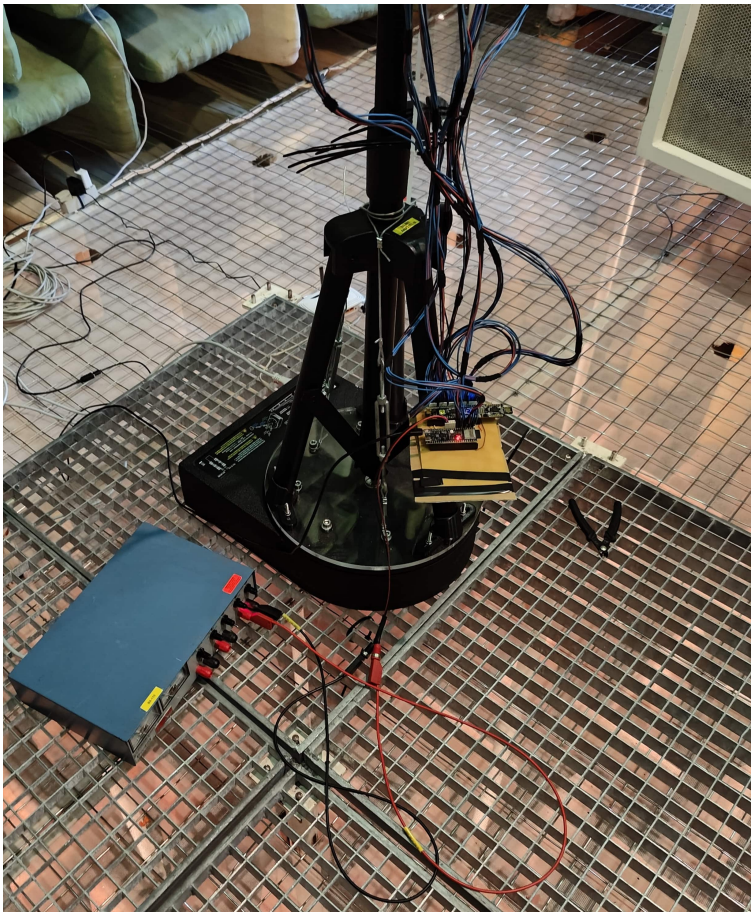


Figure B.5. Photo of new B&O power supply, connected to the PSoC 5LP, which is supplying the microphone array.

The ESP32-S3 was supplying everything before, so a B&O power supply was used for the PSoC part instead. This fixed the issue as seen on Figure B.6.

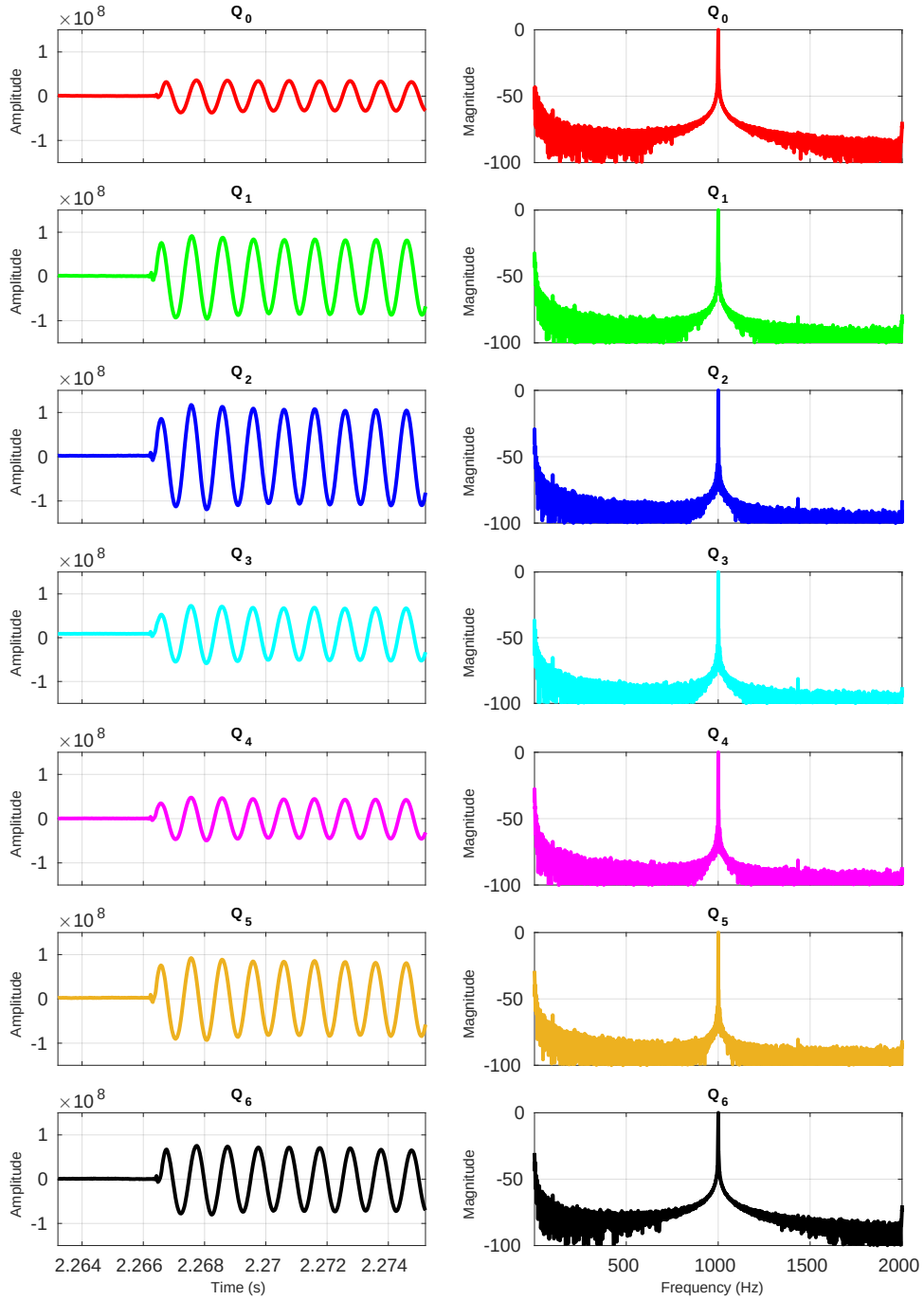


Figure B.6. Plots of measured audio when a 1 kHz sine is played directly in front of Q_3 two meters away in both time and frequency domain.

Figure B.6 shows the measured audio when a 1 kHz sine is played directly in front of Q_3 two meters away in both time and frequency domain. It is seen that the sines show no artefacts and the frequency spectrum's have a peak in 1 kHz.

Table B.4. Time of peak of first sine period for each microphone subtracted the time of peak for Q_3 in samples.

Microphone	Arrival of Signal [samples]	Expected AOS [continous samples]
Q_0	7	7.86
Q_1	1	0.89
Q_2	0	0.16
Q_3	0	0.00
Q_4	0	0.16
Q_5	1	0.89
Q_6	7	7.86

In Table B.4 it is seen that the sine arrives first to the three centre microphones and then progressively to the further away microphones.

B.4 Conclusion

The captured measurements show that the SPI setup between the PSoC 5LP and ESP32-S3 is correct, when the power supply is switched out. It also shows that the wavefront is received at slightly different intervals, which is expected since the microphones are at slightly different lengths away from the speaker.

Journal: Validation of Implementation

C

The purpose of this journal is to test the implementation of the delay-and-sum beamformer onto the ESP32-S3 described in Section 4.5. The journal contains two parts: **Microphone Calibration** and **Beamformer Measurements**. The parts share the same test setup, but calibration needs to be done in order to generate the beampatterns.

The beamformer was designed in Section 4.1 with underwater usage in mind, meaning the speed of sound was set to 1496 $\frac{\text{m}}{\text{s}}$ in the calculations. However, the tests in this journal are conducted in air, as this significantly reduces the complexity of the setup and no underwater anechoic chamber was available at the time of measuring. Therefore, in order to have a comparison for the implementation, the simulations from Section 4.1 are redone with the speed of sound in air (343 $\frac{\text{m}}{\text{s}}$).

The measurements in this journal were performed between the 19th of May and 25th of May 2024 in the anechoic chamber 4-111 on Fredrik Bajers Vej 7B.

In order to extract the beampattern of the beamformer, the system's response to an impulse at a range of incidence angles must be measured. However, generating an auditory impulse that is always identical is a challenge.

C.1 Excitation Signal

A method for exciting the system and extracting the frequency (or impulse) response must be chosen. Multiple methods for finding the impulse of a linear time-invariant system is presented.

C.1.1 Maximum Length Sequence (MLS)

The MLS method stimulates the system with a pseudo-random binary sequence. The impulse response is then extracted by performing Fast Hadamard Transform (FHT) on the output and analysing the circular cross-correlation between the in- and output. The procedure for MLS is illustrated in Figure C.1.

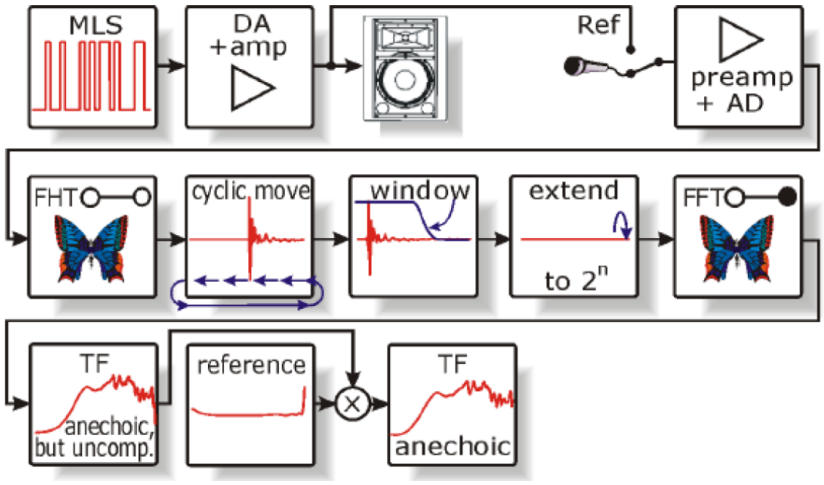


Figure C.1. Illustration of the steps in the MLS method. The output of the system is transformed with the Fast Hadamard Transform (FHT), moved, windowed, extended, FFTed, and multiplied with a reference [30].

C.1.2 Exponential Swept Sine (ESS)

Another option for estimating the impulse response is ESS. The excitation signal of the ESS is a logarithmic frequency sweep with more energy in the lower frequencies. Due to this energy discrepancy, the output of the system is adjusted by dividing the FFT of the output with the FFT of the excitation signal. The procedure for ESS is shown in Figure C.2.

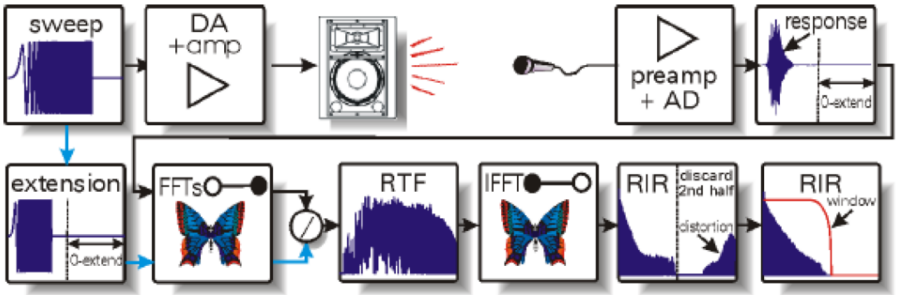


Figure C.2. Illustration of the steps in the ESS method. The output is FFTed and divided by the FFT of the exictation signal in order to account for the energy discrepancy of the input signal [30].

C.1.3 Impulse by Object (IO)

Another option is to generate an impulse with an object such as a movie clipboard or a simple clap from a person. However, replicating the exact same sound is challenging without a sophisticated setup or dedicated tool.

C.1.4 Single Tones at a Time (STT)

A more naive approach is to play a single tone or frequency range at a time, measure the output, and calculate the power. By measuring the power for some list of chosen frequencies and input angles, the frequency response of the system can be estimated.

C.1.5 Choice of Excitation Signal

It is decided to generate the beampattern through the ESS-method, as the number of steps in the procedure is the smallest. Only the frequency response is of interest, so the last three steps in the ESS-method can be disregarded (IFFT, discard, and window).

If the beampattern from ESS is unrecognisable or too noisy, the STT-method is conducted in an attempt to generate a *prettier* plot, as noise on an ESS measurement will result in noise across the frequency spectrum, whereas noise on an STT only results in noise in the single measurement.

C.2 Software Tools

The software tools used in the journal for measuring and data processing are listed in Table C.1.

Table C.1. List of the software tools used in the measurements. The name, functionality, and version of the software are presented.

Name	Functionality	Version
Visual Studio Code	Code Editor	1.89.0
ESP-IDF	Development framework for ESP32-S3	v5.2.1
ESP-IDF Visual Studio Code Extension	VScode extension	v1.7.1
MATLAB	Data processing	R2023b
Python	UDP server	3.10.12

C.3 Equipment

The list of equipment and cables used in the journal are listed in Table C.2 and C.3, respectively. The volume level of the Dell laptop and soundcard is set to the highest setting, and the gain level of the amplifier is set to 0 dB.

Table C.2. List of the components used in the measurement.

Device	Functionality	AAU-Nr.
PSoC 5LP	Microcontroller	-
ESP32-S3	Microcontroller	-
Dell Latitude E5440	UDP server Playback audio MATLAB	-
TP-Link TL-WR841N	Wi-Fi for offloading	-
B&W DM601 S2	Speaker	02144-00
Focusrite Scarlett Solo	Soundcard	2176-22
Pioneer A-616 (Modded)	Amplify audio signal	08341-00
Outline ET250-3D	Turntable to rotate microphone array	108252
DMM-4026-B-I2S-R Microphones	Capturing audio	-
B&O SN16A	Power supply	33130

Table C.3. List of the cables used in the measurement.

Type	Amount
AC-C13 EU	1
USB A-B	1
RCA	2
AC-C13 EU	1
XLR	1

C.4 Lineup

The wiring diagram of the equipment is illustrated in Figure C.3.

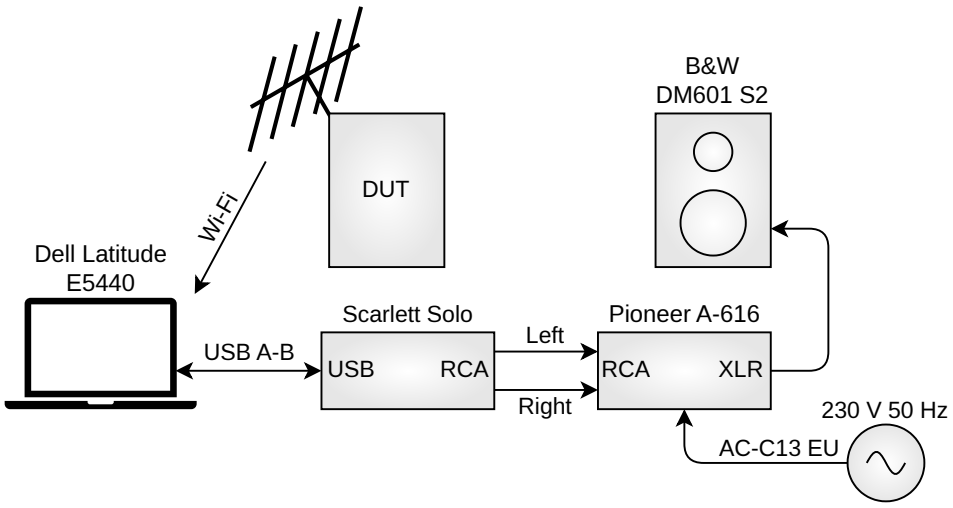


Figure C.3. The figure shows how the equipment from Table C.2 is wired together for the implementation test.

The arrangement of the speaker and microphone array is illustrated in Figure C.4. The array is pointed towards the speaker with a distance of 5.35 m to the centre of the array to the centre of the speaker. The microphones are positioned 1.42 m from the grates and so is the vertical centre of the speaker. The distance between the speaker and microphones are made as large as possible to ensure planar wavefronts.

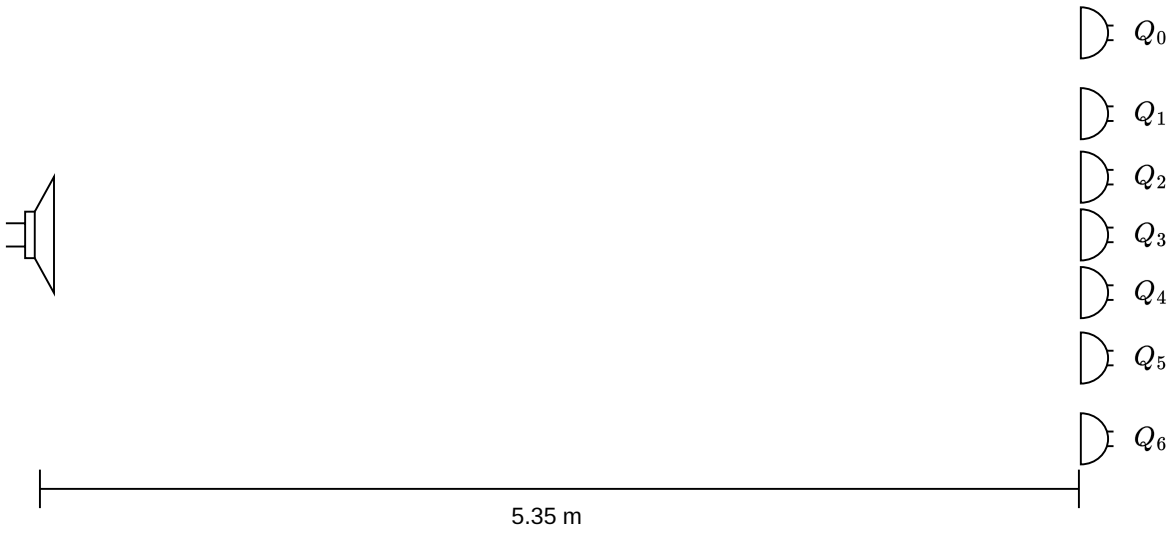


Figure C.4. The figure shows how the microphones and speaker are arranged. The illustration is not to scale.

A picture of the lineup is shown in Figure C.5. The microphone array is placed upside down in the turntable, meaning the steering angle is rotated around the pointing direction.

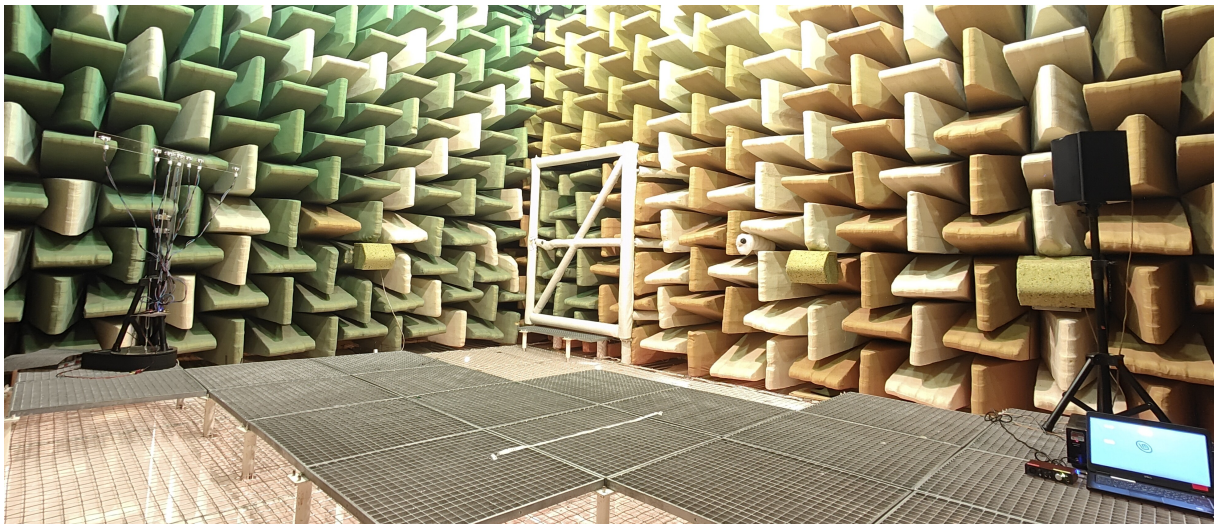


Figure C.5. The photo shows how the microphones and speaker are setup in the anechoic chamber. A Rockwool sheet was placed against the amplifier and laptop during tests in order to minimise noise influence.

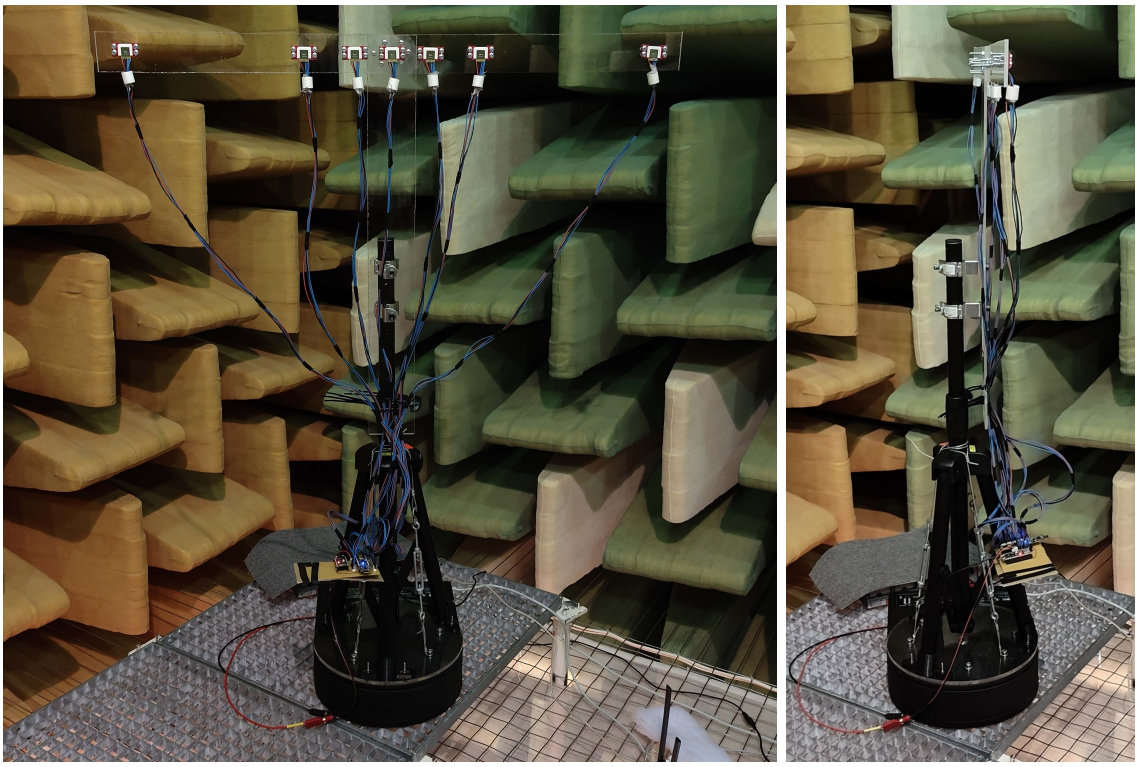


Figure C.6. The photo shows the microphone array turned to 90 degrees on the left and 180 degrees on the right. Both are taken in the anechoic chamber.

C.5 Procedures

The procedures for performing the calibration of the microphones and the beampattern measurements are listed in this section together with flowcharts for the data capture algorithms.

C.5.1 Microphone Calibration

In order for the beamformer to function as intended, the sensitivity of the microphones in the array must be similar. The software procedure for the calibration is shown in Figure C.7.

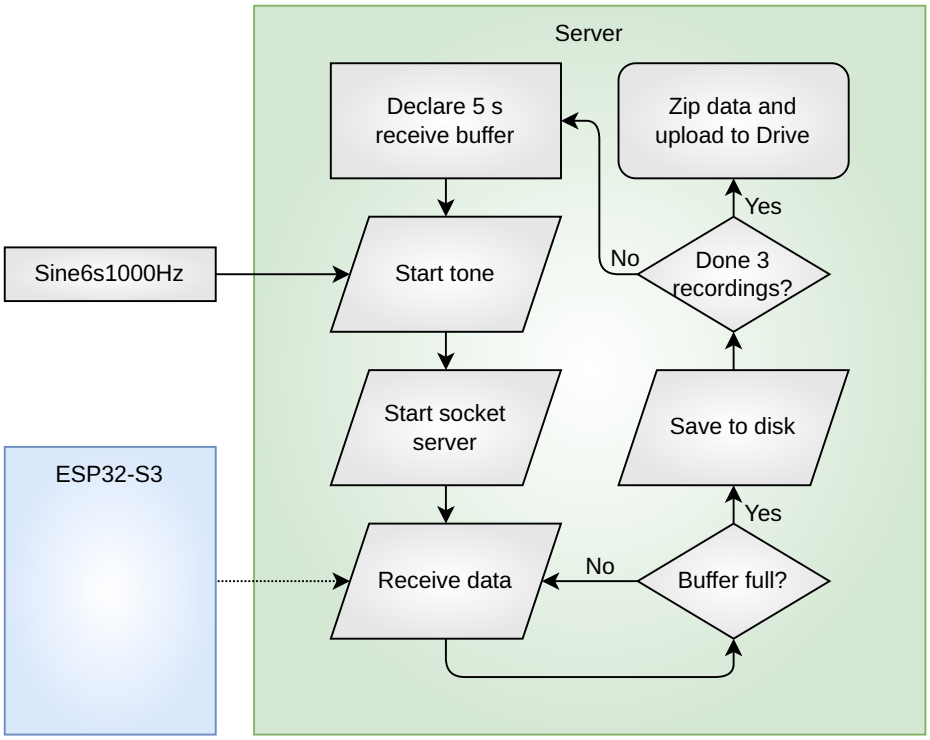


Figure C.7. Flowchart of the data capture for microphone calibration, where a 6 second 1000 Hz tone is recorded three times.

The received data is then processed to find the mean of the data and the power. From the power, gain factors for each of the microphones are determined in order to equalise the gain of the microphones. The procedure in Figure C.7 is then repeated in order to verify the calibration.

C.5.2 Beamformer Measurements - ESS

Once the microphones have been calibrated to equal gain, the beampattern can be estimated using the **ESS**-method. The excitation signal is generated using the *sweeptone*-function in MATLAB. The signal sweeps from 20 Hz to 20.8 kHz over 20 seconds starting at 2 seconds. The maximum frequency has changed from 24 kHz to 20.8 kHz due to change of sampling rate in Section 4.3. Additionally, a single period of a 1 kHz signal is inserted at 1 second for synchronisation purposes in post-processing. The excitation signal for the **ESS**-method is seen in Figure C.8.

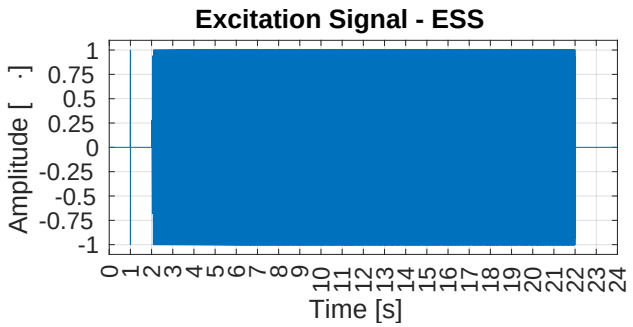


Figure C.8. Excitation signal for the ESS-method. The signal sweeps logarithmically from 20 Hz to 20.8 kHz over 20 seconds, but contains 2 seconds of silence in the beginning and end and a single period of a 1 kHz signal for synchronisation at 1 second.

The procedure for the data capture for the ESS-method is shown in Figure C.9. Three sweeps are done in order to suppress noise.

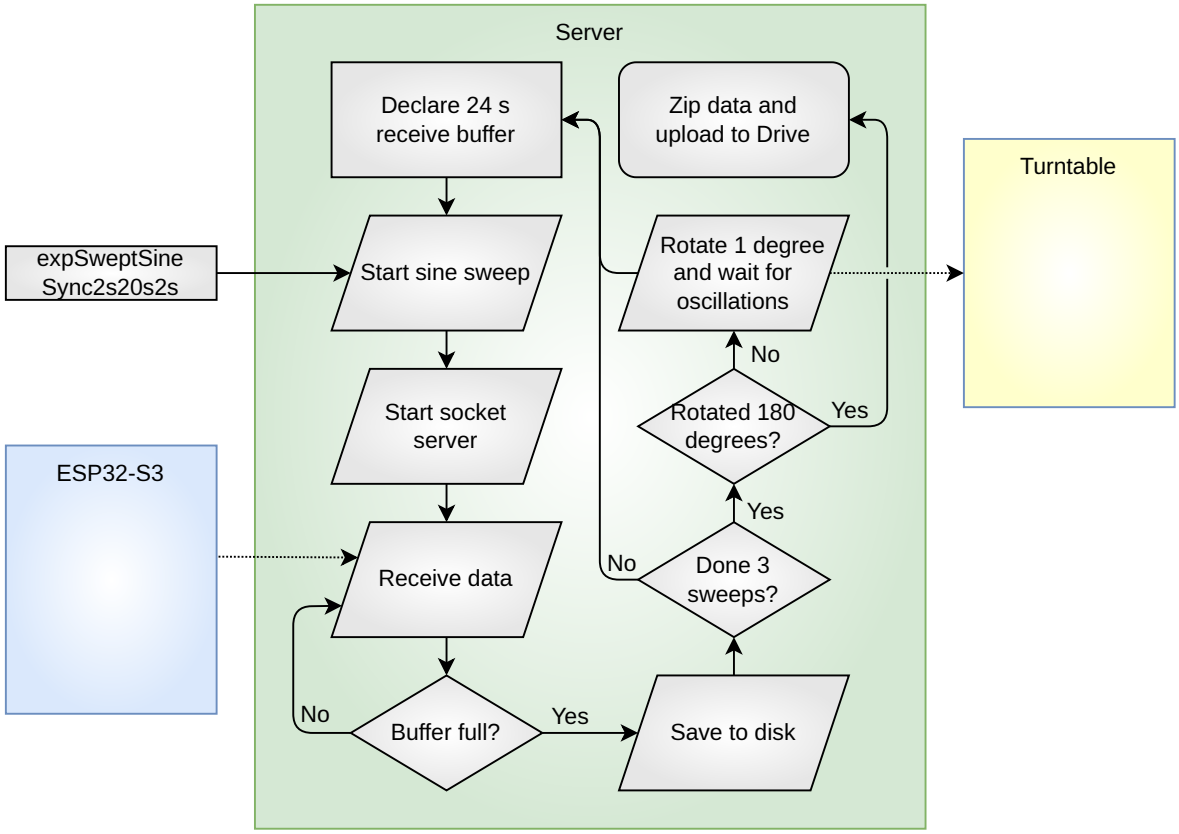


Figure C.9. Flowchart of the data capture for the ESS-method, where a 24 second sine sweep is recorded. The turntable is rotated 1 degree following the sweep with a delay to allow oscillations to cease.

The output data is then processed cf. Figure C.2 to estimate the frequency response of the system.

C.5.3 Beamformer Measurements - STT

For the STT-method, a list of single-tone signals are generated with a resolution of 200 Hz in the span of 50 Hz to 20.8 kHz and a length of 4 seconds. The procedure for the data capture is illustrated in Figure C.10. Three recordings are done in order to suppress noise.

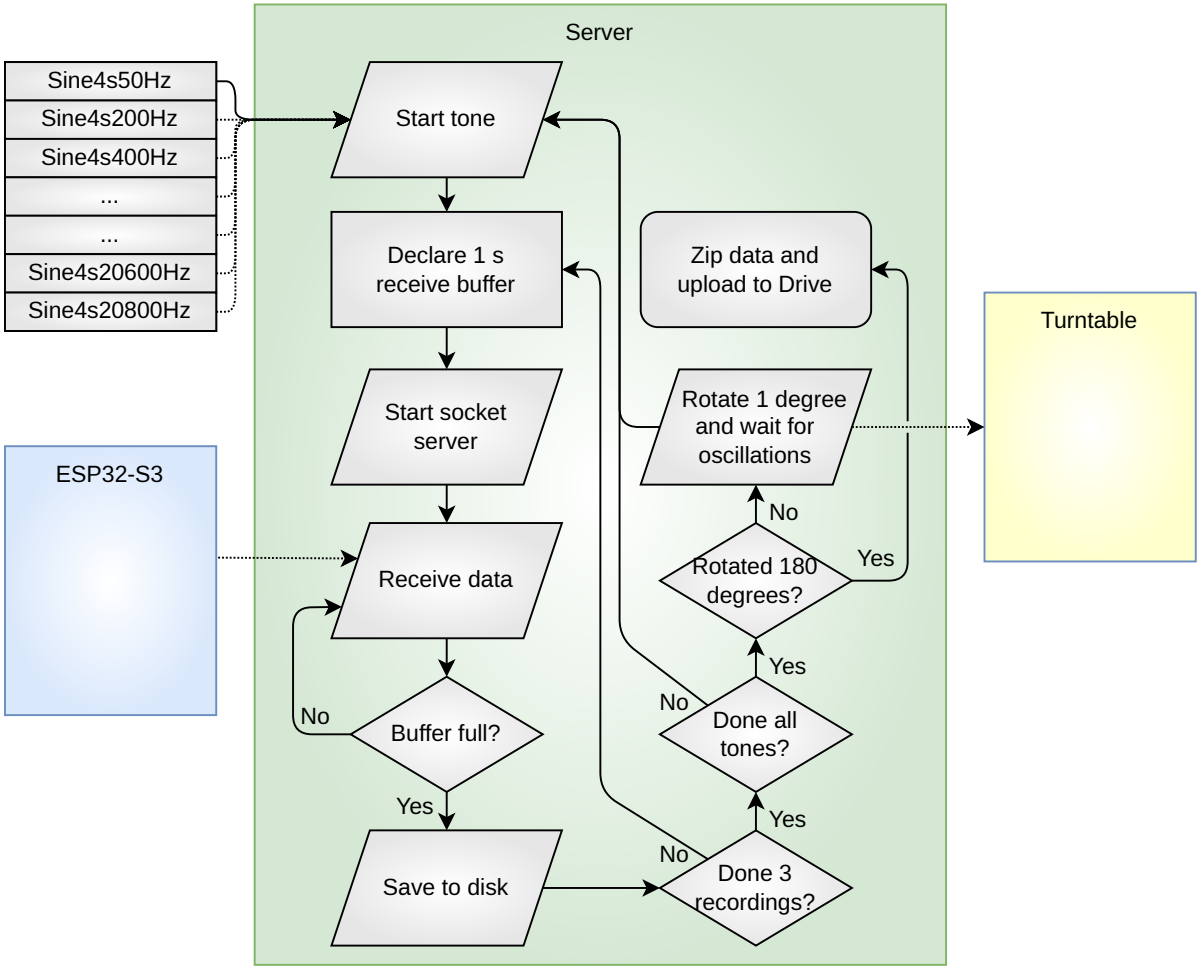


Figure C.10. Flowchart of the data capture for the STT-method, where three 1 s recordings are made for each tone. The turntable is rotated 1 degree following the rotation with a delay to allow oscillations to cease.

The power and mean of each of the recordings are calculated and logged.

C.6 Results and Discussion

The results of the measurements are presented and discussed in this section.

C.6.1 Microphone Calibration

The three raw measurements for each of the microphones during the first part of calibration are plotted below.

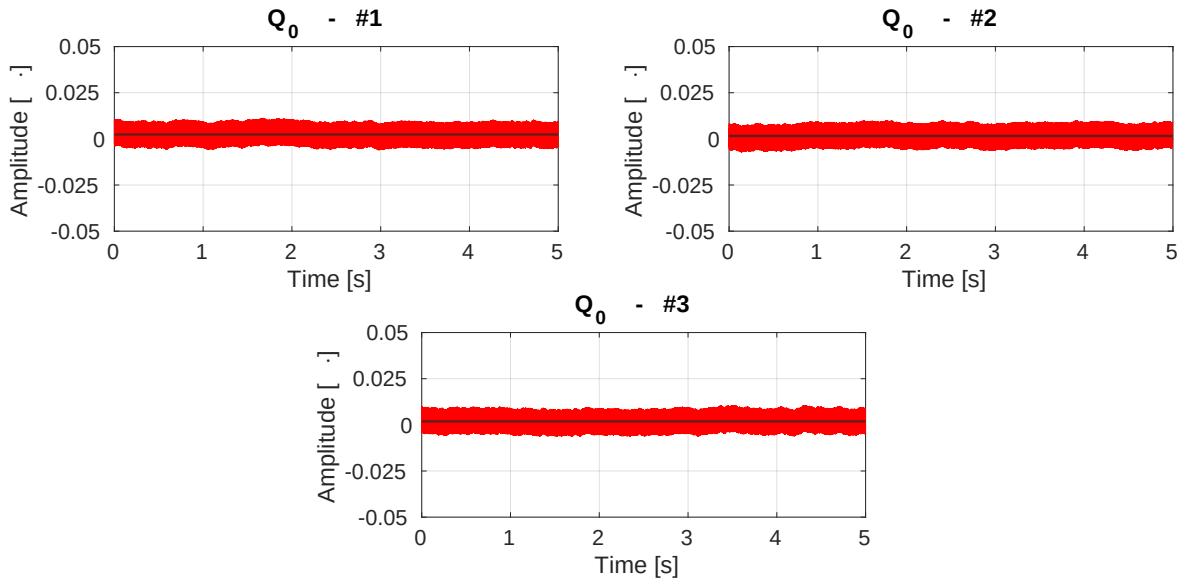


Figure C.11. Plot of the measured data before calibration for Q_0. The line shows the mean value of the data.

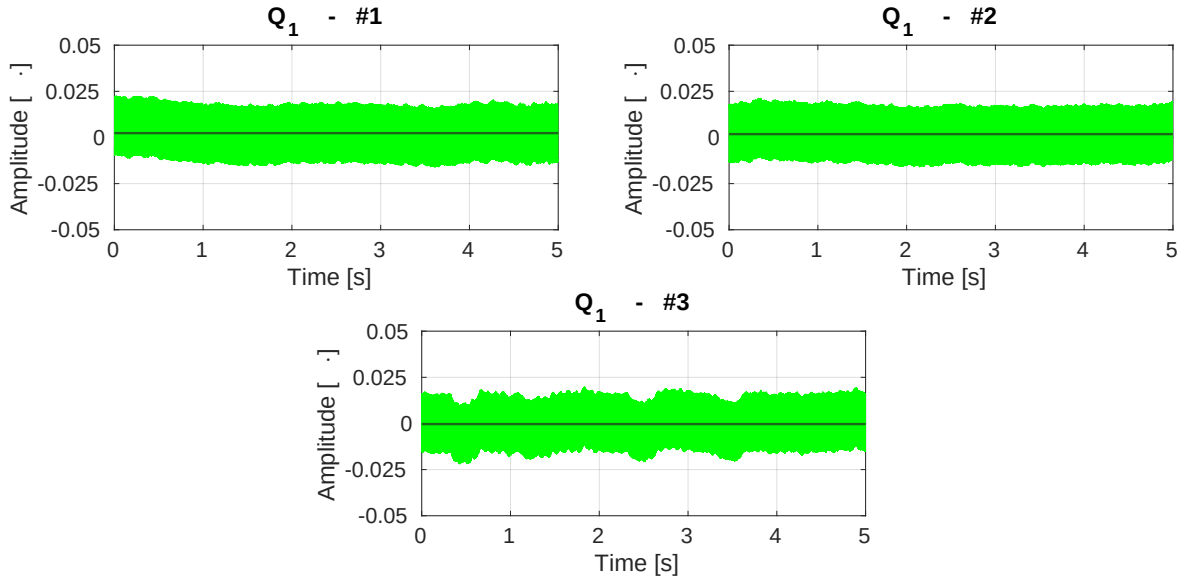


Figure C.12. Plot of the measured data before calibration for Q_1. The line shows the mean value of the data.

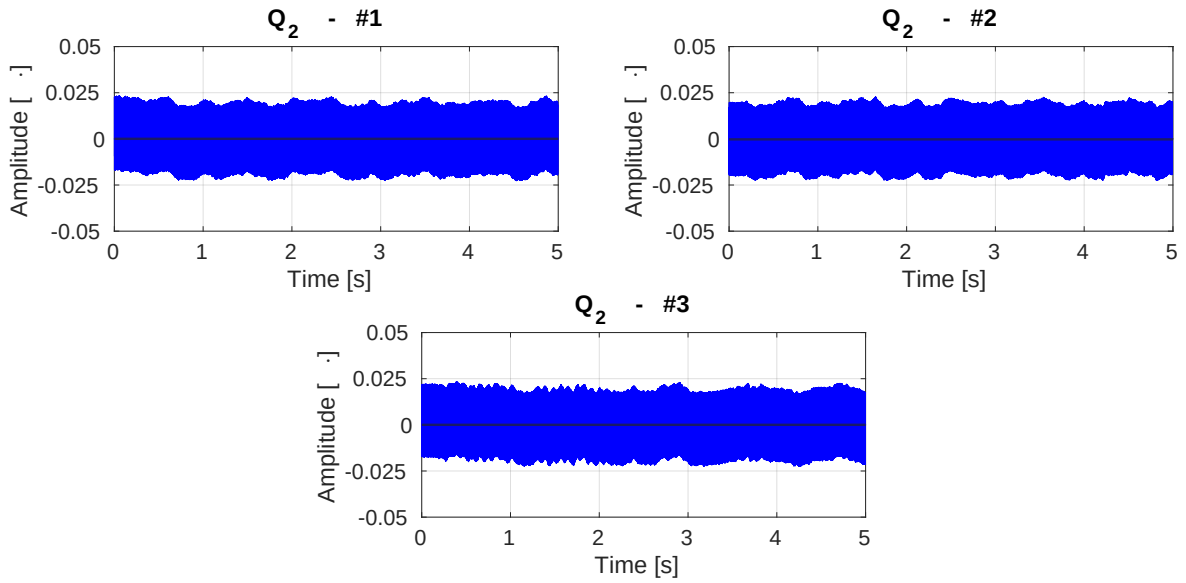


Figure C.13. Plot of the measured data before calibration for Q_2. The line shows the mean value of the data.

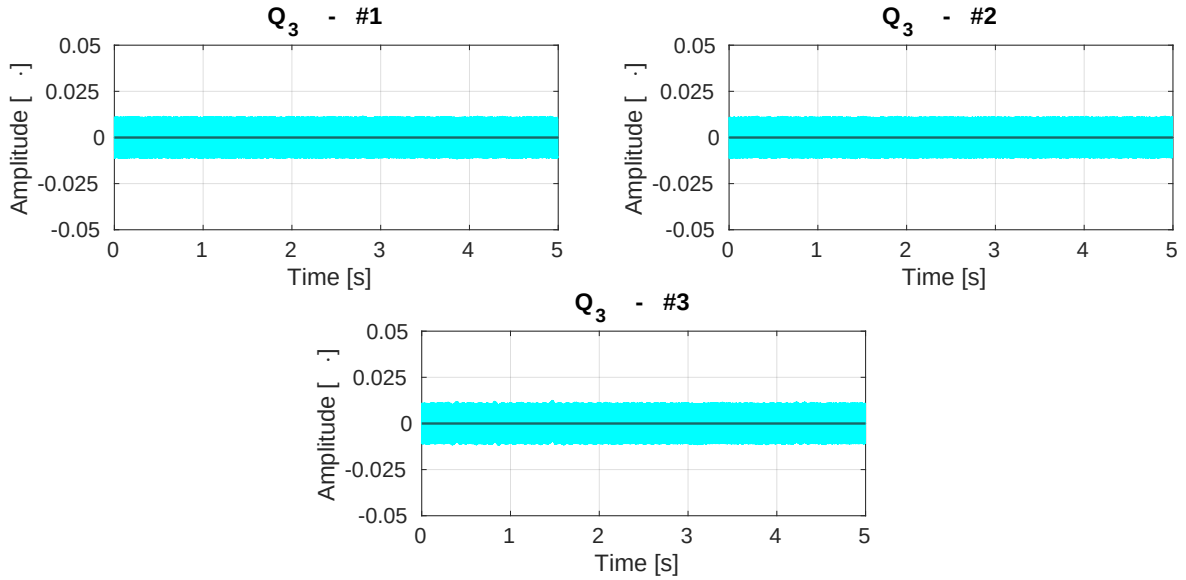


Figure C.14. Plot of the measured data before calibration for Q_3. The line shows the mean value of the data.

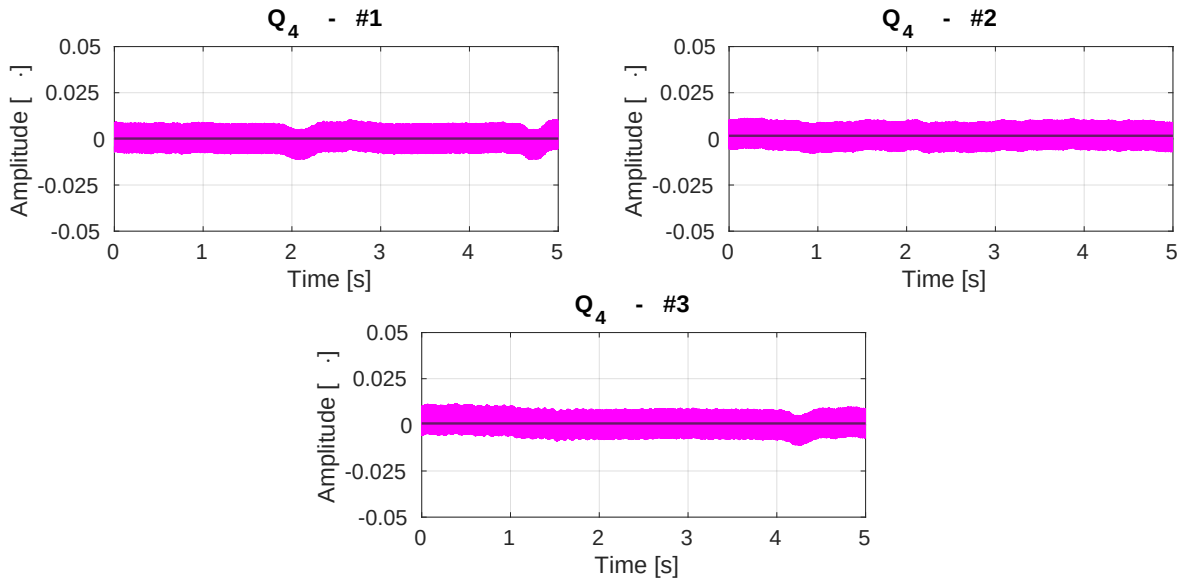


Figure C.15. Plot of the measured data before calibration for Q_4. The line shows the mean value of the data.

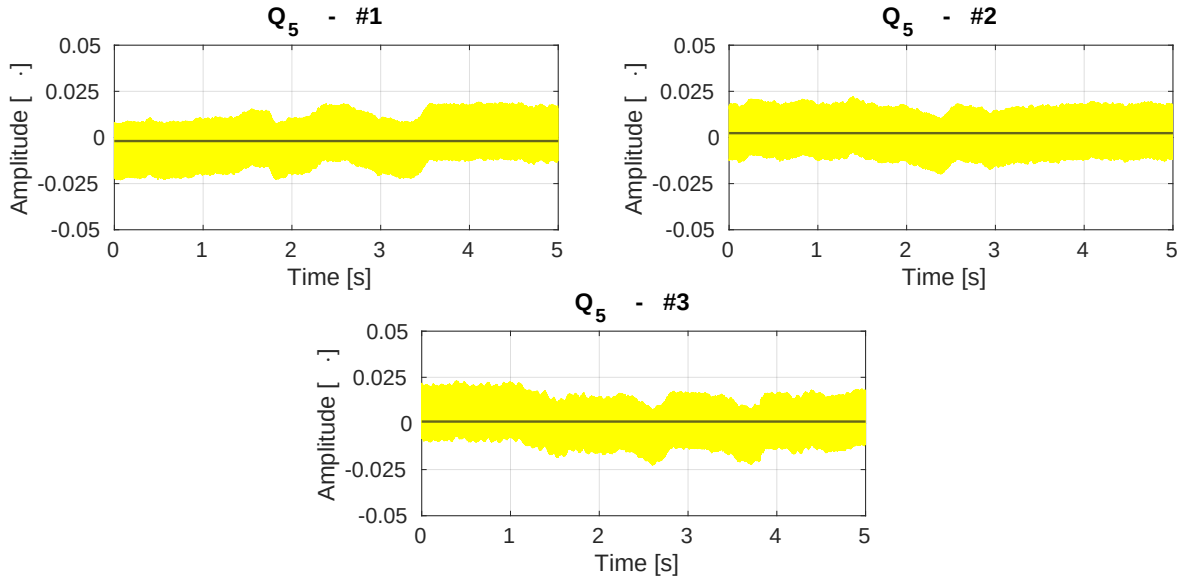


Figure C.16. Plot of the measured data before calibration for Q_5. The line shows the mean value of the data.

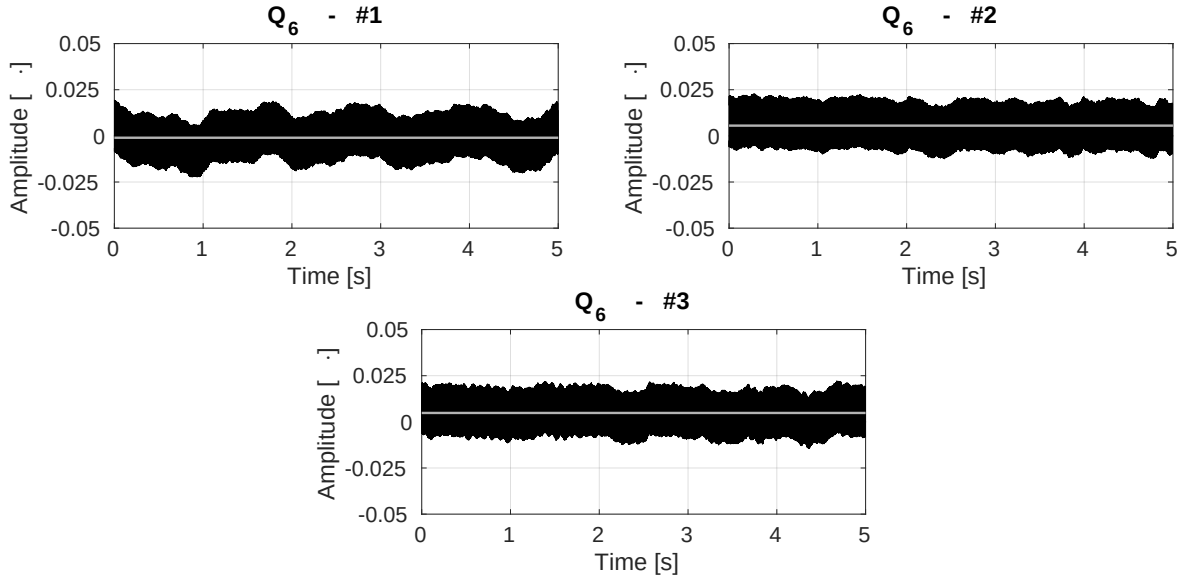


Figure C.17. Plot of the measured data before calibration for Q_6. The line shows the mean value of the data.

The expected shape of the plots is a rectangle, as the amplitude and frequency of the input signal are constant. However, variations in the amplitude occur for many of the microphones (mostly noticeable in Figure C.16 and C.17) that look like low-frequency noise. The exact reason for the variations is unknown but could be due to an oscillatory effect happening within the microphones, as it exposed to a single tone. hmm

The measured mean and power of the microphone data for each of the three runs are seen in Table C.4 together with an average of the runs and a gain factor in order to equalise the power to the microphone to the highest power (Q_2). As the means of the measurements are not constant and not too significant, it is decided to disregard the slight DC-offset.

Table C.4. Measured mean and power of the measurements plotted in the preceding figures. The gain factor is the ratio between the microphone with the highest power (Q_2) and the power of the individual microphone's power. The factor is the square root of the ratio. *micCalibrationLongDist.mlx*

		Q_6	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0
#1	**Mean**	-0.0010	-0.0019	0.0002	-0.0000	0.0000	0.0024	0.0024
	Power (10^{-3})	0.1035	0.1297	0.0326	0.0605	0.1948	0.1309	0.0297
#2	**Mean**	0.0056	0.0023	0.0017	-0.0000	-0.0003	0.0018	0.0016
	Power (10^{-3})	0.1258	0.1190	0.0345	0.0601	0.1927	0.1268	0.0265
#3	**Mean**	0.0048	0.0010	0.0008	-0.0000	0.0000	-0.0003	0.0019
	Power (10^{-3})	0.1184	0.1225	0.0332	0.0599	0.1925	0.1261	0.0274
Avg.	**Mean**	0.0031	0.0005	0.0009	-0.0000	-0.0001	0.0013	0.0019
	Power (10^{-3})	0.1159	0.1237	0.0334	0.0602	**0.1933**	0.1279	0.0279
Gain factor		1.2915	1.2501	2.4048	1.7922	1.0000	1.2293	2.6339

The same procedure is repeated but with the gain factor applied for each microphone within the

ESP32. The measurements are shown below.

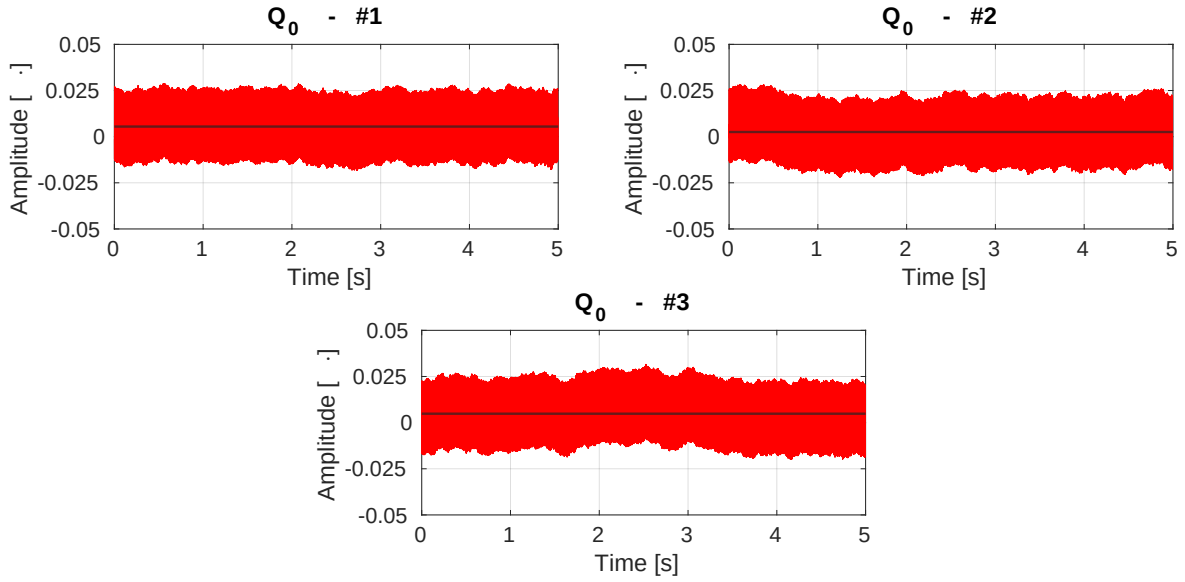


Figure C.18. Plot of the measured data post-calibration to match Q_2's gain for Q_0. The line shows the mean value of the data.

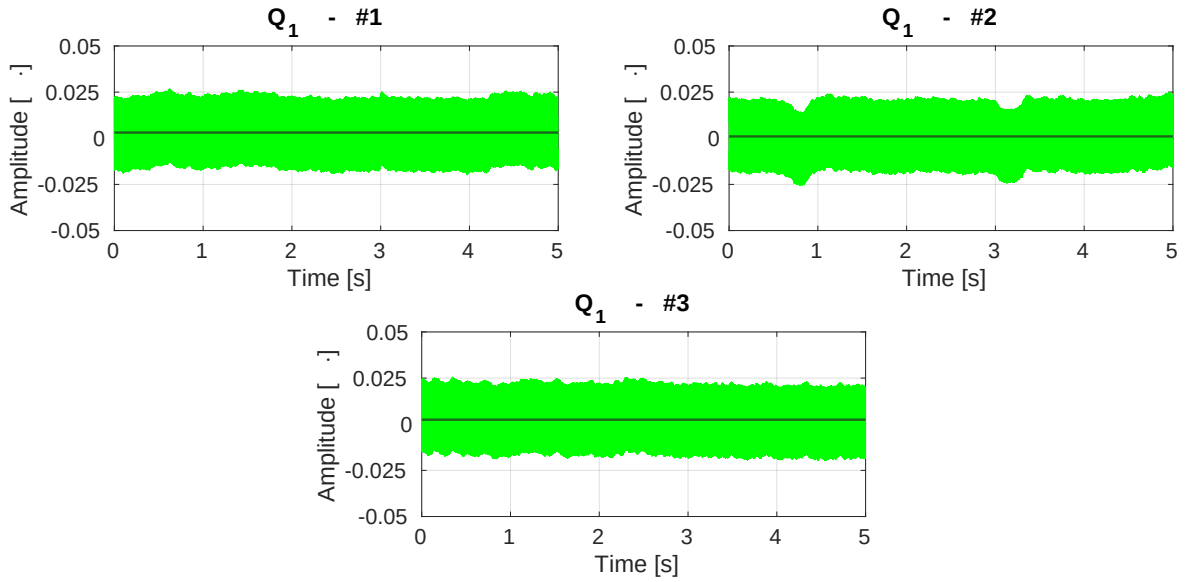


Figure C.19. Plot of the measured data post-calibration to match Q_2's gain for Q_1. The line shows the mean value of the data.

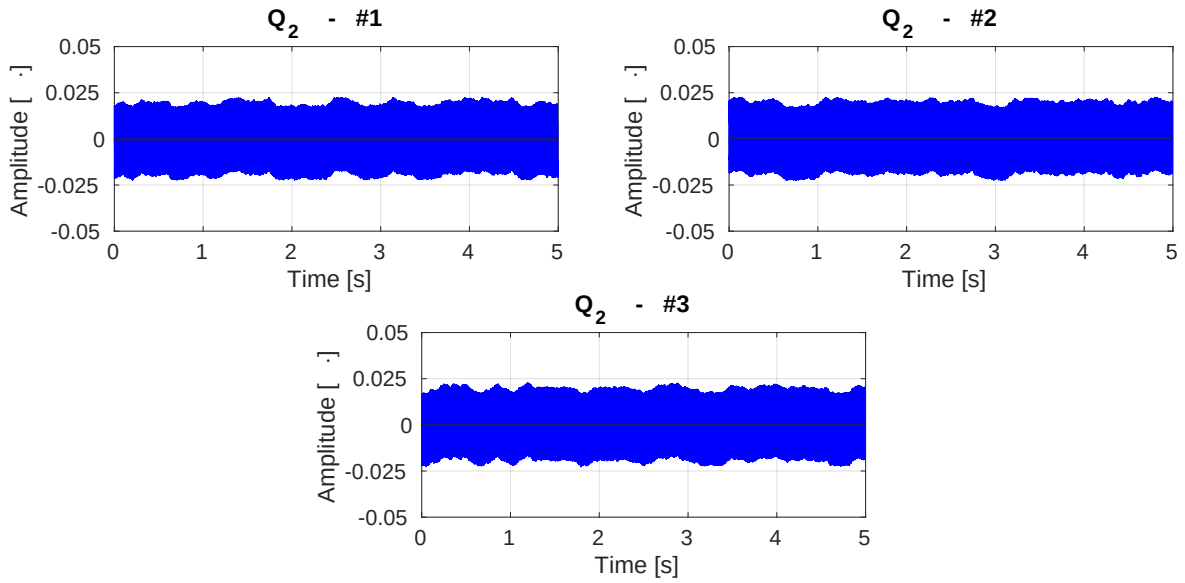


Figure C.20. Plot of the measured data for Q_2 with unchanged gain. The line shows the mean value of the data.

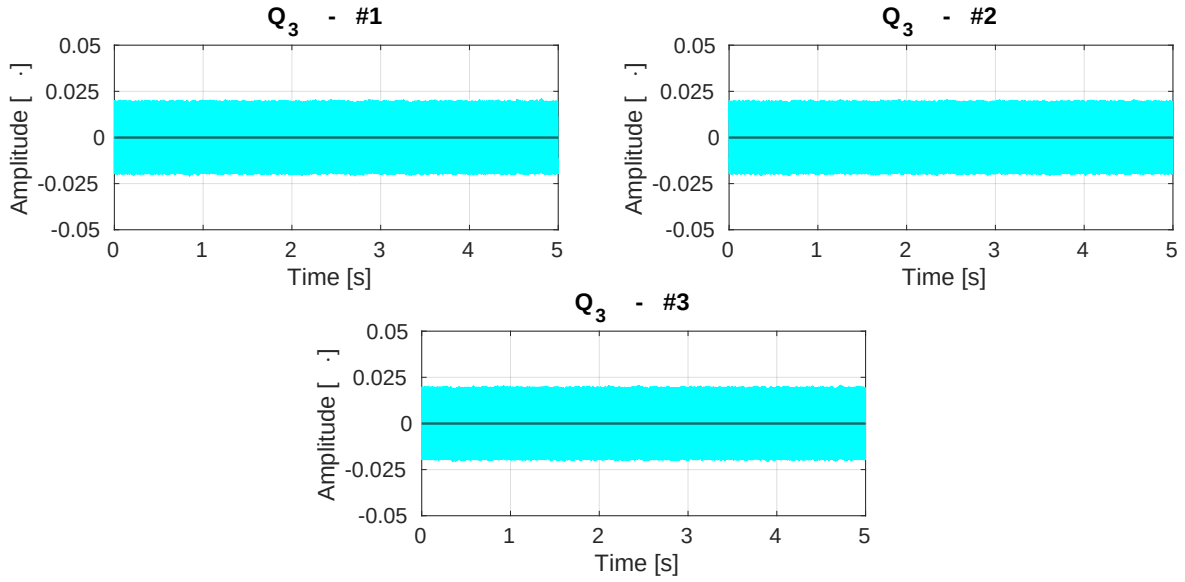


Figure C.21. Plot of the measured data post-calibration to match Q_2's gain for Q_3. The line shows the mean value of the data.

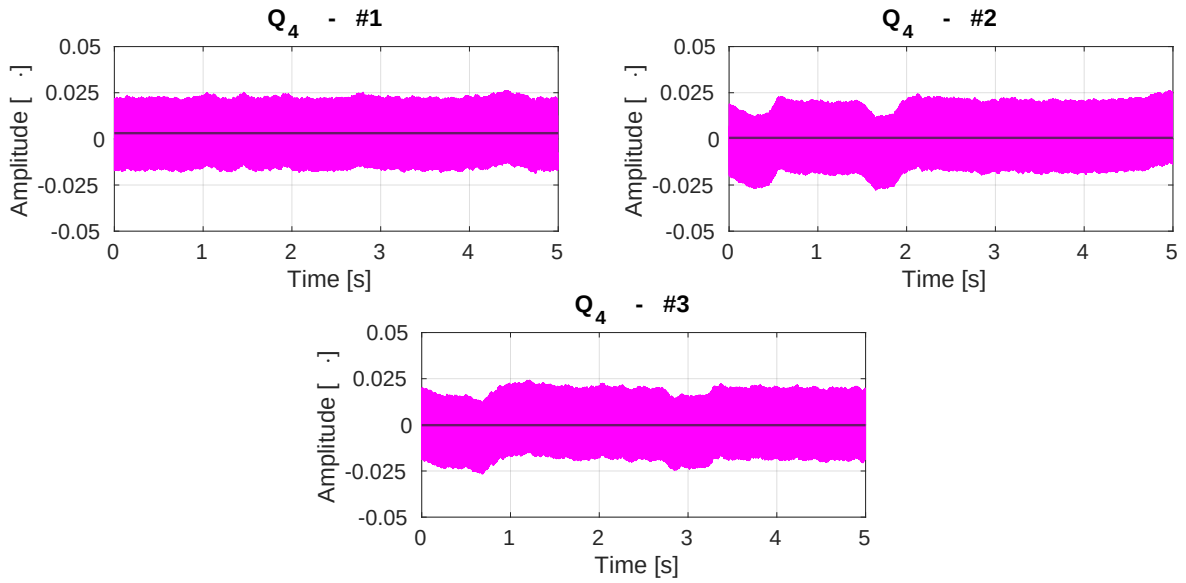


Figure C.22. Plot of the measured data post-calibration to match Q_2's gain for Q_4. The line shows the mean value of the data.

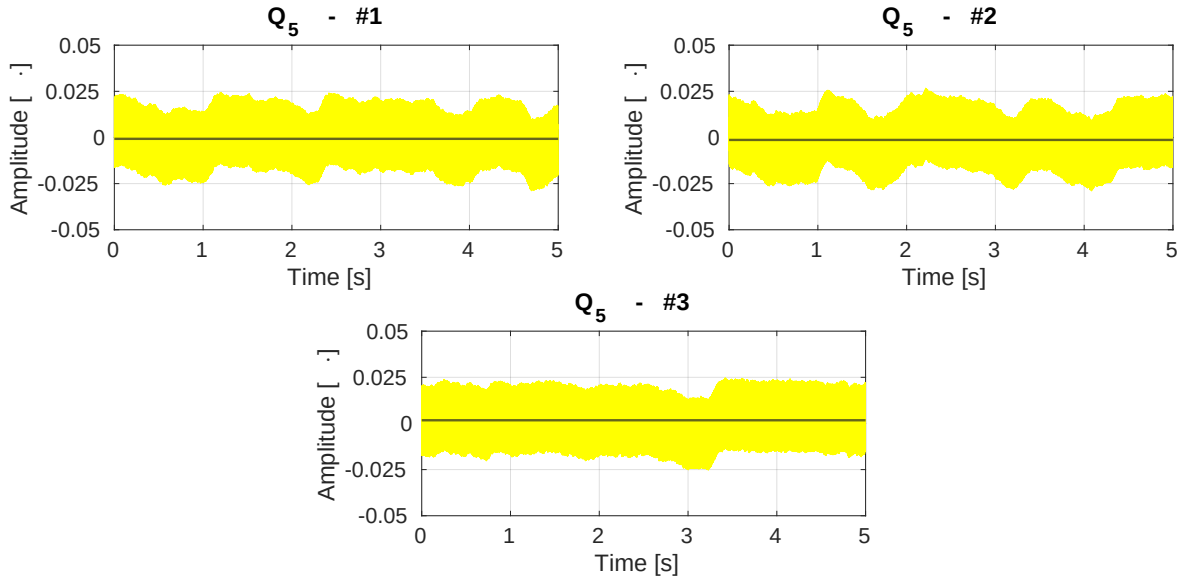


Figure C.23. Plot of the measured data post-calibration to match Q_2's gain for Q_5. The line shows the mean value of the data.

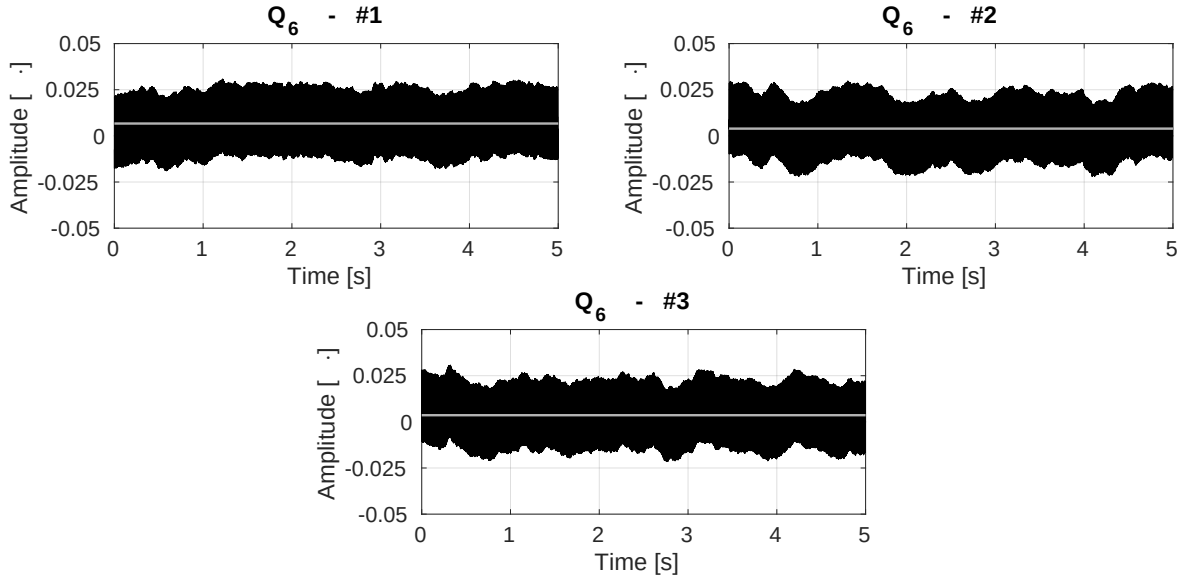


Figure C.24. Plot of the measured data post-calibration to match Q_2's gain for Q_6. The line shows the mean value of the data.

The power of the outputs are now more similar as desired. The measured mean and power post-calibration of the microphone data for each of the three runs are seen in Table C.5.

Table C.5. Measured mean and power of the post-calibration measurements plotted in the preceding figures. *micCalibrationLongDist.mlx*

		Q_6	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0
#1	**Mean**	0.0067	-0.0007	0.0031	-0.0001	-0.0003	0.0031	0.0055
	Power (10^{-3})	0.2338	0.1902	0.1974	0.1914	0.1921	0.2016	0.2231
#2	**Mean**	0.0039	-0.0013	0.0005	-0.0001	0.0004	0.0010	0.0025
	Power (10^{-3})	0.2101	0.1973	0.1960	0.1908	0.1912	0.1942	0.2021
#3	**Mean**	0.0036	0.0017	-0.0002	-0.0001	-0.0000	0.0025	0.0048
	Power (10^{-3})	0.2021	0.1844	0.1909	0.1898	0.1905	0.1966	0.2205
Avg.	**Mean**	0.0047	-0.0001	0.0012	-0.0001	0.0000	0.0022	0.0043
	Power (10^{-3})	0.2154	0.1906	0.1948	0.1907	0.1912	0.1974	0.2152

C.6.2 Beamformer Measurements - ESS

The measurements of the ESS-method are processed with *systemIImpulseResponse.mlx* that estimates the frequency response for each of the input angles. Firstly, cross-correlation is performed on the excitation signal and the output on the data between 0.5 and 1.5 seconds in order to find an alignment such the exact beginning of the sweep in the output signal can be found. Thereafter, the frequency response is estimated with the procedure denoted in Figure C.2. Finally, beampattern plots for chosen frequencies and broadband are generated from the average of the three measurements together with a plot showing the magnitude in the steering angle.

Additionally, equivalent (speed of sound) simulations are generated with *systemSim.mlx* in order to have a comparison for the measurements. The resulting beampatterns from a measurement

through the ESS-method and a steering angle set to 90/270° are seen in Figure C.25.

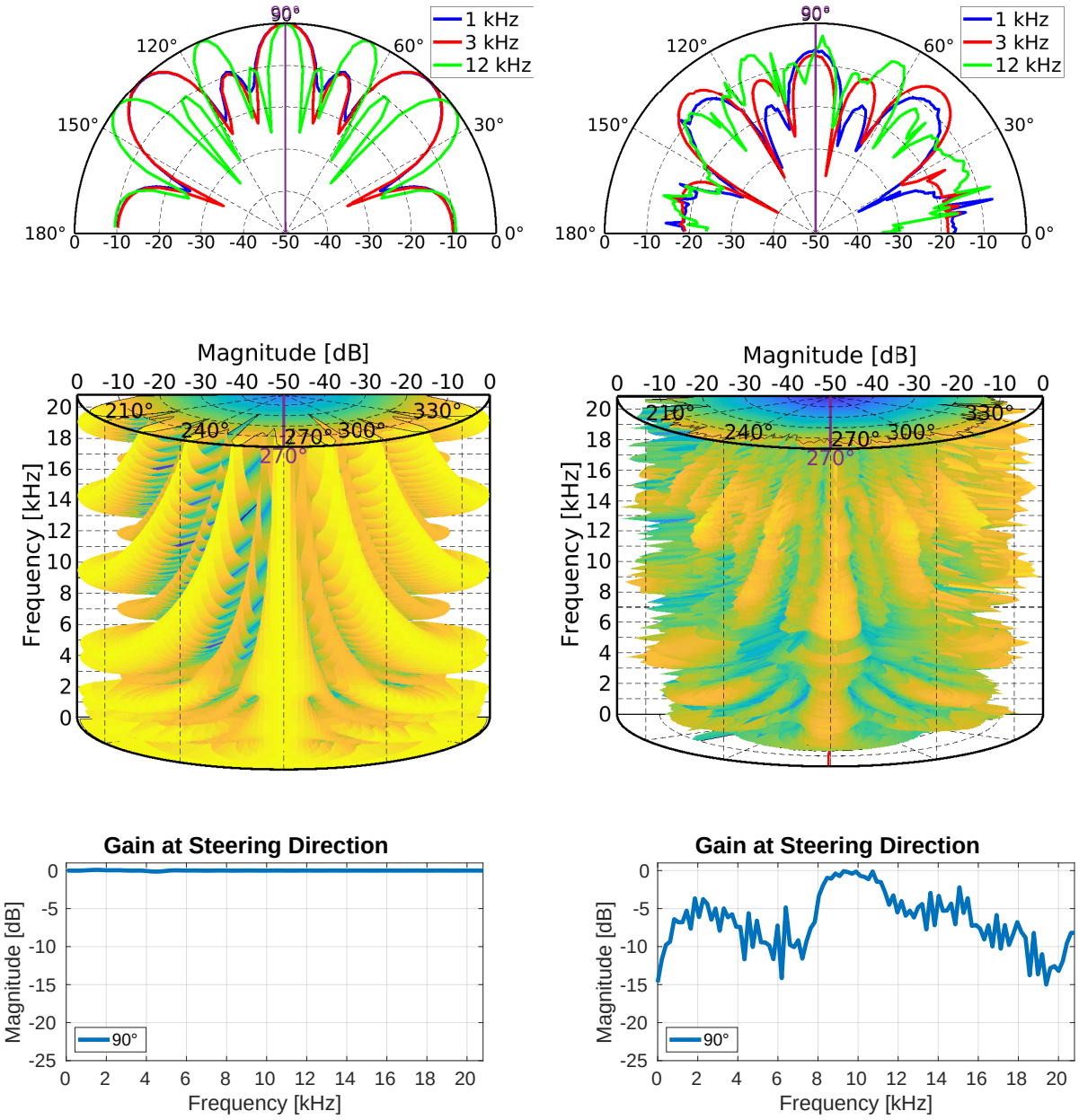


Figure C.25. Comparison plots for the results of the ESS-method for the DAS implementation (right) and a simulated equivalent (left). The steering angle is set to 90/270°.

The beampattern clearly bears resemblance to the simulated but with significant noise and varying suppression at the steering direction. The same measurement is conducted but with a steering angle set to 130/230° in order to examine if the beamformer can steer the main lobe. The processed results are seen in Figure C.26.

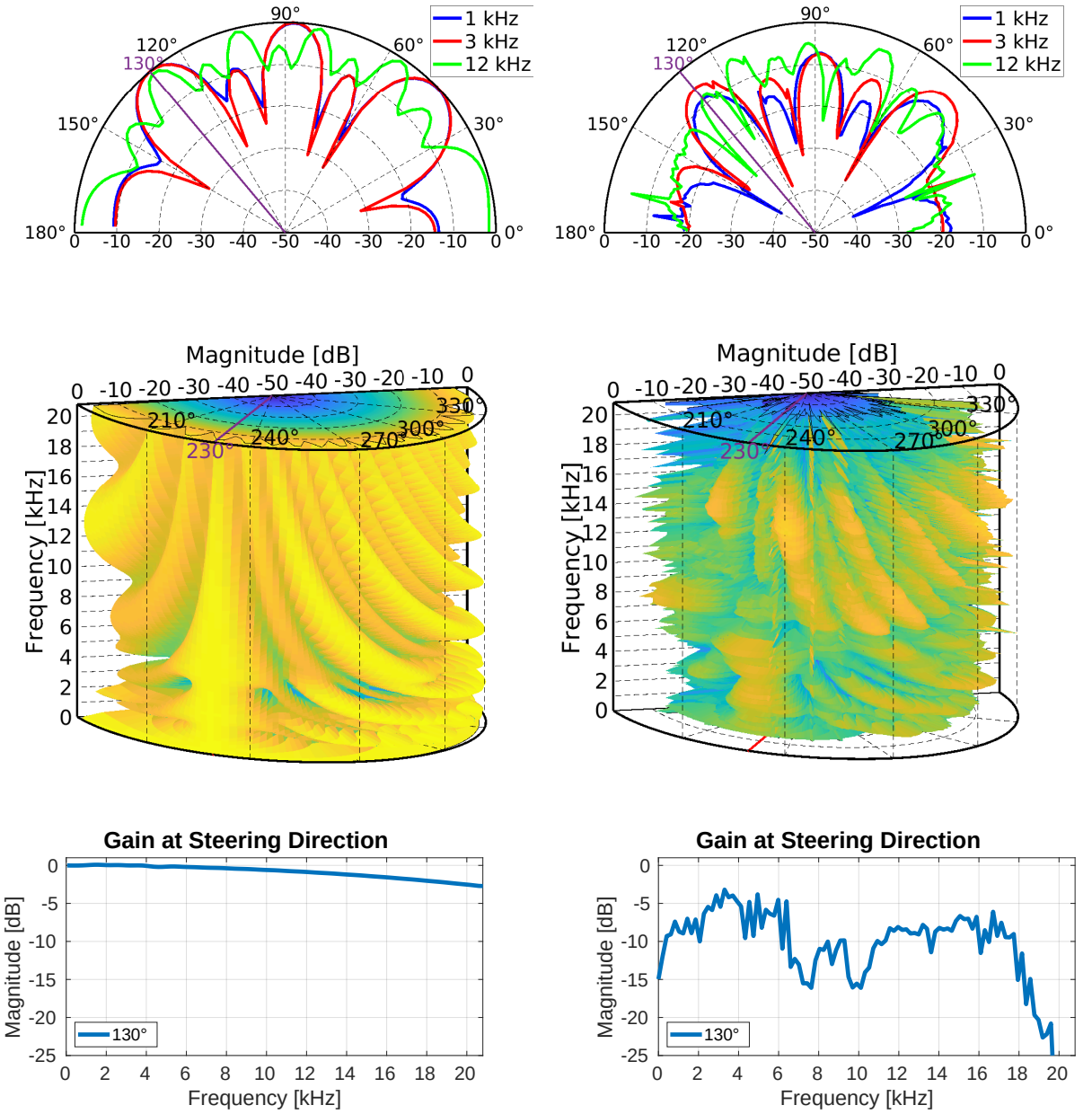


Figure C.26. Comparison plots for the results of the ESS-method for the DAS implementation (right) and a simulated equivalent (left). The steering angle is set to 130/230°.

The beamformer can clearly steer the main lobe in the same manner as the simulated with the same shape. However, the noise is still significant. Therefore, the STT-method is attempted, as it should be less susceptible to noise. The STT consists of many more smaller measurements to make the frequency sweep rather than a single measurement.

C.6.3 Beamformer Measurements - STT

The measurements of the STT-method are processed with *naiveSweepFrequencyResponse.mlx* that calculates the average power for each of the measured frequencies and input angles. The same plots as ESS are then generated.

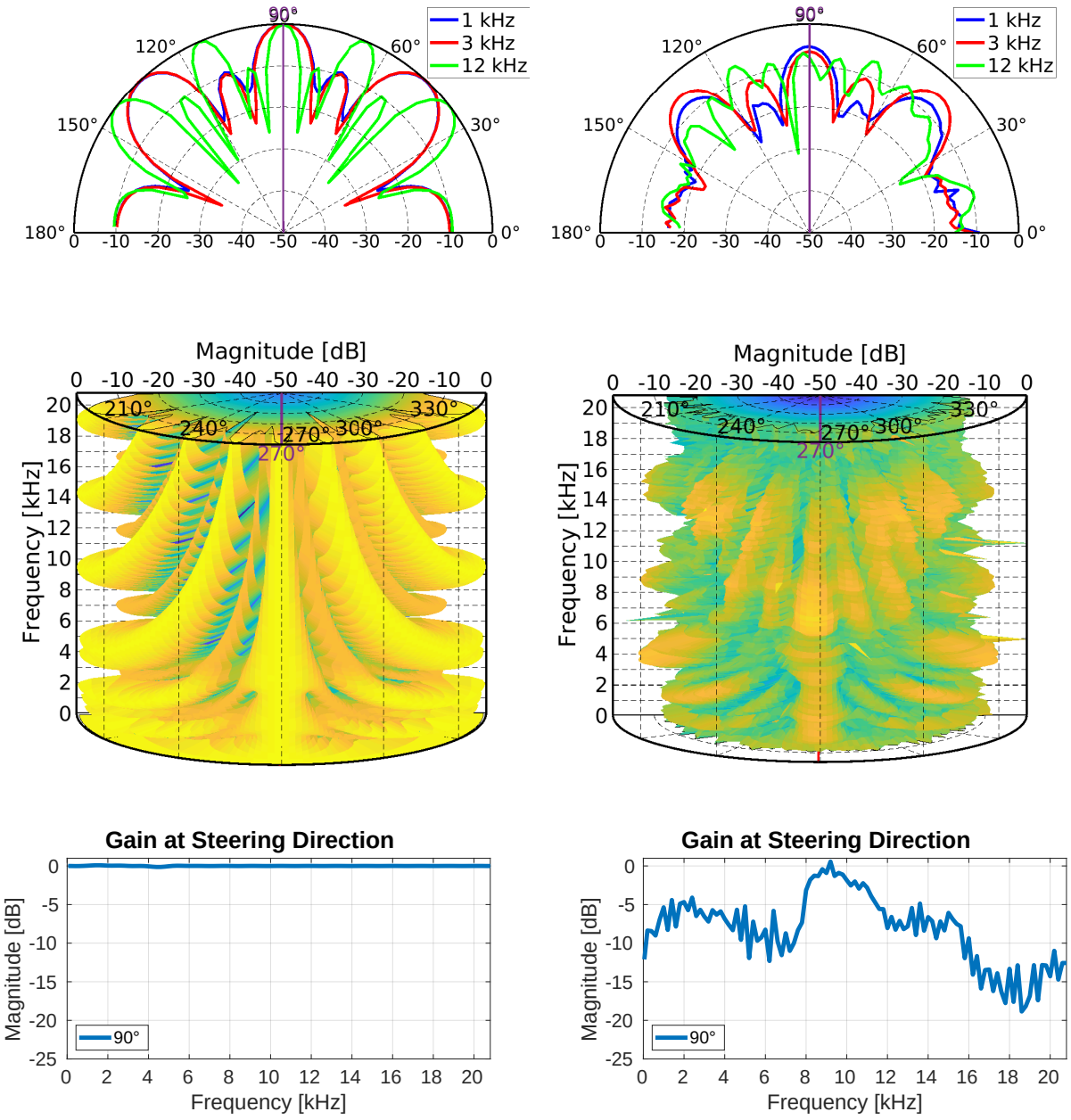


Figure C.27. Comparison plots for the results of the STT-method for the DAS implementation (right) and a simulated equivalent (left). The steering angle is set to 90/270°.

The significance of the noise has been reduced, but spikes are still noticeable and the magnitude in the steering angle is still varying which might be a combination of the frequency response of the speaker and microphones. The same measurement is conducted but with a steering angle set to 130/230°. The results of the measurements are seen in Figure C.26.

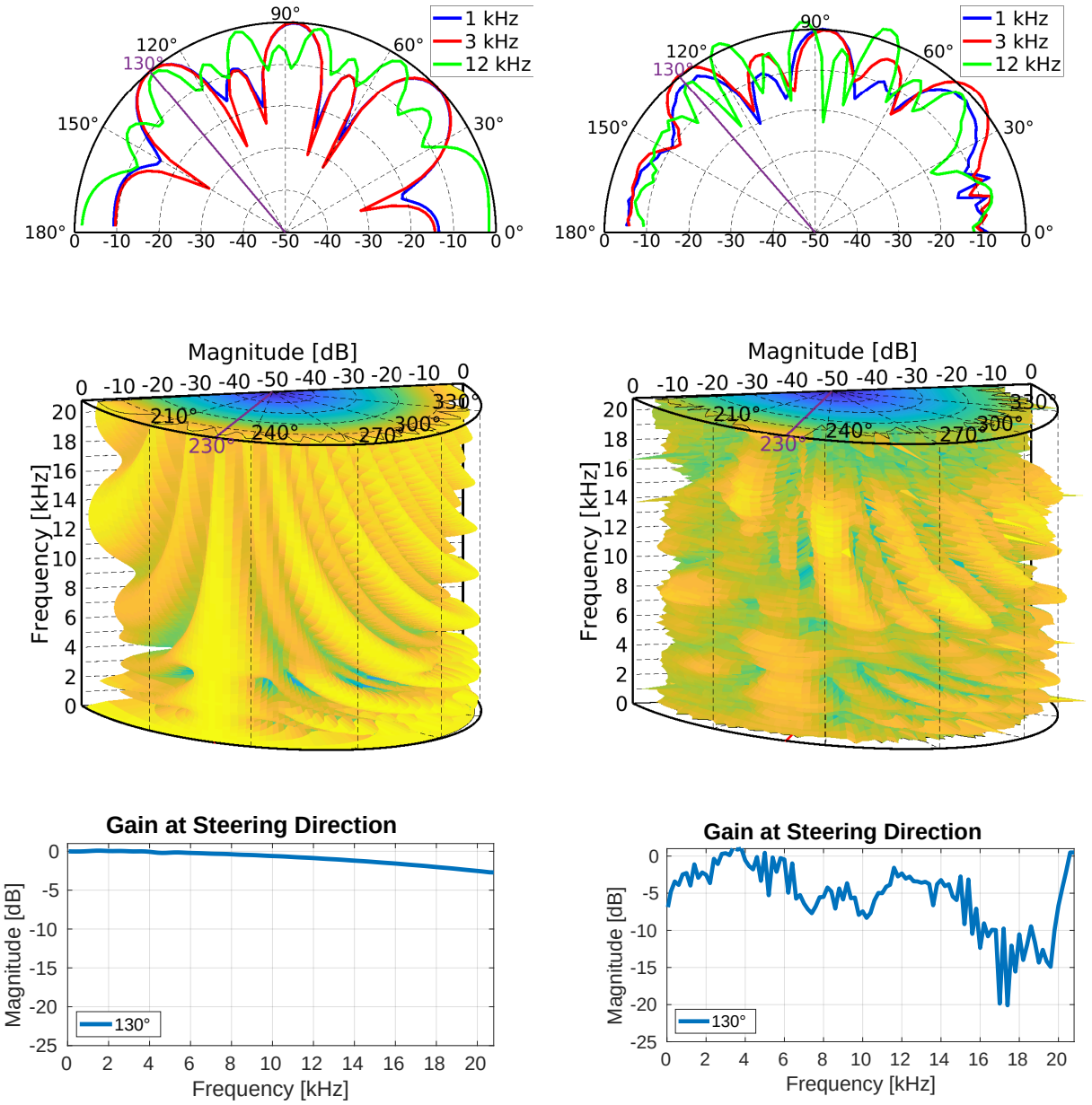


Figure C.28. Comparison plots for the results of the STT-method for the DAS implementation (right) and a simulated equivalent (left). The steering angle is set to 130/230°.

The frequency response of the setup, meaning the combined frequency response of the signal chain from the PC to the centre microphone (Q_3) is measured in order to compare it to the gain at the main lobe.

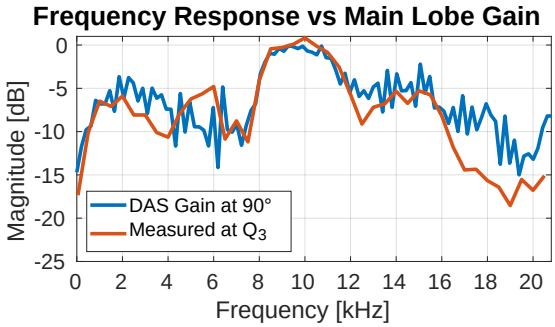


Figure C.29. Measured frequency response for the signal chain (laptop to microphone) measured with Q_3 vs. the gain at the steering direction from Figure C.25.

The frequency responses clearly share the same trend, meaning the varying gain in the steering direction and difference to the simulated are most likely due to the fact that the speaker and microphones were not frequency equalised.

C.7 Static Errors

The microphone array is not placed directly on the turntable's centre of rotation, meaning the distance from the speaker to the centre of the microphone array is not perfectly constant across the angle sweep.

The wavefront of the signal from the speaker are not truly planar even though the microphones are placed 5.3 m from the speaker. The difference in delays are listed in Table C.6.

Table C.6. List of the expected difference in delay denoted in samples for the seven microphones caused by the spherical propagation of the signal from the speaker.

Microphone	Expected AOS [continous samples]
Q_0	2.85
Q_1	0.32
Q_2	0.06
Q_3	0
Q_4	0.06
Q_5	0.32
Q_6	2.85

Additionally, the grating in the anechoic chamber was not removed as the influence was estimated to be minimal.

C.8 Conclusion

The microphones were successfully calibrated to equal sensitivity at 1 kHz. However, some low-frequency noise was evident in the measurements that remained unexplained.

The beampattern of the system was first estimated using an exponential sine sweep with steering angles set to 90/270° and 130/230°. The beampatterns estimated through the measurements turned out similar to the simulated, but influenced by noise and the frequency response of the speaker and microphones.

The same beampatterns were generated by playing a single tone at a time and measuring the power rather than performing a sweep. This reduced the influence of the noise by a margin and made the beampatterns a bit more identical to the simulated.

However, the gain at the steering direction was not constant, as the frequency response of the speaker and microphones were unaccounted for. But the implementation of the delay-and-sum beamformer is validated successfully.