Acoustic Source Localization using a Microphone Array

Project Thesis

Florian Baumgartner
florian.baumgartner@ost.ch

Alain Keller
alain.keller@ost.ch

Advisor
Hannes Badertscher

Examiner
None

Interdisciplinary Center for Artificial Intelligence
Eastern Switzerland University of Applied Sciences

January 2024

Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, ed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Acknowledgement

We take this opportunity to express gratitude to all of the people that supported us in this thesis.

Contents

Acronyms

API Application Programming Interface. 28
ARM Advanced RISC Machine. 26
CD Compact Disc. 22
CODEC Coder-Decoder. 27
CPU Central Processing Unit. 28, 47
DAC Digital-to-Analog Converter. 27
DOA Direction of Arrival. 15, 16
FPGA Field Programmable Gate Array. 8, 37
GNSS Global Navigation Satellite System. 11, 40, 46–48
GPS Global Positioning System. 11
GUI Graphical User Interface. 28
I²C Inter-Integrated Circuit. 23, 27
I²S Inter-IC Sound. 8, 27
IC Integrated Circuit. 27
IDE Integrated Development Environment. 28
IoT Internet of Things. 7
IP Internet Protocol. 47
LED Light-Emitting Diode. 22, 23, 25, 49
LVGL Light and Versatile Graphics Library. 28
MAC Media Access Control. 47
MCU Microcontroller Unit. 8, 22, 26, 40
MEMS Micro-Electro-Mechanical Systems. 7–9, 23
OST Ostschweizer Fachhochschule. 49
PCB Printed Circuit Board. 9, 23–25
PCM Pulse-Code Modulation. 7, 8
PDM Pulse Density Modulation. 7–9, 22–24, 27, 40
PoE Power over Ethernet. 40
PSRAM Pseudo Static Random Access Memory. 26
QR Quick Response. 48
RAM Random Access Memory. 26
RGB Red Green Blue. 22, 25
RTC Real-Time Clock. 22

RTK Real-Time Kinematic. 11

SD Secure Digital Memory Card. 22, 26, 29

SDIO SD Input/Output. 22

SNR Signal-to-Noise Ratio. 23

SoC System on a Chip. 26

SSL Sound Source Localization. 13, 14

TDM Time Division Multiplexing. 10, 27, 40

TDOA Time Differnece of Arrival. 14

TFT Thin-Film Transistor. 22, 25

UI User Interface. 22, 29

URL Uniform Resource Locator. 48

USB Universal Serial Bus. 22, 25, 26, 29, 46

VS Code Visual Studio Code. 28

Glossary

Adafruit Adafruit Industries is an open-source hardware company based in New York City that designs and manufactures electronic development boards. 23, 24

Arduino Is an open-source company providing software libraries and microcontroller kits. 28

1

Introduction

1.1 Background

1.2 Scope

2

Preliminaries

2.1 MEMS Microphones

Micro-Electro-Mechanical Systems (MEMS) microphones represent a significant evolution in acoustic technology. Unlike traditional microphones that rely on larger, more mechanically complex systems, MEMS microphones integrate acoustic sensing elements with electronic circuits on a tiny silicon chip. These microphones have gained immense popularity due to their compact size, robustness, and cost-effectiveness. MEMS technology allows for the production of microphones with high sound quality and excellent reliability, making them ideal for a wide range of applications including mobile devices, wearable technology, and IoT devices. Their small footprint also enables design flexibility in increasingly miniaturized electronic devices. MEMS microphones differ in their output signal types, leading to three categories: analog microphones, PDM microphones, and PCM microphones.

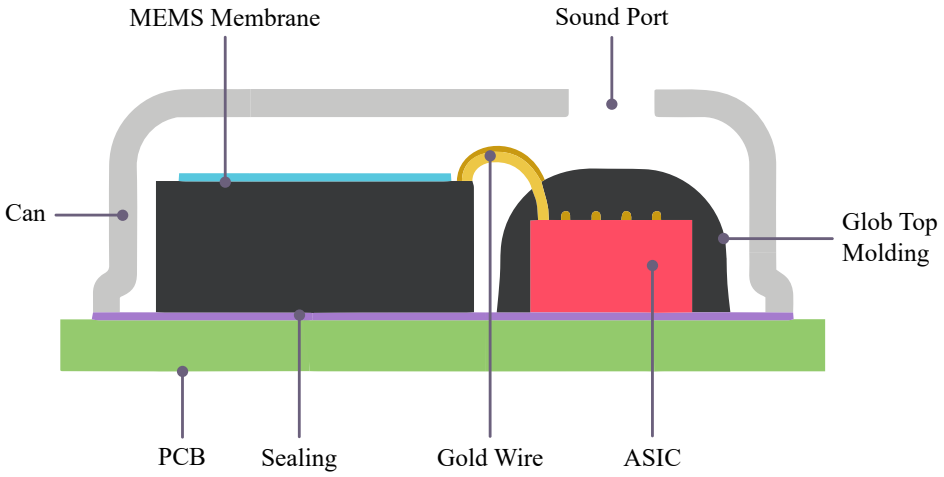


Figure 2.1: Internal structure of a MEMS microphone

2.1.1 Analog Microphones

Analog MEMS microphones convert sound into an analog electrical signal. They are simple and easy to integrate in analog circuits but may require additional signal amplification components. A disadvantage of analog microphones is that they are susceptible to noise and interference, making them unsuitable for long-distance transmission.

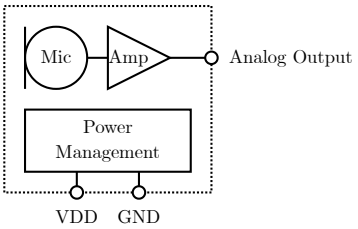


Figure 2.2: Block diagram of analog MEMS microphone

2.1.2 PDM Microphones

PDM microphones output a digital signal representing the acoustic waveform. Their digital format is noise-resistant and supports long-distance transmission, making them ideal for multiplexed, multi-microphone setups. These microphone types require a high-frequency clock signal to operate (typically 1.5 MHz to 3.25 MHz), which must be provided by the host (e.g. a MCU or FPGA). A dedicated channel select pin is used to select the microphone's output channel in multiplexed systems.

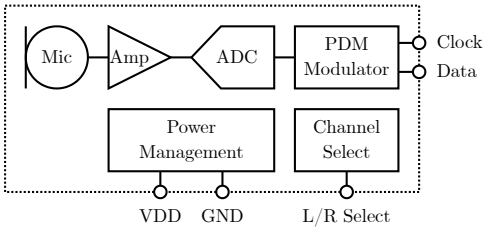


Figure 2.3: Block diagram of PDM MEMS microphone

2.1.3 PCM Microphones

PCM microphones provide a digital output using the Pulse-Code Modulation format, most often interfaced via the I^2S protocol. Compared to PDM microphones, PCM microphones have a build-in decimation filter, which simplifies the signal processing chain, as the host no longer needs to perform this task. However, PCM microphones are more complex, costly and less common in the industry than PDM microphones.

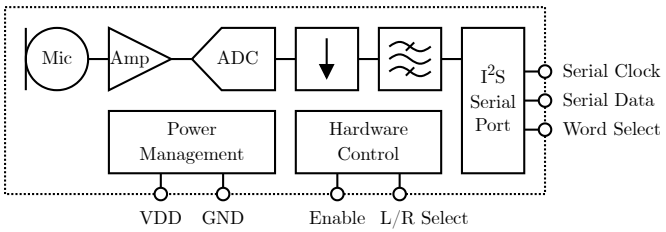


Figure 2.4: Block diagram of PCM MEMS microphone

2.1.4 Microphone Port Location

MEMS microphones can be categorized based on their port location: top-port and bottom-port. Top-port microphones have the sound inlet on the top of the package, suitable when the sound source is above the microphone. Conversely, bottom-port microphones have the inlet at the bottom, ideal for mounting on surfaces where sound comes from the side or below. For bottom-port microphones, the sound must travel through a hole in the PCB to reach the microphone, which can affect the sound quality.

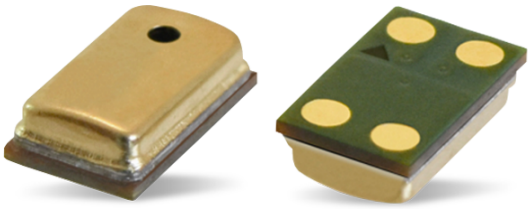


Figure 2.5: Top-port MEMS microphone

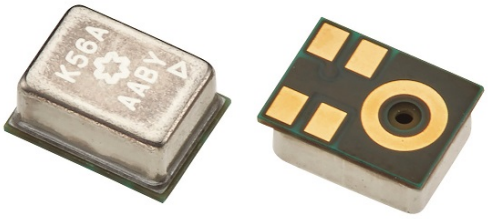


Figure 2.6: Bottom-port MEMS microphone

2.2 Pulse Density Modulation (PDM)

Pulse Density Modulation (PDM) is a modulation technique used to represent an analog signal with a binary sequence. In MEMS microphones, the diaphragm movements modulate a high-frequency carrier signal, resulting in a digital output where the density of pulses corresponds to the amplitude of the input signal. PDM simplifies the microphone design, allowing for smaller and more power-efficient devices. However, it requires a decimation filter in the signal processing chain to convert the high-frequency pulse sequence into a usable digital audio signal. The PDM bitstream is encoded from an analog signal through the process of 1-bit delta-sigma ($\Delta\Sigma$) modulation. This process uses a one-bit quantizer that outputs either a 1 or 0 depending on the amplitude of the analog signal. Due to the nature of real-world analog signals, a quantization error occurs, representing the difference between the 1 or 0 and the actual amplitude it represents. This error is negatively fed back in the $\Delta\Sigma$ process loop, influencing every subsequent quantization measurement and its error. This feedback mechanism averages out the quantization error, enhancing the accuracy of the PDM representation.

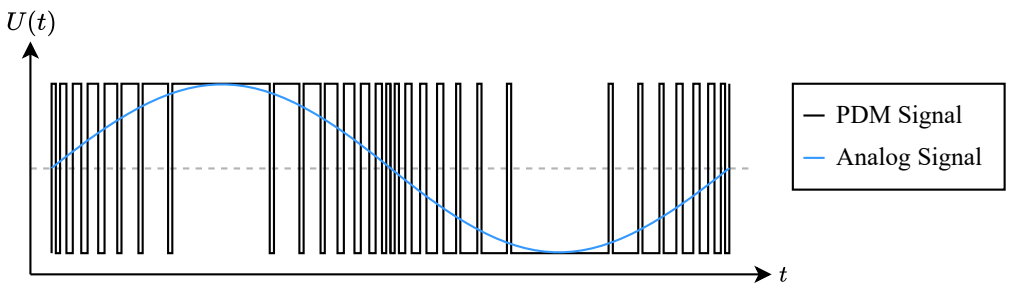


Figure 2.7: PDM Modulation Example

2.3 Time Division Multiplexing (TDM)

Time Division Multiplexing (TDM) is a method of transmitting and receiving independent signals over a common signal path by means of synchronized switches at each end of the transmission line. In the context of digital audio, TDM allows multiple audio streams to share a single communication line, with each stream getting a dedicated time slot. This technique is valuable in systems where multiple audio channels, such as in surround sound systems or multi-microphone arrays, need to be transmitted over a single cable. TDM's main advantage is its ability to efficiently handle multiple audio streams without the need for multiple physical connections. This makes it particularly useful in professional audio applications, broadcast systems, and complex audio setups. However, TDM systems can be more complex to implement and require precise synchronization to ensure that the timing of the different channels is maintained. Figure 2.8 shows an example of a TDM-16 system with 16 bits sample width.

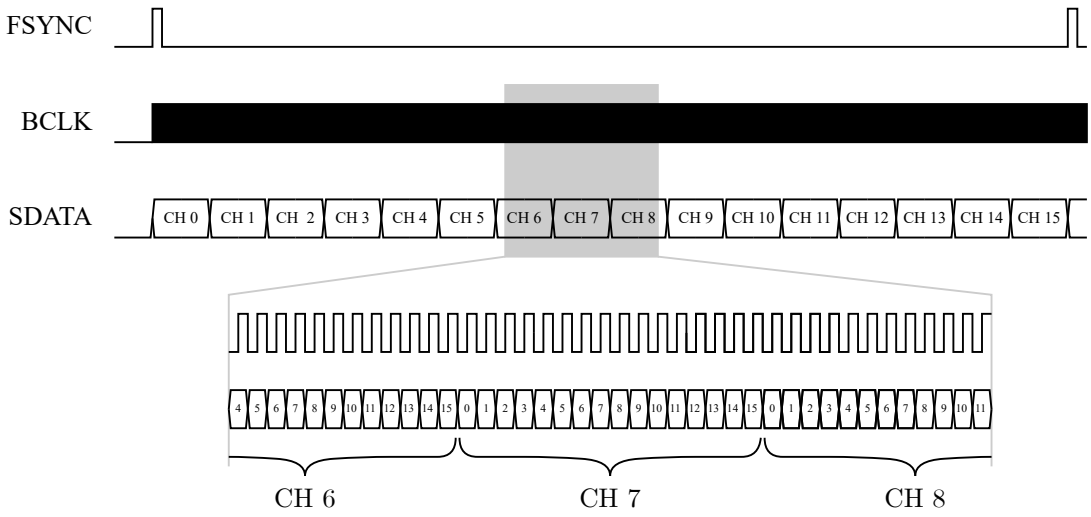


Figure 2.8: Example of TDM-16 with 16 bits sample width

Time Division Multiplexing (TDM) systems introduces tristate outputs, enabling multiple devices to share the same physical bus without interference. The TDM bus consists of three signals: *fsync*, *bclk* and *data*.

- **FSYNC (Frame Sync)**: This signal marks the start of a frame in the TDM stream. It acts as a reference point for aligning the data across all channels. In a TDM-16 system, FSYNC indicates the beginning of a sequence of 16 channels (slots).

- **BCLK (Bit Clock)**: This clock signal dictates the timing for data transmission. Each edge of the BCLK corresponds to a bit position in the data stream. For 16-bit samples, there are 16 BCLK pulses per channel, hence 256 BCLK pulses per frame.

- **Data**: The actual audio data is transmitted in a series of bits. In a 16-channel, 16-bit TDM system, each channel's data is represented by 16 bits (Left Justified).

2.4 GNSS Real-Time Kinematic (RTK)

The Global Navigation Satellite System (GNSS) Real-Time Kinematic (RTK) is an advanced technique used in geodesy and navigation that enhances the precision of position data derived from satellite-based positioning systems. RTK uses differential techniques to improve the accuracy of position information obtained from satellite systems like GPS, GLONASS, Galileo, or BeiDou. This method is particularly useful in applications requiring high precision, such as surveying, construction, and precision agriculture.

2.4.1 Custom Node as an Anchor

One method to utilize RTK for achieving enhanced position accuracy involves using a custom node as an anchor. This method requires setting up a fixed RTK base station at a known location. The base station calculates error values for satellite signals by comparing the expected signal (based on its known location) with the received signal. These error values, which account for various factors like atmospheric interference and satellite orbit errors, are then transmitted to a mobile RTK receiver. The mobile receiver uses these corrections to adjust its own satellite signal measurements, significantly improving its positional accuracy.

2.4.2 Using Data from a Public Provider

Alternatively, RTK corrections can be obtained from a public provider. In this approach, the user does not set up a custom anchor node but instead subscribes to a service that provides RTK correction data. These services operate a network of RTK base stations and compute correction factors for different regions. The correction data is typically transmitted via the internet directly to the user's RTK receiver. This method is convenient for users who do not have the resources to establish and maintain their own RTK base station. By applying these corrections, which include compensating for atmospheric drift and other errors, the user's receiver can achieve a similar level of enhanced positional accuracy as with a custom anchor node.

3

Sound Source Localization

3.1 Physical background

This section shows some underling fundamentals on which the following described theory is based on. Schribe

3.2 Sound Source Localization Methods

Sound Source Localization (SSL) is a well researched area with many applications. [1] The basic system can be brought into two categories, time based or power based methods.

3.2.1 Power based SSL

The idea of power based SSL is based of the known propagation properties of sound waves in air. BlaBlaBla However for this method to work some properties of the sound source have to be known. Additionally the sensors that measure the sound power levels have to be calibrated ... short text

3.2.2 Time Based SSL

Another group of SSL is based on the time when the signal is received by the microphones. Given a source at a location $\boldsymbol{S} = (x_S, y_S)^T$ and N microphones at locations $\boldsymbol{M_n} = (x_n, y_n)^T$ the time it takes for a acoustic signal from the source to reach a microphone is

$$t_n = \frac{\|\boldsymbol{S} - \boldsymbol{M_n}\|}{c} = \frac{\sqrt{(x_S - x_n) + (y_S - y_n)}}{c}. \tag{3.1}$$

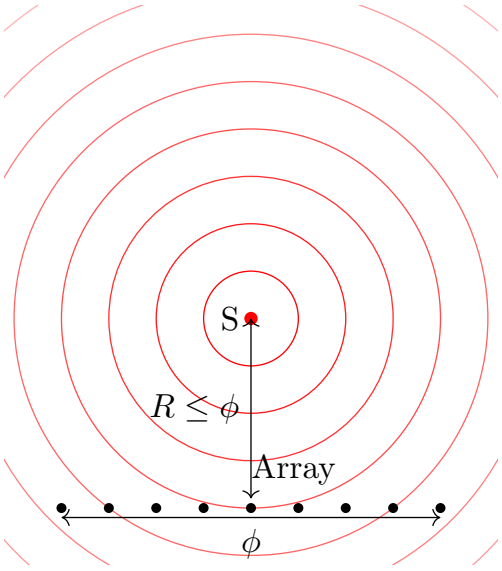
So if t_n is known for a microphone, the location of the source can be limited to point on a circle around M_n with a radius of $t_n c$. Given three or more microphones the intersection of these circles will show th location of the source. However in many cases this approach is not realistic since t_n is generally not know. It would require some sort of a synchronization between the microphones and the sources.

Near-Field

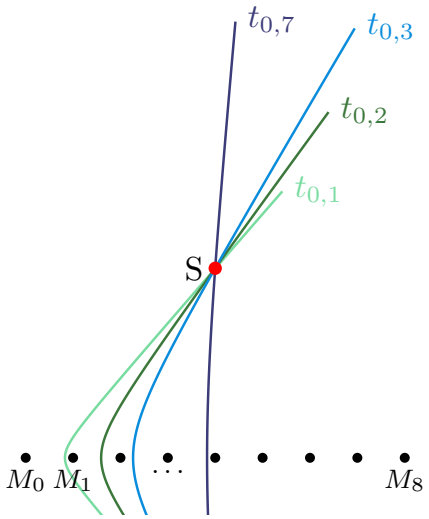
The next time property that could be used is the Time Differnece of Arrival (TDOA) which between two microphones $\boldsymbol{M_n}$ and $\boldsymbol{M_m}$ is defined as

$$t_{n,m} = t_m - t_n = \frac{\|\boldsymbol{S} - \boldsymbol{M_m}\| - \|\boldsymbol{S} - \boldsymbol{M_n}\|}{c}. \tag{3.2}$$

With this equation $\boldsymbol{S}$ can be interpreted as the set of points that lie on a hyperbola which fixed points are the Microphones and the vertices are $ct_{n,m}$m apart. To find the location of $\boldsymbol{S}$ a minimum of four microphones is now needed. This approach works only if we can assume that the curvature of the incoming sound waves at the microphones is big enough. Figure 3.1a shows such a arrangement which is generally known as the near-field scenario. It is generally said, that the near-field assumption holds when the distance from the source to the array isn't much greater than the size of the array. In Figure 3.1b it can be seen how the resulting hyperbolas from the TDOA intersect at the point $\boldsymbol{S}$.



(a) Acoustic model used for the near field SSL.



(b) A selection of the hyperbolas resulting from the time-delays $t_{m,n}$.

Figure 3.1: Near field SSL with a linear microphone Array. Since the microphones are distributed on a straight line a degree of freedom is lost and $\boldsymbol{S}$ mirrored at this line is also a possible solution.

Far-Field

When the source is much further away than the size of the Array, as depicted in Figure 3.2, the curvature of the sound wave at the array is negligible. This is generally known as the far-field case. Now the sound waves aren't modeled as spherical waves but rather as planar waves. Given such a planar wave only the direction in which the source is placed can be determined with the TDOAs.

Decission why this

In this thesis the focus is set on the far-field case, due

literature

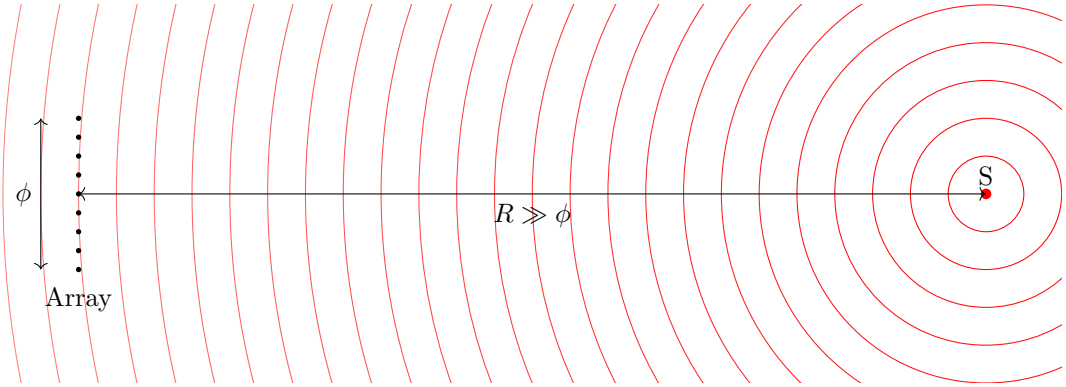


Figure 3.2: Far-Field case. The Curvature of the sound-waves from S at the array is almost zero.

3.2.3 Direction of arrival estimation

Lots of research has been made on how to estimate the Direction of Arrival (DOA) of a source with appropriate sensors.

brief liera-ture study

In this thesis the focus was set on far-field techniques considering that the goal is to detect drones which are usually several meters above ground.

3.2.4 Beamforming

Beamforming is a popular method for estimating the DOA of a signal.

To better show the fundamental principles of beamforming an example is used where the goal isn't to estimate a DOA but to send a signal in a desired direction using microphones. Figure 3.3 shows such a case in $\mathbb{R}^2$ where a linear array of N sound sources beam a sine wave $x(t) = \sin(\omega_0 t)$ in the direction of φ. This is achieved by adding a phase shift to the sine wave at each microphone resulting in $x_n(t) = sin(\omega_0 t + \omega_0 t_n)$. When the phase shifts are properly selected the points where the waves positively interfere form a beam. Given this example with a narrow band signal, the signal at each source can be written as

$$x_n(t) = \Re(sin(\omega_0 t)e^{j\omega t_n}) \tag{3.3}$$

and more generalized as

$$X(t,\varphi) = \Re\left(sin(\omega_0 t) \underbrace{\begin{pmatrix} e^{-j\omega_0 t_0(\varphi)} \\ e^{-j\omega_0 t_1(\varphi)} \\ \vdots \\ e^{-j\omega_0 t_{N-1}(\varphi)} \end{pmatrix}}_{W(\varphi)} \right). \tag{3.4}$$

The vector $W(\varphi)$ is commonly known as the steering vector. It can be seen that the steering vector is dependent on ω_0 and will therefore only have the desired effect on signals with frequencies close to ω_0. How to use beamforming with wider band signals will be shown later in this chapter.

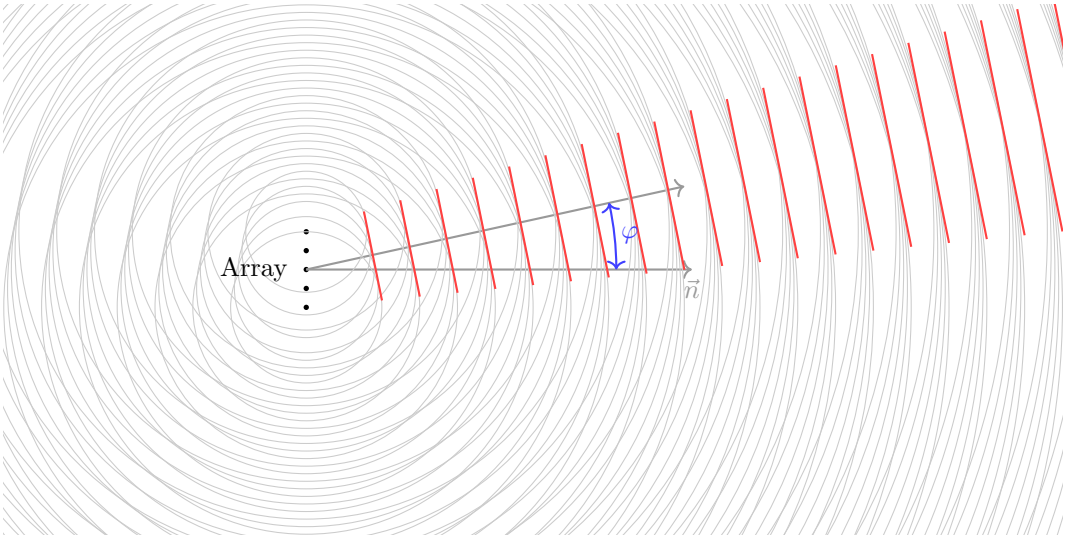


Figure 3.3: Phase differences between the array elements create positive interference at the red lines. The gray circles correspond the waves' local maxima

If the goal is to estimate the direction of a sound signal with microphones $x_n(t)$ is now defined as the signal measured by the nth microphone. Now (3.4) is rewritten to

$$\boldsymbol{Y(t,\varphi)} = \underbrace{\begin{pmatrix} x_0(t) \\ x_1(t) \\ \vdots \\ x_{N-1}(t) \end{pmatrix}}_{X(t)} \odot \underbrace{\begin{pmatrix} e^{-j\omega_0 t_0(\varphi)} \\ e^{-j\omega_0 t_1(\varphi)} \\ \vdots \\ e^{-j\omega_0 t_{N-1}(\varphi)} \end{pmatrix}}_{W(\varphi)} \tag{3.5}$$

where depending on the method used either $Y(t,\varphi)$ is further processed, or the method directly uses $X(t)$ for the DOA estimation.

Steering Vector

The calculation of the Steering Vector is dependent on different factors. The two main factor can be broken down to the geometry of the microphone array and the direction in which the source lies. Again the formulas are derived for a circular array in $\mathbb{R}^2$ and are then expanded into $\mathbb{R}^3$. A circular array where all the microphones are placed on a circle is convenient for the use of polar coordinates and the formulas can also be used for any other array described in polar coordinates.

Figure 3.4 shows such an array with five equally spaced microphones M_n. The goal of the steering vector is to delay the measured signal at each microphone so that they each have the same phase and therefore positively interfere. To calculate these delays a reference point must firstly be defined, here it is the center of the circle. Because a far-field model is used the sound pressure level is equal along lines perpendicular to its propagation direction. With the magenta line as the reference line, each measured signal from the microphones must be phase shifted with

$$\omega_0 t_n = \omega_0 \frac{\|\boldsymbol{d}_n(\varphi_s)\|}{c} \hat{\boldsymbol{d}}_n(\varphi_s) \cdot \hat{\boldsymbol{d}}_s. \tag{3.6}$$

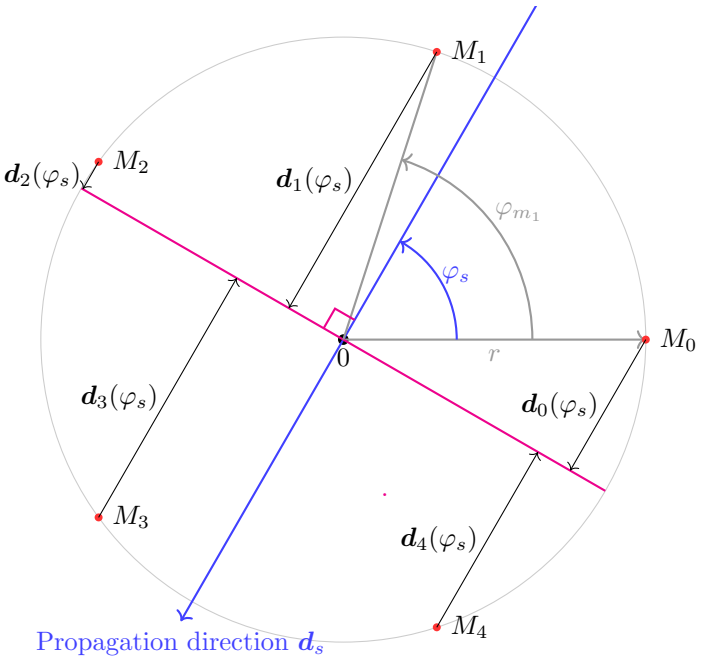


Figure 3.4: BalaBula

The Term $\hat{\boldsymbol{d}}_n(\varphi_s) \cdot \hat{\boldsymbol{d}}_s$ is needed to determine if a phase shift is positive or negative. Using the trigonometric properties of a right triangle

$$\|\boldsymbol{d}_n(\varphi_s)\|\hat{\boldsymbol{d}}_n(\varphi_s) \cdot \hat{\boldsymbol{d}}_s = r_{m_n} \cos(\varphi_s - \varphi_{m_n}) \tag{3.7}$$

and therefore

$$t_n = r_{m_n} \cos(\varphi_s - \varphi_{m_n}) \frac{1}{c}. \tag{3.8}$$

This formula can be used for any microphone with polar coordinates (φ_{m_n}, r_{m_n}) and can therefore be used for any array configuration described in polar coordinates.

To expand this idea into $\mathbb{R}^3$ the spherical coordinate system is now used. This adds the angle θ as the inclination. When $\theta = \pi/2$ it is the same situation that has already been looked at. But as θ approaches 0 the speed of the wavefront projected onto the surface of the array increases. This adds a term to (3.8)

$$t_n = r_{m_n} \cos(\varphi_s - \varphi_{m_n}) \frac{\omega \sin(\theta)}{c}. \tag{3.9}$$

Delay and Sum Beamformer

The basic beamformer is the delay and sum beamformer

$$y(t, \varphi, \theta) = \underbrace{\begin{pmatrix} x_0(t) \\ x_1(t) \\ \vdots \\ x_{N-1}(t) \end{pmatrix}}_{X(t)} \cdot \underbrace{\begin{pmatrix} e^{-j\omega_0 t_0(\varphi,\theta)} \\ e^{-j\omega_0 t_1(\varphi,\theta)} \\ \vdots \\ e^{-j\omega_0 t_{N-1}(\varphi,\theta)} \end{pmatrix}}_{W(\varphi,\theta)}. \tag{3.10}$$

Notice how instead of the Hadamard product, the scalar product is now used. $y(t, \varphi, \theta)$ is therefor the sum of the delayed signals. If the angles of the steering vector point

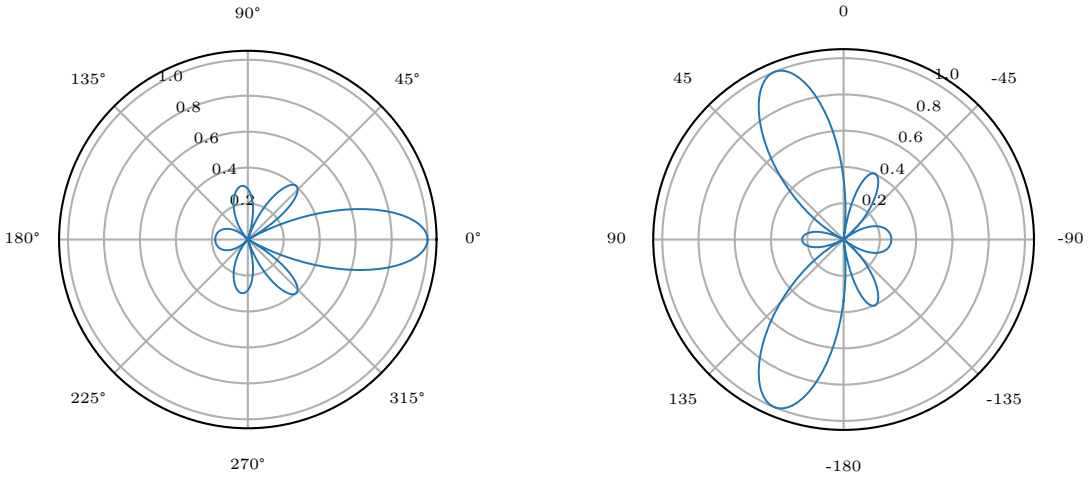
towards a source its signal will be amplified. By using the delays calculated for the steering vector $X(t)$ can be calculated to simulate a source in a specific direction

$$X(t, \varphi_s, \theta_s) = e^{j\omega_0 t} \begin{pmatrix} e^{j\omega_0 t_0(\varphi_s, \theta_s)} \\ e^{j\omega_0 t_1(\varphi_s, \theta_s)} \\ \vdots \\ e^{j\omega_0 t_{N-1}(\varphi_s, \theta_s)} \end{pmatrix} \tag{3.11}$$

The amplitude response of an array using delay and sum beamforming can now be calculated with

$$G(\varphi, \theta) = \frac{|X(0, \varphi_s, \theta_s) \cdot W(\varphi, \theta)|}{N}. \tag{3.12}$$

The response is normalized with the amount of microphones to ensure a maximum gain of 1. In Figure 3.5 two such responses with can be seen. The main lobe of $G(\varphi, \theta)$ clearly shows in the direction of the source however there are also side lobes pointing to directions without sources.



(a) $G(\varphi, \theta, \omega)$ with $\theta = \pi/2$ for $\omega_0 = 2\pi 1000$. The source is placed in the direction of $(\varphi = 0, \theta = \pi/2)$

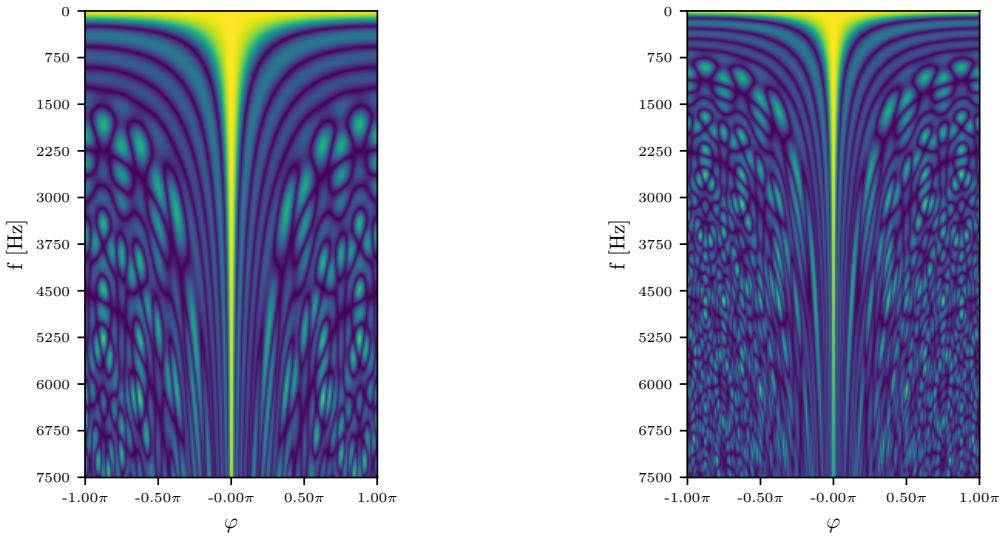
(b) $G(\varphi, \theta, \omega)$ with $\varphi = 0$ for for $\omega_0 = 2\pi 1000$. $(\varphi = 0, \theta = -45°)$ is the same as $(\varphi = 180°, \theta = 45°)$. The mirror along the horizontal axis comes from the flat geometry of the array which makes it impossible to distinguish if a sound comes from θ or $180 + \theta$.

Figure 3.5: Delay and Sum Beamformer response for a circular array with $N = 16$ microphones and a radius $r = 0.25$m.

Beamforming can be seen as spatial sampling, where the microphones are the sampling-points. So some of the concepts used in digital sampling such as the samplingtheorem and the uncertainty principle can also occur in some way in beamforming. Figure 3.6 shows $G(\varphi, \pi/2)$ for different frequencies. The array used in Figure 3.6a is the same array as in Figure 3.5 whereas Figure 3.6a uses a circular array with $r = 0.5$m and also 16 microphones. It can be seen that at around 1500hz more side lobes with a high amplitude start to appear. With this array geometry the smallest distance between two microphones is $\approx 0.25\pi/8 \approx 0.098$m, which is the wavelength of a sound wave with a frequency of approximately 3500hz. So the effect seen in Figure 3.5 is a kind of aliasing. This results from ambiguity in the phases of the steering vector. Ideally the phase shift between two neighboring microphones is unique for each angle of the steering vector.

However when they are more than $\lambda/2$ apart, phase shifts start to repeat for different steering directions.

Another effect seen in Figure 3.5 is the increase of the main lobe width when the frequency approaches 0. This is consequence of a too small array. In the used array the maximal distance between two microphones is 0.5m. For lower frequencies the phase difference between the first microphone and the last microphone in the array is too small to get destructive interferences when the steering vector is not focused on the source.



(a) Salat **(b)** Gurkensaft

Figure 3.6: Delay and Sum Beamformer response for a circular array with $N = 16$ microphones and a radius $r = 0.25$m.

Until now the formulas are only useful for narrow band signals. This theory is now expanded to be applicable for wide band signals. The narrow band delay and sum beamformer has the property, that it behaves the same in the frequency domain

$$e^{-j\omega_0 t_n} x_n(t) = \mathscr{F}^{-1}(e^{-j\omega_0 t_n} \mathscr{F}(x_n(t))) \tag{3.13}$$

since it's just a multiplication with a scalar,. The summation could also be done in the frequency domain due to the linearity of the Fourier transform. However in the frequency domain a IIR Filter can be constructed, to make a different phase shift for every frequency

$$Y_n(t) = \mathscr{F}^{-1}(e^{-j\omega t_n} X_n(\omega)). \tag{3.14}$$

By using a discrete fourier transform the beamformer can be written in matrix form

Indexchaos lösen

$$Y(\varphi, \theta) = \begin{pmatrix} X_0(\omega_l) & \dots & X_0(\omega_h) \\ \vdots & \ddots & \vdots \\ X_{N-1}(\omega_l) & \dots & X_{N-1}(\omega_h) \end{pmatrix} \odot \begin{pmatrix} e^{-j\omega_l t_0} & \dots & e^{-j\omega_h t_0} \\ \vdots & \ddots & \vdots \\ e^{-j\omega_l t_{N-1}} & \dots & e^{-j\omega_h t_{N-1}} \end{pmatrix}, \tag{3.15}$$

where ω_l is the lowest and ω_h the highest frequency of interest.

3.2.5 Other beamforming techniques

3.3 Simulator

In order to experiment with different microphone arrangements and algorithms a simulator was developed. Since the simulator wasn't the main focus of this Thesis, its functionality was kept simple. The simulator lets you place acoustic sources and microphones in a $\mathbb{R}^3$ space and calculates the measured signals at each microphone.

3.3.1 Simulation Model

For sake of simplicity the sources were modeled as omnidirectional point-sources and the microphones are omnidirectional as well. The Sound Pressure Level of a source is defined at one meter distance from their position and decreases squarely with the distance. The perceived sound at any point P can now be described as

$$y(P,t) = \sum_{I} \frac{x_i(t - \|P - S_i\|/c)}{\|P - S_i\|}. \tag{3.16}$$

Where S_i is the position of the nth Source and $x_i(t)$ is its output sound. Since the simulation is done numerically and with pre recorded audio files with a given samplingrate f_s (3.16) has to be discretized. Simply replacing $x_i(t - \|P - S_i\|/c)$ with $x_i(n - f_s\|P - S_i\|/c)$ with $n \in \mathbb{N}$ doesn't work because $f_s\|P - S_i\|/c \notin \mathbb{N}$ in most cases. To implement this $f_s\|P - S_i\|/c$ is rounded to its nearest integer d_{P,S_i}. Now the delayed signal is $x_i(n - d_{P,S_i})$. To achieve sub sample delays this signal is then filtered with a Fractional Delay FIR Filter with a delay of $f_s\|P - S_i\|/c - d_{P,S_i}$.

Reflections

wiauimeer

The simulator also allows the reflective surfaces to be placed into the room. These surfaces are defined with and have a infinite size. Additionally a dampening factor can be defined which defines how much a reflecting sound signal is damped when getting reflected. By this stage the simulator can only handle single reflective path. An already reflected sound signal can not be reflected by a second surface.

4

Acquisition-System Design

4.1 Overview

The first step after understanding the basic theory behind microphone arrays and beamforming is to apply this knowledge in a practical setting. The primary goal is to record and analyze real-world audio data from a variety of microphones. This involves the evaluation of different microphone types and array configurations. Understanding these differences is critical for further development and refinement of beamforming algorithms.

The approach extended to the development of a specialized hardware system, necessary for recording multiple audio channels simultaneously. This capability was not found in existing hardware solutions, leading to the design and development of a new system, supporting up to 32 microphones. The channel limit was chosen due to practical constraints while keeping the system complexity manageable.

Once the audio data is captured, the next phase involves applying algorithms to analyze these recordings. Applying these algorithms to the captured audio data facilitates a detailed comparison between real-world microphone performance and theoretical simulation results. This comparison is substantial for understanding the differences between practical microphone use and simulated scenarios, thus being crucial for further algorithm development and refinement.

In addition to its primary purpose, this system enables possibilities for various other applications where recording a large number of microphones is needed. Its ability to handle multiple channels simultaneously and processing them in real-time makes it a versatile tool for various use cases.

The subsequent sections describe the microphone evaluation and the development process, including hardware and firmware design of the audio acquisition system.

4.2 Key Requirements

The goal of the acquisition system is to provide a flexible microphone recording infrastructure to easily aquiring audio signals from multiple microphones.

The following key requirements have been set:

- Simultaneous recording of up to 32 microphone channels
- High-quality audio recording with 16-bit resolution and 44.1 kHz sampling rate (CD-Quality)
- Recording to a removable SD-Card in lossless WAV format
- Real-time monitoring of individual microphone channels
- Easy to use UI for configuration and operation
- Compact and portable design to enable mobile use

4.3 Key Decisions

The following section describes the key decisions made during the development of the acquisition system.

1. **MCU Selection**: As a main Microcontroller Unit (MCU) the `Teensy 4.1` was chosen due to its ability providing two TDM-16 audio interfaces, enabling support for up to 32 audio channels. Its computational performance and extensive software support in audio applications were key factors in this decision. Additionally, the `Teensy 4.1` includes a fast SDIO SD-Card interface with a built-in card holder, ideal for this application.
2. **Microphones**: Preference was given to PDM microphones due to their wide availability and suitability for use with longer cables, in comparison to other microphones types mentioned in section 2.1.
3. **Power Source**: The system is powered via a single USB cable to ensure portability and ease of use in various settings, adhering to the requirement for a compact and mobile design.
4. **Ethernet Port**: An RJ45 ethernet port was added for future development opportunities, such as streaming audio data over ethernet.
5. **Touch Display**: A touch TFT display was integrated to offer an easy-to-use UI, facilitating efficient system configuration, operation, and real-time monitoring.
6. **RGB LEDs**: RGB LEDs were employed for visual feedback on the audio levels of each microphone channel.
7. **Headphone Jack**: The addition of a headphone jack allows for real-time auditory monitoring of individual microphone channels, essential for troubleshooting.
8. **Real-Time Clock (RTC)**: An RTC was integrated to tag each recording with the current time and date, simplifying the comparison of measurement results in post.

4.4 Microphone Evaluation

Although a variety of microphone technologies exists, such as condenser, dynamic, and electret, the focus has been set on MEMS microphones due to several compelling reasons. Primarily, their widespread availability and ease of manufacturing, allowing PCB manufacturers to assemble them, make MEMS microphones a practical choice. Their compact size is advantageous in space-constrained applications, while the integrated analog frontend simplifies audio system design. Notably, MEMS microphones deliver excellent audio quality and wide bandwidth, essential for high quality sound reproduction. Moreover, their cost-effectiveness makes them suitable for microphone arrays with a large number of channels.

4.4.1 Microphone Types

In the evaluation process, four different MEMS microphones were selected. Two of them are top-ported, while the other two are bottom-ported. All four microphone types are available in large quantities at *JLCPCB*, a popular PCB manufacturer. Comparing the datasheets of the different microphones, only small differences in there specifications were found. Consequently, it was all the more intriguing to compare these microphones and to determine effective differences in audio quality and Signal-to-Noise Ratio (SNR).

Microphone Type	**Port**	**Manufacturer**	**Similar Types**
MP34DT05TR-A	Top	ST Microelectronics	-
GMA4030H11-F26	Top	INGHAi	*Knowles SPK0415HM4H-B-7*
GMA3526H10-B26	Bottom	INGHAi	*Knowles SPH0641LU4H-1*
SD18OB261-060	Bottom	Goertek	-

Table 4.1: Evaluated MEMS Microphone Types

4.4.2 Microphone Breakout Boards

To test the four evaluated microphone types, a two-layer carrier PCB was designed and manufactured. A panalized design was chosen to simplify the manufacturing process and reduce assembly costs. One panel consists of 8 breakout boards for each microphone type, resulting in a total of 32 breakout boards per panel. In total, 5 panels were manufactured, which leads to 40 microphone breakout boards per type. Each breakout board has a size of 14.0 mm x 22.0 mm and is separated by a V-groove, a common technique used in PCB manufacturing. This allows the individual breakout boards to be easily separated from each other.

Every microphone board is equipped with a slide switch for the channel selection, allowing to toggle between the left and right PDM channel. Additionally, a dual-color LED is integrated, serving as a power indicator. It lights up red if the right channel is selected or white if the left channel is selected. To enhance the ease of mounting, each board includes a threaded surface-mounted standoff nut for an M3 screw, enabling straightforward attachment to a frame.

For connectivity, a standard SH 4-Pin JST [1] connector with a 1 mm pin pitch was used. This connector type is commonly found in Adafruit *STEMMA QT / Qwiic JST* I^2C

[1] JST refers to Japan Solderless Terminal, a leading manufacturer of a diverse range of connectors, including wire-to-board, board-to-board, and wire-to-wire types.

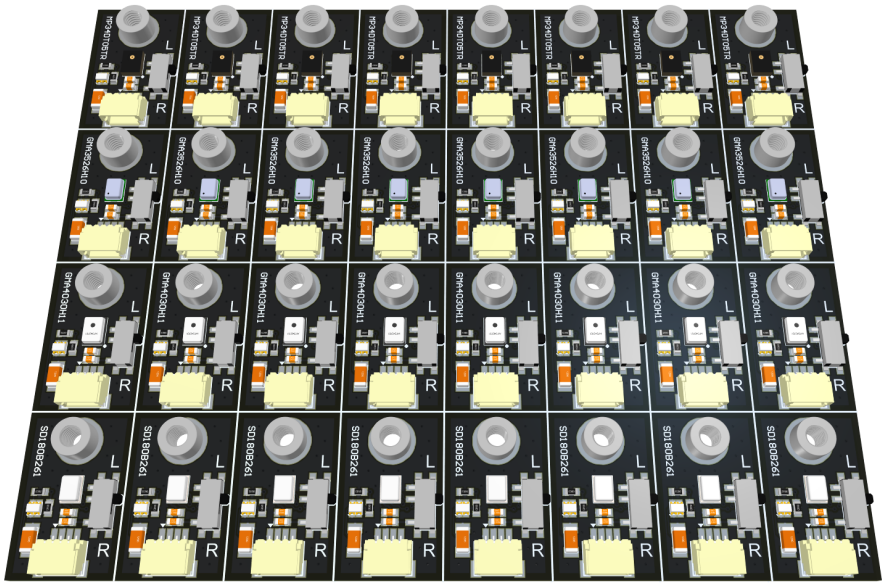


Figure 4.1: Front View of the Microphone Breakout-Boards

accessories boards. By using this specific type, a wide range of pre-assembled cables with different lengths are available, simplifying the connection between the breakout boards and the acquisition system. The microphone carrier PCB follows the same pinout as Adafruit's own PDM-Microphone breakout board (Adafruit part number: 4346), facilitating compatibility. The connector also supplies power (3.3V) to the microphones, allowing multiple units to be connected with a single cable. For the testing setup, cables measuring 40 cm in length were used (Adafruit part number: 5385).

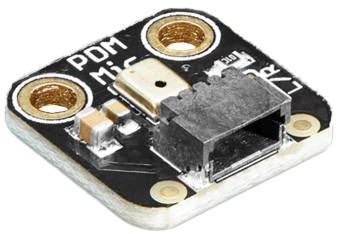


Figure 4.2: Adafruit PDM Microphone (4364)

Figure 4.3: STEMMA QT / Qwiic Cable (5385)

The table below outlines the pinout used for these microphone breakout boards:

Pin Number	Function
1	GND
2	3V3
3	PDM Data (provided by the host)
4	PDM Clock (microphone output signal)

Table 4.2: Microphone Breakout Board Pinout

4.5 Hardware Design

The hardware of the acquisition system is centralized around a 4-Layer PCB with a size of 111.0 mm x 136.0 mm. The connectors for the microphone breakout boards are located on the left and right side of the PCB (16 channels on each side). Each microphone channel is equipped with a pair of RGB LEDs. The LED near the connector visually represents the audio level for its respective channel. Another LED, situated on the white silk screen marking, indicates the active routing of the channel to the monitor headphones output. A *Monitor Selection* push button enables navigation through the microphone channels. Each button press cycles through the channel numbers, directing the selected channel to the headphones output. Additionally, a potentiometer is available for adjusting the volume of the headphones output. On the lower segment of the PCB, a 1.44" TFT touch display is embedded, serving as an interactive interface for controlling the acquisition system. Adjacent to this display is the *Record* button, which is used to start and stop the recording process. The upper part of the device houses a USB Type-C and an RJ45 connector, providing data connectivity to external devices. Moreover, each corner of the PCB is equipped with M3 surface-mounted standoff nuts, allowing for easy mounting of the acquisition system onto a microphone array frame.

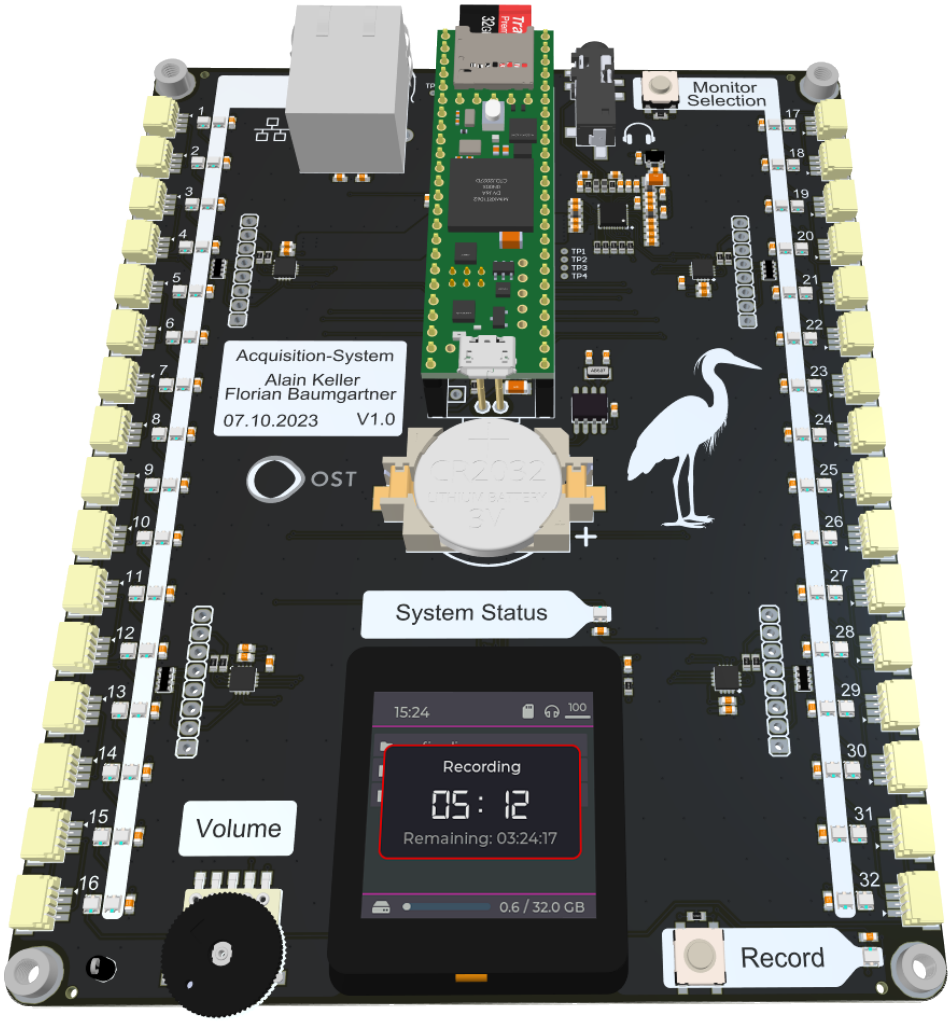


Figure 4.4: Front View of the Acquisition System

4.5.1 Block Diagram

In figure 4.5 the system block diagram is shown.

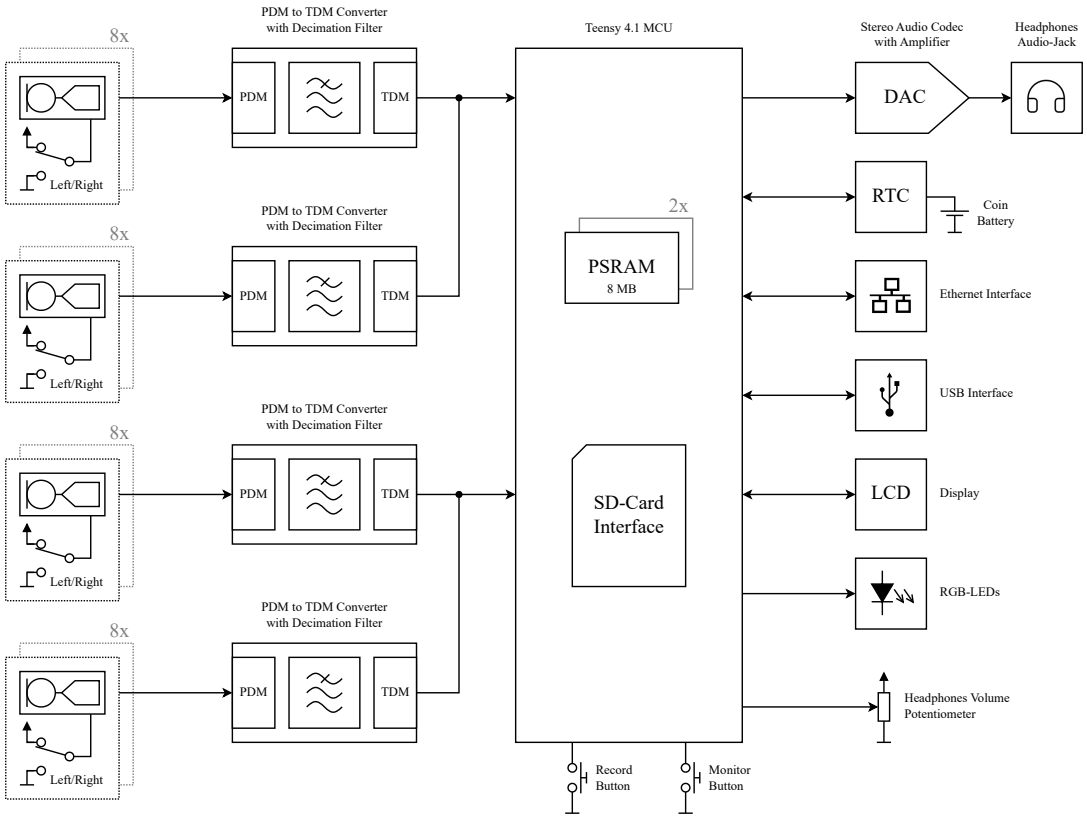


Figure 4.5: System Block Diagram of Acquisition-System

4.5.2 Microcontroller Unit (MCU)

As a main MCU the `Teensy 4.1` was chosen due to its powerful ARM Cortex-M7 processor, running at 600 MHz. The `Teensy 4.1` is a small form-factor development board, manufactured by *PJRC*. It provides a build-in programming interface, which allows to program the MCU directly via USB without the need for an external programmer. Compared to the smaller formfactor `Teensy 4.0`, the `Teensy 4.1` includes a built-in SD-Card holder and additional flash memory of 8 MB. Although the IMXRT1060 SoC provides 1 MB of RAM, for this application, two additional 8 MB PSRAM memory chips are required. They can convenient be mounted on the backside of the `Teensy 4.1` board as shown in figure 4.7.

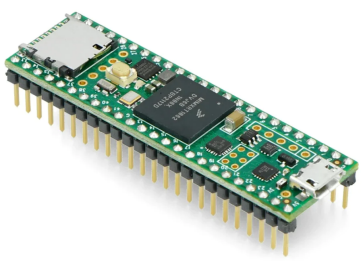


Figure 4.6: Teensy 4.1

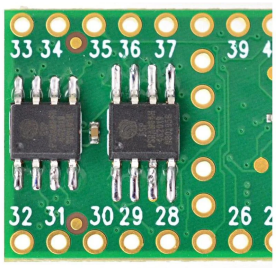


Figure 4.7: Mounting of External PSRAM

4.5.3 Audio Input

As mentioned in section 2.1.2, the use of PDM microphones requires a digital decimation filter to convert their bitstream into a usable audio format. The Teensy 4.1, despite having built-in hardware PDM decimation, is limited to two channels, requiring external decimation for systems with more audio channels.

To overcome this limitation, the *ADAU7118* from Analog Devices is employed. This Integrated Circuit (IC) is specialized in handling PDM signals, offering four PDM inputs and two clock outputs, which collectively can drive up to eight microphones in a multiplexed configuration. It outputs audio data in the TDM protocol. TDM-16, being capable of handling up to 16 channels, allows two *ADAU7118* ICs to share the same physical bus. Hence, individual configuration is necessary to ensure that each converter accesses the correct TDM slots (lower or upper 8 channels). In total, four *ADAU7118* ICs are used, providing 32 microphone channels.

The *ADAU7118* is also equipped with an I^2C interface, enabling the configuration of various parameters such as the decimation ratio, TDM bus settings, and signal drive impedances.

4.5.4 Headphones Output

The addition of a headphones output to the acquisition system introduces the capability for live monitoring of microphone inputs. This feature allows a user to connect standard headphones with a 3.5 mm jack directly to the device, enabling real-time audio feedback.

The system employs the *WM8904* audio CODEC as its Digital-to-Analog Converter (DAC). This choice is strategic, as the *WM8904* not only supports digital audio input streams like I^2S and TDM, but also includes a built-in headphones amplifier, making it particularly suitable for this application. Additionally, its I^2C interface enables configuration adjustments, including device setup and volume control.

However, it is important to note that, as of this writing, this headphones output feature has not been implemented in the firmware. Consequently, the hardware designed to support this functionality has not yet undergone testing.

4.6 Firmware Design

The firmware is written in C++ and is based on the Arduino framework that has been addopted to the Teensy microcontroller environment. As an IDE, the VS Code extension *PlatfromIO* was used, as it provides powerful development tools and a great integration of the Arduino framework.

The firmware is divided into modules running in individual threads, facilitated by *TeensyThreads* on the Teensy 4.1 microcontroller. This lightweight multitasking library allows for concurrent execution of multiple threads, optimizing system performance and resource utilization. *TeensyThreads* has a minimal memory footprint and is optimized for efficient CPU usage, making it ideal for embedded systems. In total, 6 threads are running in parallel, as shown in table 4.3.

Thread	Purpose
`Console Interface`	Handles USB virtual COM-Port
`Console Streaming`	Handles queuing of messages
`AudioUtils`	Audio Processing
`HMI`	LED Control & RTC
`Application`	Main Application Logic
`Main`	Main Thread (Background)

Table 4.3: Overview of all threads and their purpose

4.6.1 Graphical User Interface (GUI)

Light and Versatile Embedded Graphics Library (LVGL)

Light and Versatile Graphics Library (LVGL) is a free and open-source graphics library, primarily used for creating embedded GUIs. It's designed to be lightweight, consuming minimal memory and processing power, which is essential in embedded systems where resources are limited.

The decision to use LVGL in conjunction with the *NXP GuiGuider*, a graphical design tool, enables a rich set of features and enables rapid development. *GuiGuider* provides a user-friendly interface for designing GUIs, significantly simplifying the process of creating complex, visually appealing interfaces for embedded systems. It provides a code generator, which generates the necessary C-Code to translate the graphic design into LVGL API calls.

4.6.2 GUI Pages

The GUI is minimalisticly designed and straight forward to use. The navigation between the main pages is done by swiping left or right on the touchscreen. Next, the individual pages are described in detail.

Figure 4.8: Splash screen

Splash Screen

When the device is powered on, the splash screen is displayed until the boot process is finished. On average this takes about 5 seconds.

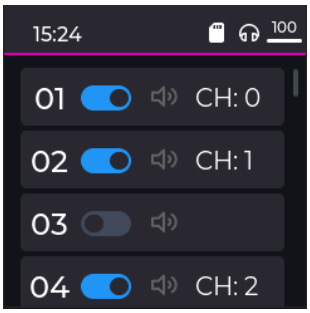


Figure 4.9: Channel Settings

Channel Settings

After the boot process is finished, the channel settings page is displayed. In the header bar located at the top of the page, the current time, USB interface status, SD-Card status and headphones volume are displayed. A list of all 32 microphone inputs is shown in the center of the page. Each input channel can be enabled or disabled by clicking on the corresponding switch. When an input is enabled, its associated channel number on the WAV file is displayed. A speaker symbol shows if the channel is currently routed to the headphones monitor output (green means active, grey means inactive).

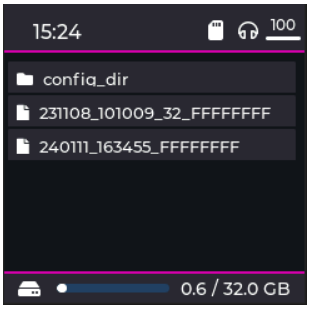


Figure 4.10: File Browser

File Browser

The file browser shows all files and folders located on the SD-Card. On the bottom of the page, a status bar indicates the current free and used space of the SD-Card. Due to the limited amount of memory, only the first 100 files and folders are displayed. This is however sufficient for most use cases.

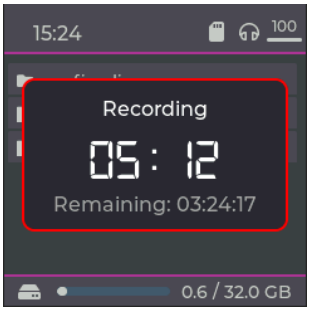


Figure 4.11: Recording

Recording

When the record button is pressed, the recording begins and a panel overlay is displayed. In the centre of the panel, the current recording time is displayed in minutes and seconds. Below, the remaining recording time is shown. When the recording is stopped, the panel overlay disappears. While the device is recording, all UI elements and the navigation are disabled.

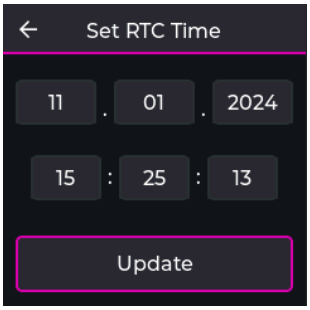


Figure 4.12: Set RTC Time

Set RTC Time

When the user clicks on the time in the header bar, the set time page is displayed. There the user can set the current time and date. After clicking on the *Update* button, the new time is set and the page is closed. To abort the process, the user can click on the arrow in the header bar.

5

Array Evaluation

5.1 Overview

5.2 Mechanical Design

Mechanical drawings of testarrays

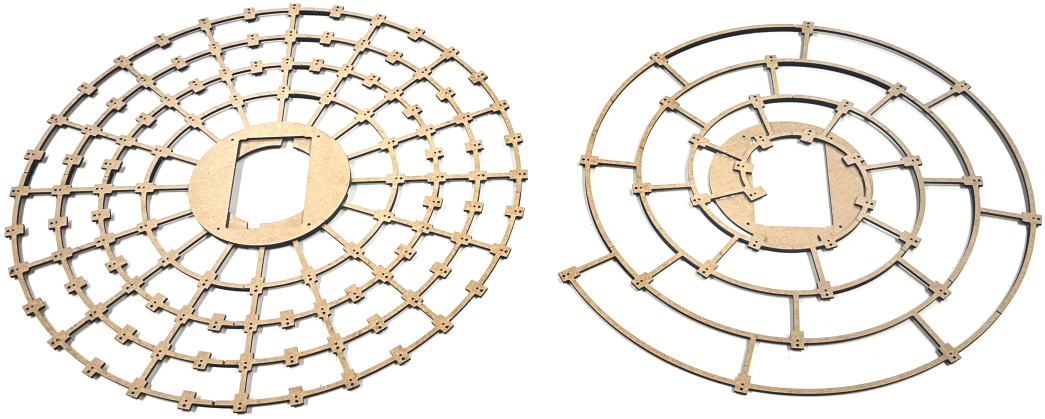


Figure 5.1: Wooden Prototype Arrays (Left: Circular Array, Right: Archimedean Spiral Array)

5.3 Metrics

TO evaluate the performance of an array several metrics are used. In the main beamwidth is proposed. is using the ratio between the main lobe power and the side lobe power. Another measure related to the main lobe with is the main lobe area. The main lobe is defined as all the points around the peak where their value is greater than half the peaks value.

Blabla

Cite Array design s40430-018-1275-5-1, array design comparisons

cite Circ array drone trackings13638-019-1632-9

besser englisch

5.4 Measurements & Findings

Blabla

5.5 Conclusion

Blabla

6

Final Design

6.1 Overview

This section covers the complete development process including the hardware, gateware, software and mechanical design of the project. It is important to note that the following documentation concentrates only on the final version of the device. Earlier hardware prototypes are not covered due to the lack of relevance.

6.1.1 Key Requirements

The main focus of the development is to design a professional looking, easy to use and eye-catching device for demonstration purposes. The project name *Audio-Beamformer* has been chosen as it is easy to remember and has potential to be seen as a trademark.

The following key requirements have been set:

- Single power adapter or power cable (e.g. no need of labor power supplies)
- Easy to install (e.g. montage on a camera tripod)
- Intuitive to operate via state-of-the-art graphical user interface
- Multiple audio streaming sources such as Bluetooth and USB input devices
- Great scalability and flexibility of the hardware and software design

6.1.2 Key Decisions

In the conceptional phase of the development, several key decisions had to be made. This contains mainly the signal flow and the division between the processing part on the Raspberry Pi and the FPGA. Further, the question had to be evaluated, if a built-in power supply or an external power adapter is preferred. And most importantly, which type of ultrasonic transducer should be used in the design. In addition, the overall dimension and scale of the final product had to be discussed. In general, most of these decisions were made according to results of simulations, physical measurements and after extensive discussions. In the following sections, each part of the project is explained in detail.

6.2 Mechanical Design

Blabla

6.3 Hardware Design

Blabla

Figure 6.1: Front view of the mainboard

6.3.1 Block Diagram

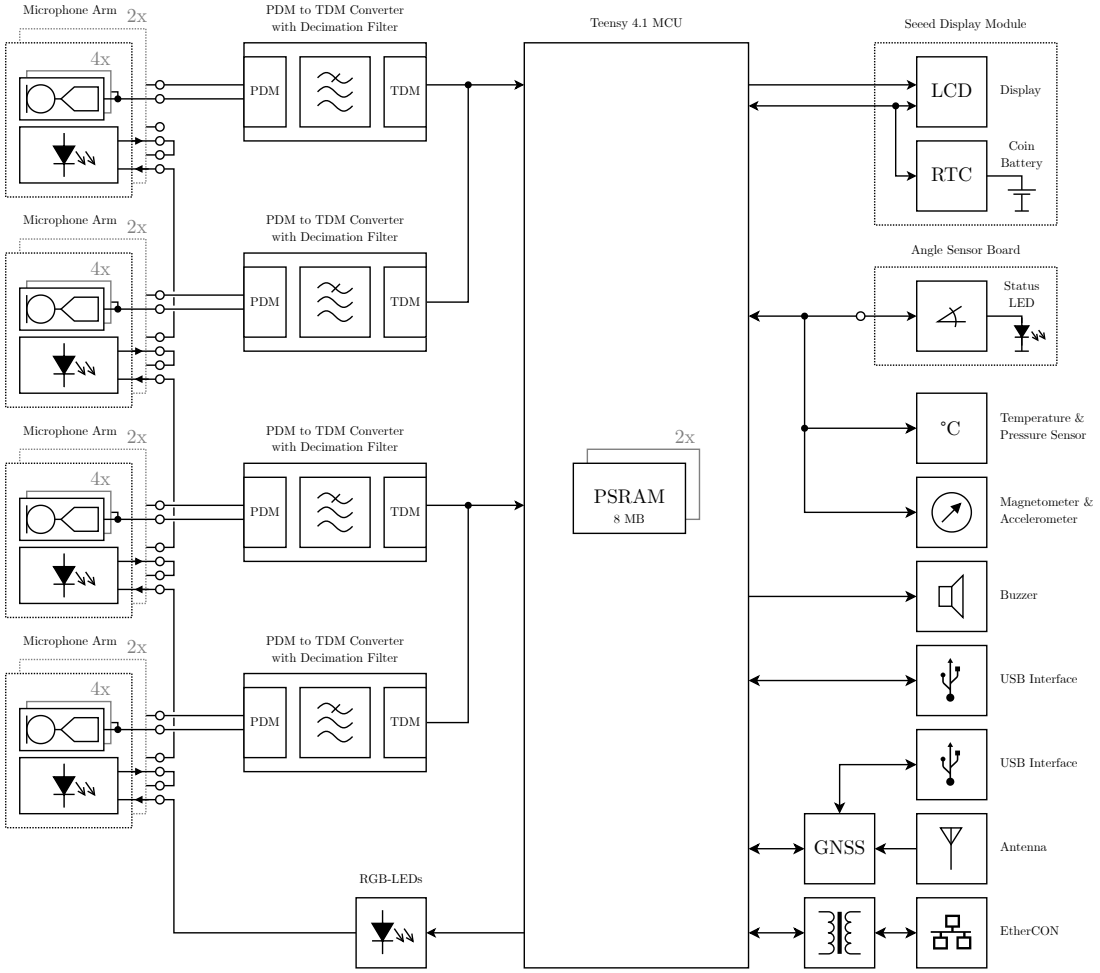


Figure 6.2: System block diagram

6.3.2 Power Supply

The device can be powered by multiple sources. Primary the device is powered by the ethernet interface, which provides a Power over Ethernet (PoE) connection. This is the preferred method, as it is the most convenient way to use the microphone array in the field. In addition, there are two USB-C ports (MCU and GNSS) that can be used to supply power when no PoE connection is available (e.g. in a laboratory environment for programming and debugging). Each supply method can be used in conjunction with each other. However, the source must be able to provide at least 12.5 W of power (5 V, 2.5 A). In figure 6.3 an overview of the power supply is shown. Note that the Teensy 4.1 has an internal 3.3 V regulator, which is powered by the systems internal 5 V supply. The 1.8 V rail is generated by a dedicated linear voltage regulator and is mainly used for the PDM to TDM converters (ADAU7118).

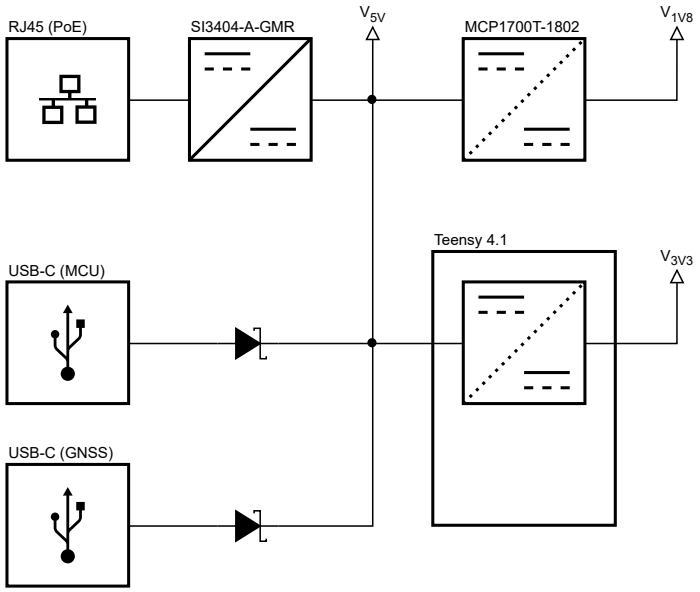


Figure 6.3: Power Supply Overview

Power over Ethernet (PoE)

6.3.3 Microcontroller Unit (MCU)

6.3.4 Audio Input

GNSS

6.3.5 Human Machine Interface (HMI)

LCD Display

RGB LEDs

6.3.6 Sensors

Magnetometer & Accelerometer Sensor

Ambient Pressure & Temperature Sensor

Angle Sensor

6.3.7 Printed Circuit Board (PCB)

Mainboard

Microphone Arms

Angle Sensor

6.3.8 Manufacturing

The PCBs were manufactured and assembled by JLCPCB. Only a few additional components were soldered by hand, such as the PDM to TDP converters (ADAU7118), the EtherCon connector and the GNSS module (NEO-M8P-2). It turned out that multiple microphone arm PCBs were faulty (damaged microphones). This was most likely caused by the soldering process, as the MEMS microphones are very sensitive to heat.

6.4 Firmware Design

Blabla

6.4.1 Overview

The firmware is written in C++ and is based on the Arduino framework that has been addopted to the Teensy microcontroller environment. As an IDE, the VS Code extension PlatfromIO was used, as it provides powerful development tools and a great integration of the Arduino framework.

The firmware is divided into modules running in individual threads, facilitated by TeensyThreads on the Teensy 4.1 microcontroller. This lightweight multitasking library allows for concurrent execution of multiple threads, optimizing system performance and resource utilization. TeensyThreads' minimal memory footprint and efficient CPU usage are critical in embedded systems where resources are constrained.

Thread	Purpose
`Console Interface`	Handles USB virtual COM-Port
`Console Streaming`	Handles queuing of messages
`Utils`	Updates operation time
`AudioUtils`	Audio Processing
`GNSS`	GNSS Data Handling
`HMI`	LED Control & RTC
`Buzzer`	Buzzer Control
`Application`	Main Application Logic
`Main`	Main Thread (Background)

Table 6.1: Overview of all threads and their purpose

6.4.2 Audio Streaming

The audio streaming module handles the buffering and transmission of the 32 audio channels. It is based on a TCP server that provides a socket connection on port 6666. The audio data is transmitted in a lossless 16-bit signed integer format, with a sample rate of 44.1 kHz.

The TCP connection asures a reliable data transmission, which is essential for the audio data. To provide a low latency audio stream, the audio data is buffered in a circular buffer of 12 MB size. This allows for a maximum delay of ca. 4.4 s.

For a continuous audio stream, a tranmission rate of

$$\frac{32\,\text{channels} \cdot 16\,\text{bit} \cdot 44100\,\text{Hz}}{10^6\,\text{bit/s}} = 22.1184\,\text{Mbit/s} \tag{6.1}$$

is required. The theoretical maximum transmission rate of 100Base-T Ethernet is 100 Mbit/s. However, the maximum transmission rate of the Teensy 4.1 is around 60 Mbit/s, which is still sufficient for the audio stream.

Due to the maximal TCP packet size of 1460 bytes, the audio data is split into concatenated packets. A frame consists of 128 interleafed samples (32 channels) and is transmitted in 8 packets. Each frame starts with a 20-byte header.

Byte Offset	Data Format	Description	Example Value
0-7	String	Magic Sequence	`HERON666`
8-11	Integer (Little Endian)	Packet Index	12345
12-19	Integer (Little Endian)	Timestamp (ns)	1616929134054668023

Table 6.2: Description of the 20-byte Packet Header

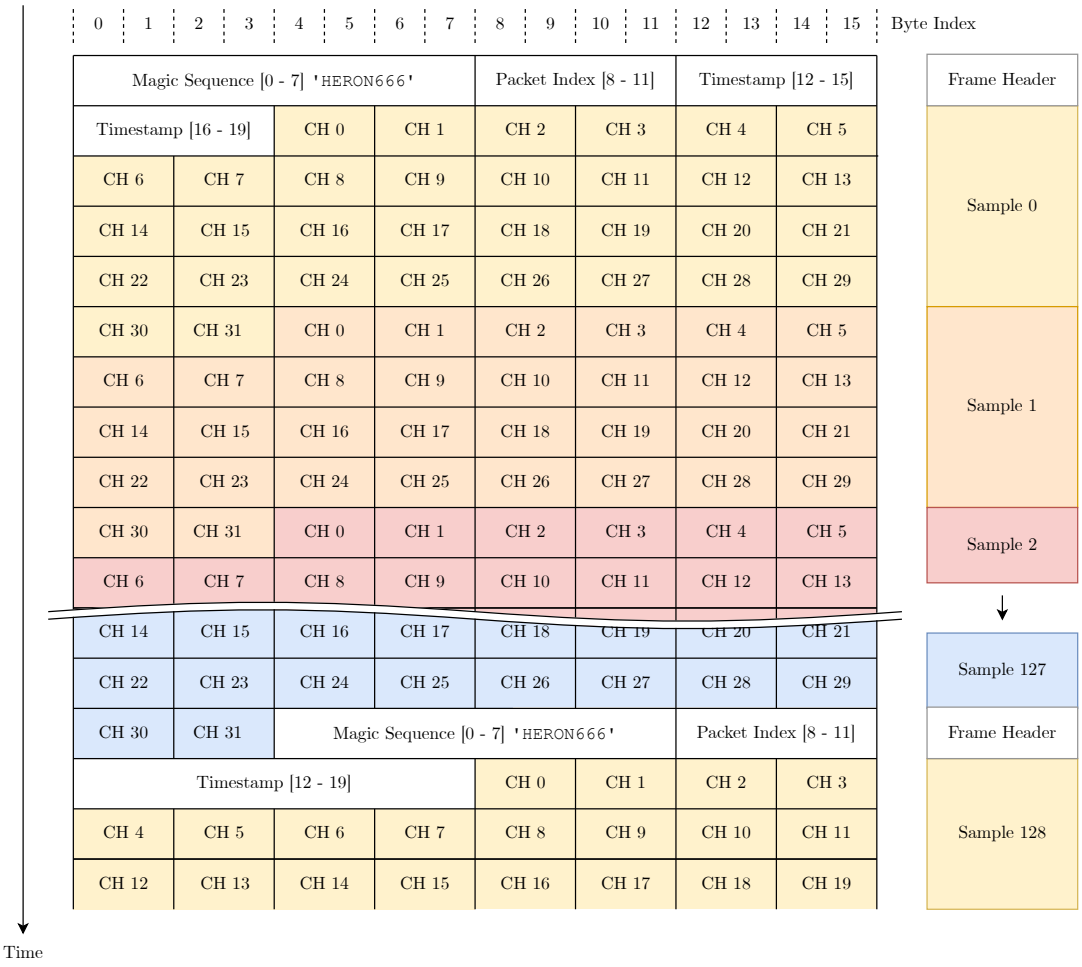


Figure 6.4: Example of a Frame with 128 Samples (32 Channels)

Remote Configuration

The

For remote configuration, the client (e.g. a PC) sends a JSON file containing the desired command and its parameters to the device.

6.4.3 Sensor Calibration

Data Field	Description	Data Type	Unit	Example Value
device_firmware_version	Firmware Version	String	-	"V0.1"
device_firmware_build	Firmware Build Date	String	-	"240110"
device_cpu_frequency	CPU Frequency	Integer	Hz	912000000
device_cpu_temperature	Current CPU Temperature	Float	°C	36.5
device_operating_time	Device Operating Time	Integer	s	15615
device_system_warning	Current System Warning Status	Boolean	-	false
ethernet_mac	MAC Address	String	-	"DE:AD:BE:EF:FE:ED"
ethernet_ip	Current IP Address	String	-	"192.168.1.10"
streaming_state	Current Streaming State	Boolean	-	true
streaming_speed	Current Streaming Speed	Float	Mbit/s	22.21
streaming_buffer	Streaming Buffer Fill Level	Float	%	15.1
sensor_heading	Device Heading	Float	°	270.0
sensor_pitch	Device Pitch	Float	°	5.2
sensor_roll	Device Roll	Float	°	2.4
sensor_temperature	Ambient Temperature	Float	°C	22.3
sensor_pressure	Ambient Pressure	Float	hPa	1013.2
sensor_altitude	Device Altitude	Float	m (MSL)	434.5
sensor_angle	Arm Angle	Float	°	45.7
sensor_magnet_detected	Magnet Detection Status	Boolean	-	true
sensor_magnet_too_weak	Magnet Too Weak Status	Boolean	-	false
sensor_magnet_too_strong	Magnet Too Strong Status	Boolean	-	false
gnss_latitude	GNSS Latitude	Float	°	40.7128
gnss_longitude	GNSS Longitude	Float	°	-74.0060
gnss_altitude	GNSS Altitude	Float	m (MSL)	344.8
gnss_magnetic_declination	GNSS Magnetic Declination	Float	°	-5.0
gnss_satelite_count	GNSS Satellite Count	Integer	-	8
gnss_fix	GNSS Fix Status	Boolean	-	true
gnss_fix_type	GNSS Fix Type	Integer	-	3
gnss_time_valid	GNSS Time Validity	Integer	UNIX (UTC+0)	1704921651

Table 6.3: Device Data JSON File Fields, Descriptions, Data Types, Units, and Example Values

Data Field	Description	Data Type	Unit	Example Value
clear_warning	Clears the warning status	Boolean	-	true / false
gnss_coefficient_xxx	GNSS RTK Coefficient XXX (Not implemented)	Float	-	-

Table 6.4: Overview of Received Commands in JSON File

6.4.4 Graphical User Interface (GUI)

The GUI provides a user-friendly interface for configuring the device and monitoring its status. It is based on the LVGL framework.

6.4.5 GUI Pages

In the following sections, the different GUI pages are explained in detail. Navigating between the pages is done by clicking on the corresponding icon in the home menu. To return to the home menu, the sub-page header bar with the arrow symbol can be pressed. Figure 6.5 shows an overview of all GUI pages and how navigation between them is done.

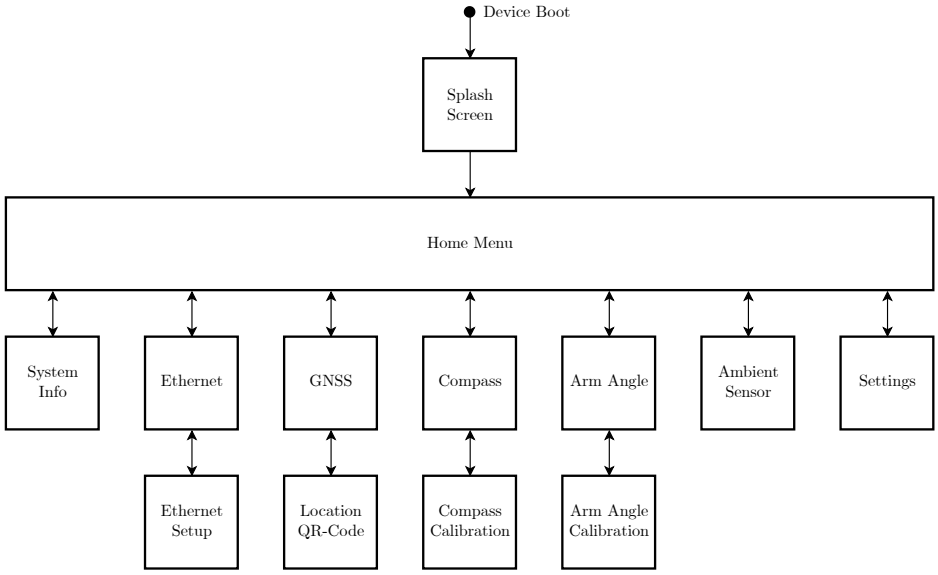


Figure 6.5: GUI Pages Overview

Figure 6.6: Splash Screen

Splash Screen

When the system is powered on, the splash screen is displayed for 5 seconds. On the bottom of the screen, the current firmware version and build date is displayed. After the system has booted, the home menu is displayed.

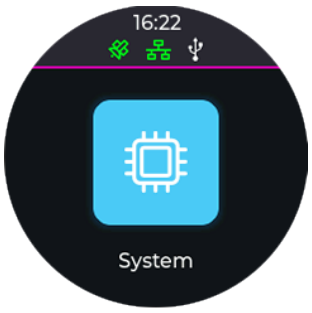


Figure 6.7: Home Menu

Home Menu

The home menu presents an intuitive way of navigating through the different submenus. On the top of the screen, the current time is displayed. A dedicated icon for the GNSS, ethernet and USB interface is shown. Per default, the icons are grayed out. When the GNSS location is valid (fix), the icon turns green. When the device is connected to the network but is not yet streaming, the icon is filled white. As soon as the audio stream is active, the icon turns green. The USB icon is filled white when a device is connected to the USB port. As soon as the virtual COM port is opened, the icon turns green.

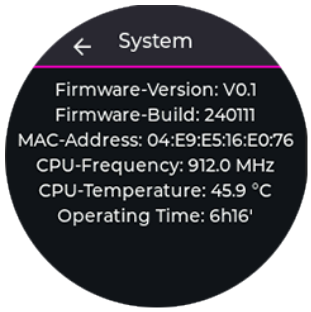


Figure 6.8: System Information

System Information

The system information page displays important device information such as the firmware version, firmware build date, etherent MAC address, CPU frequency and temperature, as well as the operating time.

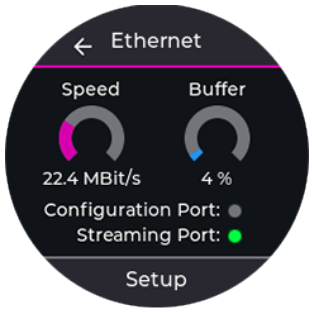


Figure 6.9: Ethernet

Ethernet

The ethernet page displays the current connection state of the streaming and configuration port. While a connection is established to either of the ports, the corresponding gray circle is filled green. Two gauge bars display the current streaming speed and the fill level of the streaming buffer. In normal operation, the indicated streaming speed should be around 22 Mbit/s and the buffer fill level near 0 %.

Figure 6.10: Ethernet Setup

Ethernet Setup

The ethernet setup page allows for configuring the IP address of the device. Below the IP address, the hard-coded streaming port (6666) and configuration port (6667) is displayed. When the IP address is changed, the user can either confirm the change or discard it by returning to the previous page. When a new IP address is confirmed, the device will instantly change its IP address and restart the servers.

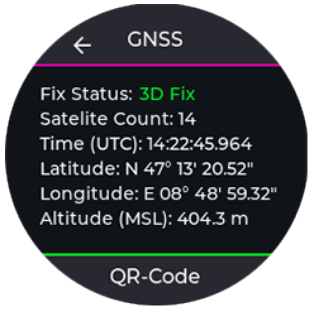


Figure 6.11: GNSS

GNSS

The GNSS page displays the current status of the GNSS module. When the Fix Status is valid (2D-Fix or 3D-Fix), the latitude, longitude and altitude is displayed. In addition, the QR-Code button gets enabled, which allows for displaying the current location in a QR-Code. The coordinates are displayed in degrees, minutes and seconds. The altitude is displayed in meters above sea level (MSL). When the time is fully resolved, it is displayed in the UTC+0 format. While the Fix Status is invalid, all fields are grayed out.

Figure 6.12: Location QR-Code

GNSS Location QR-Code

The GNSS location QR-Code page displays a QR-Code containing a google maps link to the current location. When the QR-Code is scanned with a smartphone, the web broweser will immediately redirect the user to the google maps app. The URL is formated as follows:

`google.com/maps/place/<latitude>,<longitude>`

For example, the URL directs to:

`google.com/maps/place/47.222400,8.816460`

Figure 6.13: Compass

Compass

The compass page displays the current heading and leveling of the device. The heading is indicated by a purple arrow, pointing towards geographic north. When the needle points exactly towards the top of the screen it turns green, meaning the device is facing north. The circular graphic in the centre of the screen shows the current leveling of the device. When the device is perfectly leveled, the circle turns green. When the device is tilted, the circle turns red and the tilt angle is displayed in degrees (pitch and roll). To calibrate the built-in magnetometer, the user can press the calibrate button.

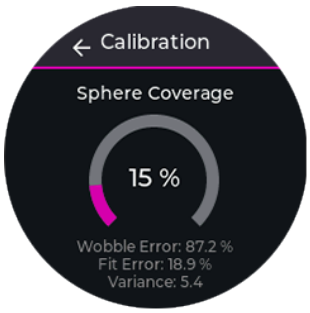


Figure 6.14: Compass Calib.

Compass Calibration

The compass calibration page displays the current status of the magnetometer calibration. By entering this page, the calibration process is starts immediately. To calibrate the magnetometer, the device has to be rotated around all three axes. This process takes around 30 seconds. As soon as the spere coverage indicator reaches 100 %, the calibration is finished, a success message is displayed and the buzzer plays a short melody. The calibration process can be aborted at any time by returning to the previous page (the calibration progress is not saved).

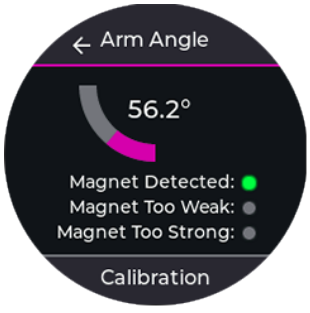


Figure 6.15: Arm Angle

Arm Angle

The arm angle page displays the current angle of the microphone array arms in degree. When the arms are fully unfolded (horizontal) the angle is 0.0 °. When the arms are fully folded towards the centre pole the angle is 90.0 °. Below the angle indicator, the current status of the magnet detection is displayed. When the magnet is detected, the status indicator turns green. If the magnet is too weak or too strong, the corresponding status indicator turns yellow. Otherwise, the status indicators are grayed out.

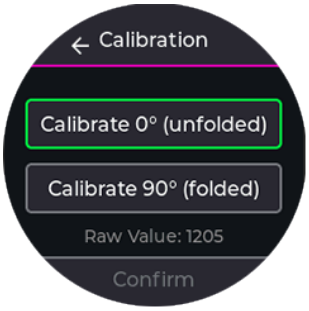


Figure 6.16: Arm Angle Calib.

Arm Angle

The arm angle calibration page lets the user calibrate the angle sensor. To calibrate the angle sensor, the arms have to be fully unfolded (horizontal). Then the upper calibrate button *Calibrate 0° (unfolded)* has to be pressed. Next, the arms have to be fully folded towards the centre pole. Then the lower calibrate button *Calibrate 90° (folded)* has to be pressed. To confirm the calibration, the user has to press the *Confirm* button.

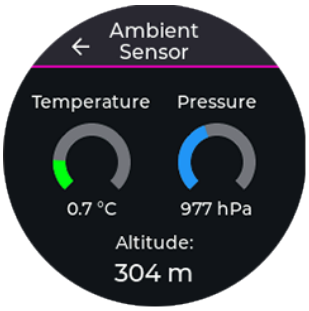
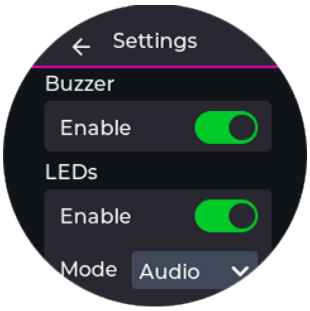


Figure 6.17: Ambient Sensor

Ambient Sensor

The ambient sensor page displays the current ambient temperature in degree Celsius and the ambient pressure in hectopascal. A gauge bar visualizes both values. Based on the current ambient pressure, the altitude above sea level is calculated and displayed in meters. Note that the altitude is only an approximation and can be inaccurate. It is strongly influenced by the current weather conditions.



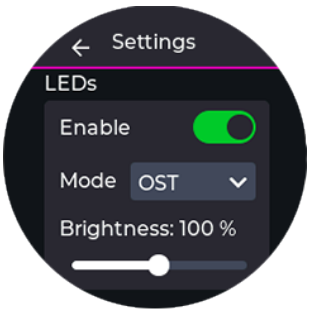


Figure 6.18: Settings

Settings

The settings page allows the user to configure the device. First, the buzzer can be enabled or disabled. Second, the LEDs can be permanently enabled or disabled. The mode selection provides a selection of different animations for the LEDs. Currently, there are two animation types implemented:

- **Audio:** The LEDs are controlled by the audio level of the microphone array.
- **OST:** The LEDs show a fluent animation in the OST color scheme.

Below the mode settings, the LED brightness can be adjusted. All settings are saved in the internal flash memory of the device and are restored after a reboot.

6.5 Software Design

Blabla

7

Measurements

7.1 Overview

This section covers the complete development process including the hardware, gateware, software and mechanical design of the project. It is important to note that the following documentation concentrates only on the final version of the device. Earlier hardware prototypes are not covered due to the lack of relevance.

8

Conclusion

A

Appendix

A.1 Declaration of Authorship

We hereby certify that the thesis we are submitting is entirely our own original work except where otherwise indicated. We are aware of the University's regulations concerning plagiarism, including those regulations concerning disciplinary actions that may result from plagiarism. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Location, Date
Rapperswil, 24. January 2024

Florian Baumgartner

Alain Keller

A.2 Data Archive

All created files and documents of this project are publicly available on GitHub. An institution called **PA-OST-2023** (`https://github.com/PA-OST-2023`) has been founded which contains repositories for each individual part of the project. A quick description of the repositories including the associated web link is listed below:

heron-administration
Description: This repository contains all confidential information of the project.
URL: `https://github.com/PA-OST-2023/heron-administration`
Type: Private

heron-literature
Description: This repository contains all literature used in this project.
URL: `https://github.com/PA-OST-2023/heron-literature`
Type: Private

heron-documentation
Description: This repository contains this document.
URL: `https://github.com/PA-OST-2023/heron-documentation`
Type: Public

heron-hardware
Description: This repository contains hardware related documents (Schematics, PCB).
URL: `https://github.com/PA-OST-2023/heron-hardware`
Type: Public

heron-firmware
Description: This repository contains firmware source code written in C++.
URL: `https://github.com/PA-OST-2023/heron-firmware`
Type: Public

heron-simulator
Description: This repository contains the simulator source code written in Python.
URL: `https://github.com/PA-OST-2023/heron-simulator`
Type: Public

heron-application
Description: This repository contains the application source code written in Python.
URL: `https://github.com/PA-OST-2023/heron-application`
Type: Public

heron-mechanical
Description: This repository contains mechanical related documents (CAD-Files).
URL: `https://github.com/PA-OST-2023/heron-mechanical`
Type: Public

heron-bastelstube
Description: This repository contains temporary and experimental files.
URL: `https://github.com/PA-OST-2023/heron-bastelstube`
Type: Private

Bibliography

[1] Thomas Kneubühler. Nachrichtentechnik 1 + 2, 2020.