LODZ UNIVERSITY OF TECHNOLOGY
Faculty of Electrical, Electronic,
Computer and Control Engineering

Master of Engineering Thesis

Electronic system for localization of sound sources in 3D space

Byczkiewicz Jacek

Student's number: 214427

Supervisor:
Dr inż. Michał Bujacz

Łódź, 2018

Acknowledgements

First and foremost, I would like to thank my supervisor Ph.D Michał Bujacz for his assistance and encouragement throughout the course of the thesis. I would like to express my gratitude to Ph.D Marcin Kociołek and Ph.D Paweł Poryzała, for their help regarding the electronic circuit design and for making the University's laboratory always available. Finally, I would like to sincerely thank my family for their patience and support and for making the thesis possible.

LODZ UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL, ELECTRONIC, COMPUTER AND CONTROL ENGINEERING

Jacek Byczkiewicz

Msc THESIS

Electronic system for localization of sound sources in 3D space

Lodz, 2018

Supervisor: Michał Bujacz, Ph.D, Eng.

ABSTRACT

The goal of the presented thesis was to design and construct a compact real world electronic system, able to localize sound sources in three-dimensional space. The system was then adapted for the practical application of tracking a ball in a sound table tennis variant for visually impaired people. The software interface can also provide the means for tracking the progress of the game using 3D visualization and a basic score display, as well as means for configuration of the sound localization process.

The theoretical review summarizes concepts of localization in general with the focus on acoustic sources. Different approaches are analyzed, among which multilateration is chosen and described in detail. The chosen concept is then implemented in practice in Python programming language. Furthermore, the work features the design and construction of a custom microphone array, the elements of which include an ADC converter PCB board and microphones with preamplifiers along with software drivers necessary for its operation.

Finally, the whole system is tested in terms of its localization capabilities. The measurement accuracy is derived for real world data collected by the microphone array. Furthermore virtual simulation is conducted in order to improve the spatial placement of the microphones. As a result, such system is capable of accurately localizing sound sources in space, in particular a ball bouncing on the table in a tennis game.

POLITECHNIKA ŁÓDZKA
WYDZIAŁ ELEKTROTECHNIKI, ELEKTRONIKI, INFORMATYKI I AUTOMATYKI

Jacek Byczkiewicz

PRACA DYPLOMOWA magisterska

Elektroniczny system do lokalizowania źródeł dźwięku w przestrzeni 3D

Łódź, 2018 r

Opiekun: dr inż. Michał Bujacz

STRESZCZENIE

Celem niniejszej pracy był projekt i konstrukcja kompaktowego systemu do lokalizacji źródeł dźwięku w przestrzeni 3D. System ten został dalej przystosowany do konkretnej aplikacji śledzenia ruchu piłeczki w grze tenisa stołowego dla osób niewidomych. Dodatkowo, przygotowane narzędzie może służyć do śledzenia postępu gry. W pracy przedstawiony został prototyp interfejsu użytkownika, z możliwością konfiguracji parametrów używanych do lokalizacji piłeczki oraz wizualizacją 3D środowiska gry.

W przeglądzie teoretycznym omówione są zagadnienia dotyczące lokalizacji źródeł akustycznych. Przedstawione są różne sposoby ich pozycjonowania, spośród których multilateracja została wybrana dla zastosowań projektu oraz zaimplementowana w języku Python. Dodatkowo praca zawiera projekt własnych podzespołów elektronicznych wchodzących w skład macierzy mikrofonowej. Są to płytka przetwornika ADC oraz projekt mikrofonu z przedwzmacniaczem.

Ostatecznie cały system poddany jest testom pod kątem dokładności lokalizacji dźwięku odbijanej piłeczki. Oprócz testów w środowisku rzeczywistym przeprowadzona jest także symulacja badająca rozmieszczenia mikrofonów w obrębie stołu do gry. Rezultatem jest system mogący lokalizować ją z dokładnością do ok. 3 cm w płaszczyźnie, co czyni go wystarczającym na potrzeby śledzenia przebiegu gry w tenisa stołowego.

Table of Contents

1 Scope of the project

The goal of the work is to create an electronic system that can detect a sound source with a given spectral characteristic and localize its position expressed in terms of a local coordinate system. This type of technology is not a novelty on its own and there are many solutions aiming at this general topic already present on the market. These however are often in form of microphone-arrays with a large number of microphones, resulting in high cost and complexity of the design. Additionally, the computational cost of analysing the signals coming from multiple microphones is significant and often cannot be performed on a small embedded platform. In this project the aim is to design a system that is not only viable in terms of localization purposes, but also one that consists of a small number of microphones and requires as little processing power as possible.

Such a solution could be used in everyday life without excessive costs or specific platform requirements. A real life application of choice is a virtual arbiter in a table tennis game for the visually impaired. Those users cannot play the game by themselves and they require additional assistance to track the game progression. With sound source localization this problem could potentially be solved, as location of the ball within the game could be tracked by its emitting sound. As the ball's location is known with respect to the tennis table, the game can be registered and assessed in terms of viable plays and score tracking. Implementing such a virtual arbiter in a prototype form, which demonstrates the basic ball tracking, is an ultimate goal of this work.

1.1 Sound table tennis

Sound table tennis is a game designed especially for people with impaired sight. The designer of a game is a Polish physical education teacher and physiotherapist Leszek Szmaj. The game is based on a standard table tennis as it utilizes the equipment used in that game. Namely sound table tennis version of a game still requires the same table and ball. In this variant however, the net between the table halves is not used and the game is played with hands rather than with paddles. The main objective of a player is to throw the ball in such a way that it bounces at least two times on his half of the table and is not caught by the opposing player. Game can be

played individually or in doubles. In individual variant it is usually played from 5 up to 15 points. The rules are straightforward, the serving player serves a ball that bounces at least two times on his half and then at least once on the opponent's half. If the opposing player catches the ball with his hands or the ball rolls off the side of the table, the score is not granted, otherwise the serving player earns one point. Game finishes when a set number of points is gained by one of the players. An additional rule is set for the defending player. It states that he cannot help himself with any part of his body other than his hands, and he cannot touch the surface of the table with his elbows [1]. The exact scoring rules are as follows. The point is not granted when:

- The ball does not bounce at least twice on the attacking player's table side
- The ball does not bounce on the defending player's table side
- The surface of the table is touched by a free hand during throwing
- A player disturbs the silence during a pass
- A player does not perform a correct throw after the referee's command within 10 seconds
- The player passes the ball that rolls on the table instead of bouncing

Based on the above rule-set it is possible to validate at least part of the rules electronically, assisting the referee or even partially eliminating the need for his presence. One can track the position of the ball bounces during a game and validate if a given pass was correct according to the rules. The event of ball catch can be detected by simply setting the timeout between bounces of a ball. If the ball didn't bounce for certain amount of time, one can assume that it was caught. Analogically, if the ball was detected outside of the table surface, the defence can be counted as failed one. This type of system can assign points and therefore perform the complete game. There is still however the need for a human referee in order to validate the rules regarding the player's technique or fair play. Despite that, the electronic system can still offer assistance for a human referee as the game history can be recorded and one can see the exact position of ball bounces throughout the game. Furthermore it could potentially allow for friendly training game sessions to be held even without the referee at all.

2 Localization problem

Problem of positioning and self-awareness in the surrounding environment is not a new one in terms of the modern world of physics and computer science. In fact it is the problem that applies not only to technical fields of science but to all living organism as well. All living organisms and many artificially created robots are equipped with sensors that allow perception of the environment and in consequence the interaction of such being with its surrounding. One can distinguish two main approaches to the positioning system: relative (inertial navigation system) and absolute. In relative approach the position is calculated based on previous location and the displacement of the robot. An example can be dead reckoning which is a navigation process where position is calculated based on estimated speed and elapsed time. This method however is prone to cumulative errors that are introduced, whenever new position is calculated. In contrary absolute positioning utilizes the base stations with known locations to narrow the location of the searched object to a certain area. This kind of positioning is more efficient as it yields smaller errors, thus has better accuracy. The most popular GPS or its European equivalent Galileo operate on the same principle. Global positioning systems are utilized across many applications, which include navigation, control of autonomous road or aerial vehicles. In more specialized applications like sonar or radar it is vital to know the location of an object based on what it is emitting. These emissions can be in form of either electromagnetic wave, like during localization of malfunctioning assets in electrical facilities, or in form of sound wave when localizing the epicentre of the earthquake.

Both electromagnetic and sound waves can be used during the localization process. Different techniques are necessary to localize source of either electromagnetic or sound radiation, however some of them are closely related due to similar physical nature of both waves. For instance echolocation techniques are utilized in radars. Despite that, there are some crucial differences that have to be noted. Sound waves are more restricted as they always require a medium to propagate, while electromagnetic waves can propagate through vacuum. Due to that fact, sound is heavily influenced by its propagation medium, which dictates the speed at which it can travel (343 m/s in air) and also attenuates any transferred signal. [2]

2.1 Acoustic Localization

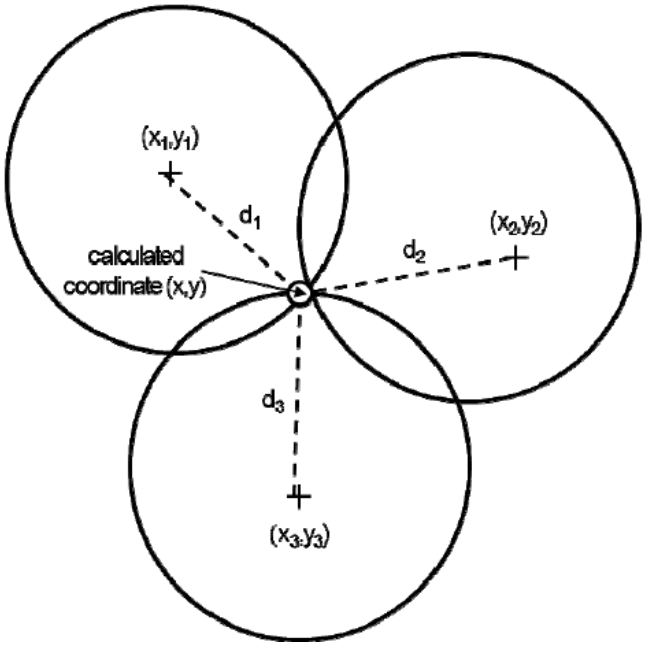
Sound localization is a fascinating topic on its own in the fields of mathematics or hardware design. The ability to find the direction of a sound and in consequence its absolute position can find many applications across various domains. Sound localization can be crucial in seismology as a tool for searching fossil fuel deposits, or in medicine like in case of the ultrasonography. Moreover it can be vital during rescue missions or in robotics to help localize the robot in indoor environment or as a part of sensory system [2]. Finally sound source localization is utilized in noise cancellation systems to filter out noise coming from the surrounding and focus only on sound coming from a selected direction. With the help of acoustic localization one can find the sound source of a certain characteristics, focus a sound beam to that direction, and amplify its amplitude. This can be useful in recording systems utilized in conference rooms, where the active speaker can automatically be detected and the camera is automatically focusing on him [3]. Examples of such technology are products made by Ploycom or Picture Teland manufacturers. These companies utilized microphone arrays in their voice recording products destined for conference rooms. Additionally, both offer an automated camera, which locates and frames the currently active presenter within the room. In case of the Picture Teland products, camera localizes the talking person based on four element microphone array [4].

Accurately predicting the position of a sound source is a complex problem requiring significant computational power. Furthermore calculations have to consider various phenomena affecting sound waves, like for instance refraction, diffraction, reverberation, diffusion or interference. This impacts the reliability of the computed results and increases complexity of the problem. Additionally, one has to take into account possible wide spectrum of acoustic sounds that typically include human hearing range – from 20Hz to 20 kHz, but also can include frequencies exceeding this interval, like infra or ultrasounds. This fact adds another layer of difficulty as the various frequency components of a wide spectrum are differently attenuated by the propagation medium. Sound sources having wide frequency characteristics can be perceived differently depending on the environment in which the sound propagates. This kind of challenges influenced creation of many different localization techniques including among others time difference of the arrival based multilateration, beam forming or constant speed localization [2].

2.1.1 Trilateration

Trilateration is a basic method used in localization systems, most commonly being the basic principle of GPS. It relies on known distance from each reference point to the searched position. In 2D scenario, the distance from the reference point with the known coordinates x, y is the radius of circle on which the searched target position is located. Having two distances, thus two circles narrows the possible solutions to only two points where both circles intersect with each other. This allows indicating the correct target position under the assumption of having proper domain. With the help of a third reference point there is only one result, which lies on the intersection of all three circles. One has to note however, that in order to obtain one truly unique solution it is also necessary to correctly place the receivers, employing all dimensions in localization space. In a 2D problem this means that receptors cannot be placed on a single straight horizontal line, and in a 3D problem they cannot be coplanar and have to be distributed in the vertical direction as well.

In real life world, finding an exact solution is rarely the case as the sensor readings are not accurate and contain some degree of error. This means that circles are usually not going to intersect, converting the problem into an optimization one, where the answer is in form of a circle, with radius proportional to the error. The idea of trilateration is illustrated in Fig, 2.1. In terms of three dimensions the technique works analogically, but the distance is a radius of a sphere, rather than a circle and one more reference point is necessary. Having three distances from three known reference point yields two solutions and four distances restricts the number of results to one.



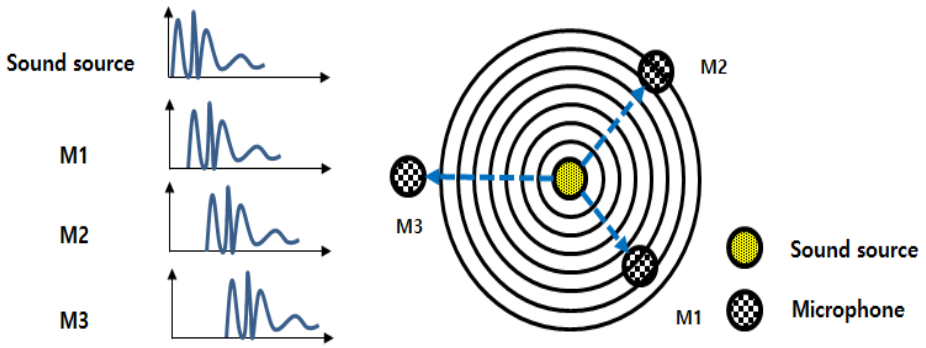
Fig, 2.1 Trilateration in 2D space [5]

This technique can be used as an audio localization technique. There is however one severe restriction of knowing the exact time at which sound is emitted from the source .The distance in this technique is calculated based on time of flight of the emitted signal to the receiver. In order to calculate it, the exact moment in time when the source started emitting is required. In some applications it is difficult or impossible to get this information, therefore methods employing time difference of the arrival are preferred. This restricts the use of this technique to localization of active sound sources, which are able to communicate with the reference points and emit the sound at an exact time.

2.1.2 Multilateration

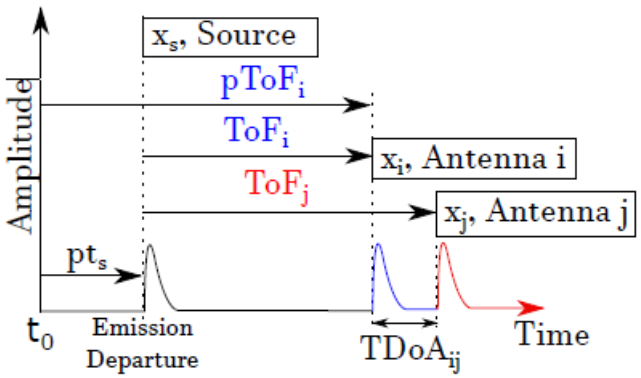
Multilateration is a technique that is more suitable for localizing passive sources as it does not require knowing the precise emission time. This method is focused on the measurement of the time delay of the arrival, in short TDoA, of the source signal to the receiver positioned at the reference point. The only data that is utilized in order to solve the coordinates of the sound source are the positions of the receivers and computed TDoA between receptors pairs. In 2D scenario at least two pairs of receivers are needed, which means three sensors. In 3D, the number of needed pairs equals to three, which requires usage of four receptors [2]. As in the

case with the previous method these receivers need to be correctly placed with at least one deployed in a different plane than the rest.



Fig, 2.2 Illustration of concept of TDOA based localization method

Performance of any multilateration technique lies in proper measurement of time variables. There are several, which can be utilized to find the source of the received signal. These are the time of flight (ToF) of the signal between source and the receiver, the time difference of arrival (TDoA) of the signal between receiving sensors and the pseudo-time of flight (pToF) of the emitted signal to the sensors. The latter is an alternative time measurement, which stands for the time of flight of the emitted signal to sensor with additional offset (pt_s). This time offset constitutes to the time elapsed between the start of a given clock t_0 and the departure of the emission from the source. All of these variables are illustrated in Fig, 2.3.



Fig, 2.3 Time variables representation foe emitting source and i-th and j-th receivers [6]

Every multilateration algorithm is formulated differently, but all of them are based on the property that the distance propagated by the emissions from the source is equal to the time

spent in flight multiplied by the speed of propagation. Since this work is devoted to acoustic localization in air, the propagation speed is assumed a constant 343 m/s. This speed can vary with the changes of the surrounding air temperature, but the differences from the point of this project are negligible. [7]

The relation of ToF t_i of the sound emitted from the source to the distance of i-th receiver from source equals:

$$D_i = t_i \cdot c$$

Equation 2-1 Distance from the source to the i-th receiver in terms of time of flight

where c is the propagation speed of the sound, D_i is the distance from the position of the i-th microphone in the array P_i (x_i , y_i , z_i) to the source position P_s (x_s, y_s, z_s). This distance can be also expressed with Euclidean metric as follows:

$$D_i = \sqrt{(x_s - x_i)^2 + (y_s - y_i)^2 + (z_s - z_i)^2}$$

Equation 2-2 Distance from source to the i-th receiver

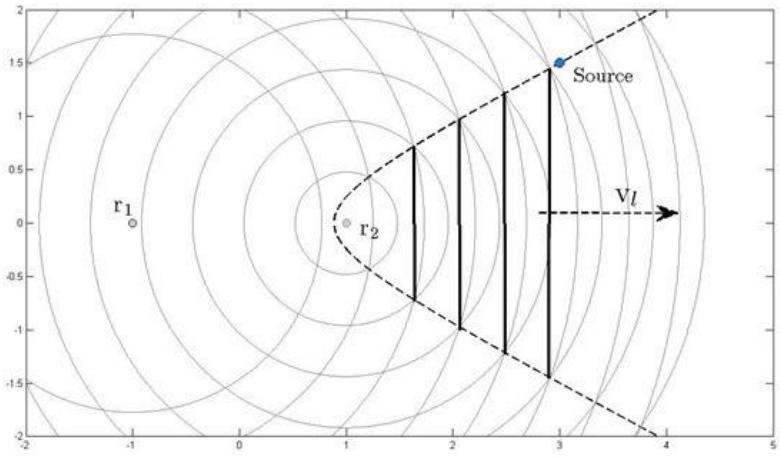
The TDoA between i-th and j-th microphones is related to the spatial variables through:

$$t_{ij} \cdot c = D_i - D_j$$

Equation 2-3 TDoA relation to the distance

2.1.3 Constant speed localization

The TDoA based multilateration method involves solving non-linear equations, which can be a complex task consuming a lot of resources. Due to that facts there are attempts to simplify the problem to linear form. This kind of approach is taken in a constant speed localization method.



Fig, 2.4 Constant speed localization, propagation of the straight front [2]

The calculation problem can be simplified by adding an additional receiver to the minimum required in previously presented method. This means four for a 2D problem and five for a 3D one. Furthermore, the receiver has an additional role of also acting as source on its own, thus also emitting the sound. When all of the receivers emit the sound in the inverse order they received the signal from the source, all emitted wave fronts will intersect at the source position at the same time. An illustration of this can be seen in Fig, 2.4. Two receivers are separated by a known distance from each other. They receive the signal from the source with the certain time delay. After receiving the signal from the sound source, both receivers start emitting with a time delay. Both circles will intersect, creating a hyperbole on which the sound source can be found. However, if the successive intersection points are joined together with a straight line they will create a straight front. Each receiver will create one straight front which propagates at a constant speed and all fronts will reach the source at the same time, meaning they will intersect at the source position. Therefore a linear system of equations can be created, where the unknowns are represented by the source coordinates and the time of the arrival. This simplifies the localization problem in terms of calculations, but at the expense of increased complexity of the overall design, as it requires more receptors, which need to perform additional tasks. [2]

2.1.4 HRTF Localization

Another technique in acoustic localization is to imitate human beings. Methods presented previously utilize time delay between sensors or distance. These techniques are on contrary significantly different than typical mechanics of the human ear. The ears make use of not only time delay information between them but also direction dependent spectral cues in the registered sound. The overall shape of the body and the ear itself are taking part in the sound localization process as those parts provide spectral cues. Furthermore phenomena of torso reflection, head diffraction or reflections on the ear shape also take important role in the whole localization process. [8]

Binaural recordings usually utilize artificial dummy heads. These contain replicas of human pinnae that are supposed to mimic the ear to provide additional cues about the direction or distance of the incoming sound. Since the pair of ears is present, localization cues can be divided into ones that use a single ear and others that require both ears. The binaural cues help to determine the direction of the incoming sound. This direction is deduced by taking into account time delay between both ears occurring when receiving the acoustic signal (interaural time delay) and also the differences of sound magnitude across the ears (interaural level difference). With the help of these cues the azimuth can be found, by comparing the amplitude and time differences of one ear to another. The monaural cues on the other hand are helpful in elevation estimations. The pinna is shaped asymmetrically in such a way that incoming sound is distorted differently based on the direction from which it arrives. This is especially true for elevation. Finally, one of the most significant monaural cues is connected with the environment around the artificial head. It is provided through the reverberation when the sound bounces around. Analysing these reverberations in similar fashion as in echolocation can be used to find the distance from a certain obstacle. This is especially noticeable in small spaces. [9]

Several algorithms have been developed to mimic the behaviour of human ears. In the recent years binaural hearing techniques based on Head Related Transfer Function (HRTFs) have become topic of the great interest in terms of mobile robot designs. Such robots would strongly benefit from these techniques as human based sound localization methods are resilient to noise

and reverberation and require only two microphones in order to localize the source in three-dimensional space. [2]

2.1.5 Beam-forming

Beam-forming is the method most commonly utilized in the sensor arrays in order to receive a signal coming from a specific direction. The simple form of beam forming assumes that waves received by the receptors are plane, which in reality is only possible when the source is placed in a great distance from the receiver. In most basic beam-forming implementation, a time delay between two receivers corresponds to a certain direction. If the signal received at one microphone is delayed by its time lag in reference to second and both signals are summed up, they will effectively double in amplitude. The sound coming from other directions on the other hand will be minimized. Using this method one can iterate with a constant step through all possible directions and find one that yields the best signal strength, thus find the direction on which lies searched sound source. It is also possible to determine elevation through beam-forming; this however requires at least two-dimensional sensor array. [10]

2.2 Multilateration - implementation

In terms of the aim of this project, which is to calculate the coordinates of the sound source, the multilateration was seen as the most suitable. Firstly, the acoustic sources that will be analysed are passive like in this specific case the bouncing ball, meaning that calculating precise time of flight will be difficult. This already discards the trilateration technique. Furthermore, the microphone array that will probe the signals cannot be too complex and large. Each additional channel dramatically increases the overhead of data transfer and thus increases the complexity of the design and cost of the equipment. Multilateration allows for the lowest number of channels in the acquisition system and requires nothing more than microphones arranged in form of an array. On contrary beam-forming is often used with large arrays and have significant computational complexity. Binaural recording despite the need for only two microphones also requires constructing an artificial head which is impractical in most of the applications. With the multilateration technique one can localize the source without any additional information, but that which was received by the sensors, provided that sensor positions are known.

In terms of actual implementation there exist many algorithms that aim to find the spatial coordinates of the source, based on TDoA. Despite having simple basic principles, the non-linear equations present in multilateration technique lead to challenges in terms of efficiency. One can distinguish two types of them, mainly iterative and non-iterative approaches. Both of them have their own set of advantages and drawbacks. The first group tends to be based on numerical methods for finding the best approximation of equation roots, like Newton – Raphson. Those kinds of algorithms have a certain restriction. They do not always converge and require initial guesses, which can significantly impact the overall calculation time [11]. Consequently, it can also cause variations in the final solution from one calculation to another. On the contrary the non-iterative algorithms tend to find the source position much faster, but at a cost of returning two solutions, negative and positive, instead of a definitive one. In GPS localization, the root selection can be performed in various ways, like for instance solving the clock error of a single receiver [12]. However, in most of the implementations, both roots are possible, with one root being closer than the other from the real source position. It is not always obvious which root gives the closest position [13].

In [6] the comparison was made between the most popular computation algorithms that aim to solve non-linear problems present in multilateration technique. Both iterative and non-iterative approaches were tested in terms of their overall accuracy and computational time. The iterative ones presented in the work were Standard Least Squares (SLS), Hyperbolic Least Squares (HLS) and Particle Swarm Optimization (PSO). In terms of non-iterative approaches, the studies were conducted on Hyperbolic Positioning Algorithm (HPA), Bancroft and Most Likelihood Estimation (MLE). As each non-iterative algorithm returns two roots, negative and positive, each root was also investigated separately and is denoted with either minus or plus sign near algorithm name. One more approach was also proposed in [6], which is a combination of iterative HLS and non-iterative MLE methods (MLE-HLS). The results obtained in this work are presented in Fig, 2.5, in terms of accuracy, and in Fig, 2.6, in terms of computational time.

$\Delta\hat{r} \leq 1$ cm	Square	Pyram	Trapez	Mean
SLS	74.3%	50.2%	67.2%	**63.9%**
HLS	82.3%	81.4%	73.6%	**79.1%**
PSO	90.7%	91.2%	90.4%	**90.8%**
HPA^{+}	79.0%	84.2%	81.5%	**81.6%**
HPA^{-}	12.8%	15.7%	18.2%	**15.5%**
$Bancroft^{+}$	83.9%	81.4%	73.7%	**79.7%**
$Bancroft^{-}$	18.6%	18.7%	26.4%	**21.2%**
MLE^{+}	83.9%	81.4%	73.7%	**79.7%**
MLE^{-}	18.6%	18.7%	26.4%	**21.2%**
MLE-HLS	100.0%	100.0%	100.0%	**100.0%**

Fig, 2.5 Percentage of positions localized with error radius less than 1cm [6]

These results are based on experiment where time variable were calculated from the spatial geometry. The TDoA was derived from subtraction of time of flight between two receivers. Whole simulation took place in a cuboid of the dimensions of 20m x 20m x 10m The receivers were placed in a fixed known positions with a three possible distribution arrangements – square, pyramidal and trapezoidal. Source point was put in a known position and was gradually moved in 1 meter steps in such a way to cover whole volume of the cuboid. The time of flight was derived from known distance from the source point to each receiver and known propagation time. This simulation assumed that time variables are highly precise and do not include any measurement errors.

Algorithm	Square	Pyram	Trapez	Mean
SLS	7.8×10^{-1}	1.3×10^{0}	5.9×10^{-1}	$\mathbf{8.9 \times 10^{-1}}$
HLS	2.4×10^{-1}	2.5×10^{-1}	9.0×10^{-1}	$\mathbf{1.9 \times 10^{-1}}$
PSO	3.3×10^{-2}	3.4×10^{-2}	3.2×10^{-2}	$\mathbf{3.3 \times 10^{-2}}$
HPA	1.6×10^{-5}	1.2×10^{-5}	1.2×10^{-5}	$\mathbf{1.3 \times 10^{-5}}$
Bancroft	2.0×10^{-4}	1.8×10^{-4}	1.8×10^{-4}	$\mathbf{1.9 \times 10^{-4}}$
MLE	9.1×10^{-5}	8.1×10^{-5}	8.1×10^{-5}	$\mathbf{8.4 \times 10^{-5}}$
MLE-HLS	1.9×10^{-4}	1.5×10^{-4}	1.5×10^{-4}	$\mathbf{1.6 \times 10^{-4}}$

Fig, 2.6 Mean computation time of each algorithm for calculated source position, time in seconds [6]

According to [6] the simulations were conducted on computer with an Intel(R) Core(TM) i7-3630QM CPU @2.40 GHz processor, with RAM memory of 8.00 GB (7.89 GB usable) using MATLAB Version R2016b. This is a strong computer platform, compared to targeted

embedded solution used in this project. Therefore the actual computational times presented in this project will differ significantly. However, the results in Fig, 2.6 present a good measure in terms of computational complexity of each algorithm.

Based on the above data, the MLE-HLS algorithm was chosen for this project. It is by far the most accurate one, provided that the time variables are measured with sufficient precision. Furthermore it is several orders in magnitude faster in terms of execution time compared to purely iterative approaches. On smaller embedded platform this becomes crucial especially if the calculations are to be performed in real time.

2.2.1 MLE-HLS algorithm

The core part of the MLE-HLS source estimation algorithm is Maximum Likelihood Estimation, which produces two roots.

The Equation 2-1 substituted into Equation 2-2 produces quadratic equation of the form:

$$D_1{}^2 = K_s - 2x_s x_1 - 2y_s y_1 - 2z_s z_1 + K_1$$

Equation 2-4

Where K_i equals to sum of coordinates squares of i-th receiver.

$$K_i = x_i{}^2 + y_i{}^2 + z_i{}^2$$

Equation 2-5

With the assumption that the receiver number 1 is the reference according which TDoA of other receivers is calculated the Equation 2-3 can be written as:

$$D_i = D_{i1} - D_1$$

Equation 2-6

Then after substituting the above equation into the quadratic equation derived earlier for D_1

$$D_{i1}{}^2 + 2D_{i1}D_1 + D_1{}^2 = K_s - 2x_s x_1 - 2y_s y_1 - 2z_s z_1 + K_1$$

Equation 2-7

And after subtraction of the original equation for $D_1{}^2$: the final equation has the form:

$$D_{i1}{}^2 + 2D_{i1}D_1 = -2x_s(x_i - x_1) - 2y_s(y_i - y_1) - 2z_s(z_i - z_1) + K_i + K_1$$

Equation 2-8

This can be rearranged into matrix equation, with source spatial coordinates on the left side:

$$\begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} = -\begin{bmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - x_1 \\ x_4 - x_1 & y_4 - y_1 & z_4 - z_1 \end{bmatrix}^{-1} \times \left\{ \begin{bmatrix} D_{21} \\ D_{31} \\ D_{41} \end{bmatrix} D_1 + \frac{1}{2} \begin{bmatrix} {D_{21}}^2 - K_2 + K_1 \\ {D_{31}}^2 - K_3 + K_1 \\ {D_{41}}^2 - K_4 + K_1 \end{bmatrix} \right\}$$

Equation 2-9

That presents the three equations that have to be solved in order to find source coordinates. There are however only three equations given and the four parameters are unknown, namely x_s, y_s, z_s and D_1, with D_1 being the distance from reference receiver 1 to the source. In order to solve this problem the additional fourth equation is needed and it can be found in Equation 2-4. By substitution of x_s, y_s, z_s from the Equation 2-9 into the Equation 2-4, one can solve it finding the quadratic solution of D_1, which yields two results. Finding this unknown, makes it possible to solve the Equation 2-9 utilizing both results of D_1, and find two roots of source coordinates [6].

The MLE algorithm presented above finds two possible solutions. It is not easy to determine which solution is the closest to the actual source position. According to results in Fig, 2.5, positive root (MLE^+) in most cases is the closest one to the source. In order to properly determine which solution is correct MLE algorithm was enhanced by one more step that aims to evaluate it. The role of the evaluation is taken by the OF function from the Hyperbolic Least Squares Algorithm. OF function has the following form and it is obtained by applying Equation 2-3 for the reference receiver with the index of 1, to the other three receivers and by adding them together [14].

$$OF(x_s, y_s, z_s) = \sum_{i=2}^{N} (D_i - D_1 - t_{i1}c)^2$$

Equation 2-10 OF evaluation function

Where:

$t_{ij} - TDoA$ between signals reveived at i-th and j-th antenna

c – speed of sound (≈ 343 m/s)

D_1- distance between i-th antenna to the source

The reason for combining OF function with MLE algorithm is that OF function is derived from all the available information of the system, meaning it takes into account all time differences of the arrivals and also considers all spatial distances from all receivers. With the two solutions returned by the MLE being in the valid domain, the preferred solution will be the one with the lowest OF value. This is because it provides minimal deviation for all system spatial parameters and relative time calculation determinations [6].

2.2.2 TDoA calculations

The TDoA calculations for any mulilateration algorithm are usually performed by using general cross correlation (GCC). GCC investigates all the possible shifts of two signals and returns an array with values, where index of a value with highest magnitude gives the delay between two signals in a number of samples. Based on the sampling rate which was used to sample original signals, one can obtain actual time delay in seconds. The procedure for calculating time delay between two signals is as follows. Firstly both signals, on which the delay is to be estimated, are transformed into frequency domain with the help of Fourier Transform. Since signals are discrete, the FFT algorithm is applied. After that they are both combined through GCC. Then the result is normalized by the Phase Transform (PHAT) and transformed back to time domain through inverse Fourier Transform. The maximum argument in the obtained histogram indicates the delay expressed in number of samples [15].

Signals recorded from two microphones can be in form:

$$x_1[n] = c_1 s[n - \tau_1]$$
$$x_2[n] = c_2 s[n - \tau_2]$$

Where:

x_1, x_2 - microphone signals

c_1, c_2 - amplitude changes

τ_1, τ_2 - delays

k - frequency bin <0, N-1>

The GCC is defined by the formula in the frequency domain:

$$\Psi G[k] = X_1^*[k] \cdot X_2[k]$$

Equation 2-11 General Cross Correlation in frequency domain

where X_1, X_2 are Fourier transforms of x_1, x_2. In order to improve the accuracy it is beneficial to apply an N-point Hamming window on the block of samples before transforming them into the frequency domain [16].

It was shown in a work [16] that the performance can be increased further by addition of Phase Transform before inverting obtained result from Equation 2-11 back to the time domain. The Phase Transform on the GCC is expressed as:

$$\Psi P[k] = \frac{\Psi G[k]}{|\Psi G[k]|}$$

Equation 2-12 Phase Transform applied on GCC

With addition of that step all of the frequency components in the result are normalized to one and thus the phase information is preserved but the magnitude is ignored. After that the inverse transform is applied to the Equation 2-12.

$$\psi P[n] = F^{-1}\{\Psi P[k]\}$$

Equation 2-13

From the above equation the histogram P[n] is obtained. It can be read in order to obtain the time delay τ in number of samples.

$$\tau = \arg\ \max\ \psi G[n]$$

Equation 2-14

The obtained histogram can also be used for polarity correction. The value of it at τ argument can be either positive or negativie. If it is positive it means that both signals are in phase.

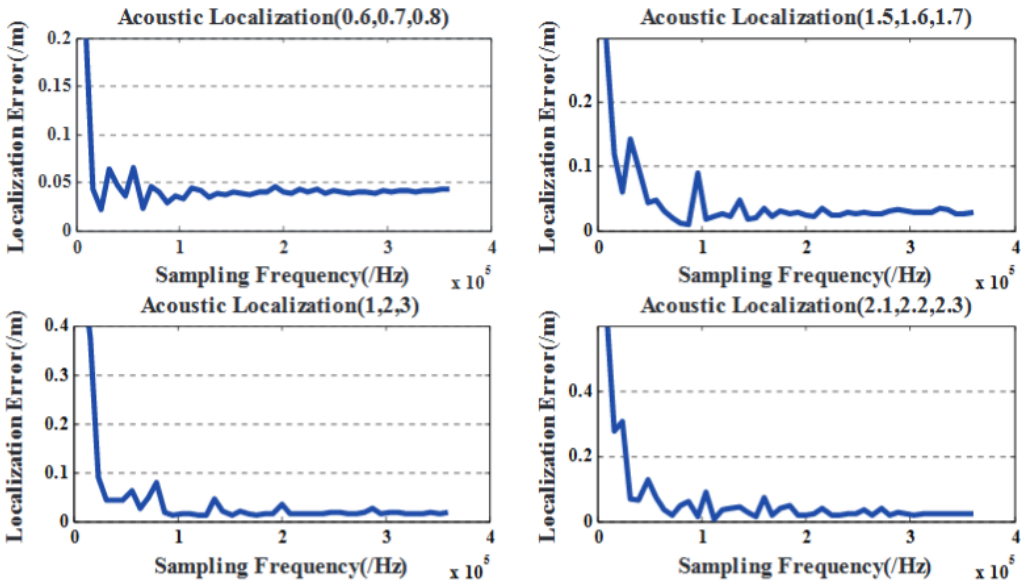
2.2.2.1 FFT block size in relation to performance

In practical implementation of the GCC-PHAT there arises a question about the FFT size in relation to methods accuracy. Since the computational complexity of the FFT rises with its block size it is beneficial to find the minimum size possible at which the result is obtained with

acceptable precision. According to [15] the best stationary performance is achieved with larger block sizes. However there is little to none gain for sizes above 65536. On the other hand sizes smaller than 1024 correspond to small increase in computation time. Therefore for general audio analysis it is recommended to use FFT sizes in a range from 1024 to 8192, which give the best trade-off between computational complexity and method's performance.

2.2.2.2 Sampling rate considerations

One more crucial issue has to be considered in real world implementation, which is the sampling rate of the signals and its effects on the TDoA calculations. Since the time delay is in fact measured in number of samples corresponding to time shift between two signals, there is a strong relation between sampling rate and accuracy of the TDoA estimation. High sampling rate at the receiver's side can contribute to high accuracy of TDoA calculations. The coordinates of the source are derived from corresponding non-linear equations involving TDoA estimations and geometric position of the receivers. Therefore accurate time delay estimations are very crucial in terms of final localization accuracy.

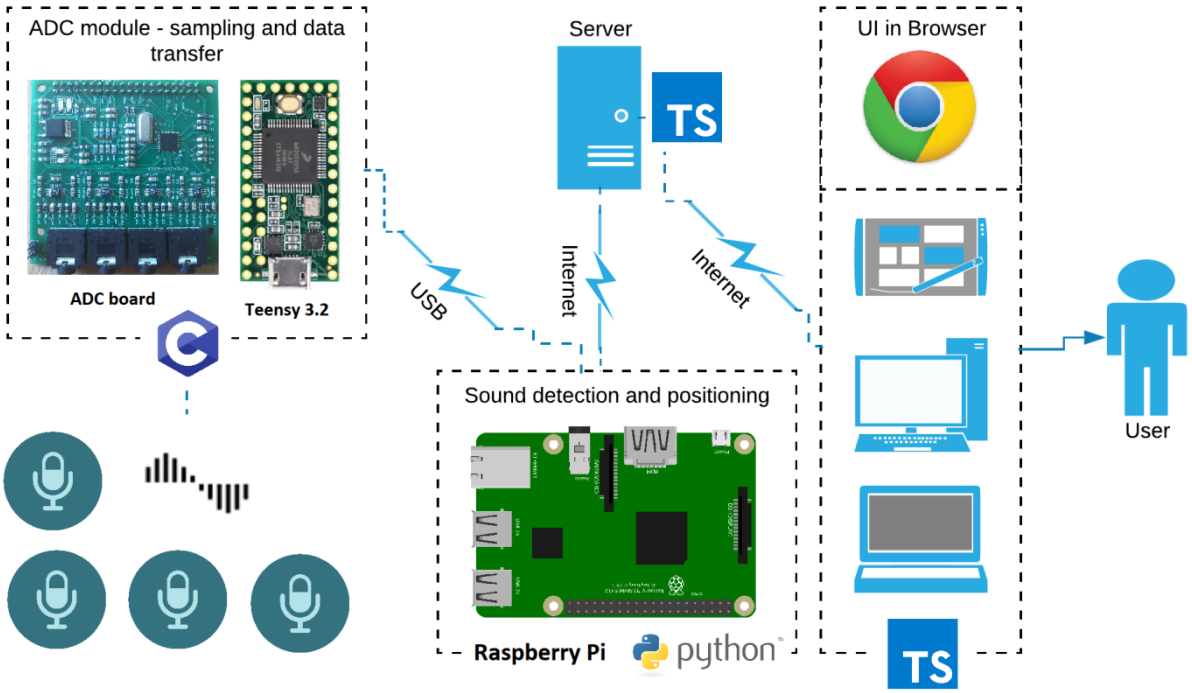


Fig, 2.7 Localization error in relation to receiver's sampling rate [17]

In [17] there was presented a study, where the sound source placed in different positions was localized and receivers were sampled with different sampling rates ranging from 8 kHz to 320 kHz. To simulate real world application the 30dB Gaussian noise was added to the source's signal. According to obtained results visible in Fig, 2.7, localization error quickly declines with the increase of sampling rate and starts to stabilize with rates over 100 kHz.

3 Practical implementation

The goal of the project is to track the location of a Ping-Pong ball by utilizing the time delays registered by the microphones placed in the surrounding space. Such a system cannot rely solely on software application as it is going to interact with the real world by registering acoustic signals in real time. Therefore it requires additional hardware in form of microphone array consisting of four microphones and analogue to digital converter able to sampling them simultaneously preserving the phase information about all signals. Those signals in digital form need to be then transferred to the main application where the MLE-HLS algorithm is implemented, to perform localization process. Finally on the software side of the project, there needs to be simple graphical user interface that reports results from each successful source position estimation, allows for basic configuration of microphone array and provides prototype of a virtual arbiter for a sound version of table tennis game.



Fig, 3.1 Overall design of the proposed solution

Top schematic of the system created in this worked is presented in Fig, 3.1. It consists of many multiple applications interacting with each other and microphone array designed especially for the project's purposes. The microphone array provides via USB the digitized audio signals coming from four microphones to the Raspberry Pi computing platform, where the raw data are processed and the MLE-HLS algorithm localizes the position of the ball whenever it hits the table's surface and emits the sound. The calculation result being the x, y, z coordinates is then transferred to the html / web socket server, which then provides user interface for any web browser client and maintains the communication between Raspberry Pi and the browser. In the following subsections, the design of each element and its role in the project is discussed in details.

3.1 Hardware

For the purpose of the project, the hardware considerations had to be made in order to construct the functional microphone array and to choose the computing platform, where the main

localization algorithm is evaluated. Due to precise application of the project, the hardware requirements fulfilling the project's needs were determined as follows:

- Able to sample at least 4 or more channels (microphones) simultaneously
- Sample signals coming from microphones at a rate of at least 40 kHz, preferably 100 kHz in order to capture audio signals raging up to 20kHz
- Able to stream digitized audio data in real time to the localization software
- Cheap and easily accessible
- As small and portable as possible

During the project development two solutions for constructing the microphone array were discussed. First approach involved utilizing internal sound card within the PC computer. On the other hand there was proposition based on external analogue to digital converter connected to Raspberry Pi computer platform.

First idea was simplistic: to utilize single PC computer to capture the microphone signals and to directly conduct calculations and display user interface. This approach however was not sufficient for the project due to the hardware and budget limitations. Probing four or more microphones at the same time requires sound card with at least four separate line or ideally microphone inputs. The microphone input in the sound card can be connected nearly directly to simple electret microphone due to the fact that such input provides bias voltage for the microphone, necessary for its operation, and gain to amplify received signal. Use of microphones designed for that input is beneficial as they are usually the cheapest on the market and easily accessible. Unfortunately sound cards are usually equipped with at most two of such inputs and their separation is often not guaranteed effectively reducing total number to only one. The line inputs on the other hand are designed for higher voltages as they do not provide amplification or any sort of power supply for the connected device. Typical line input is

- -10 dBV for consumer equipment (0.316 V RMS)
- +4 dBV for professional equipment (1.23 V RMS) [18]

This implies purchase of more expensive studio quality microphones or pre-amplifiers which significantly impacts the overall cost. Furthermore from size perspective of the overall design it

is not desirable solution either. There is also worth to note that USB microphones are not considered for this kind of application. One cannot guarantee that all USB type microphones are going to be synchronized, therefore making the small time delays between different microphones unobtainable or invalid.

Another approach was based around the design and manufacture of own hardware components, which could reduce the costs significantly in the expense of additional complexity and labour. It also allows for manufacturing of optimal hardware elements specifically tailored for the purpose of the project. Furthermore PC computer was not a requirement, so more suitable calculation platform could be chosen. In this case Raspberry PI 3B+ was used as it combined strong processor and support for Linux operating system, which could handle any possible GUI application apart from raw calculations. Ultimately this solution was chosen with two separate hardware components to be designed: ADC board and microphone.

3.1.1 Raspberry Pi

Raspberry Pi is a single board computer platform, which is developed by the Raspberry Pi Foundation. There are currently three generations of the board, with the latest being the third one. Each consecutive generation contains more powerful chipsets and differs significantly from previous one. The hardware has gone through several versions that feature variations in memory capacity, peripheral-device support and overall board shape and design. Furthermore there are various models inside each generation that differ in processing capabilities and RAM memory size, while retaining the same board size and similar peripheral design. [19]

In this project the Raspberry 3B+ model was utilized from the last generation. This device is based on Broadcom BCM2837B0 system on chip (SoC). It contains the 64 bit Cortex A53 processor with 1.4 GHz clock, VideoCore IV GPU unit and 1 GB RAM memory. The Raspberry devices do not contain any sort of flash memory on the board. Instead they rely on SD cards to store the operating system and program memory. In terms of operating system device is capable of running multiple Linux distributions and special Windows IoT Core. The official one called Raspian provided by the Raspberry Pi is a modification of Linux Debian tailored specially for the hardware present on the board. In terms of peripherals, it features four

USB 2.0 ports and thus can support many external USB based devices. Furthermore this device is equipped with WiFi antenna and therefore can be utilized as a fully featured IoT device. All specifications of the board are listed below:

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- Extended 40-pin GPIO header
- Full-size HDMI
- USB 2.0 ports
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- 4-pole stereo output and composite video port
- Micro SD port for loading your operating system and storing data
- 5V/2.5A DC power input Power-over-Ethernet (PoE) support (requires separate PoE HAT) [20]

Fig, 3.2 Raspberry Pi model 3B+, used in this project [20]

3.1.2 Teensy 3

The Teensy platform is a microcontroller development board. In the market it stands out as it offers powerful AVR or ARM microcontrollers while still maintaining very compact size. It

can be directly compared to the Arduino type boards. It offers full support for Arduino IDE including Arduino based libraries. Therefore it is easily accessible and fast for developing prototype projects. One can distinguish two generations of Teensy boards. The Teensy 2.X with 8-bits AVR processors and Teensy 3.X with 32-bits ARMs. All of the boards have built in USB port which allows for either flashing the compiled program or for a full speed USB data transfer. The board used in this project is Teensy 3.2. Its full specifications are listed below:

- 2-bit ARM Cortex-M4 72MHz CPU (M4 = DSP extensions)
- 256K Flash Memory, 64K RAM, 2K EEPROM
- 21 High Resolution Analog Inputs (13 bits usable, 16 bit hardware)
- 34 Digital I/O Pins (5V tolerance on Digital Inputs)
- 12 PWM outputs
- 7 Timers for intervals/delays, separate from PWM
- USB with dedicated DMA memory transfers
- 3 UARTs (serial ports)
- SPI, I²C, I²S,CAN Bus, IR modulator
- I²S (for high quality audio interface)
- Real Time Clock (with user-added 32.768 crystal and battery)
- 16 DMA channels (separate from USB)

This board was utilized in the project in form of middleware between ADC and the Raspberry Pi board. This type of architecture was chosen in order to allow proper readings from ADC in real time, which proved to be impossible relying solely on Raspberry board. Since the Raspberry hardware was controlled by the operating system, the fast interrupt generated by the ADC could not be handled in required amount of time resulting in obscured data. With the usage of Cortex M4 the data could be properly handled and then sent via USB connection to the Raspberry board for further calculations.

3.1.3 ADC Board

The analogue to digital converter board is the part of microphone array counting up to four microphones. Its destination is to sample the analogue signals obtained on each microphone

and to stream obtained data to the main processing unit, which is raspberry Pi 3B+ computer. Hence the board is primarily designed as an extension board for raspberry Pi, it is connected to it via its external pin header. Despite high integration with raspberry platform the goal was to make it as versatile as possible in order to conduct proper equipment tests and detect possible design flaws. Consequently the board is equipped with 4 analogue 3.5mm jack inputs and is suitable for line level analogue input signals. This setup allows for sampling any audio signal generated from most common sources like PC computer or smartphone. The communication between controller and the board itself is done through an SPI, which is popular and fast synchronous communication protocol. Again it was chosen to provide easy test environment as it provides an opportunity to easily connect the board to basically any microcontroller based device, not only raspberry PI platform.

The board is also equipped with a few utility features such as jack input detection or LEDs both destined for power indicator and as arbitrary one to be used by the user. Full list of design requirements is as follow:

- Compatibility with Raspberry 3B+ pin header
- Dimensions similar to Raspberry PI board
- Communication through SPI
- 5V power supply
- Power supply current of less than 1A
- Four jack mono inputs designed for line level signals
- Jack input detection
- Possibility of providing 5V supply or bias voltage via jack input
- LEDs for power indication and for optional use

3.1.3.1 ADC chip choice

There are different kinds of ADC chips and one can distinguish many categories based on their specification. In this application the most essential property is to sample multiple channels simultaneously. Typically most of the chips are equipped with multiple sampling channels. However one has to note that those integrated circuits usually have multiplexer that precedes

actual converter module and thus they convert each channel sequentially [21].This kind of architecture makes simultaneous conversion much harder or impossible as it requires synchronizing multiple ADC chips or compensation for samples lost due to multiplexing. To avoid these problems special group of ADC integrated circuits was chosen called simultaneous sampling ADCs. This kind of architecture guarantees that all parts of the available channels are sampled in parallel. In this category there are several solutions presented by Texas Instruments, Maxim Integrated or Analogue Devices. Ultimately MAX11043 was chosen as it supports SPI communication protocol instead of parallel one and it was the cheapest option available. The other chips like AD7761 could be potentially utilized in this project. They are however unnecessarily complex, containing up to 8 channels and are enclosed in bigger packages.

MAX11043 is 4 channels 16 – 24 bits simultaneously sampling ADC with SPI communication protocol. It is based on four independent sigma-delta ADC modules with versatile filter block and optional programmable-gain amplifier. Overall block diagram of the ADC chip is presented in the Fig, 3.3. According to datasheet MAX11043 is able to digitize simultaneously 4 signals with frequencies up to 200 kHz, where the limit is defined by the SPI interface throughput, which allows for sending up to 1600ksps. Therefore it is possible to sample at even faster rate, if less than 4 channels are active and number of samples does not exceed $1.6 \cdot 10^6$ per second in total. The sampling rate is derived from the frequency of an external crystal or ceramic resonator connected to OSCIN pins. This frequency can be than divided by 2, 3, 4 or 6 in configuration register to generate actual ADC sampling clock. As the chip is based on delta-sigma converters, sampled signal passes through decimation filters, therefore final sample rate is affected by those filters. MAX11043 features two decimation filter blocks, one which decimates by 12 and one optional, configurable by the user which can decimate by 2 or be bypassed. Taking this to account final sampling rate can be calculated by the following formula, where

f_s – final sampling rate

f_{in} – input crystal frequency

d_{config} – configuration register divider(2, 3, 4 or 6)

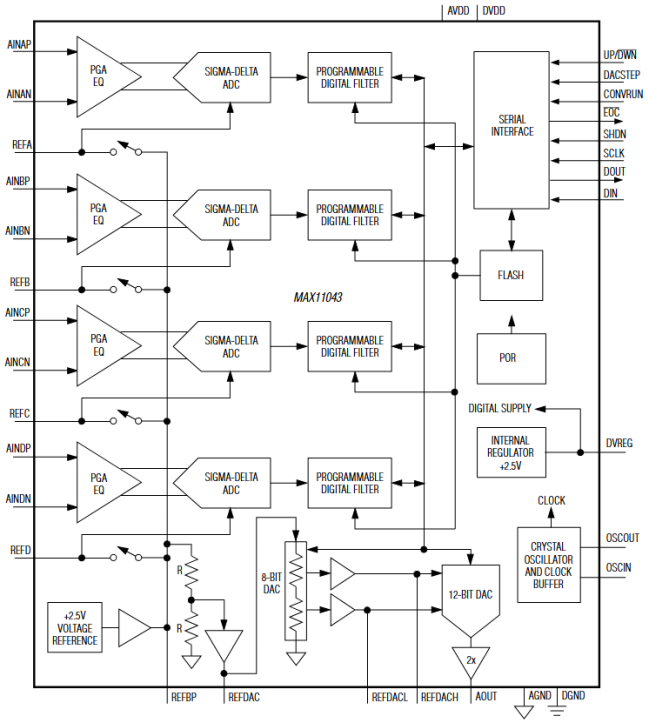
d_{filter} – second decimation filter value: either 1 or 2

$$f_s = \frac{f_{in}}{d_{config} \cdot d_{filter} \cdot 12}$$

Equation 3-1 ADC sampling frequency

ADC is also equipped with a digital filter, which consists of seven cascaded second order filter sections present on each channel. All filter coefficient and gain amplifiers are configurable and are independent for each channel. Filter can be configured either as low pass, band pass or high pass.

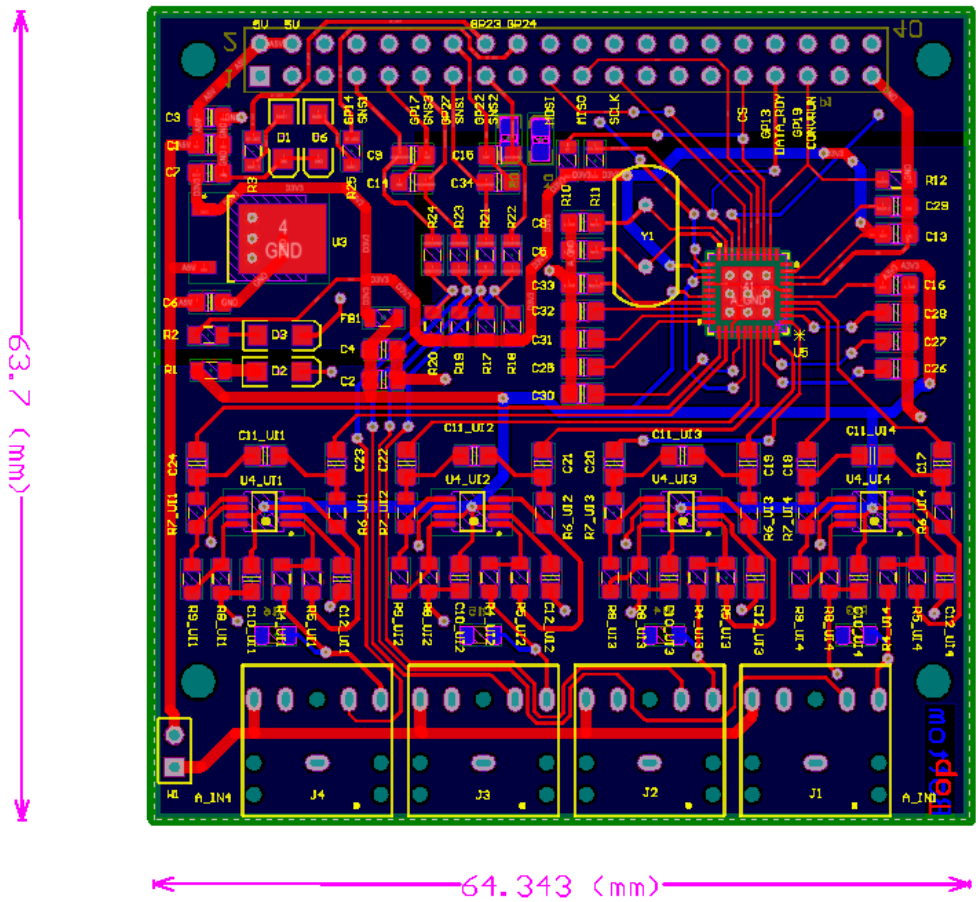
In terms of data transmission MAX11043 also features user selectable scan mode which gives the ability to read data faster. When scan mode is present on given channel the result is immediately present on the DOUT (MISO) pin as soon as chip select pin is asserted low, which signalizes beginning of SPI communication. This makes it possible to omit passing the result register address to the chip and thus simplifies and speeds up the process of reading a measurement.



Fig, 3.3 Functional diagram of MAX1043

3.1.3.2 Board overall design

The board is designed to meet previously presented requirements in section 3.1.3. As the board contains digital and analogue signals, the special care has to be taken in order to separate both kind of signals and avoid additional noise in analogue part of circuitry. Therefore it is designed to handle mixed signals, having power sources and grounds separated for digital and analogue part of the board. The top level schematic is depicted in the Fig, 3.5.



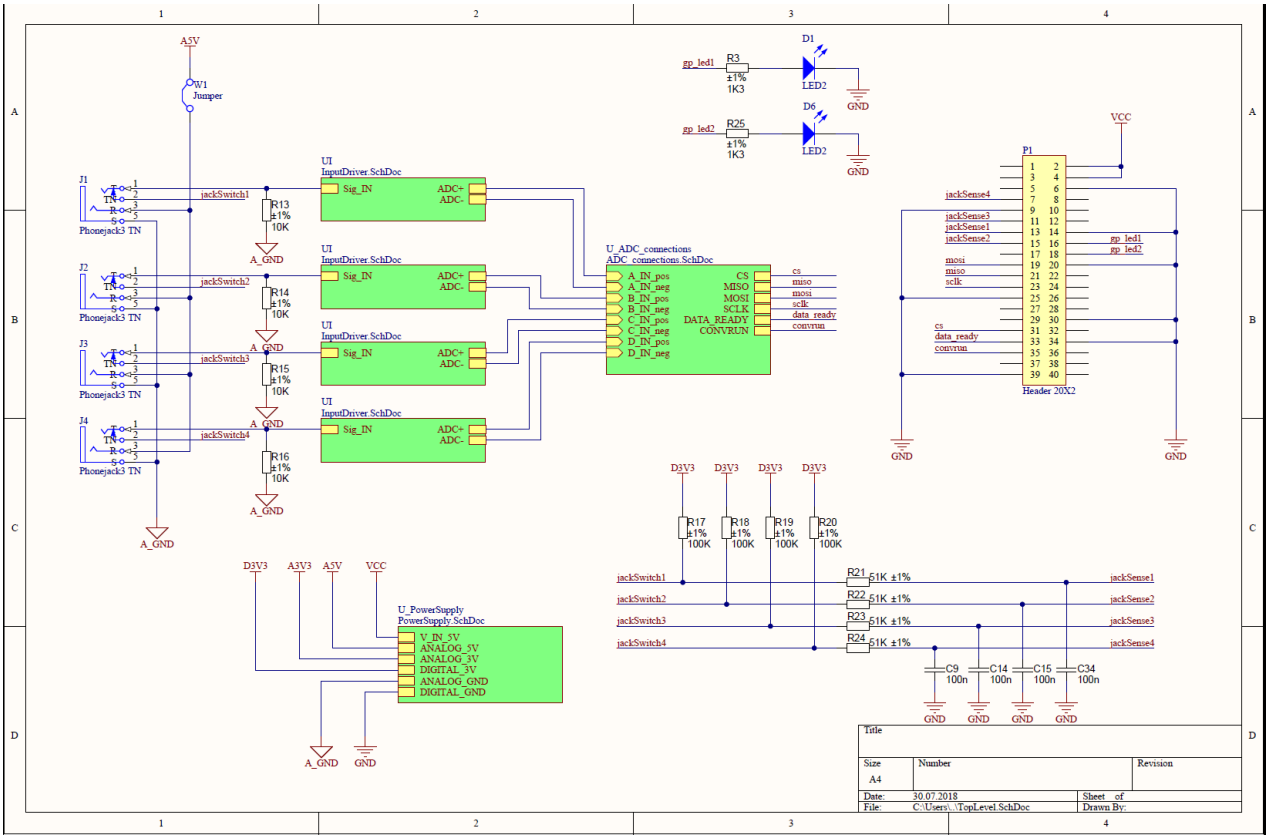
Fig, 3.4 ADC board designed for the project, all layers viewed from the top

The heart of the design is MAX11043 analogue to digital converter and its basic circuitry including decoupling and filtering capacitors. Analogue signals are provided by 4 audio stereo jacks with switches. Tip of the audio jack is used for actual audio signal transmission, which is passed further to ADC input driver module and it is pulled down to analogue ground via 10k resistor (R13 to R16 on the schematic). ADC input drivers convert single ended audio signal to

differential and provide basic filtration. Differential signal paths are then connected to the ADC through 10uF capacitors, which is a proposed design from ADC's datasheet.

Jack input detection is implemented with the help of the switch present on each female jack connector. Switch pin is connected to the 100kΩ pull-up resistor (R17 to R20) and to the output header pin through low pass filter. When the male jack connector is not present in the socket, the switch is connected to an audio input line. Therefore the output of jack sensing circuitry can be read as digital low. When male jack is inserted into the socket the switch is open, and the output is pulled up, therefore giving digital high signal. The low pass filter is present in order to block audio signals from obscuring readings on the microcontroller GPIO's connected to the output. Filter cut-off frequency is set to be 31 Hz which is sufficient for its purpose of filtering out audio signals.

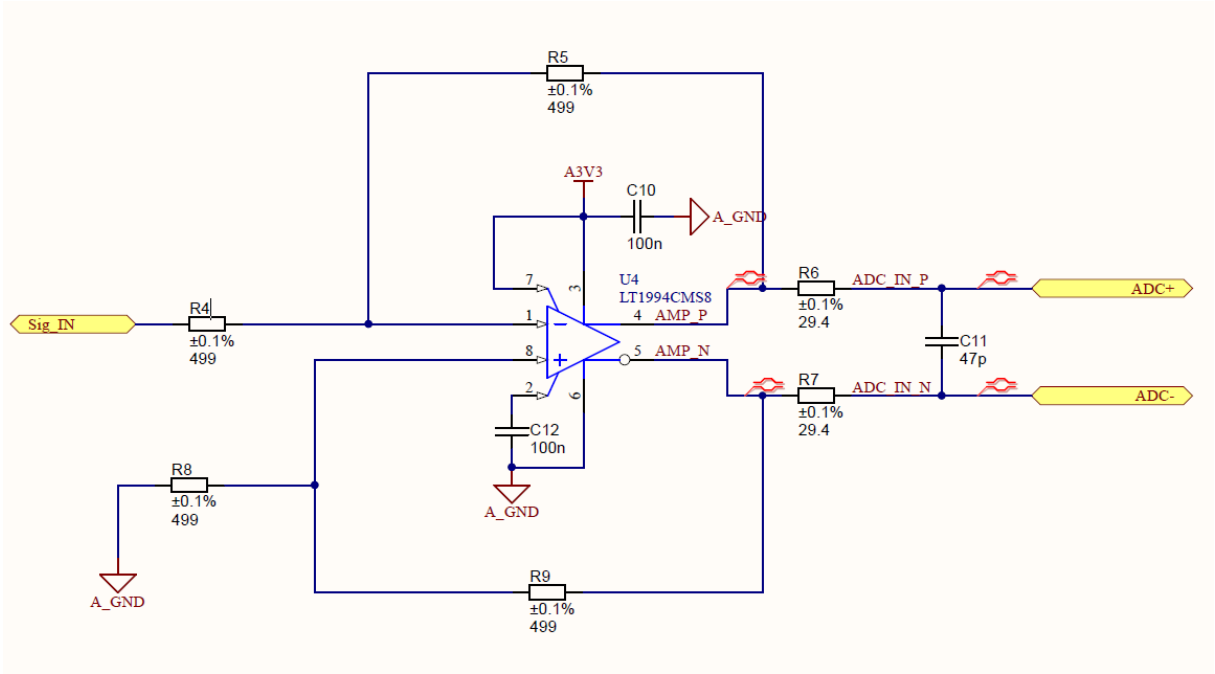
Output signals are connected to 20x2 pin output header. This allows for direct connection to a Raspberry Pi 3B+ board, without any external cables. All output routes are connected to that connector including SPI communications lines and jack sensing circuit. Furthermore, two LEDs are present on the board that are connected to the output through current limiting resistors (R3, R25). Those LED's can be used by the user in arbitrary way.



Fig, 3.5 Top Level Schematic of the PCB board for ADC

3.1.3.3 ADC driver design

The goal of the input driver circuit in this design is to convert single ended signals to differential ones and to provide initial passive filtering. Such configuration was chosen in order to match MAX11043 fully differential inputs. It is possible to feed single ended signals to the ADC, however in order to minimize distortions and achieve higher signal to noise ratio, differential signal approach was chosen. Utilizing differential signals allows for even-order harmonics cancellation and less susceptibility to common mode noise, like for instance noise present in power supply. Moreover differential circuits are more resilient to outside interference and crosstalk from nearby paths present on PCB. This is the case since the noise impacts tightly coupled traces equally, cancelling each other out. Finally differential signals produce less electromagnetic inference themselves. The reason behind this is that signal level changes produce opposing magnetic fields and in consequence they cancel each other out. [22] The schematic of input driver sub circuit is visible in Fig, 3.6. Four such blocks are present on the board, one for each analogue input.



Fig, 3.6 ADC input driver schematics

Driver is designed around fully differential amplifier manufactured by Linear Technologies – LT1994. It is rail to rail, low noise amplifier which can operate under single supply voltage source. On top of that supply voltage can range from 2.375V up to 12.6V, meaning that it can be used with low voltage applications such as in this case. Additionally Linear Technologies provides extensive datasheet with many typical application examples including single ended to differential conversion. In the application from Fig, 3.6, input signal is converted to differential equivalent with unity gain. The amplifier is powered from single 3.3V analogue voltage line destined only for analogue part of overall circuit. Since the 3.3V power supply any incoming signal will be clipped between 0 – 3.3V, which means that it can be safely passed to the ADC without risking any damage. The resistors on the differential paths have to be equal, meaning that $R_6 = R_7$, $R_4 = R_6$, $R_5 = R_9$. It is worth to point out that more accurate resistors were used with 0.1% error margin in order to minimize resistance differences between differential path pair. The gain is calculated using the following formula:

$$G = \frac{R_5}{R_4} = \frac{499\Omega}{499\Omega} = 1$$

Having unity gain the transfer function of the circuit can be expressed as follows:

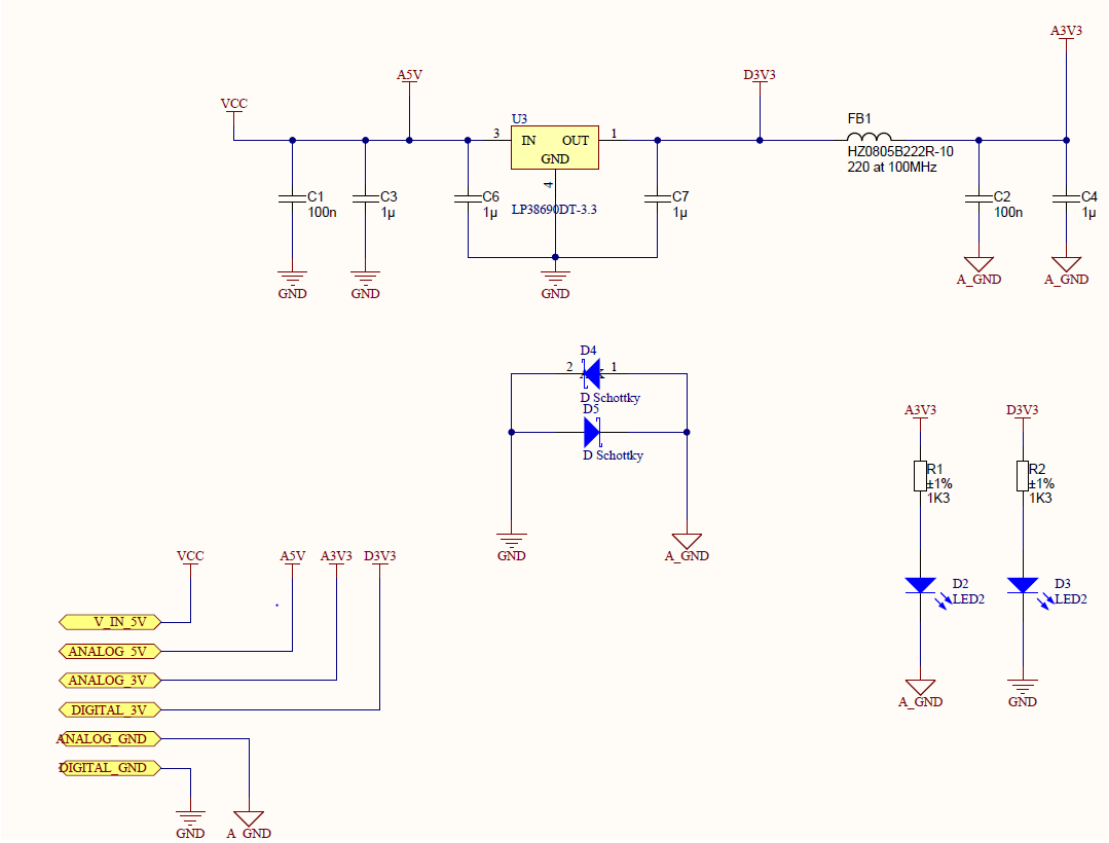
$$V_{OUT,DIFF} = V_{OUT+} - V_{OUT-} = V_{IN},$$

V_{OUT} – output voltage, V_{IN} – input voltage

Additionally circuit features decoupling resistors R_6, R_7, which also create filter in conjunction with C_{11} capacitor in order to reduce high frequency noise. The resistors are necessary as the outputs of the LT1994 operational amplifier are designed to drive 25pF capacitive loads per each output. In case of higher capacitances at least 25Ω resistor should be included. This is the case in the presented application since the ADC requires 10uF capacitors on each input.

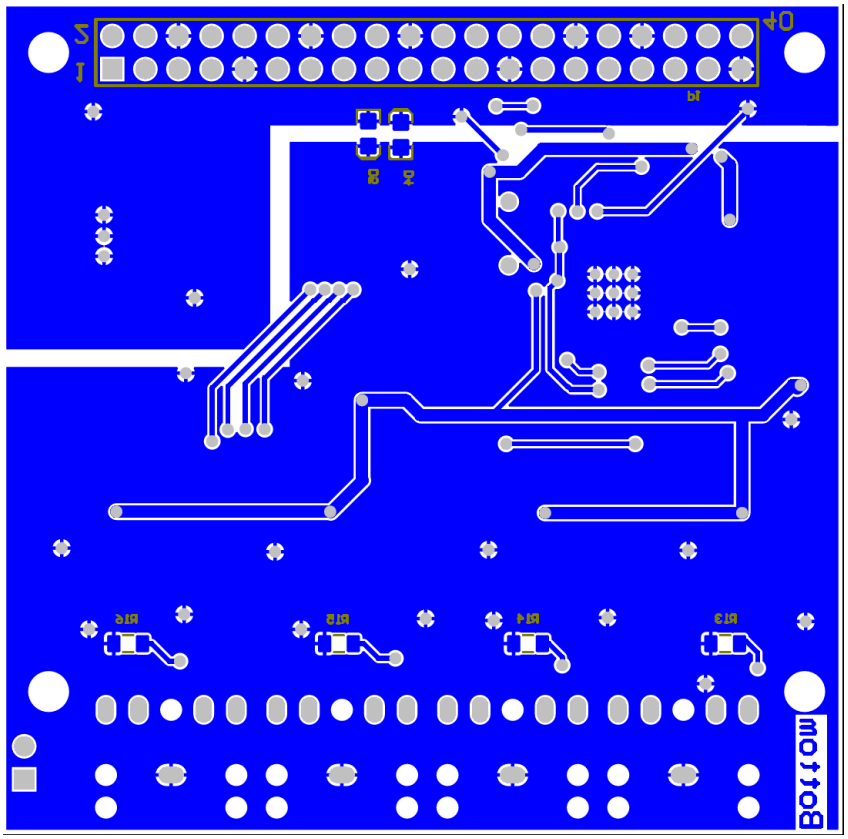
3.1.3.4 Power Supply

The power supply part of the board is designed to operate with the input voltage of 5V and to not exceed current draw of 1A. Such input voltage is dictated by the Raspberry Pi board which operates on 5V power supply and has 3.3V logic. This and the fact that ADC also requires 3.3V dictates the need for voltage level shifting and dealing with two voltage supply levels. Furthermore as mentioned earlier, board represents an example of the mixed signal circuit, therefore analogue supply has to be separated from a digital one and two separate ground planes are required. The grounds in the mixed signal circuits are kept separate to avoid interference from digital signals into the analogue circuits. The ground plane on the PCB board acts as a low impedance return path, which is helpful for decoupling high frequency currents created by the usually fast digital logic. Utilizing ground plane also helps to reduce emissions of electromagnetic interference. [23]



Fig, 3.7 Power supply schematic of the ADC board

Having above guidance in mind the power circuit in the design features separate 3.3V power supplies for analogue and digital part of the circuit called D3V3 and A3V3 on schematic respectively. A 5V input voltage is converted to 3.3V using linear voltage regulator. This voltage is then directly used as the digital power supply. On the other hand analogue counterpart is buffered by a ferrite bead. Usually it is a good idea to utilize different linear regulators for creating the analogue and digital power sources; this solution however increases the overall costs and size of the projects. Therefore ferrite bead is utilized to provide some noise isolation without the equipping the board with multiple voltage regulators.



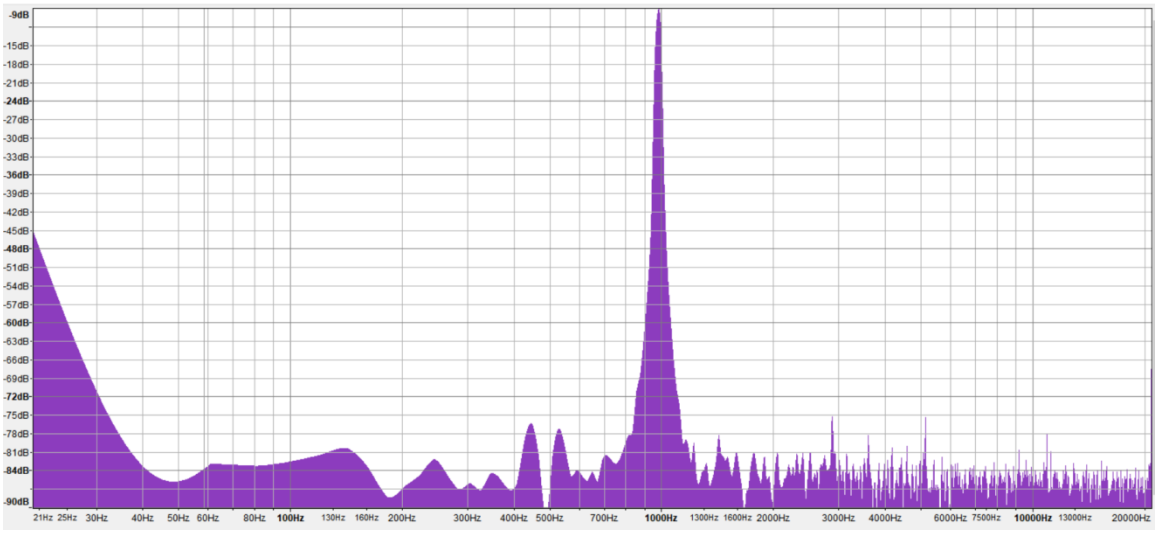
Fig, 3.8 Bottom layer of the ADC board

In terms of ground topology it consist of two separated planes connected in a single point on the PCB. The layout and bottom layer is visible in the Fig, 3.8. Bottom part of a plane is destined for analogue part of the design, where upper is for digital circuitry. Single point connection between the planes is done by the two Schotkyy diodes which are placed back to back. This kind of solution keeps ground-potential difference at below 0.3V.

3.1.3.5 Tests and possible improvements

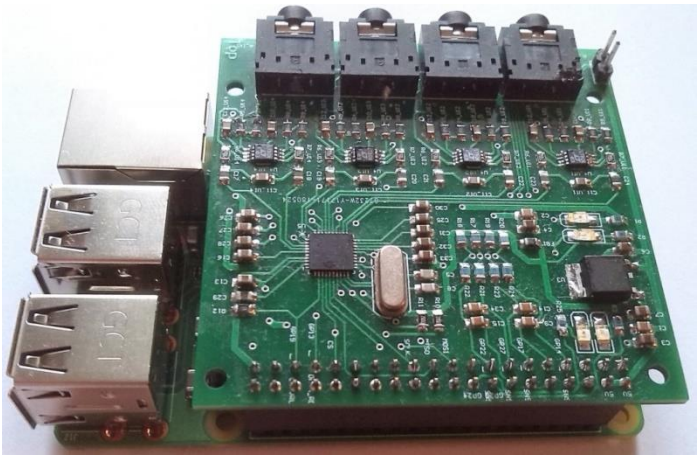
The board itself meets the requirements stated in the previous section. It performs quantization of four different analogue signals simultaneously and is easily powered and controlled by the Raspberry PI. The performance test of the board itself was done by feeding one channel with 1 kHz sine wave generated by the computer connected to the board via jack cable. The spectrum of the sampled signal is presented below. The 1 kHz sine wave is clearly dominant frequency component as expected, but the spectrum also shows that some undesired noise is present in the sampled signal. This can be a result of a noisy power supply, which was taken directly from

USB port on the Raspberry PI. Despite that fact for the purpose of sound sources localization the noise level is acceptable as the signal presence can still easily be detected and further digital processing is possible.



Fig, 3.9 Spectrum of 1 kHz signal sampled with ADC, 2048 FFT size, Hanning window

There were however shortcomings in the overall design that were overlooked and therefore project required additional changes in its principle of operation. Mainly the data communication was too demanding to be handled by Linux operating system as stated in the section 3.1.2. Fast interrupt on external pin header could not be handled in real time by the Linux operating on the Raspberry Pi and therefore additional middleware hardware was needed to act as a buffer.



Fig, 3.10 ADC board on top of the Raspberry Pi 3B+ as in the initial design

To address this issue the board was redesigned into a USB device with additional microcontroller that was focused on reading the data from the ADC chip via SPI and then to buffer it and send using USB protocol to the host. Also, the board current consumption and input voltage level of 5V allowed for powering it directly from USB. This change transferred the board from specific platform hardware to a universal design that can be connected to any platform provided that sufficient driver software for its serial interface is present on the host. To put these changes into practise the Teensy 3.2 board was used as it already has necessary USB hardware and powerful cortex M4 on the board. With these components already included the functional prototype could be manufactured without redesigning the PCB itself. The Teensy could be wired to the board via its external header with the help of the breadboard. Additional advantage of choosing Teensy was its compatibility with the Arduino IDE, which significantly reduced additional programming overhead introduced by the new middleware.

Another issue found during tests was the power supply for the externally connected microphones. Those supply lines are nearly directly connected to the overall board's power supply input. The board is mainly supplied from USB or from main power grid via rectifier and boost converters, so in practice the voltage ripple at the input is relatively high. This is in consequence has a high influence on sensitive analogue circuitry like a microphone resulting in large amount of noise in the audio signal. In order to mitigate this problem, the microphones were supplied directly from batteries instead. It was possible as the board has a jumper (W1 in the Fig, 3.5) on the power line to the jack connectors, therefore allowing for external power supply for connected devices. This solution also made it possible to use higher voltage to power microphones and as a result to diminish noise gain of the microphones due to change in bias resistor value.

3.1.4 Microphone Board

In order to acquire and listen for a specific signal, the microphones had to be deployed and connected to the ADC board. This however imposed using the preamplifier in order to pick signals of the useful voltage level, as the board was designed for line-level inputs. Such amplifier could be done independently from microphone itself, but to further reduce the size

and costs of the overall project, the microphone was chosen to be incorporated into the project as an electronic element rather than consumer – ready product.

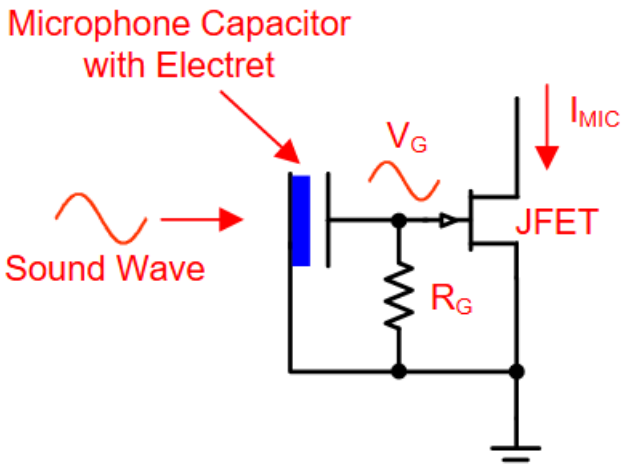
3.1.4.1 Design requirements

The requirements of the microphone board were as follows:

- Single 9V battery power supply
- Line level output signal (1V RMS)
- Small current consumption
- Gain deviation at frequency spectrum 20 – 16 kHz less than -0.5 dB
- Output if form of a cable of at least 1m length ended with 3.5mm jack

The type of microphone chosen for the purpose of the project was an electret one. That kind of microphones are common in consumer electronic as they are small, have good frequency response and are in affordable prices. The name "electret" comes from the material used in those microphones. This material has a fixed charge and it is utilized to separate the two electrodes creating a capacitor. Sound waves or in other words changes in the air pressure move one of those electrodes and in consequence change its distance in relation to the other electrode, thus modulating the capacitance of the structure. Consecutively the voltage on the capacitor changes producing an electrical signal, as the charge on the microphone is fixed.

Simplified schematic of the circuit built inside an electric microphone is shown in the Fig, 3.11. Sound produces a voltage, which in turns modulates the gate voltage of the JFET (V_G). This causes the fluctuations in the current flowing between drain and source of the JFET (I_{MIC}).



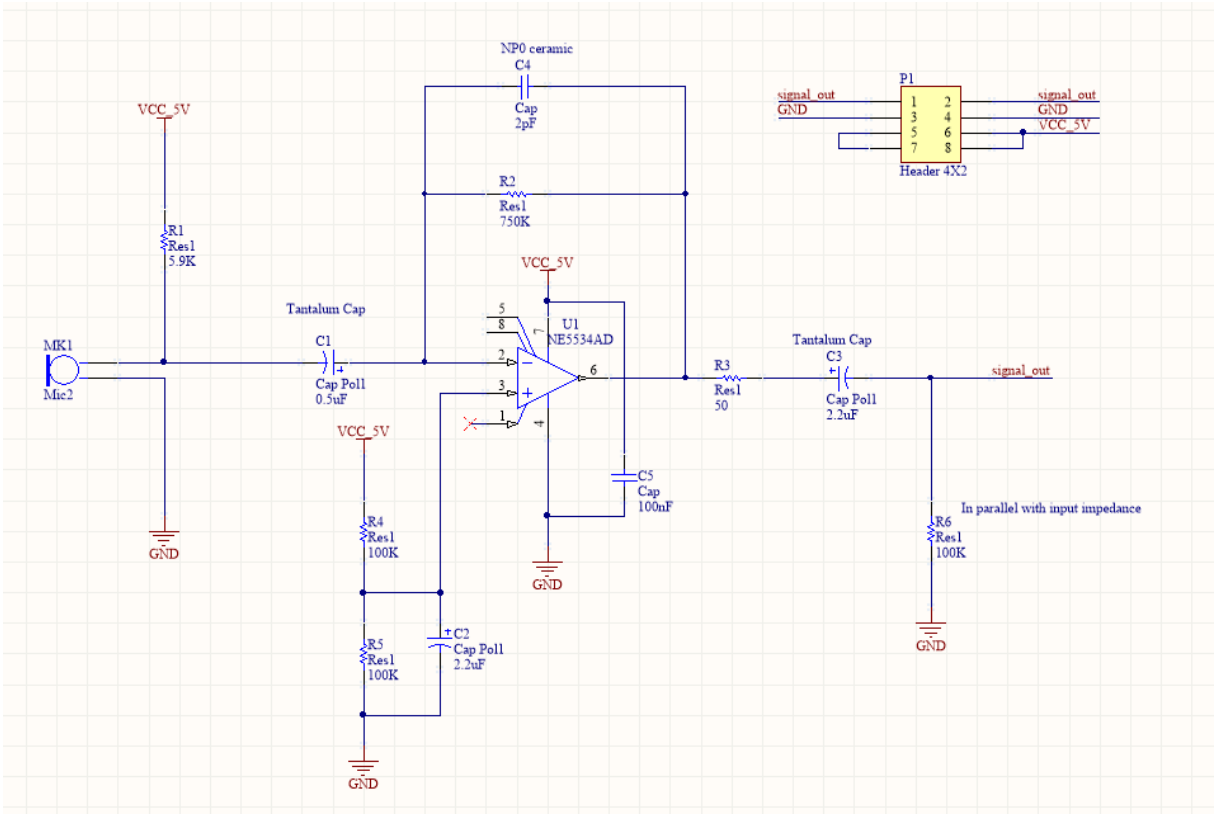
Fig, 3.11 Basic schematic of circuitry inside electret microphone

3.1.4.2 Circuit description

Based on theory, the basic circuit of the pre-amplifier can be constructed, which can be seen in Fig, 3.12. As the microphone requires bias for the internal JFET, the I_{MIC} current has to have both AC and DC components. To allow the current to flow through the capacitor C_1, its impedance has to be much lower at audio frequencies than R_1. The U_1 operational amplifier is in trans-impedance configuration and it is keeping its inverting input at a constant voltage by varying its output. Due to that fact, its output voltage can be expressed as a function of AC component of I_{MIC} and R_2 with the increment of the bias voltage applied on positive input.

$$V_{OUT} = I_{AC} \cdot R_2 + V_{BIAS}$$

Therefore the gain of this amplifier corresponds directly to the value of R_2 and its noise gain is determined by the ratio of R_2 to R_1. The circuit also features two filters, an active low pass created by R2 and C4 and a passive high pass constructed by C3 and R6. On top of that capacitor C1 blocks the DC component of the I_{MIC} current, and R5



Fig, 3.12 Microphone preamplifier schematic

3.1.4.3 Component values selection

To calculate the proper values of the component one has to firstly look into datasheet of the microphone itself. For this design the Loudity LD-MC-6035P was chosen. Its parameters are listed below:

- Sensitivity: -35dBm (0dB = 1V/Pa; 1 kHz)
- Operating voltage: 2V
- Impedance:2.2kΩ
- Frequency: 50-16000Hz
- Current consumption: < 0.5mA
- Signal to noise ratio: > 60dB
- Voltage range: 1-10V

The sensitivity of the microphone was measured at 1Pa and it is expressed in logarithmic scale using decibels. The sensitivity can be also expressed in volts per Pascal:

$$10^{\frac{-35dB}{20}} = 17.78mV/Pa$$

To get the value of current I_{MIC} flowing in the circuit per Pascal, one has to divide the obtained value by the microphones impedance.

$$\frac{17.78mV/Pa}{2.2k\Omega} 8.083\mu A/Pa$$

Knowing the current caused by the air pressure acting on the microphone, the gain of the amplifier can be calculated. However, firstly the assumption has to be made about maximum air pressure that the microphone is expected to encounter. For the purpose of the sound localization it was chosen that 90dB sound picked at the microphone should result into 1.228V (RMS) at the output, which is the line level voltage. The 90dB translates into 0.69Pa, which in turn means I_{MIC} current has to be:

$$I_{MIC} = 0.69Pa \cdot 8.083\mu \frac{A}{Pa} = 5.582\mu A$$

Therefore the resistor R2 responsible for the gain can be calculated as:

$$R_2 = \frac{1.228V}{5.582\mu A} \approx 220k\Omega$$

The capacitor C4 forms a filter with a resistor R2, furthermore it compensates for parasitic capacitance of the inverting input of the amplifier. The cut-off frequency of the filter has to be chosen in such a way to limit its impact on the microphone's spectrum. The deviation of -0.1dB (0.989) in pass band was chosen as acceptable. Therefore the cut-off frequency is expressed as:

$$f_c = \frac{f}{\sqrt{(\frac{G_0}{G_1})^2 - 1}} = \frac{16kHz}{\sqrt{(\frac{1}{0.989})^2 - 1}} = 106.98kHz$$

G_0 – gain at low side of the spectrum (20Hz)

G_1-gain at frequency f, which is the high side of microphone's spectrum (16 kHz)

Using this frequency, one can calculate the value of C_4, which is:

$$C_4 = \frac{1}{2\pi R_2 f_c} = \frac{1}{2\pi \cdot 220k\Omega \cdot 106.98kHz} \approx 7pF$$

For a proper operation the microphone needs to be biased. This is provided through the R_1 resistor. Since the 9V power supply was assumed (VCC) for the circuit and based on

microphone's operating voltage (V_o) and its current consumption (I_s), value of the R_1 can be expressed as:

$$R_1 = \frac{VCC - V_o}{I_s} = \frac{9V - 2V}{0.5mA} = 14k\Omega$$

As for a capacitor C1 it forms a high pass filter with the resistor R1. The cut-off frequency needs to be low enough to pass frequencies in the microphone's spectrum. The 0.5μF value is sufficient as it produces the corner frequency of 20Hz, which is suitable for the application purposes.

$$C_1 = \frac{1}{2\pi R_1 f_c} = \frac{1}{2\pi \cdot 14k\Omega \cdot 20Hz} \approx 0.5\mu F$$

Additional bias is also needed for the amplifier in order to fully pass both the negative and positive half of the AC-coupled signal with the single power supply. This bias is usually set to half of the power supply voltage to allow widest possible voltage swing. The resistors R4 and R5 have to be equal, but their value can arbitrary. They were chosen to be 100k to restrict the current flowing through them. One has to consider the capacitor C2 to filter out thermal noise generated by the resistors. This forms a low-pass filter with R4, R5 and C2, which cut-off frequency should be below microphone's spectrum. The 2.2uF value is sufficient for that purpose:

$$fc = \frac{1}{2\pi(R_4||R_5)C_2} \approx 1.4Hz$$

Finally, the output of the microphone's pre-amplifier has to also be AC-coupled, therefore the DC component introduced in the amplification process has to be removed and it is done by the C_3 capacitor. Apart from the capacitor two additional resistors R_3 and R_6 are included. R_3 limits the current flowing through the capacitor, while the R_6 provides a discharge path for it, which could be potentially discharged when connected to external equipment. The R_6 is also creating the high pass filter with C_3. Its cut-off frequency needs to be low enough to not attenuate microphone's spectrum. One has to also note that R_6 is in parallel with the impedance of the external device. R6 was chosen to be 100k and the combined impedance of R6 and external circuitry was assumed to be 10k. The value 2.2uF was assigned for C3, which provides sufficiently low cut-off frequency:

$$f = \frac{1}{2\pi(R_6||R_{EXT})C_3} \approx 7.2Hz$$

This leaves only the choice of the operational amplifier itself. There are many parameters that have to be considered during the choice of the amplifier. These include slew rate, gain bandwidth, spectral noise density and power supply. The minimum slew rate can be determined by calculating the maximum rate of change of the highest frequency component of the microphone's spectrum, which is 16 kHz, at line level. Line level was defined earlier as 1.228V rms, which equals to 1.736V peak to peak. Therefore required slew rate Rs is expressed as:

$$R_s = 2\pi fA = 2\pi \cdot 16kHz \cdot 1.736V = 0.1745V/\mu s$$

Naturally for a safe margin value at least ten times more is considered for real application, which is 1.745V/µs.

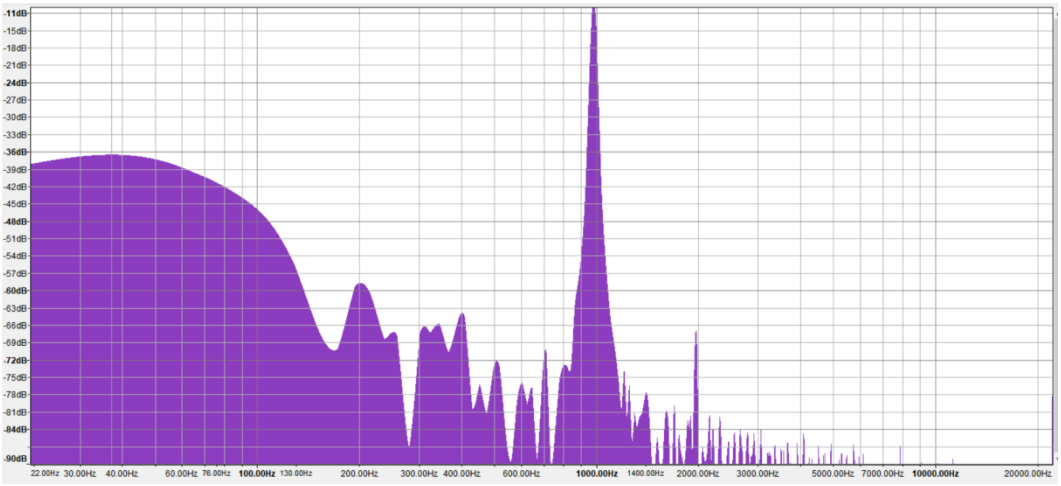
For the final design the NE5534 AD from Texas Instruments was chosen. It is an amplifier destined for audio applications therefore it is bound to perform well as a microphone pre-amplifier. Its specifications are as follows, making it sufficient for this application

- Gain bandwidth: 10 MHz,
- Slew rate: 13V/us
- Supply-Voltage range: 3 – 20V
- Equivalent input noise current: 0.4 $\frac{pA}{\sqrt{Hz}}$

3.1.4.4 Tests and performance

As with the ADC board, the microphone's performance was assessed similarly by recording 1 kHz sinus signal in a quiet room environment. The board was supplied by 9V battery and was connected to the computer analogue input on the sound card. The 1 kHz sine wave was generated by the PC computer and was played on speakers in close vicinity of the microphone. Obtained spectrum from recorded audio is presented in Fig, 3.13. As it can be seen the 1 kHz signal is clearly visible on the signal's spectrum. However one can also notice distinguishable interference in the lower side of the spectrum. This noise can come from a number of sources, but most significant part of it is a result of 50 Hz power line interference present in the surrounding environment. Part of copper on the microphone board can act as an antenna and

pick up the noise, which is then transferred via crosstalk to the signal paths on the board. To reduce the impact of this kind of noise, the unconnected copper was connected to the ground, thus reducing the 50 Hz noise amplitude. Another possible source of relatively high amplitude at low frequencies is the noise pollution present in the environment. Even though environment was quiet, some noses can still be picked up by microphone, like for instance PC cooling fan. All things considered, the microphone design still meets the project requirements as noise present in higher frequencies is marginal and valuable signal can be distinguished easily from lower frequency noise.



Fig, 3.13 Spectrum of 1 kHz signal recorded by microphone via line input on the PC, 2048 FFT size, Hanning window

3.2 Software

Continuously localizing the ball within the sound table tennis game is a demanding job that requires real time calculations done in a continuous loop. The tasks to be done within a single loop in the software algorithm can be divided into three main phases: data acquisition, recognition of the searched sound, which is bouncing ball, and finally estimation of the sound source position with the help of MLE localization algorithm. Additionally user interface is required in order to keep track of the overall game progression and to provide feedback and some means of configuration, debugging to the end user. The graphic, depicting different software components and technologies used, is presented in the Fig, 3.1.

This kind of project requires software that can be sub dived into three main parts. Firstly the Teensy had to be programmed to control the behaviour of ADC, gather the digitized samples and send them via USB to the host. Second part consisted of calculation module responsible for executing main localization algorithm in a continuous real time loop and reporting the results to the final piece of software, which was the user interface. All of them due to the different nature of tasks were written with the help of different technologies and independently from each other. This resulted in three separate programs on two platforms: embedded and Linux.

3.2.1 ADC driver

ADC driver is an embedded application, which targets Cortex M4 processor present on the Teensy 3.2 development board. The board itself is compatible with the Arduino IDE with the help of few extensions, therefore this environment was used to develop functional program. The application's core modules are written in the native C despite the presence of avr g++ compiler contained within the Arduino IDE. This along with function wrappers for hardware functionalities is done in order to allow better portability between different embedded platforms as most of them are programmed in native C and hardware related functionalities depend on the targeted device.

The goal of the application is to configure registers of the max11043 ADC, read data sampled by the device, and transfer them to USB buffer. The ADC itself has a digital pin responsible for starting the continuous conversion. Whenever it is in digital high state, the ADC is converting input signals, meaning it continuously stores freshly sampled data in output registers and indicates that new sample is ready. This indication is done by another digital pin called "data ready". This pin is in normally low state, and is pulled high when new result is ready. The controller has to read the data immediately initiating SPI data transfer by pulling chip select line low. After that, with the help of the scanning mode, content of result register is already being pushed on SPI data lines without the need for result register address. Precise timing is required as data are only valid for a short amount of time when the data ready pin is pulled low.

```
1.  if(digitalRead(DATA_READY_PIN) == LOW && !dataReady){
2.      spi_cs_low();
3.      spi_read_back((char*)buff, SAMPLE_SIZE);
4.      spi_cs_high();
5.      dataReady = true;
6.  }
7.
8.  if(Serial.dtr() && dataReady){
9.      channelData = buff[0] << 8 | buff[1];
10.     Serial.write(buff, SAMPLE_SIZE);
11.     dataReady = false;
12.   }
```

Fig, 3.14 Main loop of the application, top level function

Application works in a continuous loop, during which it is polling the data ready pin of the ADC. If it is low, the result is read from the SPI into the buffer. When the buffer contains new sample, data ready flag is raised and the sample can be passed to the USB hardware buffer. Additionally DTR line is checked to ensure that the USB port is opened by the application on the USB host.

Another approach could be to use external interrupt triggered on falling edge of the data ready signal. This however, was seen as unnecessary as the application performs only one task and it is not worth additional overhead required by entering interrupt execution context.

```
1.  void serialStatus(){
2.    static bool isConverting = false;
3.    bool serialStatus = !Serial.rts() && Serial.dtr();
4.
5.    if(serialStatus && !isConverting){
6.        MAX11043_read_samples_cont();
7.        isConverting = true;
8.    }
9.    else if(!serialStatus && isConverting){
10.       MAX11043_stop_reading_samples();
11.       isConverting = false;
12.   }
13. }
```

Fig, 3.15 Timer function, responsible for checking USB connection status

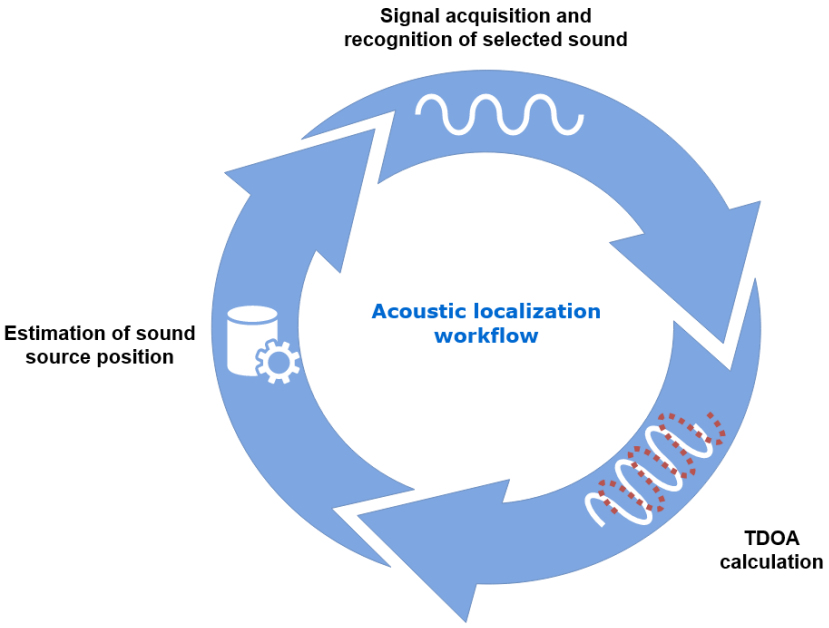
To control the behaviour of the ADC from the USB host, the additional control signals are utilized in the application. These are DTR (data terminal ready) and RTS (ready to send). Both lines are inspected with the help of the timer interrupt that periodically checks their digital value. Whenever the RTS is low and DTR is high, the conversion pin of the ADC is pulled high

marking the beginning of the continuous conversion. If the state of either RTS or DTR lines changes, the conversion is stopped and ADC is set in the idle mode.

3.3 Main localization algorithm

The main localization process takes place on the Python application within the Linux environment. In this project the Raspberry Pi was used as a computing platform to run Linux operating system. The Python programming language was chosen as it offers vast variety of specialized libraries for a signal processing and allows for easy matrix based calculations. Furthermore it offers convenient ways to implement both USB serial port data read and web socket internet communication. From the above reasons the Python was perceived as the best tool to implement functional prototype. There are however some shortcomings of this choice that have to be noted. Python is dynamically typed scripted language and also its concurrent capabilities are hindered by global interpreter lock, which means that only one thread can be executed at any giving time. This makes it not an ideal solution for real-time applications due to limited performance in comparison to languages that are compiled to machine code.

The application works in a continuous loop waiting for control messages from the user interface sent via web socket communication. When a command to start ball positioning is received, the localization process begins. The control RTS and DTR signals of the USB connection are set to low and high state respectively and the ADC driver starts sending the raw data to USB port of the Raspberry PI.



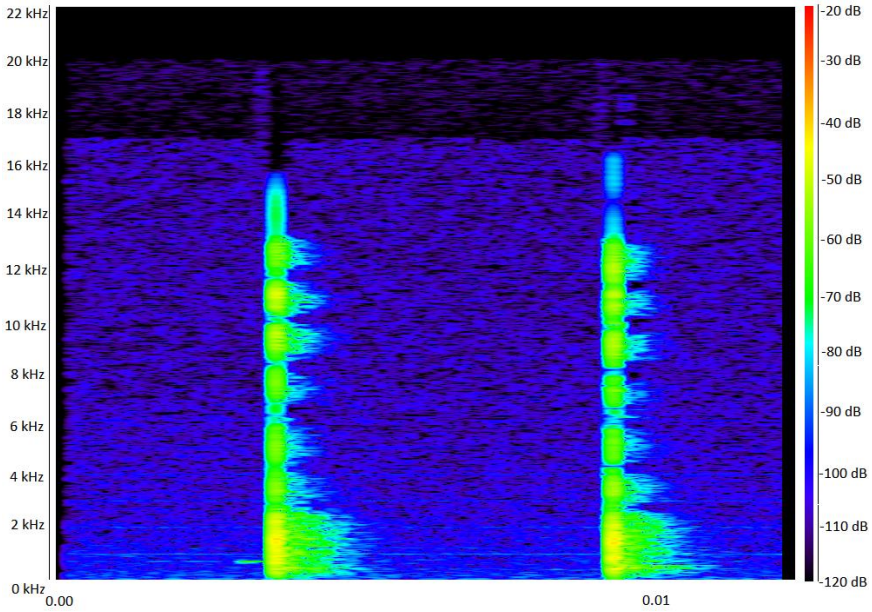
Fig, 3.16 Top level workflow of python calculation module

The raw data is continuously streamed from the USB serial port in chunks of 4096 16 bit samples per channel. It totals to 32 kb per single chunk for all channels and results in a transfer of over 333 kb/s as the signals are sampled with 41.666 kHz rate. Raw data is then unpacked to a structure represented by a list of 8 byte ADC sampling results. Each result consists of four interleaved 16 bit samples from four separate channels of the ADC board. The data are divided into their separate channels respectively. The program now checks whether the bounce sound occurred within the acquired samples. If it is found to be true, the next part occurs, which is to calculate delays on each channel according to reference channel. With time delays being calculated, the MLE-HLS algorithm is applied in order to estimate final space coordinates of the ball, which emitted the searched sound. These coordinates are then reported back to the user interface via web socket communication. This process is repeated infinitely in real time until the stop message is received from user interface.

3.3.1 Sound Recognition

Recognizing certain audio patterns in a stream of real time sound data is a challenging task on its own. Perfect sound detection and distinction between various sounds that can occur in the environment of table tennis game is out of the scope of this thesis. Therefore this process was

simplified and was done purely on the inspection of the spectrogram of the sound emitted by a bouncing ball. Below spectrogram containing two bounces of table tennis ball is presented.



Fig, 3.17 Spectrogram of two bounces made by table tennis ball

As it can be seen on the spectrogram, the ball has a distinctive frequency pattern. This sound has a wide spectrum ranging from low frequency components up to about 16 kHz. This wide spectrum and the significant magnitude in higher frequencies make this sound pattern easily recognizable even with some ambient noise. Most of the audio sources that can affect the detection in project's environment occupy lower part of the frequency spectrum. The noise in this case is expected to come from human related activities such as speech or footsteps. Both of the mentioned sources occupy mainly lower frequencies with voice ranging from 300 Hz to about 3 kHz, and up to 1 kHz for the latter case. This makes it ideal to simplify whole detection problem to simple threshold on a certain frequency band. Based on the spectrogram the 7 – 12 kHz frequency band was chosen for the sound detection purposes.

In the application, the detection process is realized based on the threshold both in the time and frequency domain. In order to save computational power and to discard sections of the audio containing silence, the sound detection algorithm was implemented. It relies on calculating the envelope of the signal. Envelope provides relatively smooth outline of the signal and allows to

reliably detect whether the given audio section has desired amplitude, thus it is not considered as a silence. The envelope e[n] for signal s[n] is expressed as:

$$e[n] = \max(s[n], e[n-1] \cdot f_r)$$

Where f_r is a release factor such as $0 < f_r < 1$ and in this application fr = 0.9993 resulting in a very smooth signal outline.

Since sound signals tend to fluctuate heavily, even such large release factor could not smooth them entirely. Therefore in order to avoid multiple detections of crossed threshold for an envelope function, single threshold was replaced by upper and lower thresholds providing some buffer. Now, sound is registered whenever envelope crosses upper threshold at least once and it is bound to end only when the lower threshold is crossed. The sound event can be properly bounded between two distinct samples in the time domain, marking the start and the end of it precisely.

ADC outputs all four channels at once. Signals coming from them differ not only in phase but also in amplitude, which depends on the distance from the emitted sound. This brings an issue as arbitrary channel cannot be used with strict amplitude threshold. To avoid this problem and to make sound detection consistent, the channel with the highest energy is chosen for envelope calculations. Energy of a discrete time signal s[n] with N samples is expressed as:

$$E_s = \sum_{n=0}^{N} s[n]^2$$

When the sound of given amplitude was found it is then extracted from the raw samples and analysed further to check if its spectral signature is similar to expected ball bouncing sound. It is divided into smaller parts which are then transformed into frequency domain effectively creating spectrogram of the provided audio. To save computational time the FFT size used is only 64, as bigger spectral resolution is unnecessary. The spectrogram is then converted to logarithmic scale. Each part in the time axis of the spectrogram is analysed in terms of the frequency axis. If multiple fragments having sufficient average magnitudes in the range from 7 kHz to 12 kHz exist, one can identify that particular sound event as a ball bounce. The actual

value of this threshold is determined empirically by inspecting the recorded samples of the sound and evaluating what value the algorithm has a tendency to produce in project's environment and setup. With the sound confirmed to be desired event and with known beginning of it on one channel, the further calculations of TDoA can take place.

This is by no means the ideal solution to detect such sound. Many ambient noises can also have high frequency components of strong enough magnitude and similar spectral pattern to trigger the detection algorithm. One such example can be finger snap. In order to properly distinguish both sounds more complex solution is required, preferably in form of neural network algorithm. This is however beyond the project's scope and trigger based solution is sufficient to tackle the main topic of sound localization.

3.3.2 Implementation of MLE – HLS

After the event of bouncing ball is detected, the program computes its main objective thus attempts to find the spatial coordinates of a ball that emitted the sound. Firstly, based on the signal from four microphones, their relative time delays are calculated with respect to the reference one, which is arbitrary chosen to be the first channel. This is done by the general cross correlation with additional phase filtering that was described in section 2.2.2. The sound detection from algorithm returns exact positions of the start and end of a sound event. Only close neighbourhood of the start point is used in order to minimize possibility of error and reduce the computational complexity. For the purpose of GCC the size of the FFT was chosen to be 512. Assuming n is the sample number where the sound starts; the GCC is computed on signal on a range < n – 256, n + 256) for all channels. Such choice allows for detecting delays up to 256 samples. In combination with sampling rate of 41.666 kHz and sound propagation speed in the air of 343 m/s, this restrains the maximal distance of microphones up to about 2m.

The Python implementation of GCC is visible below. Additionally all the Fourier transforms are filtered with Hanning window and optional interpolation factor was added, which influences the size of the reverse Fourier transform. By increasing the interpolation factor over 1, the samples are zero-padded in the frequency domain by the multiplicities of the transform size. This results in interpolation and thus better resolution of the result in a time domain.

Function returns both the result in seconds and histogram generated by the inverse Fourier transform, which was used to calculate the delay. All channels are correlated with first one to find their respective delays to the reference first channel.

```
def gcc_phat(input_signal: np.ndarray, ref_signal: np.ndarray, dft: DFT, phat: bool = False,
             delay_in_seconds: bool = True, interpolation_factor: int = 1,
             buffered_dft: bool = False, force_delay: bool = False) -
> Tuple[float, np.ndarray]:
    """Performs General cross correlation in frequency domain with optional Phase Transform filt
ering(PHAT).
       Returns a Tuple of delay(in sec or samples) and resulting histogram"""

    fft_signal = dft.transform(input_signal)
    if buffered_dft:
        fft_ref_signal = ref_signal
    else:
        fft_ref_signal = np.conj(dft.transform(ref_signal))

    corr = fft_signal * fft_ref_signal

    if phat:
        denom = np.ones(corr.shape, corr.dtype)
        tmp = np.abs(corr)
        denom[tmp != 0] = tmp[tmp != 0]
        corr = corr / denom

    histogram = dft.inverse_transform(corr, interpolation_factor)
    if force_delay:
        histogram[0] = 0
    histogram = np.fft.fftshift(histogram)

    delay = (np.argmax(histogram) -
 dft.size // 2 * interpolation_factor) / interpolation_factor

    if delay_in_seconds:
        delay = delay / dft.sampling_rate

    return delay, histogram
```

Fig, 3.18 Python implementation of GCC-PHAT

With all four delays known, the spatial position itself can be found based on MLE-HLS algorithm described in section 2.2.1. To integrate it into programming environment and to gain full advantage of its non-iterative nature, the problem has to be rearranged into single quadratic equation. The four unknowns, which are x, y, z coordinates of the source and the distance from reference microphone to the source, form a set of four equations. For readability let assume that first microphone is the reference and substitute constants related to the microphone's positions and their relative delays with matrices C, R and N.

Substituting Equation 2-5 for source coordinates into Equation 2-4 yields:

$$D_1{}^2 = x_s{}^2 + y_s{}^2 + z_s{}^2 - 2x_s x_1 - 2y_s y_1 - 2z_s z_1 + K_1$$

Equation 3-2 Equation for distance from first microphone to the sound source

D_1 – distance form first microphone to the source

x_s, y_s, z_s – sound source coordinates

x_1, y_1, z_1 – coordinates of first microphone

K_1 – sum of squares of coordinates of first microphone as stated in Equation 2-5

Equation 2-9 can be simplified to the form of:

$$\begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} = \begin{bmatrix} N_{xs} + D_1 + R_{xs} \\ N_{ys} + D_1 + R_{ys} \\ N_{zs} + D_1 + R_{zs} \end{bmatrix}$$

Equation 3-3 Simplified equation for sound source coordinates with first microphone set as reference

Where C, R, N are matrices such that:

$$C = -\begin{bmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - x_1 \\ x_4 - x_1 & y_4 - y_1 & z_4 - z_1 \end{bmatrix}^{-1}$$

$$N = \begin{bmatrix} N_{xs} \\ N_{ys} \\ N_{zs} \end{bmatrix} = C \times \begin{bmatrix} D_{21} \\ D_{31} \\ D_{41} \end{bmatrix}$$

$$R = \begin{bmatrix} R_{xs} \\ R_{ys} \\ R_{zs} \end{bmatrix} = C \times \left(0.5 \cdot \begin{bmatrix} D_{21}{}^2 - K_2 + K_1 \\ D_{31}{}^2 - K_3 + K_1 \\ D_{41}{}^2 - K_4 + K_1 \end{bmatrix} \right)$$

The TDoA in relation to first microphone is involved in variables D_{21}, D_{31}, D_{41}, which are distances between first microphone, chosen as a reference, and the rest of the receivers. According to Equation 2-3 TDoA relation to the distance, distance between microphones can be expressed as a TDoA - t_{ij} between them, multiplied by the propagation speed of a sound - c, therefore:

$$D_{ij} = t_{ij} \cdot c = D_i - D_j$$

In order to create single quadratic expression, the variables xs, ys, zs from Equation 3-3 are substituted into Equation 3-2. After some rearrangements this creates the standard quadratic equation of from:

$$a \cdot {D_1}^2 + b \cdot D_1 + c = 0$$

Equation 3-4

where variables a, b and c have a form:

$$\begin{cases} a = N^T \times N - 1 \\ b = 2\left(N^T \times R - N^T \times \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}\right) \\ c = -2R^T \times \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + R^T \times R + K_1 \end{cases}$$

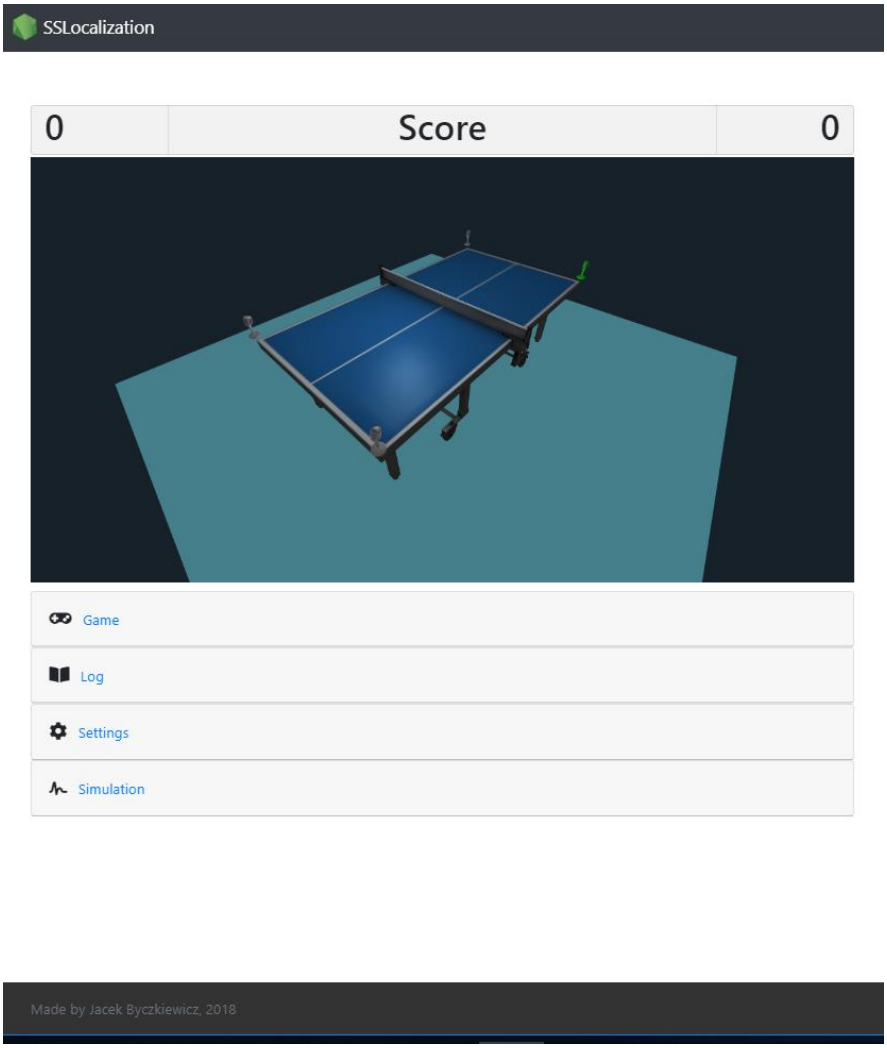
With those variables being easily computable using matrix multiplication, one can simply calculate the distance D_1 and substitute result back into Equation 3-3. The Equation 3-4 yields two solutions for unknown D_1. To determine which is more likely to be the desired solution, the HLS part of the algorithm is used, which is described in section 2.2.1. The obtained two sets of solutions for x_s, y_s, z_s are then reported to the user interface via TCP web socket connection. The root chosen by the HLS algorithm is placed in the first position of the array.

3.4 User interface

Technically speaking the sound localization problem is realized by the Python application. The coordinates of the bouncing ball are returned whenever the sound is emitted. However for the end user, which is visually impaired table tennis player, the raw coordinates does not represent any value. This kind of user requires the game to be tracked as a whole. This means some score representation and basic foul detection system indicating possible illegal moves in the game. Furthermore the Python application needs to be updated with actual positions of microphones in real world in order for the result to have any real meaning. Finally, to easily spot errors and to track game progression, visual debugging tool is necessary. To match these objectives the user interface was designed for this project.

There is large variety of approaches in terms of user interface design, but for the purpose of this application the html web interface was chosen. It offers most versatile solution that can be viewed easily on many different devices and does not require any installation process from a user. Additionally it is run on client device therefore it does not take much computational power on a platform performing sound localization. Finally html5 supports handy text to speech functionalities built in it and it is supported by most modern browsers. Such an environment helps to develop test application with audio instructions for visually impaired users.

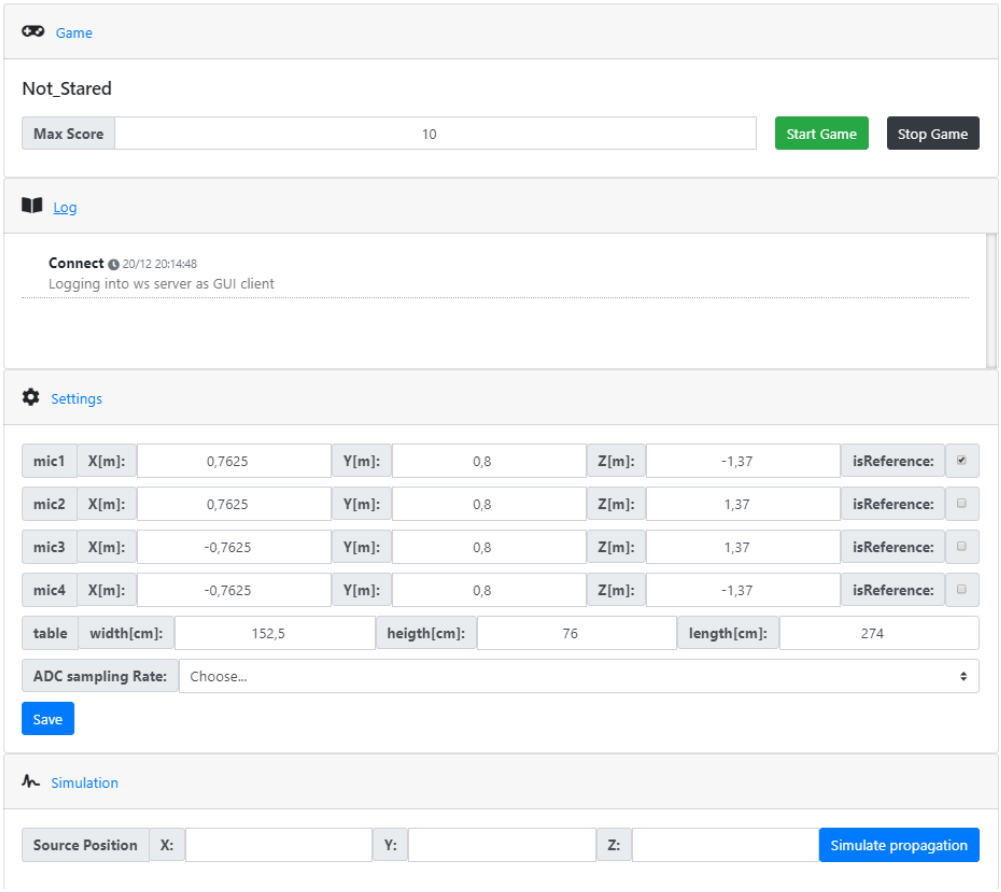
The interface itself consists of a part run by the client in the browser and backend http, web socket server hosting the page while also managing the connections between browser GUI and python worker application. The page displayed by a browser is visible in the figure below.



Fig, 3.19 Graphical interface, destined for debugging and configuration

On top of the webpage, there are two sections responsible for a scoreboard and visualization of the real world tennis table and the placement of microphones. The scoreboard updates whenever the game arbiter algorithm assigns clear score for a given player. This is supposed to happen only if two consecutive bounces are registered on the side of player serving the ball, at least one is located on the other half of the table and opposing player does not catch a ball. The latter event can be detected as ball bouncing on the floor. If the sufficient timeout passes without the sound event being detected, the ball can be considered to be caught by another player. When the score is assigned or game status changes it is announced by the text to speech feature of the hml5. This helps to interact, the visually impaired players with the interface. Of course it is not an ideal option, which servers mainly for demonstration purposes and needs to be expanded with additional functionalities to serve as complete product for such demanding user group. These include some ways for the user to interact with the interface without using its vision. It could be for instance, speech recognition API, or hardware buttons near the table, which helps to navigate through user interface options.

Additionally below the scoreboard, the 3D visualization module is located. It serves mainly debugging purposes as it helps to visualize the position in real world that matches given set of coordinates. The returned position by the sound localization algorithm is in relative coordinates system, where point value (x = 0, y = 0, z = 0) represents the centre of the game table. To match real world table, the virtual one is scaled to match the dimensions of real object. Therefore all distance relations are preserved giving a hint whether the ball position is properly computed. When a result is reported from worker application, the newly obtained ball location is marked with red sphere, which vanishes after short timeout.



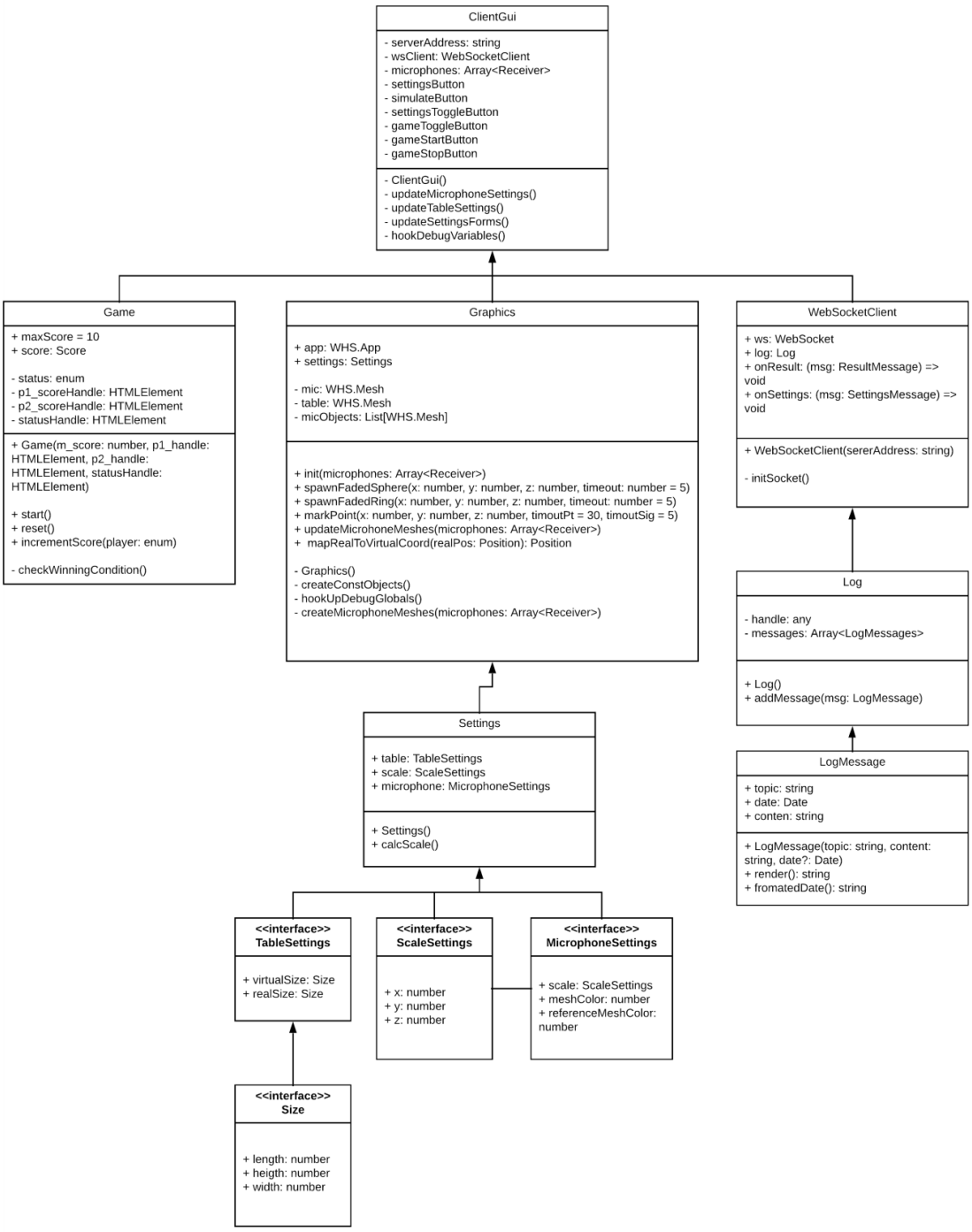
Fig, 3.20 Set of configurable options

The webpage has a simple modular design with four main configuration modules namely: game, log, settings and simulation. Game tab provides a simple configuration of the table tennis game. It displays its status and allows for setting maximum number of points before game ends and the winner is announced. Log supplies the additional debugging information about connection details and received messages from the server. The purpose of the settings tab is to set the microphones coordinates in real world, that are sent to the worker application and to configure dimensions of the game table to allow properly display ball position inside the simulation. Finally, the simulation tab supplies the debug feature to bypass the real world data and simulate the propagation of the sound virtually and recalculate its position using the MLE-HLS algorithm. This is purely debugging feature added to check independently for errors in worker's algorithm or its responsiveness.

3.4.1 Implementation details

In terms of technology, both server and webpage are programmed in Typescript language. Typescript is a superset of JavaScript maintained by the Microsoft Company. Its main advantage over plain JavaScript is the support for static typing. Therefore it adds extra layer of security to the code, while maintaining all the advantages of the JavaScript. To be run on the browser it is then compiled again to the plain JavaScript. In this project the server and GUI are merged together in a single Typescript project. However it includes both the code that runs in Node JS runtime environment (server itself) and the browser. Those are two different compilation targets for Typescript compiler. Because of that, the whole code is compiled for the NodeJS environment and the part executed by the browser is then compiled again by the Browserfy tool to the code compatible with browsers.

Browser application is divided into several classes, which have different responsibilities. Full class diagram is visible in Fig, 3.21. Game class envelops the virtual arbiter functionalities namely it handles the score and tracks overall game progression. The WebScocketClient is responsible for maintaining stable connection with the server and also maintains the log for all the message traffic that was exchange by the client. Finally the Graphics class handles the 3D simulation and its configuration like placement of microphones or table size. The simulation itself is based on a WhiteStorm JS engine. WhiteStorm is a powerful framework built on top of THREE.js graphics library. It encapsulates bare WebGL functionalities and greatly speeds up 3d scene building and animation rendering. The 3d visualization used in the application is simple with only table and microphone models being rendered. Additionally when event is detected it is marked by spawning the fading sphere indicating the place where the ball was determined to be by the worker script.

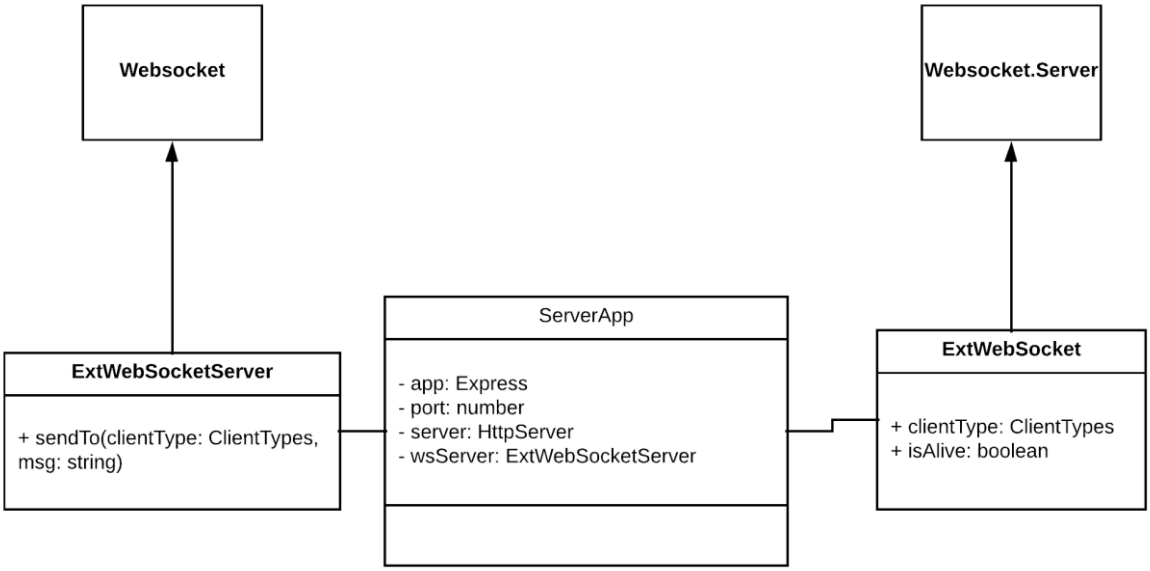


Fig, 3.21 Class diagram of the GUI displayed in browser

3.4.2 Communication Protocol

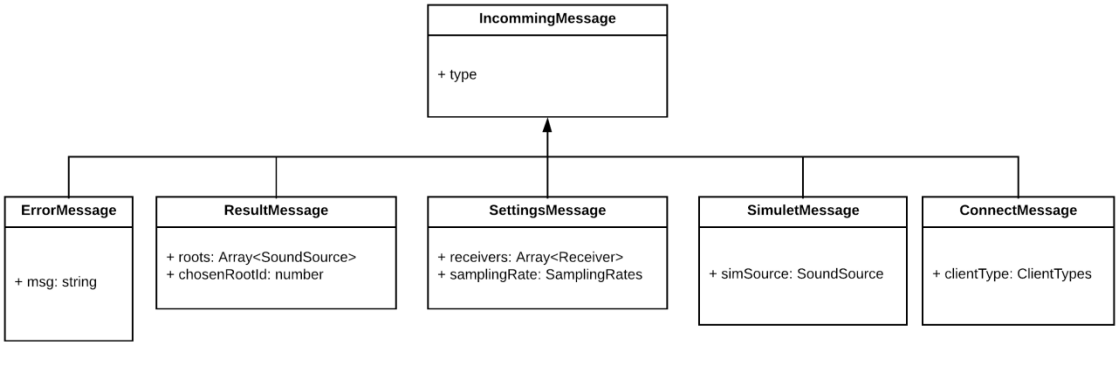
As mentioned previously the user interface is realized as client-server type application. The server serves a role of a middleman. It is connected with separate web socket connections to

both GUI and localizer application. Furthermore, express JS http server is used to dynamically render html templates and to render the webpage to a client on GET request. This type of solution is very versatile as it enables many worker scripts and many user interfaces that can operate independently from each other. Also many user interfaces can be connected to a single worker script. Additionally server can be hosted on arbitrary machine with Linux operating system.

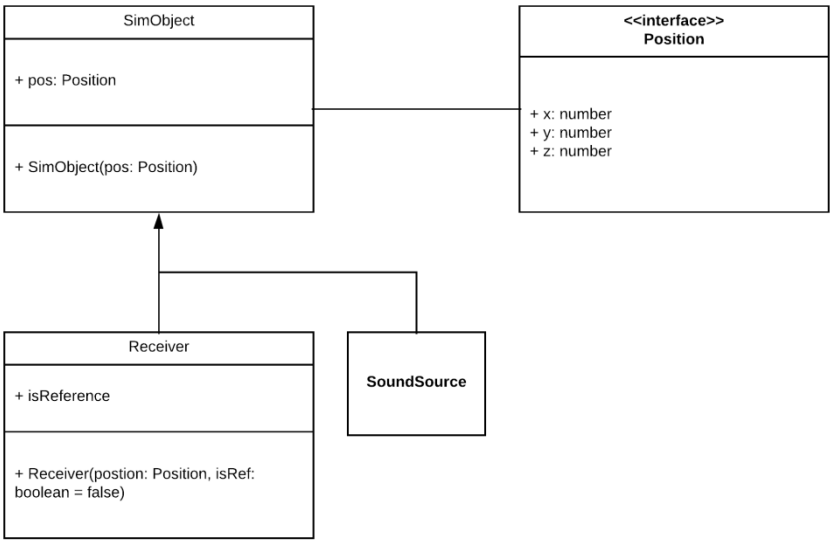


Fig, 3.22 Server class diagram

Web socket connections are responsible for entire data transmission between the applications within the project. The data are transmitted in special message objects serialized to JSON format. For the purpose of the communication there are several types of messages, which describe different parts of configuration and result data exchanges between the applications. These types include result message obtained from localization algorithm, settings regarding placement of microphones or error messages indicating failures within each particular application. Furthermore, upon connection initialization each client has to send special connect type message in which it describes itself as either a worker performing or GUI client displaying the interface. The class diagram of different messages and how the different objects are serialized to class format is presented below.



Fig, 3.23Class diagram for message structure used for communication

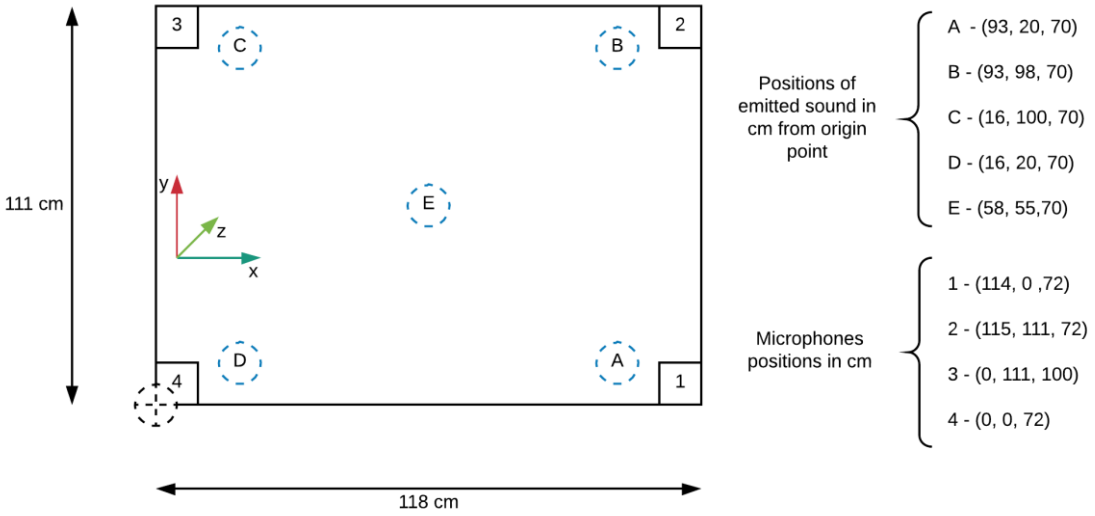


Fig, 3.24 Class diagram representing simulation object used within the GUI client

4 Solution's performance

In order to assess the viability of the proposed system the tests were conducted with the main stress put on the microphone array and the software ability to present correct locations of the emitted sound. Without accurate positions of the ball, the whole idea of tracking the table tennis game would be impossible to realize. The test setup is presented in Fig, 4.1. The microphones were placed in the corners of a table with the dimensions of 111 x 118 cm and the height of 70 cm. Due to technical limitations, the table was not as big as table tennis one, however it was big enough to record sufficiently big time delays on the microphones in the array. Three out of four microphones were placed in the same 2D plane; where microphone

number 3 was elevated by about 30 cm to ensure that the inverse of matrix C (described in section 3.3.2) exists and to meet the basic requirements of 3D localization sensor setup. Exact locations are visible in the diagram. The beginning of the coordinate system is located below bottom left corner of the table on the floor below it. The sound was being emitted from 5 distinct places on the table surface. These are marked with letter annotations on the diagram.



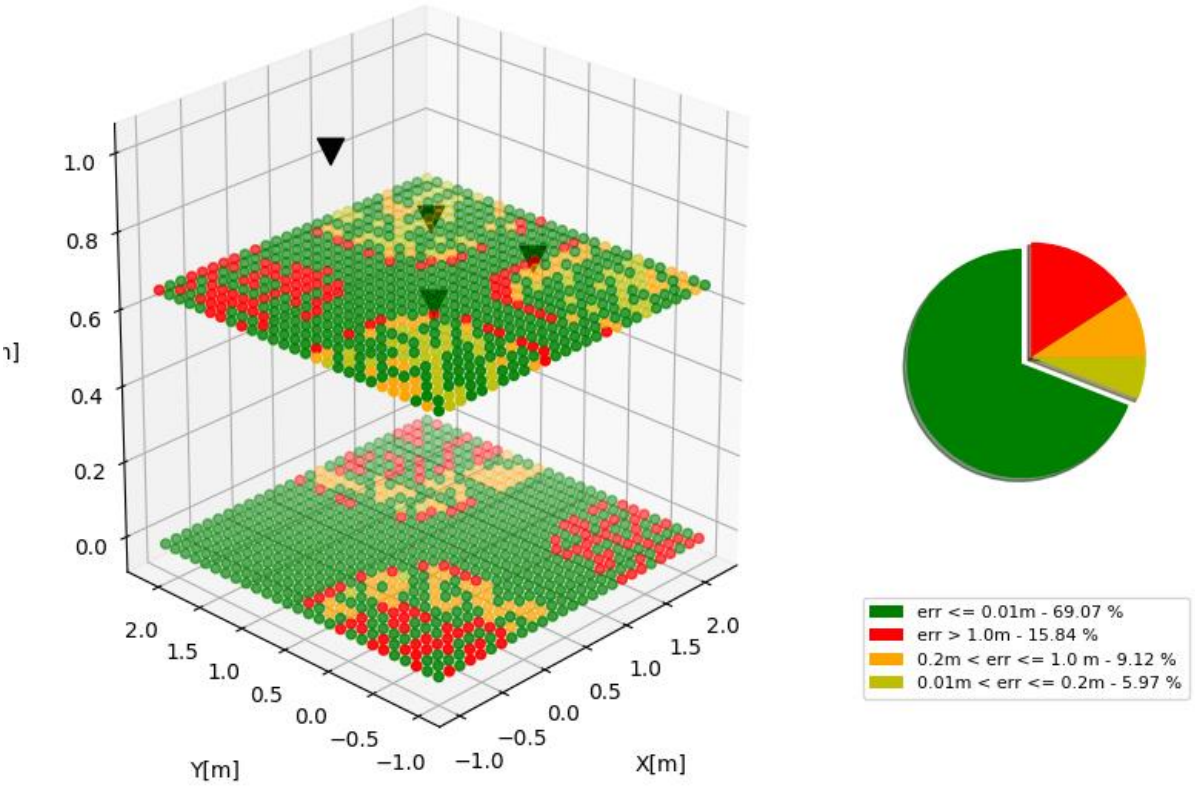
Fig, 4.1 Test setup, top view of the game table

4.1 Simulation

Before the real world test, firstly the microphone placement was simulated virtually to check how MLE-HLS algorithm is going to operate with limited accuracy of time variables. Furthermore different placements of the receivers were analysed for a game table. These positions resemble typical receiver layouts used commonly in sensor arrays, namely square trapeze and cone. The slight differences in these layouts come from the fact that any receiver cannot obstruct game that is going to track either by its placement or the way it is fixed in place.

Simulation is performed in such a way that sound source is placed in a different positions inside cuboid with a specified step in all directions. To better represent the game environment and to make graphs more readable only two planes are simulated: one which represents the table itself

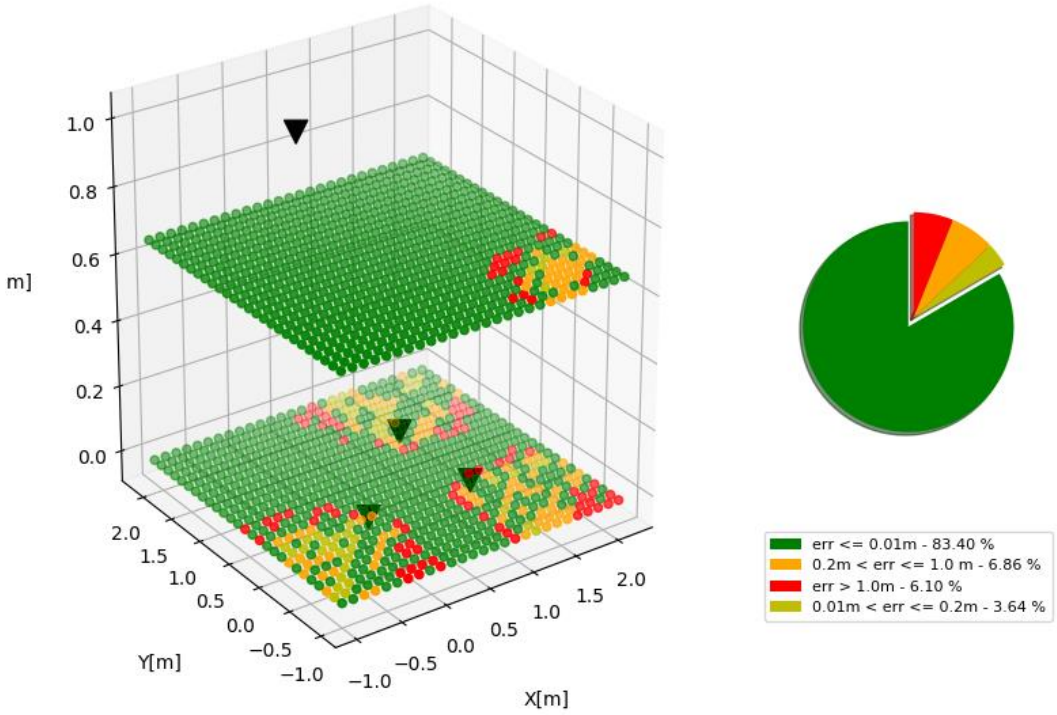
and one for the floor below it. Apart from standard table dimensions specified in Fig, 4.1, additional margin of 1m in x and y dimension is simulated. Whenever a new position of the sound source is assigned the sound is emitted at time t=0, time of flight is calculated based on spatial position of a given receiver and speed of sound constant. With known time of flight (ToF), the time difference of the arrival is computed in form of a difference in ToF between any microphone pair. The obtained result is rounded to the 5th decimal digit simulating limited accuracy of the measurement. That data along with microphone positions is then fed to the MLE-HLS algorithm implementation and the source position is calculated back by the algorithm. Then, knowing exact position of the emitted sound the accuracy is derived from Euclidean metric and result is qualified into four categories marked by a different colour. The dot on a 3D graph represents a single source position and its colour indicates how accurately the result was computed by the MLE-HLS algorithm.



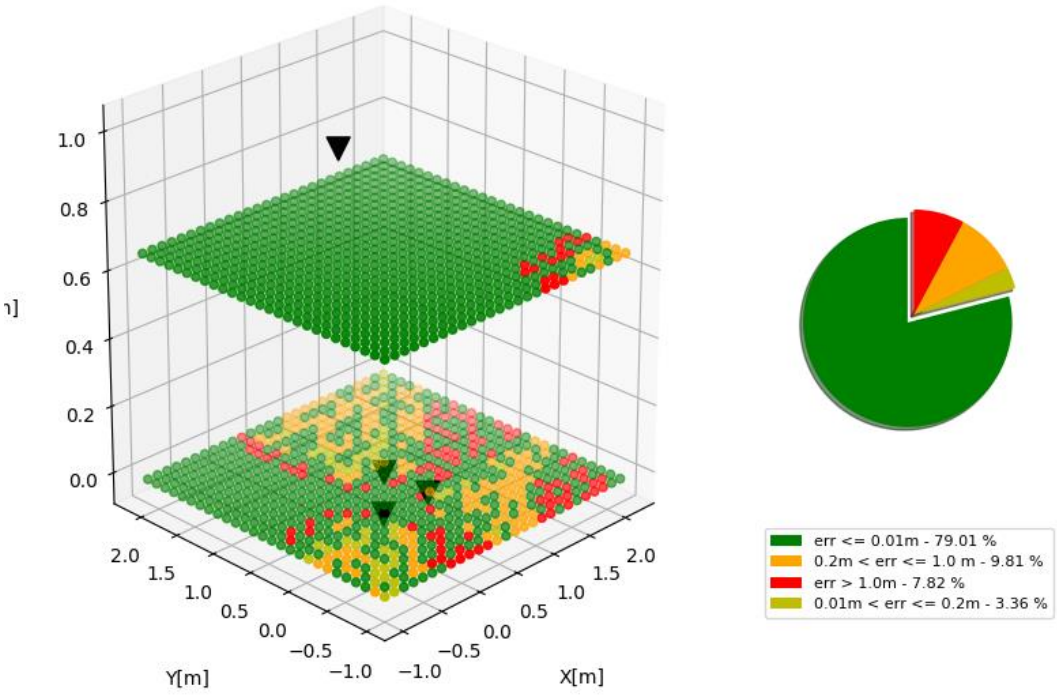
Fig, 4.2 Microphones in a square layout. Microphones are marked with black triangles M_1(0, 0, 0.72), M_2(0, 1.15, 1.0), M_3(1.18, 1.15, 0.72), M_4(1.18,0,0.72)

First layout was the square one, which is the one that was used in the real world application. Its usage was arbitrary as it was utilizing simply all edges of the table and was most convenient in terms of microphone placement, resulting in relatively small errors in their measured positions. However as it can be seen in the Fig, 4.2, the results yield over 15% of completely wrong source positions, while only about 69% is with the desired accuracy of 1cm and below. Despite the high number of invalid source position estimations, one has to note that most of them occur outside the area of the game table. Therefore the localization can still be performed relatively accurately on the table surface and directly below it.

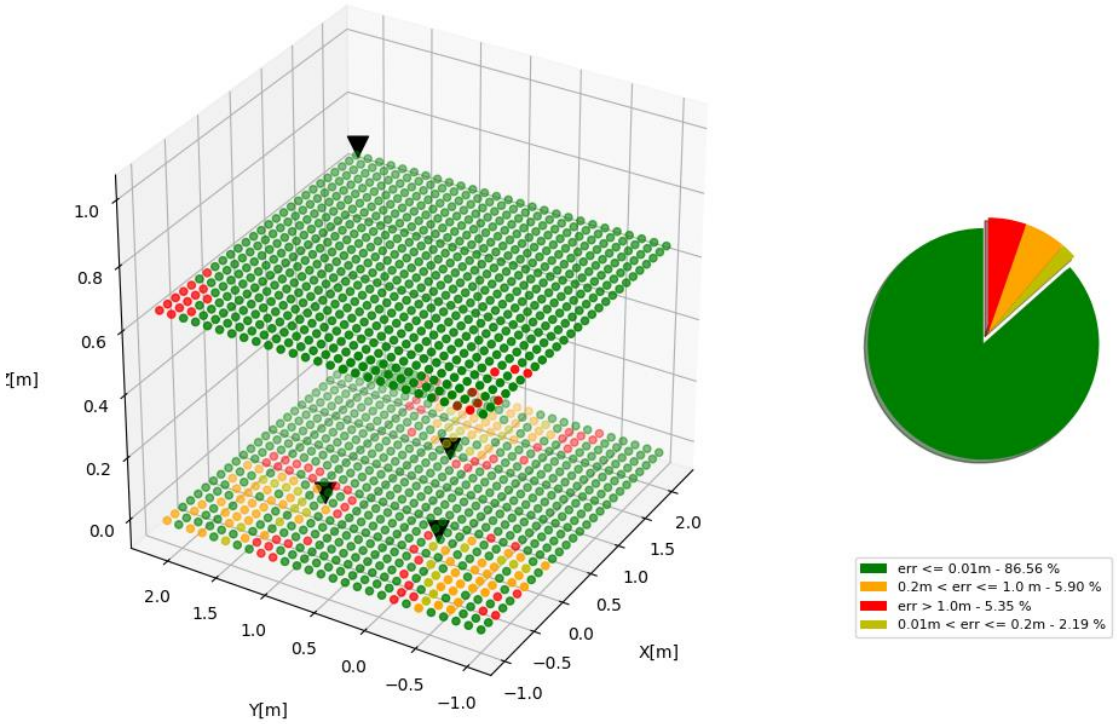
The main source of error is caused by the HLS part of the algorithm, which with limited precision of TDoA estimations, introduced by rounding, is not capable of choosing the correct solution returned by the MLE algorithm. When all of the roots returned by the MLE are taken into account all simulated source positions are classified with the best accuracy criterion. Nevertheless, the layout of the microphones can still be improved. Simply by placing the three microphones below the table, on the floor, can improve overall accuracy by a large margin. This can be seen in the Fig, 4.3. Ideally, if the three microphones located in a single plane were moved about 1m below the floor surface, the number of invalid estimations would be eliminated nearly entirely. This is however impossible to achieve in a real world application.



Fig, 4.3 Microphones are marked with black triangles M_1(0, 0, 0), M_2(0, 1.15, 1), M_3(1.18, 1.15, 0), M_4(1.18,0,0)



Fig, 4.4 Microphones in trapezoidal setup. Microphones are marked with black triangles M_1(0, 0, 0), M_2(0, 1.15, 0), M_3(0.59, 1.15, 0), M_4(1.18, 0, 1)



Fig, 4.5 Microphones in cone position. Microphones are marked with black triangles M_1(0, 0, 0), M_2(0, 1.15, 0), M_3(1.18, 0.575, 0), M_4(0.59, 1.15, 1)

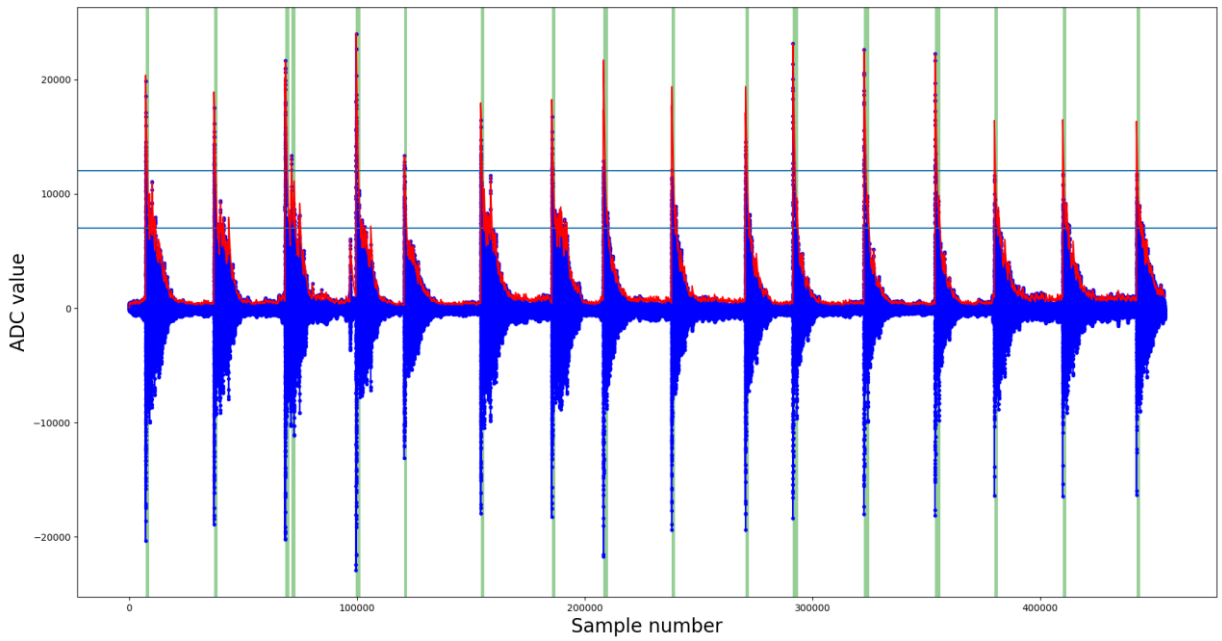
In similar fashion the cone and trapeze layout were tested. As with the square layout the accuracy of MLE-HLS calculations dropped whenever the microphones were close to the table. With the limitations of a table tennis game, the cone appeared as the best layout. Here the three microphones are placed on the floor surface, where two are underneath the table's corners and one is underneath the middle of the opposing table's side. The fourth one is elevated about 30cm above the table's surface and has x, y coordinates of (59cm, 111cm), meaning that it is located near the middle of the table close to one of its borders.

Based on the simulations, the main source of error regardless of the microphone layout was contributed to the choice of the root returned by the MLE algorithm. This is gradually more visible with the decrement in the precision of TDoA estimations. To counter this potential problem in real world test, additional considerations are taken into account when assessing the roots. Since the problem of localizing the tennis table ball is not a generic one, one can determine the domain which is acceptable for obtained sound source position. Here one cannot expect the sound of the bouncing ball from below the floor surface or too high above the table

surface, therefore invalidating roots with such coordinates in Z axis. In final version the roots obtained from MLE algorithm are validated firstly in terms of domain, and if both are within the domain, the HLS algorithm decides which one is the best.

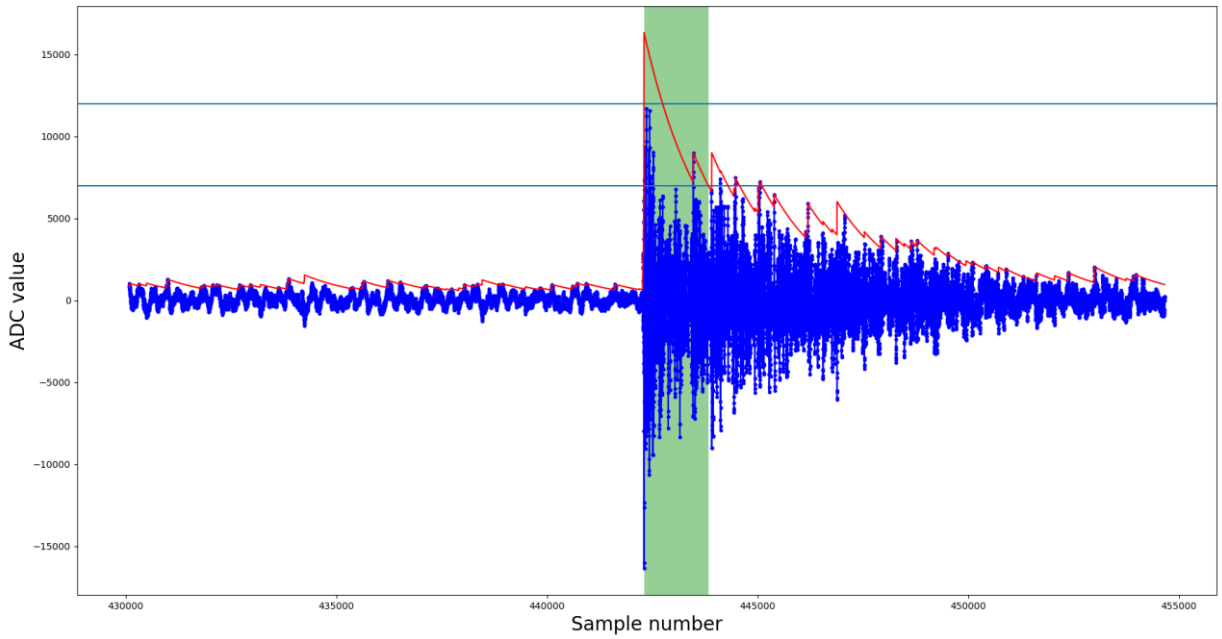
4.2 Real world test

The test was conducted in a setup depicted in Fig, 4.1. To eliminate variations in the subsequent positions in location of the sound and to test the consecutiveness of measurements, the bouncing ball sound was recorded and played from a speaker facing upwards from the table surface. With real ball it is hard to control exact landing spots with the precision up to 1cm, therefore such solution was proposed, to better determine accuracy of the design. The sound was played repeatedly from given spot at least three times before changing the speaker position to a next one. Finally to analyse the algorithm in detail, sound was recorded to 4-channel wav file during real time analysis on the Raspberry PI device. By having test session recorded it is possible to read the data from the file and recreate and inspect entire localization process in details.



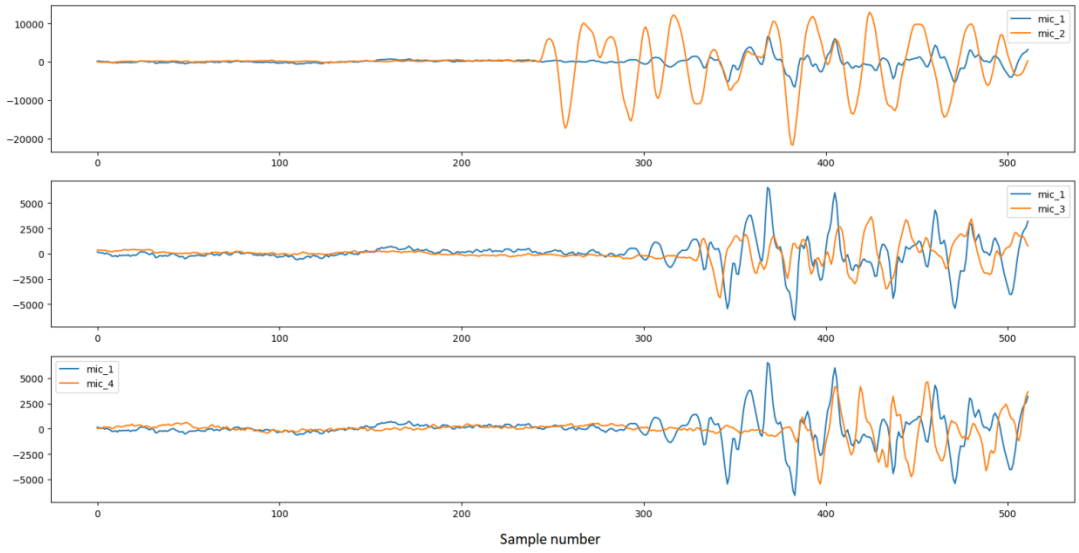
Fig, 4.6 Test session record, with sound detected as bounce by algorithm marked as green, signal envelope as red

The Fig, 4.6 presents the amplitude graph of the chosen fragments from the recorded session. It contains the audio used for the calculations of an envelope, meaning at a given time the channel with the strongest energy per data chunk was chosen. Therefore the graph does not represent only one channel in particular but it is a mix of all the ones with the strongest amplitude at a given time. The red outline of the signal marks the envelope calculated to estimate where the sound exceeds amplitude thresholds. These thresholds are marked with two horizontal lines. When upper one is exceeded, the start of the sound event is registered and when the amplitude drops below lower one, it is considered to be the end of it. The detected sound events are marked with green background. Detailed picture with only one bouncing ball sound sample is visible in Fig, 4.7. The envelope function provides relatively smooth outline of the signal as expected, which then is checked against the thresholds. The sound events are then correctly detected and narrowed to a small portion of the audio data. Only in one case, the sound event was detected falsely, when in reality no such event occurred in this time frame. Its false detection can be contributed to the reverberations of the sound present in the test's environment as the room was an office room with relatively small surface. However, one can fine tune the threshold values even further to reduce the possibility false detections by increasing the difference between lower and upper threshold.



Fig, 4.7 Single bounce event, zoomed

Another important aspect to test is to inspect if the delays exist between the four ADC channels and to ensure that they are captured properly. The Fig, 4.8 presents channels 2 to 4 in comparison to channel 1, against which the TDoA was calculated. Only the small neighborhood around event's start point is taken into account. In total 512 samples from each channel are used for general cross correlation, with the center being event's start point. As seen in the figure, the delays between the channels are registered properly. The signals have the same shape and are shifted in time. There is however difference in amplitude, which is to be expected as the sound emitted closer to a certain microphone will also appear with larger amplitude on that channel. The overall quality of the sound is high with little noise present; therefore the sound is clearly distinguishable. Due to that fact it is always possible to tell which signal was recorded first even without performing exact calculations with the general cross correlation.



Fig, 4.8 ADC signal channels fragment used in TDoA calculations

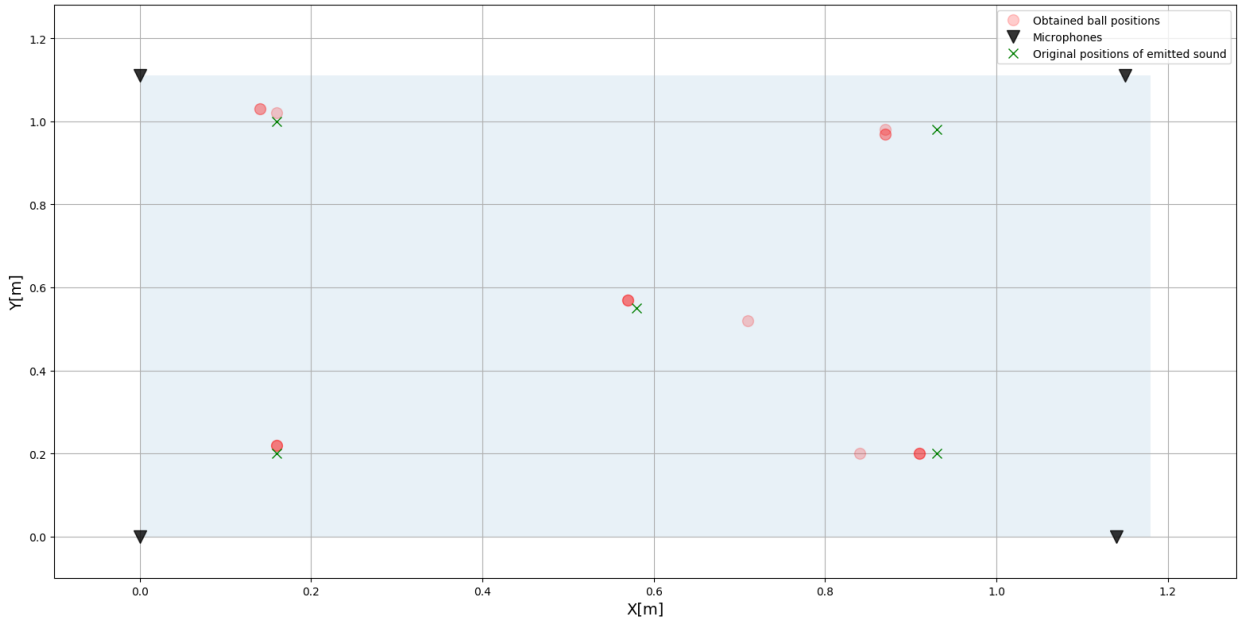
Position	Sample nr	X[m]	Y[m]	Z[m]
A	1	0.91 ± 0.03	0.20 ± 0.03	0.76 ± 0.11
	2	0.91 ± 0.03	0.20 ± 0.03	0.76 ± 0.11
	3	0.71 ± 0.02	0.52 ± 0.01	1.58 ± 0.01
	4	0.91 ± 0.03	0.20 ± 0.03	0.75 ± 0.10
	5	0.84 ± 0.04	0.20 ± 0.05	0.98 ± 0.14
B	1	0.87 ± 0.02	0.97 ± 0.04	0.75 ± 0.13
	2	0.87 ± 0.02	0.97 ± 0.05	0.75 ± 0.15
	3	0.87 ± 0.02	0.98 ± 0.06	0.71 ± 0.17
C	1	0.14 ± 0.02	1.03 ± 0.03	0.69 ± 0.07
	2	0.16 ± 0.02	1.02 ± 0.03	0.72 ± 0.07
	3	0.14 ± 0.02	1.03 ± 0.03	0.69 ± 0.07
D	1	0.16 ± 0.03	0.22 ± 0.01	0.58 ± 0.13
	2	0.16 ± 0.03	0.22 ± 0.01	0.58 ± 0.13
	3	0.16 ± 0.03	0.22 ± 0.01	0.58 ± 0.13
E	1	0.57 ± 0.01	0.57 ± 0.02	0.68 ± 0.08
	2	0.57 ± 0.01	0.57 ± 0.02	0.65 ± 0.08
	3	0.57 ± 0.01	0.57 ± 0.02	0.68 ± 0.08

Table 1 Obtained source coordinates for a real world test

The overall results are present in Table 1. Position column represent the different positions, depicted in the Fig, 4.1. Sample number is the name for the index of an event registered in a certain position. The most of the obtained sound source coordinates is consecutive within the same position in terms of x, y coordinates. Fluctuations in x, y plane are small and rarely exceed 1cm in any dimension. On the other hand, results z coordinate, representing height, tend to have significantly less accuracy and differ from one sample to another within the same position. Despite that fact, the error ranging up to 13cm is still acceptable, as the main interest for this project is localization of a ball on the plane surface. The z coordinate is only viable to distinguish whether the ball landed on the floor or on the table.

The locations returned by the algorithm are also close to the one measured during test setup as most of them are calculated within the error range. The only truly invalid estimation of the

sound source location is 3rd sample of the position A, where the TDoA was calculated poorly by the GCC. This result is however achieved after introducing domain restrictions in terms of z axis. Without them the HLS algorithm tends to choose invalid root as seen in the Fig, 4.2 for this microphone layout.



Fig, 4.9 Graphical representation of the obtained results from Table 1. Surface of the table is marked with blue rectangle

The error analysis was performed individually for each calculated result based on general uncertainty propagation during mathematical calculations. It was possible due to the additional Python library, which keeps track of the measurement uncertainty. For a microphone's coordinates the measurement error is assumed to be 0.5 cm in any axis. Furthermore the TDoA estimations done by GCC are limited by a sampling rate, which equals to 41.667 kHz. Smallest shift of one sample between the two signals results in a TDoA uncertainty of 24µs. This value was used to compute error analysis in the final results. The applied sampling rate, however is not ideal as according Fig, 2.7, optimal rate should be grater or equal to 100 kHz. However due to real time processing and hardware limitations this kind of rate could not be achieved in this project, as such amount of data could not be reliably transferred and processed in real time.

When 100 kHz is used, the TDoA error is only 10us which in turn minimizes the overall measurements error to 1cm and below. Apart from changing the sampling rate, placing microphones with higher precision also has a decent impact on the overall accuracy.

Furthermore samples can be interpolated to reduce TDoA uncertainty without physically increasing the sampling rate.

5 Conclusions

During the course of the thesis various aspects of acoustic localization and signal processing were discussed. The main objective of the thesis was to propose a system able to localize a table tennis ball as it bounces on the surface of a table and provide the prototype of a user interface able to display locations, where the ball impacts. The main target user of this application was a visually impaired tennis player playing a special variant of the tennis table game.

This was a demanding task, which required strong interaction with a real word. Therefore solution proposed in this thesis comprises of both hardware and software design. In terms of hardware it meant constructing custom equipment able to record and stream audio data in real time from four receivers at once. The software side consisted of the localization module, which processed the signal data and outputted coordinated estimations for the ball. Additionally, the prototype of the user interface was designed to visualize located ball positions in virtual 3D space and to test if it is possible to track the table tennis game progression.

The proposed system operates correctly in terms of sound localization and ball positioning. All hardware components and software applications included in the project communicate with each other. They transfer the data correctly between themselves and to the user interface, where the computed ball location is marked in a 3D simulation.

Although the results from the tests are promising, in order for the design to become a potential product, some considerations and improvements have to be taken into account. The user interface needs to be expanded in order to track the game fully and it has to be adapted for the requirements of visually impaired persons. In the state presented in the project it is only a prototype, which communicates with the calculation module and provides mostly a visual form of interaction. In terms of hardware, the microphones register sound properly and the resulting electrical signals are converted to the digital domain with negligible noise. However, the PCB

board requires further development as it cannot communicate directly with the Raspberry PI. Therefore a new design should be made in order to accommodate an additional microprocessor unit, which handles the SPI communication between the board and the USB buffer. Additionally, to improve sound quality and to diminish impact of noise one can utilize the digital filter present in the ADC module to only capture the specific frequency band necessary for audio recognition. In this project this was not utilized due to the limited time scope and since configuring such a filter required a faster clock speed than the one provided on the PCB board. Finally, the sampling rate used to sample audio signals coming from the microphones could be much higher. The limited sampling rate used in the project was due to transmission issues. Currently about 41 kHz is used, which is suitable for normal audible signal processing; however, increasing it to a value above 100 kHz would greatly benefit the overall accuracy of the acoustic localization algorithm. It has direct impact on the possible resolution of time delays between microphones and thus contributes significantly to the overall error.

Due to the broad scope of thesis, which tackles such vast topics as 3D localization, sound recognition, hardware design and construction, it was impossible to realize all the aspects of the stated goal within a single paper. In conclusion the presented approach utilizing sound localization can be used to track the game of table tennis for visually impaired, when considering the previously mentioned limitations. Despite that it may not be ideal to achieve this task. With current development of software ready to use tools for visual processing such as OpenCV it would be potentially less demanding in terms of software and hardware to develop a system that tracks the ball movement with a camera from a position above the game table. In this kind of scenario clues derived from the recorded audio would perform a supplementary role, by hinting if the ball is actually bouncing off the table or to interact with the user itself via simple speech commands.

6 Bibliography

1. **Szmaj L., Białek A.** *Tenis stołowy dźwiękowy – przepisy gry.* Październik 2011.

2. **Strumiłło, Paweł.** *Advances in Sound Localization.* Rijeka, Croatia : InTech, 2011.

3. **Jacek P. Dmochowski, Jacob Benesty, Sofiène Affes.** A Generalized Steered Response Power Method for. [Online] November 2007. http://externe.emt.inrs.ca/users/benesty/papers/aslp_nov2007.pdf.

4. **H. Wang, P.Chu.** Voice source localization for automatic camera pointing system in videoconferencing. 1997. pages 187-190.

5. A new location system for an underground mining environment using visible light communications - Scientific Figure on ResearchGate. [Online] https://www.researchgate.net/Trilateration-technique_fig3_267324735.

6. **José Manuel Fresno, Guillermo Robles, Juan Manuel Martínez-Tarifa.** *Survey on the Performance of Source Localization Algorithms.* Glasgow, UK : University of Strathclyde, 2017.

7. **Wong, Cheuk.** hypertextbook. *Speed Of Sound In Air.* [Online] 2000. https://hypertextbook.com/facts/2000/CheukWong.shtml.

8. **C Sandeep Reddy, Rishika Agarwal, Lavisha Aggarwal.** *Binaural source localization using a HRTF data model with enhanced frequency diversity.*

9. **LaValle, Steven M.** Monaural cues. *Virtual Reality.* [Online] [Cited: 01 09 2018.] http://vr.cs.uiuc.edu/node361.html.

10. **Kjellson, Angelica.** *Sound Source Localization and Beamforming for Teleconferencing Solutions.* Umeå, Sweden : Department of Mathematics and Mathematical Statistics Umeå University, 2014.

11. **Miao, P., et al.** *Location algorithm for partial discharge based on radio frequency (RF) antenna array.* Shanghai, China : In Proceedings of the 2012 Asia-Pacific Power and Energy Engineering Conference, 2012.

12. **Yang, M. and Chen, K.H.** *Performance assessment of a noniterative algorithm for Global Positioning System (GPS) absolute positioning.* s.l. : Proc. Natl. Sci. Counc. ROC(A),, 2001.

13. **El Mountassir, O., et al.** *Quantification of the performance of iterative and non-iterative computational methods of locating partial discharges using RF measurement techniques.* 2017. page 110 -120.

14. **Stewart, B.G., Nesbitt, A. and Hall, L.** *Triangulation and 3D location estimation of RFI and Partial Discharge sources within a 400 kV substation.* Montreal, Canada : In Proceedings of the 2009 IEEE Electrical Insulation Conference, 2009. page 164 - 168.
15. **Nicholas Jillings, Alice Clifford, Joshua D. Reiss.** *Performance optimization of GCC-PHAT for delay and polarity correction under real world conditions.* Rome, Italy : 134th Audio Engineering Society Convention, 2013.
16. **Reiss, A. Clifford and J.** *Calculating time delays of multiple active sources in live sound.* San Francisco, USA : 129th Audio Engineering Society Convention, 2010.
17. **Yue Kan, Pengfei Wang, Fusheng Zha, Mantian Li, Wa Gao and Baoyu Song.** *Passive Acoustic Source Localization at a Low Sampling Rate.* Harbin, China : Harbin Institute of Technology, 2015.
18. **Boller, Justin.** What's the Difference Between Line and Mic Levels? [Online] Shure. http://blog.shure.com/whats-the-difference-between-line-and-mic-levels/.
19. **Raspberry Pi foundation.** Products. [Online] [Cited: 04 September 2018.] https://www.raspberrypi.org/products/.
20. —. Raspberry Pi 3 Model B+. [Online] [Cited: 4 September 2018.] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/.
21. **Maxim Integrated.** APPLICATION NOTE 290 ADCs for Simultaneous Sampling. [Online] https://www.maximintegrated.com/en/app-notes/index.mvp/id/290.
22. **Analog Devices.** Understanding and Designing Differential Filters for Communications Systems. [Online] http://www.analog.com/en/technical-articles/understanding-and-designing-differential-filters-for-communications-systems.html.
23. **Sanjay Pithadia, Shridhar More.** Grounding in mixed-signal systems demystified. [Online] Texas Instruments. http://www.ti.com/lit/an/slyt499/slyt499.pdf.
24. **Raspberry Pi.** Mechanical Drawings. [Online] https://www.raspberrypi.org/documentation/hardware/raspberrypi/mechanical/.

7 List of figures

8 Appendix A. Overview of CD's content

The attached CD disc includes additional material in form of the digital copy of project's github repository, which contains:

- Code for the Teensy 3.2 embedded development board, which handles the ADC initialization and reads its result registers. This can be found in the folder "driver_c"
- Implementation of the MLE-HLS algorithm in Python along with sound recognition and data transfer algorithms. The source code for the application is in folder "localizator"
- Graphical user interface source code with simple HTTP and web socket server, written in Typescript; attached in the "webserver" folder.

Moreover apart from the code itself, the CD contains the design files for Altium PCB designer program. This project contains schematics and PCB layouts for the ADC and microphone boards designed during the course of the thesis. These are located in folder "docs/hardware_design". Finally, the selected important source material is included in form of pdf files located in folder "docs/reference_sources". The sources gathered there include positions such as [2], [6], [17], [15]. Folder also contains subdirectory with the datasheets for MAX11043 ADC, Loudity LD-MC-6035P microphone and LT1994 differential amplifier.

.....
(IMIĘ I NAZWISKO STUDENTA)

.....
(NR ALBUMU)

.....
(KIERUNEK STUDIÓW)

.....
(RODZAJ I FORMA STUDIÓW)

OŚWIADCZENIE

Świadomy/a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że przedkładana praca magisterska / inżynierska / licencjacka(*) na temat:
.....
.....
.....
.....
została napisana przeze mnie samodzielnie.

Jednocześnie oświadczam, że ww. praca:

- nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2000 r. Nr 80, poz. 904, z późniejszymi zmianami) oraz dóbr osobistych chronionych prawem cywilnym, a także nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
- nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.

Jestem także świadomy/a, że praca zawiera rezultaty stanowiące własność intelektualną Politechniki Łódzkiej, które nie mogą być udostępniane innym osobom i instytucjom bez zgody Uczelni.

Oświadczam również, że dołączona na nośniku elektronicznym kopia pracy jest w pełni zgodna z przedstawionym do recenzji wydrukiem.

.....
(PODPIS STUDENTA)

(*) - niepotrzebne skreślić