Politechnika Warszawska

WYDZIAŁ ELEKTRYCZNY

Institute of Theoretical Electrical Engineering and Information and Measurement Systems

Department of Information and Measurement Systems

Engineering thesis

on the faculty of Applied Automation and Robotics

Localization of sound sources using microphones

Arkadiusz Strzała

Album number 307451

thesis supervisor

dr hab. inż. Marcin Kołodziej

WARSZAWA 2023

Localization of sound sources using microphones

Abstract

Localizing the direction of arrival of sound is an issue that is crucial in many areas of modern technology. It finds application in the military where it is the basis for the operation of such technologies as airborne radar and sonar used by submarines. The purpose of the following work is to build a device capable of detecting the direction of sound using an array of microphones by taking advantage of the time difference after which the signal reaches them. An array composed of MEMS microphones will be used for this, along with an STM32F103RBT6 Microcontroller used to sample the data from the microphones.

Keywords: direction of arrival, microphone array, STM32F103RBT6, MEMS microphone

Warszawa, dnia 26.01.2023 r.

Table of contents

Chapter 1

1. Introduction

Localizing the direction of sound is a key issue in many fields of technology. As early as World War II, the first applications of this technology were seen in the design of radars, which allowed the location of enemy aircraft to be detected by analyzing signals reflected from them. Applications also found their way into sonar, which is an integral part of modern boats, especially those designed for military purposes, where the ability to detect the location of enemy ships is crucial. In this paper I will focus only on estimating the direction of sonar signals, but by the same methodology it is possible to locate the direction of incident radio waves. This technology is used in radio telescopes to observe the position of celestial bodies in space. It is also indispensable in the daily life of each of us, because it makes possible the operation of a key localization system - GPS, which, thanks to signals from satellites orbiting the Earth, can determine our location anywhere on Earth with an accuracy of a few meters.

The present work is divided into 9 chapters.

Apart from the introductory paragraph, the content of the other chapters is as follows:

- Chapter 2 discusses the purpose of the work and its scope.
- Chapter 3 contains a comparison of different methods of solving a given problem and an explanation of why I chose my method.
- The 4th chapter is a theoretical introduction to the issue. It explains in general how it is possible to solve a given problem.
- The 5th chapter describes the hardware part of the project. It explains for what reason I chose the different circuits and how I connected them together.
- Chapter 6 describes the tools used and the implementation of the microcontroller software used in the project.

- The content of Chapter 7 describes the choice of the programming language used and the analysis from the mathematical and programming side of the written program on the PC side.
- In Chapter 8 I present the results from the tests performed and their analysis.
- Chapter 9 is a summary of what I was able to achieve, what came out not quite as I expected, and ideas on how the project could be improved.

Chapter 2

2. purpose and scope of work

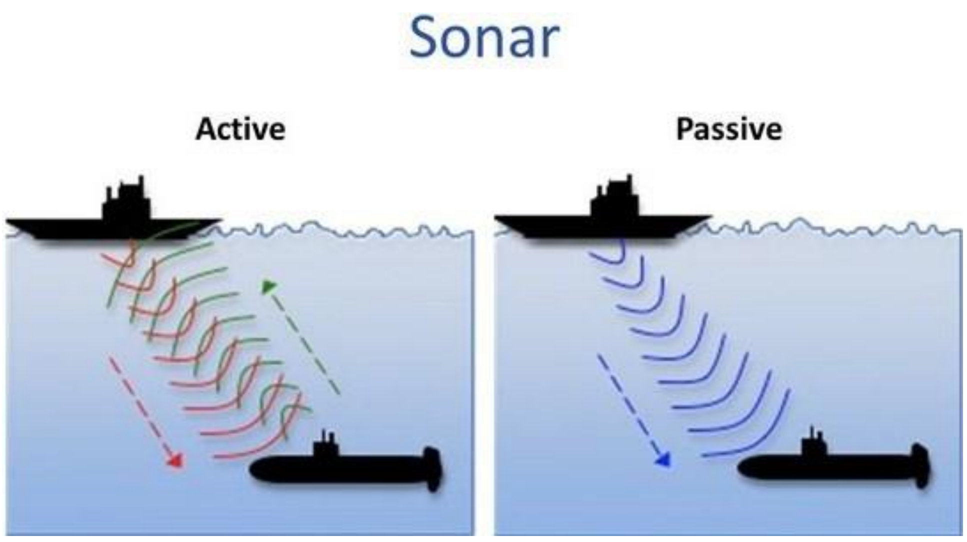
The purpose of this work is to build a device that is capable of detecting the direction of sound with the highest amplitude. As part of the work, I will carry out the design of a measurement system that will be used to continuously monitor ambient audio signals. This system will be able to transmit measurement data to a computer in real time. On the computer side, an algorithm will be implemented that will calculate the direction of sound based on the audio signals from the microphones. The system will be able to display the results of the calculation in real time in graphical form.

Chapter 3

3. Sound source detection methods

Detecting the source of a sound involves using an array of sensors tuned to receive a specific signal. Then, by analyzing the data from each sensor, the time delay between the arrival of a wave to each sensor in the array must be determined. With access to all the time delays and the exact positions of the sensors relative to each other, it is possible to geometrically calculate the angle of incidence of the signal. This approach makes it possible to determine the source of a sound, by analyzing data from many different sensors [*2*].

To get a better understanding of where this technology is used in practice, I'll zoom in on how sonar works in submarines. There are two types of sonar: active and passive [Picture 1]. Active sonar sends out sound and then listens for how long and from what direction the signal returns to the boat. This solution makes it possible to determine quite precisely the distance of objects from the boat. The downside, however, is that by sending the sound, the boat reveals its position, which is a major problem in military applications. To eliminate this inconvenience, modern submarines often use passive sonar. Its operation is based on listening to sounds, from the environment with the help of a hydrophone array. As a result, it is able to detect the direction of sound, but unfortunately it is not able to determine the distance from which the sound is coming. The purpose of this paper is to try to map the basic functionality of this sonar in a land environment using microphones.



Picture 1. Comparison of passive and active sonar performance [10*]*

Chapter 4

4. Comparison of used solutions

Detecting the difference in arrival time of signals between sensors is a key and complicated aspect of this issue. To achieve this, we use information about the phase shift of the sound of interest. We use the Fast Fourier Transform (FFT) algorithm to do this, which allows us to transform a signal from the time domain to the frequency domain and enables spectral analysis of the signal, which allows us to calculate its phase shift angle. In my project, I only use this information to calculate the time difference of the signals. This is a simple implementation method that does not require a lot of computing power, so the device can work in real time, even on a computer with lower performance.

However, there are more complex algorithms that use spectral analysis of the signal to calculate the direction of the sound. The most commonly used algorithms are ESPRIT (Estimation of Signal Parameters via Rotational Invariance Techniques), MUSIC (MUltiple SIgnal Classification) and SRP-PHAT (Steered Response Power Phase Transform). I will outline the advantages and disadvantages of these solutions below.

Esprit relies on the use of the properties of the rotational correlation matrix, which allow you to determine the angles of incidence of a sound source without having to estimate the amplitude of the signal. The algorithm uses two or more microphones in different positions to examine the signal from different perspectives.

It has the advantages of being effective for multiple sound sources and hard-to-distinguish signals. It is also simple to implement. The disadvantages, on the other hand, are that it is not very resistant to noise and does not handle detection of sound sources that are close together.

It is a good choice when there are few sound sources and well-distinguishable signals. The final accuracy of the result depends on the number of microphones and their placement [3].

MUSIC is a phase positioning method that estimates the angle of incidence of a sound source by analyzing the eigenvectors of the correlation matrix of signals from different microphones.
It performs well for multiple sound sources and hard-to-distinguish signals.
It is much less robust to noise than other phase positioning methods, and requires good microphone placement and an appropriate number of microphones for the size and shape of the room.
It is a good choice when there are many sound sources and hard-to-distinguish signals. The final accuracy of the result depends on the number of microphones and their placement, as well as the acoustic conditions. This is one of the most popular algorithms for detecting the angle of incidence of a sound source [*4*].

SRP-PHAT is a derivative of the SRP (Steered Response Power) algorithm and relies on using the phase of the microphones' signal to determine the direction of the sound source. The algorithm uses two or more microphones in different positions to examine the signal from different perspectives.
It features noise immunity, is able to determine the angle of incidence of a sound source even when there are multiple sound sources, and simplicity of implementation.
However, it has difficulty locating sound sources close to each other, and can give uncertain results when the sound source is very close to the microphones.
It is a good choice when there is a lot of noise, and when there are many sound sources. The final accuracy of the result depends on the number of microphones and their placement, as well as the acoustic conditions [*5*].

Chapter 5

5. Device design

The simplest way to solve this problem would be to purchase a sound card with three audio inputs and jack output microphones. Unfortunately, such a solution is quite expensive, as these cards can cost up to several thousand zlotys. For this reason, I decided to use a cheaper solution. I decided to use a microcontroller, which I will use to take measurements from the microphones using an analog-to-digital converter. This is a more economical solution that achieves the same goal without having to spend a lot of money on specialized equipment.

5.1. Microphones

In order for the device to work properly and with the greatest possible accuracy, the microphones must meet certain requirements. From a practical point of view, it is important that the power supply voltage of the microphone does not exceed 5V, so that I can avoid the need for an additional voltage booster converter and use the voltage straight from the USB port. Microphones should be small in size so that they can be placed close together. In order to measure the widest possible range of angles, it is important that the microphones generate an identical audio signal at the output, regardless of the angle at which the sound falls on them. It is also important that the values of unwanted noise inside the system be as small as possible to improve the accuracy of calculations. To measure sounds of different frequencies, it is important that its response is as identical as possible over a wide range of frequencies.

In summary, the microphone should meet the following requirements:

- Supply voltage < 5V.

- Small size.
- Omnidirectionality.
- High signal-to-noise value.
- Flat broadband frequency response.

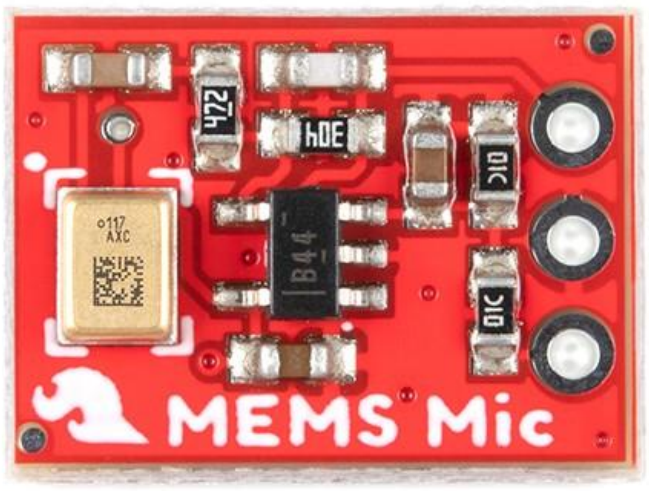
Considering the above-mentioned technical requirements, two types of microphones best meet them: micro electro mechanical system (MEMS) and electret condenser (ECM) microphones. Although both types have many similar parameters, after careful analysis I decided to use MEMS microphones for the following reasons:

- MEMS microphones have a much higher performance density than ECM microphones, which means they have better sound quality, less noise and higher precision in a given dimension.
- MEMS microphones have much more stable sensitivity under different temperature conditions compared to ECM microphones.
- MEMS microphones have less sensitivity to vibration than ECM microphones, so the system will operate more smoothly and precisely, even under vibration.
- MEMS microphones have a more uniform frequency response, especially in the high and low frequency ranges, allowing them to detect a wider range of sounds.

I used three "SparkFun Analog MEMS Microphone Breakout" microphone modules in the project. [Picture 2]. I decided on this module because [12]:

- - Has a wide frequency range from 7 Hz to 36 kHz
- - Acceptable supply voltage of 2.3 V - 3.6 V It is compatible with the microcontroller's 3.3 V supply output.

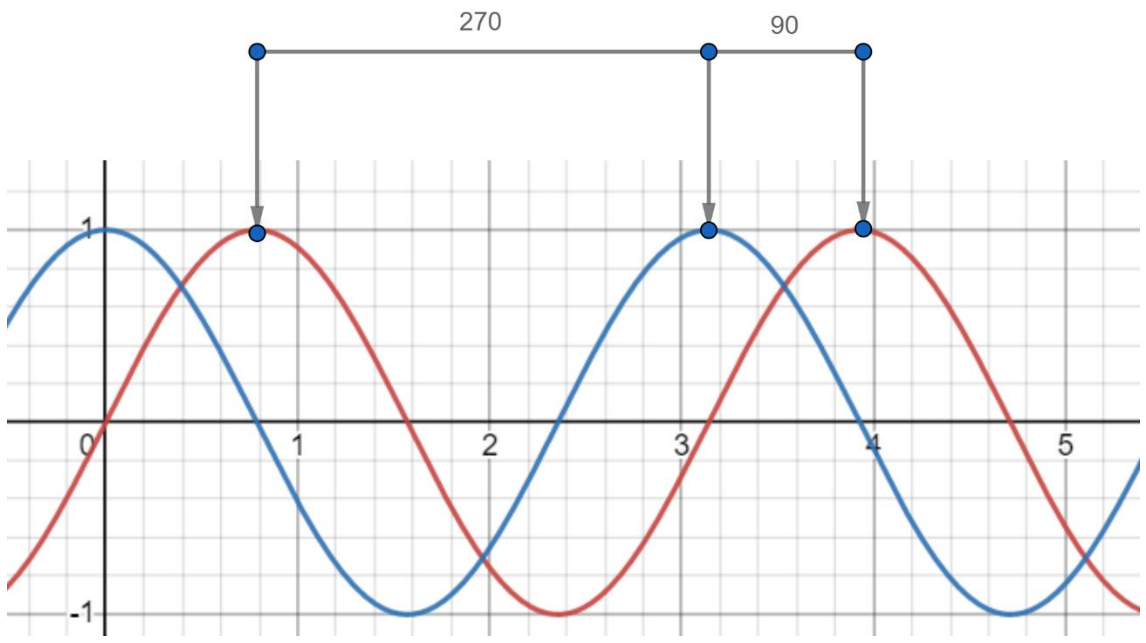
- - The output signal level which is equal to the supply voltage is ideal for working with the analog-to-digital converter used in the microcontroller design.



Picture 2. microphone SparkFun Analog MEMS Microphone Breakout [11*].*

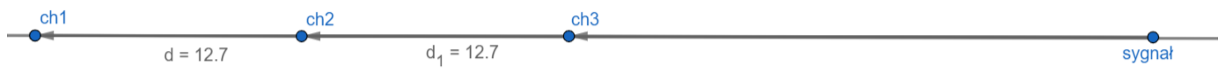
5.2. Microphone placement

The Fourier transform allows to determine in which direction the signal is shifted relative to the beginning of the data. The results are presented in the range of 360^0, but this range does not allow to precisely determine the shift of signals over time. In this situation, the result can present two possible cases: one representing the smallest angle of the shift and the other the opposite of the first one [Picture 3].



Picture 3. phase shift detection

I can't tell which is correct by making it impossible to determine whether the signal was closer to the first or second microphone. To avoid such a situation, I have to place the microphones so close to each other that the difference in the phase shift of the signals between two adjacent microphones does not exceed 180^0. The distance between the microphones is limited by the size of the modules. The smallest distance I was able to achieve was 12.7 mm. The signal takes the longest path when it falls at right angles between the two microphones [Picture 4]. So I calculated, what maximum frequency, the difference in the path that the signal will run will not be less than 12.7 mm. Suggested by the article [7], I decided to arrange the three microphones linearly because it is the simplest solution when it is not required to determine the direction of sound in three dimensions.



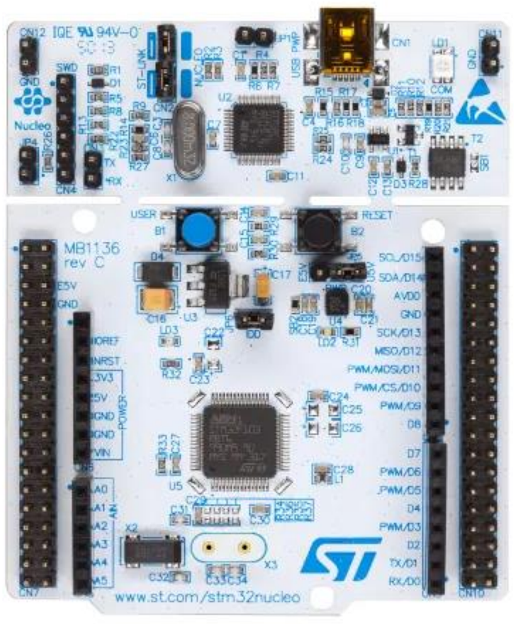
Picture 4. distance of microphones from each other

$$F_{max} = \frac{180^0}{360^0} * \frac{v_s}{d} = 13503.94\,Hz$$

The chip will correctly detect sounds up to a frequency of 13503.94 Hz.

5.3. Microcontroller

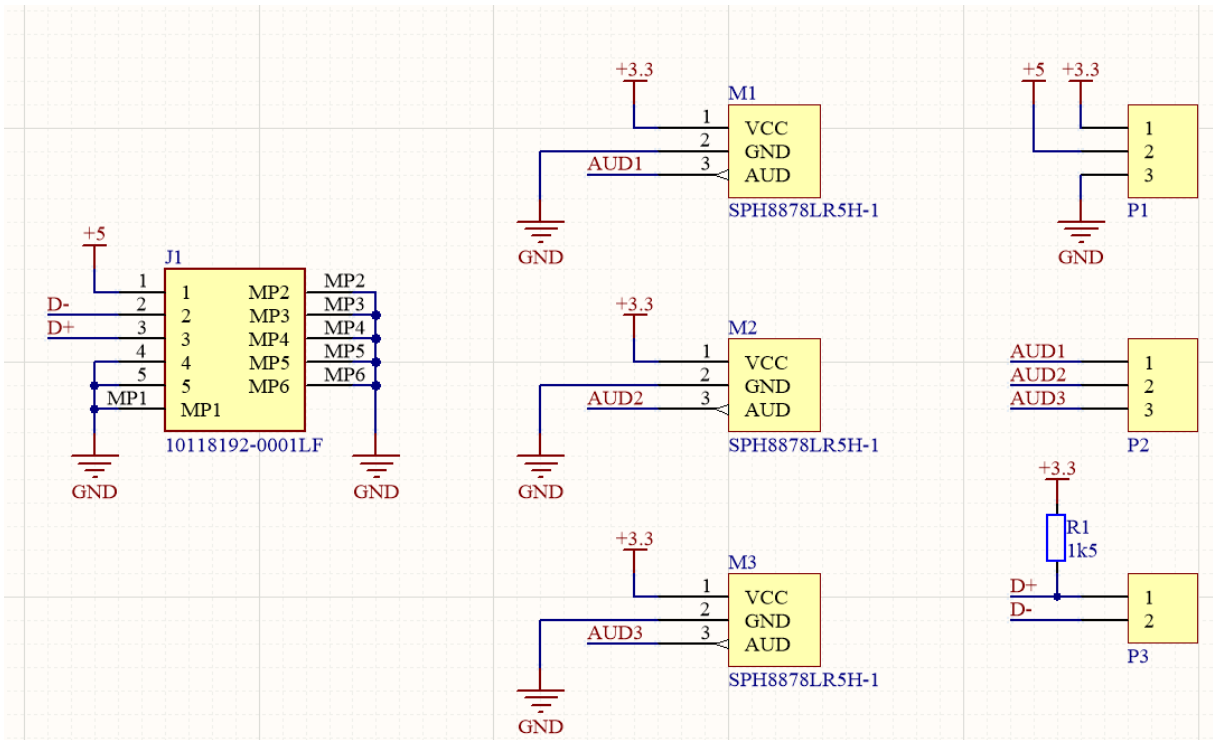
In my project, I used the Nucleo-F103RB prototype board [Picture 5], equipped with STM32F103RBT6 microcontroller manufactured by French-Italian company STMicroelectronics [*8*]. This 32-bit microcontroller is based on RISC architecture, which allows for a maximum clock frequency of 72 MHz. It has 256 KB of Flash memory and 48 KB of SRAM [*13*]. I decided on this hardware because I already had experience with programming this particular microcontroller.



Picture 5. Board Nucleo F103RBT6

5.4. Schematic diagram

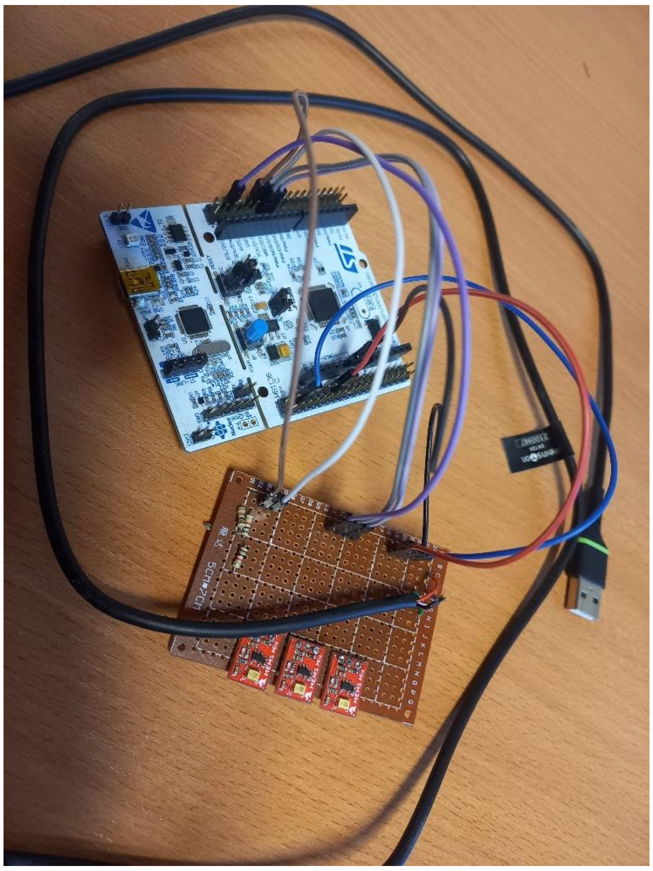
The entire circuit is powered by 5V, which comes from the USB port of the PC. This voltage is used to power the Nucleo-F103RB prototype board. This board has a built-in stabilizer, lowering the voltage to 3.3V. I use this voltage to power the microphones. In order for the PC to detect the USB device and put it into high-speed data transfer mode, the D+ bus voltage must be pulled up to 3.3V using a 1.5kΩ resistor. A schematic diagram of the system is presented in [Picture 6].



Picture 6. Schematic diagram of the measurement system

5.5. Implementation

The entire device was assembled on a prototype board. On it there are microphone modules, soldered USB cable and goldpin connectors, which allow to connect the circuit to the microcontroller. The [Picture 7] shows the finished soldered measurement circuit.



Picture 7. measurement system.

Chapter 6

6. Microcontroller software

6.1. Tool selection

The microcontroller was programmed using the STM32 CubeIDE environment, which allows easy configuration of all microcontroller inputs and outputs, clocks, communication protocols, analog-to-digital converters and more. I use the C language to program the microcontroller. This choice was made mainly because I already have experience in this environment.

6.2. Sampling frequency

The sampling frequency must be high enough to be able to correctly represent signals in the frequency range I am interested in. It is necessary to take into account the phenomenon called aliasing, which is an irreversible distortion of the signal during the sampling process that results from failure to satisfy the assumptions of the sampling theorem. This distortion is manifested by the presence of components with incorrect frequencies (aliases) in the resulting signal. Aliasing arises due to the ambiguous representation of a periodic signal by a sequence of its instantaneous values, sampled at equal intervals. By sampling a signal at a frequency of Fs, it is impossible to distinguish a harmonic signal of any frequency F0 from a harmonic waveform with frequencies k*Fs ± F0. Samples of a high-frequency signal can be misinterpreted as samples of a lower-frequency signal. The frequency after sampling can be calculated from the relationship:

$$F = |n * F_s - F_0|$$

F_0 - actual frequency of the measured signal

$n * F_s$ - the multiple of the sampling frequency lying closest to the frequency F_0

This ambiguity leads to erroneous reconstruction of the spectral components when reconstructing a continuous signal by interpolation [9]. To avoid aliasing, it is necessary to ensure that the sampled signal is band-limited to the Nyquist frequency, which is half of the sampling frequency. In this case, the value of n will be 0 so the sampled frequency will be identical to the actual frequency.

6.3. DMA

To get the best signal quality and smooth operation of the system, I used DMA (Direct Memory Access). This is a technique by which hardware connected to a microcontroller can directly transfer data to RAM or input-output ports, without the involvement of the processor. The processor only needs to program the DMA controller for the appropriate data transfer, and then release the system bus for the duration of the data transfer. The data transfer itself is already the task of the DMA controller alone so that the processor can perform other operations during this time [6].

At the beginning of the program, I initialize in the microcontroller's RAM a 16-bit buffer of size 6600 into which the data sampled by the A/D converter will be written by the DMA. This gives 2200 samples per microphone. During the measurement, 2 interrupts are generated, one when the buffer is half full and one when it is full. In these interrupts I send the sampled data from the buffer to the computer [Listing 1].

```c
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
{
	CDC_Transmit_FS(adcResultsDMA + DMA_ADC_BUF/2, sizeof(adcResultsDMA)/2);
}

void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef *hadc)
{
	CDC_Transmit_FS(adcResultsDMA, sizeof(adcResultsDMA)/2);
}
```

Listing 1. DMA Interrupts

The analog-to-digital converter operates at 12 MHz. It is not able to measure all 3 channels at exactly the same time. The time to handle one channel is data conversion time plus sampling time.

$$F_{ADC} - frequency\ of\ the\ analog\ to\ digital\ converter$$
$$F_{in} - frequency\ of\ the\ sound$$
$$c_t - data\ conversion\ time\ up\ to\ 12\ bits\ in\ cycles$$
$$s1_t - measurement\ time\ of\ 1\ channel\ in\ cycles$$
$$s2_t - measurement\ time\ of\ 2\ channel\ in\ cycles$$
$$s3_t - measurement\ time\ of\ 3\ channel\ in\ cycles$$

$$F_S = \frac{F_{ADC}}{(s1_t + c_t) + (s2_t + c_t) + (s3_t + c_t)}$$

It takes it 12.5 clock cycles to convert data from analog to digital form. To set the sampling rate I want, I change the value of the data sampling time by the DMA on each channel. I set the value on the last two channels to the minimum supported by the microcontroller of 1.5 cycles to make the difference in measurement interval between microphones as small as possible. On the first channel, I selected the value so that the sampling rate of a single channel is 125 kHz.

$$F_S = \frac{12\ 000\ 000}{(55.5 + 12.5) + (1.5 + 12.5) + (1.5 + 12.5)} = 125\ kHz$$

The value of the error angle due to sampling delay can be counted using the formula:

$$\gamma = \frac{F_{in} * (s2_t + c_t)}{F_{ADC}} * 360^0$$

The higher the frequency, the greater the error in measuring the phase shift angle. For the highest frequency that can be tested, the error is as follows:

$$\gamma = \frac{13\ 000 * (1.5 + 12.5)}{12\ 000\ 000} * 360^0 = 5.46^0$$

6.4. Transferring data to a computer

In order to transmit measurement results without data loss, it is important to select a suitable communication protocol that will ensure easy interaction with a PC and allow continuous real-time data transmission. For phase shift measurements, an analog-to-digital converter samples the signal at 125 kHz per single channel, and with 3 microphones, the system must be able to transmit 375,000 samples per second. Because of the 12-bit resolution, each sample requires a 16-bit variable, which means that the minimum data rate for this system is:

$$transfer\ speed = 375\ 000 * 16 = \ 6\ 000\ 000\ Bit/s$$

The microcontroller I have chosen can communicate using the following communication protocols:

- CAN
- I2C
- SPI
- USART
- USB

In order to effectively transfer data to the computer using CAN, I2C and SPI protocols, I would need a separate adapter, as PCs do not have direct inputs capable of handling these protocols, which would complicate the entire project, so I preferred to avoid this. This left me with a choice of USART and USB protocols. I am able to use both of these protocols via the computer's USB port. There is still the question of the speed of these protocols.

$$USART = 921600\,\frac{Bit}{s} < 6\,000\,000\,\frac{Bit}{s}$$

$$USB = 12\,000\,000\,\frac{Bit}{s} > 6\,000\,000\,\frac{Bit}{s}$$

USART is only able to transmit 921600 Bits per second which is too low. So for the project I decided to use the USB protocol.

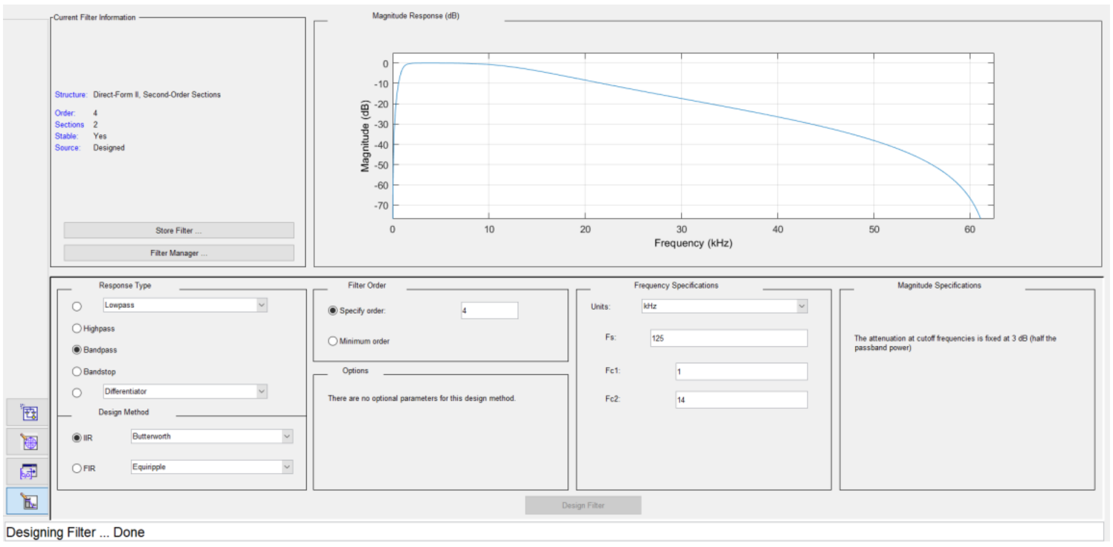
Chapter 7

7. implementation of the localization algorithm

7.1. Filtration

To effectively remove unnecessary frequencies from signals, I use a digital filter. It was designed using the "Filter Designer" tool available in the Matlab environment [Picture 8]. I chose a band-pass filter because it allows me to eliminate both lower and higher frequencies. The range of proper operation of the device is from 1 kHz to 14 kHz, so I chose a band-pass filter that will attenuate my frequencies in this range. I chose the rest of the parameters by trial and error to make its characteristics as close as possible to my expectations. The resulting filter parameters are:

- 4th order.
- IRR Butterworth.
- Range 1 kHz – 14 kHz.



Structure: Direct-Form II, Second-Order Sections
Order: 4
Sections 2
Stable: Yes
Source: Designed

Response Type: ○ Lowpass; ○ Highpass; ● Bandpass; ○ Bandstop; ○ Differentiator

Design Method: ● IIR Butterworth; ○ FIR Equiripple

Filter Order: ● Specify order: 4; ○ Minimum order

Options: There are no optional parameters for this design method.

Units: kHz
Fs: 125
Fc1: 1
Fc2: 14

The attenuation at cutoff frequencies is fixed at 3 dB (half the passband power)

Designing Filter ... Done

Picture 8. Filter Designer

Although this filter meets my requirements well, it has one inconvenience: at the beginning of the data there are large oscillations that are unrelated to the measured ambient sound. For this reason, in further calculations, I omit the first 200 samples of the filtered signal, where these oscillations are found.

7.2. Fast Fourier Transform

To detect dominant frequencies and their phase shifts in a signal, I use a method of Fourier analysis called the fast Fourier transform (FFT). This is an algorithm that allows a signal to be transformed from its original domain (often temporal or spatial) to a representation in the frequency domain. It takes a signal in the time domain and returns data in the form of complex numbers that correspond to particular frequencies in the signal. The domain is between zero and half the sampling frequency of the signal, and the resolution in the frequency domain depends on the number of samples of the signal and the sampling frequency [2] and its pitch can be calculated using the formula:

$$\Delta_F = \frac{F_s}{n_s} = \frac{125000\ Hz}{2000} = 62.5\ Hz$$

F_S – sampling rate

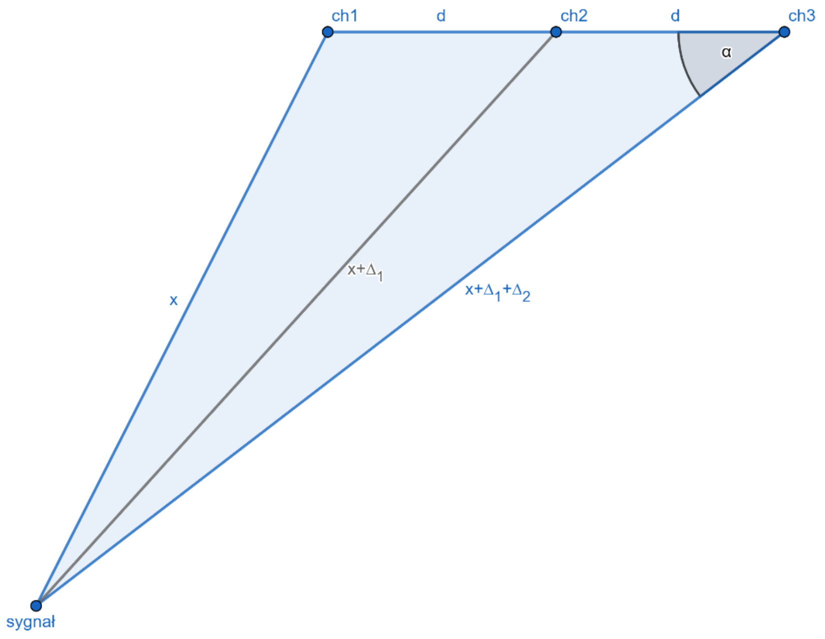
N_s – number of samples

The moduli of the complex numbers form a characteristic of the amplitude of the signal in the frequency domain while the arguments form a graph of the phase shift in the frequency domain.

With these characteristics, I am able to get information about the amplitude and phase shift of signals with frequencies of interest.

7.3. Angle of arrival calculations

When the points where the microphones and the signal source are located are connected, a triangle is formed [Picture 9]. The angle α in this triangle is the angle of incidence of sound. To be more precise in my calculations, I sum the differences in distance between microphones ch1 and ch2 and ch2 and ch3. With this averaging, in case of any errors in the measurements on the individual microphones, the final result will be much closer to reality than when calculating the angle for the two triangles separately. Using the cosine theorem, I am able to calculate the angle α. In the formula there is a variable x, which is the length of the first side of the triangle. However, this variable has no effect on the calculation, it is only important that its value is not too small to be able to build a triangle.



Picture 9. Angle of arrival calculations

d – distance between microphones

Δ_s – The difference in the path traveled by sound between 2 microphones

x – random value > 2d. Does not affect the result of the calculation

v_s – speed of sound in air = 343 m/s

$$\Delta_s = \frac{v_s}{f_{in}} * \frac{\Delta_\gamma}{360^0}$$

$$x^2 = (2d)^2 + (x + \Delta_s)^2 - 2d(x + \Delta_s)\cos\alpha$$

$$\cos\alpha = \frac{d^2 + 2x\Delta_s + \Delta_s^2}{2d(x + \Delta_s)}$$

$$\alpha = \cos^{-1}\cos\alpha$$

I calculate the angle from a point perpendicular to the microphone matrix so I have to shift the result accordingly.

$$f(x) = \begin{cases} \beta = 90 - \alpha, & \alpha > 0 \\ \beta = 90 + \alpha, & \alpha \leq 0 \end{cases}$$

7.4. Programming Language

The program for calculating the direction of incidence of sound was written in Matlab language, as it has rich sets of functions and libraries that are necessary for the project. Using the built-in solutions allows you to increase confidence that the program will work correctly, without having to install additional libraries or tools in other programming languages.

7.5. Scheme of operation of the program

The program first creates the variables sampling frequency, speed of sound in the air, a buffer to store the results and initializes the digital filter. Then it communicates with the serial port and configures an interrupt that starts the main function of the program whenever 6600 new samples appear on the USB port. [Listing 2].

```
clear device;
speed_of_sound = 343;   % m/s
Fs = 125000;            % Sampling frequency
global angles_of_arrival;
angles_of_arrival = [];

bpFilterInit =
designfilt('bandpassiir','FilterOrder',4,'HalfPowerFrequency1',1000,'HalfPowerFreq
uency2',14000,'SampleRate',Fs);

device = serialport("COM5",921600);
configureTerminator(device,"CR");
configureCallback(device, "byte", 6600, @(src, evt) callbackFcn(src, Fs,
bpFilterInit, speed_of_sound));
```

Listing 2. Inicjalizacja zmiennych i przerwania

When the interrupt is triggered, data is taken from the serial port and distributed to 3 separate channels. Subsequently, filtering of the channels and their spectral analysis is performed. When a signal with a high enough amplitude to easily distinguish it from ambient sounds is detected, the part responsible for calculations begins [Listing 3].

```
function callbackFcn(src, Fs, filtr, speed_of_sound, ~)
    global angles_of_arrival;
    data = read(src, 6600, "uint16");
    ch1 = data(1:3:end);
    ch2 = data(2:3:end);
    ch3 = data(3:3:end);

    ch1_fil = filter(filtr,double(ch1));
    ch2_fil = filter(filtr,double(ch2));
    ch3_fil = filter(filtr,double(ch3));

    [ch1_phase, ch1_frequency, ch1_amplitude] = phase_shift(ch1_fil(201:end), Fs);
    [ch2_phase, ch2_frequency, ch2_amplitude] = phase_shift(ch2_fil(201:end), Fs);
    [ch3_phase, ch3_frequency, ch3_amplitude] = phase_shift(ch3_fil(201:end), Fs);

      if (ch1_amplitude >= 7 ||  ch2_amplitude >= 7 ||  ch3_amplitude >= 7) &&
      ((ch1_frequency == ch2_frequency) && (ch2_frequency == ch3_frequency))
```

Listing 3. Data acquisition and sound detection.

In the first stage of the calculation, identification of phase shifts between microphones 1 - 2 and 2 - 3 is made [**Błąd! Nie można odnaleźć źródła odwołania.**].

```
%ch1 - ch2 difference:
    if ch1_phase > 0 && ch2_phase > 0
        if abs(ch1_phase-ch2_phase) < 180
            phase_difference_1_2 = ch1_phase - ch2_phase;
        else
            if ch1_phase > ch2_phase
                phase_difference_1_2 = 360 - (ch1_phase - ch2_phase);
            else
                phase_difference_1_2 = -(360 - (ch2_phase - ch1_phase));
            end
        end
    end

    if ch1_phase > 0 && ch2_phase < 0
        if abs(ch1_phase-ch2_phase) < 180
            phase_difference_1_2 = ch1_phase - ch2_phase;
        else
            phase_difference_1_2 = -(360 - (ch1_phase - ch2_phase));
        end
    end

    if ch1_phase < 0 && ch2_phase > 0
        if abs(ch2_phase-ch1_phase) < 180
            phase_difference_1_2 = ch1_phase - ch2_phase;
        else
            phase_difference_1_2 = (360 - (ch2_phase - ch1_phase));
        end
    end

    if ch1_phase < 0 && ch2_phase < 0
        if abs(ch2_phase - ch1_phase) < 180
            phase_difference_1_2 = ch1_phase - ch2_phase;
        else
            if ch1_phase < ch2_phase
                phase_difference_1_2 = 360 + (ch2_phase - ch1_phase);
            else
                phase_difference_1_2 = -(360 + (ch1_phase - ch2_phase));
            end
        end
    end
```

Listing 4. Identifying the phase shift between two microphones.

Based on this, I calculate the difference in distance traveled between microphones 1 - 3. Using this data, I calculate the cosine of the angle of interest and correct it so that it is presented relative to a line perpendicular to the microphone array [Listing 5].

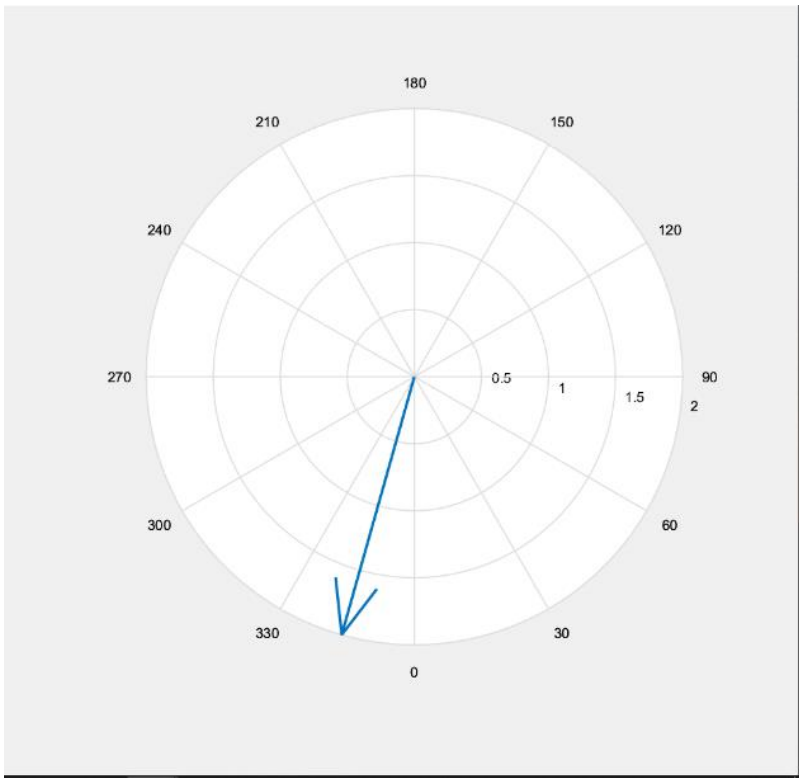
```
% Calculating the distance difference between microphones 1 and 3:
distance_difference = (speed_of_sound/ch3_frequency * phase_difference_1_3/360);
d = 2*0.0127; %distance between 2 microphones
x = 100;  %random value

cos_a =
(d^2+2*x*distance_difference+distance_difference^2)/(2*d*(x+distance_difference));

if phase_difference_1_3 > 0
    angle_of_arrival = -(90 - acosd(abs(cos_a)));
else
    angle_of_arrival = (90 - acosd(abs(cos_a)));
end

% If result of cos is greater than 1 it means thar measurements were
% incorrect and this angle must be ignored:
if not(isreal(angle_of_arrival))
    return
end

angles_of_arrival = [angles_of_arrival, angle_of_arrival];
```

Listing 5. Calculations of the direction of incidence of sound.

Matlab is not able to keep up to display the angle in real time. For this reason, I only display the arithmetic average of the last 20 measurements [Listing 6]. In addition to the smoothness of the angle display, it also gives me the opportunity to further reduce the error. I display the angle graphically as an arrow that indicates the direction of the sound [Picture 10].

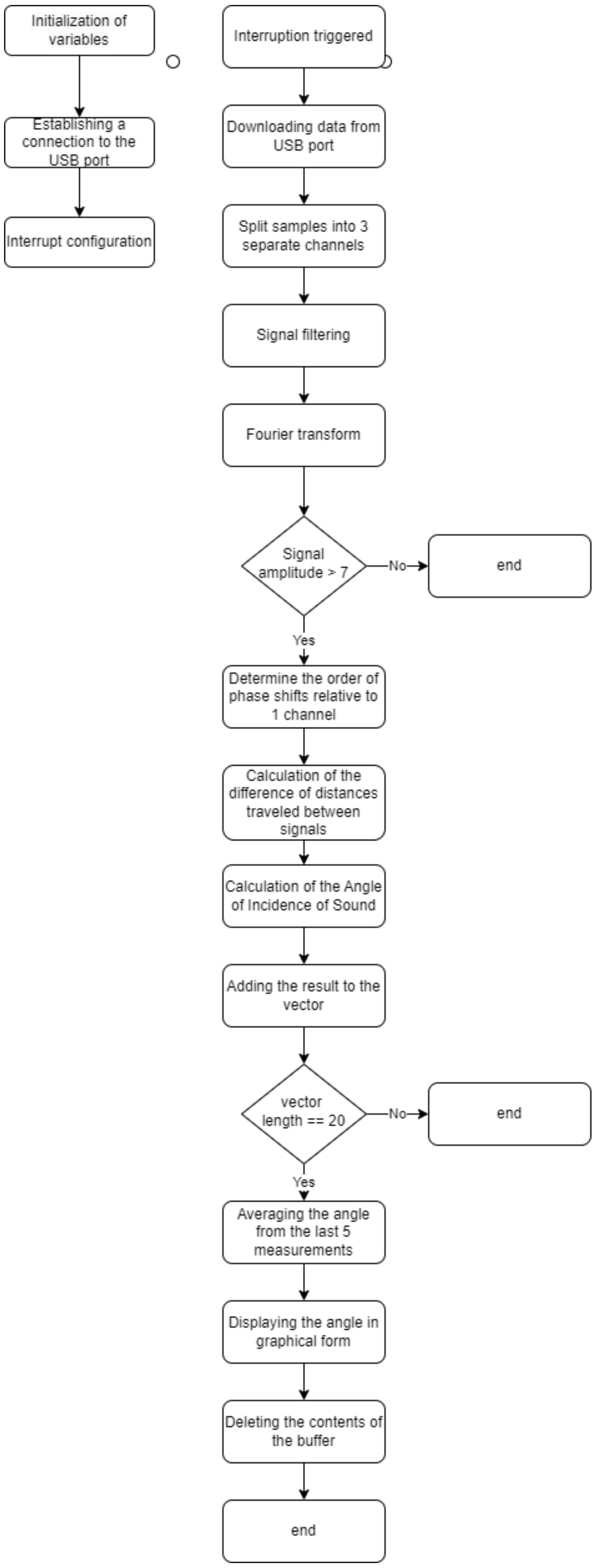
```
% Presenting average angle from last 20 measurements:
if length(angles_of_arrival) >= 20
    disp(["angle:", mean(angles_of_arrival)]);
    DOADisplay(mean(angles_of_arrival));
    angles_of_arrival = [];
end
```

Listing 6. Display of sound direction.



Picture 10. Graphical representation of sound direction.

The entire process is shown in the flowchart [Picture 11].


Initialization of variables
Establishing a connection to the USB port
Interrupt configuration
Interruption triggered
Downloading data from USB port
Split samples into 3 separate channels
Signal filtering
Fourier transform
Signal amplitude > 7
No
end
Yes
Determine the order of phase shifts relative to 1 channel
Calculation of the difference of distances traveled between signals
Calculation of the Angle of Incidence of Sound
Adding the result to the vector
vector length == 20
No
end
Yes
Averaging the angle from the last 5 measurements
Displaying the angle in graphical form
Deleting the contents of the buffer
end

Picture 11. The entire process is shown in the flowchart

Chapter 8

8. Tests

I conducted tests using a 10W speaker. After initial tests, it turned out that the range in which I am able to detect the angle is from -60^0 to 60^0. Above these ranges, the difference in the path traveled is more than the distance between the microphones. In this case, the cosine of the angle is greater than one, which makes it impossible to correctly calculate the direction of sound incidence. In terms of correct operation of the device, I conducted tests for several frequencies starting from 2 kHz and ending at 13 kHz. Higher frequencies I am unable to detect due to the distance between the microphones. The tests were conducted at distances in the range of 250 mm to 350 mm. However, it is possible to operate the device correctly also from further distances. The range of the device depends mainly on the transmitting power of the sound under test. With the speaker used in the tests, the results were correct up to a distance of about 2 meters. Below is a table with the results of the tests carried out [Table 1] [Table 2].

Real angle:	Measured angle:					
	2 kHz	3 kHz	4 kHz	5 kHz	6 kHz	7 kHz
-60	-62	-67	-59	-67	-63	-57
-50	-45	-58	-47	-47	-44	-47
-40	-42	-37	-36	-41	-40	-34
-30	-23	-34	-22	-29	-22	-20
-20	-12	-14	-18	-13	-15	-16
-10	-4	-7	-4	-7	-7	-7
0	7	2	5	5	3	3
10	16	13	20	18	17	17
20	11	19	16	26	25	22
30	32	30	32	45	33	35
40	39	38	43	56	40	44
50	54	71	64	58	64	50
60	55	42	57	67	59	62
Average measurement error:	4,92	6	5	6.69	4.46	4

Table 1. Test results 1

Real angle:	Measured angle:					
	8 kHz	9 kHz	10 kHz	11 kHz	12 kHz	13 kHz
-60	-56	-65	-63	-57	-65	-55
-50	-54	-58	-56	-50	-51	-44
-40	-44	-36	-48	-39	-37	-35
-30	-26	-30	-30	-22	-29	-24
-20	-22	-19	-23	-13	-12	-16
-10	-6	-6	-15	-3	-8	-6
0	4	8	2	1	1	6
10	15	16	8	15	11	15
20	23	22	17	27	23	25
30	32	30	32	45	33	35
40	39	38	43	56	40	44
50	54	71	64	58	64	50
60	55	42	57	67	59	62
Average measurement error:	3.77	4.46	2.85	4.46	2.92	4.77

Table 2. Test results 2

Analyzing the results of the tests, I noticed that the accuracy of the angle measurement slightly increases as the frequency of the sound increases. For lower frequencies at certain angles, the measurement error can reach as high as 20^0, while above 7 kHz the error never exceeds 10^0. A possible explanation for this phenomenon is that the measurement was not carried out in a soundproof room, which allows overlapping of sounds of the same frequency from the environment. Lower frequencies have a greater amplitude under normal conditions than higher ones, which translates into a greater effect on the value of the phase shift of the wave.

Another factor that can affect measurement accuracy is the reflection of sound off the walls. When conducting tests in an undersized room, the sound generated by the speaker can easily bounce off the walls and reach the microphones through a different path, resulting in a different phase shift than the signal that arrives directly.

Chapter 9

9. Summary

The project implemented a prototype measurement system consisting of three microphones and a microcontroller. Communication with a computer via a USB port was used, and software written in the Matlab environment was used to receive data in real time, calculate the direction of the sound source with the highest amplitude, and present the result in graphical form. Unfortunately, it turned out that the measurement range was smaller than originally intended. In the end, it was possible to achieve a range of about 120^0, and using a 10W speaker, the operating range without much measurement error was about 2 meters. The detectable frequency range is between 2 kHz and 13 kHz. This project meets the goal set at the beginning of the work but improvements are still possible. I should try to implement the more complex algorithms mentioned in Chapter 4 of the work, such as ESPRIT, MUSIC, SRP-PHAT or WAVES, and compare their accuracy and speed with the current solution. It would also be possible to write an algorithm that detects not only the signal with the highest amplitude but any number of them as needed. To facilitate the configuration of the software, I could create a dedicated graphical interface for this purpose. It would allow to set the range of frequencies the device should listen to, define the minimum amplitude level from which the detection would take place. To make the device more compact and robust it would be necessary to design a dedicated PCB. This would also allow me to not be limited by the size of the MEMS microphone modules used in the design. It would allow me to position them much closer together than 12.7 mm so I could detect much higher frequencies than I currently do.

Bibliography

[1] Page: *https://en.wikipedia.org/wiki/Direction_of_arrival*

[2] Page: *https://en.wikipedia.org/wiki/Fast_Fourier_transform*

[3] Page*:* https://en.wikipedia.org/wiki/Estimation_of_signal_parameters_via_rotational_invariance_techniques

[4] Page: *https://en.wikipedia.org/wiki/MUSIC_(algorithm)*

[5] Page: *https://en.wikipedia.org/wiki/Steered-Response_Power_Phase_Transform*

[6] Page: *https://pl.wikipedia.org/wiki/Direct_Memory_Access*

[7] “Direction Of Arrival Estimation using Microphone Array” autorstwa C Junu Jahana, MS Sinith, PP Lalu: https://ieeexplore.ieee.org/document/9673617

[8] Page: *https://pl.wikipedia.org/wiki/STM32*

[9] Page: *https://pl.wikipedia.org/wiki/Aliasing_(przetwarzanie_sygna%C5%82%C3%B3w)*

[10] Picture 1*: https://qph.cf2.quoracdn.net/main-qimg-a0b12071ca95c05006b3510d19ca2eb8-lq*

[11] *Picture* 5:

https://www.st.com/bin/ecommerce/api/image.PF259875.en.feature-description-include-personalized-no-cpn-medium.jpg

[12] MEMS microphones datasheet: *https://cdn.sparkfun.com/assets/0/5/8/b/1/SPH8878LR5H-1_Lovato_DS.pdf*

[13] STM32F103RBT6 microcontroller datasheet:

https://www.st.com/resource/en/datasheet/stm32f103ve.pdf

List of drawings

List of tables

List of listings