Politechnika Warszawska

WYDZIAŁ ELEKTRYCZNY

Instytut Elektrotechniki Teoretycznej i Systemów Informacyjno-Pomiarowych

Zakład Systemów Informacyjno-Pomiarowych

# Praca dyplomowa inżynierska

na kierunku Automatyka i Robotyka stosowana

## Lokalizacja źródeł sygnałów dźwiękowych

Arkadiusz Strzała

Numer albumu 307451

promotor
dr hab. inż. Marcin Kołodziej

WARSZAWA 2023

# Lokalizacja źródeł sygnałów dźwiękowych

## Streszczenie

Lokalizacja kierunku źródła dźwięku jest zagadnieniem, które ma kluczowe znaczenie w wielu dziedzinach współczesnej techniki. Zastosowanie znajduje w wojsku, gdzie jest podstawą działania takich technologii jak radar powietrzny i sonar z którego korzystają łodzie podwodne. Celem poniższej pracy jest zbudowanie urządzenia zdolnego wykrywać kierunek źródła dźwięku przy użyciu matrycy mikrofonów, dzięki wykorzystaniu różnicy czasu po którym dociera do nich sygnał. Zostanie do tego wykorzystana matryca złożona z mikrofonów typu MEMS oraz Mikrokontroler STM32F103RBT6 służący do próbkowania danych z mikrofonów.

**Słowa kluczowe:** kierunek źródła dźwięku, matryca mikrofonów, STM32F103RBT6, mikrofon MEMS

# Location of Sound Signal Sources

## 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.

Politechnika Warszawska
Wydział Elektryczny

## OŚWIADCZENIE

Świadomy odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa inżynierska pt.:

- została napisana przeze mnie samodzielnie,
- nie narusza niczyich praw autorskich,
- nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam, że przedłożona do obrony praca dyplomowa nie była wcześniej podstawą postępowania związanego z uzyskaniem dyplomu lub tytułu zawodowego w uczelni wyższej.

Jestem świadom, że praca zawiera również rezultaty stanowiące własności intelektualne Politechniki Warszawskiej, które nie mogą być udostępniane innym osobom
i instytucjom bez zgody Władz Wydziału Elektrycznego.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Imię i nazwisko dyplomanta: Arkadiusz Strzała

Podpis dyplomanta:

# Spis treści

# Rozdział 1

# 1. Wstęp

Lokalizacja kierunku źródła dźwięku jest kluczowym zagadnieniem w wielu dziedzinach techniki. Już podczas II wojny światowej, pierwsze zastosowania tej technologii były widoczne w konstrukcji radarów, które pozwalały na wykrywanie położenia wrogich samolotów poprzez analizę sygnałów odbitych od nich. Zastosowanie znalazło się również w sonarach, które są nieodłącznym elementem współczesnych łodzi, szczególnie tych przeznaczonych do celów wojskowych, gdzie możliwość wykrycia położenia wrogiego okrętu jest kluczowa. W niniejszej pracy skupię się tylko na estymacji kierunku sygnałów dźwiękowych lecz tą samą metodologią możliwa jest lokalizacja kierunku źródła fal radiowych. Technologia ta jest stosowana w teleskopach radiowych do obserwacji położenia ciał niebieskich w kosmosie. Jest ona również niezbędna w codziennym życiu każdego z nas, ponieważ dzięki niej możliwe jest działanie kluczowego systemu lokalizacji - GPS, który dzięki sygnałom z satelitów orbitujących nad Ziemią, potrafi z dokładnością do kilku metrów określić nasze położenie w każdym miejscu na Ziemi.

Niniejsza praca podzielona jest na 9 rozdziałów.

**Oprócz punktu wprowadzającego zawartość pozostałych rozdziałów jest następująca:**

- W rozdziale 2-gim omówiony jest cel pracy oraz jej zakres.
- Rozdział 3 zawiera porównanie różnych metod rozwiązania danego problemu oraz wyjaśnienie z jakiego powodu wybrałem moją metodę.
- 4 rozdział jest wstępem teoretycznym do zagadnienia. Wyjaśnia on ogólnie w jaki sposób możliwe jest rozwiązanie danego problemu.
- W rozdziale 5-tym opisano część sprzętową projektu. Wyjaśnione jest z jakiego powodu wybrałem poszczególne układy oraz jak je ze sobą połączyłem.
- Rozdział 6 opisuje użyte narzędzia oraz implementację oprogramowania mikrokontrolera użytego w projekcie.

- Treść rozdziału 7 opisuje wybór użytego języka programowania oraz analizę od strony matematycznej oraz programistycznej napisanego programu po stronie komputera PC.
- W rozdziale 8 prezentuję wyniki z przeprowadzonych testów oraz ich analizę.
- Rozdział 9 to podsumowanie tego co udało mi się osiągnąć, co wyszło nie do końca tak jak tego oczekiwałem oraz pomysły jak można by ulepszyć ten projekt.

# Rozdział 2

# 2. Cel i zakres pracy

Celem tej pracy jest zbudowanie urządzenia, które będzie zdolne do wykrywania kierunku źródła dźwięku o największej amplitudzie. W ramach pracy przeprowadzę projektowanie systemu pomiarowego, który będzie służył do ciągłego monitorowania sygnałów audio z otoczenia. System ten będzie w stanie przesyłać dane pomiarowe do komputera w czasie rzeczywistym. Po stronie komputera zostanie zaimplementowany algorytm, który na podstawie sygnałów audio z mikrofonów obliczy kierunek źródła dźwięku. System będzie w stanie wyświetlać wyniki obliczeń w czasie rzeczywistym w formie graficznej.

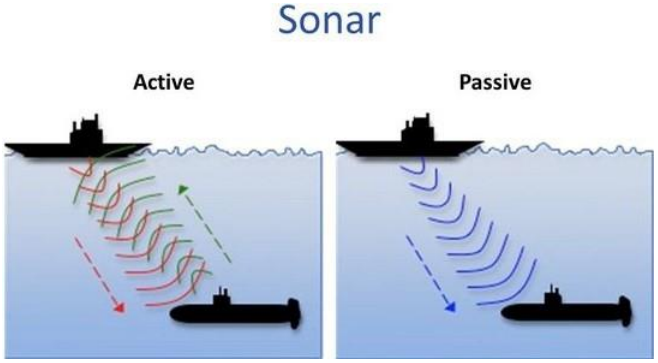
# Rozdział 3

# 3. Metody detekcji źródła dźwięku

Detekcja źródła dźwięku polega na wykorzystaniu macierzy sensorów, dostosowanych do odbioru określonego sygnału. Następnie, poprzez analizę danych pochodzących z każdego z sensorów, należy określić opóźnienie czasowe między przybyciem fali do poszczególnych sensorów w macierzy. Mając dostęp do wszystkich opóźnień czasowych oraz dokładnych pozycji sensorów względem siebie, możliwe jest obliczenie geometrycznie kąta źródła sygnału. To podejście pozwala na określenie źródła dźwięku, poprzez analizę danych pochodzących z wielu różnych sensorów [2].

Aby lepiej zrozumieć, gdzie ta technologia jest stosowana w praktyce, przybliżę działanie sonarów w łodziach podwodnych. Istnieją dwa typy sonarów: aktywne i pasywne. Różnice w ich działaniu przedstawia [Rysunek 1]. Sonar aktywny wysyła dźwięk i następnie nasłuchuje, po jakim czasie i z jakiego kierunku sygnał powraca do łodzi. To rozwiązanie pozwala dość precyzyjnie określić odległość obiektów od łodzi. Minusem jest jednak to, że wysyłając dźwięk, łódź ujawnia swoją pozycję, co jest dużym problemem w zastosowaniach militarnych. Aby wyeliminować tę niedogodność, we współczesnych łodziach podwodnych często stosuje się sonar pasywny. Działanie jego polega na nasłuchiwaniu dźwięków, z otoczenia za pomocą matrycy hydrofonów. Dzięki temu jest w stanie wykrywać kierunek źródła dźwięku, jednak niestety nie jest w stanie określić odległości, z jakiej dociera dany dźwięk.

Celem pracy jest próba odwzorowania podstawowych funkcjonalności tego sonaru w środowisku lądowym za pomocą mikrofonów.




*Rysunek 1. Porównanie działania sonaru pasywnego z aktywnym [10]*

# Rozdział 4

# 4. Porównanie stosowanych rozwiązań

Wykrywanie różnicy w czasie dotarcia sygnałów pomiędzy sensorami jest kluczowym i skomplikowanym aspektem tego zagadnienia. Aby to osiągnąć, korzystamy z informacji o przesunięciu fazy interesującego nas dźwięku. Służy do tego algorytm szybkiej transformacji Fouriera (FFT), który pozwala na przekształcenie sygnału z domeny czasu do domeny częstotliwości i umożliwia analizę spektralną sygnału, co pozwala na obliczenie jego kąta przesunięcia fazowego [2].

W moim projekcie, korzystam tylko z tej informacji do obliczenia różnicy czasu sygnałów. Jest to prosta metoda implementacji, która nie wymaga dużej mocy obliczeniowej, dzięki czemu urządzenie może pracować w czasie rzeczywistym, nawet na komputerze z niższą wydajnością.

Istnieją jednak bardziej złożone algorytmy które korzystają z analizy spektralnej sygnału do obliczania kierunku źródła dźwięku. Najczęściej stosowane algorytmy to ESPRIT (Estimation of Signal Parameters via Rotational Invariance Techniques), MUSIC (MUltiple SIgnal Classification) i SRP-PHAT (Steered Response Power Phase Transform). Poniżej przedstawię wady i zalety tych rozwiązań.

Esprit polega na wykorzystaniu właściwości rotacyjnej macierzy korelacji, które pozwalają na określenie kierunku źródła dźwięku bez konieczności estymacji amplitudy sygnału. Algorytm korzysta z dwóch lub więcej mikrofonów znajdujących się w różnych pozycjach, aby zbadać sygnał z różnych perspektyw. Jego zalety to skuteczność działania w przypadku wielu źródeł dźwięku oraz trudno rozróżnialnych sygnałów. Jest również prosty w implementacji. Wadą jest natomiast to iż jest mało odporny na szum oraz nie radzi sobie z detekcją źródeł dźwięków znajdujących się blisko siebie. Jest dobrym wyborem, gdy jest mało źródeł dźwięku i dobrze rozróżnialne sygnały. Dokładność wyniku zależy od liczby mikrofonów i ich rozmieszczenia [3].

MUSIC jest metodą pozycjonowania fazowego, która polega na estymacji kierunku źródła dźwięku poprzez analizę wektorów własnych macierzy korelacji sygnałów z różnych mikrofonów. Dobrze sobie radzi w przypadku wielu źródeł dźwięku i trudno rozróżnialnych sygnałów.

Jest dużo mniej odporny na szum niż inne metody pozycjonowania fazowego, wymaga dobrego rozmieszczenia mikrofonów i odpowiedniej ich liczby do wielkości i kształtu pomieszczenia.
To dobry wybór, gdy jest wiele źródeł dźwięku i trudne do rozróżnienia sygnały. Ostateczna dokładność wyniku zależy od liczby mikrofonów i ich rozmieszczenia oraz warunków akustycznych. Jest to jeden z najpopularniejszych algorytmów wykrywania kierunku źródła dźwięku [4].

SRP-PHAT to pochodna algorytmu SRP (Steered Response Power) i polega na wykorzystaniu fazy sygnału mikrofonów w celu określenia kierunku kierunku źródła dźwięku. Algorytm korzysta z dwóch lub więcej mikrofonów znajdujących się w różnych pozycjach, aby zbadać sygnał z różnych perspektyw.
Cechuje się odpornością na szum, jest w stanie określić kierunek źródła dźwięku nawet w przypadku, gdy jest wiele źródeł dźwięku oraz prostotą implementacji.
Ma on jednak trudności z lokalizacją źródeł dźwięku blisko siebie, może dawać niepewne wyniki w przypadku, gdy źródło dźwięku jest bardzo blisko mikrofonów.
Jest dobrym wyborem w przypadku dużej ilości szumów, oraz gdy jest wiele źródeł dźwięku. Ostateczna dokładność wyniku zależy od liczby mikrofonów i ich rozmieszczenia oraz warunków akustycznych [5].

# Rozdział 5

# 5. Konstrukcja urządzenia

Najprostszym sposobem na rozwiązanie tego problemu byłoby zakupienie karty dźwiękowej z trzema wejściami audio oraz mikrofonów z wyjściem jack. Niestety, takie rozwiązanie jest dość kosztowne, ponieważ karty te mogą kosztować nawet kilka tysięcy złotych. Z tego powodu postanowiłem skorzystać z tańszego rozwiązania. Zdecydowałem się na wykorzystanie mikrokontrolera, którym będę dokonywał pomiarów z mikrofonów za pomocą przetwornika analogowo-cyfrowego. Jest to bardziej ekonomiczne rozwiązanie, które pozwala na osiągnięcie tego samego celu, bez konieczności wydawania dużych kwot na specjalistyczne urządzenia.

## 5.1. Mikrofony

Aby urządzenie działało prawidłowo i z jak największą dokładnością, mikrofony muszą spełniać pewne wymagania. Z praktycznego punktu widzenia, ważne jest, aby napięcie zasilania mikrofonu nie przekraczało 5V, dzięki czemu uniknę potrzeby stosowania dodatkowej przetwornicy podwyższającej napięcie i będę mógł stosować napięcie prosto z portu USB. Mikrofony powinny być niewielkich rozmiarów, aby możliwe było umieszczanie ich blisko siebie. Aby mierzyć jak najszerszy możliwy zakres kątów, ważne jest, aby mikrofony generowały na wyjściu identyczny sygnał audio, niezależnie od kąta, pod jakim pada na nie dźwięk. Ważne jest także, aby wartości niepożądanych szumów wewnątrz układu były jak najmniejsze, co poprawi dokładność obliczeń. Aby mierzyć dźwięki o różnych częstotliwościach, ważne jest, aby jego odpowiedź była jak najbardziej identyczna w szerokim zakresie częstotliwości.

**Podsumowując mikrofon powinien spełniać następujące wymagania:**

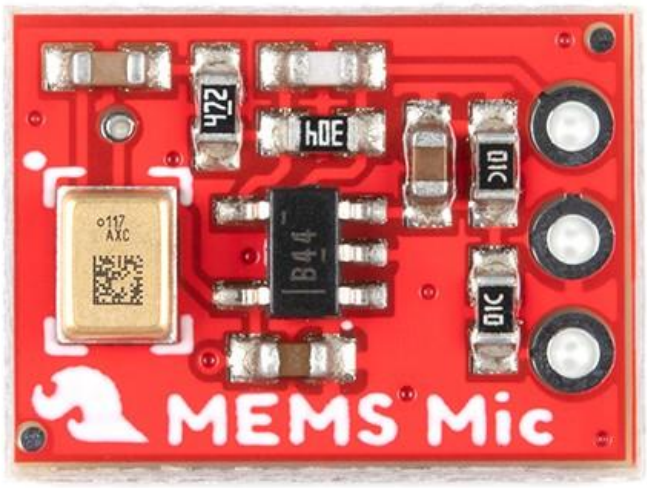
- Napięcie zasilania < 5V.
- Małe rozmiary.
- Wielokierunkowość.
- Wysoka wartość sygnału do szumu.
- Płaska szerokopasmowa odpowiedź częstotliwościowa.

Mając na uwadze wyżej wymienione wymagania techniczne, najlepiej spełniają je dwa typy mikrofonów: mikrofony MEMS (micro electro mechanical system) oraz ECM (electret condenser). Chociaż oba rodzaje charakteryzują się wieloma podobnymi parametrami, po dokładnej analizie zdecydowałem się na zastosowanie mikrofonów MEMS z następujących powodów:

- Mikrofony MEMS posiadają znacznie wyższą gęstość wydajności niż mikrofony ECM, co oznacza, że charakteryzują się one lepszą jakością dźwięku, mniejszym poziomem szumów i większą precyzją w danym wymiarze.
- Mikrofony MEMS charakteryzują się dużo bardziej stabilną czułością w różnych warunkach temperaturowych w porównaniu do mikrofonów ECM.
- Mikrofony MEMS posiadają mniejszą wrażliwość na wibracje niż mikrofony ECM, dzięki czemu układ będzie pracował płynniej i precyzyjniej, nawet pod wpływem drgań.
- Mikrofony MEMS posiadają bardziej jednolitą odpowiedź częstotliwościową, szczególnie w zakresie wysokich i niskich częstotliwości, co pozwala na wykrycie szerszego zakresu dźwięków.

W projekcie użyłem trzech modułów mikrofonów „SparkFun Analog MEMS Microphone Breakout” zaprezentowanych na [Rysunek 2]. Zdecydowałem się na ten moduł gdyż [12]:

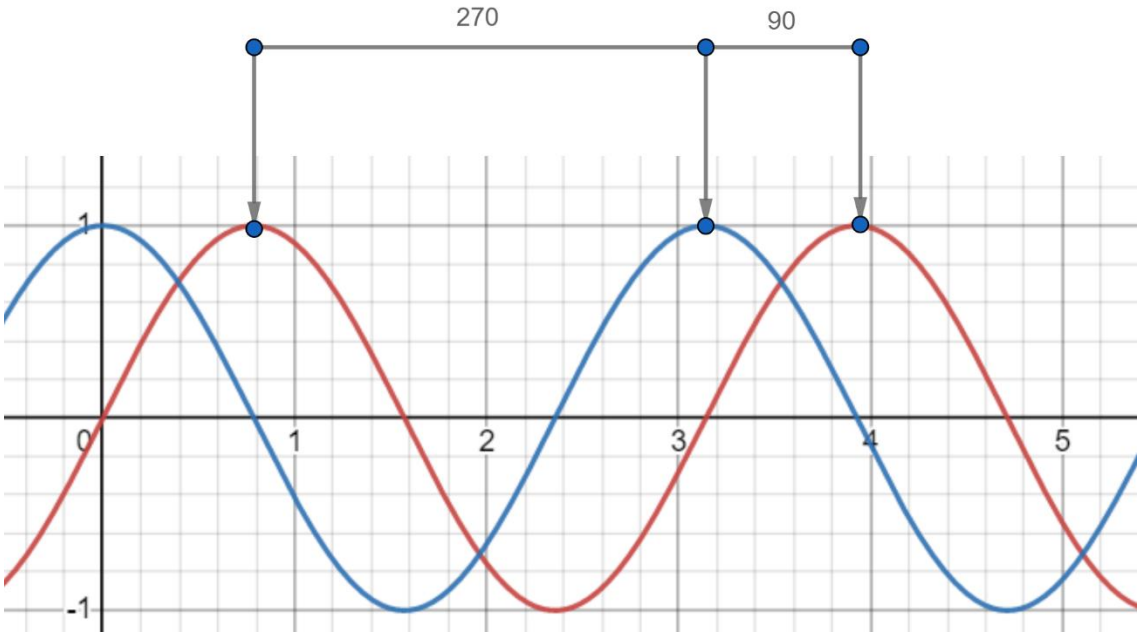
- ma szeroki zakres częstotliwości od 7 Hz do 36 kHz
- Dopuszczalne napięcie zasilania 2.3 V – 3.6 V Jest kompatybilne z wyjściem zasilającym mikrokontrolera 3.3 V.
- poziom sygnału wyjściowego który równa się napięciu zasilania nadaje się idealnie do pracy z przetwornikiem analogowo cyfrowym użytego w projekcie mikrokontrolera.




*Rysunek 2. Mikrofon SparkFun Analog MEMS Microphone Breakout [11].*

## 5.2. Ustawienie mikrofonów

Transformacja Fouriera [2] pozwala na określenie, w jakim kierunku sygnał jest przesunięty względem początku danych. Rezultaty są przedstawione w zakresie $360^0$, jednak ten zakres nie pozwala na precyzyjne określenie przesunięcia sygnałów w czasie. W takiej sytuacji, wynik może przedstawiać dwa możliwe przypadki: jeden reprezentujący najmniejszy kąt przesunięcia oraz drugi odwrotny do pierwszego jak widać na [Rysunek 3].




*Rysunek 3. detekcja przesunięcia fazowego*

Nie jestem w stanie stwierdzić który jest prawidłowy przez co określenie czy sygnał znajdował się bliżej pierwszego czy drugiego mikrofonu jest niemożliwe. Aby uniknąć takiej sytuacji, muszę umieścić mikrofony tak blisko siebie, aby różnica przesunięcia fazy sygnałów między dwoma sąsiednimi mikrofonami nie przekraczała $180^0$. Odległość między mikrofonami jest ograniczona rozmiarem modułów. Najmniejsza odległość jaką udało mi się osiągnąć, to 12.7 mm. Sygnał przebiega najdłuższą drogę, gdy pada on pod kątem prostej między dwoma mikrofonami co pokazuje [Rysunek 4]. Obliczyłem zatem, jakiej maksymalnej częstotliwości, różnica drogi, jaką przebiegnie sygnał, nie będzie mniejsza niż 12.7 mm. Sugerując się artykułem [7], zdecydowałem się na ułożenie trzech mikrofonów liniowo gdyż jest to najprostsze rozwiązanie gdy nie jest wymagane wyznaczanie kierunku dźwięku w trzech wymiarach.

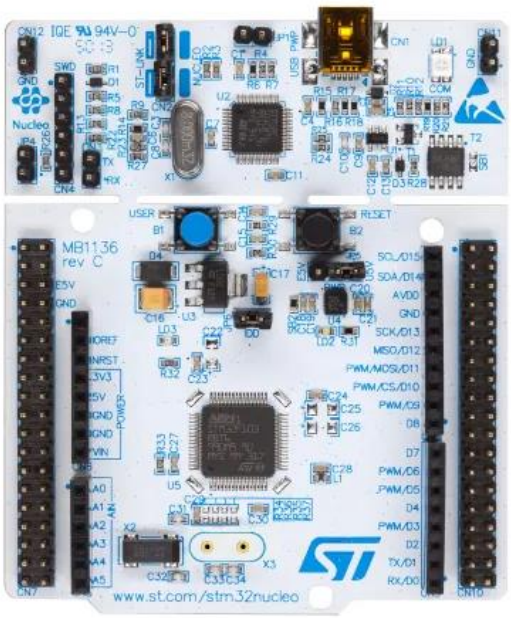

*Rysunek 4. odległość mikrofonów od siebie*

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

Układ będzie wykrywał poprawnie dźwięki do częstotliwości 13503.94 Hz.

## 5.3. Mikrokontroler

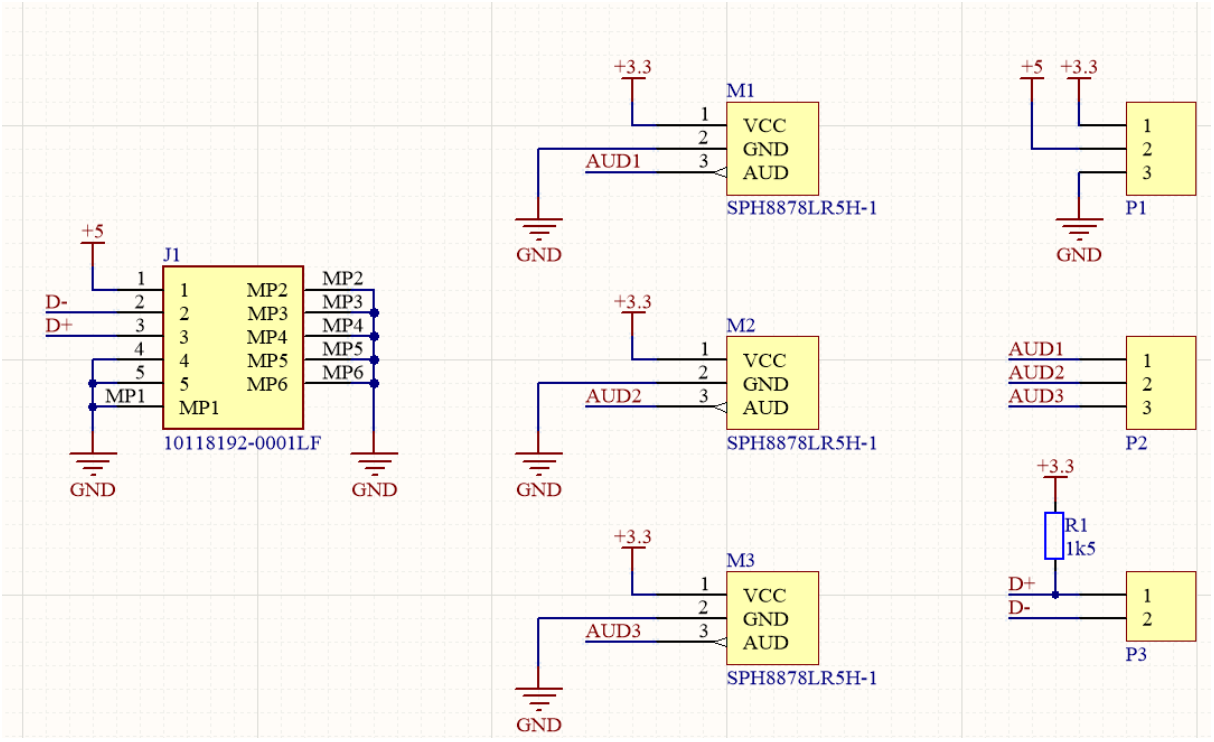
W moim projekcie zastosowałem płytkę prototypową Nucleo-F103RB zaprezentowaną na [Rysunek 5], wyposażoną w mikrokontroler STM32F103RBT6 produkowany przez francusko-włoską firmę STMicroelectronics [8]. Ten 32-bitowy mikrokontroler oparty jest na architekturze RISC, co pozwala na uzyskanie maksymalnej częstotliwości taktowania 72 MHz. Posiada on 256 KB pamięci Flash oraz 48 KB SRAM [13]. Zdecydowałem się na ten sprzęt, ponieważ miałem już doświadczenie z programowaniem tego konkretnego mikrokontrolera.




*Rysunek 5. Płytka Nucleo F103RBT6*

## 5.4. Schemat układu

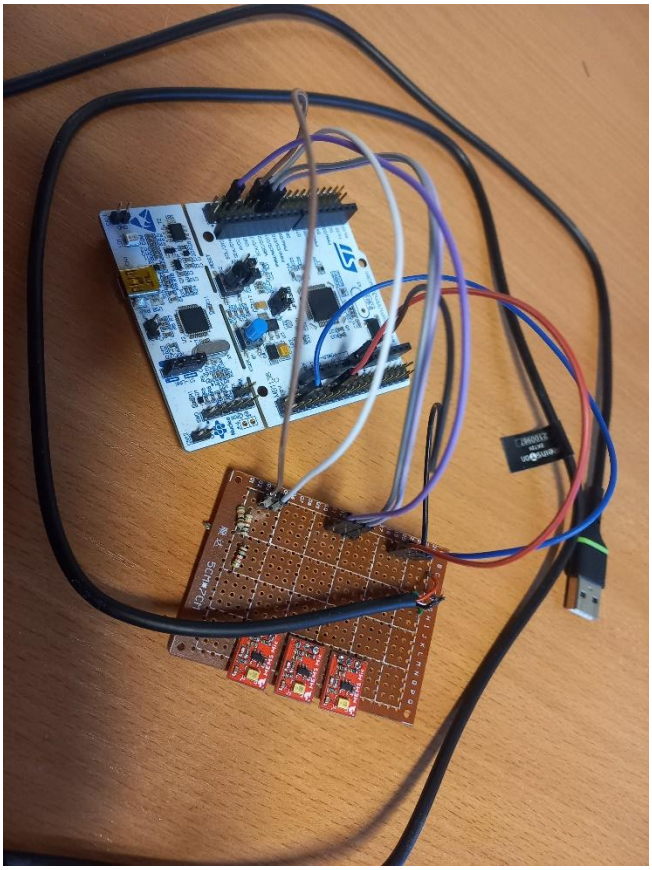
Cały układ jest zasilany napięciem 5V, które pochodzi z portu USB komputera PC. Napięcie to jest używane do zasilania płytki prototypowej Nucleo-F103RB. Płytka ta posiada wbudowany stabilizator, obniżający napięcie do 3.3V. Napięcia tego używam do zasilania mikrofonów. Aby komputer PC wykrył urządzenie USB i ustawił je w tryb szybkiej transmisji danych, należy podciągnąć napięcie szyny D+ do 3.3V za pomocą rezystora 1.5kΩ. Na [Rysunek 6] zaprezentowany jest schemat ideowy układu.




*Rysunek 6. Schemat ideowy układu pomiarowego*

## 5.5. Wykonanie

Całe urządzenie zostało zmontowane na płytce prototypowej. Na niej znajdują się moduły mikrofonów, przylutowany kabel USB oraz złącza goldpin, które pozwalają na połączenie układu z mikrokontrolerem. Na [Rysunek 7] pokazany jest gotowy zlutowany układ pomiarowy.



*Rysunek 7. Układ pomiarowy.*

# Rozdział 6

# 6. Oprogramowanie mikrokontrolera

## 6.1. Wybór narzędzi

Mikrokontroler został zaprogramowany za pomocą środowiska STM32 CubeIDE, które umożliwia łatwą konfigurację wszystkich wejść i wyjść mikrokontrolera, zegarów, protokołów komunikacyjnych, przetworników analogowo-cyfrowych i innych. Do programowania mikrokontrolera używam języka C. Ten wybór został dokonany głównie ze względu na posiadanie przeze mnie już doświadczenia w tym środowisku.

## 6.2. Częstotliwość próbkowania

Częstotliwość próbkowania musi być odpowiednio wysoka, aby możliwe było prawidłowe odwzorowanie sygnałów w zakresie częstotliwości, który mnie interesuje. Należy wziąć pod uwagę zjawisko zwane aliasingiem, które jest to nieodwracalne zniekształcenie sygnału podczas procesu próbkowania, które wynika z niespełnienia założeń twierdzenia o próbkowaniu. Zniekształcenie to objawia się obecnością w wynikowym sygnale składowych o błędnych częstotliwościach (aliasów). Aliasing powstaje z powodu niejednoznacznej reprezentacji sygnału okresowego przez ciąg jego wartości chwilowych, pobranych w równych odstępach czasu. Dokonując próbkowania sygnału z częstotliwością $F_s$ nie można odróżnić sygnału harmonicznego o dowolnej częstotliwości $F_0$ od przebiegu harmonicznego o częstotliwościach $k*F_s \pm F_0$. Próbki sygnału o wysokiej częstotliwości mogą być mylnie zinterpretowane jako próbki sygnału o niższej częstotliwości. Częstotliwość po spróbkowaniu można obliczyć z zależności:

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

$F_0$ - częstotliwość rzeczywista sygnału mierzonego

$n * F_s$ - wielokrotność częstotliwości próbkowania leżąca najbliżej częstotliwości $F_0$

Ta niejednoznaczność prowadzi do błędnego odtworzenia składowych widmowych przy rekonstrukcji sygnału ciągłego poprzez interpolację [9]. Aby uniknąć aliasingu należy zapewnić, aby sygnał próbkowany był ograniczony pasmowo do częstotliwości Nyquista, czyli połowy częstotliwości próbkowania. W takim przypadku wartość n będzie wynosić 0 więc częstotliwość spróbkowana będzie identyczna co rzeczywista.

## 6.3. DMA

Aby uzyskać jak najlepszą jakość sygnału oraz płynne działanie systemu, skorzystałem z DMA (Direct Memory Access). Jest to technika, dzięki której sprzęt podłączony do mikrokontrolera może bezpośrednio przesyłać dane do pamięci operacyjnej RAM lub portów wejścia-wyjścia, bez udziału procesora. Procesor musi tylko zaprogramować kontroler DMA do odpowiedniego transferu danych, a następnie na czas przesyłania danych zwolnić magistralę systemową. Sam transfer danych jest już zadaniem wyłącznie kontrolera DMA dzięki czemu procesor w tym czasie może wykonywać inne operacje [6].

Na początku programu inicjalizuję w pamięci RAM mikrokontrolera 16 bitowy bufor o rozmiarze 6600 do którego będą wpisywane przez DMA dane próbkowane przez przetwornik analogowo-cyfrowy. Daje to 2200 próbek na jeden mikrofon. Podczas pomiaru generują się 2 przerwania widoczne na [Listing 1]. Pierwsze gdy bufor jest w połowie zapełniony a drugie gdy jest pełen. W przerwaniach tych wysyłam spróbkowane dane z bufora do komputera.

```
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. Przerwania DMA*

Przetwornik analogowo cyfrowy pracuje z częstotliwością 12 MHz. Nie jest w stanie on zmierzyć wszystkich 3 kanałów dokładnie w tym samym czasie. Czas obsługi jednego kanału to czas konwersji danych plus czas próbkowania.

$$F_{ADC} - częstotliwość\ przetwornika\ analogowo\ cyfrowego$$
$$F_{in} - częstotliwość\ badanego\ dźwięku$$
$$c_t - czas\ konwersji\ danych\ do\ 12\ bit\ w\ cyklach$$
$$s1_t - czas\ pomiaru\ 1\ kanału\ w\ cyklach$$
$$s2_t - czas\ pomiaru\ 2\ kanału\ w\ cyklach$$
$$s3_t - czas\ pomiaru\ 3\ kanału\ w\ cyklach$$

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

Czas konwersji danych z postaci analogowej do cyfrowej zajmuje mu 12.5 cykli zegarowych. Aby ustawić pożądaną przeze mnie częstotliwość próbkowania zmieniam wartość czasu próbkowania danych przez DMA na każdym z kanałów. Wartość na dwóch ostatnich kanałach ustawiam na minimalną obsługiwaną przez mikrokontroler 1.5 cykli aby różnica w odstępie pomiaru pomiędzy mikrofonami była jak najmniejsza. Na pierwszym kanale wartość tą dobrałem tak aby częstotliwość próbkowania pojedynczego kanału wyniosła 125 kHz.

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

Wartość kąta błędu spowodowanego opóźnieniem w próbkowaniu można policzyć za pomocą wzoru:

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

Wynika z tego, że im wyższa częstotliwość, tym większy błąd pomiaru kąta przesunięcia fazowego. Dla najwyższej możliwej do zbadania częstotliwości, błąd ten jest następujący:

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

## 6.4. Przesyłanie danych do komputera

Aby przesłać wyniki pomiarów bez utraty danych, ważne jest dobranie odpowiedniego protokołu komunikacyjnego, który zapewni łatwą współpracę z komputerem PC oraz pozwoli na ciągłe przesyłanie danych w czasie rzeczywistym. W celu pomiarów przesunięcia fazowego, przetwornik analogowo-cyfrowy próbkuje sygnał z częstotliwością 125 kHz na pojedynczy kanał, a przy użyciu 3 mikrofonów, system musi być w stanie przesłać 375 tysięcy próbek na sekundę. Z uwagi na rozdzielczość 12 bitów, każda próbka wymaga zmienną 16-bitową, co oznacza, że minimalna prędkość przesyłania danych dla tego układu wynosi:.

$$prędkość\ przesyłu = 375\,000 * 16 = \ 6\,000\,000\ Bit/s$$

Wybrany przeze mnie mikrokontroler potrafi się komunikować za pomocą następujących protokołów komunikacyjnych:

- CAN
- I2C
- SPI
- USART
- USB

Do przesyłania danych do komputera za pomocą protokołów CAN, I2C oraz SPI potrzebował bym osobnej przejściówki gdyż komputery klasy PC nie posiadają bezpośrednich wejść zdolnych obsłużyć te protokoły co skomplikowało by cały projekt więc wolałem tego uniknąć. Pozostają mi więc do wyboru protokoły USART oraz USB.

Z obu tych protokołów jestem w stanie korzystać za pomocą portu USB komputera. Pozostaje jeszcze kwestia szybkości tych protokołów.

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

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

USART jest w stanie przesyłać tylko 921600 Bitów na sekundę co jest zbyt małą wartością. Do projektu zdecydowałem się użyć więc protokołu USB.

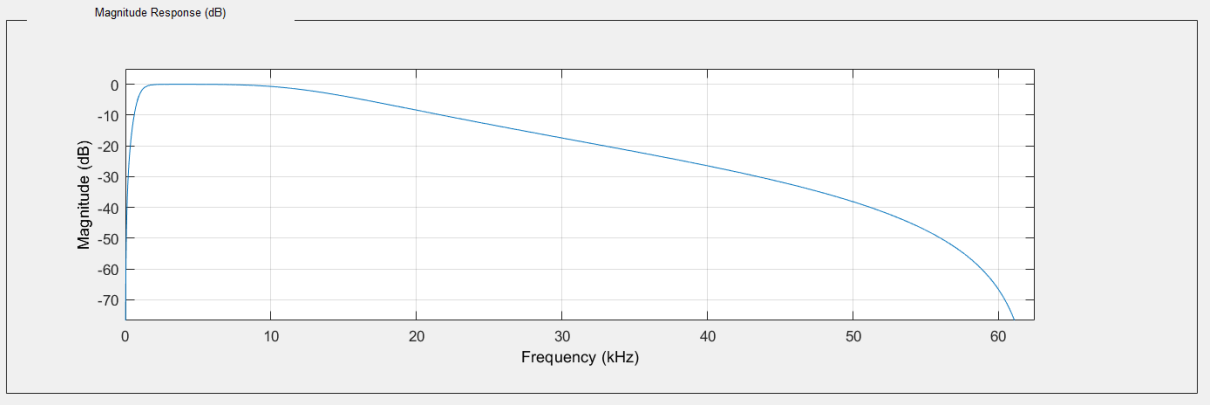
# Rozdział 7

# 7. Implementacja algorytmu lokalizacji

## 7.1. Filtracja

Aby skutecznie usunąć niepotrzebne częstotliwości z sygnałów, korzystam z filtru cyfrowego. Został on zaprojektowany za pomocą narzędzia „Filter Designer" dostępnego w środowisku Matlab. Zakres poprawnej pracy urządzenia wynosi od 1 kHz do 14 kHz więc wybrałem filtr środkowo-przepustowy który wytłumi mi częstotliwości z tego zakresu. Resztę parametrów dobrałem metodą prób i błędów aby jago charakterystyka jak najbardziej odpowiadała moim oczekiwaniom. Charakterystyka zaprojektowanego filtra przedstawia [Rysunek 8]. Otrzymane parametry filtra to:

- 4 rząd.
- IRR Butterworth.
- Zakres 1 kHz – 14 kHz.




*Rysunek 8. Charakterystyka zaprojektowanego filtru*

Choć filtr ten dobrze spełnia moje wymagania, posiada jedną niedogodność: na początku danych występują duże oscylacje, które nie mają związku z mierzonym dźwiękiem otoczenia. Z tego powodu, w dalszych obliczeniach, pomijam pierwsze 200 próbek przefiltrowanego sygnału, gdzie znajdują się te oscylacje.

## 7.2. Szybka transformata Fouriera

Aby wykryć dominujące częstotliwości oraz ich przesunięcia fazowe w sygnale, korzystam z metody analizy Fouriera, zwanej szybką transformatą Fouriera [2]. Jest to algorytm, który pozwala na przekształcenie sygnału z jego oryginalnej domeny (często czasowej lub przestrzennej) na reprezentację w domenie częstotliwości. Przyjmuje on sygnał w dziedzinie czasu i zwraca dane w postaci liczb zespolonych, które odpowiadają poszczególnym częstotliwościom w sygnale. Dziedzina ta jest w przedziale od zera do połowy częstotliwości próbkowania sygnału, a rozdzielczość w dziedzinie częstotliwości jest uzależniona od ilości próbek sygnału oraz częstotliwości próbkowania a jej skok można obliczyć za pomocą wzoru:

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

$F_s$ – częstotliwość próbkowania
$N_s$ – ilość sampli

Moduły liczb zespolonych tworzą charakterystykę amplitudy sygnału w dziedzinie częstotliwości natomiast argumenty tworzą wykres przesunięć fazowych w dziedzinie częstotliwości.

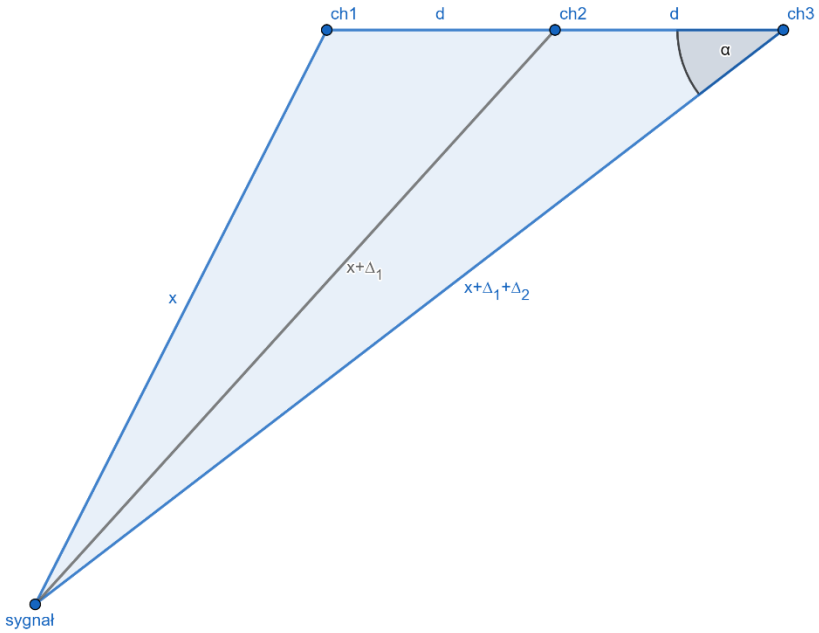
Dzięki tym charakterystykom jestem w stanie otrzymać informacje o amplitudzie oraz przesunięciu w fazie sygnałów o interesujących mnie częstotliwościach.

## 7.3. Obliczenia kierunku źródła dźwięku

Po połączeniu punktów w których znajdują się mikrofony oraz źródło sygnału powstaje trójkąt widoczny na [Rysunek 9]. Kąt α w tym trójkącie jest kątem padania dźwięku. Aby uzyskać większą precyzję w obliczeniach, dokonuję sumowania różnic odległości między mikrofonami ch1 i ch2 oraz ch2 i ch3. Dzięki temu uśrednieniu, w przypadku jakichkolwiek błędów w pomiarach na poszczególnych mikrofonach, końcowy rezultat

będzie znacznie bardziej zbliżony do rzeczywistości niż przy liczeniu kąta dla dwóch trójkątów oddzielnie.

Korzystając z twierdzenia cosinusów, jestem w stanie obliczyć kąt α. We wzorze występuje zmienna x, która jest długością pierwszego boku trójkąta. Zmienna ta nie ma jednak żadnego wpływu na obliczenia, ważne jest jedynie aby jej wartość nie była zbyt mała, aby możliwe było zbudowanie trójkąta.




*Rysunek 9. Obliczenia kierunku źródła dźwięku*

d – odległość pomiędzy mikrofonami

$\Delta_s$ – różnica drogi przebytej przez dźwięk pomiędzy 2 mikrofonami

x – losowa wartość > 2d. Nie wpływa na wynik obliczeń

$v_s$ – prędkość dźwięku w powietrzu = 343 m/s

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

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

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

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

Kąt liczę od punktu prostopadłego do macierzy mikrofonów więc wynik muszę odpowiednio przesunąć.

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

## 7.4. Język programowania

Program do obliczania kierunku źródła dźwięku został napisany w języku Matlab, ponieważ posiada on bogate zestawy funkcji i bibliotek, które są niezbędne dla realizacji projektu. Korzystanie z wbudowanych rozwiązań pozwala na zwiększenie pewności, że program będzie działał poprawnie, bez konieczności instalowania dodatkowych bibliotek czy narzędzi w innych językach programowania.

## 7.5. Schemat działania programu

Program widoczny na [Listing 2] najpierw tworzy zmienne częstotliwość próbkowania, prędkość dźwięku w powietrzu, bufor do przechowywania wyników oraz inicjalizuje filtr cyfrowy. Następnie komunikuje się z portem szeregowym i konfiguruje przerwanie które uruchamia główną funkcję programu za każdym razem gdy Na porcie USB pojawi się 6600 nowych próbek.

```
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*

Po uruchomieniu przerwania „callbackFcn” znajdującego się na [Listing 3] następuje pobranie danych z portu szeregowego oraz rozdzielenie ich do 3 osobnych kanałów. Kolejno wykonywana jest filtracja kanałów oraz ich analiza widmowa. Gdy zostanie wykryty sygnał o wystarczająco wysokiej amplitudzie aby z łatwością odróżnić go od dźwięków z otoczenia rozpoczyna się część odpowiedzialna za obliczenia.

```
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. Pobranie danych oraz detekcja dźwięku.*

W pierwszym etapie obliczeń dokonywana jest identyfikacja przesunięć fazowych pomiędzy mikrofonami 1 - 2 oraz 2 – 3. Na [Listing 4] znajdują się instrukcje warunkowe które określają przesunięcie pomiędzy fazami 1 – 2. Analogiczne instrukcje są stosowane również dla różnicy faz 2 – 3.

```
%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. Identyfikacja przesunięcia fazowego pomiędzy dwoma mikrofonami.*

Na tej podstawie w kodzie znajdującym się na [Listing 5] obliczam różnicę w pokonanym dystansie pomiędzy mikrofonami 1 - 3. Korzystając z tych danych obliczam cosinus interesującego mnie kąta oraz koryguję go tak aby był on prezentowany względem linii prostopadłej do matrycy mikrofonów.

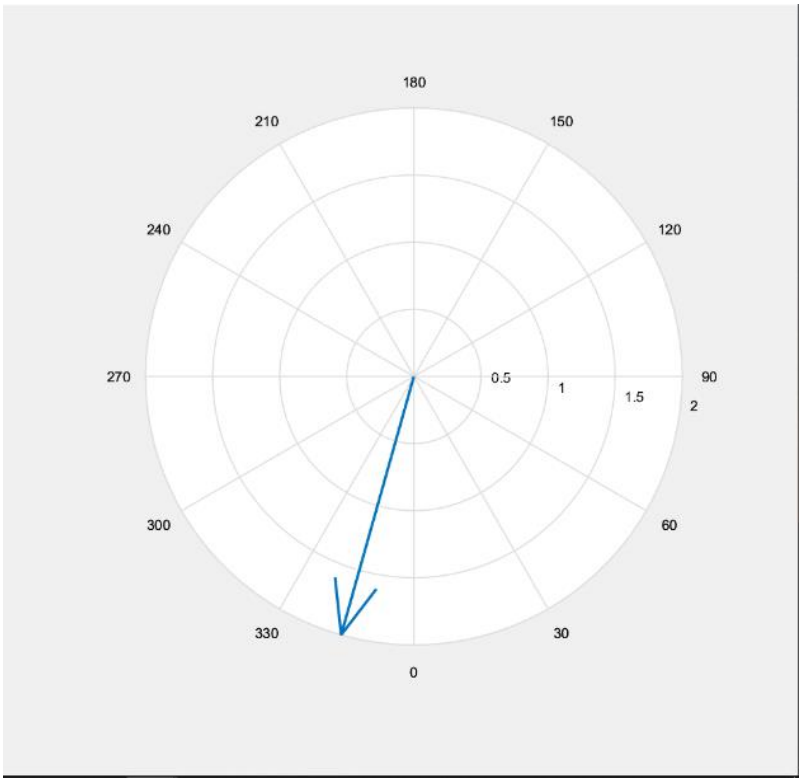
```
% Calculating the distance difference between microphones 1 and 3:
distance_difference = (speed_of_sound/ch3_frequency * phase_difference_1_3/360);
d = 2*0.0127; %distance between 2 microphones
x = 100;  %random value

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. Obliczenia kierunku źródła dźwięku.*

Matlab nie jest w stanie nadążyć aby wyświetlać kąt w czasie rzeczywistym. Z tego powodu za pomocą instrukcji warunkowej na [Listing 6] po zapełnieniu bufora do rozmiaru 20 wyświetlam tylko średnią arytmetyczną z ostatnich pomiarów. Oprócz płynności wyświetlania kąta daje mi to też możliwość dodatkowego zmniejszenia błędu. Kąt wyświetlam w postaci graficznej tak jak na [Rysunek 10] jako strzałka która sygnalizuje kierunek źródła dźwięku.

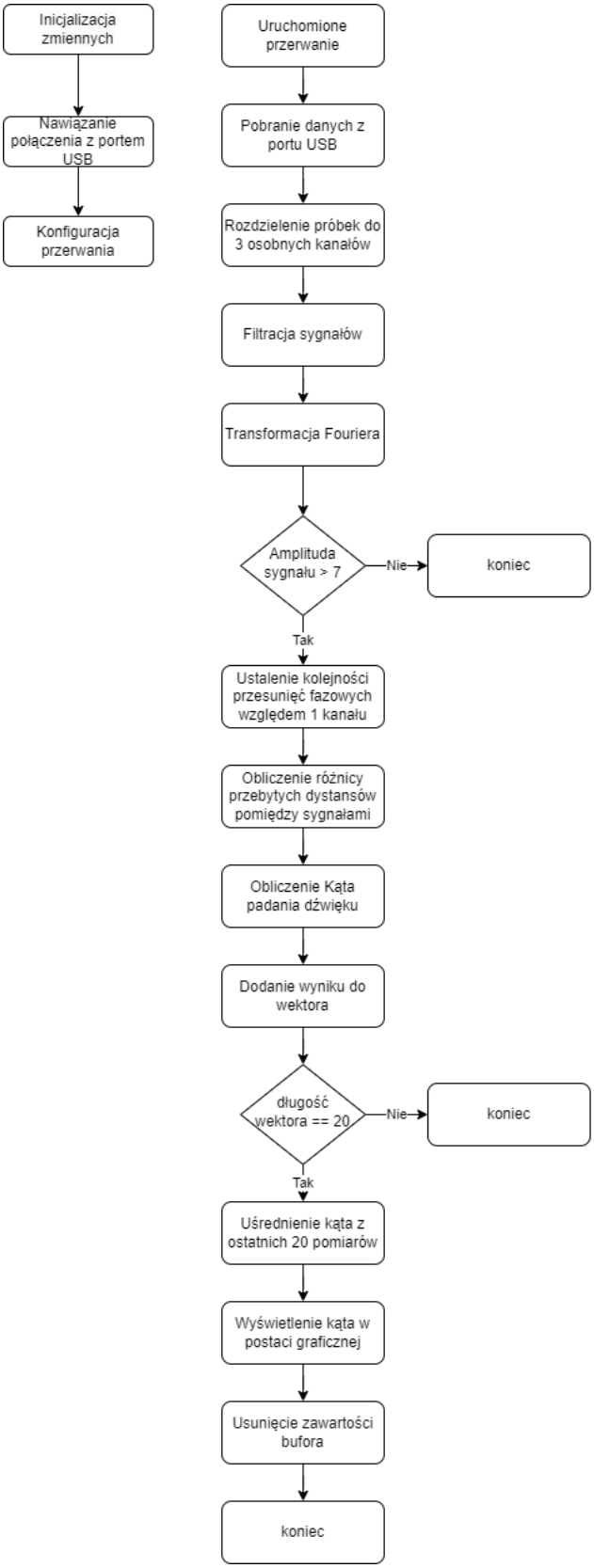
```
% Presenting average angle from last 20 measurements:
if length(angles_of_arrival) >= 20
    disp(["angle:", mean(angles_of_arrival)]);
    DOADisplay(mean(angles_of_arrival));
    angles_of_arrival = [];
end
```

*Listing 6. Wyświetlanie kierunku źródła dźwięku*




*Rysunek 10. Reprezentacja graficzna kierunku źródła dźwięku*

Cały proces przedstawia schemat blokowy na [Rysunek 11].




*Rysunek 11. Schemat działania programu*

# Rozdział 8

# 8. Testy

Testy przeprowadzałem za pomocą głośnika o mocy 10W. Po wstępnych testach okazało się że zakres w którym jestem w stanie wykryć kąt wynosi od $-60^0$ do $60^0$. Powyżej tych zakresów różnica drogi pokonanej wynosi więcej niż odległość pomiędzy mikrofonami. W takim przypadku kosinus kąta jest większy od jedynki co uniemożliwia poprawne obliczenie kierunku źródła dźwięku. W zakresie poprawnej pracy urządzenia przeprowadziłem testy dla kilku częstotliwości poczynając od 2 kHz a kończąc na 13 kHz. Wyższych częstotliwości nie jestem w stanie wykryć ze względu na odległość pomiędzy mikrofonami. Testy były przeprowadzane w odległościach w przedziale 250 mm do 350 mm. Możliwa jest jednak poprawna praca urządzenia także z dalszej odległości. Zasięg urządzenia zależy głównie od mocy nadawania badanego dźwięku. Za pomocą użytego w testach głośnika wyniki były poprawne do odległości około 2 m. Poniżej w [Tabela 1] znajdują się wyniki z przeprowadzonych testów

| **Kąt rzeczywisty** | **Kąt zmierzony:** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **f [kHz]** | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| -60 | -62 | -67 | -59 | -67 | -63 | -57 | -56 | -65 | -63 | -57 | -65 | -55 |
| -50 | -45 | -58 | -47 | -47 | -44 | -47 | -54 | -58 | -56 | -50 | -51 | -44 |
| -40 | -42 | -37 | -36 | -41 | -40 | -34 | -44 | -36 | -48 | -39 | -37 | -35 |
| -30 | -23 | -34 | -22 | -29 | -22 | -20 | -26 | -30 | -30 | -22 | -29 | -24 |
| -20 | -12 | -14 | -18 | -13 | -15 | -16 | -22 | -19 | -23 | -13 | -12 | -16 |
| -10 | -4 | -7 | -4 | -7 | -7 | -7 | -6 | -6 | -15 | -3 | -8 | -6 |
| 0 | 7 | 2 | 5 | 5 | 3 | 3 | 4 | 8 | 2 | 1 | 1 | 6 |
| 10 | 16 | 13 | 20 | 18 | 17 | 17 | 15 | 16 | 8 | 15 | 11 | 15 |
| 20 | 11 | 19 | 16 | 26 | 25 | 22 | 23 | 22 | 17 | 27 | 23 | 25 |
| 30 | 32 | 30 | 32 | 45 | 33 | 35 | 32 | 30 | 32 | 45 | 33 | 35 |
| 40 | 39 | 38 | 43 | 56 | 40 | 44 | 39 | 38 | 43 | 56 | 40 | 44 |
| 50 | 54 | 71 | 64 | 58 | 64 | 50 | 54 | 71 | 64 | 58 | 64 | 50 |
| 60 | 55 | 42 | 57 | 67 | 59 | 62 | 55 | 42 | 57 | 67 | 59 | 62 |
| **Średni błąd pomiaru:** | 4,9 | 6 | 5 | 6.7 | 4.5 | 4 | 3.8 | 4.5 | 2.9 | 4.5 | 2.9 | 4.8 |

*Tabela 1. Wyniki testów 1*

Analizując wyniki przeprowadzonych testów, zauważyłem, że dokładność pomiaru kąta nieco rośnie wraz ze wzrostem częstotliwości dźwięku. W przypadku niższych częstotliwości pod pewnymi kątami błąd mierzenia może sięgać nawet $20^0$, natomiast powyżej 7 kHz błąd ten nigdy nie przekracza $10^0$. Możliwym wyjaśnieniem tego zjawiska jest fakt, że pomiar nie został przeprowadzony w pomieszczeniu dźwiękoszczelnym, co pozwala na nakładanie się dźwięków o tej samej częstotliwości z otoczenia. Niższe częstotliwości charakteryzują się większą amplitudą w warunkach normalnych niż wyższe, co przekłada się na większy wpływ na wartość przesunięcia fazowego fali.

Innym czynnikiem, który może mieć wpływ na dokładność pomiaru, jest odbijanie się dźwięku od ścian. Przeprowadzając testy w pomieszczeniu o niedużych rozmiarach, dźwięk generowany przez głośnik może łatwo odbijać się od ścian i trafiać do mikrofonów inną drogą, co skutkuje innym przesunięciem fazowym niż sygnał, który dociera bezpośrednio.

# Rozdział 9

# 9. Podsumowanie

W ramach projektu zrealizowano prototypowy układ pomiarowy składający się z trzech mikrofonów oraz mikrokontrolera. Zastosowano komunikację z komputerem za pośrednictwem portu USB, oraz oprogramowanie napisane w środowisku Matlab, pozwalające na odbiór danych w czasie rzeczywistym, obliczenie kierunku źródła dźwięku o największej amplitudzie oraz prezentację wyniku w formie graficznej. Niestety, okazało się, że zakres pomiaru jest mniejszy niż pierwotnie założono. Ostatecznie udało się osiągnąć zasięg około $120^0$, a przy użyciu głośnika o mocy 10W, zasięg działania bez dużego błędu pomiaru wyniósł około 2 metrów.
Zakres częstotliwości, który jest możliwy do wykrycia, mieści się w przedziale od 2 kHz do 13 kHz.

Projekt ten spełnia założony na początku pracy cel lecz możliwe są jeszcze usprawnienia. Powinienem spróbować zaimplementować bardziej złożone algorytmy wymienione w 4 rozdziale pracy takie jak ESPRIT, MUSIC lub SRP-PHAT i porównać ich dokładność i szybkość działania z obecnie zastosowanym rozwiązaniem.

Możliwe było by również napisanie algorytmu który wykrywa nie tylko sygnał o największej amplitudzie lecz dowolną ich ilość w zależności od potrzeb.

Aby ułatwić konfigurację oprogramowania mógłbym stworzyć w tym celu dedykowany interfejs graficzny. Pozwalał by on na ustawianie zakresu częstotliwości których nasłuchiwać ma urządzenie, definiować minimalny poziom amplitudy od którego następowała by detekcja.

Aby urządzenie było bardziej kompaktowe i wytrzymałe konieczne było by zaprojektowanie dedykowanej płytki PCB. Dzięki temu nie miał bym też na sobie ograniczeń związanych z wielkością użytych w projekcie modułów mikrofonów MEMS. Umożliwiło by mi to ustawienie ich zdecydowanie bliżej siebie niż 12.7 mm dzięki czemu mógłbym wykrywać o wiele wyższe częstotliwości niż obecnie.

# Bibliografia


[1] Strona internetowa: https://en.wikipedia.org/wiki/Direction_of_arrival (dostęp dnia 27.01.2023).

[2] Strona internetowa: https://en.wikipedia.org/wiki/Fast_Fourier_transform (dostęp dnia 27.01.2023).

[3] Strona internetowa: https://en.wikipedia.org/wiki/Estimation_of_signal_parameters_via_rotational_invariance_techniques (dostęp dnia 27.01.2023).

[4] Strona internetowa: https://en.wikipedia.org/wiki/MUSIC_(algorithm) (dostęp dnia 27.01.2023).

[5] Storna internetowa: https://en.wikipedia.org/wiki/Steered-Response_Power_Phase_Transform (dostęp dnia 27.01.2023).

[6] Strona internetowa: https://pl.wikipedia.org/wiki/Direct_Memory_Access (dostęp dnia 27.01.2023).

[7] “Direction Of Arrival Estimation using Microphone Array” autorstwa C Junu Jahana, MS Sinith, PP Lalu, 19 stycznia 2022.

[8] Strona internetowa: https://pl.wikipedia.org/wiki/STM32

(dostęp dnia 27.01.2023).

[9] Strona internetowa: https://pl.wikipedia.org/wiki/Aliasing_(przetwarzanie_sygna%C5%82%C3%B3w) (dostęp dnia 27.01.2023).

[10] Strona internetowa: https://qph.cf2.quoracdn.net/main-qimg-a0b12071ca95c05006b3510d19ca2eb8-lq (dostęp dnia 27.01.2023).

[11] Strona internetowa: https://www.st.com/bin/ecommerce/api/image.PF259875.en.feature-description-include-personalized-no-cpn-medium.jpg (dostęp dnia 27.01.2023).

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

[13] Dokumentacja mikrokontrolera STM32F103RBT6:

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

# Spis rysunków



# Spis tabel



# Spis listingów