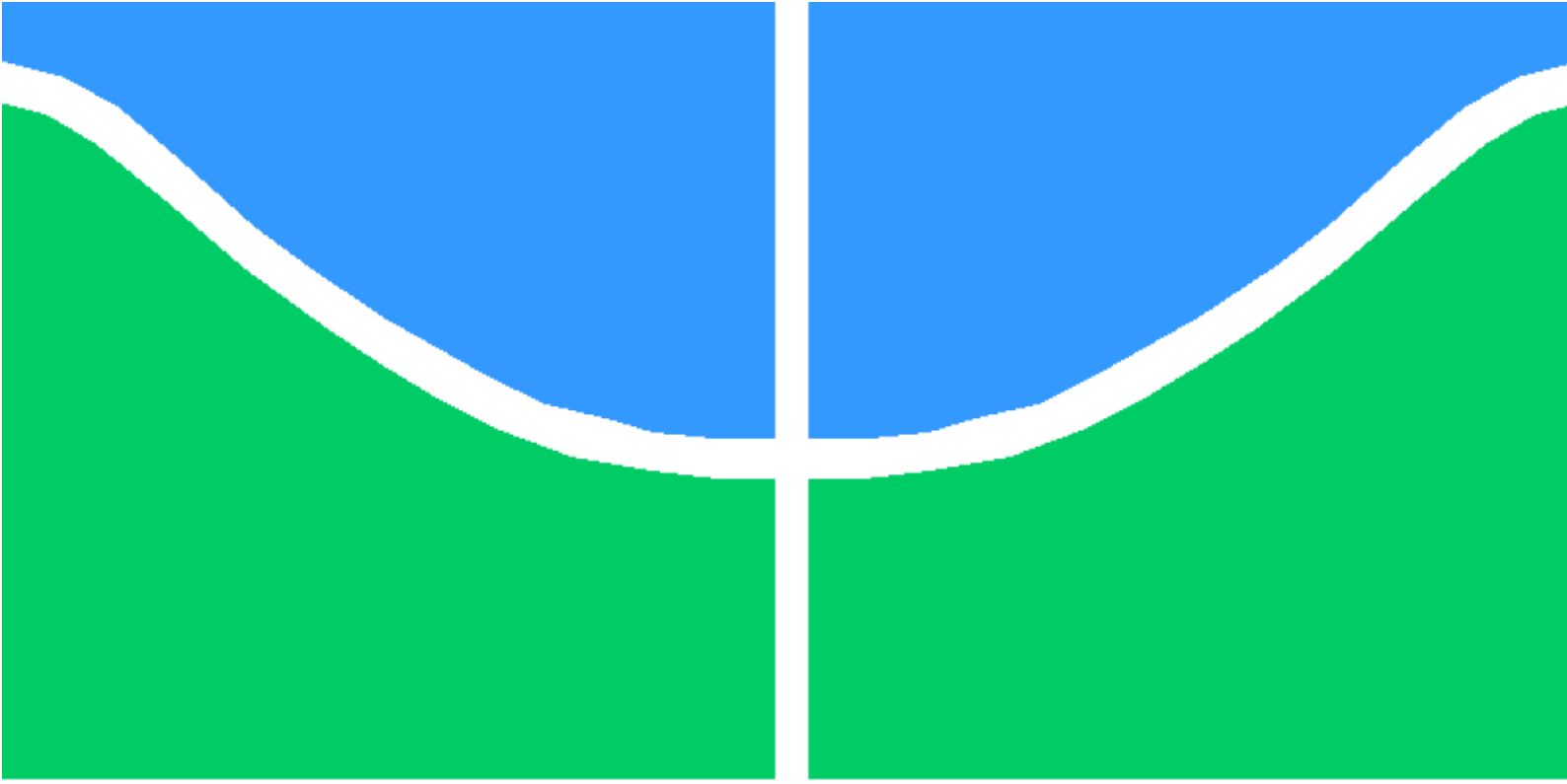

TRABALHO DE CONCLUSÃO DE CURSO

PLATAFORMA MÓVEL GUIADA PELO SOM

Autor: André de Medeiros Araújo
Orientador: Prof. Ricardo Zelenovsky

Brasília, novembro de 2018.

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PLATAFORMA MÓVEL GUIADA PELO SOM

ANDRÉ DE MEDEIROS ARAÚJO

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE ENGENHEIRO DE REDES DE COMUNICAÇÕES.

ORIENTADOR: RICARDO ZELENOVSKY

FICHA CATALOGRÁFICA

ARAÚJO, ANDRÉ DE MEDEIROS

Plataforma móvel guiada pelo som [Distrito Federal], 2018.

Trabalho de Conclusão de Curso – Universidade de Brasília, Faculdade de Tecnologia

Departamento de Engenharia Elétrica. Orientação: Prof. Dr. Ricardo Zelenovsky.

1. Plataforma embarcada guiada pelo som

2. Orientação da plataforma

3. Estimação DOA

I. ENE/FT/UnB II.

REFERÊNCIA BIBLIOGRÁFICA

ARAÚJO, ANDRÉ DE MEDEIROS (2018). Plataforma móvel guiada pelo som. Trabalho de Conclusão de Curso em Engenharia de Redes de Comunicação, 2018, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 52p.

AGRADECIMENTOS

Agradeço aos meus familiares por sempre me apoiarem em todas as fases da minha vida, principalmente durante a formação acadêmica. À minha mãe por me tranquilizar nos momentos de dificuldade, ao meu pai Geraldo por sempre me ajudar de todas as formas possíveis, à minha irmã Ana Clara por me guiar em momentos difíceis.

Agradeço também à minha namorada Daiana por sempre acreditar em mim, pela compreensão e pelo apoio fornecido durante essa jornada.

Ao meu orientador, Prof. Ricardo Zelenovsky, por toda a ajuda para a realização desse projeto e pelos momentos em aula, que me ensinou e me fez gostar das matérias.

Aos professores Alexandre Romariz e Eduardo Peixoto por aceitarem o convite para participar da banca de avaliação do projeto.

Ao colega de curso Matheus por ter me ajudado na resolução de problemas encontrados durante a realização deste projeto.

Ao meu primo Danilo por ser paciente e ajudar na correção e apresentação deste trabalho.

RESUMO

Este trabalho aborda a aplicação de um método de baixo custo computacional para estimar a direção de chegada (DOA) de um sinal sonoro em um ambiente fechado e se locomover em direção à fonte.

Foram utilizados como base alguns trabalhos anteriores orientados pelo professor Ricardo Zelenovksy, como o "Estimação de Direção de Chegada de Sinais Sonoros Utilizando Arranjo de Sensores" [13], que iniciou o estudo com um arranjo linear de microfones, e outros trabalhos que utilizam uma plataforma móvel com detecção de obstáculos [14] [15].

Para a captação dos sinais são utilizados dois microfones de eletreto e pré-amplificados construídos pelo professor Ricardo Zelenovsky com a ajuda de alguns alunos da Universidade de Brasília. Para se identificar a posição da fonte sonora é utilizado um cálculo do atraso temporal entre os sinais captados, medindo o erro quadrático médio entre as amostras que são deslocadas no tempo.

Para orientação da direção é utilizado o giroscópio do módulo MPU-6050 para que possa direcionar a plataforma móvel para a fonte de acordo com os cálculos efetuados. Para a movimentação da plataforma foram usados quatro motores DC, um para cada roda, podendo se movimentar para frente, para trás o girar sobre o próprio eixo.

O projeto todo foi programado na plataforma Arduino Due, em sua própria IDE, e o acionamento dos motores foi feito através da *motor shield* versão 2 da *Adafruit.* Os resultados obtidos com a realização deste projeto foram satisfatórios, onde a plataforma móvel se direciona para a fonte do sinal sonoro do lado esquerdo do arranjo de microfones.

ABSTRACT

This work deals with the application of a low cost computational method to evaluate the direction of arrival (DOA) of a sound signal in a closed environment and to move towards the source.

Some previous works guided by Professor Zelenovksy were used as base, such as the "Estimation of Direction of Arrival of Sound Signals Using Sensor Arrangement" [13], which began the study with a linear arrangement of microphones, and other works that use a platform with obstacle detection [14] [15].

Two electret and preamplified microphones built by Professor Ricardo Zelenovsky with the help of some students from the University of Brasilia are used to capture the signals. To identify the position of the sound source, a temporal delay calculation between the captured signals is used, measuring the mean square error between the samples that are displaced in time.

The gyroscope of the MPU-6050 module is used for directing the steering so that it can direct the mobile platform to the source according to the calculations made. For the movement of the platform four DC motors were used, one for each wheel, being able to move forward, back or rotate on the axis itself.

The project was programmed on the Arduino platform, in its own IDE, and the drive of the motors was done through the motor shield version 2 of Adafruit. The results obtained with the realization of this project were satisfactory, where the mobile platform is directed to the sound source on the left side of the microphone array.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

Símbolos Gregos

θ	Azimute de incidência	[rad]
φ	Defasagem angular	[rad]
λ	Comprimento de onda	[m]

Outros Símbolos

d	Distância entre elementos em um arranjo de sensores	[m]
v	Velocidade da onda	[m/s]
F	Frequência da onda	[Hz]
T	Tempo	[s]
f_{ams}	Frequência de amostragem do sinal	[Hz]
τ	Atraso entre sinais de tempo contínuo	[s]
k	Atraso em tempo discreto	

Siglas

ADC	*Analogic to digital converter*
AWGN	*Additive White Gaussian Noise*
A/D	Analógico para digital
DOA	Direction of arrival
FFT	*Fast Fourier Transform*
IIR	*Infinite Impulse Response*
FIR	*Finite Impulse Response*
MATLAB®	Marca registrada da MathWorks, Inc.
SNR	*Signal – Noise Ratio*
SPI	*Serial Peripheral Interface*

1 AMBIENTAÇÃO

1.1 Motivação:

O som é um sinal resultante de uma vibração que se propaga através de ondas longitudinais, através de diferentes meios. Essa forma de propagação é importante inclusive para a comunicação entre os seres vivos, o que possibilita uma reação ao mundo em sua volta permitindo uma maior percepção do espaço. Diferentes animais utilizam os sinais sonoros para poder se locomover, através da ecolocalização, podendo detectar o lugar e a distância de obstáculos utilizando ultrassom emitido pelo próprio animal, como no caso de morcegos e golfinhos. Já o ser humano possui apenas uma percepção auditiva que permite identificar a origem e a direção de um sinal sonoro, e com os avanços da tecnologia, essa percepção passou a ser utilizada em máquinas, que puderam identificar a localização de um sinal sonoro de várias formas, sendo utilizadas em diferentes tarefas.

O sinal sonoro se trata de um sinal analógico e para que uma máquina possa interpretar esse sinal é conveniente digitalizá-lo. Com a ajuda de sensores esse processo pode ser iniciado, pois estes sensores nos permitem captar, armazenar e processar sinais sonoros. Depois de captado, o sinal analógico é transformado em números por um conversor analógico/digital. Atualmente já existem várias formas de melhorar a captação e conversão desse sinal por meio de filtragem, limitando a banda do sinal. Quanto melhor for a capacidade de processamento da máquina, mais eficiente será essa digitalização.

Hoje em dia, com a criação de robôs e a implementação de inteligência artificial, é cada vez mais necessário que as máquinas entendam o mundo à sua volta, e tais sistemas sensoriais ajudam nesse entendimento com a adição de câmeras, microfones, entre outros sensores [1]. Robôs com esse tipo de entendimento facilitam várias tarefas exercidas pelo homem. No caso de um robô que possa localizar a origem de um sinal sonoro, pode ser utilizado para ajuda em resgates no meio de escombros, ou lugares onde não se tem muita visão, e o som se torna uma forma de orientação.

Este trabalho procura desenvolver um sistema embarcado em uma plataforma móvel de baixo custo, que seja capaz de detectar uma fonte sonora e estimar a direção de sua localização em um ambiente fechado. Na literatura científica, há varias formas de se estimar a direção de chegada de um sinal, como os algoritmos CAPON, MUSIC, DS, ESPIRIT e SPR-PHAT [2]. Este trabalho é a continuação do projeto “Estimação de Direção de Chegada de Sinais Sonoros Utilizando Arranjo de Sensores” [13] onde foi iniciado o desenvolvimento de

algoritmos de processamento de forma a exigir menor demanda computacional, adequando a parte de processamento a um microcontrolador acessível. O objetivo final desse trabalho é que a plataforma móvel, após adquirir o sinal sonoro e estimar a sua direção de chegada, possa alterar sua posição com relação à fonte sonora.

1.2 Organização:

O Capítulo 2 apresenta o embasamento teórico deste trabalho, abordando o arranjo linear de microfones e os conceitos-chave que serão utilizados em capítulos posteriores.

O Capítulo 3 aborda o hardware utilizado no projeto, apresentando uma explicação sobre o que foi utilizado para alcançar o objetivo final, como sensores, processador, memória externa e módulos eletrônicos.

O Capítulo 4 trata do algoritmo utilizado no projeto, explica como foi feito para se controlar a plataforma e detalha o método de estimação de direção de chegada – do inglês *DOA*. Este capítulo foi dividido em seções, abordando as etapas do processo. Discute-se também o efeito de limitações de hardware no processo de estimação.

O Capítulo 5 apresenta os resultados de ensaios realizados e as simulações feitas com a ajuda do MATLAB para verificar o desempenho do método proposto. O capítulo também apresenta algumas limitações encontradas.

Por fim, o capítulo 6 apresenta as conclusões obtidas com este projeto proposto. Discute-se também a possibilidade de trabalhos futuros e o que se pode melhorar para tornar o sistema mais eficiente.

2 EMBASAMENTO

2.1 Arranjo de sensores

A estimação da direção de chegada de um sinal sonoro depende da quantidade de microfones utilizados para isso, pois um só não é capaz de extrair as informações para determinar a direção, e depende também da forma em que eles estão dispostos. Utilizando dois ou mais microfones, o sinal captado por cada um sofre atraso, que é a variável importante para que a estimação possa ser feita, tanto em tempo quanto em fase.

Para esse projeto foram abordados microfones dispostos de forma paralela entre eles, espaçados a uma distância 'd' um do outro, alinhados em uma mesma reta, denominado e Arranjo Linear Uniforme. O ângulo de azimute θ corresponde à direção de chegada do sinal, e é formado pela interseção da frente do sinal sonoro $s(t)$ com a reta perpendicular ao eixo em que os microfones estão alinhados. A figura 2.1 demonstra esse arranjo linear dos microfones.

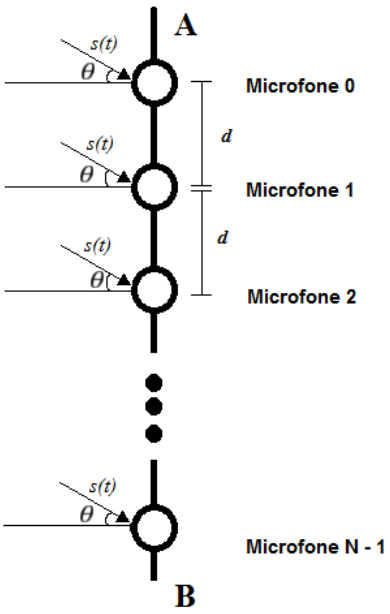

Figura 2-1: Arranjo linear uniforme de microfones para estimação de direção de chegada [13].

Tratando a onda de forma bidimensional, ao incidir sobre o arranjo de microfones, cada um recebe o sinal de forma defasada. Sendo assim o sinal $s(t)$ acaba percorrendo uma distância $d * sen(\theta)$ a mais entre cada microfone, como pode ser visto na figura 2.2.

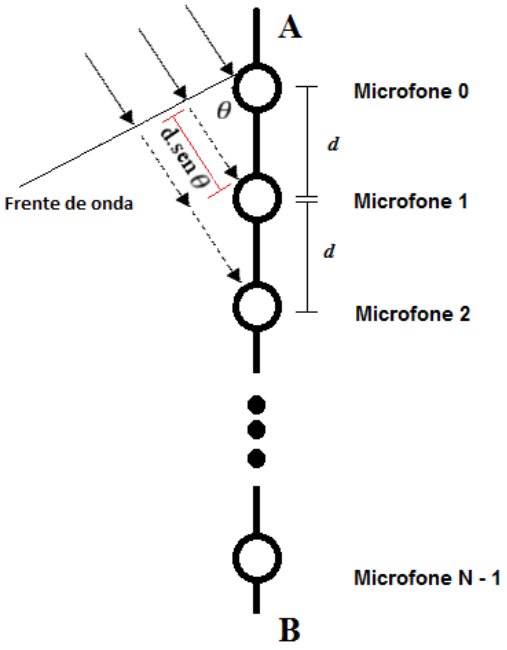

Figura 2-2: Frente de onda bidimensional incidindo sobre um arranjo linear uniforme[13].

O atraso temporal que cada microfone sofre, depende da distância que o sinal percorre a mais com relação ao primeiro microfone do arranjo, e também depende da velocidade da onda. Na equação (2.1) a seguir é demonstrada essa formulação onde 'v' é a velocidade do som, 'd' é a distância entre os microfones, 'θ' é o ângulo de azimute e 'τ' é o atraso temporal.

$$\tau = \frac{d.sen\theta}{v} \tag{2.1}$$

Com o atraso temporal pode ser obtida a defasagem angular, que consiste em multiplicar o atraso temporal pela frequência angular, como mostra a equação (2.2).

$$\varphi = \omega.\tau = 2\pi f.\frac{d.sen\theta}{v} \tag{2.2}$$

Como o som é uma onda, o atraso pode ser demonstrado também de acordo com o comprimento de onda 'λ', mostrando que sinais de frequências diferentes sofrem atrasos distintos. Decompondo a velocidade do som 'v', é obtida a equação (2.3).

$$\varphi = 2\pi.\frac{d.sen\theta}{\lambda} \tag{2.3}$$

Tomando o primeiro microfone como referência, a equação (2.3) é equivalente ao segundo microfone que está a apenas 'd' de distância do primeiro sensor. Para fazer a análise dos outros microfones, a referência continua a mesma, então a defasagem do terceiro microfone será o dobro do segundo, e a defasagem do quarto microfone será o triplo do

segundo, e assim por diante. Esse padrão se deve à distância que o microfone está com relação ao microfone de referência. Dessa forma, esse arranjo pode ser equacionado como uma matriz, onde $x_i\ (t)$ é a saída do i-ésimo microfone, em um total de M sensores, tomando a saída $x_0(t)$ do microfone 0 como referência, dando a equação matricial (2.4).

$$\begin{bmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \\ \vdots \\ x_{M-1} \end{bmatrix} = \begin{bmatrix} 1 \\ e^{-j\varphi} \\ e^{-2j\varphi} \\ \vdots \\ e^{-(M-1)j\varphi} \end{bmatrix} s(t) + \begin{bmatrix} n_0(t) \\ n_1(t) \\ n_2(t) \\ \vdots \\ n_{M-1} \end{bmatrix} \tag{2.4}$$

Em cada microfone, juntamente com o sinal de entrada há o ruído branco $n(t)$ que já faz parte do próprio sistema, e foi adicionado à equação matricial, que só é válida desde que $s(t)$ seja um sinal em banda estreita, sendo $s(t) = b(t)e^{j\omega_0 t}$, onde $b(t)$ é limitado em banda.

2.2 Ambiguidade e distância entre sensores

Para poder determinar a direção de chegada do sinal sonoro com esse arranjo de microfones, o módulo do ângulo do atraso precisa ser menor que 180 graus e aplicando essa condição na equação (2.3), obtém-se a equação (2.5), considerando que o maior valor para um seno é 1.

$$d \leq \frac{\lambda}{2} \tag{2.5}$$

Essa relação é conhecida por ser a versão espacial do Teorema da Amostragem de *Nyquist*, e fornece uma restrição para o espaçamento entre sensores para que a estimação de *DOA* seja feita de forma inequívoca.

Se a frequência característica no espectro da voz humana for de $f = 1kHz$, e com a velocidade do som de $v = 340\ ^{m}/_{s}$, através da equação (2.5) observa-se que uma distância boa para ter entre sensores adjacentes é de $d \leq 17\ cm$. Para este projeto, a distância de espaçamento entre os microfones utilizados foi de 17 cm com o intuito de se obter uma margem de angulação maior do que a margem fornecida utilizando 15 cm, como utilizado em projetos anteriores [13].

2.3 Bandas de frequência

Como foi anteriormente exposto, a equação (2.3) indica que, no arranjo linear de microfones, há uma frequência central 'f_c' concentrando a maior parte da energia do sinal. Se

o sinal for em banda larga, ele vai distribuir a energia em uma diversas faixas de frequências, cada uma delas sofrendo um atraso diferente.

Para esse projeto, a estimação da *DOA* focou na frequência central de um sinal de banda estreita, pois caso contrário não seria possível estimar a *DOA* com o método proposto. Para utilizar sinais sonoros em banda larga é necessário um elevado poder de processamento.

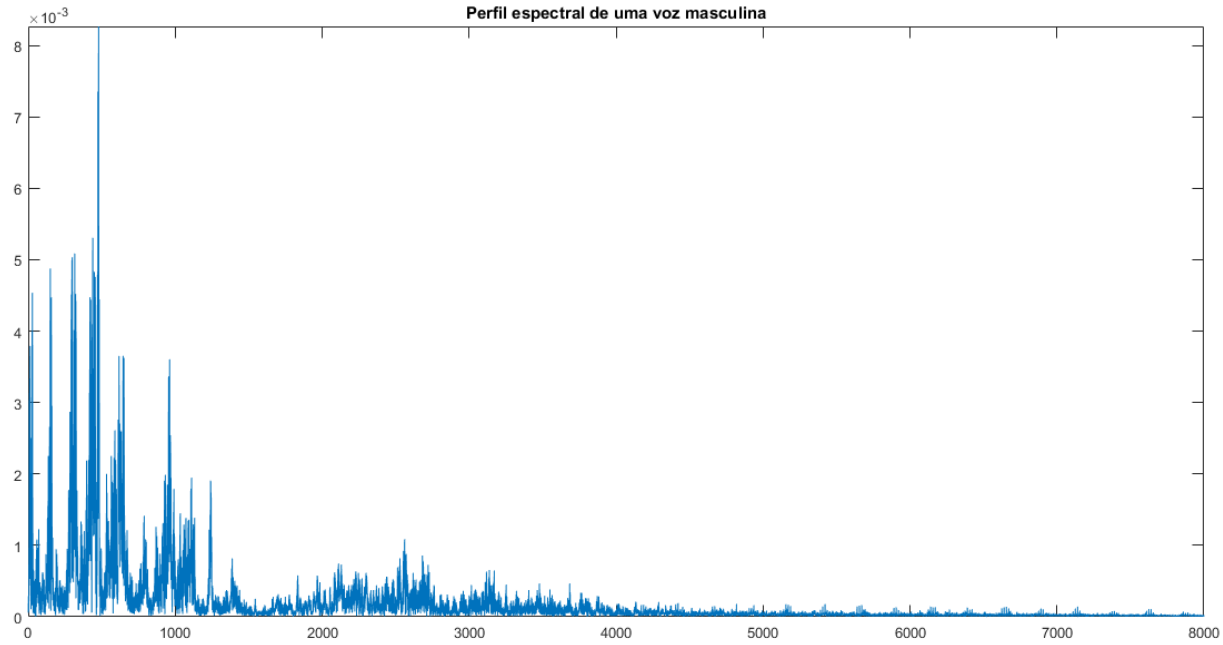

Figura 2 -2-3: Perfil espectral de uma voz masculina [13]

A frequência da voz de cada pessoa varia, principalmente comparando as vozes entre homens e mulheres. Conforme pode ser observado na figura acima, que rata de um espectro em frequência de um sinal em banda larga $s(t)$ amostrado a uma taxa de $f = 16\,kHz$ e filtrado em $f_c = 8\,kHz$, a banda do sinal sonoro varia de $1\,kHz$ até $4\,kHz$. Por esse motivo, a estimação da *DOA* com a voz humana se torna mais difícil de ser calculada.

O assovio, por sua vez, varia muito pouco de pessoa para pessoa, podendo até ser bem parecidos mesmo que seja produzido por pessoas diferentes. A energia desse som está dentro de uma faixa de frequência bem pequena, como pode ser visto na figura 2.4, próxima de uma frequência principal. Dessa forma a estimação da *DOA* se torna mais plausível de acordo com o método proposto nesse projeto.

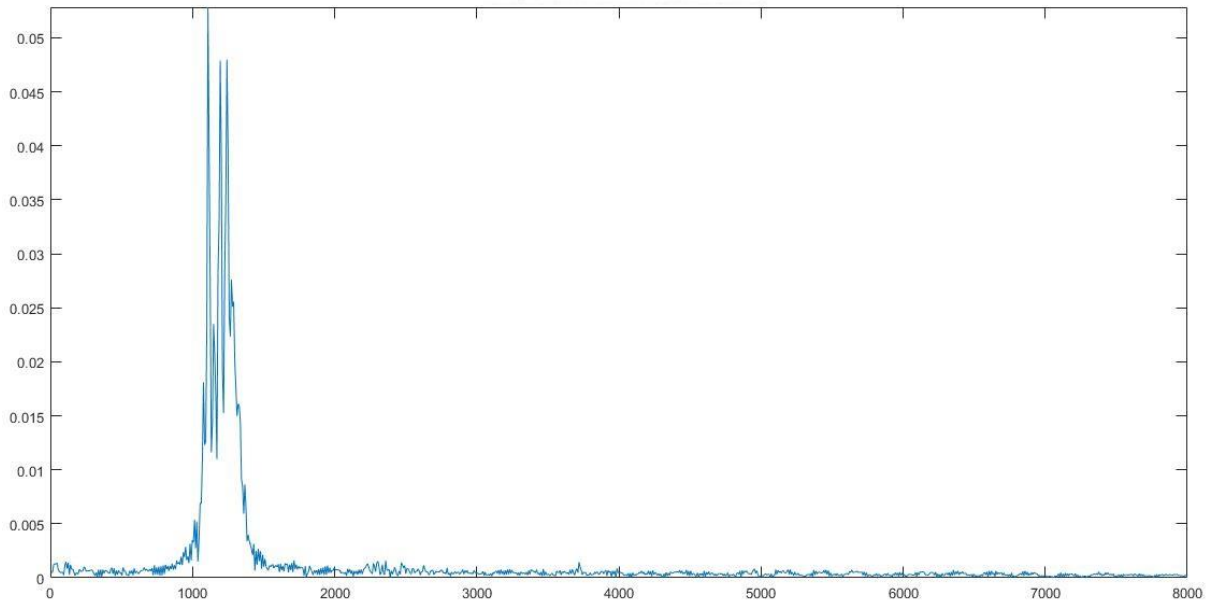

Figura 2-4: Perfil espectral de um assovio [13].

2.4 Filtros digitais

Quando os sinais sonoros são captados pelo microfone, é necessária a implementação de um filtro digital passa baixo, que é usado na digitalização de sinais para limitar a banda e assim evitar o "*aliasing*" e também para reduzir o ruído AWGN presente em todo o espectro. Para projetar esse filtro de forma eficiente, é importante analisar o perfil espectral típico do sinal sonoro do assovio recebido, com esse intuito foi projetado um filtro IIR.

Um filtro digital é caracterizado por uma função de transferência descrita no domínio Z, na forma:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M}}, \tag{2.8}$$

onde $B(z)$ e $A(z)$ são os coeficientes que caracterizam a resposta em frequência do filtro. O que define a ordem é o polinômio de maior ordem entre o numerador e o denominador.

Para projetar o filtro passa baixo foi utilizado o MATLAB, fazendo uso da ferramenta disponível *filterDesigner*. A ferramenta recebe como parâmetros de entrada:

- O tipo da resposta do filtro (Passa-baixa, passa-faixa, entre outras);
- A ordem do filtro;
- O método de design (IIR, FIR);
- A banda passante do sinal,

Com o filtro projetado, os parâmetros A(z) e B(z) da função de transferência H(z) são conhecidos e podem ser aplicados ao sinal por meio de uma rotina programada em C no próprio Arduino.

De acordo com o perfil espectral do assovio, foi programado um filtro IIR – resposta infinita ao impulso, do inglês *Infinite Impulse Response* – Butterworth de 2ª ordem, do tipo passa-baixa e com uma frequência de corte de 3 kHz, como mostra a figura 2.5.

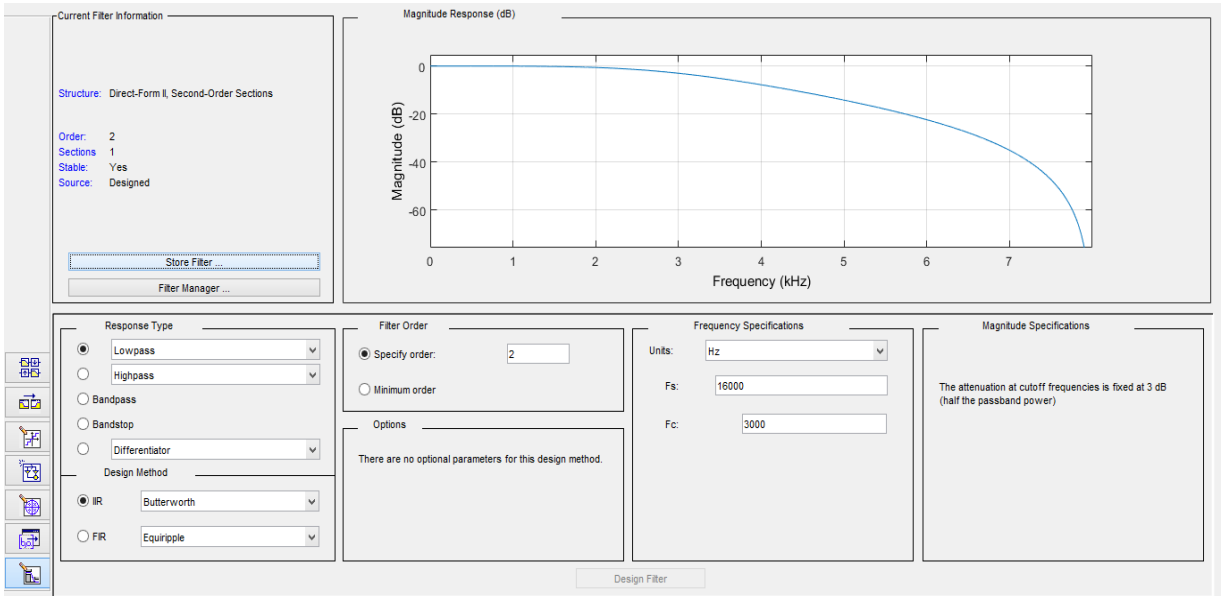

Figura 2-5: Design de filtro IIR em ambiente MATLAB

3 HARDWARE

Neste capítulo é apresentada toda a parte de projeto de hardware do sistema, englobando as etapas de orientação e movimentação da plataforma, e também a aquisição, amplificação, digitalização e processamento do sinal sonoro.

Entre os componentes mecânicos do projeto, destaca-se a plataforma de acrílico que também será apresentada, o que permite dar mobilidade ao projeto. Dentre os eletrônicos, será abordado o Arduino Due, a memória externa, o *motor shield V2* da Adafruit, o microfone e circuito pré-amplificador, o módulo MPU-6050 e os motores que dão movimento à plataforma.

3.1 Plataforma

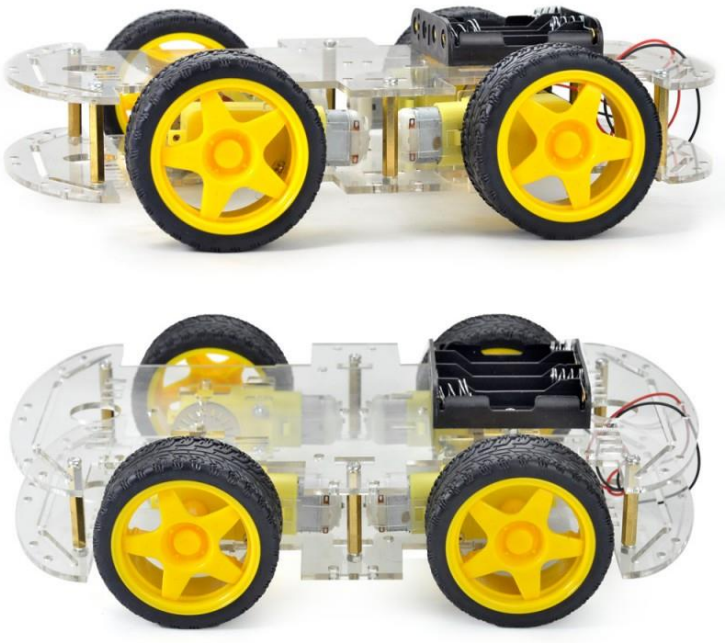

Figura 3-1: Chassi para robô 4WD – Acrílico [14]

A plataforma utilizada para dar mobilidade ao projeto é composta por duas placas de acrílico sobrepostas, presas através de parafusos. Entre as placas existe um vão no qual os motores DC estão acoplados. Na placa superior da plataforma são posicionados os microfones, o Arduino Due juntamente com o *motor shield*, e o módulo MPU-6050 posicionado em uma pequena placa, que é fixada na plataforma através de fita dupla face.

Nessa plataforma há quatro motores DC, cada um responsável por uma roda, permitindo que a plataforma possa se mover para frente ou para trás, girando todas as rodas no mesmo sentido, ou girar para esquerda ou direita, girando as duas rodas do lado direito para um sentido e as duas rodas do lado esquerdo para o sentido oposto.

3.2 Arduino

O Arduino é uma plataforma de desenvolvimento *open source*, que possue diversos modelos, podendo se adequar a diferentes tipos de projetos, o que a tornou bem aceitável no mercado e difundida no mundo. A maioria dos modelos é baseada em microcontroladores da empresa norte-americana Atmel [3]. Já existem diversos módulos acopláveis que são desenvolvidos para essas placas, como o caso do *motor shield* que será abordada mais a frente, que buscam solucionar problemas e aprimorar projetos, além de outros componentes que podem ser utilizados anexos à plataforma, como sensores, antenas, entre outros equipamentos.

O Arduino possui um ambiente de desenvolvimento (IDE) próprio, que oferece algumas bibliotecas facilitando bastante a programação, principalmente na adição de módulos. A linguagem principal utilizada para programação do Arduino é uma forma simplificada de C, embora ainda forneça suporte ao C++.

Inicialmente foi usado o Arduino MEGA 2560 para digitalizar e processar o sinal de interesse, porém nessa fase do projeto foi decidido usar o Arduino Due por possuir uma CPU mais rápida, de 84 MHz, e é baseado em um processador de 32 bits, melhorando então na velocidade de processamento dos sinais processados.

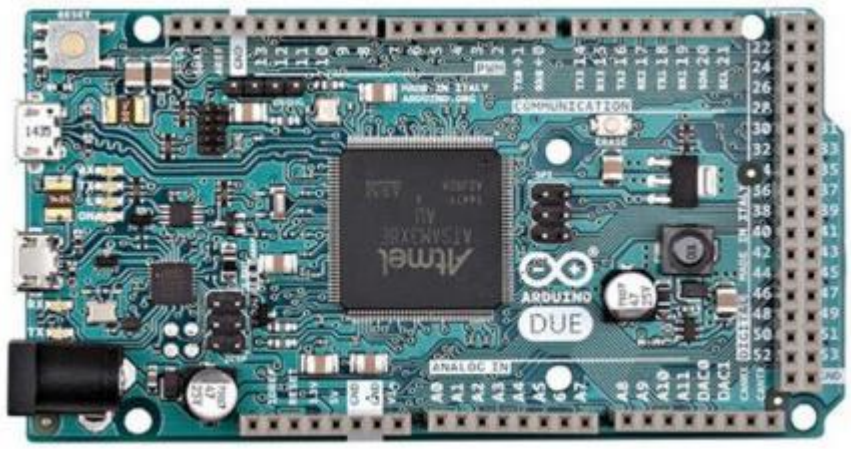

Figura 3-2: Arduino Due [4].

O Arduino Due, mostrado na figura 3.2, é baseado no ARM SAM3X8E de 84 MHz, o primeiro da linha ARM a integrar uma placa Arduino. Essa placa conta com 54 pinos digitais, mostrados na figura 3.3, 512kB de memória Flash, além de 4 portas UART e duas interfaces I²C. Conta também com nove *timers* disponíveis, um *clock* de 84 MHz e o recurso de interrupção externa, presente em todos os pinos digitais [4].

Os pinos A7 e A6, correspondentes ao canal 0 e ao canal 1 do conversor A/D respectivamente, são configurados para receber o sinal que vem do estágio de amplificação do microfone. Quanto maior é a taxa de amostragem, maior será a resolução da estimativa do ângulo θ. Por esse motivo deseja-se o maior valor possível para a frequência de amostragem (f_{ams}), porém a digitalização do sinal se torna ineficiente para taxas maiores do que 16 KHz em cada microfone, inserindo alto nível de ruído no sinal e podendo até causar erros no processo. Devido a esse fator, a taxa escolhida foi de 16 KHz para cada microfone, configurando então o relógio do processador para manter uma taxa de 32 KHz ($2f_{ams}$).

Para que os dados possam ser tratados, é importante que os microfones possam gravar um tempo suficiente para se obter uma amostra relevante do sinal de interesse, e para isso foi usada uma memória externa, que será abordada no tópico 3.3, pois a memória interna do Arduino não seria suficiente para isso.

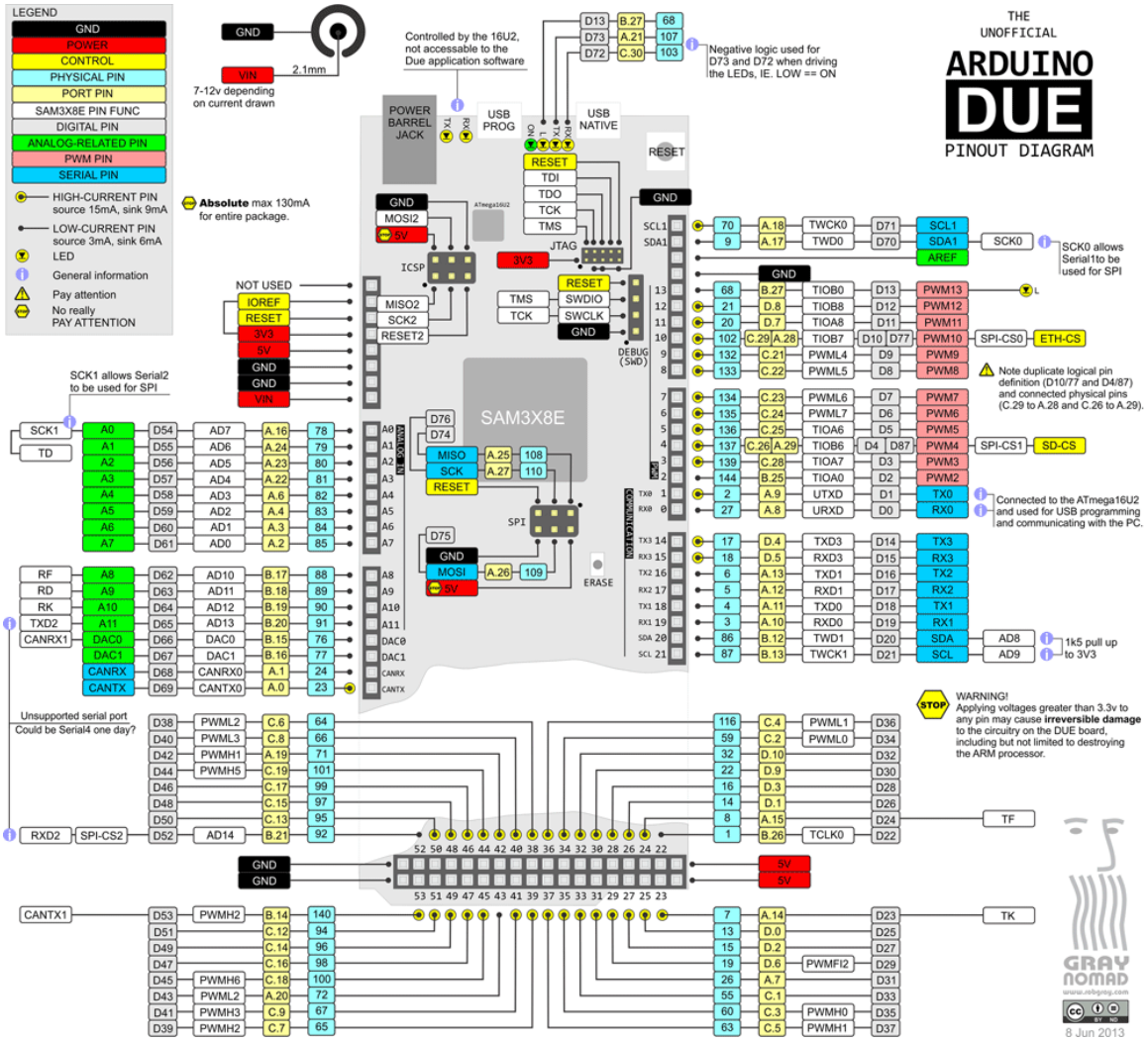

Figura 3-3: Mapa de pinos do Arduino Due [17]

3.3 Memória externa

Para evitar problemas com a memória interna do Arduino, foi utilizada a porta SPI da plataforma para enviar e salvar os dados recém convertidos pelo conversor A/D em uma memória SRAM externa. A memória utilizada foi a 23LC1024, apresentada na figura 3.4, fabricada pela Microchip. Essa memória possui 128 KB de espaço, sendo suficiente para poder armazenar os dados convertidos de dois microfones por aproximadamente 4 segundos, sendo um bom tempo para utilização dos dados.

Figura 3-4: Memória 23LC1024 [15].

A conexão da memória é feita pela pinagem disponível para conexão SPI no Arduino Due, onde os pinos são detalhados na pinagem da figura 3.3 de acordo com as suas funções segundo o *datasheet* da memória. Assim que o dado é recebido pelo microfone é convertido para sinal digital e então escrito na memória. Assim, o sinal de interesse digitalizado fica pronto para ser lido diretamente da memória, o que facilita seu processamento.

O barramento SPI faz uso do conceito Mestre e Escravo, onde o mestre é o que gerencia o barramento, responsável por gerar a linha de sincronismo (SCK) além de inicializar e finalizar as transações, enquanto o escravo só se manifesta quando é acessado. O mestre envia os dados pela linha MOSI (*Master Output Slave Input*) e o escravo envia dados para o mestre pela linha MISO (*Master Input Slave Output*). Dessa forma a comunicação é feita de forma *full-duplex* (bidirecional). No projeto foram feitas algumas funções de escrita e leitura na memória, além de configuração da SPI para facilitar o manuseio do programa.

3.4 Microfone e circuito pré-amplificador:

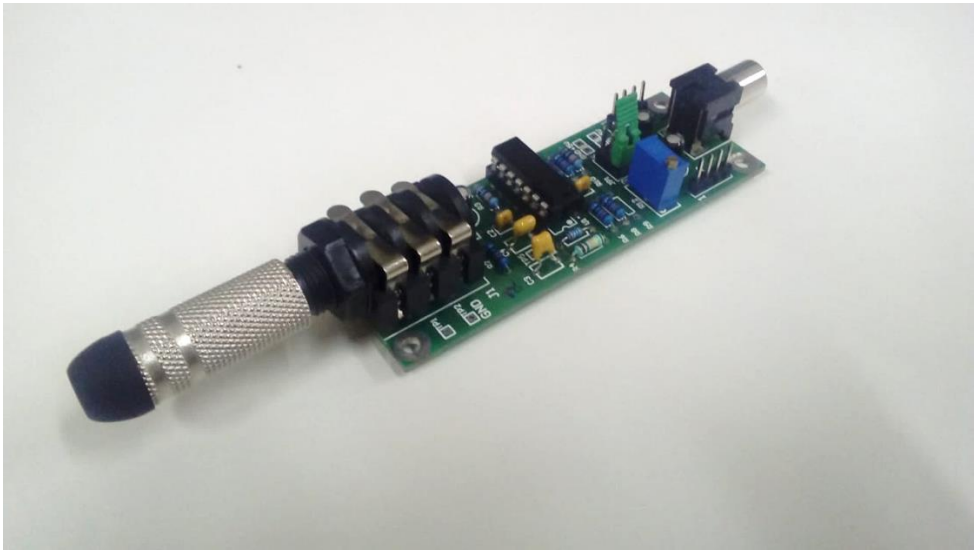

Figura 3-5: Microfone acoplado à placa do circuito utilizado [13].

Neste trabalho, foram utilizados microfones omnidimensionais. Estes microfones foram usados em diversos projetos de detecção da direção de chegada (DOA) e pela disponibilidade foram na plataforma embarcada. A curva de resposta em frequência do microfone é linear para a banda típica de sinais de voz (4kHz) e apresentando SNR de cerca de 62 dB.

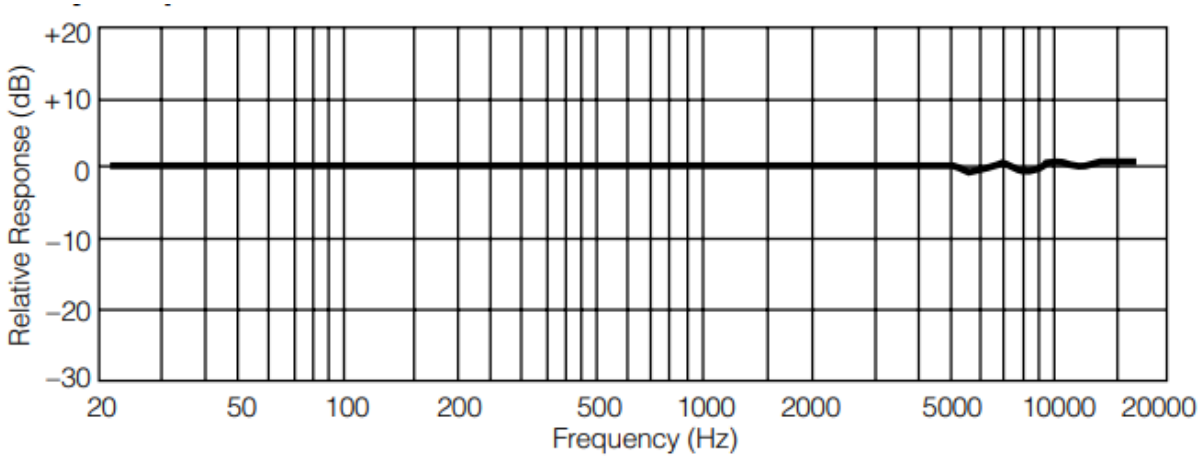

Figura 3-6: Resposta de magnitude versus frequência do microfone de eletreto [13].

O circuito de pré-amplificação do microfone foi construído em uma placa de circuito impresso utilizando os quatro amplificadores operacionais contidos no chip LME49740.

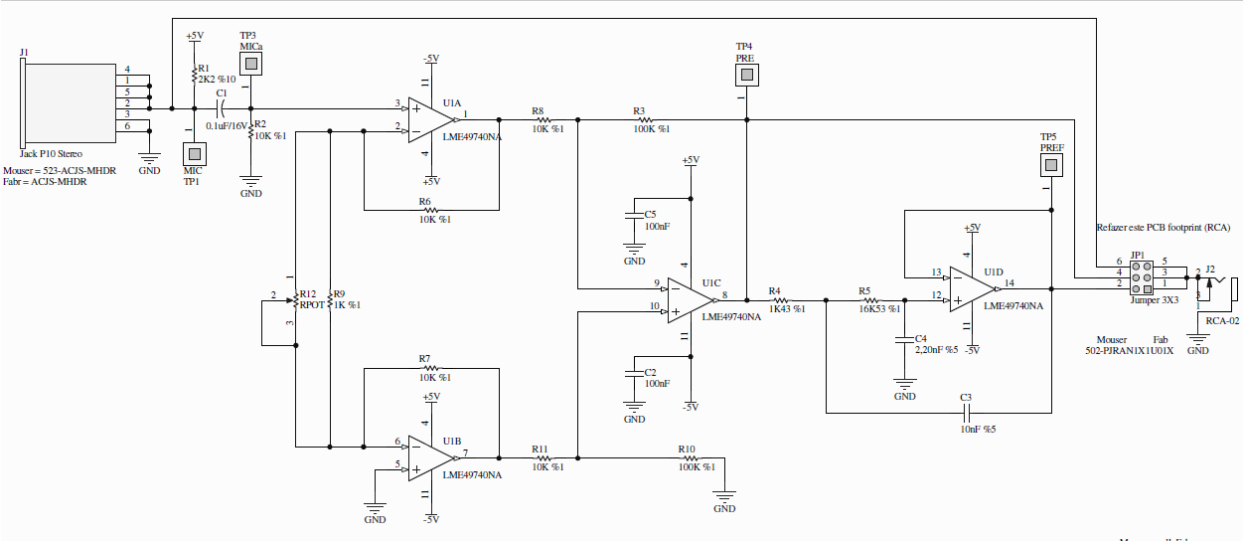

Figura 3-7: Circuito do estágio de pré-amplificação [13].

O circuito apresentado na figura 3.7 é o de um amplificador de instrumentação na sua versão típica. O sinal recebido passa por um estágio inversor e por um estágio não-inversor. Em seguida, é enviado a um comparador, ou amplificador em malha aberta, cuja saída depende da diferença entre os dois estágios anteriores. Depois é aplicado um filtro passa-baixas *Sallen key* de frequência de corte de 4,378 kHz no sinal, seguindo por um buffer de saída. A saída do buffer é conectada à entrada do conversor analógico/digital do Arduino para digitalização do sinal.

O circuito é alimentado por uma fonte externa que, neste projeto, consistiu de um conjunto de duas pilhas de 1,5V, totalizando 3V.

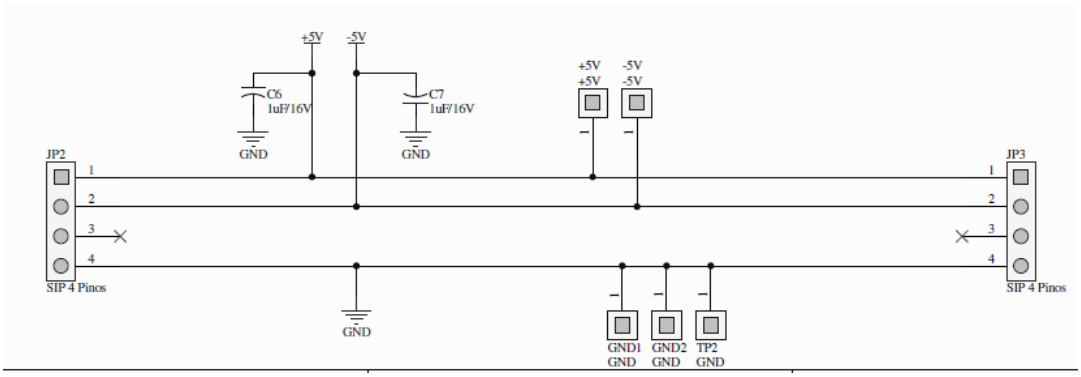

Figura 3-8: Alimentação do circuito [13], onde +5 e -5 foram substituídos por +1,5V e -1,5V.

3.5 Adafruit *Motor Shield* V2:

Com a popularização e o grande uso das placas Arduino algumas empresas independentes começaram a fabricar placas adaptadoras para conectar um Arduino a componentes externos. Esses adaptadores são conhecidos como *Shields* e permitem desempenhar diversas funções. No caso da *shield* utilizada nesse projeto, a *motor shield* V2 da Adafruit, serve para fornecer uma ligação entre o Arduino e o conjunto de quatro motores DC que movimentam a plataforma.

Essa shield é compatível com o Arduino Due, e utiliza comunicação I²C para realizar o controle. Uma explicação mais detalhada sobre o protocolo I²C pode ser encontrada na seção 3.5.1. Uma pequena alteração é necessária na placa da *shield* para ser empregada com essa versão do Arduino, pois ele opera com níveis lógicos em 3,3V, em vez dos 5V comuns aos outros Arduinos. De acordo com o manual do usuário, em um pequeno agrupamento de *pads* na *shield*, identificado como *logic*, deve ser cortada a ligação entre o *pad* do meio e o da direita, identificado como 5V, e com uma gota de solda deve ser ligado com o *pad* da esquerda, identificado com 3,3V.

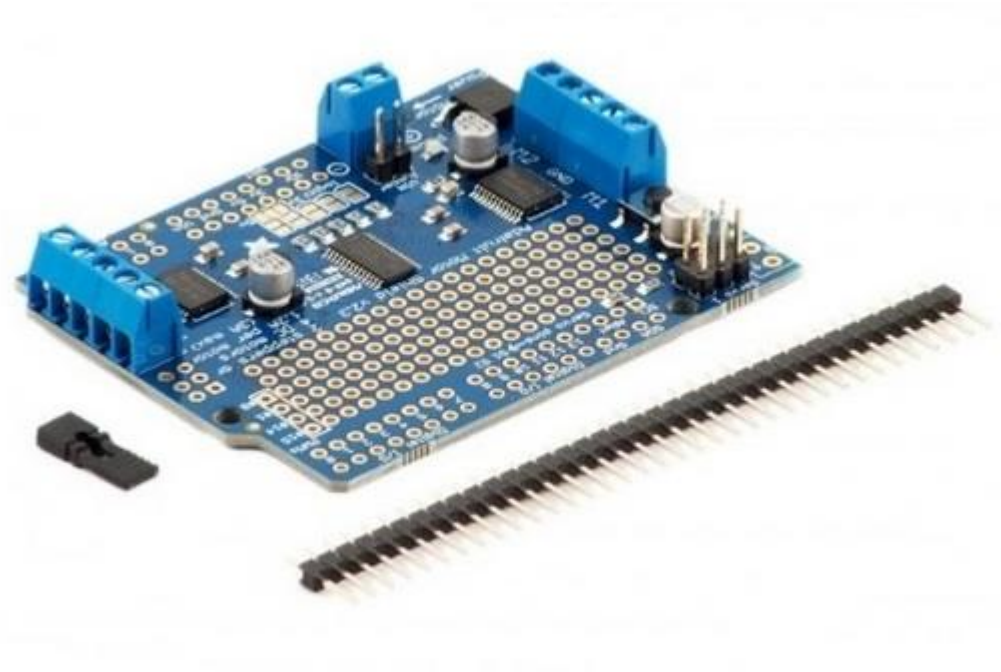

Figura 3-9: Motor Shield V2 da Adafruit [16]

Essa *motor shield* utiliza dois drivers TB6612FNG, cada um com capacidade para alimentar dois motores DC [5], além de um chip PCA9685 (inicialmente usado como um driver de LEDs) que fornece a comunicação I²C e gera os sinais PWM (*Pulse Width Modulation* – Modulação de Largura de Pulso) para movimentar os motores, liberando o

Arduino de se ocupar com essa função[6]. A alimentação dessa placa é feita pelo próprio Arduino, colocando um *jumper* nos pinos indicados no manual da placa.

3.5.1. I²C

O protocolo I²C (*Inter-Integrated Circuit*) foi desenvolvido pela Philips com o intuito de fazer a comunicação com apenas dois pinos em cada dispositivo. Essa comunicação pode ser feita entre diversos dispositivos pois o I²C possui uma topologia de barramento, permitindo que vários "escravos" se conectem a um "mestre" simultaneamente. Esses dois pinos são chamados de *Serial Clock* (SCL) e *Serial Data* (SDA). O SCL contém o sinal de *clock* fornecido pelo dispositivo mestre para que, a cada subida, um novo bit do SDA possa ser lido [7].

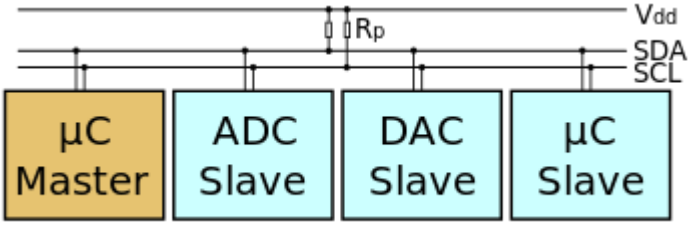

Figura 3-10: Linhas utilizadas pelo protocolo I²C

Toda comunicação I²C é controlada pelo mestre, e para que a comunicação seja iniciada é necessário que esse mestre espere o barramento estar livre. Com essa condição, o mestre gera uma condição de *start*, comunicando aos escravos conectados no barramento que deseja utilizar a linha, e assim os escravos entram em modo de escuta. Após isso, o mestre envia pelo barramento o endereço do escravo com quem deseja se comunicar, junto com um bit que indica se quer transmitir ou receber um dado. O escravo do endereço escolhido deve enviar um sinal de *acknowledge* (ACK), para então a transferência de dados ser iniciada. A cada byte recebido, o receptor deve enviar um sinal de ACK, e ao terminar a comunicação, o mestre gera uma condição de *stop*, encerrando a comunicação.

3.6 MPU-6050:

O MPU-6050 é um dispositivo integrado de 6 eixos que combina 3 eixos de um giroscópio e 3 eixos de um acelerômetro além de um processador digital de movimento (DMP). Atualmente este dispositivo é bastante utilizado em smartphones, tablets, videogames e até em câmeras fotográficas e filmadoras para a detecção da posição, movimento e até na eliminação da trepidação.

Este dispositivo pode trabalhar em escalas diferentes, ajudando principalmente na precisão das medidas. O acelerômetro trabalha nas escalas de +/- 2 g, +/- 4 g, +/- 8 g e +/- 16 g. O giroscópio trabalha nas escalas de+/- 250 graus/s, +/- 500 graus/s, +/- 1000 graus/s e +/- 2000 graus/s. Os valores entregues são de 16 bits com o sinal, varrendo uma faixa de -32.768 até 32.767 [8].

O dispositivo utilizado neste projeto foi o modelo GY-521, que é a placa mais comum utilizada no Brasil. Essa placa pode ser alimentada por 5 V, pois possui um regulador interno de 3.3 V.

Figura 3-11: Dispositivo MPU-6050 modelo GY-521

Esse dispositivo também utiliza o protocolo I²C para se comunicar com o Arduino, da mesma forma que a *motor shield* abordada na seção 3.5.

3.7 Motores DC:

Motores são mecanismos que transformam formas de energia, tais como térmica, elétrica, cinética, em energia mecânica. No caso do motor utilizado nesse projeto, o motor DC (*Direct Current* – corrente contínua) converte energia elétrica em mecânica [9], dando assim mobilidade à plataforma do projeto.

O motor DC consiste na parte do rotor, que é uma parte móvel contendo um enrolamento de armadura e o estator, que é a parte externa e fixa, onde se encontra o enrolamento de campo. O funcionamento desse motor se baseia nas leis do eletromagnetismo. De acordo com Sen (1997) [10]:

> "Uma corrente contínua corre através do enrolamento de campo para produzir um fluxo eletromagnético na máquina. A tensão induzida no enrolamento de armadura é alternada. Um comutador mecânico e um sistema

de escovas funcionam como um retificador e inversor, tornando unidirecional a tensão terminal na armadura.”

Foram utilizados 4 motores desse tipo que, de forma individual, movem as quatro rodas da plataforma, permitindo à plataforma uma rotação em torno do próprio eixo ou então uma movimentação para frente ou para trás. Os motores são alimentados e controlados pela *motor shield* apresentado na seção 3.5.

Figura 3-12: Motor DC e roda utilizados no projeto

4 MONTAGEM DO PROJETO E PROGRAMAÇÃO

O projeto foi iniciado pelo controle de direção da plataforma, e em seguida foi adicionada a programação da estimação de DOA. Juntando os dois algoritmos obtêm-se a plataforma móvel guiada pelo som. Nesse capítulo será feita uma descrição mais detalhada sobre os algoritmos do projeto.

4.1 *Firmware*

Com a alteração do Arduino MEGA utilizado no projeto 'Estimação de direção de chegada de sinal sonoro utilizando arranjo de sensores' [13] para o Arduino Due, algumas modificações no código de captura dos sensores tiveram de ser feitas, principalmente na parte que se refere aos timers utilizados pelo processador no tratamento dos dados. O Arduino Due possui três timers, cada um com três canais, totalizando nove canais, onde cada um pode gerar uma interrupção independente. Baseando-se no manual do processador utilizado, o Cortex M3 da ARM, foi verificado cada bit de configuração dos timers, e assim configurado da forma necessária. Para este projeto foi utilizado o TC0 (Timer Counter 0, canal 0) para trabalhar junto fornecendo o relógio da taxa de aquisição para a IDE com o conversor A/D do processador. Para o controle da plataforma móvel, a interrupção utilizada é externa, gerada pelo MPU-6050 a fim de medir quantos graus a plataforma está girando em tempo real.

4.2 Controle da plataforma móvel

Em projetos anteriores como o "Plataforma móvel com detecção de obstáculos" [11][12], que utilizavam a mesma plataforma móvel, o funcionamento dos motores DC também era gerido pela *motor shield*, porém não havia um controle de rotação para saber quantos graus a plataforma girava. Para ter esse controle nesse projeto foi adicionado um módulo MPU-6050 com o objetivo de usar o giroscópio dele para poder calcular a rotação exata.

O acionamento dos motores para movimentar a plataforma foi feita em rotinas diferentes com a ajuda da *motor shield* que tinham o objetivo de:

- Andar para frente: Todos os motores DC fazem com que as rodas girem no mesmo sentido, fazendo com que a plataforma avance como um todo;
- Parar – a *motor shield* corta a alimentação dos motores, fazendo com que eles parem de rodar. Essa rotina de parar não equivale a um freio;

- Girar para a direita – Os motores do lado direito da plataforma giram as rodas para trás enquanto os motores do lado esquerdo fazem as rodas girarem para frente;
- Girar para a esquerda – Os motores do lado esquerdo da plataforma giram as rodas para trás enquanto os motores do lado direito fazem as rodas girarem para frente;

Além dessas rotinas de movimentação da plataforma, foi feita uma rotina para a configuração inicial da *motor shield*, e uma outra rotina para comandar os motores DC.

Junto com a *motor shield*, o módulo MPU-6050 é utilizado para coordenar o direcionamento da plataforma. Para este trabalho foi configurada a escala de +/- 250 graus/segundo, especificado um filtro com largura de banda de 5 Hz, atraso de 18,6 ms e com taxa de amostragem de 200 Hz.

Para eliminar o erro de offset das medidas foram feitas 500 leituras do dispositivo enquanto parado em uma superfície plana. A média dessas medidas é subtraída das medições feitas, fazendo com que, independente da posição que a plataforma iniciar, as medidas iniciais sempre serão em torno de zero. Para poder determinar quantos graus a plataforma está girando foi utilizada a equação (4.1) com relaciona com Gyr_Z referente aos dados do giroscópio do eixo 'z' dividida pelo seu valor máximo de 32767, multiplicado pela escala escolhida e pelo tempo de interrupção do módulo, que nesse projeto é de 5 milissegundos. Essa equação dá o valor do ângulo percorrido no tempo medido enquanto a plataforma está girando.

$$\hat{A}ngulo = \frac{Gyr_Z}{32767} * 250\frac{g}{s} * 0.005\ s \qquad (4.1)$$

O programa fica verificando se a plataforma chegou ou passou do valor dado pela estimação de *DOA* para que a *motor shield* pare a rotação da plataforma e ande em linha reta por um determinado tempo, indo em direção à fonte do sinal sonoro.

4.3 Método de estimação de DOA

O arranjo linear proposto é de dois microfones espaçados de 17 centímetros, de acordo com a figura 4.1. Como já dito anteriormente, foi decidido utilizar esse espaçamento com o intuito de uma maior abrangência na angulação da plataforma, podendo chegar até 84,11°, e com 15 centímetros o máximo seria de 64,41°. O espaçamento entre os microfones deve ser

medido pois é importante para o cálculo de a estimação de DOA. A fonte do assovio indicada pelo azimute θ se encontra afastada.

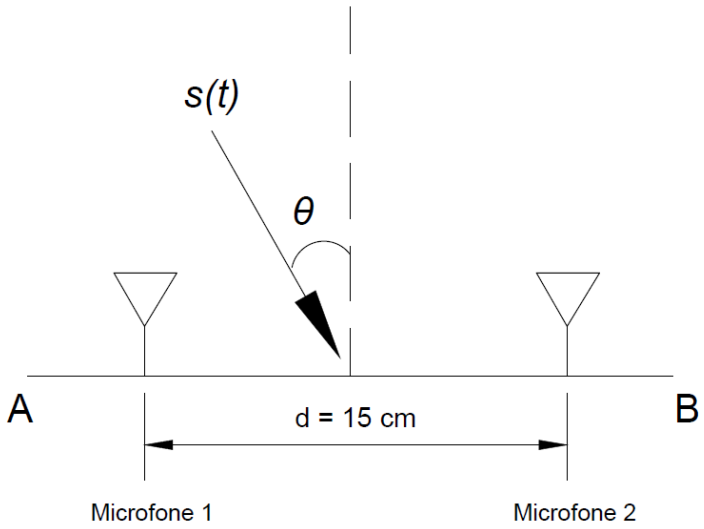

Figura 4-1: Montagem proposta de arranjo com dois microfones [13].

As equações (4.2) e (4.3) modelam o sinal recebido pelos microfones como $m_1(t)$ e $m_2(t)$. Percebe-se que o sinal está defasado em τ no sinal referente ao segundo microfone.

$$m_1(t) = A_1 s(t) + n(t), \tag{4.2}$$

$$m_2(t) = A_2 s(t - \tau) + n(t). \tag{4.3}$$

Pode-se observar essas equações modelam também o ruído branco gaussiano intrínseco ao sistema, definido por $n(t)$, e também o ganho de cada sinal, representados por A_1 e A_2.

Para que os sinais analógicos captados pelos microfones possam ser tratados por um microprocessador, eles passam pelo conversor A/D da placa, tornando-se sinais digitais em tempo discretos em uma taxa de f_{ams} $\left[\frac{amostras}{segundo}\right]$. Para poder estimar o azimute, esse sinal deve ser concentrado em uma janela de n amostras. Assim que digitalizado, cada amostra pode ser salva em um vetor, para que possa facilitar a leitura dos dados capturados.

A defasagem entre os sinais capturados através dos 2 microfones será indicada em passos na taxa de amostragem. Para se relacionar a k-ésima amostra em tempo discreto com o atraso temporal, uma análise dimensional pode ser feita como é mostrado na equação (4.4), onde k é o atraso relativo entre os sinais, em tempo discreto.

$$\tau_k = \frac{k}{f_{ams}} = \frac{[amostra]}{\left[\frac{amostra}{tempo}\right]} = [tempo] \tag{4.4}$$

Desenvolvendo a equação acima com o atraso temporal, demonstrada na equação (2.1), é obtido a relação do azimute θ com o atraso k, descrita conforme a equação (4.5):

$$\tau_k = \frac{k}{f_{ams}} = \frac{d.sen\theta.2\pi}{v}. \tag{4.5}$$

Com a finalidade de encontrar como resultado final o ângulo de azimute θ, a equação (4.5) pode ser para isso da seguinte forma:

$$\theta = sen^{-1}\left(\frac{k.v}{2\pi.d.f_{ams}}\right) \tag{4.6}$$

onde v é a velocidade do som (340 m/s), d é a distância entre os dois microfones (0,015m) e f_{ams} é a taxa de amostragem do sinal. Se k for corretamente estimado, a direção de chegada do sinal sonoro será dada por θ.

Para a realização dessa estimação com a menor taxa de erro possível, após o sinal ser digitalizado e salvo em vetores, o Arduino mesmo pode tratar os dados convertidos através de codificação em C. Para acompanhar e verificar esse processamento foi utilizada a ferramenta MATLAB para a verificação dos dados convertidos. Esse processamento consiste em:

1. Identificação da faixa de maior energia do sinal;
2. Aplicação de um filtro "média-móvel";
3. Interpolação cúbica;

4.3.1. Primeira etapa do tratamento: aplicação de filtro digital

A aquisição do sinal sonoro pelos microfones e o processo de conversão analógico/digital inserem ruídos consideráveis no sinal original, podendo causar erros consideráveis ao se determinar a direção de chegada do sinal sonoro. Para evitar isso pode-se utilizar filtros digitais para que esses ruídos sejam amenizados e ainda selecionar a banda de interesse.

Neste projeto, são utilizados dois filtros digitais. O primeiro deles se trata de um processo estatístico, chamado de "Média Móvel", que se trata de um filtro FIR – resposta finita ao impulso, do inglês *Finite Impulse Response* – que determina o valor de cada amostra de um vetor sendo igual à média entre 'n' valores adjacentes a ele, como mostra a equação

(4.7). Esses n valores são determinados pelo grau da média móvel utilizado, e nesse trabalho foi utilizado o grau 2.

$$Média\ Móvel \rightarrow y_i = \frac{1}{n}\sum_{j=1}^{n} x_{i-j} \tag{4.7}$$

Esse filtro de média móvel é aplicado pelo programa nos dados assim que são convertidos pelo conversor analógico/digital, filtrando assim o sinal todo captado pelo microfone e armazenando os dados na memória externa.

Depois de finalizada a conversão A/D, com os dados já filtrados pela média móvel e salvos na memória externa, o programa deste trabalho aplica ao sinal um outro filtro, só que dessa vez um filtro IIR, abordado na seção 2.4. O filtro foi testado inicialmente no MATLAB para a análise dos dados e então passado diretamente para o programa do projeto. Os dados utilizados na montagem desse filtro digital foram obtidos com a ferramenta *filterDesigner* do MATLAB. Essa filtragem utiliza os mesmos vetores que estão carregados na SRAM externa. No final desse procedimento, os dois vetores salvos estão filtrados pelos dois filtros aplicados.

4.3.2. Segunda etapa do tratamento: Identificação da faixa de maior energia

A faixa de maior energia captada pelos microfones é justamente a faixa que interessa para a estimação da DOA, pois é nessa faixa que está o sinal sonoro do assovio a ser estudado. Ao se encontrar essa faixa, os dados contidos nela podem ser transformados em outros vetores para que sejam manipulados diretamente na memória do Arduino, não necessitando mais ler os dados da memória externa, melhorando assim o tempo de processamento dos dados, enquanto o resto do sinal pode ser descartado. O pré-amplificador dos microfones já possui um filtro passa-baixa. Os períodos de baixa energia são descartados sem perda de informação válida para o processamento.

Para encontrar a faixa de interesse, foi tomado como referência o vetor que contém o sinal em tempo discreto do primeiro microfone. Inicialmente é calculada a média dos valores obtidos e então calcula-se o valor absoluto de cada elemento do vetor, subtraído do valor médio do sinal, em seguida é determinado um limiar empírico para detectar o início da faixa de maior energia. A partir desse ponto é utilizada uma quantidade específica de dados determinada a partir do início da faixa para realizar os cálculos.

4.3.3. Terceira etapa do tratamento: Interpolação

O limite na taxa de amostragem elevada do hardware acabou por limitar a aquisição de dados, pois a partir de uma taxa de amostragem $f_{ams} > 16\,kHz$, o sinal se torna excessivamente ruidoso, ou simplesmente fica errado. Como pode ser visto na equação (4.6) a taxa de amostragem acaba por influenciar na resolução da estimação de *DOA*, mostrando que quanto maior for a taxa, melhor será a resolução.

Para poder aumentar essa taxa de aquisição de dados aproximando da taxa ideal, foi feita a utilização de polinômios de interpolação, que são polinômios de 3° grau, transformando o conjunto de dados de entrada em um novo conjunto, com as transições suavizadas entre as amostras.

A programação da interpolação neste trabalho foi feita diretamente com o Arduino em cima do vetor salvo com a faixa de maior energia, retornando um vetor duas vezes maior que o anterior, virtualmente dobrando a taxa de amostragem para o mesmo conjunto de dados.

A interpolação consistiu em inicialmente passar os valores do vetor da faixa com mais energia para as posições pares de outro vetor com o dobro de posições do primeiro vetor, e nas posições ímpares eram colocados os valores referentes à interpolação de acordo com a equação (4.8), onde V_n é resultado da interpolação e p_n é o dado na posição n do vetor.

$$V_n = \frac{p_{n-1}}{2} + \frac{p_{n+1}}{2} \tag{4.8}$$

4.4 Defasagem em tempo discreto

Para obter a defasagem em tempo discreto dos dois sinais coletados pelos microfones é necessário verificar a similaridade entre esses sinais. Isso pode ser feita com o casamento temporal entre eles, que verifica quão próximos esses dois vetores são um do outro. Isso pode ser implementado diretamente pelo Arduíno utilizado através do cálculo da norma euclidiana.

A norma euclidiana de um vetor é descrita como a raiz quadrada da soma dos valores absolutos elevados ao quadrado de cada elemento do vetor, conforme demonstra a equação (4.9):

$$|\vec{v}| = \sqrt{\sum_{i=1}^{n} |v_i|^2} \tag{4.9}$$

Para identificar a diferença entre os vetores que armazenam os dados digitalizados, a equação (4.9) é definida como:

$$\frac{1}{n}|\vec{x}_1 - \vec{x}_2| = \frac{1}{n} \sqrt{\sum_{i=1}^{n} |x_{1,i} - x_{2,i}|^2}^2 \tag{4.10}$$

onde $x_{1,i}$ e $x_{2,i}$ representam os i-ésimos elementos dos vetores que armazenam, respectivamente, os dados do primeiro e segundo microfone. O resultado da equação (4.10) fornece um valor de erro, também chamado de Erro Quadrático Médio, que quantifica a semelhança entre os dois vetores.

Para obter a medida de defasagem entre esses dois sinais é necessário determinar o erro quadrático médio entre versões deslocadas dos vetores obtidos com a interpolação realizada anteriormente. Para cada deslocamento *k* entre os vetores, tem-se um valor calculado do erro que é salvo em outro vetor.

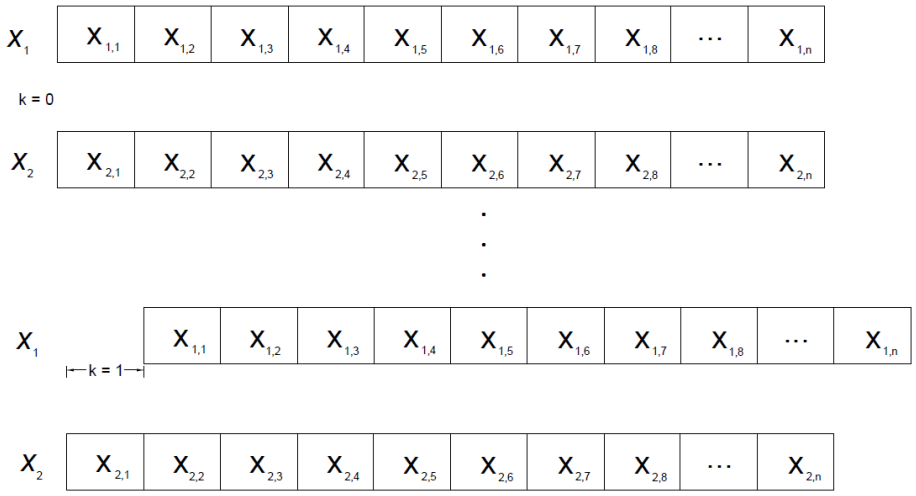

Figura 4-2: Representação da operação de correlação efetuada [13]

Dos valores armazenados nesse vetor equivalente à diferença entre os vetores para cada deslocamento, o que tiver menor valor demonstra maior proximidade entre os dois sinais. Para o propósito deste trabalho busca-se justamente o elemento *k* que contém o menor

valor em módulo dentre os valores desse vetor, que representa o atraso em tempo discreto entre os sinais. Esse valor *k* que será utilizado na equação (4.6) para poder estimar o azimute do sinal de chegada.

4.5 Resolução do estimador e limitações

Como foi dito anteriormente, a taxa de amostragem dos sinais é bem relevante para a estimação de *DOA* por defasagem temporal. Como pode ser visto na equação (4.6), o seno do ângulo de azimute é inversamente proporcional a essa taxa, mostrando que quanto maior for a taxa, melhor será a resolução para a obtenção desse ângulo. Como a digitalização do sinal analógico pelo Arduino acima de 16 kHz apresenta muitos erros, acaba por ser uma limitação do projeto.

Com o conversor A/D digitalizando o sinal a uma taxa $f_{ams} = 16\,kHz$, tomando $k = 1$ e com os microfones espaçados na distância mínima de *15 cm*, obtemos o valor de azimute mínimo que pode ser detectado pelo sistema, sendo de $\theta = 13{,}0305°$. Ao se utilizar a interpolação, tornou-se possível manter a frequência f_{ams} em 32 kHz , implicando em uma melhor resolução na obtenção do azimute, passando para $\theta = 6{,}473°$, ou seja, qualquer variação maior que 0° e menor que 6,473° será considerada igual à última, porém essa faixa é considerada pequena o suficiente para a estimação, dada as limitações de processamento do Arduino.

Outra limitação a ser abordada é o valor máximo de *k*, uma vez que, da trigonometria, o valor máximo do seno de um ângulo deve ser igual ou menor a 1, e como o número de amostras *k* é a variável dessa equação, é ela quem define também o ângulo máximo. Com essa informação aplicada na equação (4.5) tem-se que:

$$k \leq \frac{d.f_{ams}}{v} \tag{4.11}$$

Dessa forma é demonstrado que, a uma distância d fixa entre os microfones, o valor máximo de k é diretamente proporcional à taxa de amostragem. Para este trabalho, espaçando os microfones a uma distância mínima de *15 cm*, e com a ajuda da interpolação amostrar o sinal a 32 KHz, o valor limite de atraso é de $k = 8$, dando um azimute máximo de $\theta = 64{,}406°$. Os valores identificados com *k > 8* devem ser descartados.

Concluindo: a aquisição e o tratamento dos dados recebidos, segue o fluxo do diagrama apresentado na figura 4.3, que mostra todos os passos realizados pelo programa para

a aquisição do azimute de *DOA*. Cada bloco foi relatado aqui neste capítulo, desde a etapa de aquisição até a estimação do ângulo azimutal, onde o programa retorna um número inteiro para que a plataforma possa se orientar com relação à fonte do assovio. Os próximos capítulos discutirão os resultados observados, e extrairão as conclusões finais do projeto.

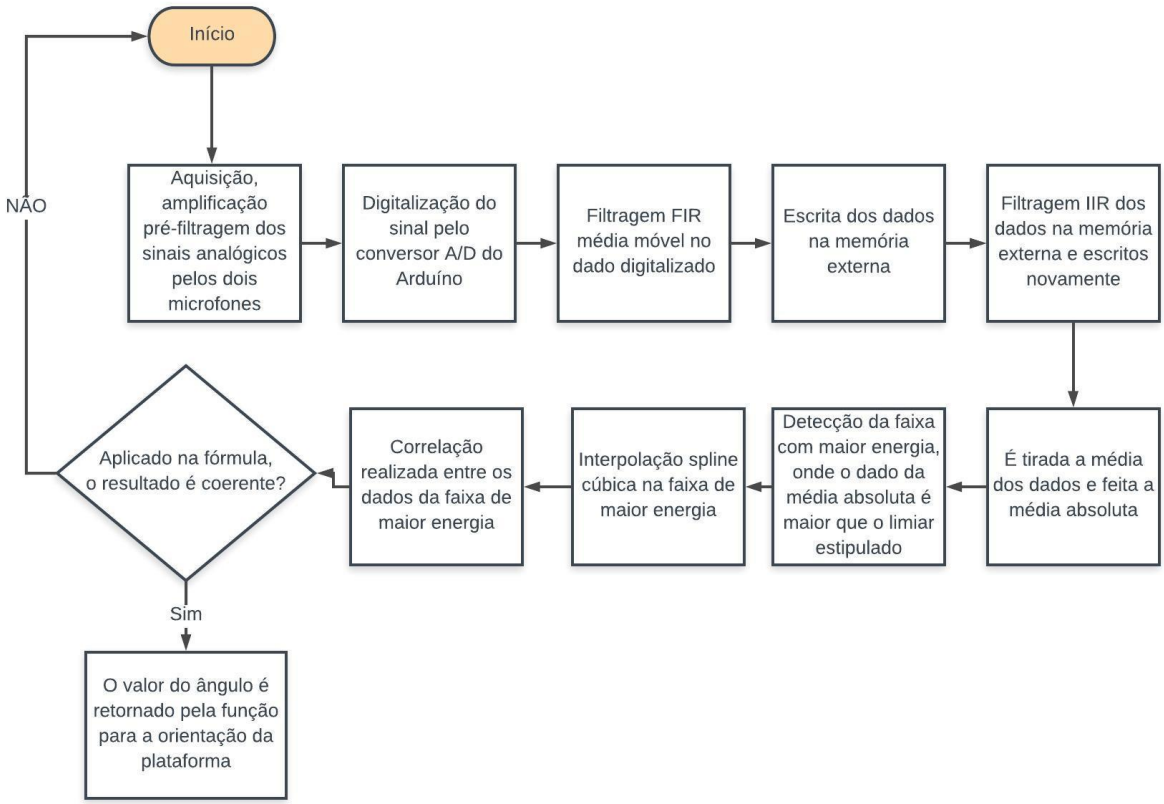

Figura 4-3: Diagrama referente ao método de estimação de *DOA* deste projeto

5 ENSAIOS E RESULTADOS

Neste Capítulo são apresentados os resultados experimentais em cada etapa do processo da estimação de direção de chegada de um sinal sonoro utilizando a defasagem temporal. Os ensaios foram feitas através da IDE do Arduino, que retornavam dados de forma serial e usados no software MATLAB para a averiguação do que foi obtido.

O arranjo linear utilizado para os ensaios foi de dois microfones espaçados de uma distância $d = 17\ cm$, e a taxa de amostragem utilizada em cada microfone foi de $f_{ams} = 16\ kHz$. Para a obtenção do sinal sonoro mais livre de ruídos externos, os ensaios foram feitos em um ambiente fechado. Para a estimação, a fonte sonora se posicionou em um azimute aproximado a 0°, 30° e 60°.

As figuras 5.1 e 5.2 mostram os sinais sonoros adquiridos sem e com a filtragem dos dados do microfone 1 e do microfone 2, respectivamente. Os sinais sem filtragem possuem muito ruído, o que pode fazer com que a determinação da faixa de maior energia seja feita de forma errada. Com a filtragem dos sinais, o procedimento para verificar a faixa do assovio torna-se mais eficaz, como pode ser visto nas figuras dos sinais filtrados. Esses sinais obtidos são referentes às equações descritas em (4.2) e (4.3). Verificando as figuras pode-se facilmente distinguir o momento do assovio do período de silêncio, onde não há nenhum sinal relevante.

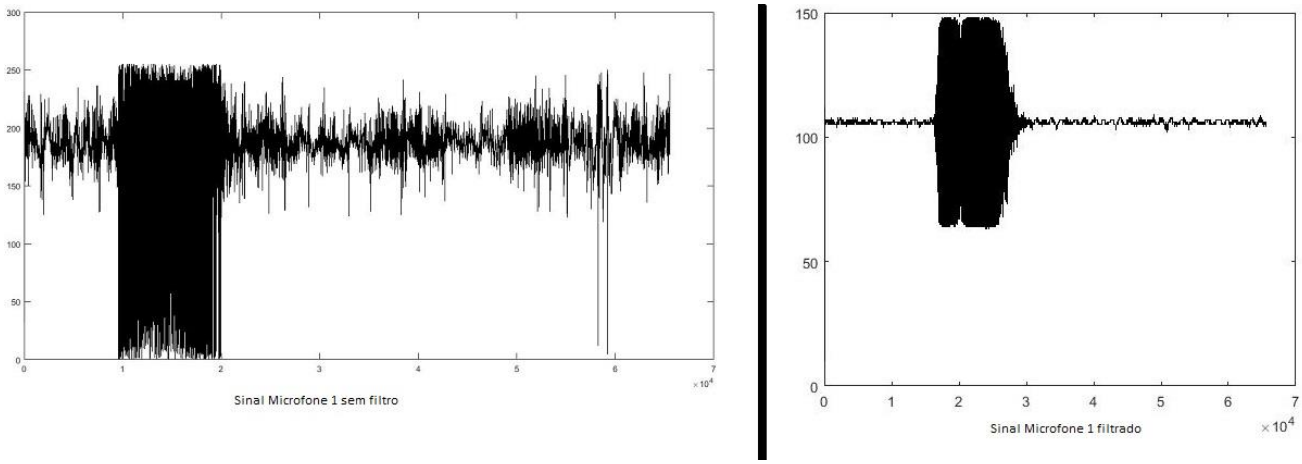

Figura 5-1: Sinais do microfone 1 sem filtro / com filtro

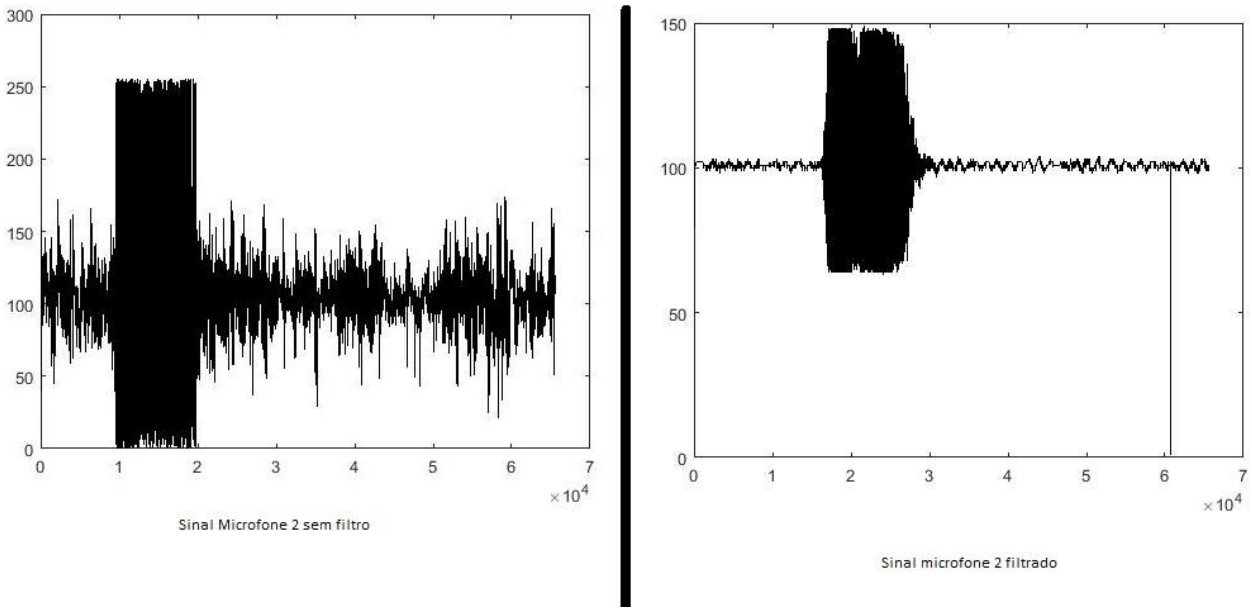

Figura 5-2: Sinal do microfone 2 sem filtro / com filtro

Feita as aquisições, o programa calcula a média do sinal, subtrai de todos os dados obtidos, para obter um sinal com média 0, e então calcula o valor absoluto dele, para que não haja sinal negativo e então é dividido todos os valores pelo seu valor máximo, assim a faixa de dados passa a ser [0,1]. Dessa forma a determinação da faixa de maior energia é feita com mais facilidade como pode ser visto na figura 5.3.

Para poder identificar a faixa procurada, foi determinado um limiar que, quando o dado fosse maior do que ele, se tratava do sinal do assovio. Nesse projeto o limiar foi de 0.29, ou seja, depois do processo acima citado, se o sinal for maior do que 0.29, o programa grava o endereço desse dado para poder usar apenas essa faixa de maior energia. Esse limiar pode ser visto na figura 5.3 como uma reta azul na horizontal, e a reta azul na vertical determina o início da faixa.

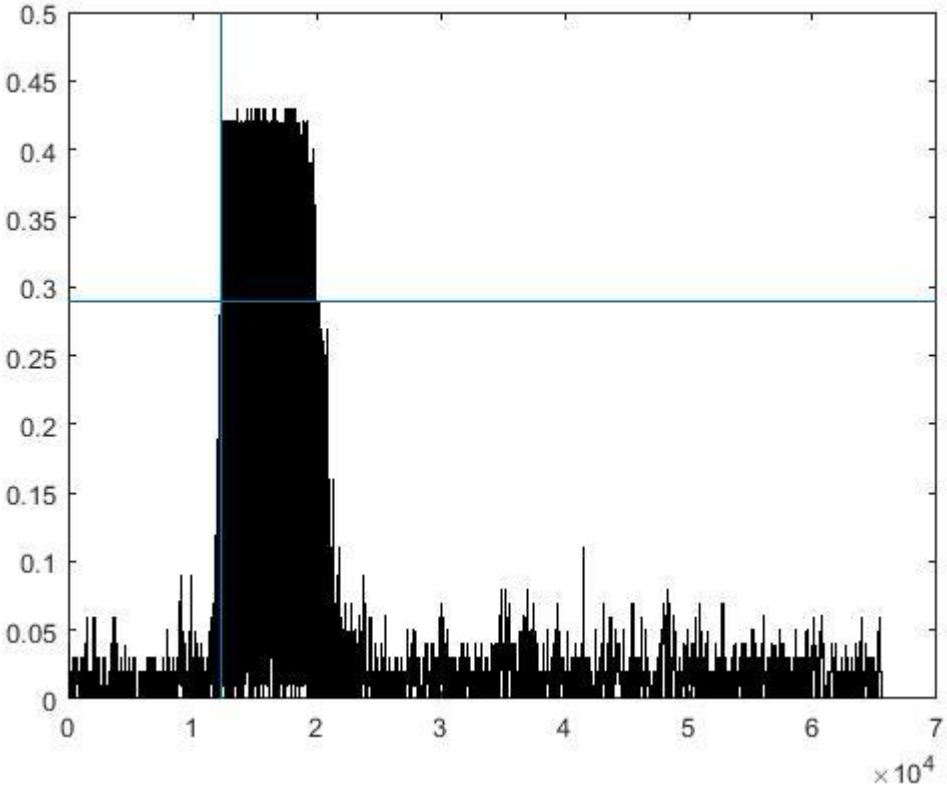

Figura 5-3: Sinal do microfone 1 filtrado com média absoluta

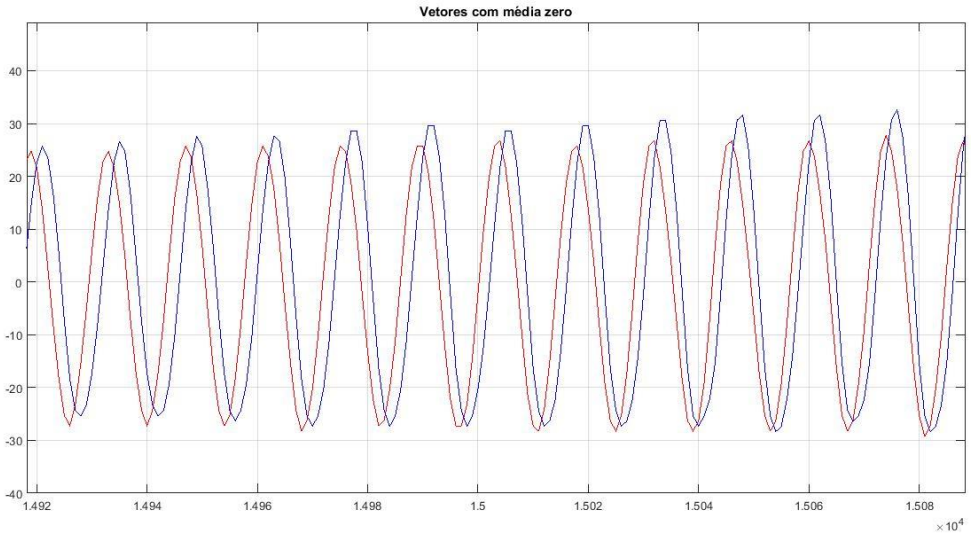

Figura 5-4: Defasagem entre os sinais dos dois microfones em uma aquisição

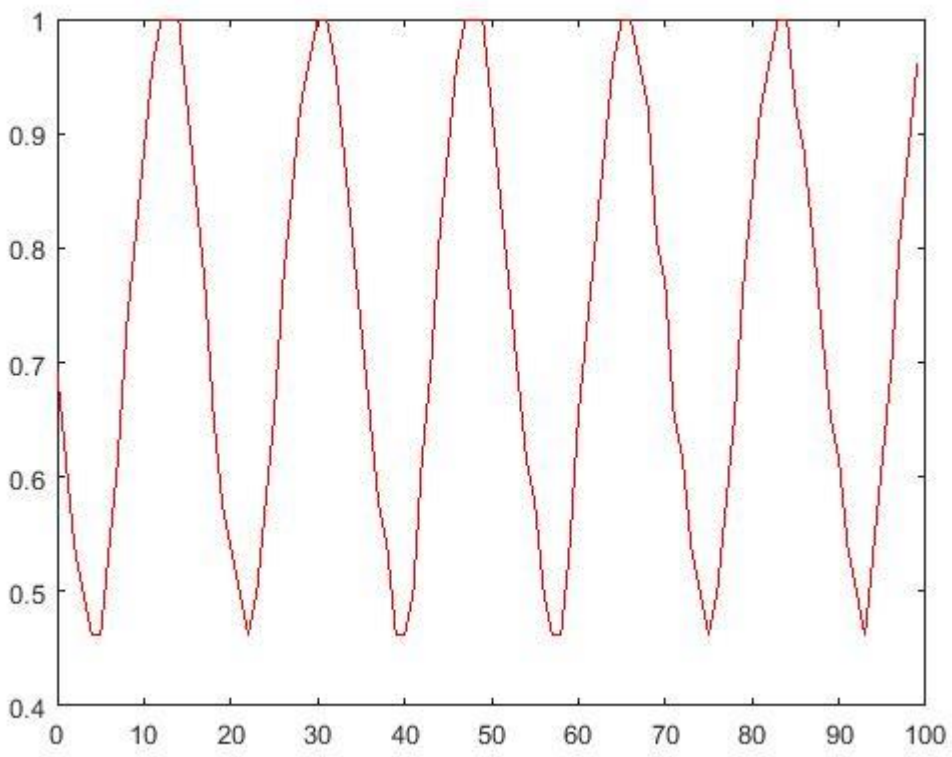

Figura 5-5: Correlação temporal entre os microfones

Analisando a figura 5.4 pode ser vista a fase entre os sinais dos dois microfones que estão captando o som, demonstrando que a defasagem não é muito elevada entre si. Dessa forma, com a relação temporal feita entre eles, se obtém o gráfico mostrado na figura 5.5, tornando possível a estimação do azimute da fonte do sinal sonoro.

Para verificar se o algoritmo utilizado estava funcionando de forma correta foi feita uma repetição de ensaios, e observando o resultado para então concluir sobre a eficiência do método. Para isso, foram escolhidos 3 ângulos preferenciais, sendo eles 0°, 30° e 60°, e foram feitos 15 ensaios para cada ângulo desse de referência, analisando o ângulo retornado pelo programa.

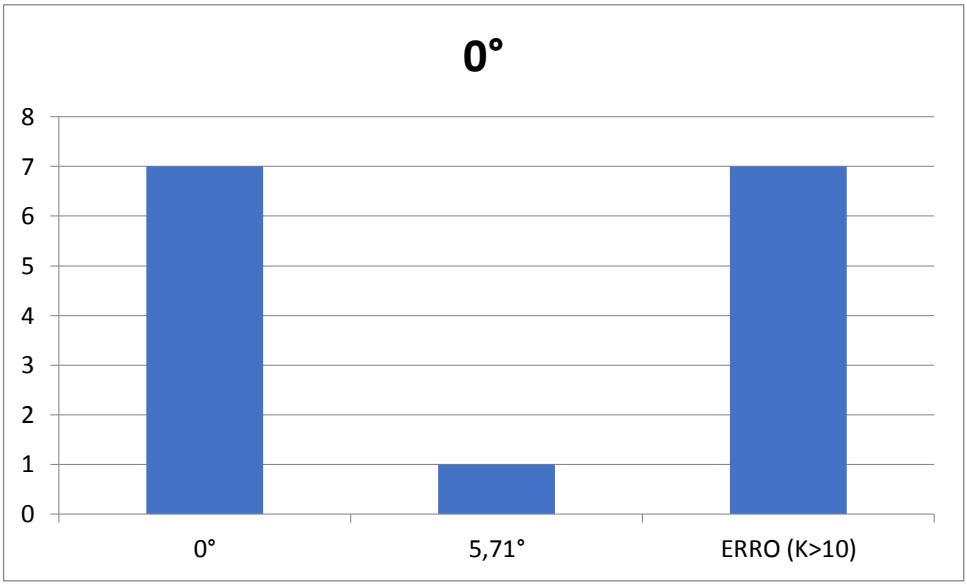

Figura 5-6: Estimativas para sinais com θ=0°

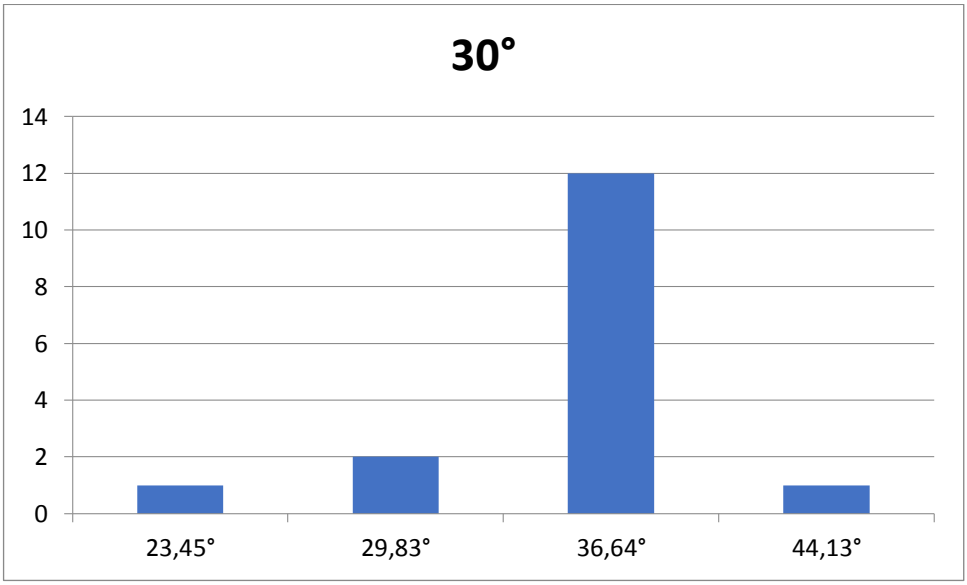

Figura 5-7: Estimativas para sinais com θ=30°

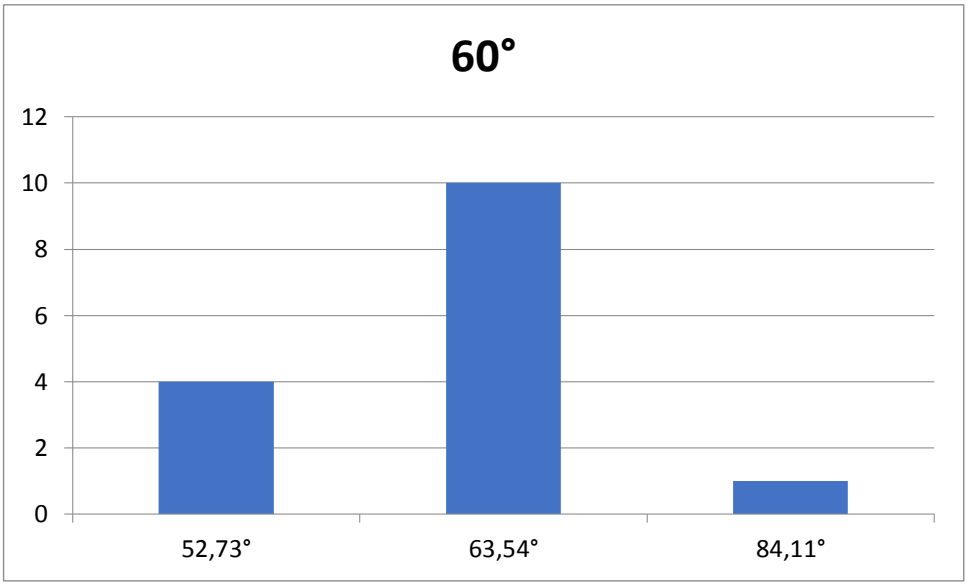

Figura 5-8: Estimativas para sinais com θ=60°

Para a verificação do algoritmo do controle foram feitos 15 ensaios fazendo com que a plataforma gire uma quantidade de graus previamente especificada. Para esses ensaios foi escolhida uma angulação de 90° para poder verificar de forma mais precisa o quanto realmente a plataforma girou.

Tabela 2: Quantidade em graus rotacionados pela plataforma em ensaios de 90°

Ensaio	Graus rotacionado
1	90,17°
2	90,58°
3	90,27°
4	91,04°
5	90,60°
6	91,21°
7	91,65°
8	90,42°
9	90,96°
10	90,10°
11	90,41°
12	90,68°
13	90,27°
14	90,79°
15	90,95°

5.1 Avaliação de resultados

Nas estimativas do azimute com $\theta = 0°$ é possível notar que basicamente metade dos dados adquiridos apresentou erro que foram ocasionados quando a relação temporal retornava o valor *k*, de atraso em tempo discreto, muito elevado, e ao se utilizar ele na equação (4.6) não era possível encontrar um ângulo.

Na tabela 1 é apresentado o valor do atraso em tempo discreto k com o seu respectivo azimute que é encontrado utilizando a equação (4.6).

Tabela 2: Valores de ângulo azimutal estimado referente ao atraso em tempo discreto medido

Atraso em tempo discreto '*k*'	Ângulo azimutal estimado
0	0°
1	5,71°
2	11,47°
3	17,36°
4	23,44°
5	29,82°
6	36,64°
7	44,13°
8	52,72°
9	63,54°
10	84,11°
>10	ERRO!

Como afirmado anteriormente, para cada ângulo de referência foram realizados 15 ensaios. Pelo método de correlação, é retornado o atraso em tempo discreto. O ensaio é concluído ao se calcular a média das estimativas utilizadas, e então se dá início a um novo ensaio. Este procedimento é repetido até que todos os ensaios sejam realizados.

Analisando-se as figuras 5.6 a 5.8, pode-se notar que o algoritmo de estimação da direção de chegada funciona e consegue acertar razoavelmente bem os ângulos de chegada. A frequência de boas estimativas é satisfatória para o algoritmo utilizado. Observa-se, porém, que certas estimativas podem se desviar bastante do resultado esperado, principalmente se tratando do azimute de 0°. Nesses casos onde o algoritmo da estimação retorna erro, a plataforma permanece imóvel, e realiza outra aquisição para realizar outra estimativa.

De acordo com a tabela 1, pode ser observado que o controle da plataforma exerce da forma que é esperado, podendo girar 1°a mais do que o esperado, porém não influenciando no objetivo final, uma vez que essa quantidade sobressalente é bem pequena.

Um vídeo exemplar, que mostra um ensaio da plataforma pode ser acessado na internet pelo link https://www.youtube.com/watch?v=dcZkC_VWsjs.

5.2 Limitações do projeto

Uma das principais limitações do projeto é de que o arranjo dos microfones não consegue identificar o sentido do sinal. Um dos fatores disso ocorrer é devido aos microfones que são sensores omnidirecionais, que detectam sinais sonoros vindos de todas as direções, dificultado a detecção do sentido do sinal.

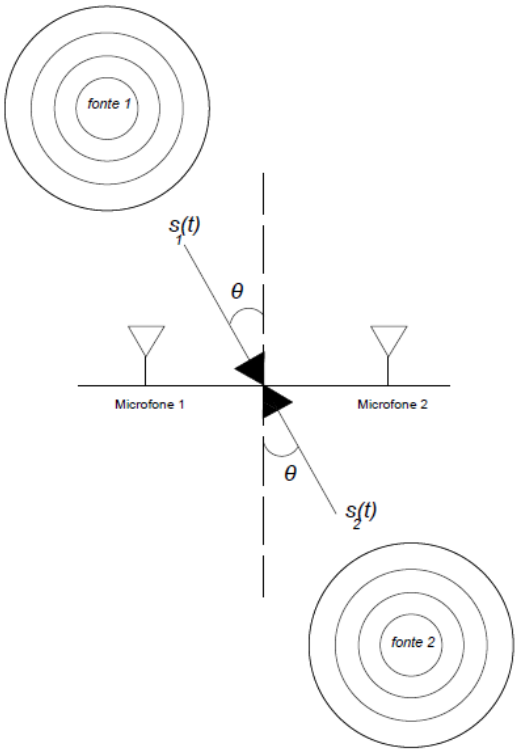

Figura 5-9: Demonstração de como o mesmo ângulo pode valer para duas fontes diferentes [13]

Outro fator que dificulta é que o algoritmo não extrai nenhuma informação que diz se a fonte do som está à frente ou atrás do arranjo. Além dessa limitação, a plataforma também só consegue estimar a direção de chegada do sinal apenas do lado do microfone 1. Isso se deve ao próprio algoritmo, que na hora de calcular a correlação só trabalha alterando apenas um vetor.

No decorrer do projeto, para a verificação dos dados era necessário utilizar a ferramenta do MATLAB, e a comunicação desse software com o Arduino foi bem trabalhosa e demorada, uma vez que as respostas obtidas tanto pela placa como pelo software eram lentas, levando um tempo maior para a conclusão do projeto.

Para a programação da mobilidade da plataforma, foi gasto muito tempo com o entendimento para utilizar o MPU-6050. Inicialmente foi pensado em utilizar uma plataforma móvel de outro projeto de localização de obstáculos, porém a plataforma já estava bem modificada, apresentando vários erros na comunicação a *motor shield* com os motores, então foi decidido utilizar uma plataforma nova para este projeto.

6 CONCLUSÃO

Este projeto teve por foco programar algoritmos em uma plataforma móvel com a utilização de um processador acessível para poder estimar a direção de chegada de um sinal sonoro e movimentar a plataforma em direção à fonte sonora. Este algoritmo pode ser utilizado em robôs para serem controlados por som, e podem ter outras várias aplicações como rastreamento de disparos de armas, localização de pessoas em lugares de difícil visibilidade e sonares.

A substituição do Arduino MEGA pelo Due implicou em uma dificuldade quanto à documentação da placa. Para a utilização dos *timers* e do conversor A/D a documentação não é muito boa, precisando ser estudado o manual para poder implementar da melhor forma.

Para controlar a plataforma, inicialmente tinha se pensado em ligar os motores DC por um determinado tempo até que girasse a quantidade de graus solicitada, porém foi observado que as rodas acabavam por deslizar e dependia muito do terreno em que estava se movimentando, por esse motivo foi utilizado o módulo com o giroscópio para ter uma maior certeza quanto à direção a ser tomada pela plataforma, e assim a rotação se tornou mais precisa.

Nos ensaios realizados algumas discrepâncias foram encontradas durante o processo de estimação da direção de chegada do sinal sonoro, mas que puderam ser corrigidas através do código do algoritmo utilizado. Podendo aumentar a velocidade de aquisição (taxa de amostragem) aumenta a amostragem do sinal, fazendo com que o algoritmo possa ser mais preciso com relação a estimação de chegada do sinal. Com esse projeto, a plataforma acaba não cobrindo todos os graus, como foi discutido no capítulo anterior, fazendo com que o arranjo não detecte exatamente todas as posições da fonte sonora.

Para a melhoria do projeto, em trabalhos futuros, deseja-se identificar de qual lado da plataforma o sinal está vindo, além de poder identificar o sentido, para saber se sinal vem da frente ou de trás da plataforma. Esse projeto pode ser juntado a outros projetos que utilizam uma plataforma móvel, como por exemplo, o de detecção de obstáculos, podendo realizar as duas funções, melhorando as ações da plataforma.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] LV, X., ZHANG, M., YUAN, G., et al., "*Robot sound source search strategy based on multi-blackboard model". In: Proceedings of the IEEE International Conference on Robotics and Biomemetics* (ROBIO), pp. 633–638, Tianjin, China, December 2010.

[2] A. T. Gontijo, Estimação da Direção de Chegada em Tempo Real por Arranjo de Microfones. Tese de Mestrado, UnB, Brasília, ago. 2010.

[3] Arduino. (s.d.). *Compare Board Specs*. Disponível em Arduino: https://www.arduino.cc/en/Products/Compare.

[4] Arduino. (s.d.). Arduino Due. Disponível em Arduino: https://www.arduino.cc/en/Main/ArduinoBoardDue.

[5] Toshiba. (s.d.). TB6612FNG. Disponível em https://cdn-shop.adafruit.com/datasheets/TB6612FNG_datasheet_en_20121101.pdf.

[6] NXP *Semiconductors*. (2015). PCA9865. Disponível em http://www.nxp.com/documents/data_sheet/PCA9685.pdf.

[7] Phillips *Semiconductors*. (2003). AN10216-01 I2. Disponível em http://www.nxp.com/documents/application_note/AN10216.pdf.

[8] MPU-6000 *and* MPU-6050 *Product Specification*. Disponível em https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[9] Chapman, S. (2005). *Electric Machinery Fundamentals* (4ª ed.). New York: McGraw-Hill.

[10] Sen, P. C. (1997). *Principles of Electric Machines and Power Electronics* (2ª ed.). Kingston: John Wiley & Sons.

[11] OLIVEIRA, IGOR; OLIVEIRA, RENATO (2016). Plataforma móvel com detecção de obstáculos. Trabalho de Conclusão de Curso em Engenharia Elétrica, 2016, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF.

[12] LIMA, LUCAS; SOARES, MATHEUS (2017). Plataforma móvel com detecção de obstáculos. Trabalho de Conclusão de Curso em Engenharia Elétrica, 2017, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF.

[13] TORRES, GUILHERME DE SOUSA (2018). Estimação de Direção de Chegada de Sinais Sonoros Utilizando Arranjo de Sensores. Trabalho de Conclusão de Curso em Engenharia Elétrica, 2018, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF.

[14] Loja online huinfinito. Disponível em http://www.huinfinito.com.br/chassis-plataformas/872-kit-chassi-plataformapara-robo-4wd-acrilico.html.

[15] Datasheet Microchip 23LC1024. Disponível em https://www.microchip.com/wwwproducts/en/23LC1024.

[16] *Adafruit motor shield v2 for Arduino.* Disponível em https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino?view=all.

[17] Arduino Due Pinout. Disponível em https://components101.com/microcontrollers/arduino-due.