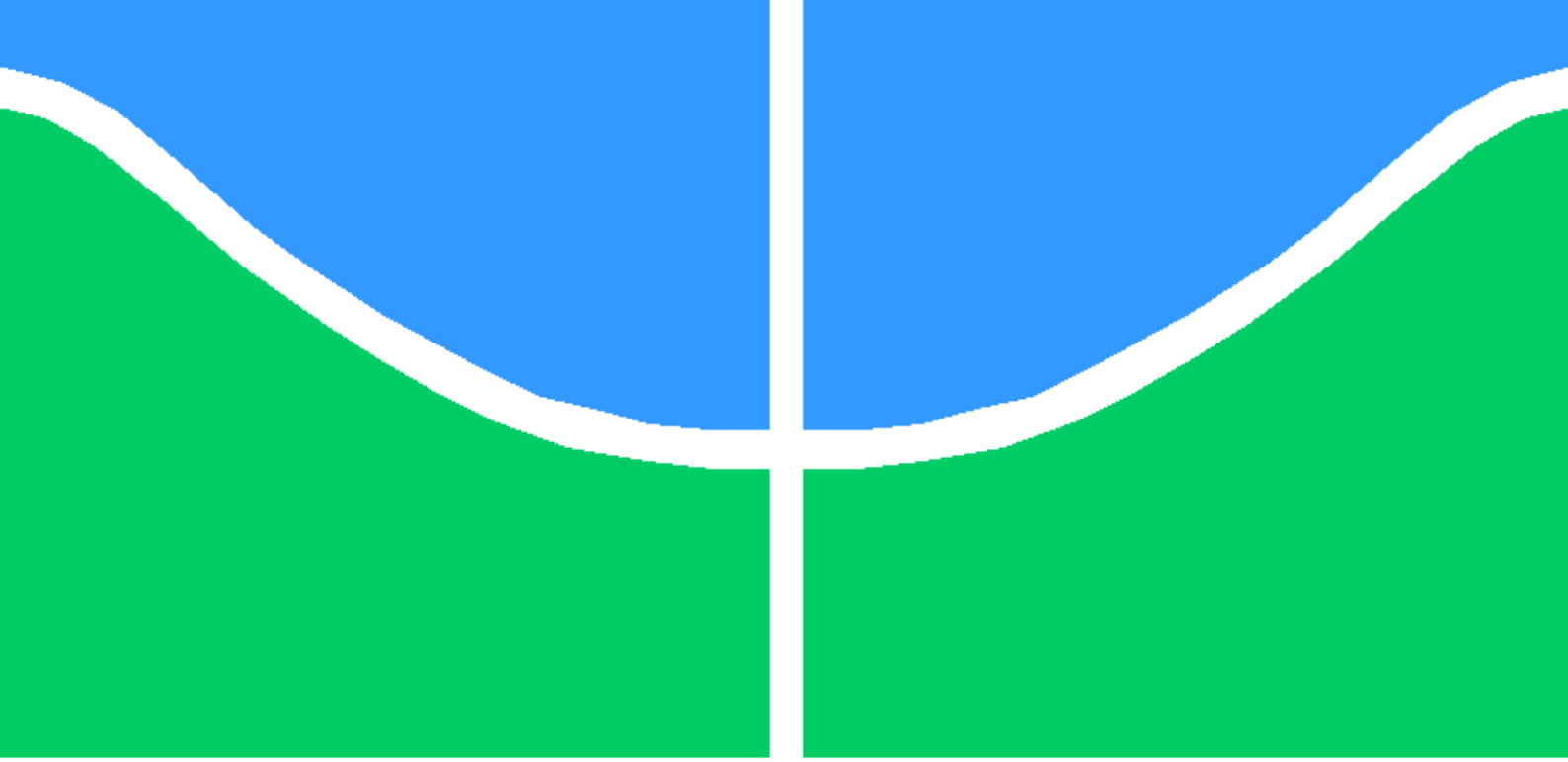


SISTEMA PARA ESTIMAÇÃO DA DIREÇÃO DE CHEGADA PARA SINAIS SONOROS

YURI SOUZA REIS

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SISTEMA PARA ESTIMAÇÃO DA DIREÇÃO DE CHEGADA PARA SINAIS SONOROS

YURI SOUZA REIS

Orientador: RICARDO ZELENOVSKY, DOUTOR - PUC-RJ, ENE/UNB

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO PPGENE.DM - XXX/AAAA

BRASÍLIA-DF, 10 DE DEZEMBRO DE 2020.

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SISTEMA PARA ESTIMAÇÃO DA DIREÇÃO DE CHEGADA PARA SINAIS SONOROS

YURI SOUZA REIS

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO ACADÊMICO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA.

APROVADA POR:

Ricardo Zelenovsky, Doutor - PUC-RJ, ENE/UnB
Orientador

Daniel Chaves Café, ENE/UnB
Examinador interno

Eduardo Peixoto Fernandes da Silva, ENE/UnB
Examinador interno

BRASÍLIA, 10 DE DEZEMBRO DE 2020.

FICHA CATALOGRÁFICA
YURI SOUZA REIS
Sistema para estimação da direção de chegada para sinais sonoros
2020xv, 38p., 201x297 mm
(ENE/FT/UnB, Graduação, Engenharia Elétrica, 2020)
Trabalho de conclusão de curso de Graduação - Universidade de Brasília
Faculdade de Tecnologia - Departamento de Engenharia Elétrica

REFERÊNCIA BIBLIOGRÁFICA

YURI SOUZA REIS (2020) Sistema para estimação da direção de chegada para sinais sonoros. Trabalho de conclusão de curso de Graduação em Engenharia Elétrica, Publicação xxx/AAAA, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 38p.

CESSÃO DE DIREITOS

AUTOR: Yuri Souza Reis
TÍTULO: Sistema para estimação da direção de chegada para sinais sonoros.
GRAU: Graduação ANO: 2020

_____

Yuri Souza Reis
Brasília/Df

Agradecimentos

Gostaria de agradecer e dedicar este trabalho a todos os meus familiares e amigos que sempre me motivaram durante os estudos, principalmente em momentos de dificuldade diante de grandes desafios em minha jornada acadêmica.

Dedico um grande agradecimento aos meus pais, os quais sempre reconheceram minhas conquistas e me incentivaram a seguir meus sonhos. Todas as minhas conquistas em minha formação intelectual são principalmente resultado do apoio incondicional dos meus pais.

Serei eternamente grato a todos os professores que contribuíram para a minha formação como profissional.

Agradeço imensamente ao meu orientador, professor Ricardo Zelenovsky, que sempre transmite seus conhecimentos com muita dedicação aos seus alunos e me possibilitou um grande desenvolvimento técnico durante nossas reuniões para a elaboração deste trabalho de conclusão de curso. Gostaria de agradecer e dedicar este trabalho a todos os meus familiares e amigos que sempre me motivaram durante os estudos, principalmente em momentos de dificuldade diante de grandes desafios em minha jornada acadêmica.

Resumo

Este documento apresenta um ensaio acerca de técnicas computacionais para estimar a direção de chegada (DOA) de um sinal sonoro em um ambiente fechado. Tal projeto foi elaborado buscando-se por uma implementação de baixo custo computacional e eficiência adequada. O baixo custo computacional está relacionado à busca de componentes eletrônicos de valor mais econômico e desempenho satisfatório para a elaboração de projetos. A eficiência, por sua vez, está relacionada a parâmetros desejados durante a operação do sistema projetado. Podemos levantar como parâmetros desejados: consumo de energia; resolução e faixa de ângulos de direção estimados; tamanho de memória e tempo de execução do código empregado para realizar a estimação; taxas de erro e vulnerabilidade do sistema a fontes de distúrbios durante o processo de estimativa.

O sinal sonoro oriundo de uma fonte a ser localizada será captado por um circuito de captação com dois microfones. Este circuito foi construído por alunos e professores da Universidade de Brasília e sua topologia também é apresentada para observação do leitor. Faz-se, além disso, uma análise sobre o modelo que explica o processo de transmissão do sinal sonoro, sua captação e as condições relacionadas à posição dos microfones e ambiente para realizar a estimativa proposta neste projeto. Estas considerações visam reduzir erros decorrentes de efeitos ondulatórios (por exemplo, a reflexão dos sinais sonoros nas paredes do ambiente de simulação) e erros que não atendem às condições do método de estimativa adotado.

Tal sinal, então, passa a ser transmitido e processado por um microprocessador do tipo MSP430, fabricado pela Texas Instruments. O algoritmo utilizado para estimar a posição da fonte sonora emprega técnicas de correlação temporal aplicada entre os sinais captados pelos microfones. Mede-se, então, o erro quadrático médio entre os dados e calcula-se o atraso da captação entre os microfones e a correspondente direção da fonte sonora.

Palavras-chave: Microprocessadores; MSP430; DOA; sinal sonoro; baixo custo; eficiência; estimativa; sistema de captação; periférico de memória; comunicação SPI; comunicação I_2C; LCD; DMA; conversor A/D; Code Composer.

Abstract

This document presents a study about computational techniques to estimate the direction of arrival (DOA) of a sound signal in a closed environment. This project was designed seeking a low computational cost implementation and adequate efficiency. The low computational cost is related to the search for electronic components of more economic value and satisfactory performance for the elaboration of projects. Efficiency, now, is related to desired parameters during the operation of the designed system. We can raise as desired parameters: energy consumption; resolution and range of estimated steering angles; memory size and execution time of the code used to perform the estimation; error rates and vulnerability of the system to sources of disturbances during the estimation process.

The sound signal from a source to be located will be captured by a captation circuit with two microphones. This circuit was built by students and professors from the University of Brasilia and its topology is also presented for the observation of the reader. In addition, an analysis is made on the model that explains the process of transmitting the sound signal, its capture and the conditions related to the position of the microphones and the environment to execute the estimate proposed in this project. These considerations intend to reduce errors resulting from wave effects (for example, the reflection of the sound signals on the walls of the simulation environment) and errors that do not satisfy the conditions of the adopted estimation method.

This signal is then transmitted and processed by an MSP430 microprocessor manufactured by Texas Instruments. The algorithm used to estimate a sound source position employs temporal correlation techniques applied between the signals captured by the microphones. The mean square error between the data is then measured and the pickup delay between microphones and a corresponding direction of the sound source are calculated.

Keywords: Microprocessors; MSP430; DOA; signal sonorous; low cost; efficiency; estimate; captation system; memory peripheral; SPI communication; communication I_2C; LCD; DMA; A/D converter; Code Composer.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

Introdução

1.1 Ambientação

Atualmente o atendimento de demandas da sociedade tem impulsionado o advento de novas tecnologias. Um ramo de pesquisa, por exemplo, visa desenvolver instrumentos de medida para estimar a direção de chegada (DOA, Direction of Arrival) de sinais, sejam eles sonoros ou eletromagnéticos. Esta ação possibilita estimar a direção de chegada do sinal em estudo, melhorando a captação desse sinal.

Antenas direcionais, conforme apresentado por de Pesquisas Espaciais, permitem a irradiação ou o recebimento de ondas eletromagnéticas. Quando atuam na captação de sinais, tais equipamentos transformam energia elétrica irradiada em energia eletromagnética guiada para uma linha de transmissão. Este sinal pode, então, ser transmitido e processado por um sistema adjacente. Tal antena pode ser posicionada por um servomecanismo, dispositivo automático utilizado para posicionar antenas que demandam uma grande quantidade de força para serem movimentadas. Esse controle pode ser guiado pela estimativa DOA, visando posicionar a antena na direção do sinal de interesse.

O projeto proposto neste documento, por sua vez, procurará desenvolver um sistema capaz de realizar uma estimativa DOA aplicada a um sinal sonoro. Cabe-nos, agora, ressaltar a importância desse ramo de estudo. A detecção de sinais sonoros visa, além de estimar a localização da fonte emissora e possivelmente quem a opera, possibilitar uma ação rápida de quem realiza este rastreamento. Após uma consulta em trabalhos da comunidade acadêmica (Prandel (2012), e J. A. Apolinario Jr. (2011b), e J. A. Apolinario Jr. (2011a), e J. A. Apolinario Jr. (2011c), Chacon-Rodriguez and Julian (2011)), podemos destacar aplicações como equipamentos voltados à detecção e estimação da direção de disparos de armas de fogo como, por exemplo, tiros feitos por snipers. Equipamentos de assistência a pessoas com deficiência auditiva também podem vir a utilizar sistemas que empregam técnicas de estimativa DOA, possibilitando detectar objetos perigosos próximos como carros (Machado (2019)) ou aplicativos que emitam sons para localização de destinos desejados por seus usuários,

podendo emitir sinais sonoros baixas frequências, imperceptíveis para o ser humano.

Durante a fase de planejamento de um projeto, estabelecem-se os objetivos que este em estágio de desenvolvimento deverá atender. Além disso, o projeto precisa estar de acordo com conceitos estipulados para um produto tais como sustentabilidade, baixo custo, economia energética, automação do processo produtivo, além de possibilitar a conectividade entre usuários e equipamentos eletrônicos presentes em diversos ambientes. Dessa forma, muitas pesquisas e projetos buscam pela implementação de sistemas com melhor desempenho, eficiência energética e menores custos.

O desenvolvimento da robótica, por exemplo, tem alcançado grandes melhorias através dos avanços no ramo da eletrônica. Essas melhorias surgem como resultado de pesquisa, desenvolvimento e utilização de novos componentes eletrônicos. A partir desses avanços são elaboradas novas topologias de projeto e sistemas operacionais mais eficientes. Entre as principais aplicações dessas inovações estão os novos sistemas inteligentes capazes de realizar o sensoriamento e a interação desses com o ambiente em que estão inseridos.

Um mecanismo de sensoriamento tradicionalmente estudado pela comunidade acadêmica trata-se da localização de objetos em um ambiente utilizando sensores. Pode-se, por exemplo, estimar a posição de um objeto através da captação de sinais sonoros por ele emitidos. A literatura acerca deste tema (ARAÚJO (2018), YONN et al. (2006), DI CLAUDIO and PARISI (2001), ONUMA et al. (2007) e ZHOU et al. (2008)) apresenta diversas técnicas, destacando-se algoritmos como DS (atraso-soma), CAPON, MUSIC, SRP-PHAT, TOPS, WAVES, C-SPRIT, I-MUSIC, discutidos em diversos trabalhos acadêmicos. Dessa forma, este ramo de estudo visa desenvolver técnicas e equipamentos capazes de processar sinais sonoros emitidos por uma fonte e estimar sua DOA.

Nesse contexto, este documento apresentará um estudo voltado para o desenvolvimento de um sistema de processamento capaz de estimar a localização de um objeto usando apenas sinais sonoros por ele emitidos. Este sistema é composto por um circuito com dois microfones para a captação de sinais sonoros e um microprocessador do tipo MSP430 para o processamento dos sinais captados e estimar a direção em que se encontra o objeto emissor de ondas sonoras.

A pesquisa aqui abordada é do tipo experimental, uma vez que avaliaremos o desempenho do sistema em estimar DOA. Serão avaliados dois sinais sonoros adequados ao projeto proposto: um assovio; e um sinal senoidal contínuo durante um tempo fixo de 3 minutos, gerado pelo software computacional Matlab.E, conforme será apresentado no embasamento teórico, deveremos estar atentos aos principais elementos envolvidos na propagação, captação e processamento dos sinais sonoros emitidos por uma fonte sonora.

A metodologia escolhida destina-se à criação de um modelo para o fenômeno em estudo e à criação de um software capaz de realizar a estimativa DOA. Em seguida, serão realizadas simulações alterando-se algumas variáveis de controle tais como a posição da fonte sonora ou parâmetros intrínsecos no código de processamento. Cada simulação fornecerá resultados

para a análise de desempenho do sistema pesquisado. Estes resultados serão transmitidos para periféricos como um Bluetooth (o qual poderá realizar uma transferência do resultado para um computador), um LCD (para mostrar para o usuário o resultado da estimativa) ou para um Arduino (o qual também pode transferir os dados recebidos para um computador). As estimativas DOA se mostraram satisfatórias e condições de simulação como a distância entre fonte emissora e detector e ambientes mais abertos mostraram-se de forte influência para a operação do projeto. Por fim, a avaliação dos resultados nos indica futuras alterações para a melhoria do sistema. Um arranjo com mais microfones, dispositivos periféricos com maior memória e frequências de operação maiores são alguns aspectos a serem explorados em futuros trabalhos.

1.2 Objetivos do projeto

Este projeto visa desenvolver um sistema capaz de realizar um sensoriamento no ambiente em que ele se encontra, estimando a direção de chegada das ondas sonoras produzidas por uma fonte sonora.Usaremos 2 microfones e um microprocessador do tipo MSP430 com o código adequado para que tenha como saída a estimativa da direção de chegada (DOA) dos sinais sonoros.

Nesse sentido, primeiramente estudaremos as características de um sinal sonoro, estabelecendo uma metodologia para executar os processos de captação e processamento dos sinais que se espera receber e à estimativa para a direção de chegada destes sinais.

Serão avaliados recursos de hardware e software utilizados neste trabalho a fim de fornecer ao sistema um desempenho adequado. Assim, serão elaborados códigos de programação para se realizar a conversão analógica-digital dos sinais captados, o armazenamento desses dados e cálculos de estimativa para a direção de chegada do sinal emitido.

1.3 Organização do texto

Este trabalho será desenvolvido a partir de capítulos. Assim, apresentaremos os seguintes conteúdos em cada capítulo:

Capítulo 1 - Introdução: descrição do tema em estudo; estabelecimento de objetivos para orientar o estudo e a obtenção da finalidade deste documento.

Capítulo 2 – Embasamento teórico: são apresentados os principais fundamentos teóricos necessários para se compreender os fenômenos estudados e os processos executados pelo projeto.

Capítulo 3 – Sistemas constituintes do projeto: são especificados os recursos de hardware e software utilizados neste projeto. É abordada a metodologia de operação desses recursos,

assim como os programas empregados para o processamento dos dados pelo sistema.

Capítulo 4 – Desenvolvimento do software para estimação de DOA: apresentação dos principais dados coletados durante a operação do sistema. Estes sinais indicarão a eficiência de cada etapa de funcionamento do sistema, possibilitando detectar possíveis erros e limitações a serem observadas para futuras correções.

Capítulo 5 – Resultados e análise: são apresentados os dados experimentais resultantes da execução do projeto, assim como observações a respeito dos principais fatores envolvidos em etapas determinantes no processamento do projeto.

Capítulo 6 – Conclusão e proposta para futuras análises: faz-se uma avaliação sobre os resultados apresentados no capítulo anterior. A partir desses dados pode-se estabelecer os limites de operação do projeto e verificar o desempenho do sistema em estimar DOA para os ângulos propostos. Esses dados serão considerados na formulação de uma conclusão a respeito do desempenho do projeto. A análise também será utilizada para indicar fatores e elementos relevantes a serem explorados para o desenvolvimento de trabalhos futuros.

Capítulo 2

Embasamento Teórico

2.1 Caracterização do sinal utilizado no projeto

Este sistema buscará estimar a direção em que se propaga uma onda sonora emitida por uma fonte. Por isso, inicialmente estudaremos as características que definem este sinal. Em seguida, estabeleceremos as funções de tratamento necessárias para se analisar e estimar o ângulo de chegada da onda sonora.

O sinal sonoro é uma onda mecânica longitudinal e periódica gerada pela vibração do ar próximo à fonte emissora. Com relação à sua propagação e espalhamento das ondas sonoras, dois conceitos se destacam.

A referência 4.12 nos apresenta a seguinte definição a respeito dos conceitos de frente de onda e raios de propagação.

> Frentes de onda são superfícies nas quais as oscilações produzidas pelas ondas sonoras têm o mesmo valor; essas superfícies são representadas por circunferências completas ou parciais em um desenho bidimensional de uma fonte pontual. Raios são retas perpendiculares às frentes de onda que indicam a direção de propagação das frentes de onda.

Inicialmente, faremos uma descrição dos principais aspectos físicos envolvidos no fenômeno em estudo. Assim, a fonte emissora será representada por uma pessoa testando o sistema com sua voz, o meio de transmissão será o ar e o receptor será formado pelo circuito com dois microfones responsáveis por converter o sinal sonoro em um sinal elétrico. Este sinal, então será armazenado e processado para que o sistema possa estimar a direção da fonte sonora.

O sinal gerado pelo microfone é do tipo analógico, possuindo infinitos valores de tensão correspondentes ao sinal sonoro. Para o tratamento e a interpretação desses dados conduziremos a leitura desses sinais elétricos para um conversor analógico-digital presente no microprocessador escolhido para este projeto. Estes dados serão digitalizados em valores

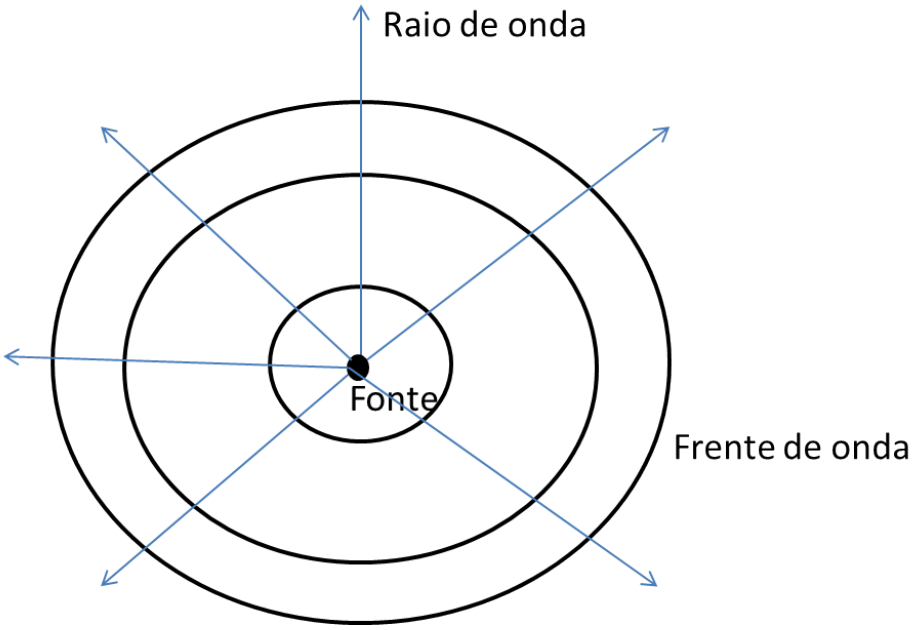


Figura 2.1: Propagação de uma onda, evidenciando as frentes de onda e os raios de propagação

discretos, os quais serão armazenados em um componente de armazenamento externo. Após o armazenamento na forma de vetores, executaremos cálculos numéricos para estimar a direção de chegada do sinal em relação ao sistema de captação sonora.

Para que o sistema possua um melhor desempenho faz-se necessário aprimorar a funcionalidade de cada componente do sistema deste projeto. Por exemplo, podemos melhorar a captação e a conversão do sinal sonoro restringindo a banda de frequência a ser captada pelo microfone aplicando-se uma filtragem sobre o sinal captado.

2.2 Sistema de captação - disposição dos sensores sonoros

Para estimar a direção do sinal utilizaremos um modelo para o sistema de captação composto por sensores sonoros (microfones) dispostos de forma linear. A captação pode ser realizada utilizando-se vários sensores; porém, faz-se necessária uma análise prévia a respeito da composição desse sistema de captação. São fatores importantes para a análise o tipo de arranjo do conjunto e a defasagem (atraso de chegada) resultante entre os sensores durante a recepção de cada um deles.

As ondas sonoras incidem sobre cada microfone com um relativo atraso entre si. Conforme será abordado posteriormente, o atraso é um parâmetro importante para estimar a direção de propagação do sinal sonoro. Portanto, com relação à quantidade de microfones empregados no sistema de captação, um microfone não será capaz de fornecer informações suficientes para o sistema. Utilizaremos um sistema com dois microfones, pois a partir dessa quantidade o sinal recebido por cada microfone apresentará atraso entre si, possibilitando

realizar a estimativa satisfatoriamente.

Para o modelo de captação observamos que cada microfone estará posicionado a uma distância d do vizinho, conforme podemos ver pela figura 2.2. A frente de onda do sinal sonoro $s(t)$ formará com a linha do eixo do sistema de captação um ângulo Θ correspondente à direção de propagação do sinal a partir de sua fonte emissora.

Trataremos a partir deste momento a respeito da influência da defasagem sobre os microfones de captação. Avaliando-se a captação do sinal de forma bidimensional, percebemos que cada microfone receberá o sinal enviado com defasagem entre si. Esta defasagem, por sua vez, origina-se de uma distância adicional percorrida pelo sinal entre cada microfone.

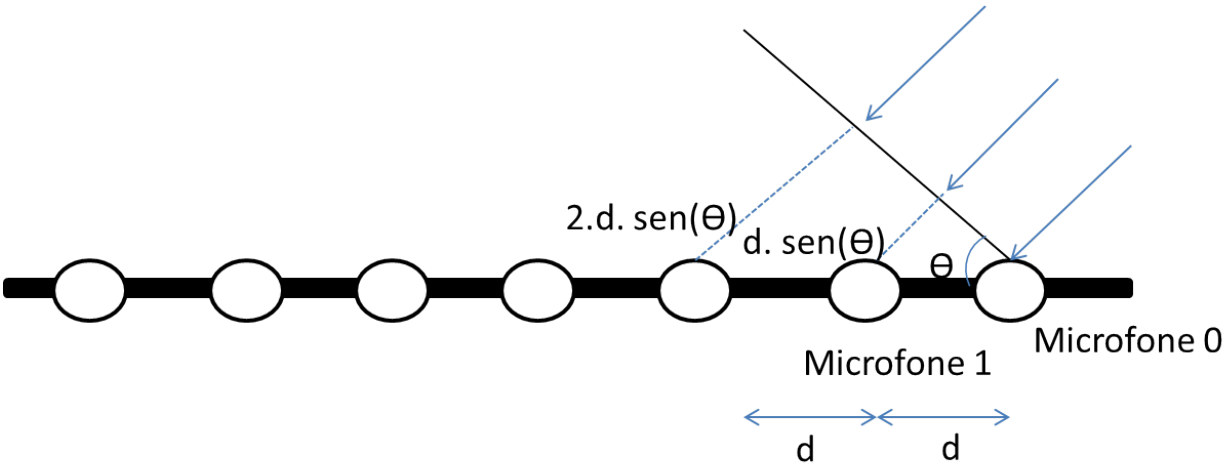


Figura 2.2: Frente de onda bidimensional incidente sobre o sistema de captação

As variáveis d e Θ determinam que essa distância adicional de defasagem será $d.sen(\Theta)$. O atraso temporal (τ), então, será calculado pela razão entre a distância adicional e a velocidade de propagação do som, v. A equação 2.1 abaixo expressa esta relação. Contudo, a razão entre v e f também pode ser expressa pela variável comprimento de onda λ.

$$\tau = \frac{d.\sin(\Theta)}{v} \tag{2.1}$$

$$\varphi = \omega.\tau = 2.\pi.f.\frac{d.\sin(\Theta)}{v} \tag{2.2}$$

A partir dessa análise inicial, discutiremos alguns aspectos referentes à da onda abordada neste trabalho. Essas características determinarão a escolha de futuros parâmetros de interesse e metodologias adotadas para se alcançar os objetivos deste projeto.

Ao utilizar dois microfones, e avaliando a equação 2.2, podemos considerar o primeiro microfone como referência para o nosso sistema. Assim, tal equação expressará valores de defasagem referentes ao segundo microfone do arranjo, distanciado a uma distância d do primeiro microfone.

Para experimentos futuros com uma quantidade maior de microfones, pode-se empregar uma análise semelhante. Escolheremos o primeiro microfone como referência. E como todos os microfones mantém a mesma distância d entre si, eles apresentarão uma defasagem proporcional à distância até o primeiro microfone.

Por exemplo, um sistema com m sensores pode ser equacionado através de uma matriz, visto na 2.4. Nesta, teremos $s_i(t)$ como saída do i-ésimo microfone, considerando $s_0(t)$ como a saída do microfone 0 (referência). A equação do sinal percebido por cada microfone é composta pelo sinal $s(t)$ captado, juntamente com sua correspondente defasagem, adicionado pela modelagem de um ruído branco $n_i(t)$. Tal sinal é limitado por um intervalo de frequências, correspondente à faixa de frequências da voz humana.

Na equação 2.4 temos em cada linha a equação que representa a função do sinal captado por cada microfone, $x_i(t)$. A primeira coluna representa esse sinal. A segunda coluna, multiplicada pelo sinal sonoro, refere-se à defasagem captada em cada microfone com relação ao sinal de referência no primeiro microfone. Conforme visto na 2.2, o deslocamento angular é proporcional à distância do microfone em relação à referência, por isso temos na segunda coluna um fator exponencial elevado a um múltiplo da defasagem. A terceira coluna, por fim, representa a modelagem de um ruído branco $n_i(t)$ mencionado anteriormente.

$$s(t) = b(t).e^{j\omega_0 t} \tag{2.3}$$

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

2.3 Posicionamento dos microfones

O posicionamento entre dois microfones adjacentes precisa obedecer a uma condição inicial importante. O ângulo de atraso (defasagem) precisa ser menor que 180 graus. Aplicando tal circunstância à equação 2.2, e sabendo-se que o maior valor possível para a função seno é 1, estabelecemos a relação. Esta relação se refere à condição de defasagem entre dois microfones adjacentes; ressaltando-se que a defasagem entre dois microfones distantes entre si poderá ser maior que 180 graus devido ao atraso (defasagem) de chegada do sinal sonoro em cada microfone, fato teoricamente já esperado.

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

Esta condição possibilita a realização adequada para o ângulo de direção do sinal de

chegada (DOA). Tal relação é tratada como versão espacial do Teorema da Amostragem de Nyquist. E, em nosso problema, esta relação será utilizada para estabelecer a distância d entre os microfones.

A onda sonora possui velocidade de $v = 340m/s$. Conforme veremos adiante, para a detecção de um sinal com frequência próxima a 1 kHz (sinal escolhido para análise), aplicados à equação 2.5, determina-se que a distância entre os microfones deverá ser menor ou igual a 17 cm. Então adotamos uma distância de 15 cm para a realização dos experimentos presentes neste trabalho.

2.4 Análise espectral do sinal sonoro

Conforme discutido anteriormente, o sinal em estudo possui uma banda espectral característica. Além disso, a equação 2.5 nos revela que componentes espectrais com diferentes frequências sofrerão distintas defasagens. Então, é conveniente escolher uma faixa restrita de frequências para destinar a operação do sistema.

Para um sinal em banda larga, o sinal recebido por cada microfone experimentará uma defasagem diferente em decorrência de componentes com frequências distintas, tornando inviável a estimação da DOA ou um elevado nível de processamento para estimar a DOA. Uma banda estreita, por sua vez, apresenta componentes com valores de frequência mais próximos, reduzindo a complexidade do sistema de estimação.

A análise espectral da voz humana mostra que esta possui muita energia na faixa de 1 kHz a 4 kHz. A figura a seguir apresenta o espectro de frequência de um sinal de voz masculina, amostrado a $f = 16kHz$ e portanto apresentando a banda até 8 kHz. Além disso, presenciamos perfis espectrais distintos entre homens e mulheres, o que geraria estimativas DOA distintas para homens e mulheres. Essas características nos indicam que a estimativa direta do sinal de voz em banda larga não apresentará uma estimativa DOA adequada.

O perfil espectral de um assovio, por sua vez, apresenta uma faixa espectral restrita e semelhante entre ambos os sexos. As componentes apresentam uma energia distribuída nas vizinhanças de uma frequência central, indicando a adequação de um modelo em banda estreita. Dessa forma, mostra-se evidente que a escolha de uma faixa restrita como um assovio deverá simplificar a estimação de defasagem entre os dois microfones e da direção de chegada. Além disso, realizamos testes experimentais utilizando o sinal de um tom senoidal com frequência central de 1 kHz, amostrados a 50 kHz durante 5 minutos.

2.5 Filtragem para minimizar o ruído branco

Conforme abordado pela equação 2.4, nosso modelo para o sinal sonoro em estudo possui a presença do ruído branco aditivo gaussiano (AWGN) durante a captação do sinal. Essa

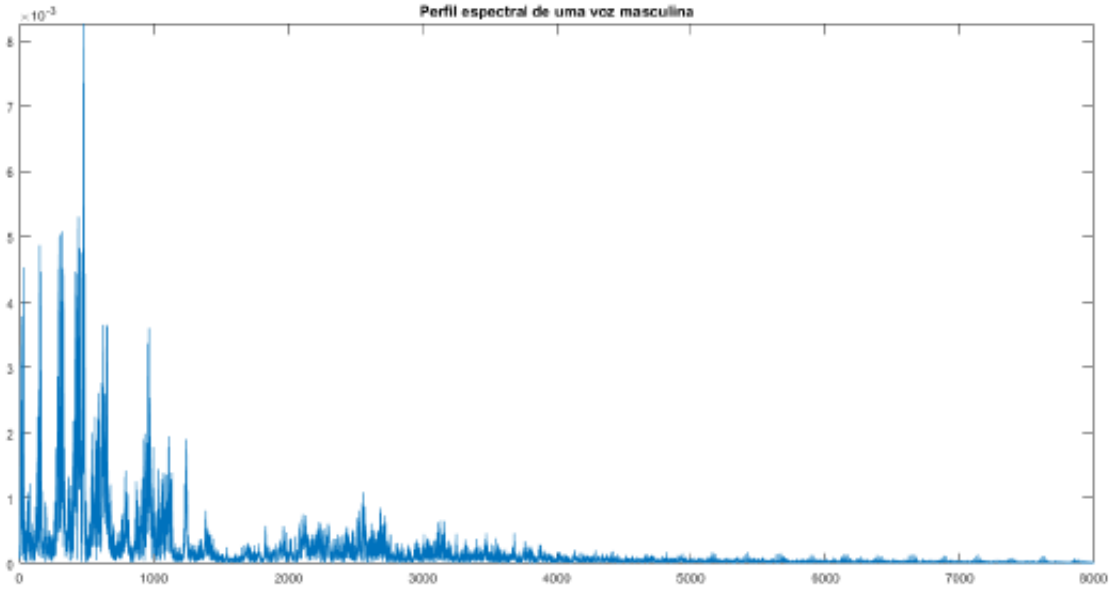


Figura 2.3: Perfil espectral de uma voz masculina

componente, então, precisa ser atenuada através da presença de um filtro digital.

Inicialmente observamos o perfil espectral do assovio recebido pelos microfones e, utilizando o software MATLAB, projetamos um filtro adequado para atenuar o ruído branco, aumentando assim a relação sinal-ruído. Através da ferramenta filterDesigner, podemos calcular a função de transferência necessária para implementar o filtro digital desejado.

O sinal analógico recebido pelo sistema de captação é transmitido ao MSP430, digitalizado e, em seguida, passa pela aplicação de uma rotina capaz de desempenhar a função do filtro projetado anteriormente. Vale ressaltar que este filtro é projetado com alguns parâmetros de interesse, os quais são apresentamos abaixo.

1. Banda passante do sinal;
2. Tipo da resposta do filtro – passa- baixa, passa-faixa, passa-alta;
3. O método do design (IIR, FIR);
4. A ordem do filtro.

Com relação a estes parâmetros faremos algumas considerações. Primeiro, a banda passante será definida após uma análise espectral prévia, verifica-se quais frequências são de interesse para o projeto, atenuando-se o sinal de frequências indesejadas. Dessa forma, teremos interesse em filtros passa-faixa ou passa-baixa, selecionando frequências entre 1 kHz e 4 kHz e com uma frequência de corte, por exemplo, de 3 kHz.

Segundo, analisaremos o método de design do filtro (IIR e FIR). Ambos são filtros digitais implementados por convolução. Filtros do tipo FIR (Finite Impulse Response) possuem

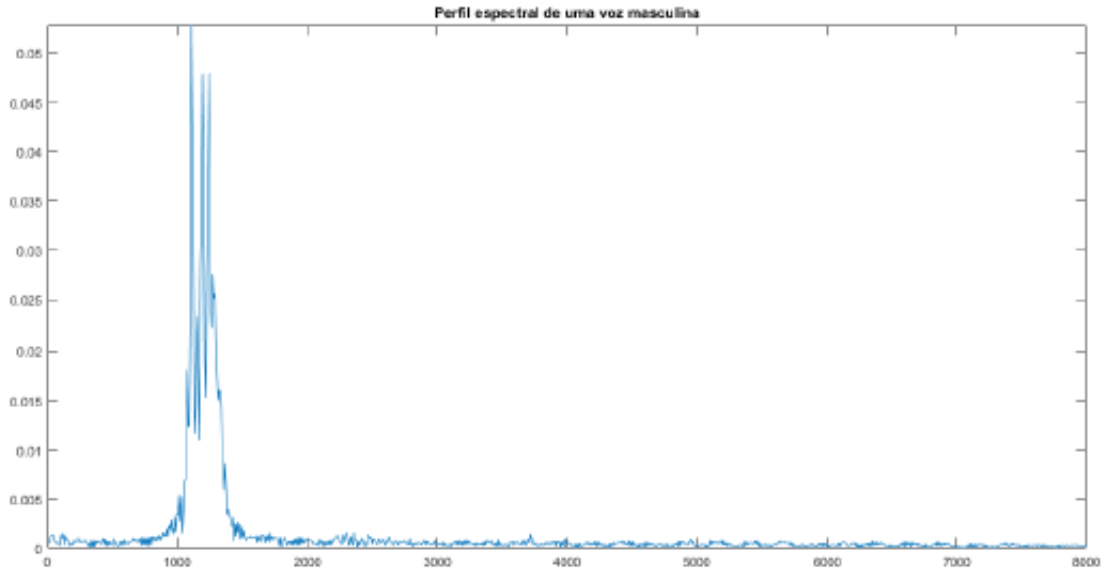


Figura 2.4: Perfil espectral de um assovio

resposta ao impulso finita e apresentam uma saída composta pela média ponderada de amostras do sinal de entrada. Filtros recursivos do tipo IIR (Infinite Impulse Response) podem ter resposta ao impulso infinitamente longa. Sua resposta é composta por senóides com amplitude que decaem exponencialmente.

Entre as características de cada filtro podemos fazer as seguintes observações. O filtro FIR: pode atingir uma melhor linearidade de fase; é estável (ou seja, para um sinal de entrada limitado, sua saída também será limitada). O Filtro IIR do tipo Butterworth apresentam respostas suaves em todas as frequências;decréscimo monotônico na frequência de corte. O Filtro IIR do tipo Chebyshev apresenta transições mais agudas com um filtro de ordem mais baixa, produz erros absolutos menores e uma execução mais veloz.

Por fim, projetamos um filtro adequado para o tratamento do sinal de assovio. Selecionamos um filtro simples que atendesse aos parâmetros de interesse do projeto. As seguintes características escolhidas foram: filtro passa-baixa, com frequência de corte de 3 kHz, projetado pelo método IIR do tipo Butterworth de 2ª ordem.

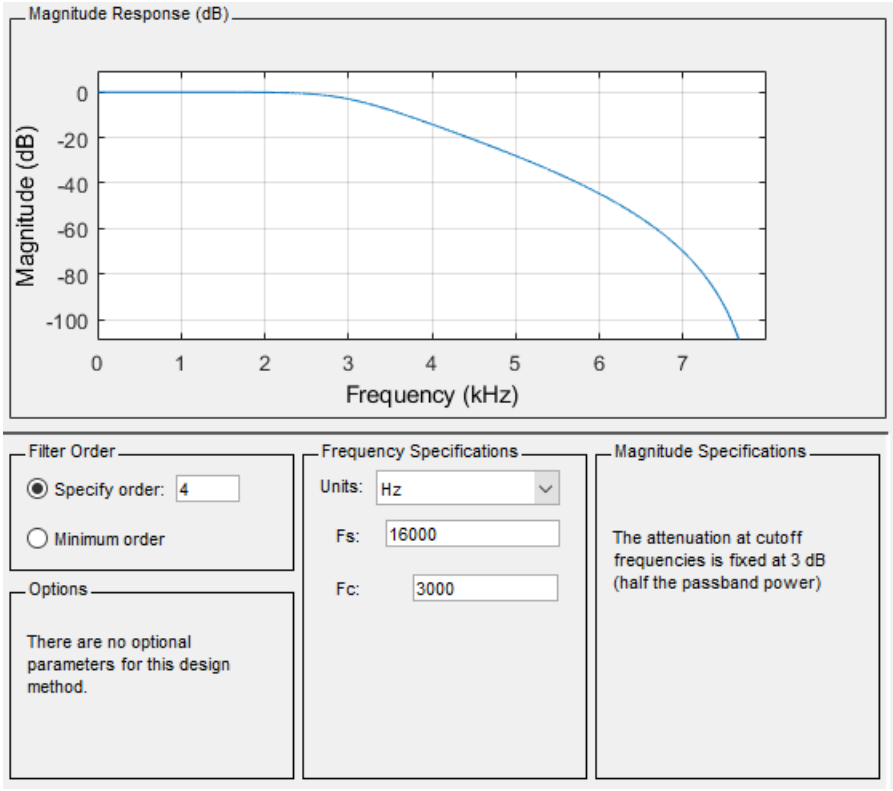


Figura 2.5: Projeto do filtro digital

Capítulo 3

Sistemas Constituintes do projeto

Apresentamos neste capítulo uma análise sistemática mostrando os principais elementos utilizados na implementação deste projeto. Podemos dividi-la em uma abordagem envolvendo aspectos relacionados ao harware e software do projeto, destacando os dois elementos fundamentais deste projeto: sistema de captação (microfone e circuito de pré-amplificação) e sistema de processamento (microprocessador MSP430 e memória externa).

3.1 Sistema de captação - microfone e circuito de pré-amplificação

Este sistema tem a função de captar o sinal sonoro e convertê-lo em um sinal elétrico com características adequadas para ser enviado para o MSP430 para posterior processamento. Os microfones utilizados foram previamente projetados em um trabalho anterior(Gontijo (2007)) voltado para o estudo de métodos para estimativa da direção de chegada (DOA) de sinais sonoros.

Os microfones usados são do tipo POM-3535L-3-R, Fabricado pela PUI Audio, Inc. Eles são omnidirecionais, dessa forma eles são capazes de captar sinais oriundos de todas as direções de maneira igual. Além disso, eles apresentam uma relação sinal-ruído de 62 dB, possuindo uma resposta linear adequada para o intervalo de frequência correspondente à faixa espectral da voz humana. A curva de resposta em frequência apresentada a seguir evidencia tal fenômeno para ondas com frequência de 20 a 20 kHz.

Associado ao microfone, aplicamos pré-amplificador, responsável pelo primeiro tratamento ao sinal de interesse. Este pré-amplificador foi desenvolvido para um projeto anterior e trabalha com alimentação simétrica. Para que o sinal por ele gerado pudesse ser empregado com o MSP430 foram feitas algumas adaptações na alimentação.

O circuito do sistema de captação é alimentado por uma fonte externa composta por 4 pilhas de 1,5V em série, fornecendo 6V ao circuito. Adicionamos uma configuração adaptada

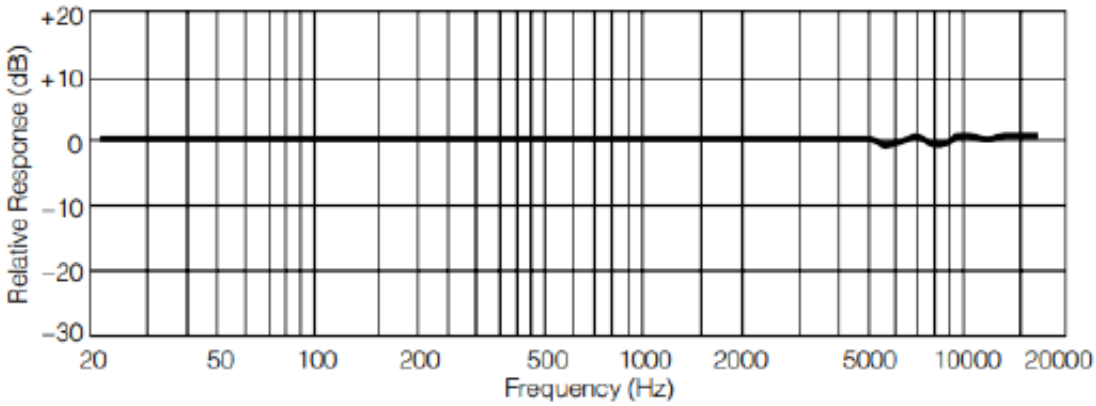


Figura 3.1: Curva de resposta em frequência de um microfone empregado

a fim de obtermos uma tensão de referência em torno de 3V. A topologia a seguir apresenta uma saída com valor médio em 3V e valores de saída entre 0 e 6V.

A seguir iremos aborda sobre a topologia utilizada para o circuito de captação. Nela utilizamos o chip LME49740, o qual possui 4 amplificadores operacionais à disposição. O sinal captado pelo microfone é direcionado para dois estágios, um inversor e outro não-inversor. A saída de cada um deles é conectada a um novo estágio comparador (um amplificador em malha aberta), o qual gera uma saída correspondente à diferença dos estágios inversor e não inversor. De fato, trata-se de um amplificador de instrumentação. Esse sinal, então passa por um estágio final responsável por aplicar um filtro passa-baixas com freqência de corte de 4,378 kHz e armazenar o sinal em um buffer de saída. Esse sinal de saída estará ligado a um canal de entrada do MSP para a leitura do sinal analógico captado pelo microfone correspondente.

3.2 MSP430

O MSP430 é um microprocessador de baixo custo e alta performance desenvolvido pela Texas Instruments. Este microprocessador possibilita grande liberdade para o usuário elaborar e executar diversos algoritmos conforme a necessidade de cada projeto. Implementamos, em linguagem C, os códigos de execução do nosso projeto, utilizando para isso o software Code Composer Studio, desenvolvido e disponibilizado gratuitamente pela Texas Instruments.

O código desenvolvido e implementado no microprocessador tem como principais funções a recepção dos sinais captados por cada microfone, a conversão analógico-digital do sinal em estudo, o armazenamento dos dados em uma memória externa e o processamento desses dados para a estimativa da direção de chegada do sinal (DOA). Este processamento aplica técnicas de média móvel, algoritmos para aplicação do filtro IIR, discutido anteriormente e projetado com o auxílio da ferramenta filterDesign presente no software MATLAB, e o cálculo final para a estimativa DOA obtida por meio da correlação dos sinais captados por

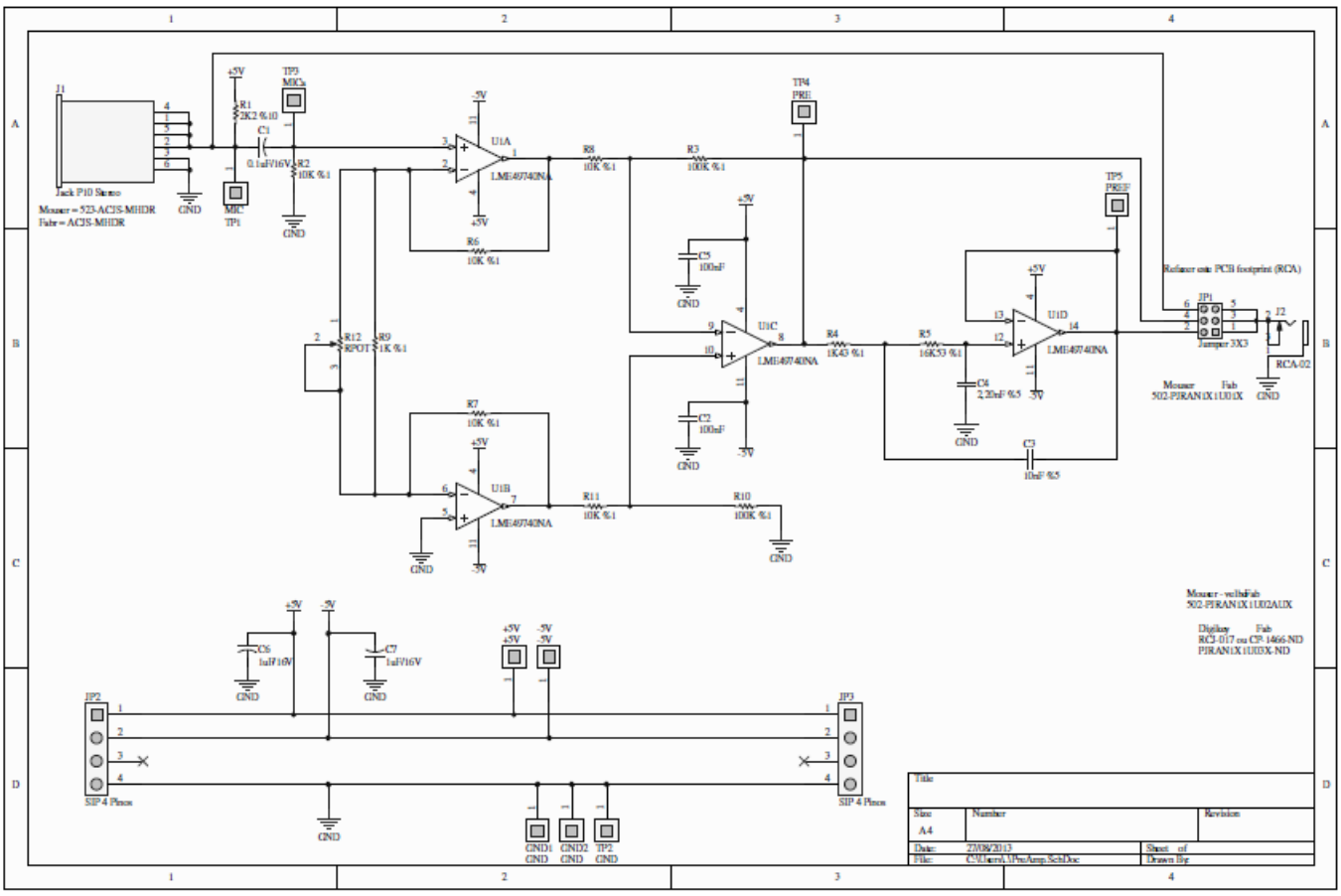


Figura 3.2: Circuito do pré-amplificador

cada microfone, resultando na estimativa do atraso temporal e ângulo de chegada do sinal. A figura 3.4 mostra um diagrama de blocos ilustrando os principais elementos envolvidos no processamento realizado pelo MSP.

3.3 Memória externa

Para que o sistema tenha um melhor desempenho e gerenciamento de memória, foram utilizados dois chips do tipo 23LC1024. Esse tipo de chip é produzido pela Microchip e atuará como periférico de memória SRAM. Este periférico possui memória de 128 KB e será responsável por armazenar os dados do sinal digitalizados pelo MSP. Portanto, de cada microfone podemos armazenar 128 K amostras de 8 bytes. Isso nos permite analisar a captação durante um período de tempo maior.

A comunicação entre o MSP e as memórias é feita por um barramento SPI. será realizada de modo serial (SPI). Os sinais, após a conversão são armazenados nestas memórias. Após a aquisição, é iniciada a fase de processamento e estimação. Estes serão lidos e passarão por uma conversão analógica-digital. Tais valores serão transmitidos serialmente para os periféricos de memória e armazenados na forma de um vetor. Após o preenchimento da memória, esses vetores passam a ser lidos da memória e processadas para estimar o atraso

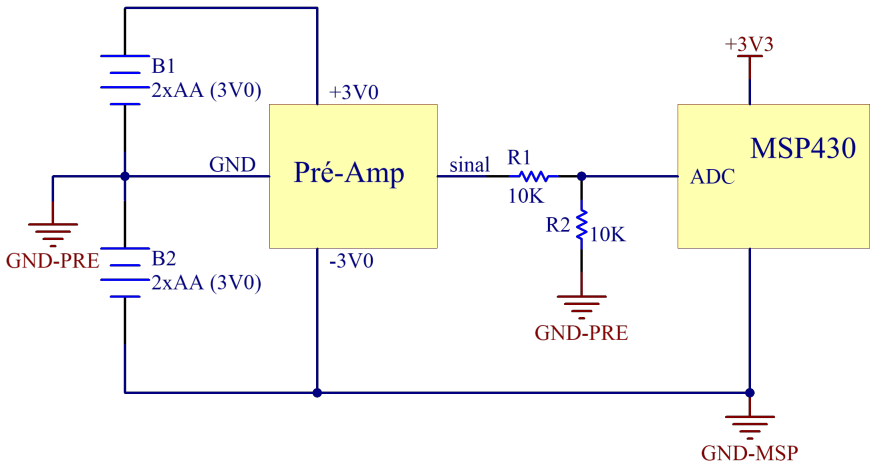


Figura 3.3: Circuito de alimentação

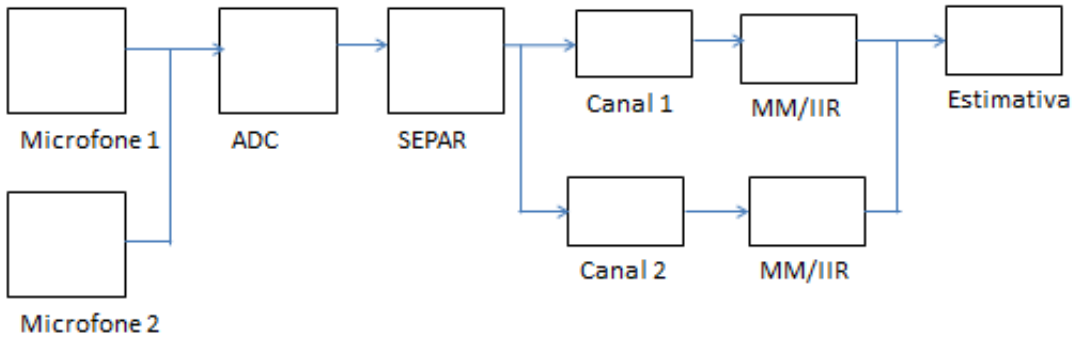


Figura 3.4: Diagrama de blocos para o sistema de processamento

temporal e a direção de chegada do sinal.

A comunicação SPI está baseada no conceito Mestre e Escravo. Nesta situação, o mestre é um elemento responsável por controlar os barramentos de comunicação; assim, ele controla a linha de sincronismo (SCK) para sinalizar o início e fim das transmissões. O escravo, por sua vez, passa a transmitir ou receber sinais quando acionado pela linha de sincronismo.

A comunicação SPI (Serial Peripheral Interface) possui um protocolo desenvolvido pela empresa Motorola. Nela, quatro linhas são utilizadas para realizar a comunicação entre mestre e escravo. A transmissão destes dois elementos ocorre de forma simultânea em ambos os sentidos, ou seja, ela é bidirecional, também chamada full-duplex.

Duas linhas são utilizadas para coordenar o funcionamento do componente mestre e do componente escravo. O mestre utiliza uma linha chamada de MOSI (Master Output Slave Input) para enviar dados para o escravo. O escravo, por sua vez, envia dados por uma linha chamada de MISO (Master Input Slave Output).

A figura 3.5 ilustra a relação mestre-escravo presente no protocolo SPI. Como exemplo de aplicação que emprega este protocolo, podemos citar o uso de um transmissor Bluetooth serial SPP-C que permite enviar dados para o PC e também processar com o Matlab a fim de

verificar análises de desempenho do sistema.

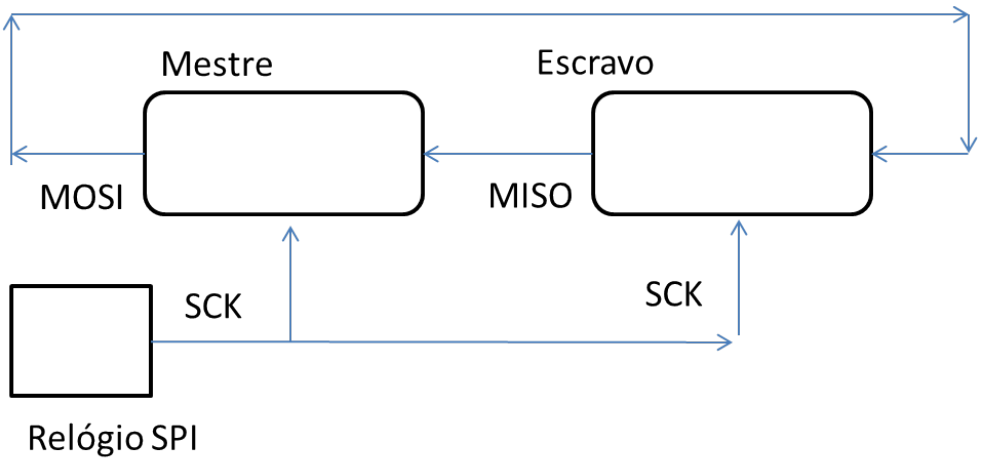


Figura 3.5: Diagrama de blocos do protocolo SPI

Outro protocolo de comum uso é o I_2C (Inter- Integrated Circuit), desenvolvido pela empresa Phillips. Este é empregado no funcionamento de um periférico do tipo LCD. Dessa forma, torna-se possível vizualizar no display do periférico a estimativa DOA após o processamento dos dados.

Este protocolo estabelece uma comunicação síncrona e bidirecional entre o microprocessador e o periférico utilizando 2 condutores: a linha de sincronismo, Serial Clock (SCL); e a linha de transmissão de dados, Serial Data (SDA). Com esta configuração pode-se gerenciar a comunicação entre um elemento atuando como mestre e vários elementos atuando como escravos.

Conforme visto anteriormente, o mestre gerencia a comunicação. Inicialmente o mestre precisa verificar se alguma transmissão está sendo executada, iniciando uma nova transmissão assim que o barramento esteja livre. Em seguida, o mestre deve iniciar a comunicação enviando um sinal de start para os escravos da linha. Os escravos passam para o modo de espera. Então o mestre transmite o endereço do escravo com quem será realizada a comunicação. Além disso, o mestre envia um bit indicando se o escravo vai transmitir ou enviar dados. O escravo correspondente envia um sinal de acknowledge (ACK) para iniciar a transmissão de dados pela linha SDA. A transmissão é gerenciada por meio de sinais de controle. O receptor deve enviar um sinal de ACK a cada transmissão de byte e o mestre deve enviar um sinal de stop para finalizar a comunicação.

Para que o projeto execute sua função corretamente, deve-se estabelecer as condições de operação do microprocessador MSP430. O algoritmo elaborado através do software Code Composer definirá quais funcionalidades do microprocessador serão utilizadas no projeto, tais como a configuração de pinos de entrada e saída, configurações de funcionamento dos sistemas de transmissão da leitura de valores captados pelos microfones, transmissão para acionamento do LCD e transmissão de outros periféricos tais como um transmissor Blueto-oth. Cabe ressaltar a configuração do Acesso Direto à Memória (DMA), recurso capaz de

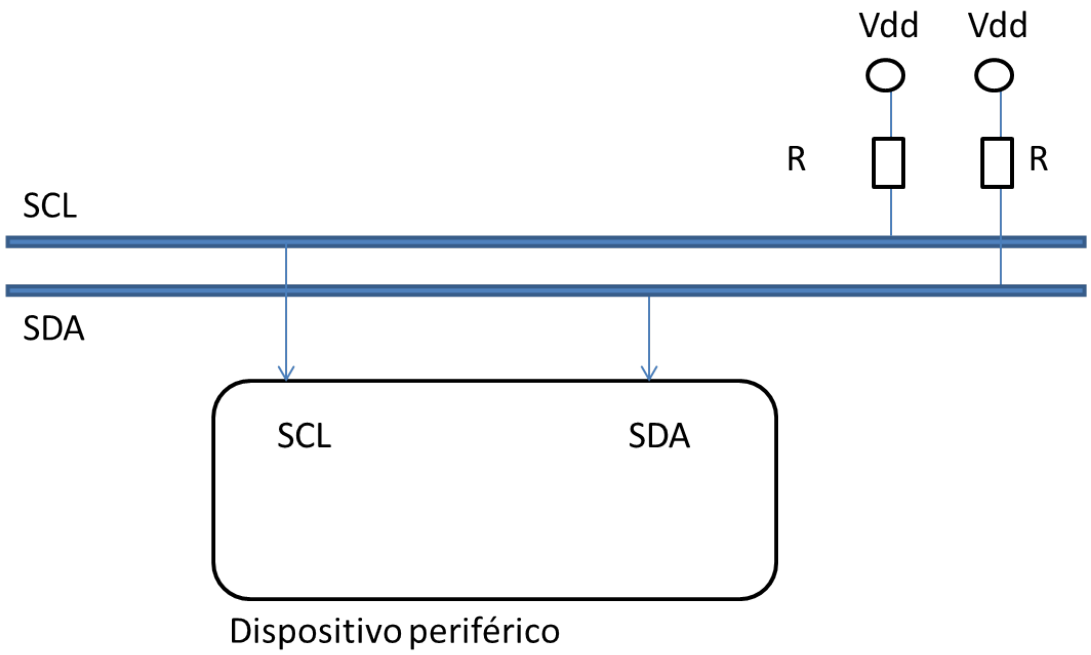


Figura 3.6: Diagrama de blocos do protocolo I2C

melhorar a velocidade de armazenamento do processamento em execução.

Vale ressaltar a importância da gestão de dados através do DMA. Este recurso permite que periféricos acessem diretamente a memória RAM sem ocupar o processador. Essa característica possibilita uma melhora de desempenho na transferência de grande quantidade de dados, realizando a cópia de um bloco de memória e a transferência deste de um dispositivo para outro.

Com relação ao sistema, podemos apresentar algumas observações a respeito das configurações do MSP430. Acionamos os LEDs para sinalizar a operação do sistema. Um LCD foi configurado para mostrar o processo executado pelo sistema, assim como o resultado da estimativa executada. Este componente foi configurado para funcionar a partir da comunicação $I_2C(UCSB0)$. Os pinos de comunicação (P3.0 e P3.1) foram configurados para operarem a 400 kbps.

A comunicação entre o MSP430 e a memória externa é realizada de acordo com o protocolo $SPI(USCI - B1)$. A configuração dessa transmissão caracterizou o MSP430 atuando como mestre, estando a operação executada com uma velocidade definida pelo clock SMCLK.Serão utilizados dois canais, cada um referente aos dados de cada microfone. A conversão A/D do MSP430 também possui uma velocidade condicionada pela frequência de um clock SMCLK com saída no modo single.

Em uma visão geral, o código criado para o projeto configura as portas de entrada, sincronismo, transmissão e recepção. Elaboramos constantes e funções para estabelecer e gerenciar o funcionamento de rotinas como timers, conversão analógico-digital, transmissão serial (SPI), leitura e escrita na memória e cálculo para correlação temporal e estimativa

DOA. Essa medida visa destacar variáveis de uso recorrente e organizar as principais funções utilizadas pelo código principal do projeto. Isto também facilita o processo de detecção de erros durante a avaliação da eficiência do código.

Capítulo 4

Desenvolvimento do software para estimação de DOA

Abordaremos neste capítulo os principais aspectos envolvidos na elaboração do código implementado neste projeto.

4.1 Firmware

Para a fase de criação do código principal do projeto precisamos listar as funções desempenhadas pelo sistema e criar scripts auxiliares responsáveis por executar as principais funções do código. Assim, são criados códigos para execução de funções e estabelecimento de constantes globais.

A inicialização do código, então, deve carregar o valor das constantes utilizadas ao longo da execução e executar as funções de configuração de elementos de controle tais como pinos de entrada e saída, botões, timers, interrupções, conversor analógico-digital (A/D), transmissão seria (SPI) MSP-memória externa.

A configuração de pinos e timers é feita através da configuração de portas programáveis (GPIO, do inglês General Purpose Input/Output). São configurados dois botões para se reinicializar a execução do sistema e selecionar o modo de operação do mesmo. Leds foram utilizados para indicar etapas de processamento do sistema.

Para que o processo de conversão analógico-digital tenha maior velocidade, faremos a configuração do conversor A/D e o recurso de DMA (do inglês, Direct Memory Access). Dessa forma, o conversor fará a digitalização do sinal recebido, salvando-o diretamente na memória do MSP.

O controlador de DMA assume o barramento da CPU e transfere os dados da memória do ADC diretamente para a memória do MSP. Além de configurarmos o modo de operação de cada um desses recursos, configurarmos os timers TA0.1 para a operação do conversor

A/D e do DMA.

Para a apresentação de etapas de processamento e estimativa final DOA, realizamos a configuração de um LCD. Acionamos este periférico através do barramento I2C pelo MSP. Configuramos os pinos P3.0 e P3.1 para atuarem como SDA e SCL utilizados na transmissão serial.

4.2 Método de estimação de DOA

Conforme foi apresentado na seção 2.3, o sistema de captação possui dois microfones posicionados a uma distância de 15 cm. Esta é uma condição necessária para que o ângulo de atraso (defasagem) seja menor que 180 graus. A disposição do arranjo é mostrada abaixo.

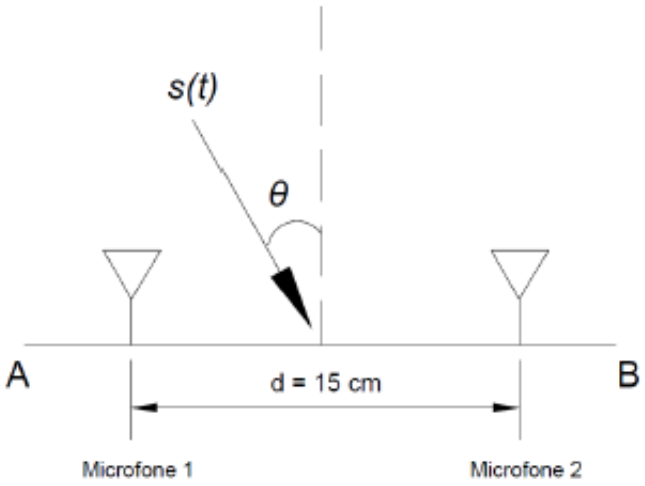


Figura 4.1: Disposição dos microfones

Durante a abordagem apresentada na seção 2.2, podemos equacionar o sinal recebido pelos microfones como uma função do sinal original multiplicado por um ganho oriundo dos amplificadores associados aos microfones e somados a um ruído gaussiano. Além disso, o sinal do microfone 2 apresenta um atraso temporal em relação ao microfone 1. Estas expressões são apresentadas logo abaixo.

$$y_1(t) = G_1(t).s(t) + n1(t) \tag{4.1}$$

$$y_2(t) = G_2(t).s(t) + n2(t) \tag{4.2}$$

O método de estimação aplica métodos númericos discretos para estimar o valor de DOA a partir do resultado da conversão A/D. O valor de cada sinal é amostrado a uma taxa de 50 kHz. Como a estimativa DOA é obtida através de um processo de correlação entre os sinais referentes a cada microfone, a estimativa se torna mais eficiente ao realizarmos este processo

com um conjuto grande de valores. Então podemos armazenar o conjunto de valores digitalizados na forma de um vetor, assim será mais fácil o armazenamento e o cálculo elemento a elemento dos sinais registrados.

O processamento para estimação consiste em determinar o quanto um sinal está atrasado em relação ao outro. Este atraso é medido em número de amostras e no presente texto será indicado pela letra "k".

$$\tau = \frac{k}{f_{amost}} = \frac{d.\sin(\Theta)}{v} \tag{4.3}$$

$$\tau = \frac{k}{f_{amost}} = \frac{d.\sin(\Theta)}{v} \tag{4.4}$$

$$\Theta = \arcsin(\frac{k.v}{d.f_{amost}}) \tag{4.5}$$

Com k = 1, teremos a resolução do sistema de estimação que então, como se vê pela equação, é dependente da frequência de amostragem. Substituindo-se os parâmetros do ensaio, ou seja, v = 34.000 cm/s e d = 15 cm temos a relação entre resolução e frequência de amostragem.

$$\Theta = \arcsin(\frac{34000}{15.f_{amost}}) = \arcsin(\frac{6800}{3.f_{amost}}) \tag{4.6}$$

Tabela 4.1: Relação entre resolução e frequência de amostragem

f_{amost}	Θ
$10kHz$	13º
$20kHz$	6,5º
$25kHz$	5,2º
$30kHz$	4,3º
$40kHz$	3,2º
$50kHz$	2,6º

Com o atual protótipo se chegou a uma frequência de amostragem de 50 kHz, porém, como são dois canais, na verdade o valor é de 25 kHz. Em suma a resolução esperada é de 5,2 graus.

Este cálculo é realizado ao final do processo de quantização e armazenamento dos valores, por isso ele está sujeito a uma série de erros de medida e configuram o cálculo como uma estimativa. Faz-se, então, necessário um tratamento dos dados para os cálculos finais apresentem um erro mínimo. Os dados salvos em forma de vetores apresentaram melhores resultados em nossos ensaios após a implementação das seguintes técnicas.

1. Atenuação de ruído com a aplicação de filtros digitais;
2. Identificação da faixa de maior energia;

3. Interpolação por spline cúbica.

4.3 Atenuação de ruído com a aplicação de filtros digitais

Os dados salvos na forma de vetor estão sujeito a ruídos inerentes aos processos de conversão presentes na captação dos microfones e na digitalização do conversor A/D. Os ruídos adicionados ao sinal original representam um elevado risco para a eficiência da estimativa de DOA. Por isso, é recomendável a aplicação de filtros capazes de atenuar tais ruídos.

Primeiro desenvolvemos um filtro do tipo FIR – resposta finita ao impulso, do inglês Finite Impulse Response. Este filtro é aplicado após a conversão analógico-digital e atua sobre o vetor recém-criado. O vetor resultante terá como novos elemetos uma média dos n elementos adjacentes do vetor anterior e será armazenado na memória externa. Esta técnica é conhecida como Média Móvel e constitui um processo para suavizar flutuações rápidas e destacar tendências presentes em todo conjunto amostrado. O valor de n elementos adjacentes envolvidos nesse cálculo é conhecido como grau da média móvel e neste trabalho equivale a $n = 2$.

$$M_{movel} = \frac{1}{n}.\sum_{i=1}^{n} v_i \tag{4.7}$$

Após o filtro média móvel, aplicamos um filtro IIR com a intenção de selecionar a banda de frequências do assovio, que está entre 1 kHz e 2 kHz. Esse filtro foi apresentado na seção 2.5.

4.4 Identificação do sinal no tempo

Para que o estimador possa funcionar corretamente, ele deve evitar as amostras sem o sinal do assovio. Estes intervalos sem sinal serão chamados de intervalos de silêncio.

A partir dos dados salvos na memória externa na forma de vetor, detectamos a faixa de maior energia do sinal já amostrado em tempo discreto. Para essa etapa calculamos a média dos elementos deste vetor e com esse valor avaliamos a intensidade de cada amostra. Calculamos a diferença entre o valor absoluto de cada elemento e a média calculada anteriormente. Estabelecemos um valor mínimo para detecção. Caso a diferença seja maior que o mínimo de detecção, trataremos este ponto como o início da faixa de maior energia. Utilizamos, então, uma quantidade de elementos localizados após esta posição para realizar os cálculos de estimação.

4.5 Interpolação linear

Já vimos que a frequência de amostragem limita a resolução do estimador de direção de chegada. Com o MSP430 chegamos ao valor de 50 kHz, correspondendo da 25 kHz por canal. Para o presente trabalho este foi considerado um valor razoável, que resultou na resolução de 5,2 graus. Pretende-se utilizar uma frequência de amostragem elevada, uma vez que esta medida aumenta a resolução dos dados. Porém isto implica na adição de ruídos que podem comprometer a aquisição de dados.

Para diminuirmos o efeito desses ruídos, desenvolvemos uma rotina responsável por aplicar polinômios de interpolação do tipo spline cúbica. Estas funções são polinômios de 3º grau que serão aplicadas sobre o vetor para criar um novo vetor com variações de valor mais suavizadas entre seus elementos.

A rotina criada tomará os elementos do vetor da faixa com mais energia copiando-os para um novo vetor em posições pares. Para as posições ímpares será feito o cálculo da interpolação, contida na equação 4.7, medindo a média entre os valores adjacentes localizados nas posições pares. Este processo gera um vetor com o dobro de tamanho e taxa de amostragem.

$$v_n^{'} = \frac{v_{n-1} + v_{n+1}}{2} \tag{4.8}$$

4.6 Estimativa para o atraso temporal

Para calcularmos o atraso temporal entre os sinais de cada microfone precisamos primeiramente analisar o conceito de atraso, o qual equivale ao momento em que o mesmo sinal deslocado temporalmente passa a possuir os mesmos valores do sinal original. Dessa forma, faz-se necessário inspecionar os elementos dos vetores que armazenam os sinais de cada microfone.

Inicalmente tais vetores são diferentes entre si, uma vez que representam o mesmo sinal captados em diferentes instantes. Ao deslocarmos o vetor do microfone esquerdo em relação ao vetor do microfone direito, podemos avaliar a similiaridade entre estes conjuntos. O deslocamento sobre as amostras que gerar a menor diferença entre os vetores, indicará o atraso temporal sofrido pelo sinal sonoro ao ser captado por cada microfone.

A correlação é utilizada com a finalidade de se verificar a similiaridade entre os vetores armazenados na memória externa. A rotina empregada neste projeto tem como objetivo avaliar o erro quadrático médio entre os dois vetores, realizando a operação elemento a elemento. Realiza-se novamente o cálculo do erro médio quadrático para versões deslocadas de um dos vetores, registrando-se o deslocamento que resulte no menor erro quadrático médio, ou seja, o deslocamento temporal entre os sinais recebidos por cada microfone.

A rotina primeiramente calcula a diferença entre os elementos de uma mesma posição,

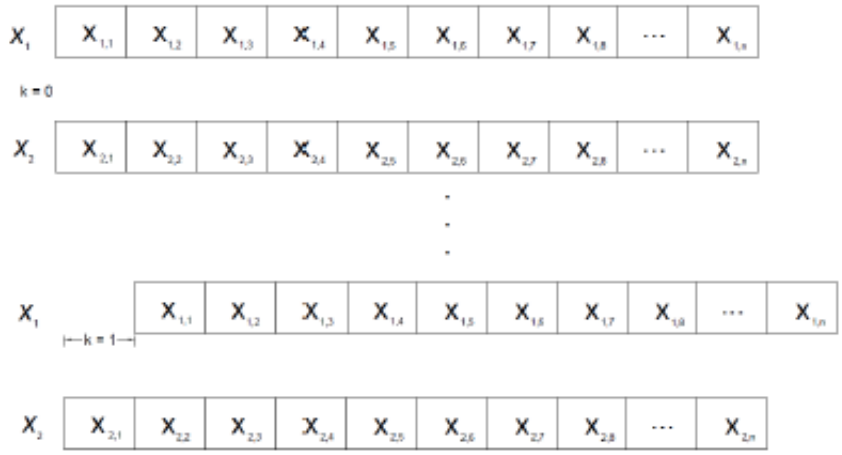


Figura 4.2: Ilustração de um deslocamento entre elementos de um vetor para o cálculo de correlação temporal entre dois vetores

eleva ao quadrado, soma estes valores, calcula a raiz quadrada e divide pelo número de elementos do vetor (n). O resultado é guardado em uma variável de Erro Mínimo e o deslocamento é guardado em outra variável para deslocamento final, a qual posteriormente será utilizada para calcular o atraso temporal. Essa operação é feita novamente para um número de 0 até n-1 deslocamentos. O resultado final é comparado com a variável de Erro Mínimo atual e caso apresente um valor menor, será atualizado para o novo valor mínimo e o deslocamento final correspondente. Esta relação é expressa na equação abaixo. As variáveis $v1_i$ e $v2_i$ indicam os elementos na posição i de cada microfone.

$$Erro = \left|\overrightarrow{v1} - \overrightarrow{v2}\right| = \frac{1}{n}.\sqrt{\sum_{i=1}^{n} |v1_i - v2_i|^2} \tag{4.9}$$

4.7 Observações finais para o projeto

Ao longo deste trabalho abordamos os principais fatores envolvidos no funcionamento do sistema. Inicialmente discutimos a respeito da disposição dos microfones, os quais estarão 15 cm de distância entre si. É importante ressaltar que durante um ensaio a fonte sonora deverá estar direcionada para os microfones e a uma distância capaz de se detectado pelos microfones.

Outro aspecto a se considerar é a taxa de amostragem. Como foi visto na seção 4.2, uma maior taxa de amostragem possibilita ao sistema uma maior resolução para a estimação de DOA. Pode-se então variar a taxa de amostragem adotada no conversor A/D a fim de se obter uma estimativa com melhor resolução. Com o aumento na taxa de amostragem, intensifica-se a influência dos ruídos oriundos deste conversor sobre os dados digitalizados, necessitando da atuação de um filtro do tipo FIR para atenuar a presença de ruídos.

A partir da equação 4.5 iniciamos uma análise a respeito das limitações do sistema. A frequência de amostragem foi configurada com o valor de 50 kHz, o que equivale a 25 kHz por canal. Assim, vamos avaliar o ângulo mínimo estimado pelo sistema. Nessa situação podemos considerar uma distância $d = 15cm$, $k = 1$ atraso relativo entre os sinais e chegaremos à conclusão de que o sistema é capaz de detectar um ângulo mínimo de 5,2 graus, o que indica uma resolução razoável para um sistema.

O valor de k indica o atraso relativo entre os sinais. Para um valor mínimo de k, obtemos o valor mínimo de ângulo estimado pelo sistema. Um valor crescente de atraso, por sua vez, terá como estimativa um ângulo maior de DOA. A equação 4.5, porém, possui como condição fundamental o fato de possuir um valor máximo de 1 para a função seno. Assim, a condição do argumento implica na relação abaixo para k. Isto nos indica que o sistema possui a limitação de estimar um ângulo máximo de 85,81 graus para um atraso de $k = 22$. Por fim, este sistema possui resolução de 2,6 graus e ângulo máximo para estimação de 85,81 graus, oferecendo um desempenho muito satisfatório.

$$\Theta = \arcsin(\frac{k.v}{d.f_{amost}}) \tag{4.10}$$

$$\frac{k.v}{d.f_{amost}} < 1 \tag{4.11}$$

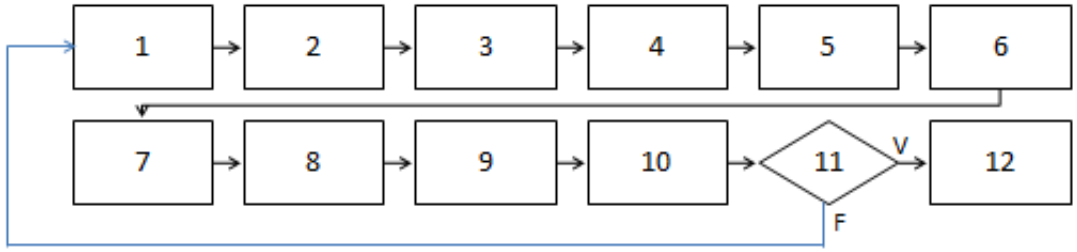
$$k < \frac{d.f_{amost}}{v} \tag{4.12}$$

Substituindo-se os valores $d = 15cm$, $f_{amost} = 25kHz$ e $v = 340m/s$

$$k < \frac{15.25000}{34000} = 11 \tag{4.13}$$

$$\Theta = \arcsin(\frac{10.34000}{15.25000}) = 65^{o} \tag{4.14}$$

Conhecendo as limitações do sistema, seus componentes e funcionalidades podemos realizar ensaios para verificar o funcionamento do sistema. Um diagrama de blocos é apresentado abaixo para ilustrar os componentes e funções constituintes do sistema. Isto facilita o entendimento do sistema e facilita a detecção de eventuais erros durante uma operação.



1- Inicialização do sistema
2- Captação, amplificação e pré-filtragem dos sinais analógicos dos dois microfones
3- Leitura e conversão A/D dos sinais analógicos pelo MSP
4 – Aplicação da técnica de média móvel sobre o sinal digitalizado– filtro FIR
5 – Escrita dos dados na forma de vetor na memória externa
6 – Leitura dos dados, aplicação do filtro IIR e escrita dos valores resultantes na memória externa
7- Leitura dos dados. Cálculo da média e média absoluta dos dados
8- Identificação da faixa de maior energia
9- Interpolação spline cúbica na faixa de maior energia
10 – Correlação aplicada sobre os elementos do vetor de dados para estimar o atraso
11- Cálculo para a estimativa do ângulo de chegada do sinal. Verificar a validade dos resultados
12- Apresentação das etapas sendo executadas e do resultado final no LCD

Figura 4.3: Diagrama referente ao funcionamento do estimador DOA

Capítulo 5

Resultados e análise

Após a apresentação dos principais aspectos envolvidos no funcionamento deste projeto, realizaremos ensaios para verificarmos o desempenho do sistema, assim como suas limitações.

5.1 Condições para a realização dos experimentos

Inicialmente realizamos testes com sinais sonoros de assovio de curta duração e com uma curta distância entre a fonte emissora e o sistema de estimação. Estes testes mostram-se satisfatórios e indicaram um sucesso inicial do projeto. Em seguida, foram realizados testes de maior duração temporal utilizando um sinal próximo a uma senóide com frequência de 1 kHz, amostrado a uma frequência de 50 kHz. Para o funcionamento do sistema foi carregado o programa DOA desenvolvido no software Code Composer e manteve-se a conexão do MSP com o computador para energizar este microprocessador. Em seguida foi realizada a alimentação do circuito de captação com baterias antes de iniciarmos os ensaios.

Pretende-se verificar o desempenho do sistema para estimar a direção de uma fonte sonora. Foram feitos ensaios para a determinação de ângulos com valor entre -70 a 70 graus, em passos de 10 graus, contabilizando 15 estimativas. Os resultados foram transmitidos de forma serial para um microcontrolador Arduino, o qual foi responsável por transmitir por canal bluetooth os resultados para verificar a validade dos dados estimados através do software MATLAB. Os ensaios mostraram a influência dos processos de filtragem sobre os sinais captados e processados a partir dos microfones 1 e 2. Os sinais, assim que são adquiridos são acompanhados por um forte ruído Gaussiano. Percebe-se um nível de tensão diferente de zero durante o período de amostragem, registrando-se intensidades de tensão mais elevadas durante a captação do sinal. A aplicação do filtro reduz consideravelmente as tensões oriundas do ruído presente durante o processo de amostragem e melhoram a estimativa feita pelo sistema.

Após o processo de conversão analógico-digital e armazenamento na memória externa,

passa-se a operar sobre esses dados. A determinação da faixa de maior energia é feita através de uma rotina específica. Esta começa calculando-se a média dos valores registrados no vetor referente a cada sinal. Faz-se a subtração entre o valor de cada elemento do vetor e a média calculada. Calcula-se o valor absoluto desse resultado e divide-se pelo valor máximo presente no vetor do sinal.

O resultado dessa rotina fornece um valor positivo, o qual será comparado com um valor de referência a partir do qual pode-se verificar a existência de um sinal presente no vetor. Este processo realiza uma boa estimativa para a faixa de maior energia quando aplicamos um valor de referência de 200, definida como uma constante global em nosso programa. Valores acima da referência tem seu endereço gravado para que o programa possa trabalhar com a faixa de maior energia do sinal amostrado.

O processo de correlação seguinte, necessário para a estimação do ângulo de chegada do sinal, passa a ser realizado de forma repetitiva sobre os intervalos de maior energia. Finalmente são registrados os resultados para os experimentos propostos.

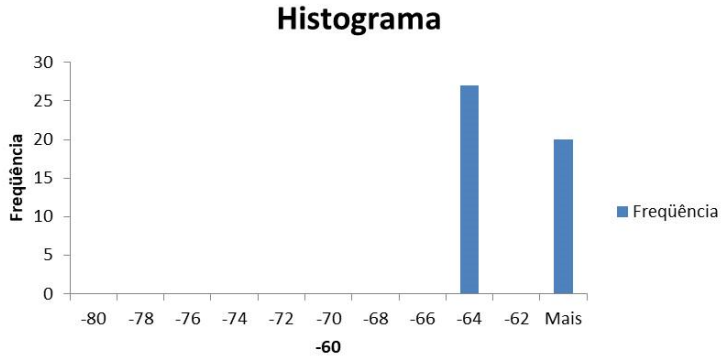


Figura 5.1: Histograma para a estimação de -70º

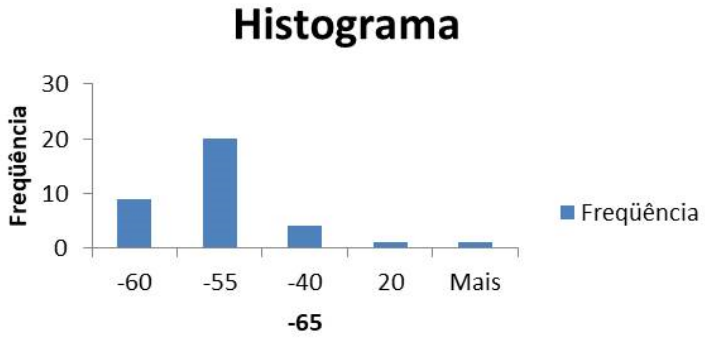


Figura 5.2: Histograma para a estimação de -60º

A plotagem dos resultados do ensaio são mostrados a seguir. Nela percebemos de uma forma geral uma grande diferença ente os valores estimados e os valores de referência esperados. Destaca-se a precisão da estimativa para valores específicos como 10 graus, possuindo

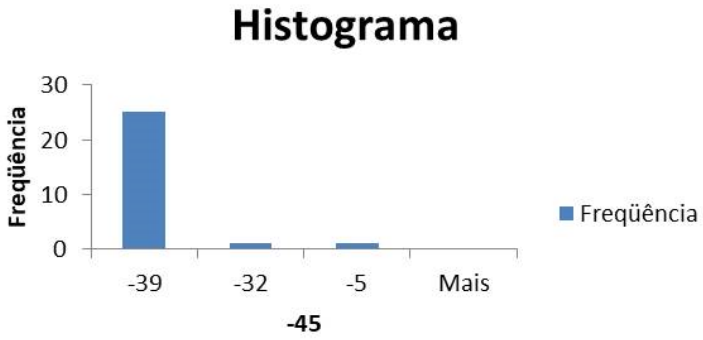


Figura 5.3: Histograma para a estimação de -40º

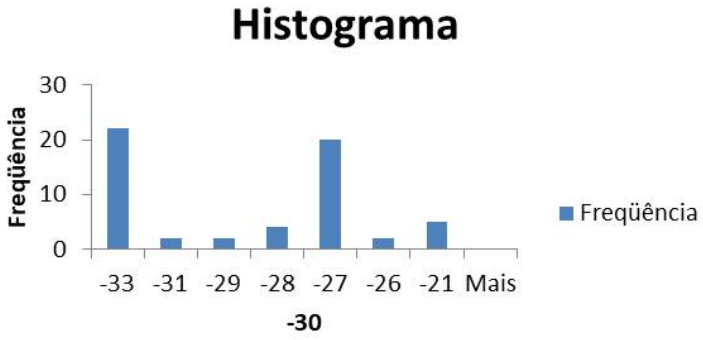


Figura 5.4: Histograma para a estimação de -30º

uma melhor precisão para ângulos entre 10 e 30 graus. Este sistema apresenta maiores distorções para valores superiores a 40 graus e ângulos próximos a zero grau.

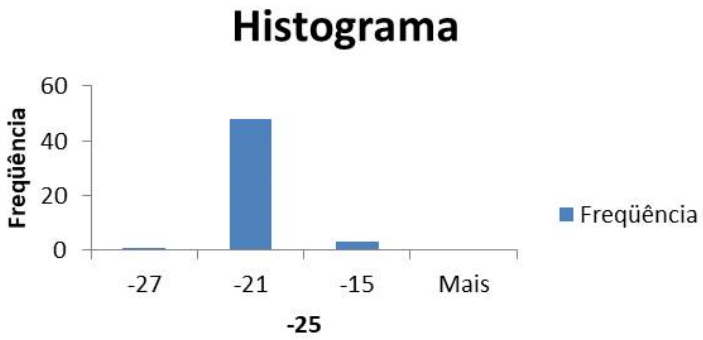


Figura 5.5: Histograma para a estimação de -20º

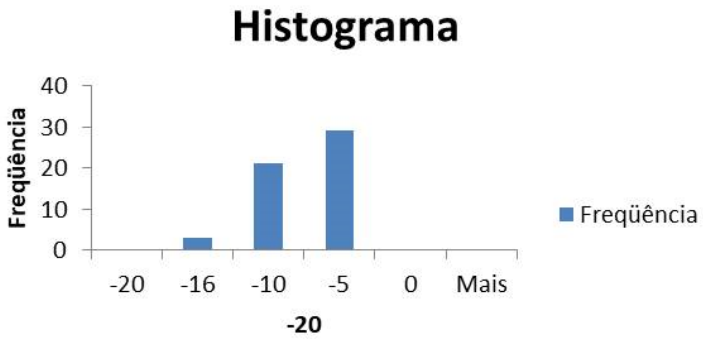


Figura 5.6: Histograma para a estimação de -10º

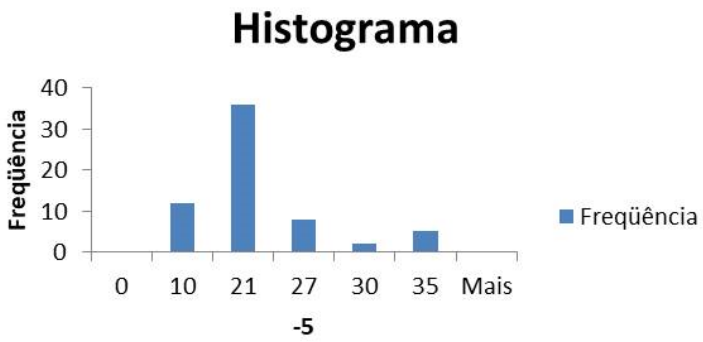


Figura 5.7: Histograma para a estimação de 0º

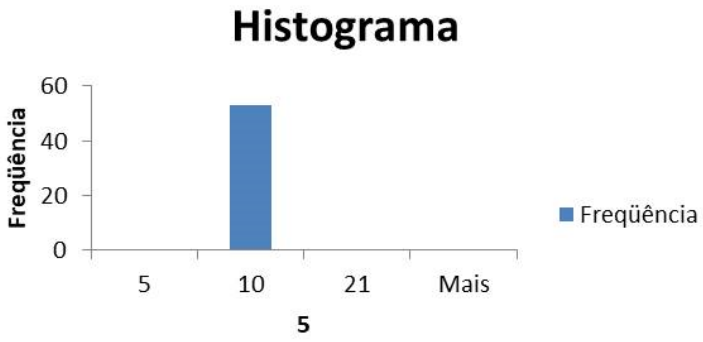


Figura 5.8: Histograma para a estimação de 10º

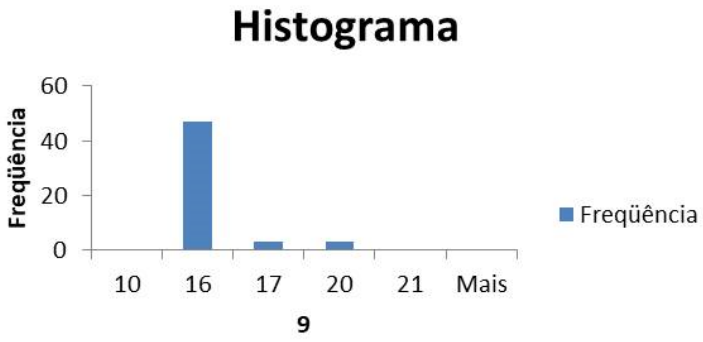


Figura 5.9: Histograma para a estimação de 20º

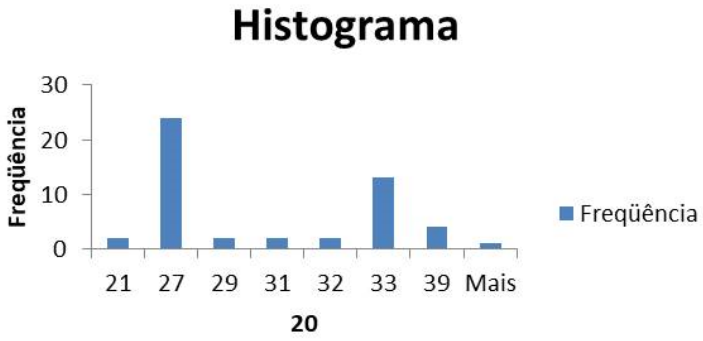


Figura 5.10: Histograma para a estimação de 30º

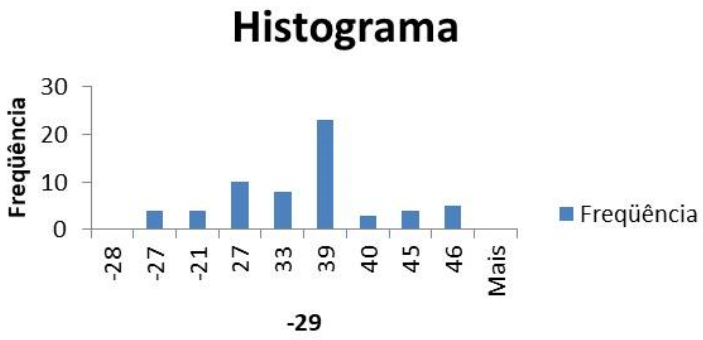


Figura 5.11: Histograma para a estimação de 40º

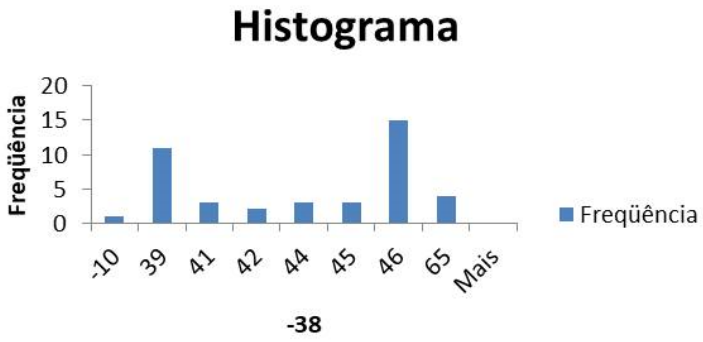


Figura 5.12: Histograma para a estimação de 50º

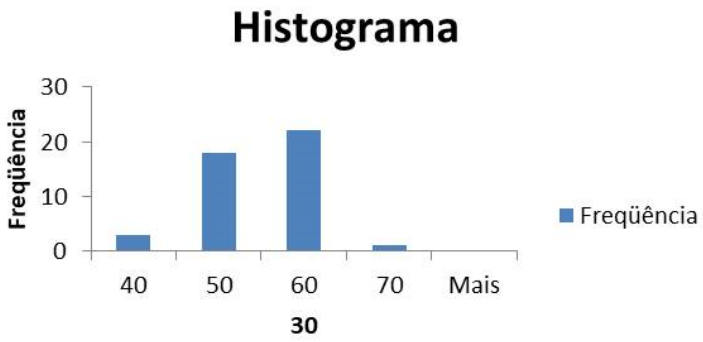


Figura 5.13: Histograma para a estimação de 60º

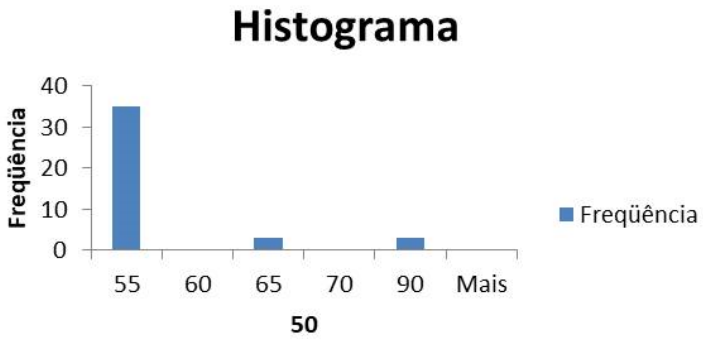


Figura 5.14: Histograma para a estimação de 70º

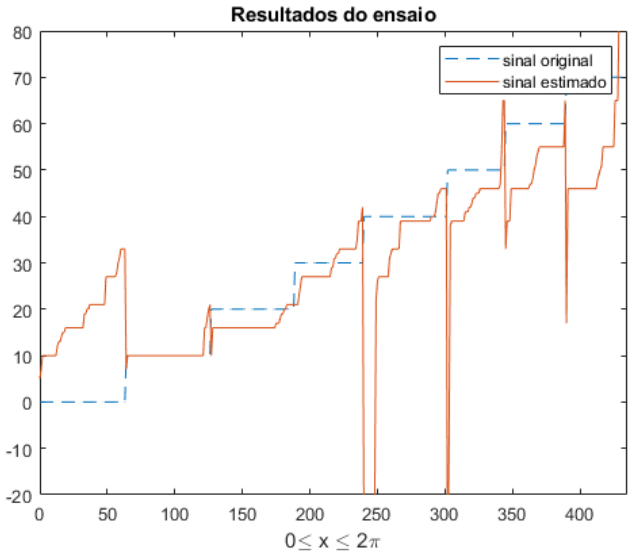


Figura 5.15: Curvas evidenciando os ângulos de referência e os ângulos estimadas pelo sistema

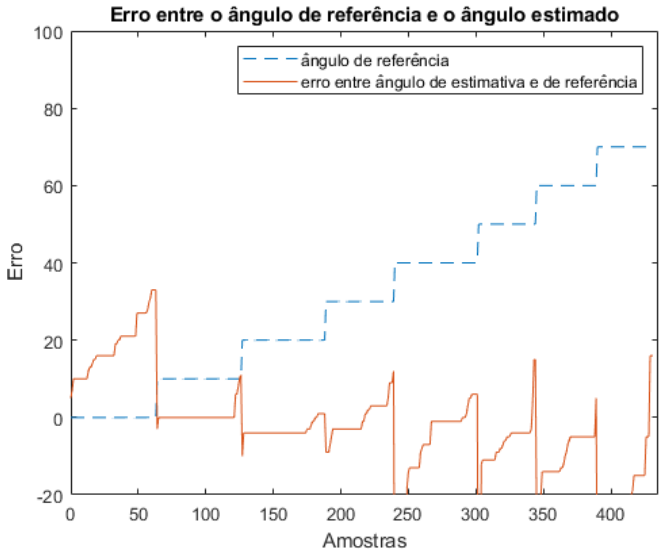


Figura 5.16: Curvas evidenciando os ângulos de referência e os erros de estimação

Capítulo 6

Conclusões e Proposta para Futuras Análises

A fase experimental deste projeto mostrou características importantes previamente discutidas durante o embasamento teórico, tais como as limitações do sistema e a importância de fatores como a disposição dos microfones e o direcionamento da fonte sonora em relação ao sistema de captação. Verificou-se também como a distância pode influenciar sobre a intensidade dos sinais transmitidos pela fonte e captados pelos microfones.

Inicialmente observamos dificuldades no sistema para a estimação do ângulo de 0 grau. O processo de correlação deste ângulo apresentou em vários resultados valores grandes de k. Conforme visto na seção 4.4, k possui uma condição a ser satisfeita para a realização correta do processo de estimativa, a saber k deve ser inferior a 22. Além disso, podemos destacar que as estimavas mais satisfatórias ocorreram para ângulos no intervalo de 10 a 30 graus.

Com uma resolução de 5,2 graus e ângulo máximo para estimação de 85,81 graus, foram obtidos resultados razoáveis para a estimativa da direção de chegada do som. Resultados positivos indicam que a fonte se encontrava à esquerda do sistema de captação e valores negativos, à direita do sistema. Resultados com grandes valores de defasagem ou valores absurdos podem ser caracterizados como erro na estimação, sendo necessário realizar outro processo de aquisição e estimativa para se obter um resultado mais coerente.

Existem fatores que limitam a eficiência deste sistema conforme foi discutido anteriormente. A captação omnidirecional torna o sistema sujeito a ondas oriundas de todas as direções. O sistema, por exemplo, estará sujeito à limitação em determinar se o sinal se origina em uma fonte à frente ou atrás do arranjo de microfones. Dessa forma, torna-se necessária uma atenção para o fenômeno de reflexão em um ambiente fechado, por isso realizamos os ensaios em um ambiente aberto.

Por fim, verificou-se a eficiência do programa implementado. Os principais parâmetros avaliados foram: qualidade das estimativas; velocidade de processamento; e tamanho da memória utilizada pelo microprocessador e pela memória externa. Foram aplicadas várias

alterações sobre o código principal com o objetivo de se alcançar melhores estimativas e parâmetros de processamento.

A realização deste experimento apresenta como perspectivas para trabalhos futuros, adaptações em cada etapa do projeto. Por exemplo, pode-se implementar alterações no sistema de captação tais como o uso de microfones com melhor relação sinal-ruído, novas topologias para o circuito de pré-amplificação. Pode-se, inclusive, utilizar um novo arranjo contendo vários microfones para estimar o problema da capitação omnidirecional dos microfones. Além disso, pode-se pesquisar a implementação de novas técnicas de estimativa DOA.

Referências Bibliográficas

ARAÚJO, A. D. M. (2018). Plataforma móvel guiada pelo som. *Trabalho de Conclusão de Curso*.

Chacon-Rodriguez, A. and Julian, P. (February 2011). Evaluation of gunshot detection algorithms. vol. 58(no. 2).

de Pesquisas Espaciais, C. R. S., . R. M. C. Pesquisa e desenvolvimento de tecnologias eletromecÂnicas de movimentaÇÃo de antenas.

DI CLAUDIO, E. and PARISI, R. (Out. 2001). Waves: Weighted average of signal subspaces for robust wideband direction finding. *IEEE Trans. On Signal processing*, Vol. 49(n. 10):p.2179–2190.

e J. A. Apolinario Jr., I. L. F. (Agosto 2011a). Doa of gunshot signals in a spatial microphone array: performance of the interpolated generalized crosscorrelation method.

e J. A. Apolinario Jr., I. L. F. (Fevereiro 2011b). Gcc-based doa estimation of over-lapping muzzleblast and shockwave components of gunshot signals.

e J. A. Apolinario Jr., I. L. F. (Setembro 2011c). Localização de atirador por arranjo de microfones.

Gontijo, A. T. e Costa, M. V. S. (Janeiro 2007). Desenvolvimento do hardware e estudo dos métodos utilizados na estimação de doa por meio de arranjos de sensores: prótese auditiva inteligente.

Machado, W., J. C. A. R. . L. W. S. (2019). Rede neural artificial aplicada na detecÇÃo de direÇÃo do sinal sonoro.

ONUMA, Y., ICHIGE, K., and ARAI, H. (Nov. 2007). Doa estimation for wideband signals using modified c-sprit method. *TENCON 2007 - IEEE Region 10 Conference*, pages p.1–4.

Prandel, P. C., F. I. L. . A. J. J. A. (2012). Detecção e estimação em tempo real de direção de chegada de sinais impulsivos.

YONN, Y., KAPLAN, L., and MCCLELLAN, J. (Jun 2006). Tops: News doa estimator for wideband signals. *IEEE Transactions on Signal processing*, Vol. 54(n. 6).

ZHOU, L., ZHAO, Y.-j., and CUI, H. (Jun. 2008). High resolution wideband doa estimation based on modified music algorithm. pages p.20–22.

```
1
2  // DOA - main
3  #include <msp430.h>
4  #include "Defines.h"
5  #include "Globais.h"
6  #include "leds_chaves.h"
7  #include "lcd.h"
8  #include "crono.h"
9  #include "sram.h"
10 #include "bt.h"
11 #include "adc.h"
12 #include "filtros.h"
13 #include "resto.h"
14 void energia(void); //Esperar sinal
15 void amostra(void); //Amostrar sinal
16 void onda(void);
17 int main(void)
18 {
19 volatile unsigned long tempo;
20 //long adc_cont;
21 //long adr=0;
22 //volatile int atz, per;
23 int i;
24 int x,y,z,doa;
25 int aux[400];
26 long xx=1234567891L;
27 volatile unsigned int atz12,atz21,per12,per21;
28 unsigned int qdr; //Quadrante 0 (Dir) ou 1 (Esq)
29 WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer
30
31 relogio();
32 chaves_config();
33 leds_config();
34 crono_inic();
35 crono_zera();
36 spi_inic();
37 sram_reg_wr(1,SRAM_MODO_SEQ);
38 i2c_config();
39 pcf_adr=pcf_qual_adr();
40 lcd_inic();
41 lcd_BL();
42 //bt_inic(460800L);
43 //bt_inic(38400L);
44 bt_inic(250000L);
45 __enable_interrupt(); //GIE=1
46 scp1(); //Pino P2.4
47 scp2(); //Pino P2.5
48 //prn = estado das chaves
```

```
49 prn=P2IN&BIT1;
50 prn|=(P1IN&BIT1)>>1;
51 prn^=3;
52 lcd_cursor(0);
53 lcd_str("Modo prn = ");
54 lcd_udec16nz(prn);
55 __delay_cycles(20000000);
56 lcd_cursor(0);
57 lcd_str(" ");
58 /*
59 sram_num_16b (1, 0L, QTD, 0, 1, 32767);
60 separa_canais();
61 bt_str("SRAM 1\n");
62 bt_sram_dump_hex16(1, 0L, QTD/2);
63 bt_str("SRAM 2\n");
64 bt_sram_dump_hex16(2, 0L, QTD/2);
65 while(TRUE);
66 */
67 //rampa();
68 // Iniciar aquisicao
69 ta0p1(FS);
70 while(TRUE){
71 adc_dma_liga();
72 lcd_cursor(0);
73 lcd_str("Sinal ? ");
74 energia(); // Esperar sinal com energia
75 lcd_cursor(0);
76 lcd_str("Processando ... ");
77 lcd_cursor(0x40);
78 lcd_str(" ");
79 if (prn>0){ lcd_cursor(0); lcd_str("Amostrando... "); }
80 amostra();
81
82 adc_dma_desliga();
83 if (prn>0){ lcd_cursor(0); lcd_str("Separando canais"); }
84 separa_canais();
85 if (prn>1){
86 lcd_cursor(0);
87 lcd_str("Tx Canais puros ");
88 bt_sram_dump_udec16(1, 0L, QTD/2);
89 bt_crlf(2);
90 bt_sram_dump_udec16(2, 0L, QTD/2);
91 bt_crlf(8);
92 }
93 if (prn>0){ lcd_cursor(0); lcd_str("Filtro MM 4... "); }
94 mm_16b(1, 0L, QTD/2);
95 mm_16b(2, 0L, QTD/2);
96 if (prn>0){ lcd_cursor(0); lcd_str("Filtro IIR... "); }
97 //iir_16b(1, 0L, QTD/2);
```

```
98  //iir_16b(2, 0L, QTD/2);
99  if (prn>1){
100 lcd_cursor(0);
101 lcd_str("Tx Canais Proc ");
102 bt_sram_dump_udec16(1, 0L, QTD/2);
103 bt_crlf(2);
104 bt_sram_dump_udec16(2, 0L, QTD/2);
105 bt_crlf(2);
106 }
107 doa=doa_est((long)COR_INI);
108 bt_dec16nz(doa);
109 bt_crlf(1);
110 if (prn>1){
111 bt_dec32(FS); bt_crlf(1);
112 bt_dec32(QTD); bt_crlf(1);
113 bt_dec16(COR_TAM); bt_crlf(1);
114 bt_dec16(COR_INI); bt_crlf(1);
115 bt_dec16(COR_PASSO); bt_crlf(1);
116 bt_dec16(COR_PTS); bt_crlf(1);
117 bt_dec16(DOA_QTD); bt_crlf(1);
118 bt_dec16(JZ_MIN_MED); bt_crlf(1);
119 bt_dec16(JZ_FAIXA); bt_crlf(1);
120 }
121 lcd_cursor(0x40);
122 lcd_str(" DOA = ");
123 lcd_dec16nz(doa);
124 }
125 while(TRUE);
126 return 0;
127 }
128 // Esperar sinal
129 void energia(void){
130 volatile unsigned int energia;
131 led_VD();
132 led_VM();
133 energia=0;
134 while(energia < LIMIAR_ENERGIA){
135 while(adc_fim == FALSE);
136 adc_fim=FALSE;
137 if (adc_fase==0)
138 energia= adc_energ(vy);
139 else
140 energia= adc_energ(vx);
141 }
142 led_vm();
143 }
144 // Amostrar o sinal
145 void amostra(void){
146 long adc_cont;
```

```
147 long adr;
148 adr=0;
149 adc_cont=0;
150 for (adc_cont=0; adc_cont<QTD; adc_cont+=ADC_TAM){
151 while(adc_fim == FALSE);
152 adc_fim=FALSE;
153 Scp1();
154 SCP2();
155 if (adc_fase==0)
156 sram_wr_seq_16b(1, adr, vy, (long) ADC_TAM);
157 else
158 sram_wr_seq_16b(1, adr, vx, (long) ADC_TAM);
159 scp2();
160 adr+=2*ADC_TAM; //Contar passos de 16 bits
161 }
162 led_vd();
163 }
164 // TA1.1 (P2.0) Gerar Onda quadrada
165 void onda(void){
166 P2DIR |= BIT0;
167 P2SEL |= BIT0;
168 TA1CTL = TASSEL_2 | MC_1;
169 TA1CCR0 = 20000; //10kHz com SMCLK=20MHz
170 TA1CCR1 = TA1CCR0/2;
171 TA1CCTL1 = OUTMOD_6;
172 }
```

```
1
2  // Defines
3  #ifndef DEFINES_H_
4  #define DEFINES_H_
5  #define TRUE 1
6  #define FALSE 0
7
8  #define DOA_QTD 15
9  #define COR_TAM 200
10 #define COR_INI 4*COR_TAM
11 #define COR_PASSO 300
12 #define COR_PTS 100
13 #define ATZ_MAX_50K 11
14
15 #define JZ_MIN_MED 5
16 #define JZ_FAIXA 10
17
18 # define FS 50000L
19 //# define FS 10000L
20 // Limiar de energia para caracterizar sinal presente
21 // Usado por adc_espera()
```

```
22 # define LIMIAR_ENERGIA 200
23 // Total de amostras de 16 bits - Janela de tempo
24 #define QTD (32*1024L) //Limite = 65.536 int (16 bits)
25 // Definir tamanho dos vetores usados no ADC
26 #define ADC_TAM 128L //? Qtd de bytes entregues pelo ADC ? Confirmar!
27
28 #define MM_TAM 4
29 #define IIR_ORD 4 //ordem do filtro
30 //Definir estado das chaves
31 #define ABERTA 1
32 #define FECHADA 0
33 #define DBC 50
34 #define SMCLK 20000000L
35 #define ACLK 32768
36 #define BT_FILA_TAM 100 // tamanho do buffer do Bluetooth
37 #endif /* DEFINES_H_ */
```

```
1
2 #include "Defines.h"
3 #ifndef GLOBAIS_H_
4 #define GLOBAIS_H_
5
6 volatile unsigned int prn;
7
8 volatile unsigned int bac1=0,bac2=0;
9 // Tabelas de DOA
10 // 0 1 2 3 4 5 6 7 8 9 10 11 12
11 char DOA_50K[]={0, 5, 10, 16, 21, 27, 33, 39, 46, 55, 65, 86, 99}; //12
       valores mais o 99 para o erro
12 // 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
13 char DOA_100K[]={0, 3, 5, 8, 10, 13, 16, 19, 21, 24, 27, 30, 33, 36, 39,
       43, 46, 50, 55, 59, 65, 72, 86, 99}; //23 valores mais o 99 para o
       erro
14
15 // b=[b0 b1 b2 b3 ...]
16 // a=[1 a1 a2 a3 ...]
17 // 1) IIR - Butterwoth ORD=4, FS=50K ==> FS=25K por canal
18 // b = {0.0279 0, -0.0557, 0, 0.0279 }; //b Original
19 int b[IIR_ORD+1] = { 0x0007, 0x0000, 0xFFF2, 0x0000, 0x0007 };
20 // a = { 1.0000 -3.3650, 4.3462, -2.5651, 0.5869 }; //a Original
21 int a[IIR_ORD+1] = { 0x0100, 0xFCA3, 0x0458, 0xFD70, 0x0096 };
22 // am = {-1.0000, 3.3650, -4.3462, 2.5651, -0.5869}; //a negado Original
23 int am[IIR_ORD+1] = { 0xFF00, 0x035D, 0xFBA8, 0x0290, 0xFF6A };
24 /*
25 // 2) IIR - Butterwoth ORD=4, FS=100K ==> FS=50K por canal
26 // b = {0.0078, 0, -0.0156, 0, 0.0078 }; //b Original
27 int b[IIR_ORD+1] = { 0x0002, 0x0000, 0xFFFC, 0x0000, 0x0002 };
28 // a = { 1.0000, -3.7052, 5.1803, -3.2409, 0.7660 }; //a Original
```

```
29 int a[IIR_ORD+1] = { 0x0100, 0xFC4C, 0x052E, 0xFCC3, 0x00C4 };
30 // am = {-1.0000, 3.7031, -5.1797, 3.2383, -0.7656 }; //a negado Original
31 int am[IIR_ORD+1] = { 0xFF00, 0x03B4, 0xFAD2, 0x033D, 0xFF3C };
32 */
33 /*
34 // IIR - TESTE
35 // b=[0.08984375, 0.00000000, -0.17968750, 0.00000000, 0.08984375]
36 int b[IIR_ORD+1]={0x0017, 0x0000, 0xFFD2, 0x0000, 0x0017}; //Numerador
37 long b32[IIR_ORD+1]={ 0x00001760, 0x00000000, 0xFFFFD140, 0x00000000, 0
       x00001760}; //Numerador em 32 bits
38 //int b[IIR_ORD+1]={0x0020, 0x0000, 0xFFC0, 0x0000, 0x0020}; //Numerador
39 // a=[1.00000000, -2.04296875, 2.07421875, -1.14843750, 0.34765625]
40 int a[IIR_ORD+1]={0x0100, 0xFDF5, 0x0213, 0xFEDA, 0x0059}; //Denominador
41 // vetor a negado
42 int am[IIR_ORD+1] ={0xFF00, 0x020B, 0xFDED, 0x0126, 0xFFA7}; //
       Denominador
43 long am32[IIR_ORD+1]={0xFFFF0000, 0x00020B6D, 0xFFFDEC27, 0x00012673, 0
       xFFFFA700}; //Denominador em 32 bits
44 //int a[IIR_ORD+1]={0x0100, 0xFE00, 0x0200, 0x180, 0x0040}; //Denominador
45 */
46
47 int passo=10;
48
49 int vx[ADC_TAM+MM_TAM+IIR_ORD+1],vy[ADC_TAM+MM_TAM+IIR_ORD+1]; //vetores
       para os dados convertidos
50 int adc_fase; //Indica qual buffer usar
51 int adc_cont; //Contador dentro de cada fase
52 int adc_fim;
53 int dma_cont;
54 int ps1=ABERTA;
55 int ps2=ABERTA;
56 int pcf_adr;
57
58 char lcdi_vet[40]; //Vetor com string a imprimir
59 volatile int lcdi_ix;
60 volatile int lcdi_cont;
61 volatile int lcdi_ocupado=FALSE;
62
63 volatile unsigned char crono_cont; //Contador auxiliar
64 volatile unsigned char crono_erro; //Erro entre Start/Stop
65
66 volatile unsigned char bt_fila[BT_FILA_TAM]; // porta serial/bluetooth
67 volatile unsigned char bt_pin, bt_pout; // ponteiros (indexadores para
       fila)
68 volatile int bti_ocupado=FALSE; //Interrup ainda ocorrendo
69 #endif /* GLOBAIS_H_ */
```

```
1
```

```
2  #include <msp430.h>
3  #include "Defines.h"
4  #include "leds_chaves.h"
5  // Estado chave S1
6  // TRUE se foi acionada
7  int chave_s1(void){
8  if ((P2IN&BIT1) == FECHADA){
9  if (ps1 == ABERTA){
10 ps1=FECHADA;
11 debounce(DBC);
12 return TRUE;
13 }
14 }
15 else{
16 if (ps1 == FECHADA){
17 ps1=ABERTA;
18 debounce(DBC);
19 }
20 }
21 return FALSE;
22 }
23 // Estado chave S2
24 // TRUE se foi acionada
25 int chave_s2(void){
26 if ((P1IN&BIT1) == FECHADA){
27 if (ps2 == ABERTA){
28 ps2=FECHADA;
29 debounce(DBC);
30 return TRUE;
31 }
32 }
33 else{
34 if (ps2 == FECHADA){
35 ps2=ABERTA;
36 debounce(DBC);
37 }
38 }
39 return FALSE;
40 }
41 // Configurar Leds
42 void leds_config(void){
43 P1DIR |= BIT0; //Led VM
44 P1OUT &= ~BIT0; //VM apagado
45 P4DIR |= BIT7; //Led VD
46 P4OUT &= ~BIT7; //VD apagado
47 P2OUT &= ~(BIT4 | BIT5); //Bits Scope P2.4 e P2.5
48 P2DIR |= BIT4 | BIT5; //Bits Scope P2.4 e P2.5
49 }
50 // Configurar Chaves
```

```
51  void chaves_config(void){
52  P2DIR |= ~BIT1; //Chave S1
53  P2REN |= BIT1; //Pullup
54  P2OUT |= BIT1; //Ligado
55  P1DIR |= ~BIT1; //Chave S2
56  P1REN |= BIT1; //Pullup
57  P1OUT |= BIT1; //Ligado
58  }
59  // VM Aceso
60  void led_VM(void){
61  P1OUT |= BIT0;
62  }
63  // VM Apagado
64  void led_vm(void){
65  P1OUT &= ~BIT0;
66  }
67  // VM Invertidoo
68  void led_Vm(void){
69  P1OUT ^= BIT0;
70  }
71  // VD Aceso
72  void led_VD(void){
73  P4OUT |= BIT7;
74  }
75  // VD Apagado
76  void led_vd(void){
77  P4OUT &= ~BIT7;
78  }
79  // VD Invertidoo
80  void led_Vd(void){
81  P4OUT ^= BIT7;
82  }
83  // Debounce das chaves
84  void debounce(unsigned int x){
85  volatile unsigned int z;
86  for (z=0; z<x; z++);
87  }
88  // scp1=1
89  void SCP1(void){
90  P2OUT |= BIT4;
91  }
92  // scp1=0
93  void scp1(void){
94  P2OUT &= ~BIT4;
95  }
96  // Inverter scp1
97  void Scp1(void){
98  P2OUT ^= BIT4;
99  }
```

```
100 // scp2=1
101 void SCP2(void){
102 P2OUT |= BIT5;
103 }
104 // scp2=0
105 void scp2(void){
106 P2OUT &= ~BIT5;
107 }
108 // Inverter scp2
109 void Scp2(void){
110 P2OUT ^= BIT5;
111 }
```

```
1
2  #ifndef LCD_H_
3  #define LCD_H_
4  // Bits para controle do LCD
5  #define BIT_RS BIT0
6  #define BIT_RW BIT1
7  #define BIT_E BIT2
8  #define BIT_BL BIT3
9  #define BIT_D4 BIT4
10 #define BIT_D5 BIT5
11 #define BIT_D6 BIT6
12 #define BIT_D7 BIT7
13 // Baud Rate para I2C com SMCLK = 20 MHz
14 #define BR500K 40 //500 kbps
15 #define BR400K 50 //400 kbps
16 #define BR100K 200 //100 kbps
17 #define BR50K 400 // 50 kbps
18 #define BR10K 2000 // 10 kbps
19 // Delay para depois do STOP
20 #define ATZ_STOP 1000
21 void lcd_crono(unsigned long c);
22 void lcdi_crono(unsigned long c);
23 void lcd_crono_prep(unsigned long c, char *v);
24 void lcd_dec16(int c);
25 void lcd_dec16nz(int c);
26 void lcd_udec16nz(unsigned int c);
27 void lcdi_udec16nz(unsigned int c);
28 void lcd_udec16(unsigned int c);
29 void lcdi_udec16(unsigned int c);
30 void lcd_dec8(char c);
31 void lcd_udec8(char c);
32 void lcd_hex16(int c);
33 void lcd_hex8(char c);
34 void lcdi_spc(char c);
35 void lcd_spc(char c);
```

```
36 char lcd_ocupado(void);
37 char lcd_status(void);
38 void lcd_status_prep(char c1, char c2);
39 char lcd_status_ler(void);
40 void lcd_cursor(char x);
41 void lcd_cmdo(char x);
42 void lcd_cmdo_aux(char x);
43 void lcdi_str(char *pt);
44 void lcd_str(char *pt);
45 void lcd_char(char c);
46 void lcd_char_aux(char c);
47 void lcd_BL(void);
48 void lcd_bl(void);
49 void lcd_Bl(void);
50 void lcd_inic(void);
51 void lcd_aux(char dado);
52 void pcf_STT_STP(void);
53 void pcf_write(char dado);
54 int pcf_qual_adr(void);
55 int pcf_ack(int adr);
56 void delay(long limite);
57 void i2c_config(void);
58 int lcd_lgt; //Back light BIT_BL=ligado 0=apagado
59 extern int pcf_adr;
60 extern char lcdi_vet[40]; //Vetor com string a imprimir
61 extern volatile int lcdi_ix;
62 extern volatile int lcdi_cont;
63 extern volatile int lcdi_ocupado;
64 #endif /* LCD_H_ */
```

```
1
2 #ifndef CRONO_H_
3 #define CRONO_H_
4 unsigned long crono_ler(void);
5 void crono_inic(void);
6 void crono_start(void);
7 void crono_stop(void);
8 void crono_zera(void);
9 unsigned char crono_calibra(void);
10 extern volatile unsigned char crono_cont; //Contador auxiliar
11 extern volatile unsigned char crono_erro; //Erro entre Start/Stop
12 #endif /* CRONO_H_ */
```

```
1
2 #ifndef SRAM_H_
3 #define SRAM_H_
4
```

```
5  #define SRAM_INSTR_RD 0x03
6  #define SRAM_INSTR_WR 0x02
7  #define SRAM_MR_RD 0x05 //Instr para ler Registrador de Modo
8  #define SRAM_MR_WR 0x01 //Instr para escrever no Registrador de Modo
9
10 #define SRAM_MODO_BYTE 0x00 //Modo Byte
11 #define SRAM_MODO_PAG 0x80
12 #define SRAM_MODO_SEQ 0x40 //Modo Sequencial
13 void separa_canais(void);
14 void sram_num_16b(char qual, long adr, long qtd, int ini, int passo, int
       fim);
15 void sram_wr_seq_16b(char qual, long adr, int *vet, long qtd);
16 void sram_rd_seq_16b(char qual, long adr, int *vet, long qtd);
17 void sram_wr_seq(char qual, long adr, char *vet, long qtd);
18 void sram_rd_seq(char qual, long adr, char *vet, long qtd);
19 void sram_wr(char qual, long adr, char dt);
20 char sram_rd(char qual, long adr);
21 void sram_reg_wr(char qual, char dt);
22 char sram_reg_rd(char qual);
23 void spi_inic(void);
24 char spi_transf(char x);
25 void sram_cs1(void);
26 void sram_CS1(void);
27 void sram_cs2(void);
28 void sram_CS2(void);
29 void sram_CS12(void);
30 extern int vx[],vy[]; //vetores para os dados convertidos
31 #endif /* SRAM_H_ */
```

```
1
2  #ifndef BT_H_
3  #define BT_H_
4  void bt_sram_dump_hex16(char qual, long adr, long qtd);
5  void bt_sram_dump_hex8(char qual, long adr, long qtd);
6  void bt_sram_dump_dec16(char qual, long adr, long qtd);
7  void bt_sram_dump_udec16(char qual, long adr, long qtd);
8  void bt_sram_16b(char qual, long adr, long qtd);
9  void bt_cmdo_at(void);
10 void bt_at_tx_rx(char *vai);
11 void bti_crono(unsigned long c);
12 void bt_dec32(long c);
13 void bti_udec16(unsigned int c);
14 void bt_udec16(unsigned int c);
15 void bt_dec16nz(int c);
16 void bti_dec16(int c);
17 void bt_dec16(int c);
18 void bti_hex32(unsigned long c);
19 void bt_hex32(unsigned long c);
```

```
20 void bti_hex16(unsigned int c);
21 void bt_hex16(unsigned int c);
22 void bti_hex8(char c);
23 void bt_hex8(char c);
24 void bti_spc(char qtd);
25 void bt_spc(char qtd);
26 void bti_crlf(char qtd);
27 void bt_crlf(char qtd);
28 void bti_str(char *pt);
29 void bt_str(char *pt);
30 void bt_char(char x);
31 void bt_inic(long brate);
32 void bt_fila_inic(void);
33 char bt_poe(char cha);
34 char bt_tira(char *cha);
35 extern volatile unsigned char bt_fila[]; // fila para o serial -
      bluetooth
36 extern volatile unsigned char bt_pin, bt_pout; // ponteiros (indexadores
       para fila)
37 extern volatile int bti_ocupado; //Interrup ainda ocorrendo
38 #endif /* BT_H_ */
```

```
1
2 #ifndef ADC_H_
3 #define ADC_H_
4 unsigned int adc_energ(int *vet);
5 void adc_dma_liga(void);
6 void adc_dma_desliga(void);
7 void adc_conv_dma(int qtd, int *vet);
8 void adc_conv(int qtd, int *vet);
9 void dma0_adc_config(int *dst);
10 void adc_config(void);
11 void ta0p1(long freq);
12 void rampa(void);
13 //Bacalhau para debug
14 extern volatile unsigned int bac1,bac2;
15 extern int passo; //Passo para rampa - testar ADC
16 extern int vx[],vy[]; //vetores para os dados convertidos
17 extern int adc_fase; //Indica qual buffer usar
18 extern int adc_cont; //Contador dentro de cada fase
19 extern int adc_fim;
20 extern int dma_cont;
21 extern volatile unsigned int bac;
22 #endif /* ADC_H_ */
```

```
1
2 #ifndef FILTROS_H_
```

```
3  #define FILTROS_H_
4  int doa_est(long adr);
5  int juiz(int *vet);
6  void bolha(int *vv);
7  int doa_um(long adr
8  );
9  unsigned int cor1(long ad, unsigned int *atz12, unsigned int *atz21,
       unsigned int *per12, unsigned int *per21);
10 unsigned int cor_atz_per(int *vet, unsigned int *per);
11 void mm_16b(int qual, long adr, long qtd);
12 void iir_hw_32b(int qual, long adr, long qtd);
13 void iir_hw_16b(int qual, long adr, long qtd);
14 void iir_16b(int qual, long adr, long qtd);
15 extern volatile unsigned int prn;
16 extern char DOA_50K[];
17 extern int vx[],vy[]; //vetores para os dados convertidos
18 extern int b[],a[],am[]; //Coeficientes do filtro IIR
19 extern long b32[],am32[]; //Coeficientes do filtro IIR 32 bits
20 #endif /* FILTROS_H_ */
```

```
1
2  #ifndef RESTO_H_
3  #define RESTO_H_
4
5  #define FLLN(x) ((x)-1)
6  void separa_canais(void);
7  void str_dec32(long c, char *msg);
8  void str_dec16(unsigned int c, char *msg);
9  void str_dec16nz(int c, char *msg);
10 void str_udec16nz(unsigned int c, char *msg);
11 void str_udec16(unsigned int c, char *msg);
12 void str_hex32(unsigned long c, char *msg);
13 void str_hex16(unsigned int c, char *msg);
14 void str_hex8(unsigned char c, char *msg);
15 void str_crlf(char qtd, char *msg);
16 void str_spc(char qtd, char *msg);
17 void relogio(void);
18 #endif /* RESTO_H_ */
```