

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE
SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE ENGENHARIA ELETRÔNICA**

DIOGO TAVARES

***CROSSOVER* DIGITAL COM EQUALIZAÇÃO
PARAMÉTRICA AJUSTÁVEL**

FLORIANÓPOLIS, 2020

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE
SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE ENGENHARIA ELETRÔNICA**

DIOGO TAVARES

***CROSSOVER* DIGITAL COM EQUALIZAÇÃO PARAMÉTRICA AJUSTÁVEL**

Trabalho de conclusão de curso submetido
ao Instituto Federal de Educação, Ciência
e Tecnologia de Santa Catarina como parte
dos requisitos para obtenção do título de
engenheiro eletrônico

Orientador:
Prof. Dr. Fernando Santana Pacheco
Coorientador:
Dr. Rosalfonso Bortoni

FLORIANÓPOLIS, 2020

Tavares, Diogo

Crossover digital com equalização paramétrica ajustável. / Diogo Tavares; orientador, Prof. Dr. Fernando Santana Pacheco; coorientador, Dr. Rosalfonso Bortoni - Florianópolis, SC, 2020.

81 p. : il. color.

Trabalho de Conclusão de Curso (Eng. Eletrônica) – Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina. Inclui referências.

1. DSP. 2. Áudio. 3. Crossover. I. Pacheco, Fernando S. II. Bortoni, Rosalfonso. III. Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina. IV. Crossover digital com equalização paramétrica ajustável.

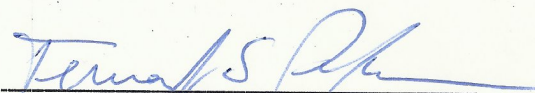
CROSSOVER DIGITAL COM EQUALIZAÇÃO PARAMÉTRICA AJUSTÁVEL

DIOGO TAVARES

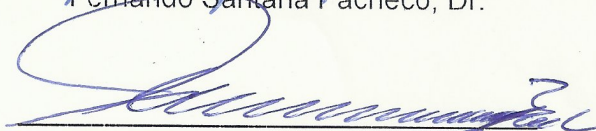
Este trabalho foi julgado adequado para obtenção do Título de Engenheiro Eletrônico e aprovado na sua forma final pela banca examinadora do Curso de Engenharia Eletrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 7 de outubro de 2020.

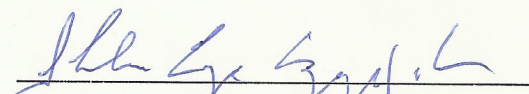
Banca examinadora:



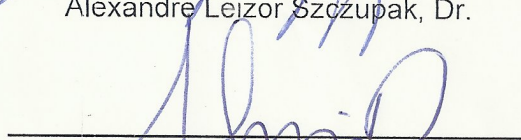
Fernando Santana Pacheco, Dr.



Rosalfonso Bortoni, Dr.



Alexandre Leizor Szczupak, Dr.



André Luís Dalcastagné, Dr.

AGRADECIMENTOS

- Agradeço aos meus pais Tito Tavares e Josanne S Pinheiro, por sempre priorizarem meus estudos.
- Agradeço ao meu orientador, Fernando Santana Pacheco, e coorientador, Rosalfonso Bortoni, pelas orientações ao longo desse trabalho.
- Agradeço ao IFSC e ao DAELN, por proporcionar o ambiente no qual obtive tantos conhecimentos durante esta graduação.

RESUMO

Este trabalho tem como objetivo a implementação de um sistema digital de áudio, utilizando DSP (*Digital Signal Processing*), o qual consiste em um *crossover*, com um canal de entrada mono e dois canais de saída, e um equalizador paramétrico de uma banda. Foi feita a implementação em *hardware*, que consiste em uma plataforma com uma placa STM32F4 *Discovery* integrada com uma placa *WolfsonPi*, e também uma interface com *display* e botões para que o usuário possa ajustar os seguintes parâmetros em tempo real: volume de saída, frequência de corte e ordem do *crossover*, frequência central, ganho e largura de banda do equalizador paramétrico e também a opção de ligar e desligar cada filtro. O algoritmo, o qual foi testado primeiramente em MATLAB, consiste de filtros digitais tipo IIR e o cálculo dos coeficientes são feitos sempre que há variação de algum parâmetro, ou seja, são calculados em tempo real e não previamente tabelados. Primeiramente é feito o processamento do equalizador, a partir do canal de entrada, e em seguida separados em dois canais, onde um realiza um filtro passa-baixas e outro um passa-altas, ambos com a mesma frequência de corte. No processamento de todos os filtros foram utilizados filtros de segunda ordem (ou biquadráticos). Para o *crossover*, devido ao fato de haver defasagem de 180° na frequência de cruzamento entre os filtros de segunda ordem (tipo *Butterworth*), houve cancelamento do sinal na região de sobreposição dos canais, conhecido como cancelamento por fase. Isto foi solucionado com a inversão da polaridade do sinal do filtro passa-altas. Foi implementada também a opção para filtros de quarta ordem (tipo *Linkwitz-Riley*), cujo defasamento entre os canais é de 360° na frequência de cruzamento (*crossover*), não havendo problemas de cancelamento. O sistema foi avaliado com a plataforma APx da Audio Precision. Como resultado final, é possível, por meio da navegação do *display* com os botões, alterar todos os parâmetros estipulados, sem ruídos audíveis.

Palavras-chaves: DSP, áudio, *crossover*, equalizador.

ABSTRACT

This work aims to implement a digital audio system, using DSP (Digital Signal Processing), which consists of a crossover, with a mono input channel, two output channels, and one band parametric equalizer. The hardware platform, consists of a STM32F4 Discovery board integrated with a WolfsonPi board, as well as an interface with display and buttons for controls. The user can adjust the following parameters in real time: output volume, cutoff frequency and crossover order, central frequency, gain and bandwidth of the parametric equalizer, and also the option of turn each filter on or off. The algorithm, which was tested in MATLAB at first, consists of IIR digital filters, and the calculation of its coefficients are done whenever there is a variation in any parameter, that is, they are calculated in real time and not previously tabulated. First, the equalizer is processed from the input channel, and then separated into two channels, where one performs a low-pass filter and the other a high-pass filter, both with the same cutoff frequency. In order to process all the filters, second-order filters, or biquads, were used. For the crossover, due to the fact that there is a 180° lag in the crossover frequency between the second order filters (Butterworth type), the signal was canceled in the channels overlapping region, what is known as phase cancellation. This was solved by reversing the polarity of the high-pass filter signal. The option for fourth-order filters (Linkwitz-Riley type) was also implemented, whose lag between channels is 360° in the crossover frequency and does not generate cancellation problems. The system was tested with the Audio Precision APx platform. As a final result, it is possible, navigating through the display with the buttons, to change all the stipulated parameters, without any audible noise.

Keywords: DSP, audio, filters, crossover, equalizer.

LISTA DE ILUSTRAÇÕES

Figura 1 – Dispositivo <i>Crossover</i> com entrada de dois canais, 8 canais de saída e 4 bandas.	14
Figura 2 – Exemplo de sistema básico de PA (<i>Public Address</i>) para sonorização de eventos, com processamento, mostrando seus diversos componentes.	19
Figura 3 – Magnitude da resposta em frequência de um alto-falante modelo WPU1505, medida no seu eixo central (linha contínua), e a 45° (linha tracejada).	20
Figura 4 – Sistema de áudio digital, com conversores de entrada e saída e o processamento digital utilizando os valores binários amostrados. . .	22
Figura 5 – Amostragem de um sinal. Em A, o sinal analógico, mostrando os instantes que serão amostrados. Em B, as amostras resultantes que serão armazenadas.	23
Figura 6 – Magnitudes das respostas em frequência dos tipos mais utilizados de filtros seletores de sinais: (a) passa-baixas; (b) passa-altas; (c) passa-faixa; (d) rejeita-faixa	25
Figura 7 – <i>Crossover Butterworth</i> com ganho de -3dB e $Q = 0,707$. As curvas apresentadas são: o filtro passa-baixas (azul), o passa-altas (vermelho), a soma das duas (roxo), resultando em uma saliência de até 3 dB na região de corte.	31
Figura 8 – <i>Crossover Butterworth</i> com corte em -6 dB e $Q = 0,5$. As curvas apresentadas são: o filtro passa-baixas (azul), o passa-altas (vermelho), e a soma das duas (roxo), resultando em uma resposta plana . . .	31
Figura 9 – <i>Crossover Linkwitz-Riley</i> . As curvas apresentadas são: o filtro passa-baixas (magenta), o passa-altas (verde), e a soma das duas (azul), resultando em uma resposta plana.	32
Figura 10 – Equalizador paramétrico com $f_c = 10$ kHz, $Q = 3$, e ganho/corte de 9 dB (respectivamente, em vermelho e azul).	33
Figura 11 – Diagrama de blocos do sistema	36
Figura 12 – Diagrama da metodologia de projeto utilizada.	38
Figura 13 – Fluxograma do algoritmo de processamento.	39
Figura 14 – Resposta em frequência com magnitude e fase. Nota-se o ganho de 9 dB e a fase passando por zero em 500 Hz. $Q = 3$	41
Figura 15 – FFT dos sinais de entrada (vermelho) e de saída (azul) do sinal <i>sweep</i> de 20 Hz a 20 kHz. Apresenta ganho de 9 dB em 500 Hz. Fator $Q = 3$	42

Figura 16 – Variação do ganho para $f_c = 500$ Hz e $Q = 3$, sendo $G = 12, -12, 9, -9, 6, -6, 3$ e -3 dB.	43
Figura 17 – FFT do sinal de saída do sistema para variações do ganho entre -12 e $+12$ dB, com f_c igual a 500 Hz.	44
Figura 18 – Variação da largura de banda para $f_c = 500$ Hz e $G = 9$ e -9 dB, com $Q = 3, 6$ e 9	45
Figura 19 – Resposta da saída do sistema para $Q = 3, 6$ e 9 , com $G = 9$ e -9 dB e $f_c = 500$ Hz.	46
Figura 20 – Variação da frequência para $Q = 3$ e $G = 9$ dB, com $f_c = 100, 1000$ e 10000 Hz.	47
Figura 21 – Saída do sistema para $f_c = 100, 1000$ e 10000 Hz, $Q = 3$ e $G = 9$ dB.	48
Figura 22 – Magnitude e fase das respostas em frequência para <i>crossovers</i> de segunda ordem. A Figura mostra o passa-baixas (em azul), o passa-altas (em vermelho), suas respectivas fases, e a soma das duas (em roxo) que representa a saída total do <i>crossover</i>	50
Figura 23 – Magnitude das FFTs dos sinais: de entrada (em laranja), saída (em roxo), passa-baixas (em azul) e passa-altas (em vermelho).	51
Figura 24 – Magnitude e fase da resposta em frequência do <i>crossover</i> de segunda ordem com a inversão da polaridade. A Figura mostra o passa-baixas (em azul), o passa-altas (em vermelho), suas respectivas fases, e a soma das duas (em roxo) que representa a saída total do <i>crossover</i> , com sua fase sobreposta ao passa-baixas.	52
Figura 25 – Magnitudes das FFTs de: sinal de entrada sobreposto ao sinal de saída (roxo), passa-baixas (em azul) e passa-altas (em vermelho).	53
Figura 26 – Magnitude e fase da resposta em frequência para o <i>Linkwitz-Riley</i> . A Figura mostra o passa-baixas (em azul), o passa-altas (em vermelho), suas respectivas fases, e a soma das duas (em roxo) que representa a saída total do <i>crossover</i>	54
Figura 27 – Magnitudes das FFTs de: sinal de entrada sobreposto ao sinal de saída (em roxo), passa-baixas (em azul) e passa-altas (em vermelho).	55
Figura 28 – Comparação entre as respostas em frequência de segunda ordem sem inversão de polaridade (azul e vermelho) e quarta ordem (roxo e verde), em azul claro as saídas sobrepostas.	56
Figura 29 – Saída do sistema com equalizador e <i>crossover</i>	57
Figura 30 – Exemplo de Buffer ping-pong com acesso à memória por DMA.	60
Figura 31 – Foto da vista superior do kit de desenvolvimento STM32F4 <i>Discovery</i>	60
Figura 32 – Foto da vista superior da placa <i>Wolfson Pi</i>	61

Figura 33 – Kit de desenvolvimento STM32F4 <i>Discovery Kit</i> integrado com <i>Wolfson Pi</i>	61
Figura 34 – Interface de áudio <i>Fast Track pro</i>	62
Figura 35 – <i>Software OcenAudio</i>	63
Figura 36 – <i>Software Ableton Live</i>	64
Figura 37 – Blocos da arquitetura de <i>software</i>	65
Figura 38 – Placa para interface com <i>display oled</i> SSD1306 128 x 64 pixels e botões <i>push-button</i>	67
Figura 39 – Visão geral do <i>software</i> APx 500 <i>Audio Precision</i>	70
Figura 40 – Medições em bancada. Sistema desenvolvido conectado ao equipamento APx 525 <i>Audio Analyzer</i> da <i>Audio Precision</i>	71
Figura 41 – Captura de tela do <i>software</i> APx 500, mostrando os valores mínimos de amplitude para frequência de 100 Hz e THD menor que 0,5%.	72
Figura 42 – Captura de tela do <i>software</i> APx 500, mostrando os valores máximos de amplitude para frequência de 100 Hz e THD menor que 0,5%.	72
Figura 43 – Resposta da variação do ganho do equalizador em 1 kHz, Q = 6 e ganho de +/-3 dB a +/-18 dB como passos de 3 dB. <i>Crossover</i> desligado.	73
Figura 44 – Resposta do equalizador com variação do Q em 140 Hz, G = +/-9 dB e Q variando de 3 a 9 com passo de 3. <i>Crossover</i> ligado com corte em 900 Hz.	74
Figura 45 – Resposta do equalizador com variação da frequência em 100 Hz, 500 Hz, 2 kHz e 5 kHz, com G = +/-9 dB e diferentes valores de Q. <i>Crossover</i> desligado.	74
Figura 46 – Saídas do <i>crossover</i> em 500 Hz para segunda e quarta ordem.	75
Figura 47 – Saídas do <i>crossover</i> em 900 Hz para segunda e quarta ordem.	75
Figura 48 – Resposta da soma dos canais do <i>crossover</i> de 500 Hz de segunda ordem, com e sem inversão de fase.	76

LISTA DE TABELAS

Tabela 1 – Coeficientes para o equalizador com $f_c = 500$ Hz, $G = 9$ dB e $Q = 3$.	40
Tabela 2 – Coeficientes para o <i>crossover</i> (de segunda ordem) sem inversão de polaridade com $f_0 = 500$ Hz e $Q = 0,5$, onde PB é o passa-baixas e PA, o passa-altas.	49
Tabela 3 – Coeficientes para o <i>crossover</i> com inversão de polaridade, com $f_0 = 500$ Hz e $Q = 0,5$, onde PB é o passa-baixas e PA, o passa-altas.	52
Tabela 4 – Valores médios de ciclos na execução de algumas seções do processamento.	69

LISTA DE ABREVIATURAS E SIGLAS

dB	<i>Decibel</i>
Hz	<i>Hertz</i>
DSP	<i>Digital Signal Processing</i>
IIR	<i>Infinite Impulse Response</i>
FIR	<i>Finite Impulse Response</i>
PA	<i>Public Address</i>
ADC	<i>Analog Digital Converter</i>
DAC	<i>Digital Analog Converter</i>
CODEC	<i>Coder Decoder</i>
I2S	<i>Inter IC Sound</i>
BLT	<i>Bilinear Transform</i>
MAC	Multiplicação e Acumulação
DFT	<i>Discrete Fourier Transform</i>
FFT	<i>Fast Fourier Transform</i>
ARM	<i>Advanced RISC Machine</i>
DAELN	Departamento Acadêmico de Eletrônica
SIMD	<i>Single Instruction Multiple Data</i>
DMA	<i>Direct Memory Access</i>
IDE	<i>Integrated Development Environment</i>
CMSIS	<i>Cortex Microcontroller software Interface Standard</i>
LINSE	Laboratório de Circuitos e Processamento de Sinais
THD	<i>Total Harmonic Distortion</i>
AP	<i>Audio Precision</i>

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Justificativa	15
1.2	Definição do Problema	16
1.3	Objetivo Geral	16
1.4	Objetivos Específicos	17
2	FUNDAMENTAÇÃO TEÓRICA	18
2.1	Sistemas de áudio	18
2.1.1	Processadores de áudio	19
2.1.2	Alto-falantes	19
2.2	Processamento Digital de Sinais	21
2.2.1	Áudio digital	22
2.2.2	Filtros seletores de sinais	24
2.2.3	Filtros digitais	25
2.2.4	<i>Crossover</i>	27
2.2.4.1	<i>Butterworth</i> de segunda ordem	30
2.2.4.2	Linkwitz-Riley	32
2.2.5	Equalizador paramétrico	32
2.2.6	Processamento de áudio e complexidade computacional	34
2.2.7	Algoritmos	35
3	DESENVOLVIMENTO	36
3.1	Concepção	36
3.1.1	Requisitos do sistema	37
3.2	Projeto	37
3.2.1	Definição do algoritmo	38
3.2.2	Testes em Matlab	39
3.2.2.1	Testes para o Equalizador Paramétrico	40
3.2.2.2	Testes para o <i>crossover</i>	48
3.3	Implementação em <i>hardware</i>	58
3.3.1	Ambiente e kit de desenvolvimento	58
3.3.1.1	O Kit STM32F4 <i>Discovery</i> e <i>Wolfson Pi</i>	58
3.3.1.2	Placa de Áudio <i>M-Audio Fast Track Pro</i>	61
3.3.1.3	Biblioteca CMSIS-DSP	62
3.3.1.4	<i>Softwares</i> de áudio	63
3.3.2	Arquitetura do <i>software</i>	64
3.3.3	Implementação dos algoritmos	65

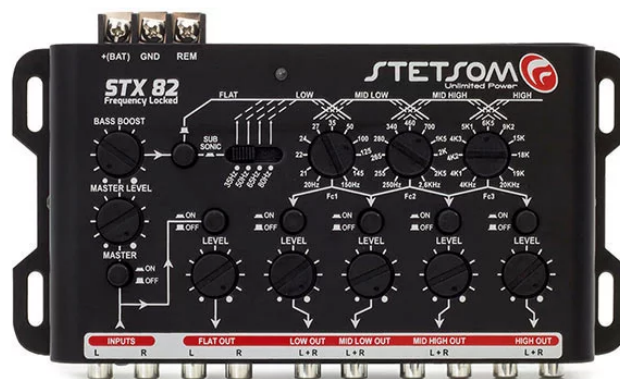
3.3.4	Implementação da interface com o usuário	66
3.4	Testes de implementação	68
4	APRESENTAÇÃO DOS RESULTADOS	70
4.1	Figuras de mérito do sistema	71
4.1.1	Faixa dinâmica e THD	71
4.1.2	Resposta em frequência	72
4.1.2.1	Resposta do equalizador	73
4.1.2.2	Resposta do crossover	75
5	CONSIDERAÇÕES FINAIS	77
5.1	Conclusão	77
5.2	Sugestões de trabalhos futuros	77
	REFERÊNCIAS	79

1 INTRODUÇÃO

Um sistema de áudio de qualidade deve ser capaz de reproduzir toda a informação que um programa musical contém, independente do seu gênero, ou seja, executar todo o seu conteúdo harmônico de forma equilibrada. Sendo o espectro de frequências audíveis pelo ser humano de aproximadamente 20 Hz a 20 kHz, o sistema deve se propor a executar essa faixa. Sabe-se que um único tipo de alto-falante não é capaz de reproduzir, efetivamente, todo esse espectro mas sim uma banda específica, como é o caso dos *woofers* (reprodução dos sons graves), *mid-ranges* (reprodução dos sons médios) e *tweeters* (reprodução dos sons agudos). Mesmo que existam os alto-falantes *full-range*, que se propõem a reproduzir frequências na faixa entre 30 Hz a 20 kHz (AARTS, 2006), estes podem não ser interessantes para algumas aplicações, como sistemas de grandes potências, por sofrerem perda de rendimento.

Para que cada alto-falante opere com melhor qualidade e rendimento, torna-se necessário utilizar filtros seletores de frequência, selecionando a faixa em que cada alto-falante atua (SILVA; SILVA, 2002). Isto gera uma execução com menos distorções, melhor eficiência, diminuindo o risco de dano e contribuindo para um possível aumento de vida útil. Estes sistemas são chamados de filtros de *crossover* ou simplesmente *crossovers*. A Figura 1 mostra um destes dispositivos, que processa o áudio digitalmente (PREMIER SHOP, 2017).

Figura 1 – Dispositivo Crossover com entrada de dois canais, 8 canais de saída e 4 bandas.



Fonte: PREMIER SHOP, 2017

Além da questão do alto-falante trabalhar em uma banda limitada, este pode não apresentar uma resposta plana na banda em que atua, ou ainda, a acústica do ambiente pode ocasionar interferências na resposta em frequência do sistema, como cancelamentos ou sobreposições das ondas sonoras. Para realizar correções pontuais, pode-se utilizar um equalizador, o qual permite a atenuação ou amplificação de uma determinada banda de frequência.

Estes sistemas podem ser puramente analógicos ou atuarem via processamento digital, por *software*. Neste trabalho será desenvolvido um sistema de crossover e equalização utilizando processamento digital de sinais (DSP, do inglês - *Digital Signal Processing*).

1.1 Justificativa

Um alto-falante é um sistema que converte sinais elétricos em sinais acústicos, ou seja, é um transdutor eletroacústico. Dentre as especificações de um alto-falante, uma das mais importantes é a faixa de frequências de operação. Se o aparelho receber uma grande potência em frequências nas quais não é capaz de operar de forma eficiente, serão ocasionadas distorções, superaquecimento das bobinas e eventual falha por queima ou outros danos mecânicos. Além disso, a utilização de um *crossover* no sistema pode trazer mais definição e equilíbrio para o áudio, tornando a experiência de quem ouve mais agradável. Outro problema em sistemas de áudio é a possibilidade de interferências destrutivas. Quando dois ou mais alto-falantes em localizações distintas reproduzem o mesmo sinal, a sobreposição das ondas provenientes das duas fontes pode levar à redução de amplitude ou mesmo significativo cancelamento em determinados pontos do espaço. Existe ainda outro fator, que é a resposta do próprio alto-falante, onde mesmo na sua região de operação ideal pode apresentar irregularidades, ou seja, uma resposta não plana de frequências. Para a correção de problemas desta natureza, pode-se lançar mão dos equalizadores.

1.2 Definição do Problema

A implementação de um sistema de *crossover* e equalizador utilizando DSP via *software* traz benefícios e também desafios. Os benefícios são: flexibilidade, escalabilidade, repetibilidade, redução de tamanho e custo. Porém, existem questões referentes ao processamento, como capacidade e eficiência do *crossover*, ao algoritmo, à implementação em tempo real e às características dos filtros utilizados. No caso deste trabalho os filtros serão do tipo IIR (do inglês - *Infinite Impulse Response*).

As características intrínsecas ao *crossover* também devem ser verificadas e solucionadas. A sobreposição das bandas nas regiões de corte pode causar alterações na resposta total, que deve ser plana. Filtros produzem deslocamentos de fase e possuem taxas de atenuação, o que pode fazer com que as sobreposições não resultem em uma resposta plana.

Como a variação dos parâmetros será em tempo real, ao realizá-la, não poderá haver ruído no áudio oriundo do processamento. Os cálculos feitos para gerar os novos coeficientes dos filtros são realizados quando há variação dos parâmetros, como por exemplo, a frequência. O passo utilizado para realizar a variação também não deve afetar o áudio, como variar a frequência em passos de 10 Hz, por exemplo.

1.3 Objetivo Geral

Implementar, em *hardware*, com processamento digital e em tempo real, um *crossover* com um canal de entrada monofônico e dois canais de saída com as respectivas bandas de frequências (baixas e altas), e um equalizador paramétrico de uma banda. Os parâmetros poderão ser ajustados pelo usuário, em tempo real, por meio de uma interface composta por um *display* e botões para navegação e alteração dos valores. Busca-se, com este trabalho, desenvolver um protótipo de produto, validando a possibilidade de emprego pelo usuário, em uma situação real.

1.4 Objetivos Específicos

São objetivos específicos deste trabalho:

- a) Apresentar os conceitos de sistemas de áudio, processamento digital de sinais e filtros relacionados com este trabalho.
- b) Definir os requisitos e projetar o sistema *crossover* e equalizador.
- c) Implementar inicialmente os algoritmos em MATLAB, validando o cálculo dos coeficientes e a resposta em frequência (magnitude e fase) ao final do processamento.
- d) Implementar o sistema em *hardware*, com botões e *display* para interface com o usuário e ajuste dos parâmetros.
- e) Avaliar o desempenho do sistema em relação à resposta em frequência, precisão dos parâmetros e ajuste pelo usuário.
- f) Apresentar os resultados obtidos.

2 FUNDAMENTAÇÃO TEÓRICA

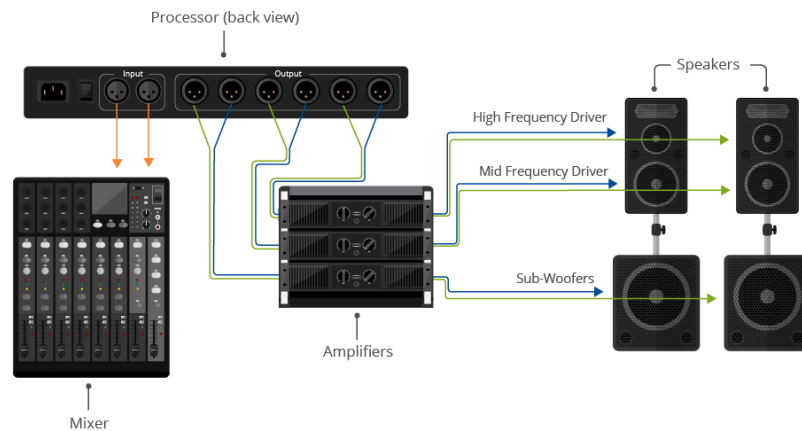
Neste capítulo, será feito um embasamento sobre os principais conceitos e tecnologias envolvidos neste trabalho. Serão apresentados os princípios sobre sistemas de áudio, abordando processadores digitais de sinais, plataforma de implementação deste trabalho, e alto-falantes. Além disso, serão apresentadas as características as quais o processamento tem por objetivo melhorar. Serão abordados conceitos sobre processamento digital de sinais, desde áudio digital, filtros digitais, métodos, algoritmos, *software* e complexidade computacional. O intuito é fornecer uma base para entendimento do que foi necessário para a implementação e aplicação deste trabalho.

2.1 Sistemas de áudio

Um sistema de áudio consiste, basicamente, de uma fonte de sinal de áudio, um sistema de processamento e amplificação desse sinal, e os transdutores de saída, que irão reproduzir, em forma de som, o sinal amplificado.

Existem vários tipos de sistemas para diferentes finalidades. Por exemplo, existem sistemas voltados para estúdio de gravação, instalações multimídia, *home-theaters*, sonorização para eventos ao vivo, automotivos, entre outros. No entanto, para otimizar suas eficiências, melhorar a qualidade e a experiência do ouvinte, muitos sistemas utilizam outros componentes que realizam algum tipo de processamento como equalização, efeitos, dinâmica, *crossover*, correções, entre outros. A Figura 2 mostra um sistema de sonorização, com processador e saídas do *crossover* para amplificação independente para cada banda de frequência.

Figura 2 – Exemplo de sistema básico de PA (*Public Address*) para sonorização de eventos, com processamento, mostrando seus diversos componentes.



Fonte: (BOCKRATH, 2016)

2.1.1 Processadores de áudio

Hoje em dia, a maioria dos sistemas de processamento é digital, sendo menores em tamanho e custo se comparados a modelos analógicos, além de possuírem mais recursos devido à aplicação de algoritmos, cálculos matemáticos e armazenamento de dados. Esses sistemas produzem modificações nos sinais, podendo melhorar a performance de todo o sistema de áudio. São muito utilizados na aplicação de equalização, *crossover*, efeitos, controle de dinâmica e correções de atraso entre caixas de som.

Com o avanço dos *chips* dedicados para DSP, a capacidade de processamento está cada vez mais robusta e barata, oferecendo diversas possibilidades que não são possíveis com circuitos analógicos.

2.1.2 Alto-falantes

Alto-falantes são transdutores eletroacústicos responsáveis pela reprodução do áudio, ou seja, a saída do sistema. De nada adianta ter um bom sistema de processamento e amplificação se os alto-falantes forem de má qualidade, mal dimensionados ou mal utilizados. É preciso entender bem suas faixas de operação em frequência,

potências, adequar as impedâncias e até seus posicionamentos mecânicos nas caixas no ambiente e em relação a outros alto-falantes (SILVA; SILVA, 2002).

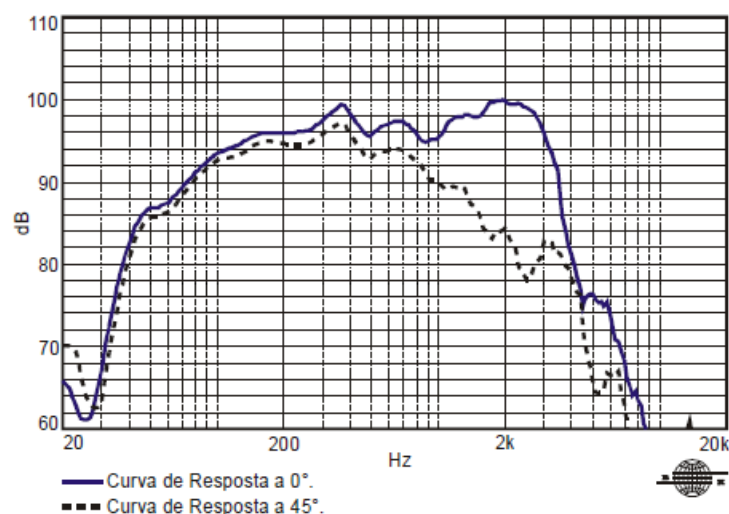
Geralmente, alto-falantes operam com boa eficiência em uma faixa de frequências limitada. Existem modelos que se propõem a reproduzir todo o espectro audível, porém, geralmente são inadequados para atingir frequências muito baixas e para aplicações de grande porte.

Os alto-falantes são classificados de acordo com suas bandas de operação, mas não há uma definição precisa de banda versus tipo, pois variam muito conforme o seu projeto e fabricante. De maneira geral, são classificados como (SMITH, 1999):

- *Sub-woofer*: 20 Hz a 200 Hz. Utilizado quando se necessita executar frequências extremamente baixas.
- *Woofer*: 50 Hz a 3 kHz.
- *Mid-range*: 300 Hz a 5 kHz. Faixa que contém grande parte das frequências.
- *Tweeter*: 3 kHz a 20 kHz.

A Figura 3 apresenta a resposta de um *Mid-range* (SILVA; SILVA, 2002).

Figura 3 – Magnitude da resposta em frequência de um alto-falante modelo WPU1505, medida no seu eixo central (linha contínua), e a 45° (linha tracejada).



Fonte: SILVA, 2002

Na Figura 3, percebe-se que não há uma resposta em frequência plana na banda de operação do alto-falante WPU1505 (média de amplitude em 96 dB, para nível de pressão sonora). Isto ocorre na maioria destes dispositivos (SILVA; SILVA, 2002). Pode-se notar, ainda, que conforme o ouvinte afasta-se para o lado, saindo do eixo frontal, ocorre atenuação das frequências mais altas. Isto ocorre porque ondas acústicas com frequências maiores possuem maior diretividade. Portanto, muitos fatores podem afetar a resposta acústica de sistemas de áudio. Logo, quanto maior a precisão dos transdutores em função do conteúdo harmônico do sinal, associado a possibilidades de correção, mais fidelidade ao áudio original o sistema apresentará.

2.2 Processamento Digital de Sinais

Sendo um sinal uma variável que contém algum tipo de informação, no caso o som, esse sinal pode ser aplicado a um sistema, que dependendo da sua finalidade, pode apresentar um sinal diferente na saída, modificando-o.

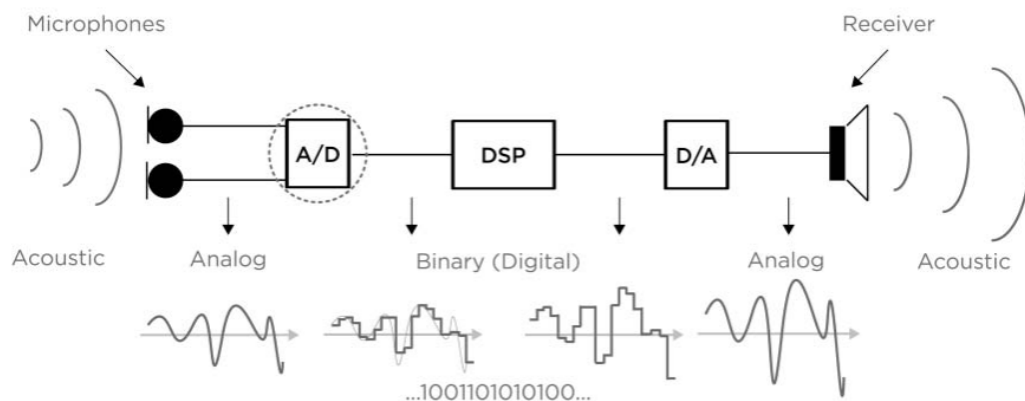
Um sinal digital é a representação de um sinal contínuo por valores em intervalos de tempo definidos, que pode tanto ter sua origem na amostragem de um sinal analógico real, como pode também ser gerado diretamente no domínio digital. No caso da amostragem de um sinal analógico, como por exemplo a voz captada por um microfone, o sistema de amostragem faz aquisições dos valores de amplitude em determinados instantes de tempo (espaçados de acordo com o período de amostragem), e esses valores são convertidos (quantização) e armazenados em representação binária. O sinal que era contínuo no tempo e amplitude, se torna um sinal digital discreto no tempo e em amplitude.

O processamento digital trabalha com os aspectos dos sinais utilizando técnicas digitais. Os valores são computados por um processador, utilizando as amostras armazenadas e equações numéricas. Diferentemente do processamento analógico, pode-se realizar qualquer tipo de cálculo numérico sobre os sinais digitais, uma vez que se tenha o *hardware* adequado para isso (DINIZ; SILVA; NETTO, 2004). Esta seção apresentará os conceitos sobre processamento digital para sistemas de áudio.

2.2.1 Áudio digital

Como mencionado anteriormente, os sinais podem ser representados digitalmente, originando-se tanto pela conversão analógica/digital, como também pela síntese já no domínio digital. A seguir, serão explicados os conceitos de amostragem e quantização, que dizem respeito à conversão analógica-digital, e também o conceito inverso, a conversão digital-analógica, que é o processo aplicado ao sinal, para que este possa ser novamente reproduzido de forma audível em alto-falantes. A Figura 4 apresenta o fluxo de um sinal que é adquirido no meio acústico, convertido para digital, processado e convertido novamente em acústico.

Figura 4 – Sistema de áudio digital, com conversores de entrada e saída e o processamento digital utilizando os valores binários amostrados.

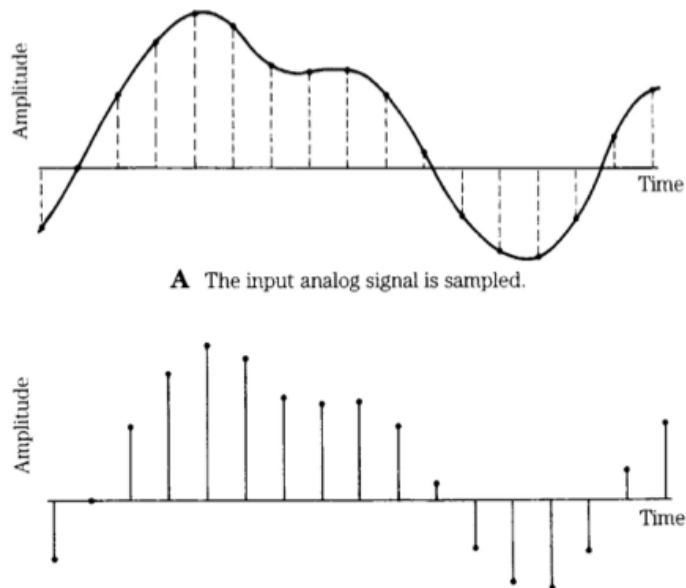


Fonte: (BAEKGAARD et al., 2013)

Para que um sinal seja amostrado, ele deve passar por um conversor analógico/digital (ADC, do inglês *Analog to Digital Converter*), e ter uma taxa de amostragem maior do que o dobro da sua frequência máxima, para que não haja perda de resolução e erros por recobrimento (*aliasing*) (LAVRY, 2004). As taxas mais comuns para áudio profissional são de 44100 Hz e 48000 Hz. Depois do processo de amostragem no tempo, também a amplitude é discretizada, sendo que cada amostra é representada por um valor dentro de uma faixa limitada pelo número de bits empregado, o chamado *bit-depth*. Em áudio, as resoluções de quantização mais utilizadas são de 16 ou 24 bits (ZAIDI, 2010?). A Figura 5 mostra os pontos de um sinal que são adquiridos a

cada intervalo de amostragem. Esses valores serão então convertidos para binário, caracterizando um sinal discreto no tempo e em amplitude, um sinal digital.

Figura 5 – Amostragem de um sinal. Em A, o sinal analógico, mostrando os instantes que serão amostrados. Em B, as amostras resultantes que serão armazenadas.



Fonte: (ECMT, 2005)

Para que o áudio digitalizado possa ser novamente ouvido em algum dispositivo, o sinal digital deve ser convertido para analógico. Da mesma maneira que o ADC, um conversor digital-analógico (DAC, do inglês *Digital to Analog Converter*) também é sincronizado por um sinal de relógio (*clock*), e possui resolução de saída, geralmente de 16 ou 24 bits, que representa os passos de amplitude. Logo, a forma de onda convertida para analógica terá suas variações em degraus, e a aplicação de um filtro passa-baixas amortiza essas variações.

Sistemas de áudio que possuem conversão AD/DA muitas vezes utilizam CODECs (do inglês - *Coder Decoder*) de áudio, os quais são dispositivos que codificam os dados digitais em pacotes de protocolo de comunicação e vice-versa (STUART, 1998). Um protocolo muito utilizado em áudio é o I2S (do inglês - *Inter-IC Sound*), desenvolvido pela *Philips* (PHILIPS, 1986). Este protocolo leva a informação do CODEC para a unidade de processamento, que irá realizar a tarefa que lhe é designada. Para enviar o áudio para a saída, o processador envia os dados via protocolo novamente ao

CODEC, que é responsável por disponibilizar o áudio na saída. CODECs podem ser implementados apenas em *software* ou em *hardware*, que pode já possuir um conversor AD/DA.

2.2.2 Filtros seletores de sinais

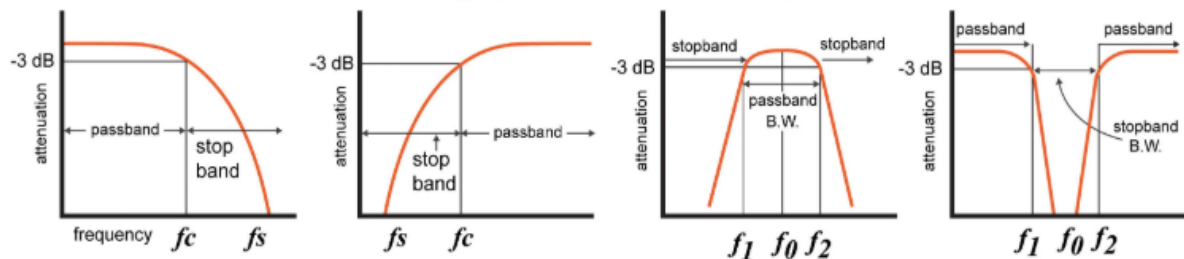
Filtros são utilizados em praticamente todos os sistemas eletrônicos, sendo um dos assuntos clássicos em processamento de sinais. Consistem em selecionar componentes de frequência do sinal, que constituem a chamada banda de passagem, atenuando uma outra faixa de frequências, chamada de banda de rejeição (NOCETI FILHO, 1998). Uma aplicação clássica é a remoção de ruídos de alta frequência em sinais através da utilização de filtros passa-baixas (IFEACHOR; JERVIS, 2002). Os tipos de filtros seletores de sinais comumente utilizados são:

- Passa-baixas (*low pass*): permitem a passagem das frequências abaixo de uma frequência de corte e atenuam as que estão acima desta frequência.
- Passa-altas (*high pass*): permitem a passagem das frequências acima de uma frequência de corte e atenuam as mais baixas (inverso do passa-baixas).
- Rejeita-faixa (*band stop*): atenua uma faixa de frequência, possuindo duas frequências de corte, uma inferior e uma superior.
- Passa-faixa (*band pass*): atenua frequências baixas e altas, também possui duas frequências de corte.
- *Peak e notch*: amplifica ou atenua, respectivamente, em torno de uma frequência específica, com certa largura da banda de corte, dependendo da aplicação. (HANNA, 1994).

Essa classificação considera que idealmente não há qualquer distorção na banda de passagem e que a frequência de corte é um parâmetro que determina o limite entre deixar passar ou não a energia de uma banda de frequências do sinal. Filtros realizáveis, ou seja, os que são realmente implementados, não apresentam esse

comportamento abrupto passa/não passa. A Figura 6 mostra as magnitudes das respostas em frequências de quatro tipos de filtros, em que nota-se esse comportamento de atenuação não-abrupta.

Figura 6 – Magnitudes das respostas em frequência dos tipos mais utilizados de filtros seletores de sinais: (a) passa-baixas; (b) passa-altas; (c) passa-faixa; (d) rejeita-faixa



Fonte: (DAVIS, 2017)

Todo filtro possui uma taxa de atenuação associada a sua ordem. Quanto maior a ordem de um filtro, maior essa taxa e maior seletividade este apresentará. O conceito de ordem de um filtro vem da sua função de transferência, a transformada de Laplace (ou a transformada z , no caso de filtros digitais) de sua resposta ao impulso.

É importante destacar que, além da magnitude, filtros também afetam a fase dos sinais.

2.2.3 Filtros digitais

Um filtro digital é a implementação de um algoritmo matemático, feito em *hardware* e/ou *software*, que processa o sinal de entrada de modo a se obter as características do filtro desejado. A preferência pelo uso de filtros digitais vem crescendo, devido às suas vantagens, que são (IFEACHOR; JERVIS, 2002):

- A performance não varia com fatores do ambiente externo, como a temperatura, da forma que ocorre nos analógicos.
- A resposta em frequência pode ser ajustada automaticamente por *software*, como é o caso de filtros adaptativos.

- Tanto o sinal de saída quanto o de entrada podem ser armazenados para uso posterior de forma muito mais simples que em um sistema analógico.
- Pode-se implementá-los em processadores cada vez menores e mais baratos.
- O desempenho do filtro pode ser replicada unidade a unidade.
- Pode-se alterar a quantidade de canais e realizar modificações sem modificar o *hardware*.

E quanto às desvantagens, estas são:

- Limitação de velocidade de amostragem e processamento, que impede que frequências muito altas possam ser processadas. Isto depende da tecnologia disponível e da aplicação.
- Efeitos do tamanho de palavra finito, que pode truncar valores e propagar ruídos ao longo do processamento. Fenômeno mais recorrente em sinais que exigem extrema precisão.
- Tempo de desenvolvimento geralmente maior do que filtros analógicos.

Filtros digitais são sistemas lineares e invariantes de tempo discreto descritos por uma equação de diferenças a coeficientes constantes da forma (INGLE; PROAKIS, 2000)

$$y(n) = \sum_{k=0}^M b_k x(n-k) - \sum_{k=1}^N a_k y(n-k), \quad (2.1)$$

onde $y(n)$ representa o sinal de saída, $x(n)$, o sinal de entrada, e os coeficientes a_k e b_k caracterizam a resposta do filtro, sendo a_k associados aos polos e b_k aos zeros da função de transferência.

De modo geral, filtros digitais podem ser classificados de acordo com a característica de suas respostas ao impulso. Desse modo, há filtros FIR (do inglês - *Finite*

Impulse Response) e IIR (do inglês - *Infinite Impulse Response*), o qual será utilizado neste trabalho. Em um filtro FIR, os coeficientes a_k de (2.1) são nulos.

Em um filtro IIR, o valor da amostra de saída $y(n)$ no instante n é a função das amostras passadas da saída $y(n - k)$, bem como da amostra presente e das amostras passadas da entrada $x(n - k)$, sendo k o número de amostras atrasadas. A dependência com as amostras passadas da saída descreve uma estrutura recursiva ou com realimentação¹.

Para o caso de filtros de segunda ordem, a equação expandida é:

$$y(n) = b_0x(n) + b_1x(n - 1) + b_2x(n - 2) - a_1y(n - 1) - a_2y(n - 2) \quad (2.2)$$

Existem algumas formas de realizar o filtro de 2.2. Quando são computados primeiro os termos associados aos coeficientes b_k e depois os associados a a_k , tem-se a forma direta I. Essa forma é numericamente mais robusta para implementação (KEIL CMSIS-DSP, 2010) e será adotada neste trabalho.

Com filtros FIR, pode-se obter resposta de fase linear. Contudo, para um mesmo conjunto de requisitos de projeto, filtros FIR apresentam ordem maior que os IIR, levando a um custo computacional mais elevado.

2.2.4 Crossover

Em um sistema de filtros *crossover* ocorre a sobreposição de diferentes filtros, sendo estes mutuamente exclusivos, ou seja, no caso de dois canais um passará as altas e o outro passará as baixas frequências (LINKWITZ, 1976); este trabalho trata exatamente deste caso. Será implementado um filtro passa-altas e um passa-baixas, com a mesma frequência de corte, e com atenuação de modo que ao somar as duas bandas resulte em uma resposta plana em frequência.

A topologia de filtro utilizada é a *Butterworth* de segunda ordem, por ser mais usual em aplicações em áudio e com mais referências disponíveis; a outra topologia

¹ Para ser exato, também há filtros FIR implementados de forma recursiva.

também utilizada, a *Linkwitz-Riley*, deriva desta. O filtro passa-baixas desta topologia deriva da seguinte função de transferência no domínio s:

$$H_{pb}(s) = \frac{1}{s^2 + \frac{s}{Q} + 1} \quad (2.3)$$

E o passa-altas deriva da seguinte função de transferência:

$$H_{pa}(s) = \frac{s^2}{s^2 + \frac{s}{Q} + 1} \quad (2.4)$$

onde Q é o fator de qualidade do filtro, fazendo o ajuste da sua largura de banda. Para o cálculo dos coeficientes, são utilizadas as equações 2.5 e 2.6, sendo f_0 a frequência de corte do filtro e f_s , a frequência de amostragem. Elas foram convertidas para o domínio digital, utilizando o método da transformação bilinear (BLT, do inglês - *Bilinear Transform*), a partir das equações 2.3 e 2.4. A necessidade de pré-distorção para compensar a distorção (*warping*) na resposta em frequência deste método já foi levada em consideração.

$$\omega_0 = \frac{2\pi f_0}{f_s} \quad (2.5)$$

$$\alpha = \frac{\sin(\omega_0)}{2Q} \quad (2.6)$$

Para o passa-baixas, os coeficientes do filtro de segunda ordem são:

$$b_0 = \frac{1 - \cos(\omega_0)}{2} \quad (2.7)$$

$$b_1 = 1 - \cos(\omega_0) \quad (2.8)$$

$$b_2 = b_0 \quad (2.9)$$

$$a_0 = 1 + \alpha \quad (2.10)$$

$$a_1 = -2\cos(\omega_0) \quad (2.11)$$

$$a_2 = 1 - \alpha \quad (2.12)$$

Para o passa-altas, os coeficientes são:

$$b_0 = \frac{1 + \cos(\omega_0)}{2} \quad (2.13)$$

$$b_1 = -(1 - \cos(\omega_0)) \quad (2.14)$$

$$b_2 = b_0 \quad (2.15)$$

$$a_0 = 1 + \alpha \quad (2.16)$$

$$a_1 = -2\cos(\omega_0) \quad (2.17)$$

$$a_2 = 1 - \alpha \quad (2.18)$$

Estes coeficientes são então normalizados, dividindo-se todos por a_0 , para serem utilizados na equação 2.5, e representam a topologia de um *Butterworth* de segunda ordem. Pode-se obter quarta ordem, ou ordens mais elevadas, aplicando novamente o filtro ao sinal de saída, o que é conhecido como cascadeamento.

Para que o *crossover* funcione corretamente, deve-se obter uma resposta plana em frequência, e na região de sobreposição, não pode haver cancelamentos ou

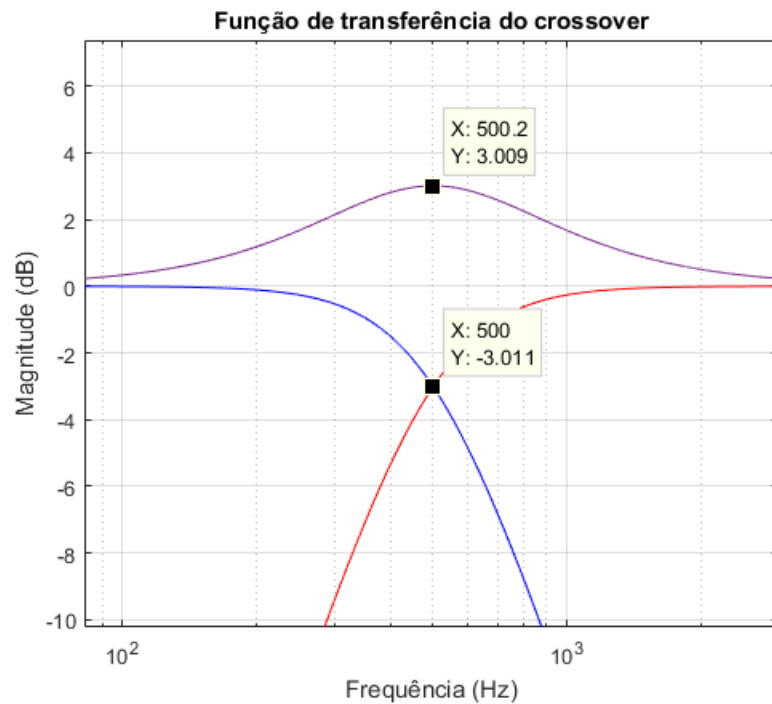
aumento da amplitude. Os tópicos a seguir, mostram as características de *crossovers Butterworth* de segunda ordem, e também os de quarta ordem, conhecidos como topologia *Linkwitz-Riley*. Foram escolhidas estas estruturas por serem bem difundidas no meio profissional e também para fins de comparação.

2.2.4.1 *Butterworth de segunda ordem*

A característica deste tipo de filtro é apresentar um ganho de -3 dB (ou seja, uma atenuação de 3 dB) na frequência de corte, com taxa de atenuação de 40 dB/dec (ou 12 dB/oitava). Porém, no caso do *crossover*, como a soma de dois canais de áudio resulta em um ganho de 6 dB, deve-se fazer com que na frequência de corte haja um ganho de -6 dB, e isto pode ser ajustado pelo fator de qualidade Q.

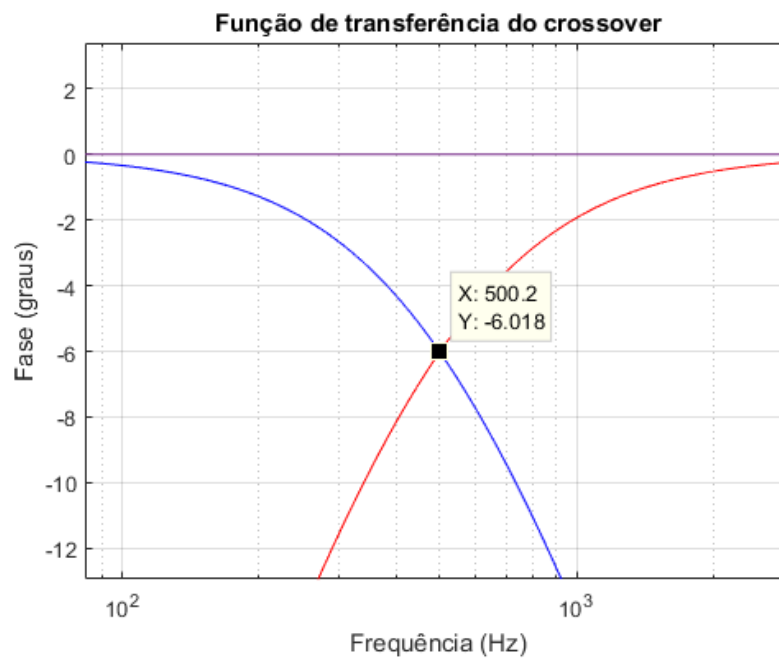
Outro fator que deve ser ajustado para o *crossover* é a questão do defasamento de 180° que existe entre o passa-baixas e o passa-altas. Na região de sobreposição, haverá cancelamentos, sendo muito acentuado na frequência de corte. Uma solução adotada em circuitos analógicos é a inversão da polaridade do passa-altas, que passará a ficar com 360° de defasagem em relação ao passa-baixas (LINKWITZ LAB, 2006). A Figura 7 mostra as curvas de magnitude da resposta em frequência de um *crossover Butterworth* de segunda ordem com ganho de -3 dB na frequência de corte, com inversão de polaridade, além de apresentar a soma dos canais. A Figura 8 apresenta o mesmo conjunto de curvas, mas para a configuração com ganho de -6dB na frequência de corte, obtida com $Q=0,5$.

Figura 7 – Crossover Butterworth com ganho de -3dB e $Q = 0,707$. As curvas apresentadas são: o filtro passa-baixas (azul), o passa-altas (vermelho), a soma das duas (roxo), resultando em uma saliência de até 3 dB na região de corte.



Fonte: AUTOR (2020)

Figura 8 – Crossover Butterworth com corte em -6 dB e $Q = 0,5$. As curvas apresentadas são: o filtro passa-baixas (azul), o passa-altas (vermelho), e a soma das duas (roxo), resultando em uma resposta plana



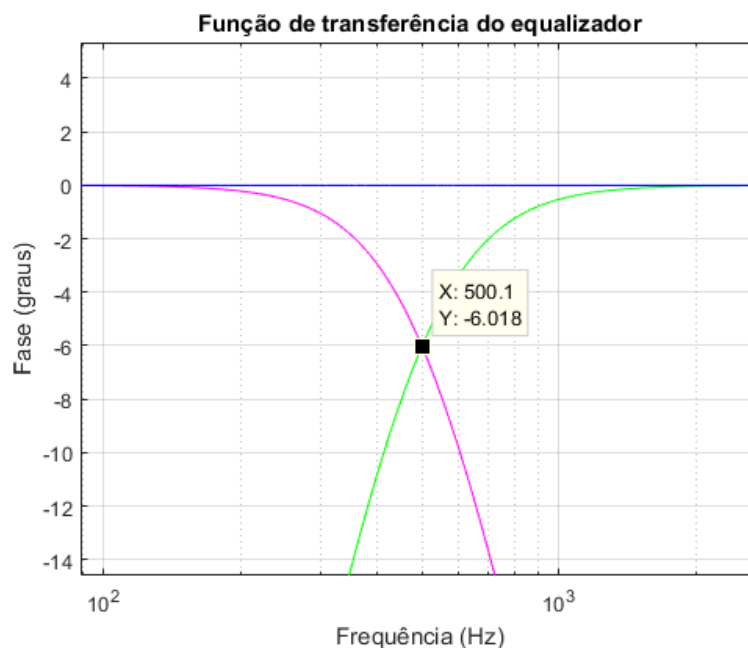
Fonte: AUTOR (2020)

2.2.4.2 Linkwitz-Riley

Esta topologia é uma implementação de filtros *Butterworth* de quarta ordem aplicada a *crossovers*. As características de quarta ordem tornam a resposta desta configuração ideal por natureza, sem a necessidade de ajustes, pois apresentam atenuação de 6 dB na frequência de corte e a defasagem entre os filtros passa-altas e passa-baixas é de 360° .

Para obter o filtro de quarta ordem, basta aplicar o de segunda ordem duas vezes, em cascata. A figura 9 mostra as curvas de um *crossover Linkwitz-Riley* (de quarta ordem). Não houve necessidade de inversão de polaridade.

Figura 9 – Crossover Linkwitz-Riley. As curvas apresentadas são: o filtro passa-baixas (magenta), o passa-altas (verde), e a soma das duas (azul), resultando em uma resposta plana.

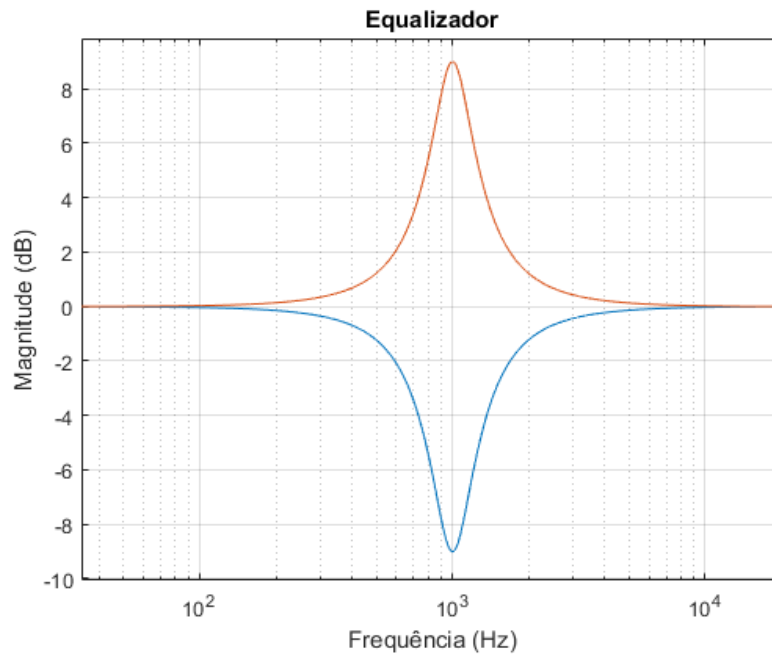


Fonte: AUTOR (2020)

2.2.5 Equalizador paramétrico

É um filtro que possibilita a amplificação ou atenuação em uma frequência, e possui a possibilidade do ajuste da frequência de central f_c , do ganho G , e o fator de qualidade Q (quanto maior o valor de Q , mais seletivo é o equalizador).

Figura 10 – Equalizador paramétrico com $f_c = 10$ kHz, $Q = 3$, e ganho/corte de 9 dB (respectivamente, em vermelho e azul).



Fonte: AUTOR (2020)

As equações, no domínio s , para o equalizador são:

$$H_{Ganho}(s) = \frac{s^2 + \frac{Ks}{Q} + \Omega_0^2}{s^2 + \frac{s}{Q} + \Omega_0^2} \quad (2.19)$$

$$H_{Corte}(s) = \frac{s^2 + \frac{s}{Q} + \Omega_0^2}{s^2 + \frac{Ks}{Q} + \Omega_0^2} \quad (2.20)$$

$$\Omega_0 = 2\pi f_0 \quad (2.21)$$

$$K = 10^{\frac{G}{20}} \quad (2.22)$$

Sendo G o ganho em dB na frequência central f_c . A partir das funções de transferência anteriores, as equações no domínio digital para o cálculo dos coeficientes são definidas. Primeiramente, as equações auxiliares:

$$B = \frac{f_0}{Q} \quad (2.23)$$

$$a = \frac{1 - \tan\left(\frac{\pi B}{F_s}\right)}{1 + \tan\left(\frac{\pi B}{F_s}\right)} \quad (2.24)$$

$$b = \cos\left(\frac{2\pi f_0}{F_s}\right) \quad (2.25)$$

Para os coeficientes de um equalizador de segunda ordem, o cálculo já é normalizado para a_0 :

$$b_0 = 0,5(1 + a + K - Ka) \quad (2.26)$$

$$b_1 = b + ba \quad (2.27)$$

$$b_2 = 0,5(1 + a - K + Ka) \quad (2.28)$$

$$a_1 = b_1 \quad (2.29)$$

$$a_2 = a \quad (2.30)$$

2.2.6 Processamento de áudio e complexidade computacional

Para que o processamento de sinais seja realizado, é necessário que as amostras estejam armazenadas e disponíveis. Existem duas formas de processamento em áudio, uma é quando todo o sinal está disponível, previamente gravado, com todas as amostras já armazenadas. Este é o caso de um processamento em MATLAB, por exemplo. Outra forma, é o processamento em tempo real, ou seja, é continuamente realizado enquanto o som está sendo executado. As amostras são armazenadas em um bloco de memória, chamado *buffer*, que geralmente tem um tamanho dado por uma potência de 2, entre 256 e 2048 amostras. Os tamanhos podem variar, de acordo com

o propósito do processamento, pois quanto maior o *buffer*, mais tempo é necessário para o processamento, ocasionando latência. No contexto deste projeto, a complexidade computacional está relacionada a quanto tempo e recursos do *hardware* são necessários para realizar o processamento de modo que as requisições de tempo sejam atendidas. No caso, o tempo máximo necessário para computar cada *buffer* é o tempo que não cause ruídos perceptíveis.

O processador pode executar as operações matemáticas por ponto fixo ou ponto flutuante, porém há uma diferença no tempo de cálculo entre estes dois tipos, sendo, geralmente, o ponto flutuante mais custoso, necessitando mais ciclos. Os processadores mais modernos, como os da família ARM-M4 (utilizado no projeto), apesar de não serem dedicados para DSP, apresentam otimizações em *hardware* para as operações de multiplicação e soma (MAC - multiplicação e acumulação), e apresentam também pouca diferença em termos de tempo de execução entre as operações entre ponto fixo e flutuante (32 bits).

2.2.7 Algoritmos

Algoritmo é uma sequência de passos, bem definidos, necessários para que uma tarefa seja realizada do início ao fim. Todo *software*, em sua essência, possui um algoritmo, pois este executa uma tarefa e necessita de instruções absolutamente precisas. Um algoritmo extremamente importante para a análise de frequências é a FFT (*Fast Fourier Transform*), que consiste em um método computacional eficiente para o cômputo da DFT (*Discrete Fourier Transform*). Este algoritmo realiza a transformação do sinal, do domínio do tempo para o domínio da frequência, o que possibilita a visualização de todo o espectro, com cada componente de frequência. Ao longo do desenvolvimento é mencionado o uso da FFT para a obtenção dos gráficos em frequência dos sinais, pois é o termo mais usual na prática.

Este trabalho busca implementar, de forma eficiente, filtros IIR de segunda ordem, os descritos pela equação 2.5. O próximo capítulo tratará em mais detalhes o algoritmo desenvolvido.

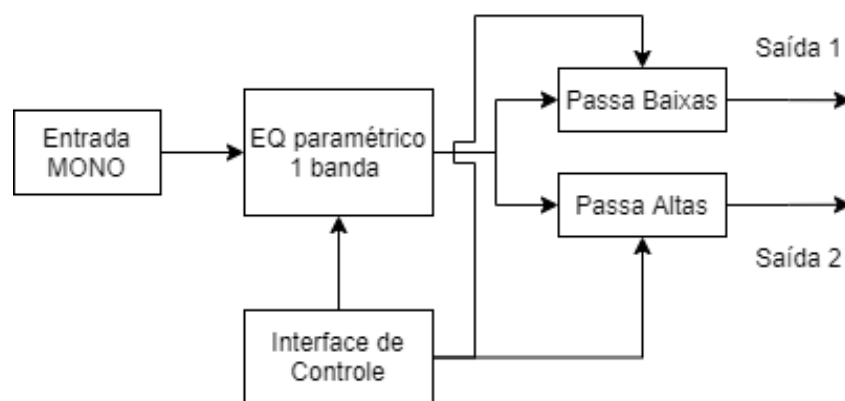
3 DESENVOLVIMENTO

Este capítulo apresenta a implementação do projeto, desde sua concepção, requisitos, recursos para o desenvolvimento, aplicações, validações, soluções e testes.

3.1 Concepção

A partir do objetivo geral, foi desenvolvida a prova de conceito de um protótipo de produto que possa ser utilizado por um usuário final, em uma situação real. Foi desenvolvido um sistema de filtros de áudio com o kit de *hardware* STM32F4 *Discovery* mais a placa de áudio *WolfsonPi*, além de *display* e botões para interface com o usuário. A Figura 11 apresenta o diagrama de blocos do sistema, onde será processada uma entrada mono e as saídas se darão nos dois canais disponíveis na placa *WolfsonPi* (saída 1 no canal direito e saída 2 no canal esquerdo, ambas em mono). Os controles agirão sobre os parâmetros de cada bloco específico, como será mencionado na Seção 3.1.1.

Figura 11 – Diagrama de blocos do sistema



Fonte: AUTOR (2020)

3.1.1 Requisitos do sistema

Os requisitos foram estabelecidos a partir das necessidades básicas para que o sistema atenda a demanda de uma aplicação prática. A disponibilidade de recursos para sua realização também influenciou nas definições, como a quantidade de canais do *crossover*. Os requisitos definidos são:

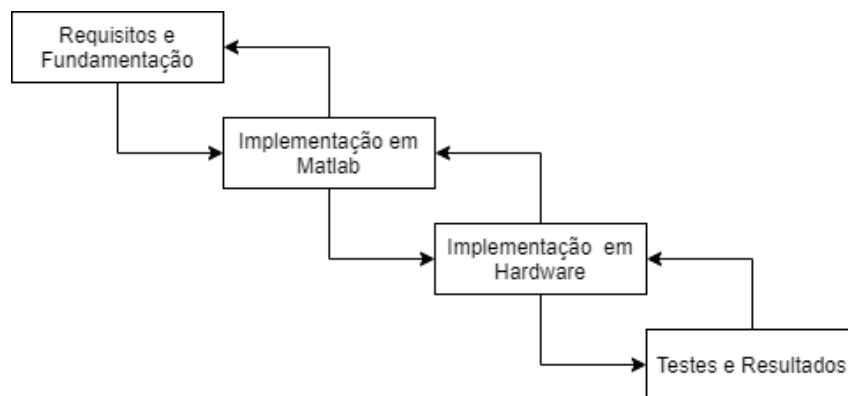
- Um *crossover* de duas bandas/canais de saída, para um canal de entrada mono, com ajuste de frequência, seleção entre *Butterworth* e *Linkwitz-Riley* e opção de liga/desliga para o filtro. A faixa de frequências de corte vai de 60 Hz a 3 kHz, com taxa de variação de 20 Hz.
- Um equalizador paramétrico para uma frequência central, com ajuste de frequência, ganho, fator de qualidade Q e opção de liga/desliga para o filtro. A faixa de frequências de ajuste para a frequência central está entre 60 Hz e 8 kHz, com taxa de variação de 20 Hz. O ganho G será de +/- 18 dB, Q será de 1 a 15, onde G e Q terão taxa de variação de passo 1.
- Controle do volume de saída de 0 a 100 com passo 5.
- Interface com o usuário, por meio de *display* e botões para navegação.
- Distorção harmônica menor do que 0,5%; valor escolhido pois a percepção humana está na faixa de aproximadamente 1%, e pode variar com a frequência, amplitude e outros fatores (WILLIAMS, 2017). O valor de 0,5% foi estabelecido de maneira a ficar em um patamar mediano entre todas estas variações, pois não serão feitas análises aprofundadas neste quesito nos testes.

3.2 Projeto

A metodologia de execução pretendida para este trabalho, bem como o seu cronograma inicial, foi feita pelo modelo em cascata, onde cada etapa do projeto é bem definida e executada por completo, linearmente, antes da etapa seguinte.

No entanto, esta é uma estrutura que apresenta limitações, e sofre com o risco de que, se ocorrer algum problema nos estágios finais por uma escolha equivocada nos estágios anteriores, o projeto tem que ser praticamente refeito. Foram tomadas, então, algumas decisões de modo a tornar o desenvolvimento mais dinâmico como, por exemplo, após cada nova implementação feita em MATLAB, esta já era testada em *hardware*, isoladamente, para verificar se haveria algum problema. Esse processo iterativo facilitou a detecção rápida de problemas e as correções. Basicamente, o diagrama de execução ficou como mostra a Figura 12.

Figura 12 – Diagrama da metodologia de projeto utilizada.



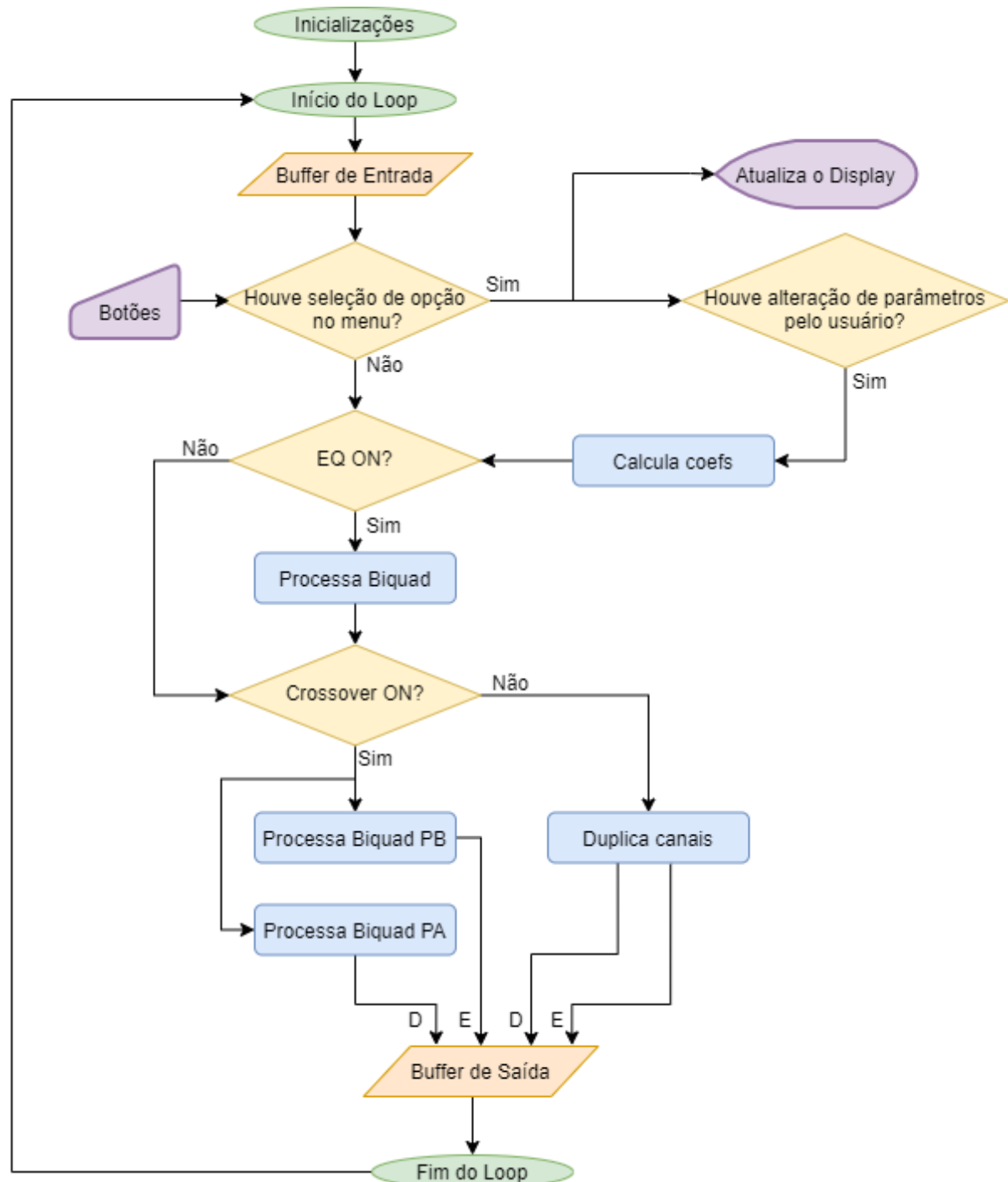
Fonte: AUTOR (2020)

Para fins de organização, a descrição nas subseções a seguir apresenta o desenvolvimento de forma linear, como o modelo em cascata. Os códigos do projeto podem ser encontrados no repositório do *github* (TAVARES, 2020).

3.2.1 Definição do algoritmo

Inicialmente, foi preciso definir os blocos que compõem o algoritmo. Primeiro, foi definida a ordem em que o processamento seria feito, de acordo com a sequência dos filtros (equalizador, passa-baixas e passa-altas). Em seguida, a forma de implementar a equação 2.5, de modo a aplicar cada filtro, a partir de coeficientes conhecidos, e, por fim, calculá-los dinamicamente, quando houvesse variação de algum dos seus parâmetros, utilizando as equações apresentadas no capítulo anterior. O fluxograma da Figura 13 mostra a concepção do algoritmo.

Figura 13 – Fluxograma do algoritmo de processamento.



Fonte: AUTOR (2020)

3.2.2 Testes em Matlab

Optou-se por realizar testes com o MATLAB, onde pode-se desenvolver códigos em alto nível de abstração. Além disso, pode-se obter e operar funções de trans-

ferência com facilidade e também gerar gráficos.

Inicialmente, foi implementado o equalizador, calculando seus coeficientes pelas equações 2.24 a 2.33 da Seção 2.2.4, com seus parâmetros fixos. Os coeficientes obtidos foram aplicados com a função *filter* do MATLAB em um sinal *sweep* (senóide com varredura da frequência ao longo do tempo). Foi obtida resposta em frequência (com a função *freqz* do MATLAB) dos sistemas e também a transformada de Fourier (FFT) dos sinais de entrada e saída do filtro. A análise foi feita variando todos os parâmetros (f_c , G e Q) para verificar a resposta.

Para o *crossover*, foi realizado o mesmo procedimento do equalizador, de maneira isolada, calculando os coeficientes pelas equações 2.8 a 2.21 da Seção 2.2.3. Neste caso, foi analisada a resposta do passa-altas e do passa-baixas para filtros de segunda e quarta ordens, e também a soma dos canais. Após as análises isoladas, foi feita a aplicação de todos os filtros, onde o sinal de saída do equalizador é repetido para servir de entrada para os dois filtros do *crossover*, obtendo-se, por fim, a saída de dois canais.

3.2.2.1 Testes para o Equalizador Paramétrico

A seguir, serão apresentados os resultados obtidos para um equalizador com $f_c = 500$ Hz, G = 9 dB e Q = 3. A Tabela 1 apresenta os coeficientes calculados para estes parâmetros, já normalizados por a_0 :

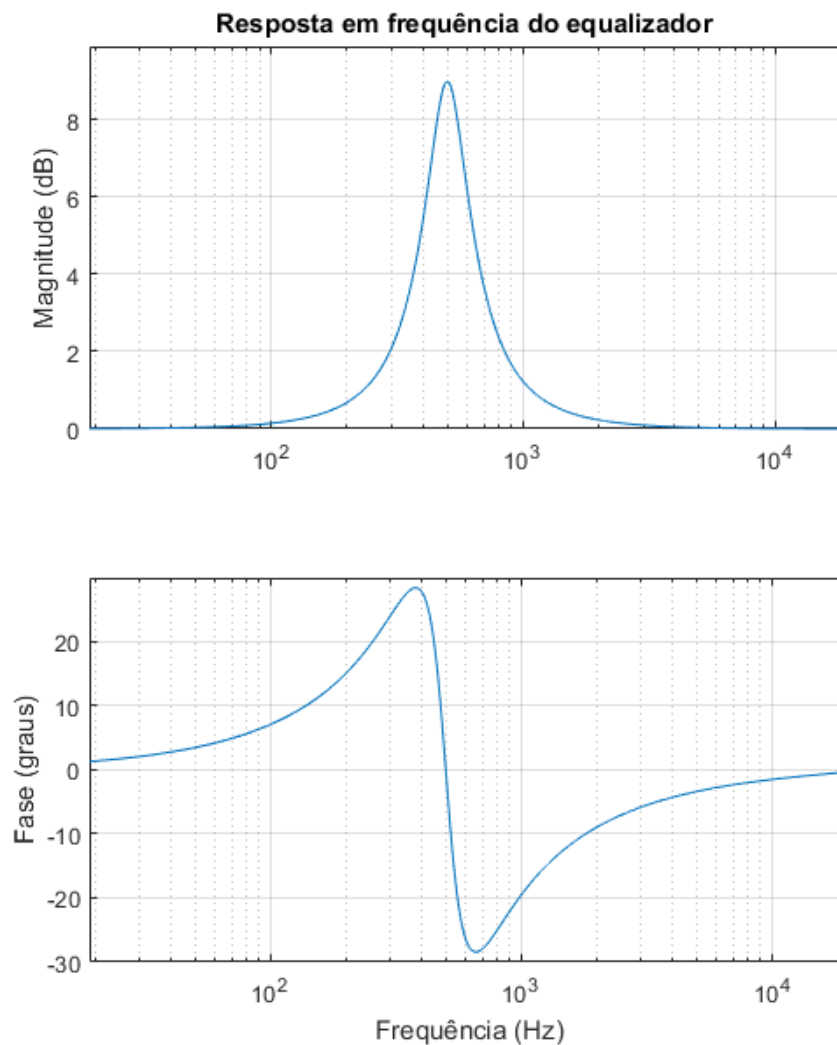
Tabela 1 – Coeficientes para o equalizador com $f_c = 500$ Hz, G = 9 dB e Q = 3.

Coeficiente	Valor
a_0	1
a_1	-1.9742
a_2	0.9784
b_0	1.0196
b_1	-1.9742
b_2	0.9588

Fonte: AUTOR (2020)

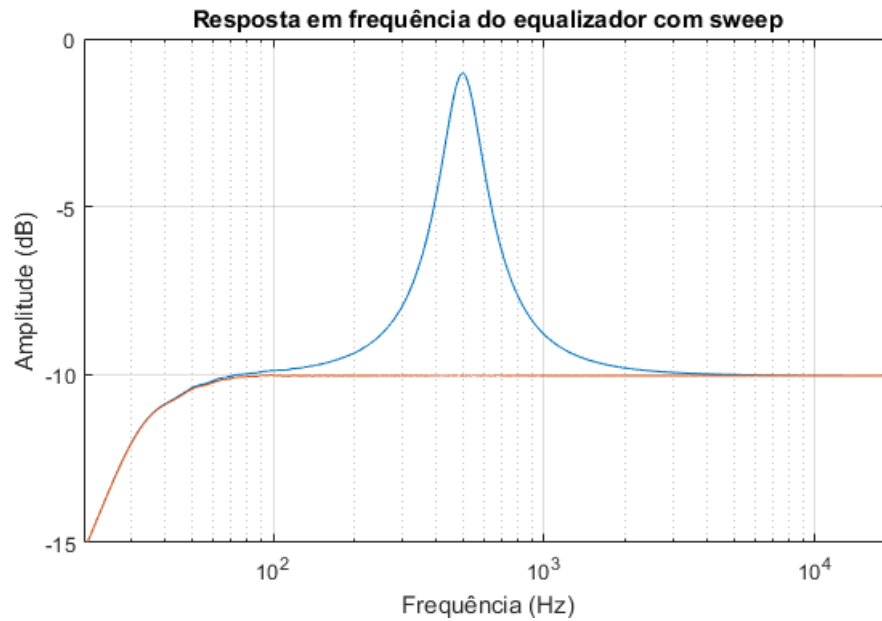
As Figuras 14 e 15 apresentam gráficos feitos em MATLAB para o equalizador, de sua resposta em frequência e FFT dos sinais de entrada e saída. Na Figura 15 e nas demais figuras que mostram a FFT do sinal *sweep*, a amplitude na parte plana está em -10 dB. O nível do sinal de entrada foi reduzido propositalmente para não ocorrer saturação no ganho do equalizador. Em todos os testes com *sweep* foi utilizado o mesmo sinal.

Figura 14 – Resposta em frequência com magnitude e fase. Nota-se o ganho de 9 dB e a fase passando por zero em 500 Hz. $Q = 3$.



Fonte: AUTOR (2020)

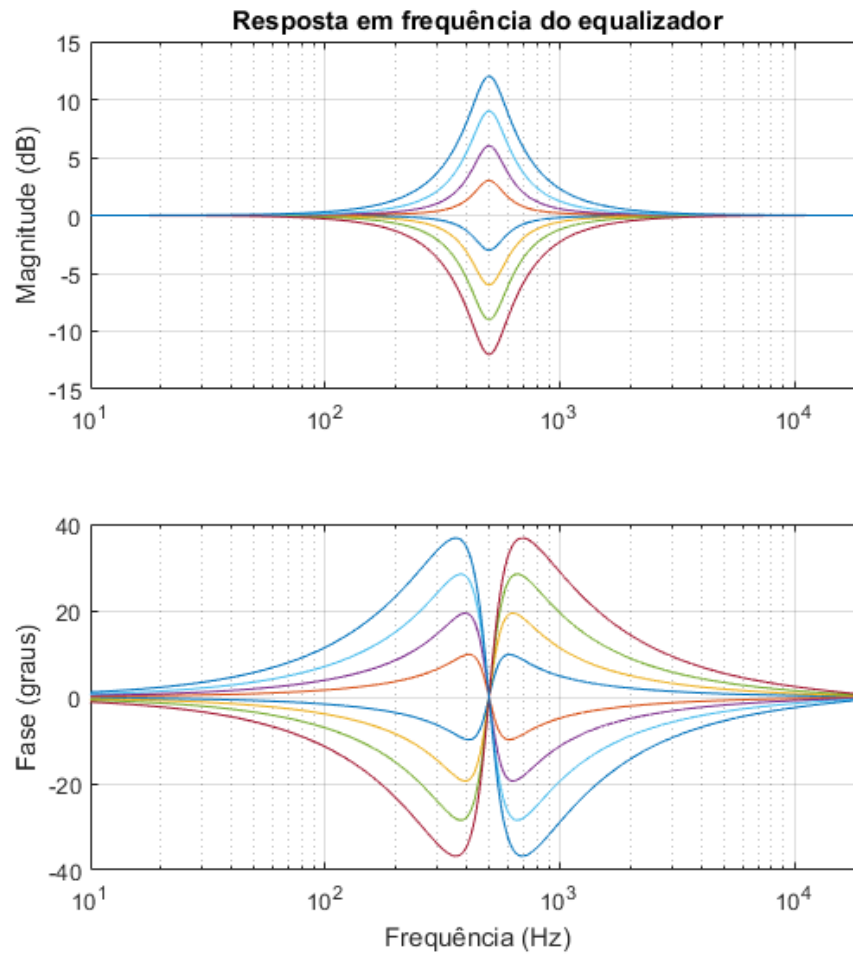
Figura 15 – FFT dos sinais de entrada (vermelho) e de saída (azul) do sinal sweep de 20 Hz a 20 kHz. Apresenta ganho de 9 dB em 500 Hz. Fator Q = 3.



Fonte: AUTOR (2020)

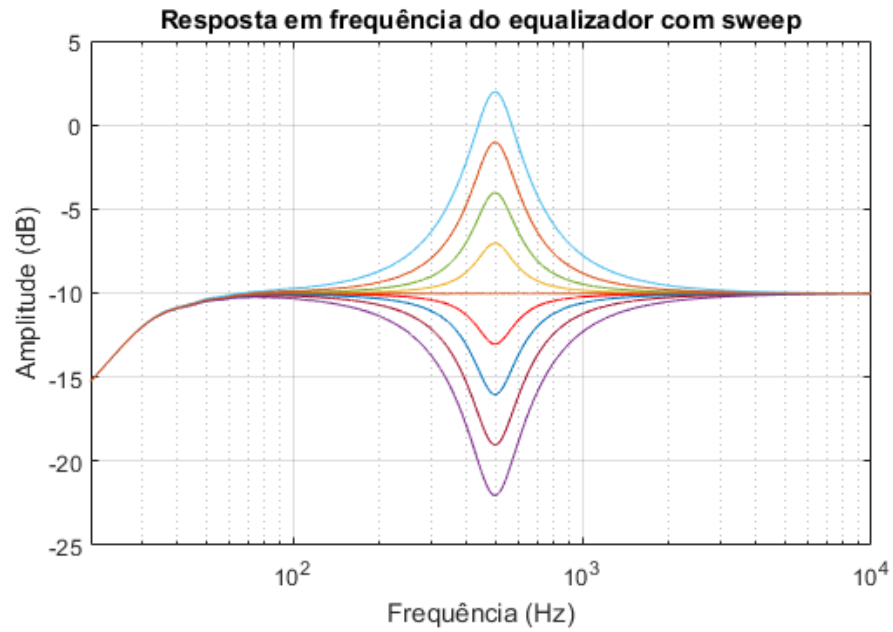
As Figuras 16, 18 e 20, mostram a resposta em frequência para diferentes valores dos parâmetros de ganho, fator de qualidade e frequência central. As Figuras 17, 19 e 21 mostram a FFT do sinal de saída do sistema para cada conjunto de parâmetros do equalizador.

Figura 16 – Variação do ganho para $f_c = 500$ Hz e $Q = 3$, sendo $G = 12, -12, 9, -9, 6, -6, 3$ e -3 dB.



Fonte: AUTOR (2020)

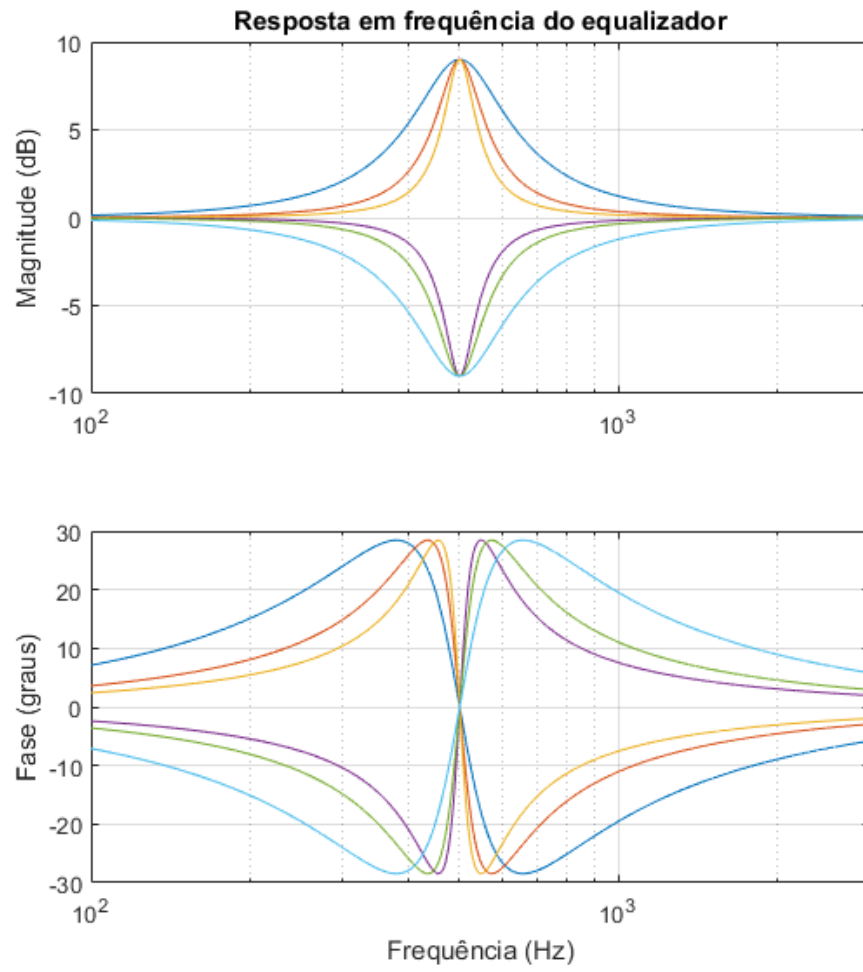
Figura 17 – FFT do sinal de saída do sistema para variações do ganho entre -12 e +12 dB, com f_c igual a 500 Hz.



Fonte: AUTOR (2020)

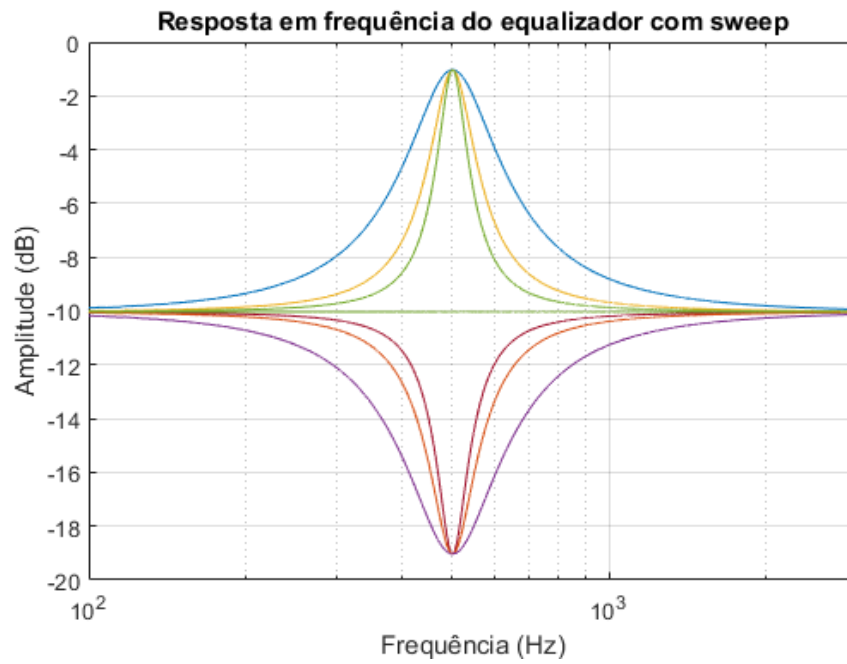
Na largura de banda, quanto maior o valor de Q , mais seletiva é a faixa, tanto de frequência como de fase, como pode-se notar nas Figuras 18 e 19.

Figura 18 – Variação da largura de banda para $f_c = 500$ Hz e $G = 9$ e -9 dB, com $Q = 3, 6$ e 9 .



Fonte: AUTOR (2020)

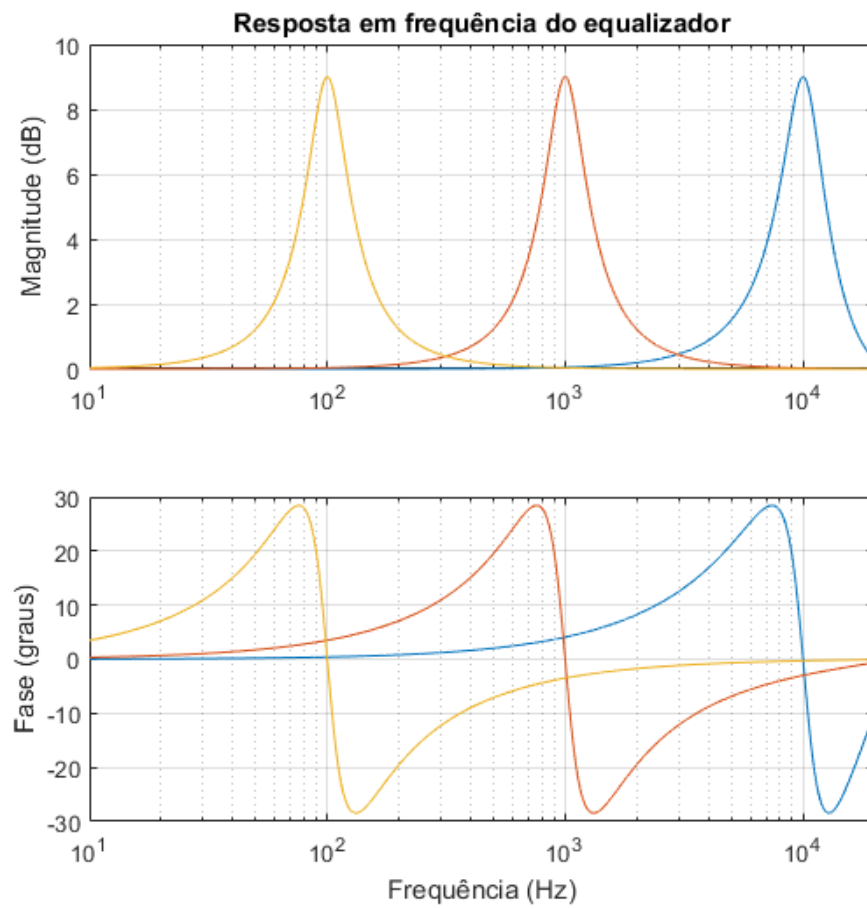
Figura 19 – Resposta da saída do sistema para $Q = 3, 6$ e 9 , com $G = 9$ e -9 dB e $f_c = 500$ Hz.



Fonte: AUTOR (2020)

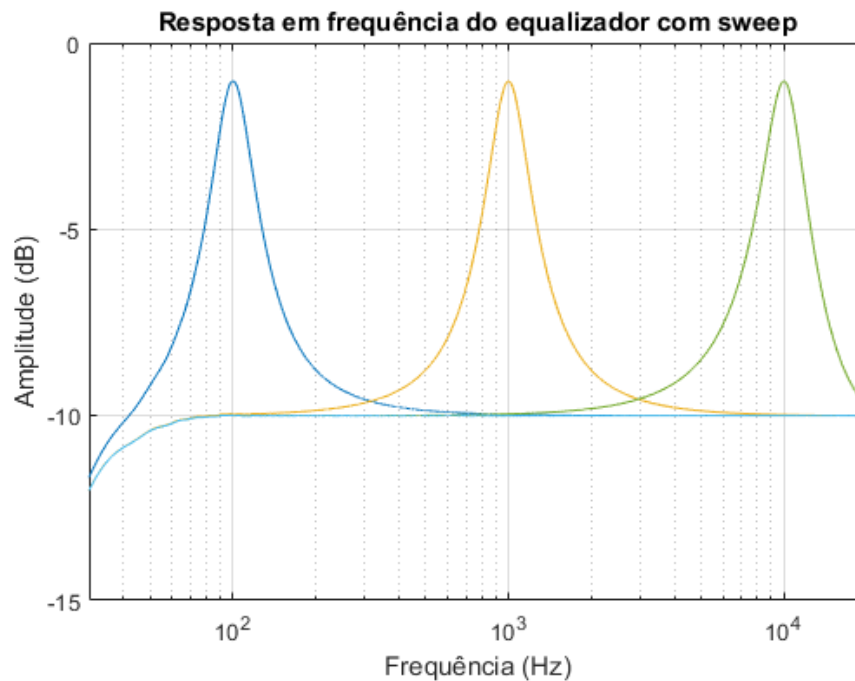
Percebe-se que na variação da largura de banda o ganho permanece o mesmo. Nas Figuras 20 e 21 são mostradas diferentes frequências para o equalizador.

Figura 20 – Variação da frequência para $Q = 3$ e $G = 9$ dB, com $f_c = 100, 1000$ e 10000 Hz.



Fonte: AUTOR (2020)

Figura 21 – Saída do sistema para $f_c = 100, 1000$ e 10000 Hz, $Q = 3$ e $G = 9$ dB.



Fonte: AUTOR (2020)

Percebe-se a consistências nas respostas obtidas, tanto nas variações de parâmetros, como entre a resposta em frequência e a saída do sistema com sinal real.

3.2.2.2 Testes para o crossover

Serão apresentados, nesta seção, os resultados para o *crossover*. Inicialmente será feita a análise para a topologia *Butterworth* de segunda ordem, em seguida para *Butterworth* de segunda ordem com polarização invertida, e por fim, a topologia de quarta ordem *Linkwitz-Riley*.

Para o *crossover*, os canais devem se cruzar em -6 dB, ou seja, este deve ser o ganho na frequência de corte para que a soma dos dois resulte em uma resposta plana. No caso dos filtros de segunda ordem, para estes apresentarem -6 dB em f_0 , o valor de Q deve ser 0,5, pois aumentará a faixa de atenuação diminuindo o ganho na frequência de corte (500 Hz, nestes testes).

A Tabela 2 apresenta os coeficientes calculados para o passa-baixas e passa-altas com os parâmetros estabelecidos, já normalizados por a_0 :

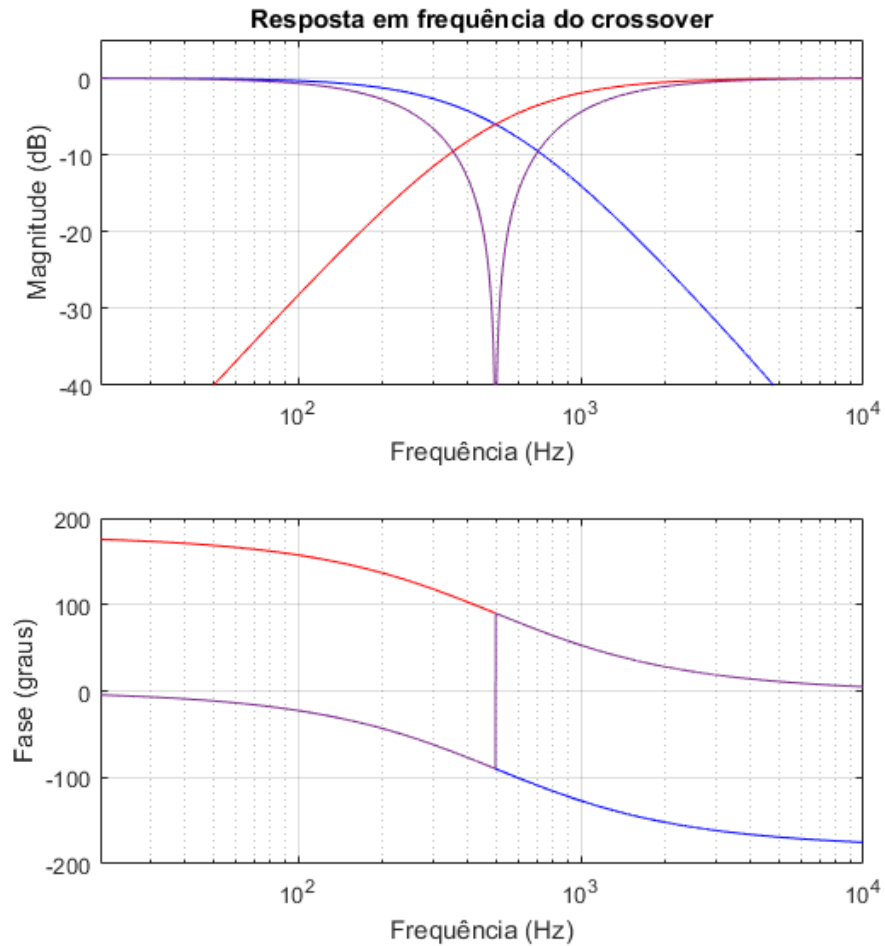
Tabela 2 – Coeficientes para o *crossover* (de segunda ordem) sem inversão de polaridade com $f_0 = 500$ Hz e $Q = 0,5$, onde PB é o passa-baixas e PA, o passa-altas.

Coeficiente	Valor para o PB	Valor para o PA
a_0	1	1
a_1	-1.8732	-1.8732
a_2	0.8772	0.8772
b_0	0.0010	0.9376
b_1	0.0020	-1.8753
b_2	0.0010	0.9376

Fonte: AUTOR (2020)

As respostas em frequência do passa-baixas, passa-altas e da soma dos dos filtros é apresentada na Figura 22. A resposta do sistema para a aplicação no sinal *sweep* é mostrada na Figura 23. A saída total do sistema é a soma dos canais (soma direta dos sinais).

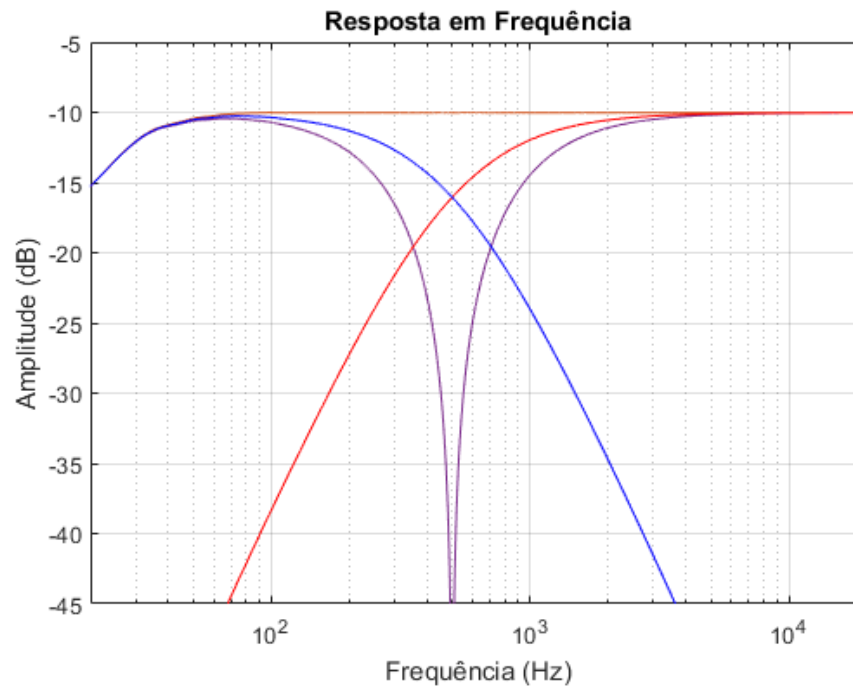
Figura 22 – Magnitude e fase das respostas em frequência para *crossovers* de segunda ordem.
A Figura mostra o passa-baixas (em azul), o passa-altas (em vermelho), suas respectivas fases, e a soma das duas (em roxo) que representa a saída total do *crossover*.



Fonte: AUTOR (2020)

Percebe-se a atenuação de -6 dB em 500 Hz e a defasagem de 180° entre o passa-baixas e o passa-altas, e também a taxa de atenuação de 40 dB/dec, como era esperado. O mesmo ocorre na resposta da aplicação do filtro no sinal, como é mostrado na Figura 23.

Figura 23 – Magnitude das FFTs dos sinais: de entrada (em laranja), saída (em roxo), passa-baixas (em azul) e passa-altas (em vermelho).



Fonte: AUTOR (2020)

Nas Figuras 22 e 23, nota-se o corte proeminente em 500 Hz no sinal de saída devido à defasagem de 180° . No sistema real este cancelamento de frequências ocorre nas ondas no meio acústico, mas para a análise foi feita a soma dos sinais para representar o fenômeno no próprio sinal de saída e também na resposta em frequência. Os cancelamentos de frequências começam a ocorrer assim que as bandas começam a se sobrepor, chegando a ser praticamente total na frequência de corte.

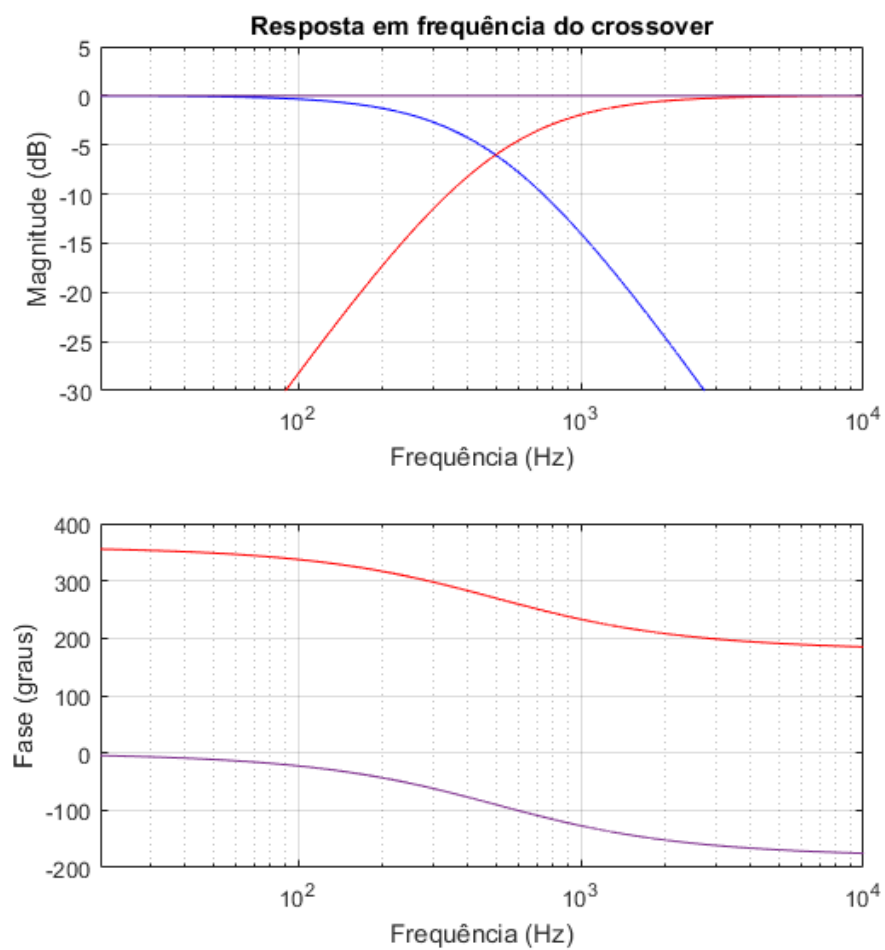
Uma solução para este problema é a inversão de polaridade de um dos canais, uma vez que a defasagem é de 180° . No caso, foi feita a inversão do sinal do canal do passa-altas diretamente pelos coeficientes. Sendo os coeficientes b_0 , b_1 e b_2 da função de transferência os responsáveis pelo numerador, trocando o sinal destes coeficientes no passa-altas, consegue-se a inversão da polaridade. A Tabela 3 apresenta os coeficientes com a inversão de polaridade, que são os mesmos da tabela anterior, com a troca do sinal nos coeficientes b_0 , b_1 e b_2 do passa-altas.

Tabela 3 – Coeficientes para o crossover com inversão de polaridade, com $f_0 = 500$ Hz e $Q = 0,5$, onde PB é o passa-baixas e PA, o passa-altas.

Coeficiente	Valor para o PB	Valor para o PA
a_0	1	1
a_1	-1.8732	-1.8732
a_2	0.8772	0.8772
b_0	0.0010	-0.9376
b_1	0.0020	1.8753
b_2	0.0010	-0.9376

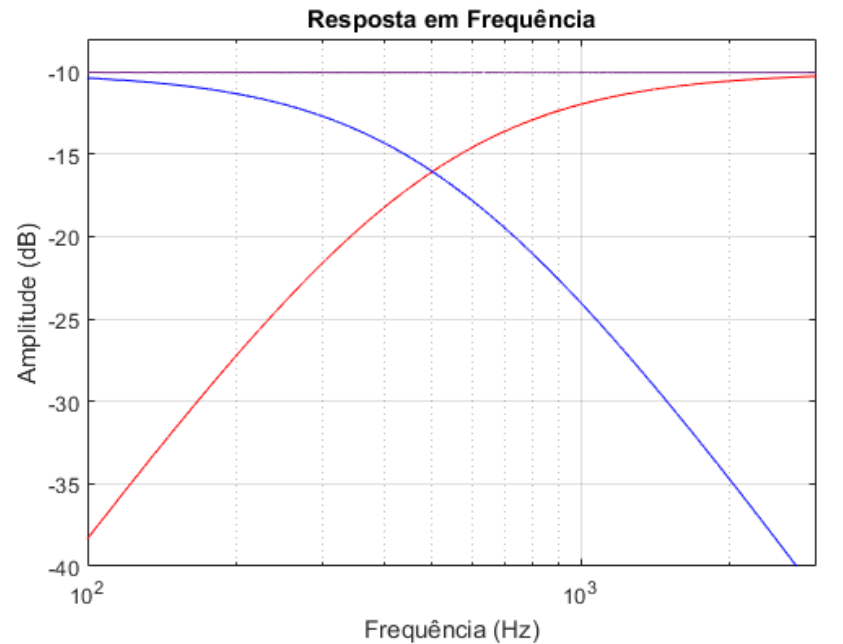
Fonte: AUTOR (2020)

Figura 24 – Magnitude e fase da resposta em frequência do crossover de segunda ordem com a inversão da polaridade. A Figura mostra o passa-baixas (em azul), o passa-altas (em vermelho), suas respectivas fases, e a soma das duas (em roxo) que representa a saída total do crossover, com sua fase sobreposta ao passa-baixas.



Fonte: AUTOR (2020)

Figura 25 – Magnitudes das FFTs de: sinal de entrada sobreposto ao sinal de saída (roxo), passa-baixas (em azul) e passa-altas (em vermelho).

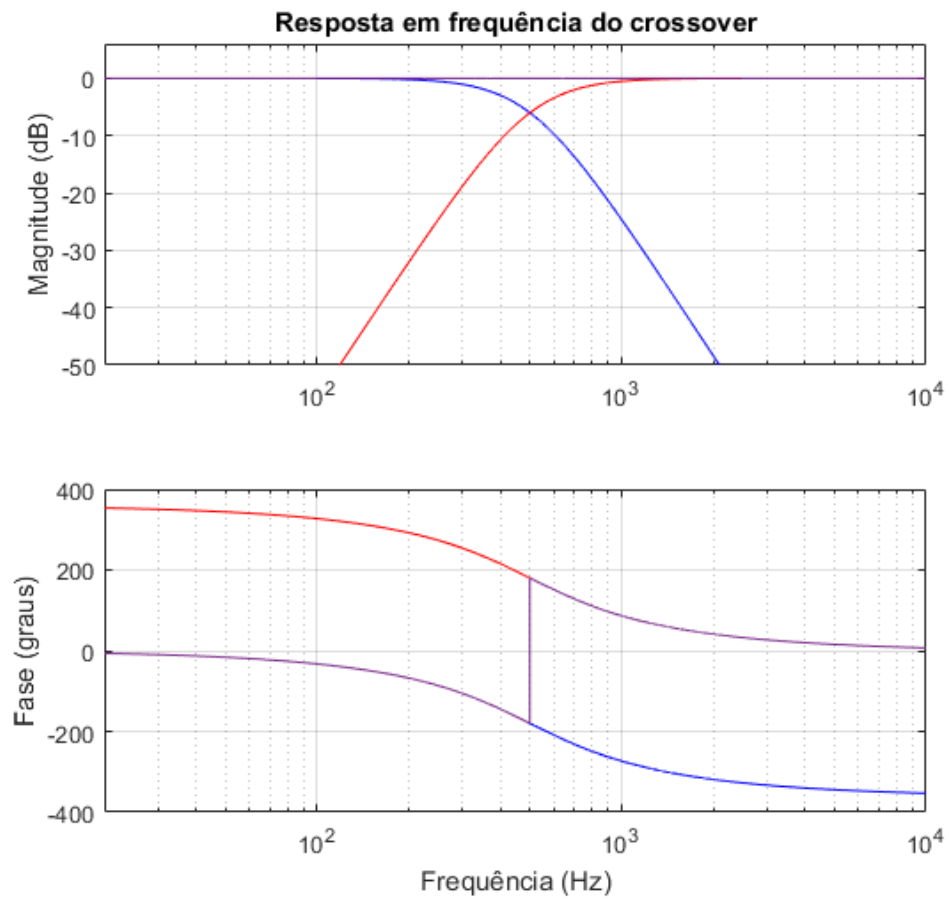


Fonte: AUTOR (2020)

É notório que não há cancelamentos neste caso. Todos os demais parâmetros se mantêm, como a frequência de corte e o seu ganho de -6 dB e a taxa de atenuação de 40 dB/dec.

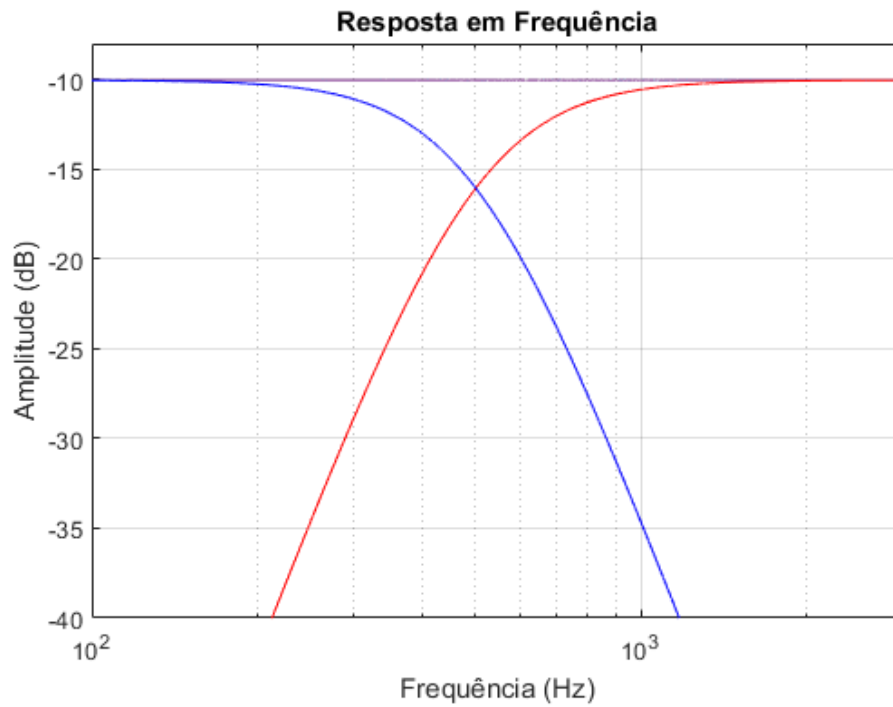
Para o caso do *crossover Linkwitz-Riley* de quarta ordem, o ganho de -6 dB ocorre para o valor de $Q = 0,707$, e a defasagem esperada é de 360° . Para conseguir esta ordem nos filtros é feita a aplicação dos filtros de segunda ordem duas vezes em sequência. Logo, pode-se utilizar os coeficientes da Tabela 1 ou da Tabela 2. As Figuras 26 e 27 mostram os resultados.

Figura 26 – Magnitude e fase da resposta em frequência para o *Linkwitz-Riley*. A Figura mostra o passa-baixas (em azul), o passa-altas (em vermelho), suas respectivas fases, e a soma das duas (em roxo) que representa a saída total do crossover.



Fonte: AUTOR (2020)

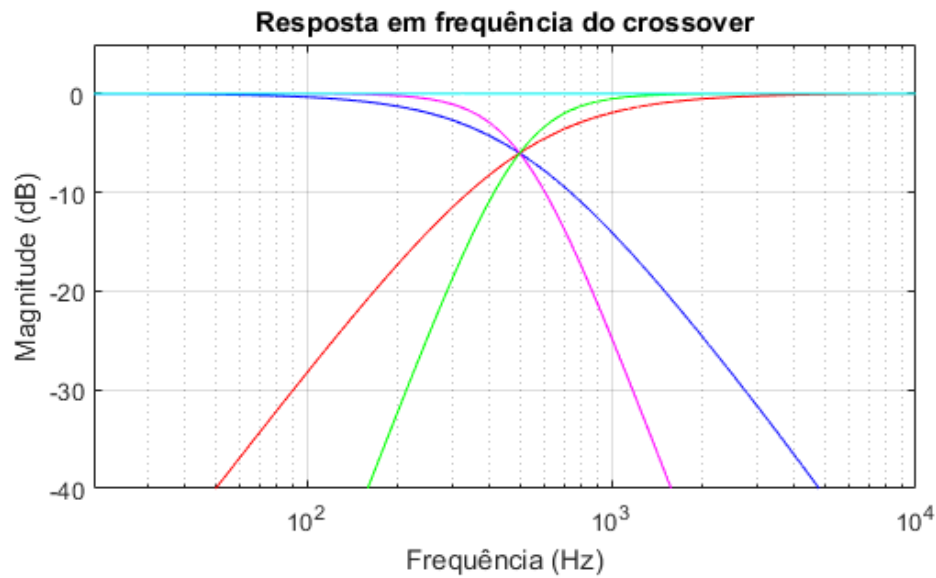
Figura 27 – Magnitudes das FFTs de: sinal de entrada sobreposto ao sinal de saída (em roxo), passa-baixas (em azul) e passa-altas (em vermelho).



Fonte: AUTOR (2020)

Observam-se as mesmas características do *crossover* de segunda ordem, porém, com o de quarta ordem, a taxa de atenuação passa a ser de 80 dB/dec. A Figura 28 apresenta uma comparação entre o *Butterworth* (sem inversão) e *Linkwitz-Riley*.

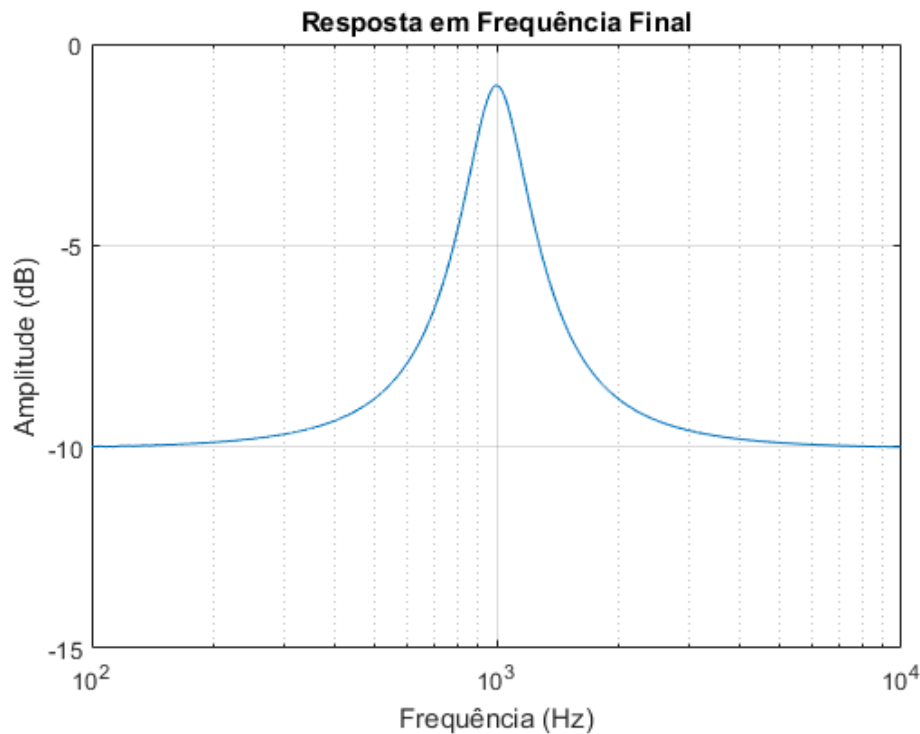
Figura 28 – Comparação entre as respostas em frequência de segunda ordem sem inversão de polaridade (azul e vermelho) e quarta ordem (roxo e verde), em azul claro as saídas sobrepostas.



Fonte: AUTOR (2020)

A seguir será mostrada a saída do sistema com cascadeamento do equalizador em 1kHz, com $G = 9$ dB e $Q = 3$, e o *crossover Linkwitz-Riley* em 500 Hz. Percebe-se a resposta plana do *crossover* e a ação do equalizador em 1 kHz.

Figura 29 – Saída do sistema com equalizador e *crossover*.



Fonte: AUTOR (2020)

Como o MATLAB não realiza processamento em tempo real, o sinal de áudio é processado como um único *buffer*. Para realizar a variação dos parâmetros durante a execução do áudio, foi feita uma rotina que separa o vetor total do sinal de áudio em vetores menores, criando os *buffers*; com isso, pode-se obter o comportamento similar ao que será implementado em *crossovers* de tempo real. Os períodos de tempo utilizados para cada variação de parâmetro foram contabilizados pelos períodos dos próprios *buffers*, ou seja, cada *buffer* tem um tempo aproximado de 200 ms para 48 kHz de frequência de amostragem, e 9600 amostras. Então, 1 segundo era obtido pela contagem de 5 *buffers*. Isto foi feito apenas para se ter uma estimativa se haveria ruídos perceptíveis no áudio, não havia a necessidade utilizar alguma outra função do MATLAB mais complexa para os intervalos de tempo.

Foram testadas as variações de todos os parâmetros, mas principalmente a variação das frequências, que é o mais comum na prática. Estas variações foram automatizadas a cada 200 ms, 1 s e 2 s aproximadamente, tendo como base o tempo do *buffer*. Os passos de frequência utilizados foram 1 Hz, 10 Hz, 50 Hz, 100 Hz e

200 Hz, ou seja, em um dado intervalo de tempo, os coeficientes foram recalculados com o passo do parâmetro, incrementando e decrementando, fazendo a varredura do espectro. Nos parâmetros de ganho e fator de qualidade utilizou-se o passo unitário. Não foi perceptível nenhum ruído no áudio proveniente das variações.

3.3 Implementação em *hardware*

Após validar a aplicação das equações para o cálculo dos coeficientes no MATLAB, foi feita a implementação no kit de desenvolvimento disponibilizado pelo Departamento Acadêmico de Eletrônica (DAELN) do IFSC. Para escrever e editar o código, compilar e depurar o sistema foi utilizada a IDE (do inglês - *Integrated Development Environment*) *Eclipse*. Para gerar os sinais e realizar as análises do áudio, foram utilizados os *softwares* *OcenAudio* e *Ableton*.

A metodologia de implementação no *hardware* foi a mesma realizada com o MATLAB, como mencionado anteriormente. Cada etapa foi feita no *hardware* assim que foi validada no MATLAB, o que foi considerado como a realimentação do processo de desenvolvimento, no início da Seção 3.2 (Figura 12). Após implementados os algoritmos para o processamento dos filtros, foram implementados os controles e a navegação.

3.3.1 Ambiente e kit de desenvolvimento

Serão apresentados nesta seção os recursos utilizados para a implementação do sistema, tanto a plataforma de *hardware* STM32F407 integrada com o *Wolfson Pi*, a biblioteca utilizada para a implementação de DSP, a placa e os *softwares* de áudio.

3.3.1.1 O Kit STM32F4 Discovery e Wolfson Pi

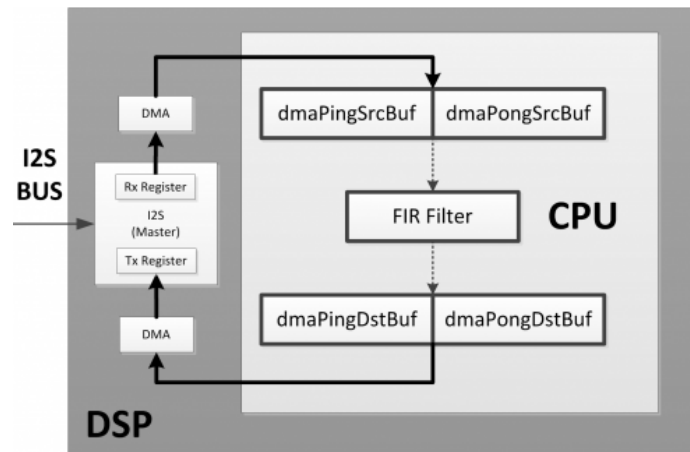
O kit disponibilizado pelo DAELN consiste na integração de uma plataforma que realiza a interface AD/DA para a entrada e saída do áudio estéreo, o *Wolfson Pi*. Esta placa já possui os conectores dos cabos e os CODECS que realizam a conver-

são AD/DA, e faz a comunicação via protocolo I2S com a placa de processamento, a STM32F4 *Discovery*. O *Wolfson Pi* é utilizado apenas como interface para o meio externo, sendo o microcontrolador STM32F407 de fato o núcleo do projeto.

O processador da placa do *Discovery* é um *ARM Cortex M4*, de 32 bits, que possui bom desempenho para o processamento em ponto flutuante e instruções otimizadas para processamento digital de sinais, como SIMD (do inglês - *Single Instruction Multiple Data*) (STMICROELECTRONICS, 2017).

Para o processamento de áudio, é comum utilizar o que se chama de *buffer ping-pong*, que consiste de um *buffer* dividido em duas partes iguais, em que ocorre a alternância entre o bloco que faz a aquisição das amostras na memória, e o bloco que processa e disponibiliza as novas amostras para a saída. O bloco de processamento determina o tempo entre a alternância dos blocos. Enquanto o processamento é feito em todas as amostras de um bloco, o outro já é preenchido por novas. Quando o segundo bloco está completo, ocorre então a inversão dos blocos; o processamento passa a ser feito onde há novas amostras carregadas, e as que foram processadas são disponibilizadas na saída. Portanto, o processamento ocorre sempre do início do *buffer ping-pong* até a metade, e depois da metade até o final, e assim sucessivamente. O método do acesso à memória que o *buffer ping-pong* utiliza é o DMA (do inglês - *Direct Memory Access*), sendo que as amostras recebidas e que serão transmitidas pelo I2S são armazenadas em regiões da memória *flash*. O DMA realiza a transferência otimizada para a SRAM, onde os dados serão manipulados (NEVES, 2016). A Figura 30 mostra o diagrama de blocos de um *buffer ping-pong* com DMA.

Figura 30 – Exemplo de Buffer ping-pong com acesso à memória por DMA.



Fonte: (NEVES, 2016)

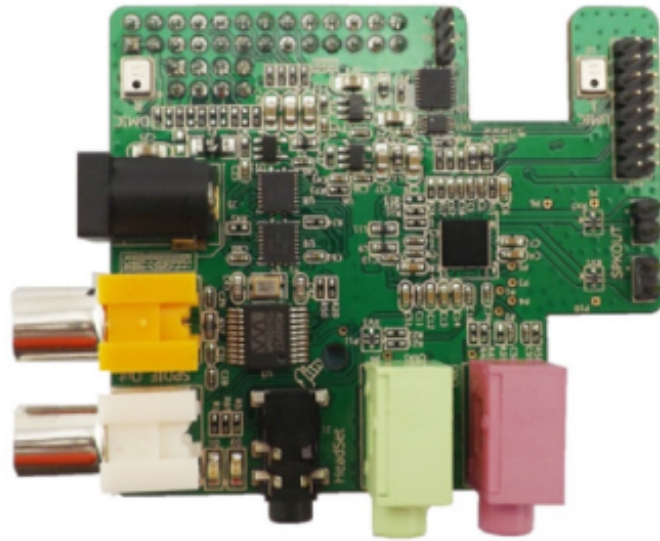
O STM32F407 também possui um ST-LINK/V2, que é uma ferramenta embarcada para realizar a depuração do sistema. Utilizando o driver pode-se verificar a execução cada linha de código, os valores das variáveis, etc., dentro do ambiente *Eclipse*. As Figuras 31, 32 e 33 mostram as placas utilizadas.

Figura 31 – Foto da vista superior do kit de desenvolvimento STM32F4 Discovery.



Fonte: (STMICROELECTRONICS, 2017)

Figura 32 – Foto da vista superior da placa *Wolfson Pi*



Fonte: (WOLFSON MICROELECTRONICS, 2014)

Figura 33 – Kit de desenvolvimento STM32F4 *Discovery Kit* integrado com *Wolfson Pi*



Fonte: AUTOR (2020)

3.3.1.2 Placa de Áudio M-Audio Fast Track Pro

Para conectar as saídas do kit ao computador, foi utilizada uma interface de áudio com 2 canais de entrada, a *M-Audio Fast Track Pro*, que conecta-se ao computador via USB. Ela possui a opção de somar os canais de entrada, o que foi útil para a análise do *crossover*, e também pode-se utilizá-la como um pré-amplificador independente, sem necessitar do computador. A Figura 34 apresenta a placa utilizada.

Figura 34 – Interface de áudio *Fast Track pro*



Fonte: (TECLACENTER, 2015)

3.3.1.3 Biblioteca CMSIS-DSP

De maneira geral, os processadores para sistemas embarcados apresentam restrições de memória e processamento. Sendo assim, muitos fabricantes disponibilizam bibliotecas que otimizam e também fazem algumas abstrações de *hardware*, tornando a implementação menos complicada e mais funcional, com compatibilidade para as diferentes linhas de plataformas que o fabricante possui, que no caso é a ARM.

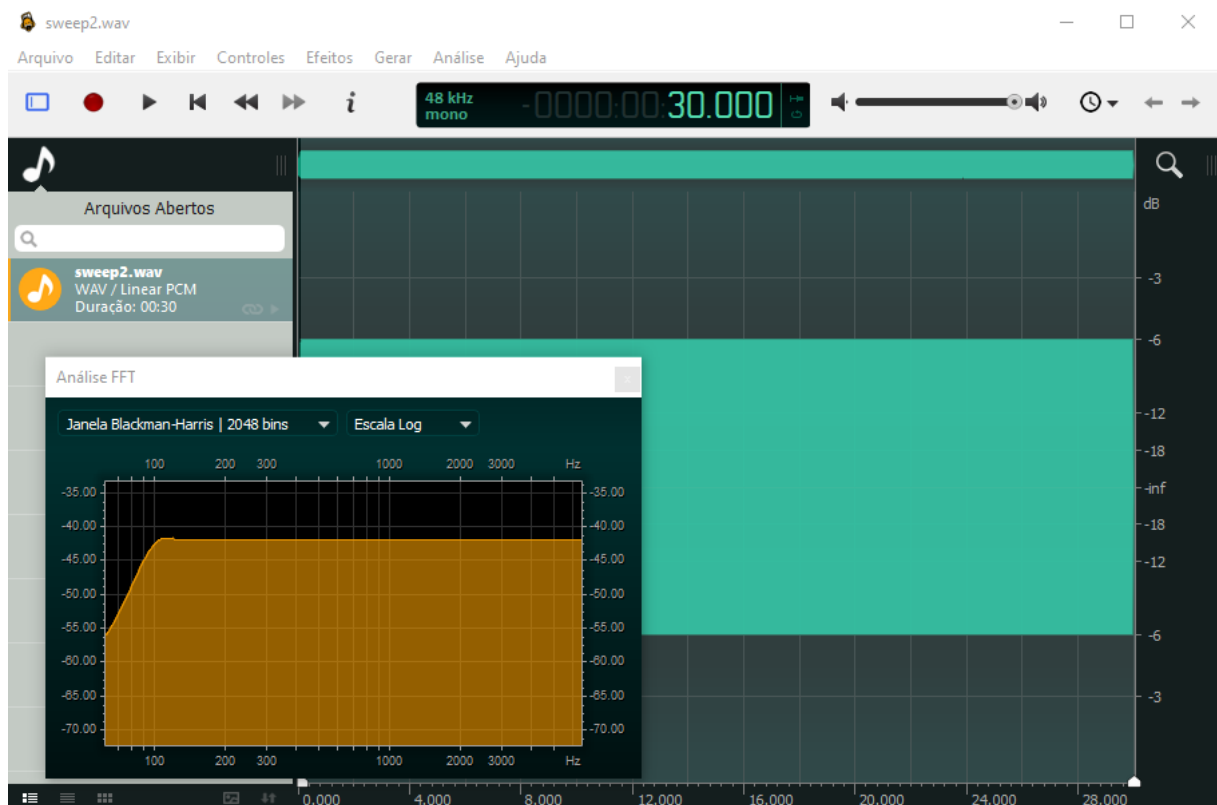
A biblioteca CMSIS-DSP (do inglês - *Cortex Microcontroller Software Interface Standard*) disponibiliza funções para o processamento eficiente de funções, desde as trigonométricas, utilizadas para o cálculo dos coeficientes dos filtros, até específicas para DSP (KEIL CMSIS-DSP, 2010).

As funções de processamento digital de sinais utilizadas neste trabalho são para a implementação de estruturas no formato de biquadradas (filtros IIR de segunda ordem), sendo necessário algumas estruturas de variáveis para sua inicialização, que conterão os dados, como os *buffers* de I/O e de coeficientes. São utilizados cinco coeficientes por estágio, sendo todos normalizados por a_0 . Como os coeficientes são armazenados em um vetor na estrutura que a função utiliza, para o filtro de quarta ordem apenas duplica-se o estágio de segunda ordem, ficando assim com um vetor de dez coeficientes.

3.3.1.4 Softwares de áudio

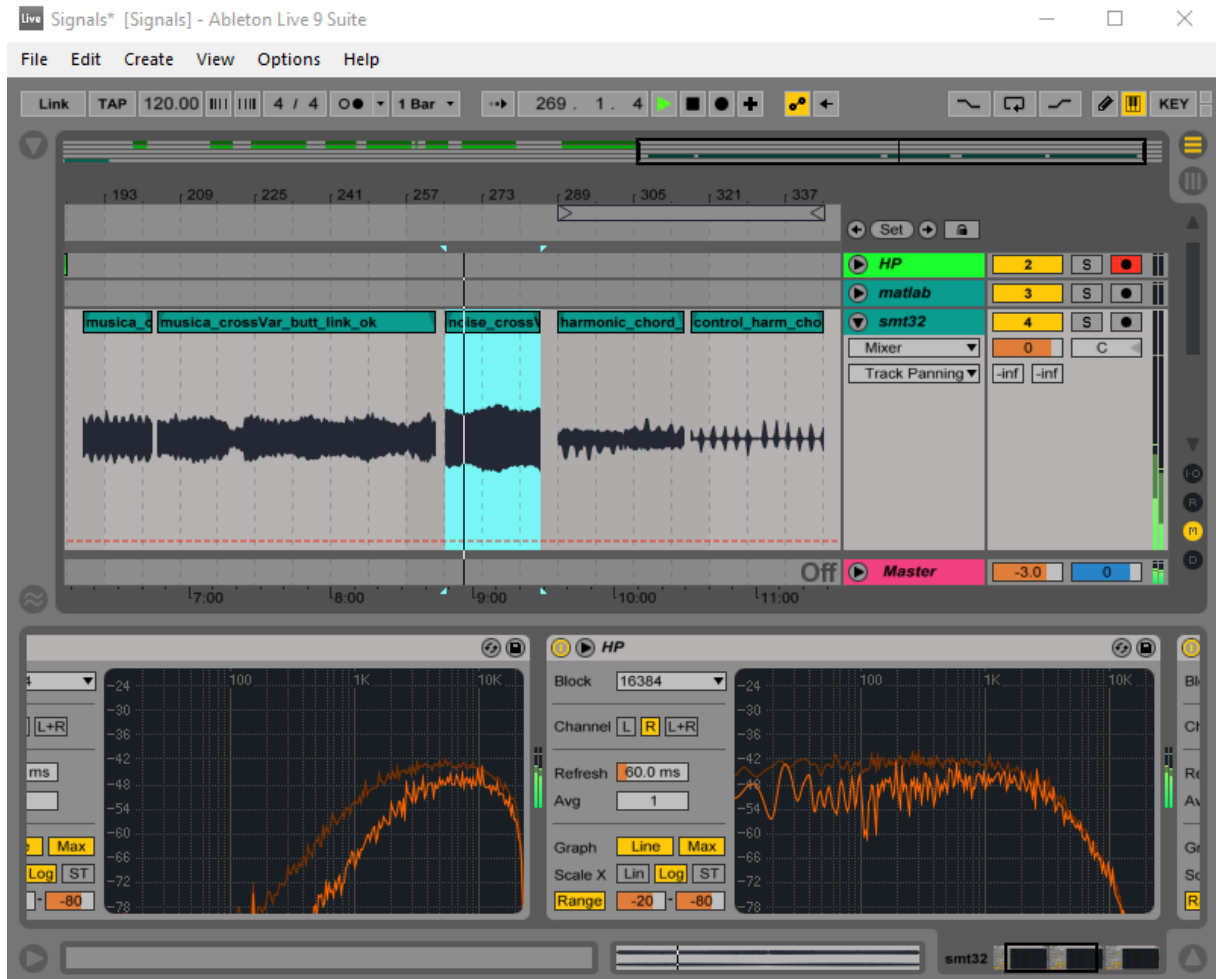
Para gerar os sinais de testes, visualizar as formas de onda e FFT, separar os canais e analisar os resultados, foram utilizados os *softwares OcenAudio*, que é um *freeware* (*software* gratuito) desenvolvido no Laboratório de Circuitos e Processamento de Sinais (LINSE/UFSC). Por haver a familiaridade com o *Ableton Live*, que é utilizado profissionalmente para produção musical, este também foi utilizado por oferecer praticidade na gravação, manipulação dos arquivos nos canais e ferramentas que poderiam ser interessantes para alguma análise específica. Nele foi utilizado o analisador de espectro por canais, ferramenta para transformar o sinal estéreo em mono, opções de “mutar” e “solar” canais. As Figuras 35 e 36 mostram as telas dos softwares utilizados.

Figura 35 – Software OcenAudio.



Fonte: AUTOR (2020)

Figura 36 – Software Ableton Live.

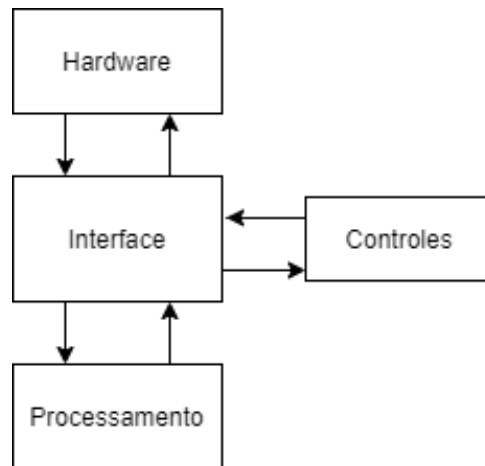


Fonte: AUTOR (2020)

3.3.2 Arquitetura do *software*

A implementação do sistema foi feita utilizando linguagem C e conceitos de arquitetura de *software*, que consiste na separação das camadas em níveis de abstração, sendo elas: o *hardware*, processamento, controles e interface. Buscou-se fazer o desacoplamento, tanto em arquivos distintos como em funções, onde cada parte do código fica isolada, o que facilita a implementação, a depuração e também algum reaproveitamento de código futuro. A Figura 37 mostra os blocos do sistema.

Figura 37 – Blocos da arquitetura de *software*.



Fonte: AUTOR (2020)

A camada do *hardware* está associada ao arquivo *main.c*, onde são feitas as inicializações, configurações, tratadas as interrupções e onde os *buffers* acessam a memória pelo DMA.

A camada da interface está no arquivo *interface.c*, que conecta todo o sistema; é por meio dela que os dados dos controles de parâmetros do usuário são adquiridos, a partir de interrupções, e, com isso, é atualizado o *display*. Ela faz a conexão com a camada de *hardware* e executa o processamento. Os filtros foram criados em arquivos, que contêm as estruturas para os dados de parâmetros e as suas funções.

Foram feitas otimizações para que o código e o processamento sejam mais eficientes, como a utilização de vetores de estruturas de dados e também um vetor para os *buffers* de entrada e saída. Isto trouxe flexibilidade nas chamadas de funções, em que, para um respectivo filtro, necessita-se passar apenas o índice do vetor para acessar seus parâmetros por ponteiros. A intenção ao utilizar esses vetores foi também utilizar o recurso de ponteiros.

3.3.3 Implementação dos algoritmos

O fluxograma da Figura 13 (Seção 3.2.1), mostra como o sistema foi implementado. A plataforma de *hardware* possui seu *buffer* de entrada com dois canais, sendo utilizado apenas o esquerdo para a implementação, pois trata-se de um sistema

monofônico. Logo, considera-se que os canais de entrada são iguais. A partir do *buffer* de entrada, é feita a aplicação da função *arm_biquad_cascade_df1_f32*, da CMSIS, primeiramente para o equalizador, onde sua saída é duplicada e alimenta os filtros do *crossover*. É feita a lógica de acordo com as opções de ligar e desligar os filtros, direcionando o sinal de acordo com a seleção do menu, como já mostrado no fluxograma da Figura 13.

As opções de navegação são tratadas nas funções que são chamadas pelas interrupções dos respectivos botões. Estas funções realizam a lógica e a alteração dos parâmetros.

Os coeficientes são calculados com valores iniciais pré-definidos, antes do início do *loop* principal, e são atualizados sempre que ocorre uma variação de parâmetros. Os cálculos dos coeficientes foram otimizados, pois apresentam algumas funções trigonométricas, que demandam certo tempo de processamento. Para melhorar o desempenho, foram utilizadas as funções da biblioteca CMSIS, e as repetições de cálculos nas equações foram colocadas em variáveis. Assim, é feito apenas um cálculo de cosseno e seno, e as repetições destes, ou uso de tangentes são feitos pelo cálculo já armazenado nas variáveis. Para coeficientes com o mesmo valor, são apenas copiados os já existentes. Todos estes valores já são armazenados na estrutura do filtro, que é passada via ponteiro. A maior parte das operações aritméticas foi feita com pontos flutuantes.

3.3.4 Implementação da interface com o usuário

O controle do sistema pode ser feito pelo usuário e é realizado por botões *push-button*, sendo dois para incremento e decremento linear dos valores, e um para a seleção. As ações destes botões são feitas por meio de interrupção e o tempo de *debounce* (tempo para evitar que a trepidação no contato gere acionamentos indesejados) teve que ser em torno de 200 ms para que ficasse estável.

Foi utilizado um *display oled* SSD1306 128x64 pixels, que possui uma boa resolução e baixo consumo. No entanto, a sua integração gerou alguns problemas ini-

cialmente, devido ao tempo gasto pela comunicação I2C utilizada entre os dispositivos, o que produzia ruídos. Foi utilizada uma biblioteca para o uso do *display*, e, mesmo realizando a atualização da tela apenas no acionamento dos botões, aconteciam *glitches* (ruídos, como sons de *click* no áudio). Foi necessário inspecionar o código da biblioteca, e então realizar a sua utilização de forma diferente, pois não estava sendo utilizada interrupção para receber a resposta do I2C, e cada pacote de dados enviados ao *display* deve retornar uma resposta para que o próximo seja enviado, sendo necessários oito pacotes para a atualização completa da tela. A atualização da tela desta maneira trava o processamento até que se finalizem os oito pacotes.

Primeiramente, foi estudado um meio de utilizar a interrupção, porém, seria necessário realizar o incremento para o envio dos oito pacotes de dados assincronamente. Logo, foi feito o teste realizando este incremento dos pacotes sincronamente, sem interrupção, um por vez, a cada iteração do loop principal, e isto já foi suficiente.

O modo como este problema foi detectado foi pela contagem dos ciclos gastos, que será explicado mais adiante. Para validar o funcionamento, foi feita a verificação por ciclos mas, principalmente, verificado o efeito no áudio e FFT para analisar se harmônicos estranhos apareciam. A Figura 38 mostra a placa da interface.

Figura 38 – Placa para interface com *display oled* SSD1306 128 x 64 pixels e botões *push-button*.



Fonte: AUTOR (2020)

3.4 Testes de implementação

As verificações do desempenho do sistema e também a detecção de problemas foram analisadas a cada novo incremento de funcionalidade. Foram feitas algumas verificações em MATLAB, mas apresenta-se aqui o que foi feito para o *hardware* e as características do sistema real.

As verificações no áudio foram realizadas utilizando os *softwares* de áudio, onde foram analisadas as formas de onda, os espectros de frequência e também o som em si. Buscou-se detectar anomalias e verificar se a resposta apresentava as características esperadas. Foram utilizados vários sinais de entrada para os testes: o sinal *sweep*, ruído branco, acordes com poucos harmônicos, com muitos harmônicos e também músicas. Estes sinais foram aplicados na entrada do kit e a sua saída foi conectada na placa M-Audio em dois canais separados, de modo que pudessem ser gravados no computador e analisados.

Para a verificação do desempenho do processamento foi feita a depuração do código, analisando valores de parâmetros e marcações em alguns trechos para constatar se estava ocorrendo o esperado; também foi feita a contagem dos ciclos de *clock* de algumas funções e seções do processamento. As funções para salvar os arquivos dos valores dos ciclos e para imprimir os dados no console do depurador produzem distorção no som, devido ao seu tempo de execução. Foi constatado, realizando as medições dos tempos gastos, que quantidades de ciclos acima de 1,5 milhão no *loop* principal começam a produzir ruídos. Por meio da contagem dos ciclos, detectou-se o problema que o *display* apresentava. A Tabela 4 mostra a média de ciclos de alguns casos medidos.

Tabela 4 – Valores médios de ciclos na execução de algumas seções do processamento.

Seção Medida	Número médio de ciclos
Função <i>if</i>	24
Divisão com inteiros	27
Divisão com ponto flutuante	40
Função <i>armbiquad</i>	21152
Cálculo dos coeficientes	2240
Preenchimento do <i>buffer</i> de entrada	31770
Preenchimento do <i>buffer</i> de saída	55440
<i>Display</i> sem otimização	4417495
<i>Display</i> com otimização	660620
Processamento Total Médio Otimizado	812526

Fonte: AUTOR (2020)

Nota-se que o *display* sem otimização ultrapassa o valor constatado que afeta o áudio. Com a otimização, o valor total do processamento fica em torno da metade deste limite.

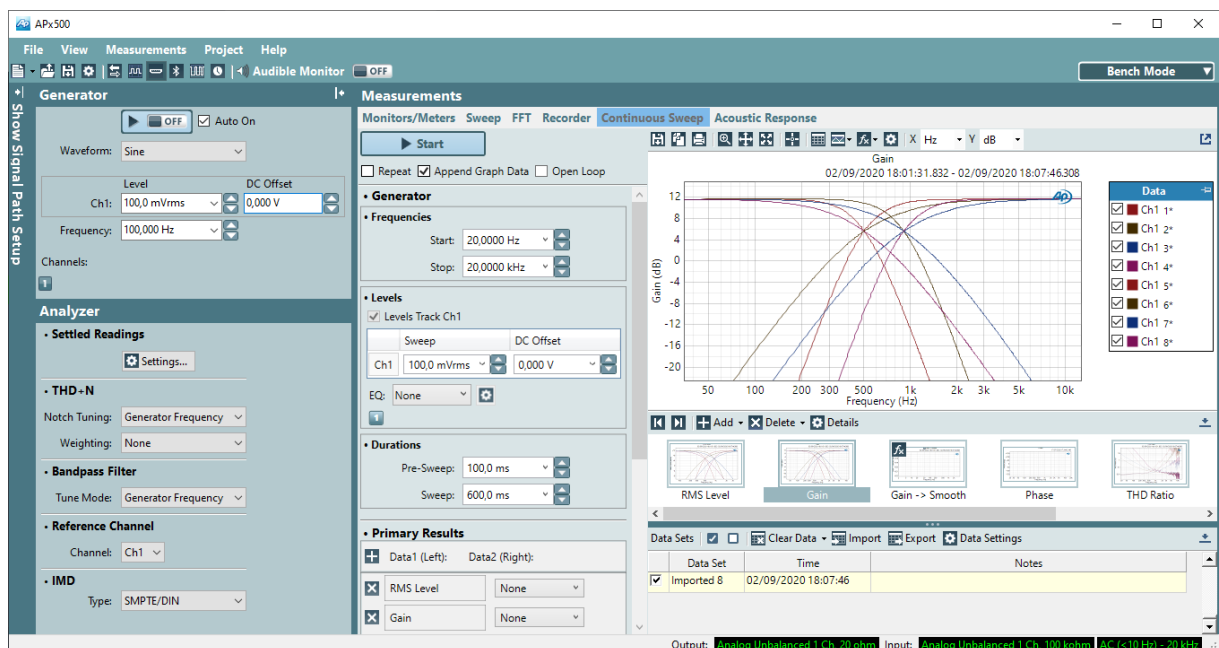
4 APRESENTAÇÃO DOS RESULTADOS

Neste capítulo serão apresentados os resultados medidos no sistema implementado em *hardware*. Os testes foram realizados no laboratório da Bortoni Technology (BORTONI TECHNOLOGY, 2020).

Foi utilizada a plataforma APx da *Audio Precision*, a qual possui sua interface de *hardware*, o APx 525 *Audio Analyzer* e o *software* APx 500. O APx 525 possui as saídas e entradas de sinal que são conectados ao sistema a ser medido, como sua entrada e retorno para análise. No *software*, o sinal que vai para o sistema medido é configurado, e também onde são realizadas as análises e as aquisições que o APx 525 recebe como retorno do sistema a ser medido. Foi utilizado um sinal de *sweep* de 20 a 20 kHz, com amplitude de 100 mVrms, e também um sinal senoidal de 100 Hz.

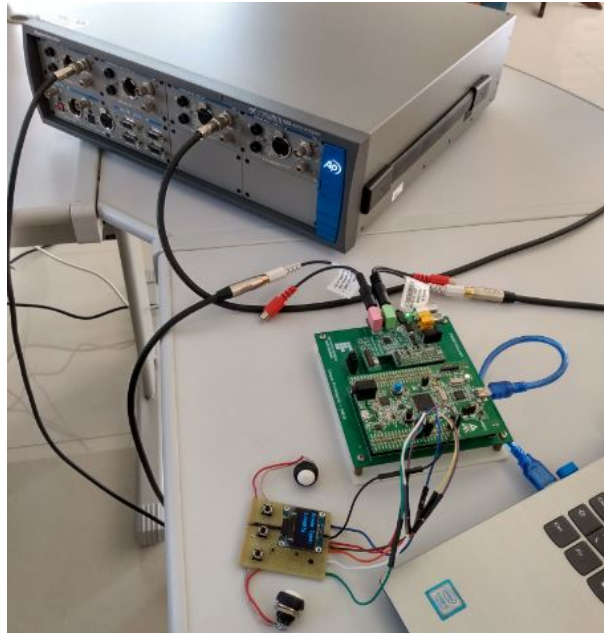
Para a análise da soma dos canais do *crossover* foi utilizada a placa da MAudio, que possibilita somar os canais e deixar a saída em mono, a qual foi capturada pelo sistema da AP (*Audio Precision*). Portanto, a soma dos canais para a análise foi feita no meio elétrico. As Figuras 39 e 40 mostram a tela do *software* e a bancada.

Figura 39 – Visão geral do *software* APx 500 *Audio Precision*.



Fonte: AUTOR (2020)

Figura 40 – Medições em bancada. Sistema desenvolvido conectado ao equipamento APx 525 Audio Analyzer da Audio Precision.



Fonte: AUTOR (2020)

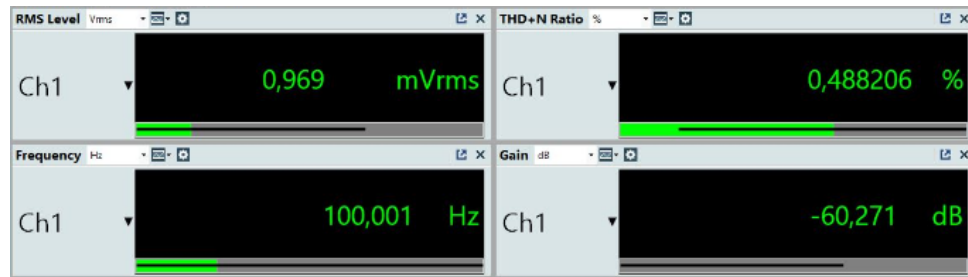
4.1 Figuras de mérito do sistema

4.1.1 Faixa dinâmica e THD

A análise a seguir foi feita para verificar os valores máximos e mínimos de amplitude de entrada e saída do sistema, com distorção harmônica dentro do tolerável. Foi utilizado uma senóide de 100 Hz variando sua amplitude na entrada do sistema, e também a sua saída pelo ajuste do volume. As medições foram feitas com o equalizador desligado e no canal de saída referente ao passa-baixas do *crossover*.

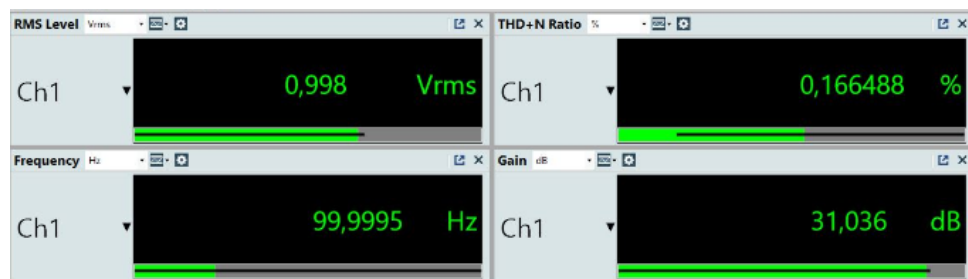
Para a amplitude máxima de entrada foi constatado o valor apresentado como a máxima amplitude para que não ocorra a distorção da onda já na entrada da placa *WolfsonPi*, isso por característica da própria placa. Esta aferição foi feita visualmente pelo software, por isso o valor da THD ficou bem abaixo do limite estipulado. As Figuras 41 e 42 mostram as capturas de tela feitas das medições.

Figura 41 – Captura de tela do software APx 500, mostrando os valores mínimos de amplitude para frequência de 100 Hz e THD menor que 0,5%.



Fonte: AUTOR (2020)

Figura 42 – Captura de tela do software APx 500, mostrando os valores máximos de amplitude para frequência de 100 Hz e THD menor que 0,5%.



Fonte: AUTOR (2020)

Estes valores extremos representam as regiões com maior THD (do inglês - *Total Harmonic Distortion*), que representa as harmônicas que o sistema gera sobre as frequências originais. As análises mostraram uma amplitude mínima de saída de -60,2 dB com THD de 0,488%, e máxima de 31 dB com THD de 0,166%. A média do THD nas demais amplitudes ficou em 0,15%, estando dentro do estipulado para o projeto.

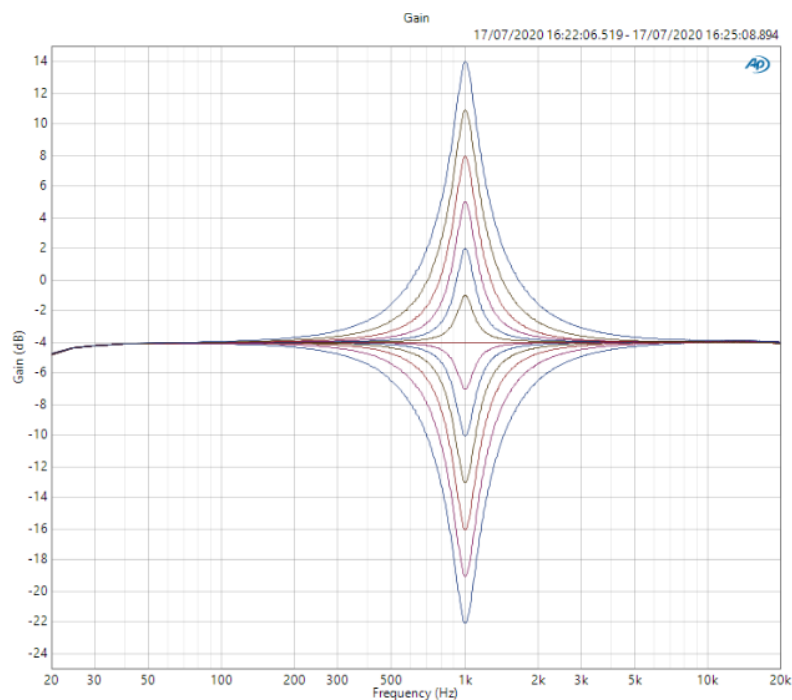
4.1.2 Resposta em frequência

A seguir serão apresentadas as figuras obtidas nas medições. As variações de amplitude ocorreram porque a saída (volume) do sistema foi sendo modificada ao longo das análises. O passo utilizado na interface para variar o volume é de 5 em 5, e vai de 0 a 100, logo não foi possível atingir precisamente a amplitude de 0 dB, ficando em torno de -2 dB ou 2 dB. Quando o *crossover* é desligado, a amplitude é diminuída em 6 dB para que não haja sobreposição dos 2 canais e dobra da amplitude.

4.1.2.1 Resposta do equalizador

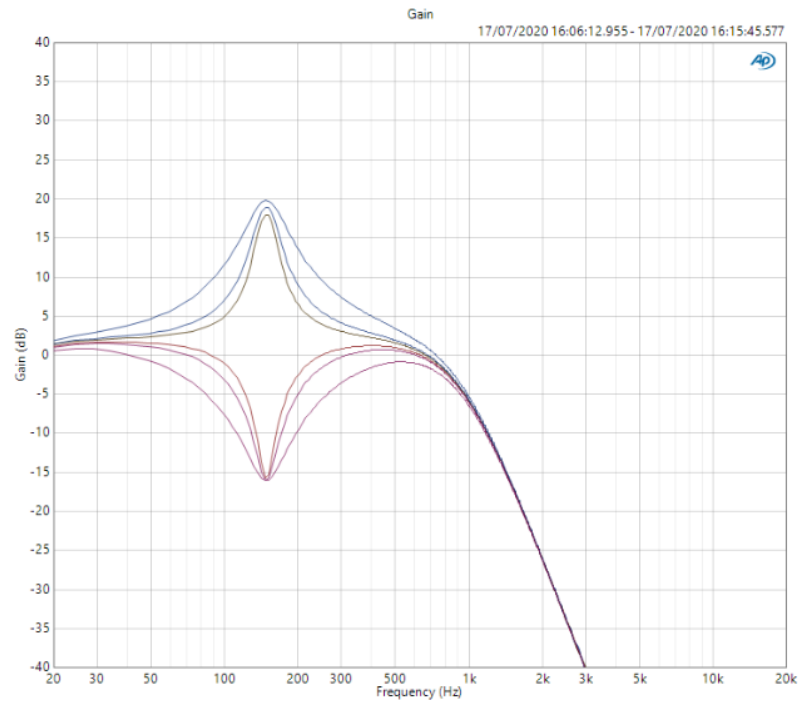
As Figuras 43, 44 e 45 apresentam as respostas para a variação do ganho (G), fator de qualidade (Q) e frequência (f_0). De modo geral, observa-se o funcionamento esperado do equalizador. Nota-se uma variação na amplitude do ganho positivo conforme é feita a variação do parâmetro Q, porém a causa não foi identificada, uma vez que os cálculos são iguais para a amplitude positiva e negativa, e não há parâmetros que possam afetar o ganho nas equações. Os cálculos utilizados foram exatamente os mesmos feitos em MATLAB e que haviam funcionado perfeitamente.

Figura 43 – Resposta da variação do ganho do equalizador em 1 kHz, $Q = 6$ e ganho de ± 3 dB a ± 18 dB como passos de 3 dB. Crossover desligado.



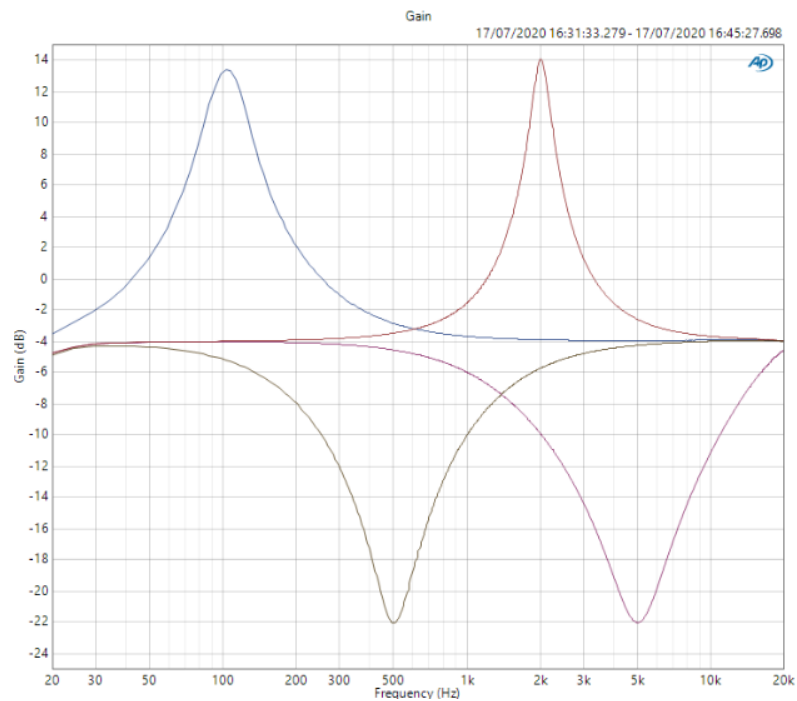
Fonte: AUTOR (2020)

Figura 44 – Resposta do equalizador com variação do Q em 140 Hz, $G = \pm 9$ dB e Q variando de 3 a 9 com passo de 3. Crossover ligado com corte em 900 Hz.



Fonte: AUTOR (2020)

Figura 45 – Resposta do equalizador com variação da frequência em 100 Hz, 500 Hz, 2 kHz e 5 kHz, com $G = \pm 9$ dB e diferentes valores de Q. Crossover desligado.

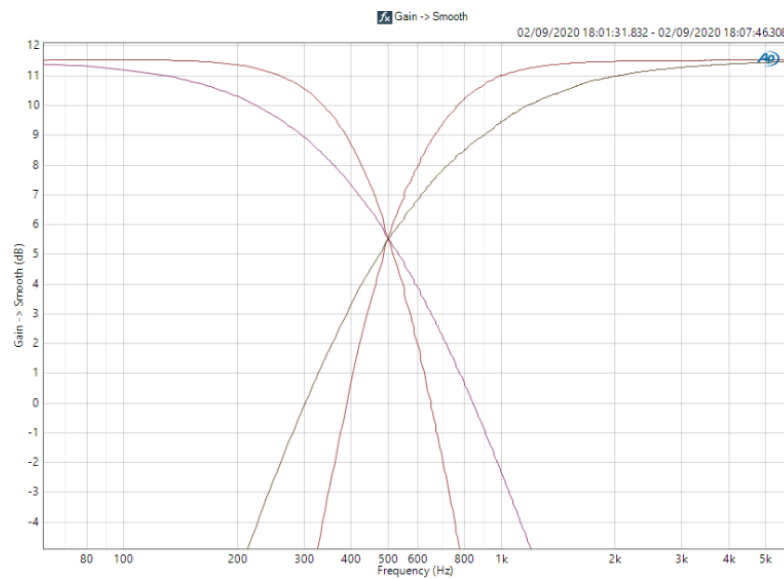


Fonte: AUTOR (2020)

4.1.2.2 Resposta do crossover

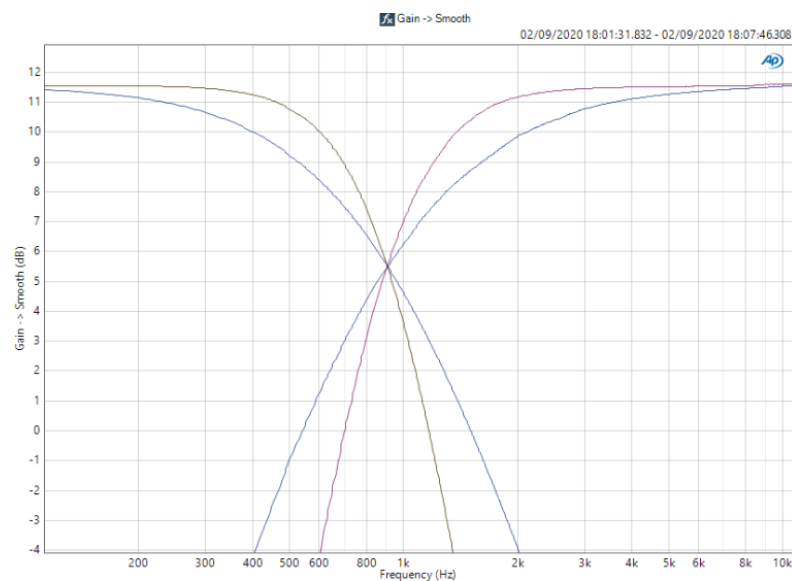
As Figuras 46 e 47 apresentam as respostas para segunda e quarta ordens, em diferentes frequências. A Figura 48 apresenta a saída do sistema para o *crossover* de segunda ordem com e sem inversão de polaridade. Em todos os casos observa-se o funcionamento esperado do *crossover*.

Figura 46 – Saídas do *crossover* em 500 Hz para segunda e quarta ordem.



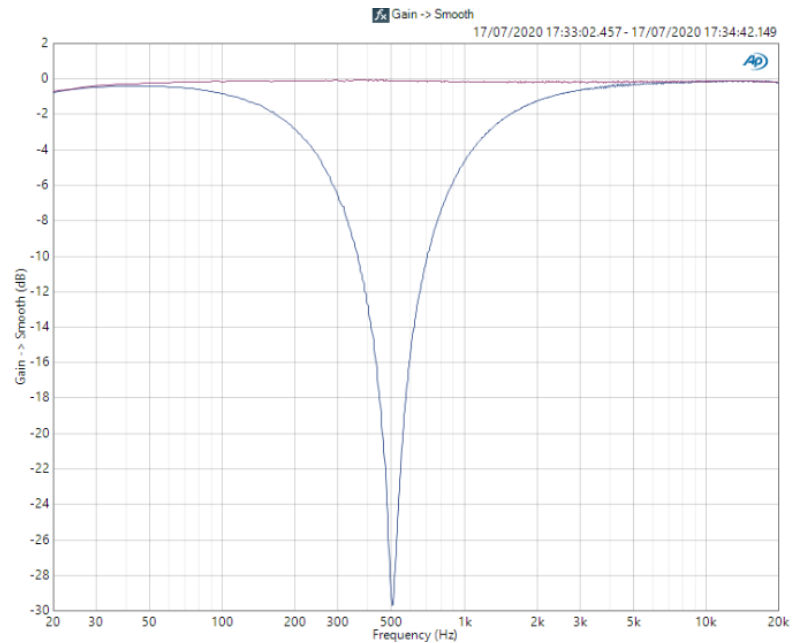
Fonte: AUTOR (2020)

Figura 47 – Saídas do *crossover* em 900 Hz para segunda e quarta ordem.



Fonte: AUTOR (2020)

Figura 48 – Resposta da soma dos canais do *crossover* de 500 Hz de segunda ordem, com e sem inversão de fase.



Fonte: AUTOR (2020)

Observa-se o cancelamento em 500 Hz, devido ao defasamento. Com a inversão da polaridade não há cancelamento e a resposta é plana na mesma região.

5 CONSIDERAÇÕES FINAIS

5.1 Conclusão

Foi possível implementar o sistema proposto no *hardware* disponível, o qual apresentou resultados dentro das especificações. As respostas foram coerentes entre os cálculos e as simulações feitas em MATLAB e a implementação no *hardware*, caracterizando bem o conceito e a validação do protótipo de produto, que foi a proposta principal deste projeto. Na implementação no kit de desenvolvimento houve alguns desafios, que demandaram uma análise minuciosa do que ocorria no processamento, de modo que o problema pudesse ser resolvido, como foi o caso do *display*. A interface, apesar de básica, mostrou-se bem funcional, e o usuário pode ajustar os parâmetros do sistema, que são precisos e coerentes entre o *display* e a resposta que o sistema de fato apresenta.

Quando o parâmetro Q é variado na aplicação em *hardware*, houve uma pequena variação no ganho quando este é positivo. Não foi identificada a causa desta anomalia. No mais, todos os parâmetros funcionaram perfeitamente.

5.2 Sugestões de trabalhos futuros

Para a melhoria do projeto, alguns pontos podem ser aprimorados. Pode-se implementar mais filtros, e talvez seja necessário melhorar o desempenho do processamento, também pode-se melhorar a usabilidade e praticidade para o usuário. Alguns pontos sugeridos:

- Implementar mais canais para o *crossover* e mais bandas de equalizadores.
- Buscar meios de reduzir o cascadeamento do processamento, aplicando os coeficientes nas mesmas rotinas ao invés de rotinas independentes para cada filtro cascadeado.
- Colocar o *display* em interrupção.

- Realizar processamento de dinâmica do sinal de entrada, evitar ganho excessivo e tratar variações grandes de amplitude (compressor-limitador).
- Implementar controles com botão tipo *encoder*, ou melhorar o incremento dos botões.
- Corrigir a causa do problema da variação do parâmetro Q.

REFERÊNCIAS

AARTS, R. M. **Optimally sensitive and efficient compact loudspeakers**. *Acoustical Society of America*, 2006. Citado na página 14.

BAEKGAARD, L. B. et al. **Designing Hearing Aid Technology to Support Benefits in Demanding Situations**. 2013. Disponível em: <<https://www.hearingreview.com/hearing-products/designing-hearing-aid-technology-to-support-benefits-in-demanding-situations-part-1>>. Acesso em: 20/08/2020. Citado na página 22.

BOCKRATH, D. **Amplifier Classes Explained**. 2016. Disponível em: <<https://www.bhphotovideo.com/explora/pro-audio/tips-and-solutions/amplifier-classes-explained>>. Acesso em: 05/06/2020. Citado na página 19.

BORTONI TECHNOLOGY. **Audio Science, Engineering and Voodoos**. 2020. Disponível em: <<https://www.bortoni.net/>>. Acesso em: 10/09/2020. Citado na página 70.

DAVIS, N. **An Introduction to Filters - Technical Articles**. 2017. Disponível em: <<https://www.allaboutcircuits.com/technical-articles/an-introduction-to-filters/>>. Acesso em: 05/08/2020. Citado na página 25.

DINIZ, P. S. R.; SILVA, E. A. B.; NETTO, S. L. **Projeto e Análise de Sistemas**. In: *Processamento Digital de Sinais*. [S.l.: s.n.], 2004. Citado na página 21.

ECMT, E. C. M. T. **Sampling Theory - Making Music with Computers**. 2005. Disponível em: <https://legacy.earlham.edu/~tobeyfo/musictechnology/4_SamplingTheory.html>. Acesso em: 20/06/2020. Citado na página 23.

HANNA, C. **Real-Time Control of DSP Parametric Equalizers**. 1994. Disponível em: <http://www.thatcorp.com/datashts/AES13-041_Control_of_DSP_Parametric_EQs.pdf>. Acesso em: 20/03/2020. Citado na página 24.

IFEACHOR, E. C.; JERVIS, B. W. **Application Approach**. In: *Digital Signal Processing*. [S.l.: s.n.], 2002. Citado 2 vezes nas páginas 24 e 25.

INGLE, V. K.; PROAKIS, J. G. In: *Digital Signal Processing using MATLAB*. [S.l.: s.n.], 2000. Citado na página 26.

KEIL CMSIS-DSP. **CMSIS DSP Software Library - Documentation**. 2010. Disponível em: <<https://www.keil.com/pack/doc/CMSIS/DSP/html/index.html>>. Acesso em: 23/06/2020. Citado 2 vezes nas páginas 27 e 62.

LAVRY, D. **Sampling Theory For Digital Audio**. 2004. Disponível em: <<http://lavryengineering.com/pdfs/lavry-sampling-theory.pdf>>. Acesso em: 28/05/2020. Citado na página 22.

LINKWITZ LAB. **Crossovers**. 2006. Disponível em: <<http://www.linkwitzlab.com/crossovers.htm>>. Acesso em: 03/04/2020. Citado na página 30.

LINKWITZ, S. **Active Crossover Networks for Noncoincident Drivers**. 1976. Disponível em: <http://www.thatcorp.com/datashts/AES13-041_Control_of_DSP_Parametric_EQs.pdf>. Acesso em: 20/05/2020. Citado na página 27.

NEVES, F. **Ping-pong Buffer para sistemas embarcados**. 2016. Disponível em: <<https://www.embarcados.com.br/ping-pong-buffer/>>. Acesso em: 23/06/2020. Citado 2 vezes nas páginas 59 e 60.

NOCETI FILHO, S. In: *Filtros Seletores de Sinais*. [S.l.: s.n.], 1998. p. 276. Citado na página 24.

PHILIPS. **I2S bus specification**. 1986. Disponível em: <https://web.archive.org/web/20070102004400/http://www.nxp.com/acrobat_download/various/I2SBUS.pdf>. Acesso em: 29/05/2020. Citado na página 23.

PREMIER SHOP. **O que é e para que serve o crossover no Som Automotivo**. 2017. Disponível em: <<https://blog.premiershop.com.br/o-que-e-e-para-que-serve-o-crossover-no-som-automotivo/>>. Acesso em: 10/05/2020. Citado na página 14.

SILVA, C. C.; SILVA, H. S. **ALTO-FALANTES E CAIXAS ACÚSTICAS, CARACTERÍSTICAS E UTILIZAÇÃO**. ELETRÔNICA SELENIUM S.A., 2002. Citado 3 vezes nas páginas 14, 20 e 21.

SMITH, M. T. **Projeto e Análise de Sistemas**. In: *Audio Engineer's Reference Books*. [S.l.: s.n.], 1999. ISBN 978-1136119743. Citado na página 20.

STMICROELECTRONICS. **UM1472 User manual - Discovery kit with STM32F407VG MCU**. 2017. Disponível em: <https://www.st.com/resource/en/user_manual/dm00039084-discovery-kit-with-stm32f407vg-mcu-stmicroelectronics.pdf>. Acesso em: 20/04/2020. Citado 2 vezes nas páginas 59 e 60.

STUART, J. R. **Coding High Quality Digital Audio**. 1998. Disponível em: <<https://web.archive.org/web/20070627075502/http://www.meridian.co.uk/ara/coding2.pdf>>. Acesso em: 27/05/2020. Citado na página 23.

TAVARES, D. **Crossover Digital com equalizador paramétrico**. 2020. Disponível em: <https://github.com/diogo0001/TCC_DSP_Filter/tree/master>. Acesso em: 15/06/2020. Citado na página 38.

TECLACENTER. **M-AUDIO FAST TRACK PRO INTERFACE**. 2015. Disponível em: <<https://www.teclacenter.com.br/m-audio-fast-track-pro-interface.html>>. Acesso em: 24/06/2020. Citado na página 62.

WILLIAMS, D. **Harmonics and THD in Audio Systems**. 2017. Disponível em: <<https://www.allaboutcircuits.com/technical-articles/total-harmonic-distortion-audio-systems-harmonics-subjectivist-rationalist/>>. Acesso em: 10/09/2020. Citado na página 37.

WOLFSON MICROELECTRONICS. **WM5102I - Audio Hub CODEC with Voice Processor DSP**. 2014. Disponível em: <https://www.mouser.com/datasheet/2/76/WM5102_v4.2-1141741.pdf>. Acesso em: 10/06/2020. Citado na página 61.

ZAIDI, A. **Advanced Digital Signal Processing**. 2010? Disponível em:
<[http://www-syscom.univ-mlv.fr/~zaidi/teaching/dsp-esipe-oc2/Course-Notes_
Advanced-DSP.pdf](http://www-syscom.univ-mlv.fr/~zaidi/teaching/dsp-esipe-oc2/Course-Notes-_Advanced-DSP.pdf)>. Acesso em: 29/05/2020. Citado na página 22.