UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ENGENHARIA DA COMPUTAÇÃO

RENAN RIBEIRO

DESENVOLVIMENTO DE UM EQUALIZADOR GRÁFICO DIGITAL DE 10 BANDAS

PATO BRANCO
2017

RENAN RIBEIRO

DESENVOLVIMENTO DE UM EQUALIZADOR GRÁFICO DIGITAL DE 10 BANDAS

Trabalho de Conclusão de Curso como requisito parcial à obtenção do título de Bacharel em Engenharia de Computação, do Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Rafael Cardoso
Coorientador: Prof. Dr. Gustavo Weber Denardin

PATO BRANCO
2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Engenharia de Computação

TERMO DE APROVAÇÃO

Às 8 horas e 20 minutos do dia 05 de dezembro de 2017, na sala V003, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, reuniu-se a banca examinadora composta pelos professores Rafael Cardoso (orientador), Gustavo Weber Denardin (coorientador), Everton Luiz de Aguiar e Fabio Luiz Bertotti para avaliar o trabalho de conclusão de curso com o título **Desenvolvimento de um Equalizador Gráfico Digital de 10 Bandas**, do aluno **Renan Ribeiro**, matrícula 1294903, do curso de Engenharia de Computação. Após a apresentação o candidato foi arguido pela banca examinadora. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

_____	_____
Rafael Cardoso Orientador (UTFPR)	Gustavo Weber Denardin Coorientador(UTFPR)
_____	_____
Everton Luiz de Aguiar (UTFPR)	Fabio Luiz Bertotti (UTFPR)
_____	_____
Profa. Beatriz Terezinha Borsoi Coordenador de TCC	Prof. Pablo Gauterio Cavalcanti Coordenador do Curso de Engenharia de Computação

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

RIBEIRO, Renan. Desenvolvimento de um equalizador gráfico digital de 10 bandas. 2017. 53f. Trabalho de Conclusão de Curso de Bacharelado em Engenharia de Computação - Universidade Tecnológica Federal do Paraná. Pato Branco, 2017.

Este trabalho de conclusão de curso propõe o desenvolvimento de um equalizador gráfico digital de 10 bandas utilizando o *kit* de desenvolvimento STM32F746G Discovery. Um equalizador gráfico de áudio é um dispositivo formado por um conjunto de filtros concebido para compensar características indejáveis na resposta de magnitude, atuando no espectro audível humano, de 20 Hz a 20 kHz. Os filtros digitais que compõem o equalizador foram projetados a partir da norma ISO 266. O equalizador proposto neste trabalho oferece uma interface gráfica de usuário na qual é possível alterar o ganho individual de cada um dos 10 filtros e alterar algumas características como volume e balanço. Essa interface gráfica foi implementada em um *display* LCD-TFT com *touchscreen* capacitivo disponível no *kit* de desenvolvimento.

Palavras-chave: Equalizador. Filtros digitais. *Graphical User Interface*. ISO 266.

ABSTRACT

RIBEIRO, Renan. Development of a 10-band digital graphic equalizer. 2017. 53f. Trabalho de Conclusão de Curso de Bacharelado em Engenharia de Computação - Universidade Tecnológica Federal do Paraná. Pato Branco, 2017.

This work proposes the development of a 10-band digital graphic equalizer using the STM32F746G Discovery development kit. An audio graphic equalizer is a device formed by a set of filters designed to compensate for undesirable characteristics in the response of magnitude, operating in the human audible spectrum, from 20 Hz to 20 kHz. The digital filters that make up the equalizer were designed from ISO 266. The equalizer proposed in this work offers a graphical user interface where it is possible to change the individual gain of each of the 10 filters and to change some characteristics such as volume and balance. This graphical interface was implemented using a LCD-TFT display with capacitive touchscreen available in the development kit.

Palavras-chave: Equalizer. Digital filters. Graphical User Interface. ISO 266.

Lista de ilustrações

Lista de códigos

Lista de tabelas

Lista de abreviaturas e siglas

ADC	*Analog to Digital Converter* (Conversor Digital Analógico)
CA	Corrente Alternada
CD	*Compact Disc* (Disco Compacto)
DAC	*Digital to Analog Converter* (Conversor Digital Analógico)
DSP	*Digital Signal Processor* (Processador Digital de Sinais)
EQ	Equalizador
FIR	*Finite Impulse Response* (Resposta Finita ao Impulso)
FT	Função Transferência
GUI	*Graphical User Interface* (Interface Gráfica do Usuário)
I^2C	*Inter-integrated Circuit* (Circuito Inter-Integrado)
I^2S	*Inter-IC Sound* (Circuito Inter-Integrado de Som)
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
IIR	*Infinite Impulse Response* (Resposta Infinita ao Impulso)
RLC	Resistor, Indutor e Capacitor
RTOS	*Real Time Operating System* (Sistema Operacional de Tempo-real)
SAI	*Serial Audio Interface* (Interface Serial de Áudio)
SPI	*Serial Peripheral Interface* (Interface Periférica Serial)

Sumário

1 INTRODUÇÃO

Este capítulo está dividido da seguinte maneira: a Seção 1.1 apresenta uma visão geral do tema abordado, a Seção 1.2 apresenta os objetivos deste trabalho e a Seção 1.3 trata da justificativa desta pesquisa.

1.1 Considerações Iniciais

A percepção auditiva tem início antes do nascimento (HUOTILAINEN; NÄÄTÄNEN, 2010), sendo esta responsável por receber e interpretar estímulos sonoros provenientes do ambiente. Essa característica faz com que as pessoas sejam capazes de reconhecer vozes, compreender a fala e apreciar uma música.

A música é fonte de entretenimento para as pessoas. Há pessoas que ouvem determinada música motivadas pela letra, ou pelo sentimento que o compositor quis transmitir, enquanto outras, pela melodia. Por outro lado a música pode ser considerada um bem de consumo, pois é algo que as pessoas buscam para se satisfazer.

Para Lemos (2014), como o mercado dos bens de consumo é amplo, uma das prioridades das empresas é a inovação. E, mais especificamente, no setor dos equipamentos eletrônicos, tem-se duas situações: as inovações que se utilizam da eletrônica analógica e as que se utilizam da eletrônica digital. Os equipamentos eletrônicos não precisam, necessariamente, ser somente de um dos tipos. Computadores, máquinas ditas estritamente digitais, possuem em seus circuitos componentes analógicos (BRAGA, 2014).

No início da década de 1980, em uma parceria entre as empresas Sony e Philips, foi lançado o *compact disc* (CD). Ao contrário dos produtos de áudio até então desenvolvidos, o CD não utiliza a captura contínua dos sons, mas sim, a captura em amostras do som em intervalos de tempo iguais. Ao serem reproduzidas em um conversor analógico digital (ADC), essas amostras digitais são recombinadas em um conversor digital analógico (DAC) para formar um fluxo contínuo de som e serem reproduzidas de forma analógica. Também na década de 1980 foram desenvolvidos os primeiros aparelhos eletrônicos digitais para recompor um sinal analógico de áudio, tais quais os CD *players*, os equalizadores, processadores de efeitos e os gravadores digitais (SONY, 2010).

Ao aparelho ou sistema no qual se ajusta o ganho de determinada frequência de todo o espectro audível, dá-se o nome de equalizador (EQ). O equalizador é uma das peças fundamentais de correção de som para um audiófilo. O equalizador é capaz de fazer com que uma

música tenha características sonoras específicas, dependendo dos ajustes implantados.

Equalizadores são dispositivos concebidos para compensar características indesejáveis na resposta de frequência do sistema de áudio. Um equalizador gráfico de áudio atua no espectro audível humano, de 20 Hz a 20 kHz. O termo gráfico se refere ao modo como os botões do tipo *slider* são definidos no painel do equipamento, de tal modo que a posição dos botões deslizantes representam o gráfico da resposta de frequência desejada (BALLOU, 2008). Equalizadores são, geralmente, baseados em filtros de primeira ou segunda ordem, dos tipos *shelving filters* ou *peak filters* para Zölzer et al. (2011) e dos tipos passa-banda e rejeita-banda para Montgomery (2005). Filtros rejeita-banda e *shelving filters* aumentam ou diminuem o ganho nas bandas de baixa ou alta frequência. Filtros passa-banda e *peak filters* atuam aumentando ou diminuindo o ganho em uma frequência central.

Para aplicações de áudio, Tan (2008) descreve o equalizador digital como o sistema para se obter o som desejado, alterando o ganho dos filtros de diferentes bandas de frequência. Outras aplicações citadas por Tan (2008) incluem: ajustar o som levando em consideração a acústica de uma sala, remover ruídos indesejados e aumentar o ganho em determinada banda passante.

Este trabalho tem como objetivo desenvolver um equalizador gráfico de 10 bandas com processamento em tempo real, utilizando-se dos estudos sobre processamento digital de sinais (DSP) e filtros digitais. Os filtros digitais desenvolvidos serão, então, implementados em um kit de desenvolvimento Discovery Kit with STM32F746NG MCU 32-bit, sendo possível atuar nos ganhos individuais de cada filtro.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um equalizador gráfico digital de 10 bandas, com captura e reconstrução de sinal de áudio analógico.

1.2.2 Objetivos Específicos

- Definir a frequência central e a largura de cada uma das bandas de frequência, segundo a norma ISO 266, que define um conjunto de frequências preferenciais, parâmetros de corte, largura de banda e quantidade de filtros (ACOUSTICS. . . , 1997).

- Projetar os filtros digitais IIR utilizando aproximação de Butterworth.

- Realizar a comunicação entre o microcontrolador e o *codec* de áudio, para a realização da captura dos sinais analógicos e transformação em palavras digitais e a conversão das palavras digitais processadas em sinais analógicos.
- Implementar os filtros digitais IIR no *kit* de desenvolvimento STM32F746G *Discovery* utilizando a linguagem de programação C.
- Implementar uma *Graphical User Interface* (GUI), sensível ao toque para o controle do equalizador.
- Realizar testes de validação para verificar se o áudio é processado sem atrasos na taxa de amostragem de 48 kHz e verficar se o ganho de saída de cada uma das bandas do áudio processado é condizente com o resultado teórico.

1.3 Justificativa

Dizer que o áudio está perfeito é subjetivo, ou seja, é algo que pertence à consciência de cada indivíduo e está baseado na sua interpretação individual. Com base nisso, surge a necessidade de ajustar certas frequências do som ou da música, atenuando frequências indesejáveis como ruídos e microfonia, ou amentando o ganho em determinadas frequências. Essas alterações podem ser feitas para corrigir problemas de acústica de uma sala ou simplesmente pelo gosto do ouvinte, seja na fase de produção do conteúdo de áudio, ou quando já está disponível para o consumidor final.

Para Zölzer (2008) o equalizador é um dos equipamentos mais importantes para o processamento de sinais de áudio. As funções mais complexas são usadas em estúdios de gravação. Entretanto, em quase todos os bens de consumo como rádios de automóveis e amplificadores *hifi*, funções simples de filtros são utilizadas para a equalização do áudio (ZÖLZER, 2008).

A vantagem de implementar um equalizador digital, ao invés de um equalizador analógico, está na simplicidade de alteração do projeto. Para um EQ digital, aumentar o número de bandas, dimensionar a ordem e a aproximação dos filtros digitais é um trabalho apenas de *software*. Uma vez desenvolvido o conjunto do *hardware*, todas as alterações de composição dos filtros podem ser feitas apenas alterando o código de programação. O mesmo não acontece com um EQ analógico, para mudar a aproximação de um EQ de 2ª ordem, de Butterworth para Bessel, por exemplo, é necessário que todo o projeto de *hardware* seja refeito.

2 REFERENCIAL TEÓRICO

Este capítulo tem por objetivo realizar uma breve contextualização sobre os conceitos teóricos que serão utilizados no desenvolvimento deste trabalho.

2.1 Sinais

2.1.1 Introdução

Sinais são matematicamente representados por meio de funções de uma ou mais variáveis independentes e contêm informações sobre o comportamento ou natureza de algum fenômeno (OPPENHEIM; WILLSKY; NAWAB, 1983). Por exemplo, o sinal de fala humana, originado do ar que sai dos pulmões, passando pelas cordas vocais, pode ser representado matematicamente pela pressão acústica por uma função do tempo.

A Figura 1, mostra a variação da pressão em função do tempo. A imagem foi capturada por um microfone, dispositivo que transforma as vibrações eletromecânicas em corrente elétrica.

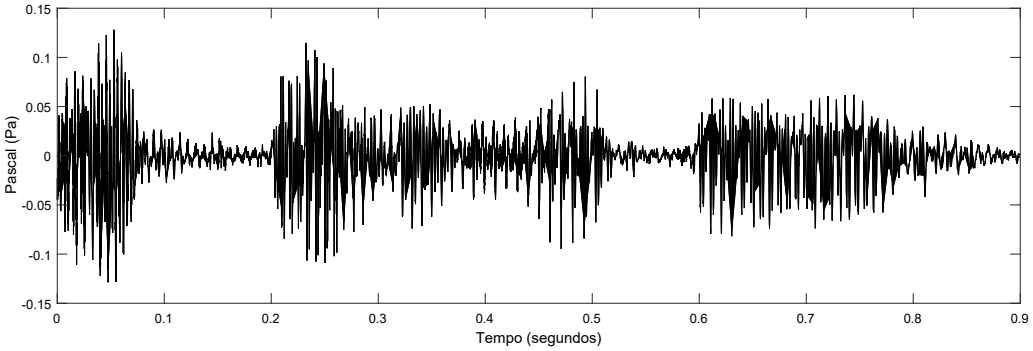


Figura 1 – Forma de onda de um sinal de fala. Áudio capturado durante o pronunciamento da frase: "Equalizador Gráfico".

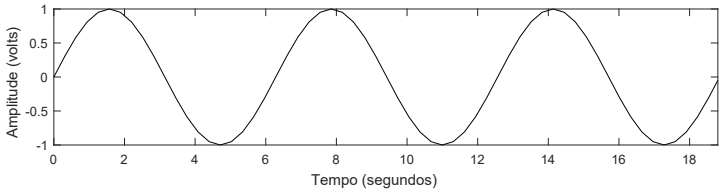
Fonte: Autoria própria.

2.1.2 Classificação de Sinais

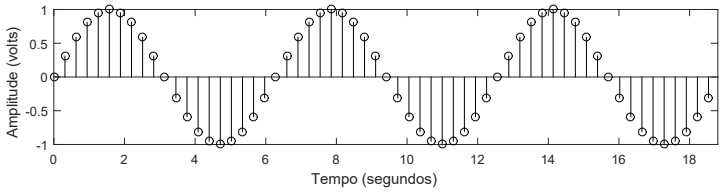
2.1.2.1 Sinais contínuos e discretos no tempo

Para Haykin e Veen (2001), um sinal $x(t)$ é um sinal de tempo contínuo se ele for definido para todo tempo t. Já um sinal $x[n]$ é um sinal de tempo discreto se ele for definido somente

em instantes conhecidos de tempo. A Figura 2a mostra um sinal contínuo em função do tempo, já a Figura 2b, um sinal discreto em função do tempo.



a) Sinal contínuo.



a) Sinal discreto.

Figura 2 – Sinais contínuo e discreto em função do tempo.

Fonte: Autoria própria.

Sinais de tempo contínuo surgem naturalmente quando uma forma física como um onda sonora ou uma intensidade luminosa é convertida, por meio de um transdutor, em um sinal elétrico. Um sinal sonoro e uma corrente elétrica, ambos em função do tempo, são exemplos de sinais de tempo contínuo. Um sinal de tempo discreto é habitualmente obtido por meio de um sinal de tempo contínuo em amostras igualmente distribuídas no tempo.

2.1.2.2 Sinais analógicos e digitais

Um sinal analógico pode assumir infinitos valores de amplitude. Por outro lado, um sinal digital assume apenas valores finitos em sua amplitude e tempo (LATHI, 2006). Sinais analógicos podem ser limitados por uma faixa de valores máximos e mínimos, mas há, ainda, nessa faixa, infinitas possibilidades de valores. Sinais puros de áudio são analógicos, como os sinais capturados por um microfone. Sinais digitais possuem um número finito de valores dentro de uma faixa. Esse número pode ser baixo, como dois valores distintos (binários), ou pode ser alto, limitado pelo ADC do equipamento, embora não seja infinito.

2.2 Filtros

2.2.1 Conceitos Introdutórios

Filtros eletrônicos são projetados para separar os sinais desejáveis dos não desejáveis, ou em outras aplicações, simplesmente mudam o conteúdo do espectro em frequência alterando a forma de onda de um sinal (THEDE, 2004).

Existem muitos tipos de filtros e muitas maneiras que eles podem ser classificados. A seletividade de frequência de um filtro é provavelmente o método mais comum de classificação (THEDE, 2004). Os filtros podem ser do tipo passa-baixas, passa-altas, passa-banda e rejeita-banda. Cada nome indica qual faixa de frequência será alterada. Por exemplo, o filtro passa-baixas permite que as baixas frequências passem com atenuação mínima, enquanto as altas frequências seriam significativamente reduzidas.

Thede (2004) descreve ainda que filtros podem ser classificados quanto ao método para se aproximar do gabarito do filtro. Alguns métodos de aproximação focam em obter uma banda passante linear, enquanto outros enfatizam a capacidade do filtro para atenuar os sinais.

Os filtros são, ainda, classificados pelo método de implementação utilizado. Filtros analógicos são construídos para filtrar sinais analógicos usando componentes em placas de circuito, enquanto filtros digitais podem ser parte de um sistema digital maior que contenham outras funções.

2.2.2 Seletividade do Filtro

A seletividade é o método mais comum de classificação dos filtros (THEDE, 2004). Denominações como passa-baixas, passa-altas, passa-banda e rejeita-banda são usados para categorizar os filtros, mas é preciso mais informações para descrever completamente um filtro.

Segundo Thede (2004), há duas especificações necessárias para definir um filtro completamente. A primeira é a especificação de frequência utilizada para descrever a banda passante e de parada, que pode ser em Hertz (Hz) ou em radianos/segundo (rad/seg). A variável f é a frequência medida em Hz, enquanto w é a frequência angular medida em rad/seg.

$$A_{dB} = 20log(\frac{V_o}{V_i}) \tag{2.1}$$

A segunda especificação necessária para o projeto do filtro é a característica do ganho na banda de passagem e parada. O ganho do filtro, representado na Equação 2.1 é simplesmente a razão entre o nível do sinal de saída, V_o, e o nível do sinal de entrada, V_i. Se o ganho do filtro é maior que 1, o sinal de saída é maior que o sinal de entrada, enquanto que, se o ganho

for menor que 1, o sinal de saída é menor que o sinal de entrada. Tipicamente, o ganho de um filtro é representado em dB.

2.2.2.1 Filtro Passa-Baixas

A função básica de um filtro passa-baixas é deixar passar as baixas frequências com o mínimo de atenuação e atenuar as altas frequências (GOBIND, 1976). Como demonstrado na Figura 3, A_{pass} é o máximo ganho da banda passante, A_{stop} é o ganho mínimo da banda de parada, f_c é a frequência de corte e f_{stop} é a frequência da banda de parada. Na banda passante o filtro varia entre 0 dB e A_{pass}, enquanto o ganho na banda de parada pode variar entre A_{pass} e o infinito negativo. Com base nisso, a seletividade do filtro passa-baixas pode ser especificada em 4 parâmetros: o ganho da banda de parada A_{stop}, o ganho da banda de passagem A_{pass}, a frequência de corte f_c e a frequência de parada f_{stop} (THEDE, 2004).

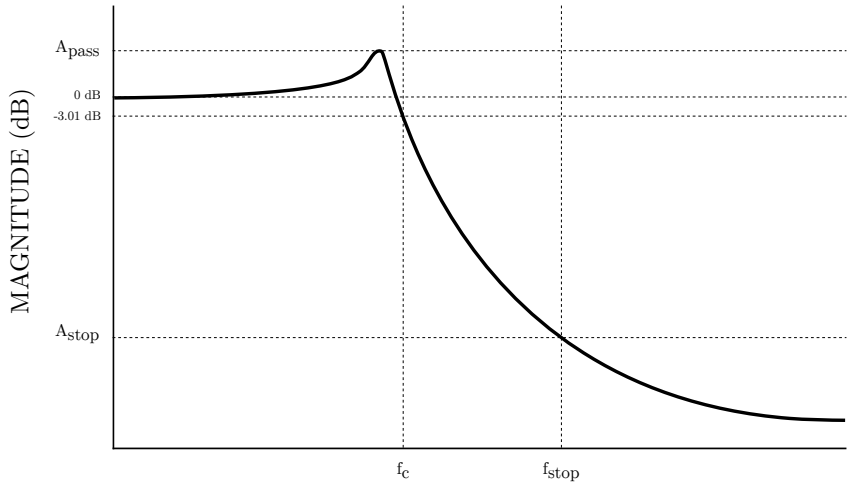


Figura 3 – Filtro Passa-Baixas.

Fonte: Autoria própria.

2.2.2.2 Filtro Passa-Altas

Os filtros passa-altas são usados para eliminar as baixas frequências de um sinal (THEDE, 2004). Em um filtro passa-altas, ilustrado na Figura 4, a banda passante se estende desde f_c até o infinito (para filtros analógicos).

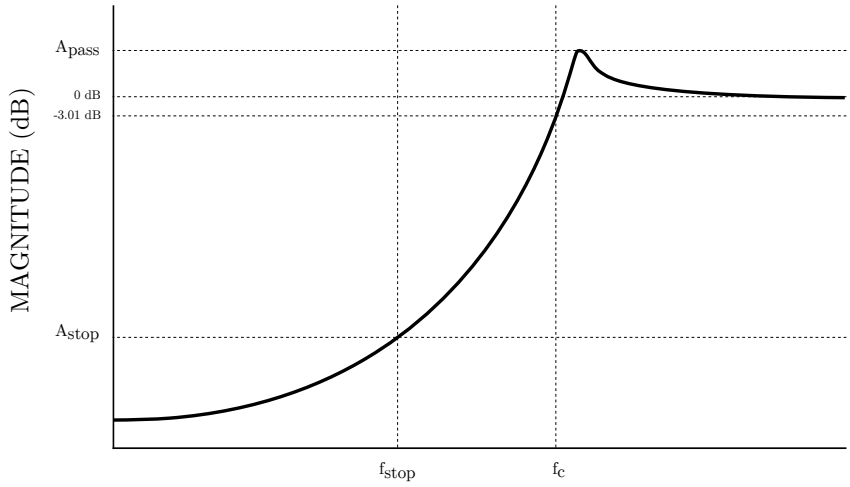


Figura 4 – Filtro Passa-Altas.

Fonte: Autoria própria.

2.2.2.3 Filtro Passa-Banda

Um filtro passa-banda permite que esteja presente na saída determinada banda de frequência atenuando as frequências acima e abaixo dessa banda (THEDE, 2004). Para o filtro passa-banda, cujo digrama de Bode é representado na Figura 5, a banda passante é definida entre f_{c1} e f_{c2}. O filtro passa-banda contém duas faixas de atenuação. A primeira se estende de 0 Hz (DC) até f_{stop1}. Já a segunda faixa de atenuação, se estende de f_{stop2} até o infinito (para filtros analógicos). Para o projeto de um filtro passa-banda, há ainda a escolha de A_{pass}, A_{stop1} e A_{stop2} ambos em dB.

Um bom exemplo para a aplicação de um filtro passa-banda é o processamento de sinais de voz. A voz humana tem o conteúdo de frequência localizado principalmente na faixa entre 300 Hz e 3000 Hz. Neste caso, seria possível definir f_{c1} como 300 Hz e f_{c2} como 3000 Hz. E as frequências de borda de parada, f_{stop1} e f_{stop2} com a atenuação desejada.

2.2.2.4 Filtro Rejeita-Banda

Filtros do tipo rejeita-banda são usados para rejeitar uma determinada faixa de frequência (GOBIND, 1976). A Figura 6 ilustra a resposta em frequência de um filtro rejeita-banda. A faixa de interrupção existe entre f_{stop1} e f_{stop2}. Já a faixa de passagem, se entende desde 0 Hz até f_{c1} e de f_{c2} até o infinito (para filtros analógicos).

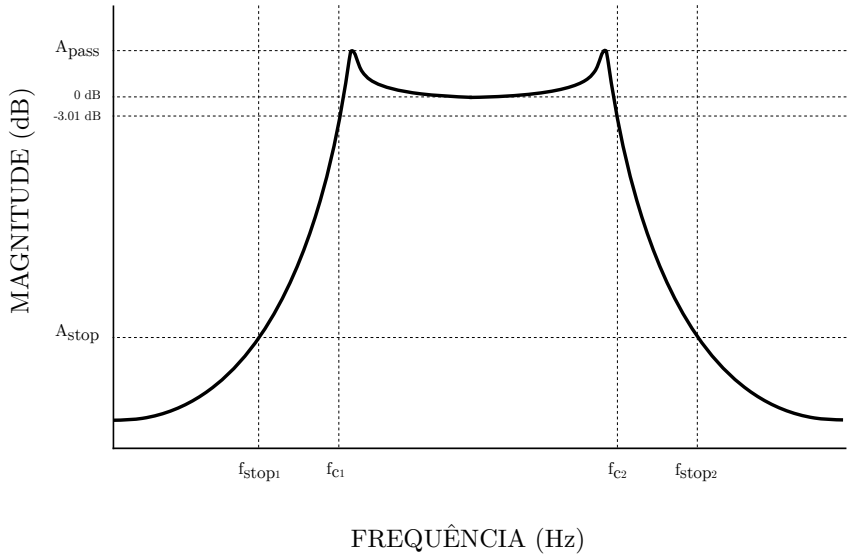


Figura 5 – Filtro Passa-Banda.

Fonte: Autoria própria.

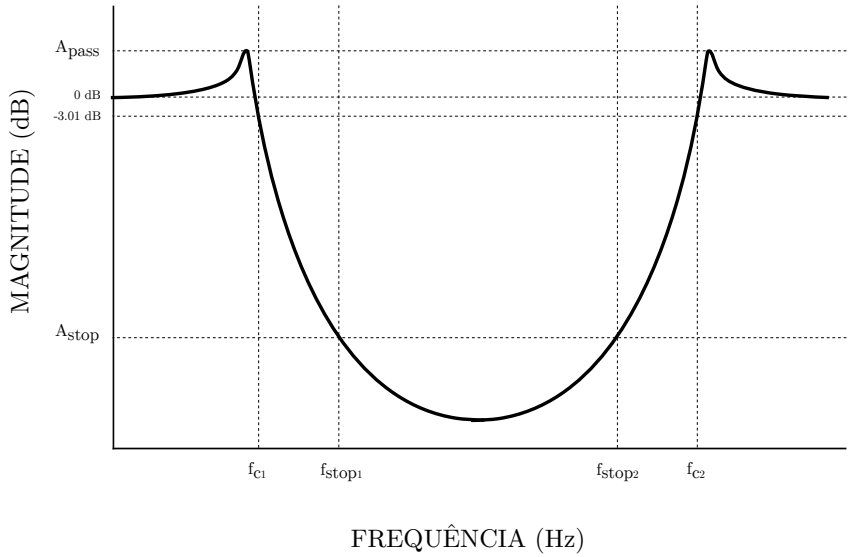


Figura 6 – Filtro Rejeita-Banda.

Fonte: Autoria própria.

2.2.3 Classificação Quanto à Implementação do Filtro

Conforme Thede (2004), após um filtro ser completamente especificado, é necessário escolher entre a tecnologia analógica e digital para implementá-lo. As diferenças de projeto de um filtro analógico e digital se baseiam principalmente nas diferenças entre sinais analógicos e digitais.

Qualquer sinal pode ser representado no domínio do tempo, traçando sua amplitude em

função do tempo. No entanto, as variações de amplitude e tempo podem ser contínuas, referindo-se a sinais analógicos, ou discretas, relacionadas aos sinais digitais.

2.2.4 Filtros Analógicos

Filtros analógicos são filtros contínuos no tempo, ou ainda, filtros que podem ser implementados com amplificadores operacionais, resistores, indutores e capacitores (PAARMANN, 2001). Os conceitos da teoria de filtros analógicos ajudam a compreender estudos mais avançados, como os de filtros digitais.

2.2.4.1 Projeto de um Filtro Analógico

Para iniciar o projeto de um filtro analógico é necessário, inicialmente, determinar ou encontrar as seguintes especificações: ganho da banda passante A_{pass}, ganho da banda de rejeição A_{stop}, frequência(s) de corte(s) f_c e frequência(s) de parada(s) f_{stop}. Com os parâmetros iniciais identificados, faz-se necessário definir a seletividade do filtro.

Para cada seletividade há uma função transferência (FT) relacionada. Carter e Mancini (2004) descreve a FT de 2^a ordem de cada uma das seletividades. Sendo elas:

- Passa-Baixas

$$A_i(s) = \frac{A_o}{1 + a_i s + b_i s^2} \tag{2.2}$$

- Passa-Altas

$$A_i(s) = \frac{A_\infty}{1 + \frac{a_i}{s} + \frac{b_i}{s^2}} \tag{2.3}$$

- Passa-Banda

$$A(s) = \frac{\frac{A_m}{Q} s}{1 + \frac{1}{Q} s + s^2} \tag{2.4}$$

- Rejeita-Banda

$$A(s) = \frac{A_0(1 + s^2)}{1 + \frac{1}{Q} s + s^2} \tag{2.5}$$

Nas equações 2.2, 2.3, 2.4 e 2.5 as grandezas A_0, A_m, A_∞, a_i e b_i representam, respectivamente o ganho nas baixas frequências, o ganho da frequência central, o ganho nas altas frequências e os coeficientes dados pelas aproximações como Butterworth, Chebyshev e Bessel. Nas equações 2.4 e 2.5, Q é o fator de qualidade definido pela razão entre a frequência central f_m e a largura de banda B_w.

Caso seja necessário implementar um filtro de ordem superior, o projeto pode ser realizado a partir de blocos de segunda ou primeira ordem em cascata, como mostrado na Figura 7.

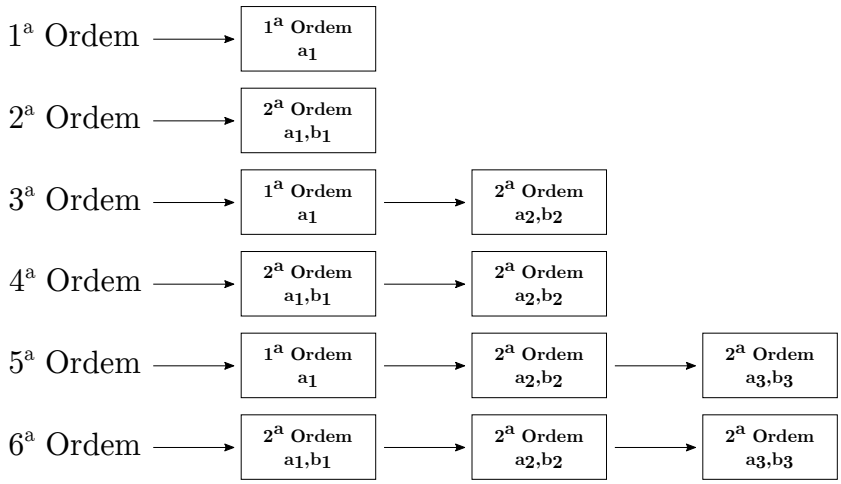


Figura 7 – Blocos de primeira e segunda ordem em cascata para projeto do filtro de ordem superior.

Fonte: Adaptado de Carter e Mancini (2004).

Com os parâmetros iniciais definidos e escolhida a seletividade entre passa-baixas, passa-altas, passa-banda e rejeita-banda, o próximo passo para projeto do filtro analógico é definir a melhor aproximação para o filtro ideal. A Tabela 1, mostra o valor dos coeficientes para a construção de um filtro de até 8^a ordem, para a aproximação de Butterworth.

Com os parâmetros iniciais, a função transferência da seletividade e os coeficientes das aproximações, é possível projetar o filtro analógico.

2.2.5 Aproximação do Filtro

A aproximação de um filtro consiste em encontrar uma função cujas características estejam dentro de uma faixa aceitável (GOBIND, 1976). Há diferentes maneiras de se aproximar um filtro de uma resposta ideal. Por exemplo, uma aproximação que enfatiza a necessidade de distorção mínima na banda passante pode comprometer sua transição na banda de parada. Outros projetos podem necessitar de uma rápida transição na banda de parada, permitindo alguma distorção na banda passante. As aproximações mais populares são conhecidas como: Butterworth, Chebyshev e Bessel.

2.2.5.1 Aproximação de Butterworth

A aproximação de Butterworth proporciona um ganho na banda passante monotonicamente decrescente e sem oscilações. Carter e Mancini (2004) citam que a aproximação de Butterworth é muita utilizada como filtro *anti-aliasing* em aplicações de conversão de dados, sendo necessário haver precisão dos sinais em toda a banda passante.

Tabela 1 – Tabela dos coeficientes de Butterworth.

Coeficientes de Butterworth					
Ordem(n)	i	a_i	b_i	$k_i = fc_i/fc$	Q_i
1	1	1.000	0.0000	1.0000	-
2	1	1.4142	1.0000	1.0000	0.71
3	1	1.0000	0.0000	1.0000	-
	2	1.0000	1.0000	1.272	1.00
4	1	1.8478	1.0000	0.719	0.54
	2	0.7654	1.0000	1.390	1.31
5	1	1.0000	1.0000	1.000	-
	2	1.6180	1.0000	0.859	0.62
	3	0.6180	1.0000	1.448	1.62
6	1	1.9319	1.0000	0.676	0.52
	2	0.4142	1.0000	1.000	0.71
	3	0.5176	1.0000	1.479	1.93
7	1	1.0000	1.0000	1.000	-
	2	1.8019	1.0000	0.745	0.55
	3	1.2470	1.0000	1.117	0.80
	4	0.4450	1.0000	1.449	2.25
8	1	1.9616	1.0000	0.661	0.51
	2	1.6629	1.0000	0.829	0.60
	3	1.1111	1.0000	1.206	0.90
	4	0.3902	1.0000	1.512	2.56

Fonte: Adaptado de Carter e Mancini (2004).

A Figura 8 mostra a resposta em frequência de um filtro passa-baixas com aproximação de Butterworth. Quanto maior a ordem do filtro, mais plana é a banda passante e mais rápida a transição entre f_c e f_{stop}.

2.2.5.2 Aproximação de Chebyshev

Segundo Carter e Mancini (2004), a aproximação de Chebyshev proporciona uma rápida transição entre a frequência de corte f_c e a banda de parada. No entanto, o ganho da banda passante contém ondulações. Quanto maior a ordem do filtro, menor a transição entre f_c e f_{stop}.

2.2.5.3 Aproximação de Bessel

Os filtros projetados utilizando a aproximação de Bessel podem fornecer uma resposta de fase linear (CARTER; MANCINI, 2004). No entanto, a banda passante de uma aproximação Bessel não é tão linear quanto a banda passante de uma aproximação Butterworth e a transição entre a frequência de corte f_c e f_{stop} não é tão abrupta quanto a de Chebyshev.

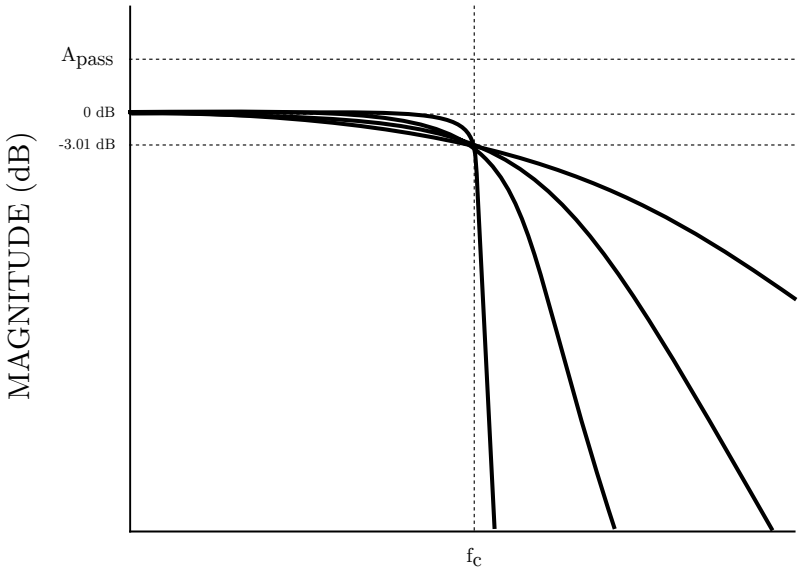


Figura 8 – Aproximação de Butterworth.

Fonte: Autoria própria.

2.2.6 Filtros Digitais

Para Smith (1999), apesar de filtros analógicos serem baratos e possuírem grande alcance em amplitude e frequência, os filtros digitais possuem um desempenho superior dependendo da forma como os problemas de filtragem são tratados. Smith (1999) exemplifica que é possível projetar um filtro digital passa-baixas com ganho de $+/-0,0002$ dB para uma frequência de 1000 Hz e um ganho menor que $0,0002$ dB para frequências acima de 1001 Hz.

Kuo, Lee e Tian (2006) define um filtro digital como um algoritmo matemático implementado em *hardware*, *firmware* ou *software* para alcançar alguns objetivos de filtragem. (SMITH, 1999) cita que uma forma de implementar um filtro digital é por meio da convolução entre o sinal de entrada e a resposta ao impulso do filtro desejado. Esse tipo de implementação tem a denominação *Finite Impulse Response* (FIR). Outra maneira de implementar filtros digitais é por meio da recursão. Os filtros recursivos são uma extensão dos filtros por convolução, porém, necessitam de valores passados da saída, além dos valores de entrada. A resposta ao impulso dos filtros recursivos são senóides que diminuem exponencialmente em amplitude. Isso faz com que sua resposta ao impulso seja infinita. Por essa característica, os filtros recursivos são chamados de *Infinite Impulse Response* (IIR).

2.2.7 Filtros Digitais FIR

Tan (2008) descreve um filtro FIR pela seguinte relação entre entrada e saída:

$$y(n) = \sum_{i=0}^{K} b_i x(n-i), \tag{2.6}$$

em que b_i representa os coeficientes do filtro. Aplicando a transformada Z em ambos os lados da equação 2.6, obtêm-se a função transferência que representa o filtro FIR:

$$H(z) = \frac{Y(z)}{X(z)} = b_0 + b_1 z^{-1} + \cdots + b_K z^{-K}. \tag{2.7}$$

Para Kuo, Lee e Tian (2006) as vantagens de um filtro digital FIR são: ser estável, pois sua saída não é dependente das saídas passadas e poder gerar um filtro com fase linear.

2.2.8 Filtros Digitais IIR

Usualmente, o projeto de um filtro digital IIR se inicia com o projeto de um filtro analógico, aplicando em seguida, técnicas de mapeamento para transformá-lo do plano S para o plano Z (KUO; LEE; TIAN, 2006).

Um filtro IIR é descrito usando equações diferença (TAN, 2008), como mostrado a seguir:

$$y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_M x(n-M) - a_1 y(n-1) - \cdots - a_N y(n-N). \tag{2.8}$$

Aplicando a transformada Z na equação 2.8:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}, \tag{2.9}$$

em que b_i e a_i são os coeficientes do numerador e denominador, respectivamente.

Kester (2000) descreve as seguintes vantagens para a escolha dos filtros IIR: filtros IIR são mais eficientes que filtros FIR, pois necessitam de menos memória e processos de multiplicação; filtros IIR podem ser projetados com base em filtros analógicos; filtros IIR podem apresentar problemas de instabilidade em ordens elevadas, mas isso se torna muito menos provável quando projetados em cascatas de segunda ordem. Como citado por Kester (2000), se o tempo de processamento for crucial para o projeto dos filtros, a implementação IIR é a melhor escolha.

2.2.8.1 Transformada Bilinear

Tan (2008) define os seguintes passos para o projeto de um filtro digital IIR pela transformada bilinear: (1) transformar as especificações do filtro digital para especificações do

filtro analógico, (2) projetar e obter a função transferência do filtro analógico e (3) aplicar a transformada bilinear.

O mapeamento do plano S para o plano Z utilizando a transformada bilinear se dá pela equação 2.10:

$$H(z) = H(s)|_{s_{(tb)}}, \tag{2.10}$$

onde

$$s_{(tb)} = \frac{2(z-1)}{T(z+1)}, \tag{2.11}$$

sendo T o período de amostragem.

Aplicando a transformada bilinear na equação do filtro de interesse, chega-se na equação geral:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}. \tag{2.12}$$

Os coeficientes a_i e b_i pertencem ao conjunto dos números reais.

2.3 ISO 266

O padrão internacional ISO 266 tem o objetivo de especificar uma série de frequências preferenciais, a fim de fornecer uma base comum para comparar resultados de medidas acústicas (ACOUSTICS. . . , 1997). O padrão internacional fornece uma série de relações com a qual é possível definir os parâmetros de corte de um conjunto de filtros passa-banda. A série de frequências é calculada a partir da frequência central de 1000 Hz. A relação entre duas frequências centrais é dada por:

$$f_{c0} = 2^N, \tag{2.13}$$

em que N é 1/3 para 1/3 de oitava de largura de banda. As larguras de banda mais comuns são: 1/1, 1/3, 1/6, 1/12 e 1/24 de oitava.

A partir da frequência central f_{c0}, calculada por (ADDING. . . , 2017):

$$f_{c0} = \sqrt{f_{c1} \cdot f_{c2}}, \tag{2.14}$$

e da relação entre f_{c1} e f_{c2}, dada por:

$$f_{c1} = \frac{f_{c2}}{2^N}, \tag{2.15}$$

é possível determinar as frequências de corte inferior f_{c1} a partir de (ADDING..., 2017)

$$f_{c1} = \frac{f_{c0}}{2^{\frac{N}{2}}}, \tag{2.16}$$

e superior f_{c2} como (ADDING..., 2017):

$$f_{c2} = f_{c0} \cdot 2^{\frac{N}{2}}. \tag{2.17}$$

Para a implementação de um EQ de 10 bandas será necessário calcular as frequências centrais e de corte para N=1, ou seja, para 1 oitava de largura de banda. A Tabela 2 mostra as frequências centrais com suas respectivas frequências de corte inferior e superior para um EQ de 1 oitava de largura de banda.

Tabela 2 – Cálculo das frequências centrais e de corte para um equalizador de 10 bandas.

Frequência Inferior f_1 (Hz)	Frequência Central f_0 (Hz)	Frequência Superior f_2 (Hz)
22	32	44
44	64	88
88	125	177
177	250	355
355	500	710
710	1000	1420
1420	2000	2840
2840	4000	5680
5680	8000	11360
11360	16000	22720

A Figura 9 mostra a resposta em frequência de cada uma das bandas definidas.

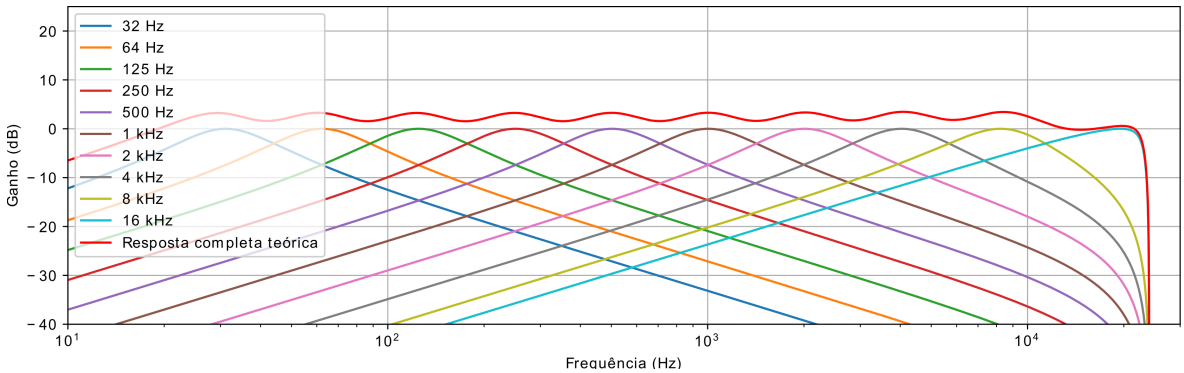


Figura 9 – Resposta em frequência das bandas definidas.

Fonte: Autoria própria.

3 METODOLOGIA

Este trabalho tem como objetivo, já especificado anteriormente, desenvolver um equalizador gráfico digital de 10 bandas, com captura e reconstrução de áudio analógico. A Figura 10, estruturada na forma de um diagrama de blocos, define a ordem das etapas em que o áudio será submetido.

Figura 10 – Etapas em que o áudio será submetido.

Fonte: Autoria própria.

O sistema que representa a implementação do equalizador gráfico de forma detalhada está representado na Figura 11.

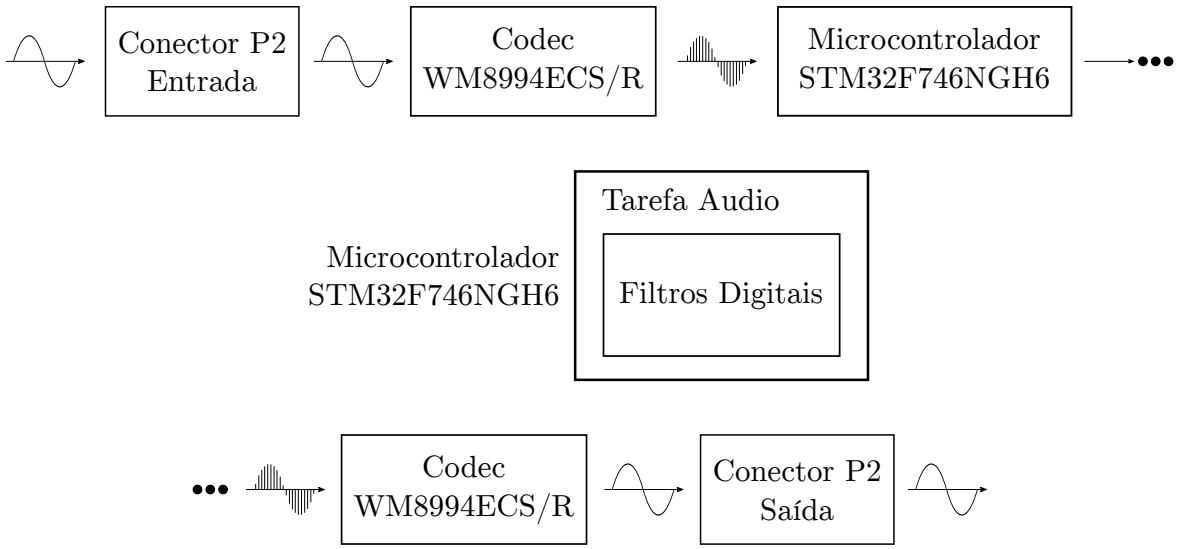


Figura 11 – Diagrama de blocos do sistema.

Fonte: Autoria própria.

O áudio proveniente de alguma fonte analógica é inserido no sistema por meio de um conector do tipo P2 disponível no *kit* de desenvolvimento. Este conector de entrada está conectado diretamente às entradas analógicas do *codec* de áudio. O *codec* é o responsável por codificar o áudio analógico e por meio de uma *Serial Audio Interface* (SAI) o áudio em formato digital é disponibilizado para processamento no microcontrolador.

A SAI é uma interface integrada ao *kit* de desenvolvimento que fornece comunicação com dispositivos de áudio externos como amplificadores, ADCs, DACs, *codecs* e processadores de áudio. A interface SAI disponibiliza os protocolos de comunicação I^2S *Standard*, I^2S MSB, I^2S LSB-*Justified*, multiplexação por divisão de tempo (TDM) e modulação de código de pulso (PCM).

O microcontrolador tem a tarefa de processar o áudio e disponibilizar uma GUI para que o usuário possa interagir com o EQ. No microcontrolador se faz uso do *Real Time Operating System* (RTOS) Keil RTX, com duas tarefas, *Display* e *Audio*, permitindo que a interação com a tela não prejudique o processamento e reprodução do áudio. A tarefa *Audio* possui uma prioridade de execução maior que a tarefa *Display* e nela será implementado o conjunto de filtros que compõem o EQ. Por fim o áudio é enviado novamente ao *codec*, convertido em forma analógica e então, enviado ao conector P2 de saída para que seja reproduzido em um dispositivo analógico.

3.1 Materiais

3.1.1 Microcontrolador

Será utilizado o *kit* de desenvolvimento STM32F746G Discovery, desenvolvido pela STMicroelectronics, tecnologia ARM CORTEX M7 32-bits. O microcontrolador STM32F746NGH6 contido no *kit* de desenvolvimento possui quatro canais de comunicação I^2C, seis SPI, três I^2S, duas SAI, frequência máxima de operação do processador de 216 MHz, display colorido LCD-TFT com 4.3 polegadas, 480x272 pixels e *touch screen* capacitivo, 1MB de memória *Flash*, 340KB de memória RAM, unidade de ponto flutuante, saída de alimentação para aplicações externas de 3.3 V ou 5 V e um codec de áudio WM8994ECS/R (STMICROELECTRONICS, 2017).

O codec de áudio WM8994ECS/R desenvolvido pela CIRRUS possui 4 DACs e 2 ADCs conectados à interface SAI do microcontrolador. A comunicação entre o codec e o microcontrolador é feita através do barramento I^2C compartilhado com o módulo de câmera do kit (CIRRUS LOGIC, 2016).

3.1.2 Codec WM8994ECS/R

O circuito integrado WM8994 é um *codec* de áudio projetado para ser usado em *smartphones* e dispositivos com requisitos multimídia. O *codec* possui um amplificador classe D e AB selecionáveis por *software*, além de um amplificador para fones de ouvido (CIRRUS LOGIC, 2016). Algumas características do *codec* são (CIRRUS LOGIC, 2016):

- Suporta até 24 *bits* de resolução;
- Suporte a taxas de amostragem de 8 kHz a 96 kHz;
- 4 canais DAC e 2 canais ADC;
- Interface para microfone;
- Amplificador estéreo para alto-falantes classe D e AB (2x2W);
- Suporte aos protocolos: I^2S, DSP, MSB-*first left* e MSB-*first right*.

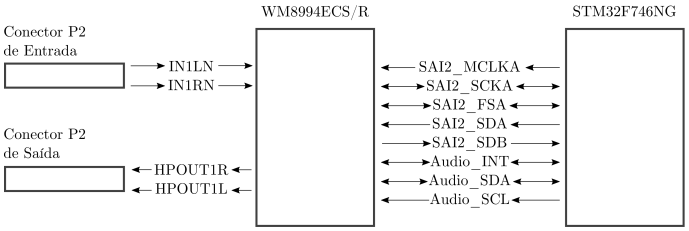


Figura 12 – Diagrama de comunicação do *codec* com o microcontrolador.

Fonte: STMicroelectronics (2017).

A Figura 12 mostra o diagrama de comunicação entre o *codec* de áudio WM8994ECS/R com o microcontrolador STM32F746NG.

3.1.3 Keil RTX

O RTOS Keil RTX implementa a API CMSIS-RTOS, que é uma interface genérica de sistema operacional de tempo real para dispositivos baseados nos processadores Cortex-M (KEIL, 2017). O RTOS Keil RTX possui (KEIL, 2017):

- 3 opções de escalonamento: *Round-Robin*, preemptivo e colaborativo;
- Operação de tempo real de alta velocidade com baixas latências de interrupção;
- Número ilimitado para criação de tarefas com 256 níveis de prioridade;
- Número ilimitado para criação de semáforos, *mutex* e temporizadores.

3.1.4 Linguagem de Programação

A implementação do software será feita em linguagem C. A linguagem C permite a manipulação de *bits*, *bytes* e endereços (SCHILDT, 1996). É uma linguagem portável, permitindo que o mesmo código seja implementado em outros microcontroladores.

A linguagem C possui controle de fluxo e estrutura de dados e um rico conjunto de operadores (KERNIGHAN; RITCHIE, 1988). A linguagem C proporciona as construções de controle de fluxo necessárias para um programa ser bem estruturado: tomadas de decisão (*if - else*), seleção de um conjunto de valores (*switch*), *loops* com testes de parada (*while - for*) e saída forçada de *loops* (*break*). Funções podem retornar valores básicos de tipos de dados, estruturas, uniões e ponteiros (KERNIGHAN; RITCHIE, 1988). Uma variável podem ser interna a uma função, externa mas conhecida somente na origem ou visível para todo o programa.

Pela linguagem C ser uma linguagem compilada e não interpretada, é possível obter um melhor desempenho do código executável. Além disso, o *kit* de desenvolvimento STM32F746G Discovery tem as bibliotecas do seu microcontrolador implementadas na linguagem C. Também as *Integrated Development Environment* (IDE) disponíveis para desenvolvimento das aplicações, possuem seus compiladores em C.

3.2 Métodos

A implementação do equalizador será dividida em 4 etapas, sendo elas: projeto e implementação dos filtros, comunicação entre o microcontrolador e o codec, implementação da interface gráfica e, validação experimental.

Na primeira etapa de desenvolvimento, baseando-se nas 10 frequências centrais com suas respectivas largura de banda, definidas segundo a norma ISO-266, serão projetados os filtros de tempo contínuo. Será utilizada a aproximação Butterworth para os filtros analógicos do tipo passa-banda. Apesar dessa aproximação não fornecer uma fase linear como Bessel e uma rápida transição como Chebyshev, a aproximação de Butterworth fornece uma banda passante plana, entretanto, na aproximação Butterworth pode haver cancelamento de fase. Com as especificações de cada um dos 10 filtros analógicos e definindo-se a taxa de amostragem, será possível obter os filtros digitais.

Os filtros digitais serão do tipo resposta ao impulso de duração infinita (IIR), pois é possível obter filtros de ordem reduzida em cada uma das 10 bandas de atuação. O mesmo não vale para os filtros do tipo resposta ao impulso de duração finita (FIR), que nas bandas de atuação de baixa frequência geram filtros de ordem elevada, o que pode comprometer a implementação em microcontroladores. Com os filtros digitais do tipo IIR, será criado uma

estrutura de programação padrão para os filtros a partir do que é mostrado em Zölzer et al. (2011), com a saída do filtro IIR dependente da entrada, de amostras passadas da entrada e de amostras passadas da saída.

Na sequência, é desenvolvida a etapa de comunicação entre o microcontrolador e o codec. O codec é um circuito integrado que converte sinais analógicos em sinais digitais e sinais digitais em sinais analógicos. Mais especificamente, um codec de áudio transforma um valor de tensão em uma palavra digital e converte uma palavra digital em um valor de tensão. É definido, por limitação do codec, que a resolução das conversões ADC e DAC serão de 24 *bits*. O codec de áudio disponibiliza um controle de interface de tempo em *Inter-integrated Circuit* (I^2C) e *Serial Peripheral Interface* (SPI) e duas interfaces digitais de áudio para comunicação com processadores externos. Cada interface digital de áudio suporta quatro formatos de dados de áudio: *Left justified*, *Right justified*, *Inter-IC Sound* (I^2S) e *DSP mode*. Para aquisição, comunicação e reprodução do áudio, será utilizado o *driver STM32746G Discovery Audio*, disponibilizada pela STMicroelectronics, fabricante do *kit* de desenvolvimento.

Com a estrutura de aquisição, processamento e reconstrução dos sinais implementada, é iniciado o processo de desenvolvimento da interface gráfica. A interface será composta de: 10 botões do tipo *slider* simulando os equalizadores analógicos, nos quais é possível atuar em cada um dos ganhos; um *slider* para controle do balanço da saída de som, atuando nos ganhos dos canais direito e esquerdo; um *slider* para controle do volume; e, um último controle, para ligar e desligar o processamento do áudio. A Figura 13 sugere um possível *design* para o EQ gráfico.

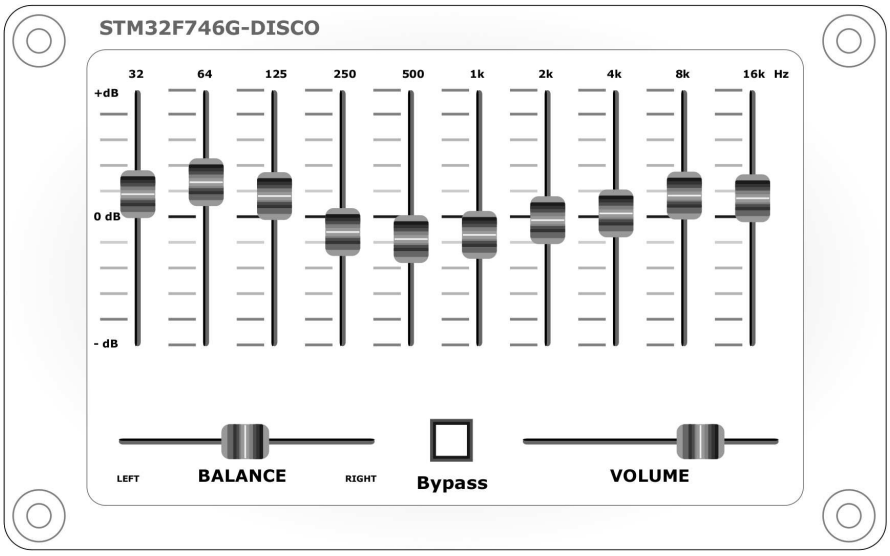


Figura 13 – Esboço para interface gráfica do equalizador.

Fonte: Autoria própria.

No bloco DSP se encontra toda a fase de processamento do áudio digital. É nessa etapa, por meio da interface gráfica, que os ajustes efetuados pelo usuário são processados.

4 DESENVOLVIMENTO

Este capítulo apresenta alguns aspectos da implementação do sistema proposto. Inicialmente é descrito o projeto e implementação dos filtros. Por fim é apresentado o desenvolvimento da captura, processamento e reprodução de áudio e desenvolvimento da interface de usuário.

4.1 Projeto dos Filtros Digitais

O projeto dos filtros digitais IIR foi executado baseado na Transformada Bilinear com *Prewarping* e na norma ISO 266, os dois métodos foram descritos anteriormente. Primeiramente, foi preciso projetar os filtros em tempo contínuo. Os filtros são de segunda ordem e possuem seletividade passa-banda.

Como demonstração do que foi projetado, utilizando os dados estabelecidos nas Tabelas 1 e 2 em um filtro passa-banda com frequência central $f_c = 1000Hz$, frequência de corte inferior $f_{c1} = 710Hz$, frequência de corte superior $f_{c2} = 1420Hz$ e taxa de amostragem T de 48 kHz é possível obter a equação para o filtro digital com as características desejadas. Os passos a seguir descrevem o procedimento para o projeto do filtro digital.

O projeto se inicia a partir da equação de primeira ordem de um filtro passa-baixas:

$$A(s) = \frac{A_o}{1 + a_i s}, \tag{4.1}$$

sendo $a_i = 1$, a Equação 4.1 fica:

$$A(s) = \frac{A_o}{1 + s}. \tag{4.2}$$

Para que o filtro digital seja do tipo passa-banda deve haver a substituição $s = \frac{s^2+\omega_0^2}{s \cdot W}$ na Equação 4.2, o que resulta em:

$$A(s) = \frac{W \cdot s}{s^2 + W \cdot s + \omega_0^2}. \tag{4.3}$$

Na Equação 4.3 a grandeza W é definida por $W = \omega_{ah} - \omega_{al}$ e a grandeza ω_0 é definida por $\omega_0 = \sqrt{\omega_{al} \cdot \omega_{ah}}$, na qual $\omega_{al} = \frac{2 \cdot \tan(\pi \cdot f_{c1} \cdot T)}{T}$ e $\omega_{ah} = \frac{2 \cdot \tan(\pi \cdot f_{c2} \cdot T)}{T}$.

Para o filtro com frequência central $f_c = 1000Hz$, frequência de corte inferior $f_{c1} = 710Hz$, frequência de corte superior $f_{c2} = 1420Hz$ e taxa de amostragem de 48 kHz as grandezas ω_0 e W valem, respectivamente, 6158 rad/sec e 4483 rad/sec. Substituindo esses valores na Equação 4.3, tem-se:

$$A(s) = \frac{4483 \cdot s}{s^2 + 4483 \cdot s + 6158^2}. \tag{4.4}$$

BP	GANHO	b0	b1	b2	a0	a1	a2
32 Hz	0.001437827246	1	0	-1	1	-1.997107744	0.9971243739
64 Hz	0.00287153176	1	0	-1	1	-1.994190812	0.9942569137
125 Hz	0.005791367032	1	0	-1	1	-1.988151908	0.9884172678
250 Hz	0.01151642669	1	0	-1	1	-1.975902915	0.9769671559
500 Hz	0.02271109633	1	0	-1	1	-1.950357795	0.9545778036
1 kHz	0.04443644732	1	0	-1	1	-1.894631386	0.9111270905
2 kHz	0.08526040614	1	0	-1	1	-1.766451597	0.8294791579
4 kHz	0.1582833678	1	0	-1	1	-1.453446746	0.6834332347
8 kHz	0.280515343	1	0	-1	1	-0.6794729233	0.4389693141
16 kHz	0.4790437818	1	0	-1	1	0.8675424457	0.04191241786

Tabela 3 – Tabela de coeficientes para um EQ de 1 oitava.

Aplicando na Equação 4.4, a transformada bilinear, descrita pelas Equações 2.10 e 2.11, é obtido:

$$A(z) = \frac{-0.0444z^{-2} + 0.0444}{1 - 1.8954 \cdot z^{-1} + 0.9111 \cdot z^{-2}}. \tag{4.5}$$

Para facilitar posteriormente o desenvolvimento do algoritmo dos filtros, foi isolado o ganho do numerador:

$$A(z) = \frac{0.0444(1 - z^2)}{1 - 1.8954 \cdot z + 0.9111 \cdot z^2}. \tag{4.6}$$

A Equação 4.6 é a equação que descreve um filtro IIR passa-banda Butterworth de segunda ordem com frequência central de 1000 Hz. Relacionando a Equação 2.12 com a Equação 4.6 é obtido os coeficientes b_0, b_1, b_2, a_0, a_1 e a_2.

A Tabela 3 mostra os coeficientes dos 10 filtros do equalizador.

4.2 Implementação dos Filtros Digitais

A implementação prática dos filtros digitais em linguagem de programação pode ser feita a partir de diferentes formas ou estruturas. Essas realizações são equivalentes matematicamente, mas possuem diferenças no desempenho quando implementadas (KUO; LEE; TIAN, 2006). Uma das possíveis realizações é a Forma Direta I. Considerando as equações de segunda ordem dos filtros, na forma genérica da Equação 2.12, o fluxo do diagrama de sinal pode ser representado como na Figura 14:

A Figura 14 pode ser interpretada como duas funções de transferência em cascata. A Forma Direta I necessita de 5 coeficientes e 4 variáveis de estado por estágio de segunda ordem.

Rearranjando a Figura 14 é possível chegar a uma nova realização, a Forma Direta II, como mostrado na Figura 15.

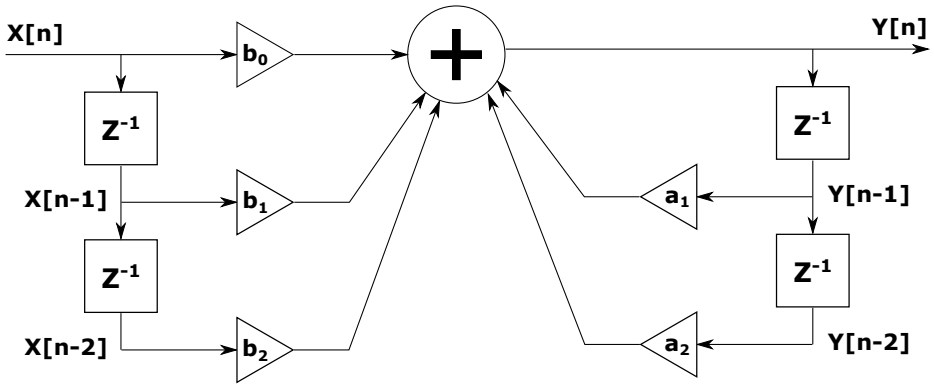


Figura 14 – Diagrama de blocos representando o fluxo de sinal de um filtro digital IIR.

Fonte: Baseado em Kuo, Lee e Tian (2006).

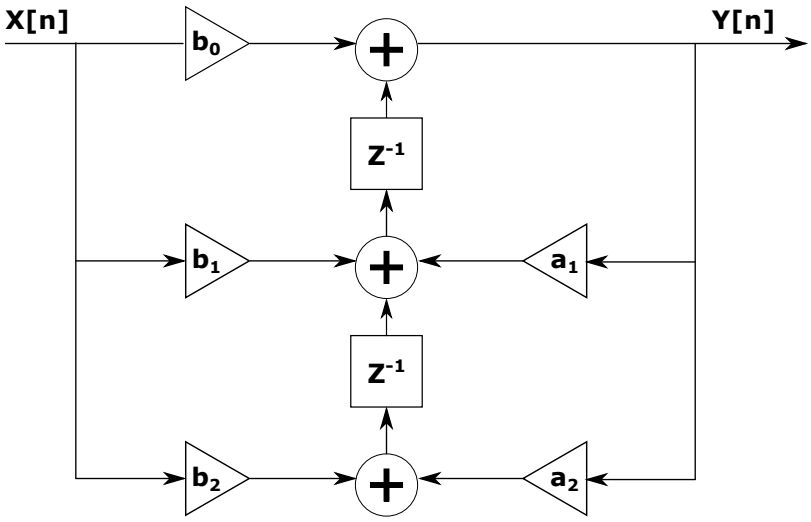


Figura 15 – Diagrama de blocos representando o fluxo de sinal de um filtro digital IIR.

Fonte: Baseado em Kuo, Lee e Tian (2006).

A realização Forma Direta II apresenta uma redução no uso de memória por conter apenas duas variáveis de estado por estágio de segunda ordem. A partir da Forma Direta II foi possível definir um algoritmo para processar uma amostra de áudio. O Código 4.1 mostra o algoritmo do filtro utilizado para processar uma amostra de áudio, na qual y é a saída, x é a entrada, b_i e a_i são os coeficientes do filtro e $d1$ e $d2$ são as amostras passadas.

É necessário, porém, que algumas modificações sejam feitas para que o algoritmo funcione de forma contínua, ou seja, para inúmeras amostras de áudio. O Código 4.2 mostra o algoritmo do filtro implementado no microcontrolador. Esse algoritmo foi implementado de forma a prever uma possível utilização em cascata, ou seja, filtros de ordem maior que 2. No Código 4.2, *NUMSTAGES* é um número inteiro que representa a ordem do filtro, *delay0CH1*,

delay1CH1, *delay2CH1* são vetores que armazenam os estados anteriores da saída, e, *coef-Filt* é um vetor que armazenas todos os coeficientes b_i e a_i e ganho dos filtros seguindo o padrão da Tabela 3.

Os índices de *coef-Filt*, correspondentes aos dados da Tabela 3 são acessados segundo a seguinte representação: *coef-Filt* $= (X \cdot$ *cf-num* $) + Y + (j * 7)$, onde X representa 0 para o filtro de 32 Hz, 1 para o filtro de 64 Hz, 2 para o filtro de 125 Hz, e assim sucessivamente, *cf-num* $= NUMSTAGES \cdot 7$, Y é a posição do coeficiente com relação na Tabela 3 e por fim, j é usado para prever uma possível utilização de filtros de ordem maior que 2. Para cada banda passante, há uma execução do algoritmo do filtro. Como o sistema é estéreo, há a execução de 20 filtros. A Figura 16 mostra o diagrama de 10 filtros em paralelo, respectivos a um canal, de modo que os ganhos *K1* a *K10*, são os ganhos setados pelo usuário no *display* do *kit* de desenvolvimento.

```
1 y[n] = b0*x[n] + d1;
2 d1   = b1*x[n] + a1*y[n] + d2;
3 d2   = b2*x[n] + a2*y[n];
```

Código 4.1 – Algoritmo genérico dos filtros.

```
1 outCH1[0] = vec_f[i];
2 for(j=0; j<NUMSTAGES; j++){
3         delay0CH1[0] = outCH1[0]*coef_Filt[(0*cf_num)+(j*7)] -
             delay1CH1[0]*coef_Filt[(0*cf_num)+5+(j*7)] - delay2CH1
             [0]*coef_Filt[(0*cf_num)+6+(j*7)];
4         outCH1[0] = delay0CH1[0] + delay2CH1[0]*coef_Filt[(0*cf_num)
             +3+(j*7)];
5         delay2CH1[0] = delay1CH1[0];
6         delay1CH1[0] = delay0CH1[0];
7 }
```

Código 4.2 – Algoritmo do filtro de 31 Hz implementado no microcontrolador.

4.3 Implementando o Equalizador

O equalizador proposto neste trabalho deve ser capaz de processar uma fonte contínua de áudio, e ainda oferecer ao usuário uma GUI para que haja interação com o sistema. Para que essa interação fosse possível e o áudio fosse processado em tempo real, foi utilizado o sistema operacional Keil RTX. Este sistema operacional em tempo real, Keil RTX, é o responsável por gerenciar os recursos de *hardware* do kit de desenvolvimento.

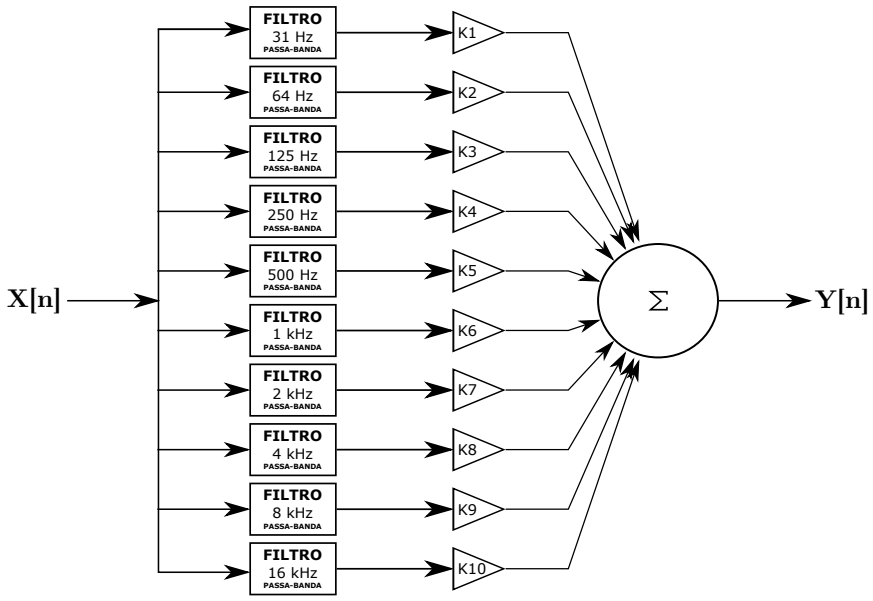


Figura 16 – Diagrama dos filtros em paralelo respectivos a um canal.

Fonte: Autoria própria.

Foi necessária a criação de duas tarefas, representadas pela Figura 17. A primeira, denominada Audio, foi responsável pelos filtros, configurar e inicializar os parâmetros de captura e reprodução do áudio. A segunda tarefa, Display, é responsável por atualizar e oferecer interação com o ganho dos filtros e alguns parâmetros da reprodução de áudio.

Figura 17 – Tarefas criadas para desenvolvimento do equalizador.

Fonte: Autoria própria.

Para instalar uma tarefa neste RTOS, foi utilizado as funções *osThreadDef* e *osThreadCreate*. A função *osThreadDef* cria um conjunto de parâmetros para uma tarefa. Os parâmetro são: nome, prioridade, número de instâncias da tarefa e tamanho da pilha. O Código 4.3 mostra detalhes desses parâmetros e a definição das duas tarefas utilizadas.

```
1  /*
2   * param.   name          nome da tarefa.
3   * param.   priority      prioridade inicial da tarefa.
4   * param.   instances     número de possíveis instâncias da tarefa.
5   * param.   stacksz       tamanho (em bytes) da pilha.
6   */
7  //osThreadDef(name, priority, instances, stacksz);
8
9  //Definição da tarefa responsável pela configuração, reprodução e
       processamento do áudio
10 osThreadDef (Audio, osPriorityAboveNormal, 1, 16000);
11
12 //Definição da tarefa responsável pela interação com o usuário
       através do display
13 osThreadDef (Display, osPriorityIdle, 1, 2000);
```

Código 4.3 – Definição dos parâmetros das tarefas utilizadas.

Para que as tarefas possam ser utilizáveis é necessário chamar a função *osThreadCreate*. Essa função cria a tarefa, adiciona esta tarefa em uma lista de tarefas ativas e muda o estado da tarefa para pronto. O Código 4.4 mostra o algoritmo utilizado para criar as duas tarefas.

```
1  /*
2   * Cria a tarefa, adiciona esta tarefa em uma lista de tarefas
       ativas e seta o estado da tarefa para pronto.
3   * param[in]  thread_def    definicação da tarefa referenciada com
       osThread.
4   * param[in]  argument      ponteiro como argumento de inialização.
5   * retorna o ID da tarefa ou NULL em caso de erro.
6   */
7  //osThreadCreate (const osThreadDef_t *thread_def, void *argument);
8
9  //Definição da tarefa responsável pela interação com o usuário
       através do display
10 osThreadCreate(osThread(Display), NULL);
```

```
11 //Criação da tarefa responsável pela configuração, reprodução e
      processamento do áudio
12 osThreadCreate(osThread(Audio), NULL);
```

Código 4.4 – Criação e ativação das tarefas.

4.3.1 Tarefa Áudio

Para que os dados de áudio pudessem ser lidos e processados em um fluxo contínuo foi criada a tarefa Audio. Esta tarefa é responsável pelo processamento do áudio, além de configurar os dispositivos utilizados para captura e reprodução de áudio.

Para configurar os dispositivos de *hardware* necessários para capturar e reproduzir o áudio, foi utilizado o *driver STM32746G Discovery Audio*, disponibilizada pela STMicroelectronics, fabricante do *kit* de desenvolvimento. O Código 4.5 mostra a configuração utilizada no EQ e descreve brevemente os parâmetros de cada função.

```
1  /* Inicializa a gravação e reprodução do áudio
2   * param  InputDevice   seleciona a entrada de áudio: microfone ou
       linha
3   * param  OutputDevice  seleciona a saída de áudio: fone de ouvido,
       falantes ou ambos
4   * param  AudioFreq     frequência de áudio a ser utilizada pelo SAI
5   * param  BitRes        resolução em bits
6   * param  Channel       número de canais
7   */
8  BSP_AUDIO_IN_OUT_Init(InputDevice, OutputDevice, AudioFreq, BitRes,
      Channel);
9
10 /* Inicializa os buffers */
11 memset((uint16_t*)AUDIO_BUFFER_IN, 0, AUDIO_BLOCK_SIZE*2);
12 memset((uint16_t*)AUDIO_BUFFER_OUT, 0, AUDIO_BLOCK_SIZE*2);
13 audio_rec_buffer_state = BUFFER_OFFSET_NONE;
14
15 /* Inicia a gravação */
16 BSP_AUDIO_IN_Record((uint16_t*)AUDIO_BUFFER_IN, AUDIO_BLOCK_SIZE);
17
18 /* Inicia a reprodução */
19 BSP_AUDIO_OUT_SetAudioFrameSlot(CODEC_AUDIOFRAME_SLOT_02);
20 BSP_AUDIO_OUT_Play((uint16_t*)AUDIO_BUFFER_OUT, AUDIO_BLOCK_SIZE *
      2);
```

```
21
22 /* Seta um volume inicial */
23 BSP_AUDIO_OUT_SetVolume(VOL_INICIAL);
```

Código 4.5 – Configuração para aquisição de reprodução de áudio.

A função *BSP-AUDIO-IN-OUT-Init* configura todo o *hardware* necessário para a aplicação de áudio (codec, I^2C, GPIO e DMA). Essa função retorna AUDIO-OK se a configuração foi correta e um valor diferente de AUDIO-OK para uma configuração incorreta. Os parâmetros utilizados foram: entrada de linha, saída pelo fone de ouvido, taxa de amostragem de 48 kHz, 16 bits de resolução (limitação do *driver STM32746G Discovery Audio*), 2 canais (estéreo). As funções *BSP-AUDIO-IN-Record* e *BSP-AUDIO-OUT-Play* respectivamente, inicializam a captura e reprodução do áudio.

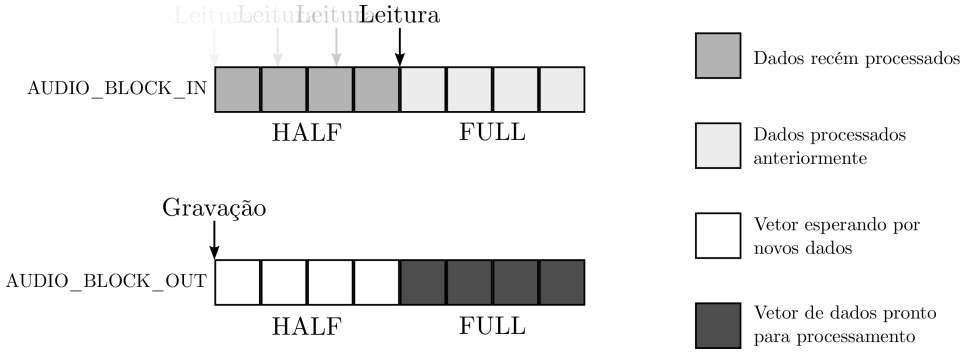


Figura 18 – Áudio carregado na primeira metade da lista AUDIO-BLOCK-IN sendo copiado para a primeira metade da lista AUDIO-BLOCK-OUT.

Fonte: Autoria própria.

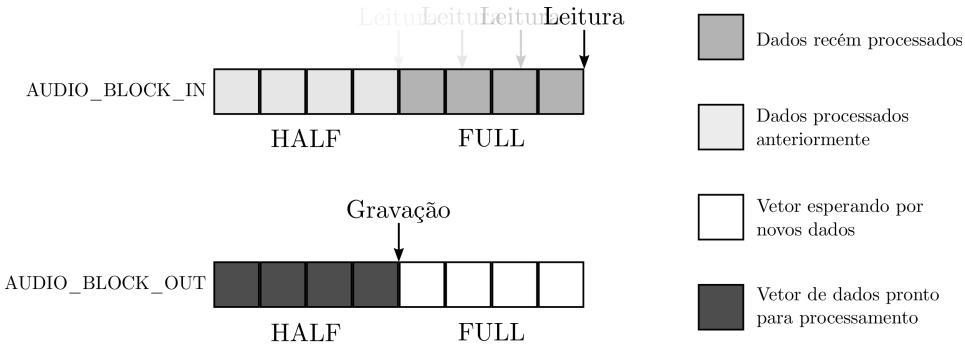


Figura 19 – Áudio carregado na segunda metade da lista AUDIO-BLOCK-IN sendo copiado para a segunda metade da lista AUDIO-BLOCK-OUT.

Fonte: Autoria própria.

A configuração apresentada pelo Código 4.5 inicializa a reprodução e aquisição em blocos de áudio. Como representado na Figuras 18 e 19 existem dois vetores de tamanho X, AUDIO-

BLOCK-IN e AUDIO-BLOCK-OUT, onde HALF indica que metade do vetor está preenchido e Full representa que todo o vetor foi preenchido. Os vetores podem ser de qualquer tamanho, porém, quanto maior o vetor, maior o atraso. Por exemplo, para os vetores AUDIO-BLOCK-IN e AUDIO-BLOCK-OUT com tamanho $X = 512$ cada, 256 posições serão utilizadas pelo canal esquerdo e 256 posições são utilizadas pelo canal direito, sendo a primeira amostra para o canal esquerdo, a segunda para o direito, terceira para o esquerdo, e assim sucessivamente. Sabendo que a taxa de amostragem é 48 kHz, o atraso para um bloco de dados de tamanho 512 é $\frac{512/2}{48000} = 5,33ms$. Inicialmente esses dois vetores são preenchidos com valores zero. Após a chamada das funções de captura e reprodução do áudio, inicia-se em um laço infinito, a troca de valores entre vetores.

Já no laço infinito, o áudio é capturado e armazenado na lista AUDIO-BLOCK-IN. Como mostrado na Figura 18, após a preenchimento de metade do vetor AUDIO-BLOCK-IN, esses dados são copiados para a primeira metade do vetor AUDIO-BLOCK-OUT, a partir do qual é reproduzido. Da mesma maneira acontece na segunda metade das listas, Figura 19. Após o áudio ser armazenado na segunda metade de AUDIO-BLOCK-IN ele é então copiado para e AUDIO-BLOCK-OUT e reproduzido.

A função que executa os filtros digitais foi inserida entre a troca de dados de AUDIO-BLOCK-IN e AUDIO-BLOCK-OUT. Esses dois vetores, AUDIO-BLOCK-IN e AUDIO-BLOCK-OUT, são divididos em duas partes, primeira metade e segunda metade, para que enquanto uma das partes é preenchida pela entrada de dados de áudio, a outra metade possa ser processada. Os filtros são executados como descrito e representados anteriormente no Código 4.2 e na Figura 16.

4.3.2 Tarefa Display

Para que houvesse a interação entre o usuário e o EQ, foi implementada em uma tarefa chamada Display, uma GUI, na qual por meio de um *display com touchscreen* capacitivo, alguns parâmetros do EQ podem ser ajustados. Por definição do projeto, Figura 13, os parâmetros que podem ser alterados pelo usuário, são: volume do sistema, balanço, ganho de cada um dos filtros e ainda, um botão que torna a saída de áudio *bypass*. A tarefa Display está descrita no Código 4.6.

```
1 void Display(const void* argument) {
2         //Variáveis para controle
3         uint8_t flag_vol = VOL_INICIAL, flag_volANT = VOL_INICIAL;
4 
5         //Função que cria e inicializa o ambiente gráfico
6         sliders_creator();
```

```
7
8           while (1) {
9                   HAL_GPIO_WritePin(GPIOG, GPIO_PIN_7, GPIO_PIN_SET);
10                  /* Se o dispositivo touch está habilitado */
11                  #ifdef RTE_Graphics_Touchscreen
12                          /* Executa a função touch */
13                          if(osOK)GUI_TOUCH_Exec();
14                  #endif
15
16                  flag_vol = vet_VOL[SLIDER_GetValue(hVOL)];
17                  if(flag_vol != flag_volANT){
18                          flag_volANT = flag_vol;
19                          BSP_AUDIO_OUT_SetVolume(vet_VOL[
                               SLIDER_GetValue(hVOL)]);}
20
21
22                  slid_AP[0]=SLIDER_GetValue(hSli1);
23                  //...
24                  slid_AP[9]=SLIDER_GetValue(hSli10);
25
26                  if(SLIDER_GetValue(hBALANCE)<50){
27                          LR_VOL[1] = ((float)SLIDER_GetValue(hBALANCE
                               )*2)/100;
28                          LR_VOL[0] = 1;}
29                  else if(SLIDER_GetValue(hBALANCE)>50){
30                          LR_VOL[0] = ((100-(float)SLIDER_GetValue(
                               hBALANCE))*2)/100;
31                          LR_VOL[1] = 1;}
32                  else if(SLIDER_GetValue(hBALANCE)==50){
33                          LR_VOL[1] = 1;
34                          LR_VOL[0] = 1;}
35
36                  /*Flag usada para bypass*/
37                  CHBX_Vl=CHECKBOX_GetState(hCB);
38
39                  /* Callback para atualização da tela */
40                  GUI_Exec();
41
```

```
42                  osDelay(50);
43          }
44  }
```

Código 4.6 – Tarefa Display.

A tarefa Display ao ser executada pela primeira vez, cria e inicializa os objetos do ambiente gráfico por meio da chamada da função *sliders-creator*, Código 4.7.

```
1  void sliders_creator(void){
2          /* Criação dos sliders e checkbox */
3          hVOL = SLIDER_CreateEx(290,230,180,20,0,WM_CF_SHOW,0,0);
4          hBALANCE = SLIDER_CreateEx(10,230,180,20,0,WM_CF_SHOW,0,0);
5          hSli1 = SLIDER_CreateEx(posX,30,35,180,0,WM_CF_SHOW,
               SLIDER_CF_VERTICAL,0);
6          //...
7          hSli10 = SLIDER_CreateEx(posX,30,35,180,0,WM_CF_SHOW,
               SLIDER_CF_VERTICAL,0);
8          hCB = CHECKBOX_CreateEx(230,232,20,20,0,WM_CF_SHOW,
               SLIDER_CF_VERTICAL,0);
9
10         /* Alteração da faixa de valores */
11         SLIDER_SetRange(hSli10, 0, 60);
12         //...
13         SLIDER_SetRange(hSli1, 0, 60);
14         SLIDER_SetRange(hVOL, 0, 32);
15
16         /* Inicializando valores dos sliders */
17         SLIDER_SetValue(hVOL, 9);
18         SLIDER_SetValue(hBALANCE, 50);
19         SLIDER_SetValue(hSli10, 30);
20         //...
21         SLIDER_SetValue(hSli1, 30);
22 }
```

Código 4.7 – Criação e inicialização dos objetos.

Após ser executada pela primeira vez, a tarefa Display segue uma rotina de verificação de toque na tela, atualização dos valores dos objetos e atualização da tela. Para a criação dos objetos foi utilizado a biblioteca gráfica STemWin, baseada na SEGGER emWin. Essa biblioteca é uma parceria entre a STMicroelectronics e a SEGGER Microcontroller GmbH. A biblioteca permite a criação de GUI com qualquer STM32, monitores e controladores

LCD-TFT, aproveitando as acelerações de *hardware* quando disponível. A biblioteca fornece decodificação JPG, GIF e PNG, *widgets* como caixas de seleção, botões, calendários, medidores, fornece também um servidor VNC, e uma ferramenta chamada GUIBuilder que cria GUIs com simples arrastar e soltar.

5 RESULTADOS

Para validar o equalizador foi realizado dois conjuntos de testes. O primeiro conjunto de testes teve como objetivo, verificar o tempo de processamento das tarefas Audio e Display, para analisar se o áudio era processado e reproduzido sem atrasos. O segundo conjunto de testes comparou a resposta de cada filtro implementado com a resposta teórica.

5.1 Verificação da Disponibilidade do Sistema

A Figura 20, mostra o primeiro teste realizado. Nesta figura, o gráfico de linhas amarelas representa o intervalo de aquisição e processamento de metade da lista de dados (HALF de AUDIO-BLOCK-IN e AUDIO-BLOCK-OUT, descritos anteriormente) e o gráfico de linhas azuis representa a aquisição e processamento da segunda metade da lista de dados (FULL de AUDIO-BLOCK-IN e AUDIO-BLOCK-OUT). O tamanho do bloco/vetor de dados utilizado foi de 2048, 1024 para para a primeira metade e 1024 para a segunda metade, sendo que dessas 1024 posições, 512 são para o canal esquerdo e 512 para o canal direito, o que leva ao período $Ta = \frac{2*1024/2}{48000} = 21,34ms$ condizente com o período mostrado na Figura 20.

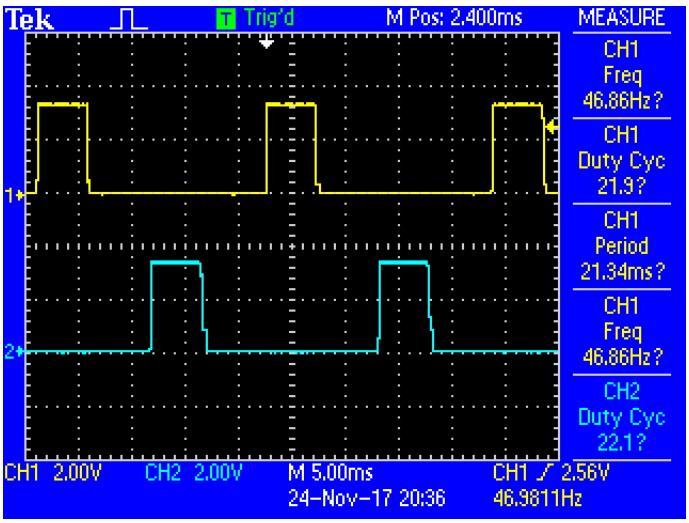


Figura 20 – Intervalos de tempo ocupados pelos processamentos HALF e FULL dos blocos de áudio.

Fonte: Autoria própria.

A Figura 21 une em um gráfico as etapas HALF e FULL. Com essa imagem é possível observar que as etapas de aquisição, processamento e preparo para reprodução do áudio

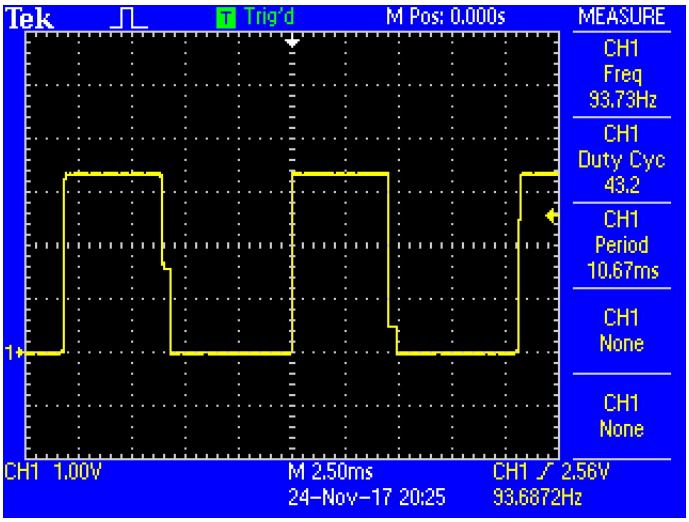


Figura 21 – Intervalos de tempo unidos ocupados pelos processamentos HALF e FULL dos blocos de áudio.

Fonte: Autoria própria.

ocupam menos de 50% do tempo disponível de processamento. O tamanho do vetor de dados utilizado é de 2048, 1024 para para a primeira metade e 1024 para a segunda metade, sendo que dessas 1024 posições, 512 são para o canal esquerdo e 512 para o canal direito, o que leva ao período $Ta = \frac{1024/2}{48000} = 10,67ms$ condizente com o período mostrado na Figura 20.

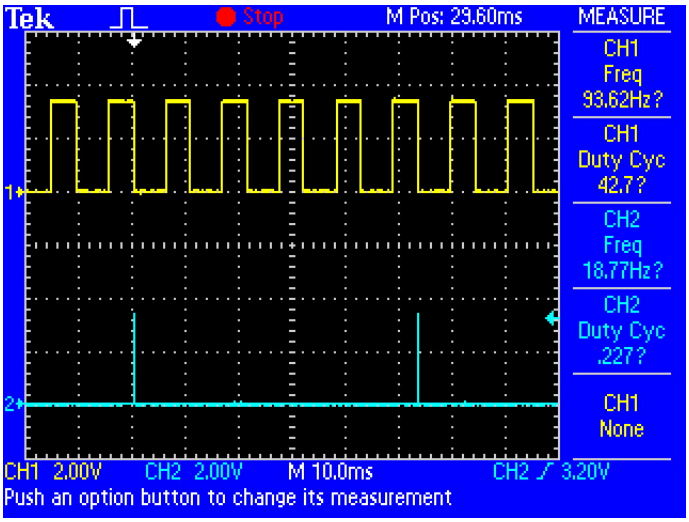


Figura 22 – Intervalo de tempo da tarefa Audio no gráfico superior e intervalo de tempo da tarefa Display no gráfico inferior.

Fonte: Autoria própria.

Ainda relacionado ao primeiro conjunto de testes, a Figura 22 mostra o intervalo de tempo

entre as rotinas das tarefas Audio e Display. O gráfico superior é relativo à tarefa Audio e o gráfico inferior à tarefa Display. Com esse gráfico é possível observar que a tarefa Display tem uma frequência de execução em torno de 20 vezes por segundo, parâmetro este definido no corpo da tarefa Display. Também é possível observar que a tarefa Display tem um impacto relativamente pequeno no sistema enquanto não há interação com o *display*, o que faz com que ela também não cause sobreposições de áudio.

A Figura 23 mostra o aumento no tempo de execução da tarefa Display quando há interação com o *display touchscreen*. Em ambas as imagens, o gráfico superior, em amarelo, é relativo à tarefa Audio e o gráfico inferior, em azul à tarefa Display.

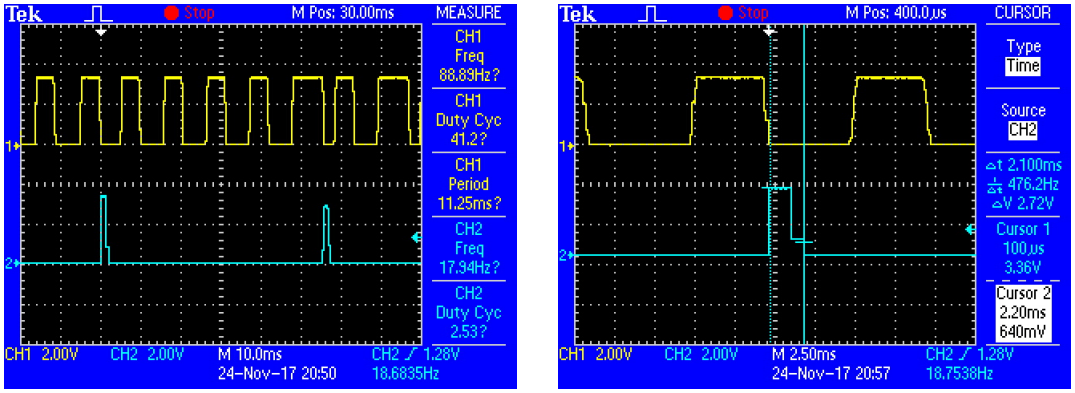


Figura 23 – Relação de tempo entre a tarefa Audio e a tarefa Display quando há interação com o *display touchscreen*.

Fonte: Autoria própria.

5.2 Resposta do Equalizador

O segundo conjunto de teste comparou a resposta dos filtros digitais IIR Butterworth passa-banda de segunda ordem com a resposta teórica. Para a realização deste teste foi utilizado um gerador de funções e um osciloscópio. Foi feita uma varredura na faixa audível, de 20 Hz a 20 kHz utilizando o gerador de funções. O osciloscópio foi utilizado para verificar o ganho de saída em cada ponto de medição.

A Figura 24 compara a resposta do equalizador com todos os filtros configurados em 0 dB de ganho (curva preta) com a resposta teórica (curva vermelha). É possível observar que a resposta real é muito próxima à resposta teórica em toda a faixa de valores.

O segundo teste envolvendo a resposta do equalizador foi feito utilizando uma configuração aleatória para o ganho de cada um dos filtros. Essa configuração dos ganhos está representada

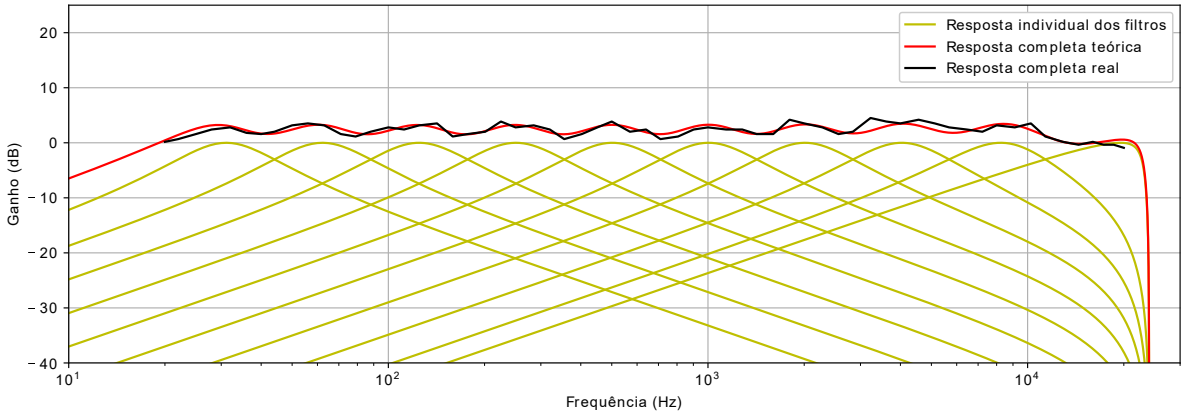


Figura 24 – Resposta do equalizador com ganho dos filtros configurados em 0 dB.

Hz	**31**	**64**	**125**	**250**	**500**	**1k**	**2k**	**4k**	**8k**	**16k**
dB	+5.0	+14.0	+10.0	+4.0	-3.0	-13.0	-6.0	+0.0	+2.0	+7.0

Tabela 4 – Tabela dos ganhos aleatórios para verificação da resposta do equalizador.

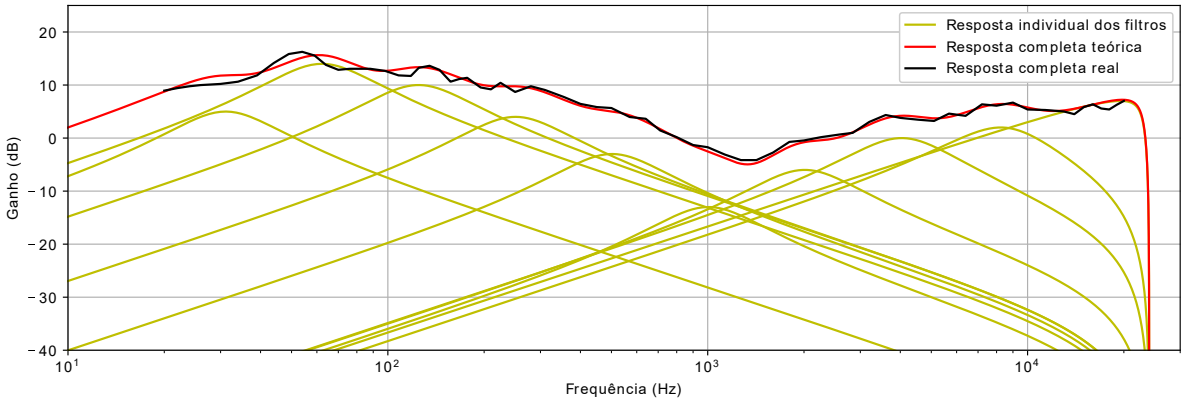


Figura 25 – Resposta do equalizador com ganho dos filtros configurados em um padrão estabelecido.

na Tabela 4 para o resultado obtido na Figura 25 e na Tabela 5 para o resultado obtido na Figura 26.

As Figuras 25 e 26 comparam a resposta do equalizador (curva preta) com o ganho dos filtros configurados aleatoriamente, com a resposta teórica (curva vermelha). Mais uma vez, é possível observar um bom resultado em toda a faixa de teste, com valores muito próximos dos ideais nos dois casos de teste.

Hz	**31**	**64**	**125**	**250**	**500**	**1k**	**2k**	**4k**	**8k**	**16k**
dB	+0.0	+5.0	+10.0	+15.0	+10.0	+5.0	+0.0	-5.0	-10.0	-15.0

Tabela 5 – Tabela dos ganhos aleatórios para verificação da resposta do equalizador.

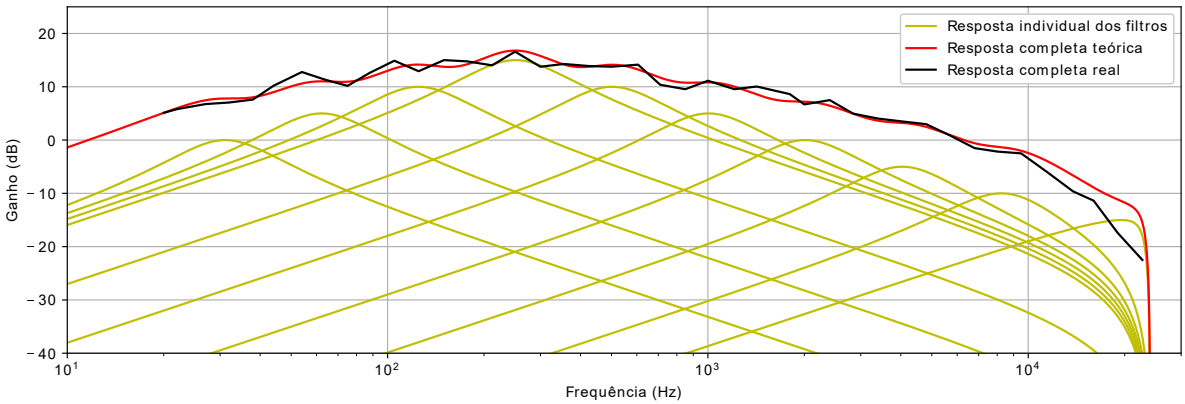


Figura 26 – Resposta do equalizador com ganho dos filtros configurados em um padrão estabelecido.

6 CONCLUSÃO

Com os resultados obtidos neste trabalho foi possível validar a implementação de um equalizador gráfico digital de bandas com uma interface gráfica sensível ao toque, utilizando o *kit* de desenvolvimento STM32F746G Discovery.

Os resultados obtidos nos teste de disponibilidade do sistema foram muito satisfatórios, comparando com o tempo ocioso após cada rotina de aquisição, processamento e preparo de reprodução do áudio. Conseguiu-se provar a implementação de um equalizador gráfico digital no *kit* proposto. Um resultado positivo foi a constatação da possibilidade do EQ funcionar em modo estéreo.

Os testes de resposta do equalizador forneceram dados compatíveis com o teórico, validando o projeto de filtros digitais. Há alguns picos de diferença entre o resultado real e o teórico, no entanto, essa diferença pode ter sido acentuada no momento da leitura dos dados da tela do osciloscópio, pois o sinal de entrada era baixo, 40 mVpp. Foi utilizado esse sinal de baixa amplitude para que não houvesse distorções nas respostas dos filtros configurados com ganhos elevados.

O *kit* de desenvolvimento fornece um amplo e bem documentado material de apoio, com vários exemplos, bibliotecas e *drivers*. A biblioteca utilizada para criação da GUI fornece vários recursos interessantes para desenvolvedores em diversas áreas, o que possibilitou a criação de uma interface limpa e que remetesse a um equalizador tradicional.

O *driver* (STM32746G Discovery Audio) utilizado para captura e reprodução do áudio, é um *driver* completo e com funções de fáceis configurações. Ele disponibiliza todos os recursos necessários para aplicações em áudio, como configuração dos *hardwares* necessários e funções como *Play*, *Stop* e *Pause*. Uma possível melhoria em trabalhos futuros seria a tentativa de reconfigurar o *driver* (STM32746G Discovery Audio) para que pudesse aumentar a resolução de 16 *bits* para 24 *bits*.

Referências

ACOUSTICS Preferred frequencies. 1997. Disponível em: ⟨https://www.iso.org/obp/ui/#iso:std:iso:266:ed-2:v1:en⟩. Acesso em: 5 mai. 2017.

ADDING decibels of one-third octave bands to level of one octave band and vice versa. 2017. Disponível em: ⟨http://www.sengpielaudio.com/calculator-octave.htm⟩. Acesso em: 7 dez. 2017.

BALLOU, G. *Handbook For Sound Engineers*. 4. ed. Burlington: Elsevier, 2008.

BRAGA, N. C. Eletrônica analógica e digital - sistemas de numeração. *Instituto Newton C. Braga*, 2014. Disponível em: ⟨http://www.newtoncbraga.com.br/index.php/eletronica-digital/90-licao-1-eletronica-analogica-e-digital-sistemas-de-numeracao⟩. Acesso em: 21 mar. 2017.

CARTER, B.; MANCINI, R. *Op Amps For Everyone*. 3. ed. Burlington, MA, USA: Elsevier, 2004.

CIRRUS LOGIC. *Multi-channel Audio Hub CODEC for Smartphones*. [S.l.], 2016. Rev. 4.5.

GOBIND, D. *Principles of Active Network Synthesis and Design*. 1. ed. [S.l.]: John Wiley and Sons, 1976.

HAYKIN, S. S.; VEEN, B. V. *Sinais e sistemas*. 1. ed. [S.l.]: Bookman, 2001.

HUOTILAINEN, M.; NÄÄTÄNEN, R. Auditory perception and early brain development. *Encyclopedia on Early Childhood Development*, Finlândia, jun. 2010. Disponível em: ⟨http://www.child-encyclopedia.com/brain/according-experts/auditory-perception-and-early-brain-development⟩. Acesso em: 25 mar. 2017.

KEIL. *CMSIS-RTOS Documentation*. [S.l.], 2017. Version 1.03.

KERNIGHAN, B. W.; RITCHIE, D. M. *The C programming Language*. [S.l.]: Prentice-Hall Englewood Cliffs, 1988.

KESTER, W. *Mixed Signal and DSP Design Techniques*. 1. ed. USA: Elsevier, 2000.

KUO, S. M.; LEE, B. H.; TIAN, W. *Real-Time Digital Signal Processing*: Implementations and applications. 2. ed. England: John Wyiley and Sons Ltd, 2006. 646 p.

LATHI, B. P. *Sinais e Sistemas Lineares*. 2. ed. [S.l.]: Bookman, 2006.

LEMOS, M. Bens de consumo: o foco no produto. *Fundação Estudar*, jun. 2014. Disponível em: ⟨https://www.napratica.org.br/bens-de-consumo-o-foco-no-produto/⟩. Acesso em: 20 mar. 2017.

MONTGOMERY, J. Implementing a 10-band stereo equalizer on the dsp56311 evm board. 2005. Disponível em: ⟨http://www.nxp.com/assets/documents/data/en/application-notes/AN2110.pdf⟩. Acesso em: 5 mai. 2017.

OPPENHEIM, A. V.; WILLSKY, A. S.; NAWAB, S. H. Signals and systems. Prentice-Hall Englewood Cliffs, 1983.

PAARMANN, L. D. *Analog Filters: A Signal Processing Perspective*. 1. ed. New York: Kluwer Academic Publisher, 2001.

SCHILDT, H. *C Completo e Total*. 3. ed. São Paulo: Makron Books, 1996.

SMITH, S. W. *The Scientist and Engineer's Guito to Digital Signal Processing*. 2. ed. San Diego, California: California Technical Publishing, 1999. 650 p.

SONY. In: CORPORATE Info. Tokyo: Sony GLOBAL, 2010. Disponível em: ⟨https://www.sony.net/SonyInfo/CorporateInfo/History/history.html⟩. Acesso em: 24 mai. 2017.

STMICROELECTRONICS. *Discovery kit for STM32F7 Series with STM32F746NG MCU - User manual*. [S.l.], 2017. Rev. 4.

TAN, L. *Digital Signal Processing*: Fundamentals and applications. 1. ed. Burlington, USA: Elsevier, 2008. 837 p.

THEDE, L. *Pratical Analog and Digital Filter Design*. 1. ed. Norwood, Mass, USA: Artec House, 2004.

ZÖLZER, U. *Digital Audio Signal Processing*. 2. ed. Chippenham, England: John Wyiley and Sons Ltd, 2008. 320 p.

ZÖLZER, U. et al. *DAFX: Digital Audio Effects*. 2. ed. Hamburg: John Wiley and Sons Ltd, 2011.