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

MATHEUS BAUER

SISTEMA PARA EQUALIZAÇÃO PARAMÉTRICA DE ÁUDIO EM TEMPO REAL

TRABALHO DE CONCLUSÃO DE CURSO 2

PATO BRANCO
2018

MATHEUS BAUER

SISTEMA PARA EQUALIZAÇÃO PARAMÉTRICA DE ÁUDIO EM TEMPO REAL

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. Gustavo Weber Denardin

PATO BRANCO

2018

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

UTFPR

TERMO DE APROVAÇÃO

Às 10 horas e 20 minutos do dia 11 de dezembro de 2018, na sala V107, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, reuniu-se a banca examinadora composta pelos professores Gustavo Weber Denardin (orientador), Diogo Ribeiro Vargas e Everton Luiz de Aguiar para avaliar o trabalho de conclusão de curso com o título **Sistema para equalização paramétrica de áudio em tempo real**, do aluno **Matheus Bauer** matrícula 01298003, 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.

_____
Prof. Gustavo Weber Denardin
Orientador (UTFPR)

_____
Prof. Diogo Ribeiro Vargas
(UTFPR)

_____
Prof. Everton Luiz de Aguiar
(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

BAUER, Matheus. Sistema para equalização paramétrica de áudio em tempo real. 2018. 54f. Monografia (Trabalho de Conclusão de Curso 2) - Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2018.

Os equalizadores digitais estão presentes em quase todos os sistemas multimídia atualmente. O presente trabalho proporciona o desenvolvimento de um equalizador paramétrico digital de cinco bandas composto por filtros do tipo IIR de oitava ordem implementados no *kit* de desenvolvimento STM32F746G Discovery. O equalizador foi implementado com duas abordagens, em *software* foi utilizada a biblioteca DSP do CMSIS, com filtros na estrutura forma direta II transposta, e em *hardware* fez-se o uso do codec WM8994. Um equalizador é basicamente um conjunto de filtros com a função de atenuar características indesejadas no sinal de áudio. Foi desenvolvida uma interface gráfica para possibilitar o ajuste da frequência central, largura de banda, ganho do equalizador e volume geral em um *display* LCD com tela *touchscreen*.

Palavras-chave: Equalizador, Interface gráfica, Filtros digitais.

ABSTRACT

BAUER, Matheus. Real-time parametric audio equalization system. 2018. 55f. Monografia (Trabalho de Conclusão de Curso 2) - Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2018.

Digital equalizers are present in almost all multimedia systems today. The present work proportion the development of a five-band digital parametric equalizer composed of eighth-order type IIR filters implemented in the STM32F746G Discovery development kit. The equalizer was implemented with two approaches, in software was used the DSP library of the CMSIS, with filters in the structure direct form II transposed, and in hardware was made use of the codec WM8994. An equalizer is basically a set of filters with the function of attenuating undesired characteristics in the audio signal. A graphical interface has been developed to allow adjustment of the center frequency, bandwidth, equalizer gain and overall volume on an LCD display with touch screen.

Keywords: Equalizer, Graphic interface, Digital filters.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE SIGLAS

ADC	*Analog to Digital Converter* (Conversor Analógico para Digital)
AIF	*Digital Audio Interface* (Interface de Áudio Digital)
BSP	*Board Support Package* (Pacote de Suporte da Placa)
CC	Corrente Contínua
CD	*Compact Disc* (Disco Compacto)
CPU	*Central Processing Unit* (Unidade Central de Processamento)
DAC	*Digital to Analog Converter* (Conversor Digital para Analógico)
DSP	*Digital Signal Processor* (Processador de Sinal Digital)
DVD	*Digital Video Disc* (Disco de Vídeo Digital)
FIR	*Finite Impulse Response* (Resposta Finita ao Impulso)
GUI	*Graphical User Interface* (Interface Gráfica do Usuário)
I^2C	*Inter-integrated Circuit* (Circuito Inter-Integrado)
I^2S	*Inter-IC Sound* (Circuito Inter-Integrado do Som)
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
IIR	*Infinite Impulse Response* (Resposta Infinita ao Impulso)
ISSCC	*International Solid-State Circuits Conference* (Conferência Internacional de Circuitos em Estado Solido)
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)
SOS	*Second Order Sections* (Seções de Segunda Ordem)

SUMÁRIO

1 INTRODUÇÃO

Nesse capítulo são apresentadas as considerações iniciais, que definem o escopo e o contexto do projeto, bem como os objetivos e a justificativa do desenvolvimento do projeto.

1.1 CONSIDERAÇÕES INICIAIS

A percepção auditiva do ser humano é muito importante em diversos aspectos para o próprio indivíduo e para a sua interação com outras pessoas e com mundo que o cerca. O que a pessoa ouve causa diversos benefícios para a saúde, como alívio de dores, melhora da memória e até mesmo um estímulo para a prática de atividade física. A música, por exemplo, pode provocar sensações de bem-estar ao ouvinte. Essa sensação é devida ao ritmo e à harmonia da música (NETO, 2018).

A fim de que um sinal de áudio, uma música, por exemplo, se torne mais agradável ao ouvinte é necessário que esse sinal seja submetido a um processamento. Uma das formas mais comum de processamento de um sinal de áudio é a filtragem em relação às suas frequências, o qual é, muitas vezes, chamado de equalização (SILVA, 2005).

Na década de 80 ocorreu a transição dos circuitos analógicos para os circuitos digitais. Devido ao aumento considerável do uso de *Compact Disc* (CD) e *Digital Video Disc* (DVD), entre outros sistemas digitais, nesse período fez-se necessário o desenvolvimento de novas tecnologias para implementar essas aplicações. Nesse contexto, a empresa Texas Instruments apresentou, em 1982, o primeiro Processador Digital de Sinais (do inglês *Digital Signal Processor* - DSP), o TMS32010, na *International Solid-State Circuits Conference* (ISSCC) (BRAGA, 2014).

Os DSPs são microprocessadores especializados em processamento digital de sinais, em aplicações que requerem alto desempenho de processamento (BODANESE, 2008), como aplicações de áudio e vídeo operando em tempo real. Com o sinal digitalizado, pode-se executar diversos algoritmos, dentre eles podemos citar os específicos para, eliminar ruídos, comprimir ou descomprimir o sinal, adicionar efeitos sonoros, equalizar o sinal, entre outras aplicações (BODANESE, 2008). É importante ressaltar que a tecnologia de áudio e vídeo se encontra em franca evolução, visto que atualmente o padrão de áudio em *streaming* utiliza taxas de amostragem de 96 kHz a 24 bits e o padrão de vídeo opera com 3840 × 2160 *pixels* a uma taxa de atualização de 120 Hz. Para acompanhar esse cenário, os processadores digitais

de sinais também têm evoluído constantemente.

O equalizador é uma das ferramentas mais utilizada no processo de tratamento de áudio, pois seu objetivo é modificar o espectro de frequência de um determinado sinal, tornando a resposta do sistema mais adequada a um determinado propósito. Por exemplo, para se obter um som mais grave deve-se amplificar as faixas de frequências mais baixas e para sinais mais agudos, amplificar as faixas de frequências mais altas (ROSA, 2012). O mesmo pode ser aplicado em qualquer outra frequência dentro do intervalo de 20 Hz a 20 kHz, que é a faixa percebida pela audição humana (ROSA, 2012).

Os equalizadores podem ser gráficos ou paramétricos. A principal diferença entre eles é que os equalizadores gráficos possuem botões deslizantes para dar ganho ou atenuar as frequências selecionadas (GRYTZ, 2003), possuindo, geralmente, 31 bandas de equalização. Há um botão específico para cada banda, tornando possível modificar várias bandas ao mesmo tempo. Já o equalizador paramétrico possui de três a dez filtros, todos eles com a capacidade de alterar simultaneamente três parâmetros específicos: amplitude, frequência central e largura de banda. Ressalta-se que o equalizador gráfico atua somente na amplitude, pois a frequência central e a largura de banca de cada canal é fixo.

Neste trabalho é apresentado o desenvolvimento de um equalizador paramétrico digital de três bandas, além de um filtro digital passa-baixas e um filtro digital passa-altas. A implementação foi realizada utilizando um *kit* de microcontrolador com unidade de ponto flutuante e instruções específicas de processamento digital de sinais. O dispositivo possui uma interface gráfica por meio da qual é possível alterar, em tempo real, os parâmetros de equalização.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Implementar um equalizador paramétrico digital de cinco bandas para dois canais de áudio, utilizando um *kit* de desenvolvimento de sistemas microcontrolados com *Central Processing Unit* (CPU) de 32 bits, codec de áudio e *display* sensível ao toque.

1.2.2 Objetivos Específicos

- Configurar o sistema de aquisição de áudio do *kit* para a taxa de aquisição de dados utilizada;
- Determinar as faixas de frequências utilizadas em cada banda;
- Definir a técnica de equalização digital a ser utilizada;
- Desenvolver uma interface gráfica para ajuste dos parâmetros do equalizador;
- Implementar os filtros digitais parametrizáveis no microcontrolador;
- Testar e avaliar os resultados do equalizador paramétrico.

1.3 JUSTIFICATIVA

Para que um som se torne mais agradável esse sinal deve passar por um processamento, pois um áudio com ruído ou distorções pode causar desconforto ao ouvinte. O equalizador paramétrico é considerado um processamento adequado para corrigir um sinal com as características citadas, pois permite alterar três parâmetros essenciais para o tratamento do sinal, tais como: ganho, frequência central e largura de banda. Além de que um equalizador possibilita alterar as configurações dos filtros em tempo real.

O sistema desenvolvido, como resultado da realização desse trabalho, consiste em um equalizador paramétrico digital de cinco bandas. Sua implementação foi realizada a partir de um *kit* de um microcontrolador que tem um custo em torno de oito vezes menor se comparado a um equalizador paramétrico digital comercial.

2 REFERENCIAL TEÓRICO

Neste capítulo são abordados os conceitos necessários para a compreensão do desenvolvimento do projeto. Inicialmente é tratado sobre processo de conversão do sinal analógico para digital, posteriormente sobre filtros, seguido de equalizadores, apresentando a distinção entre equalizadores gráficos e paramétricos.

2.1 REPRESENTAÇÃO DISCRETA DE UM SINAL

Os sinais podem ser classificados pela maneira que eles são definidos como uma função do tempo. Um sinal $x(t)$ é um sinal de tempo contínuo se ele estiver definido para todos os instantes de tempo t . Esses sinais são gerados pela conversão de uma forma de onda física, como uma onda acústica, em um sinal elétrico. Ao utilizar um microfone, por exemplo, essa conversão é efetuada por meio de um transdutor, que converte as variações de pressão sonora em variações de tensão elétrica (HAYKIN; VEEN, 2001).

Um sinal de áudio analógico pode ser representado por uma tensão elétrica. Quando esse sinal é aplicado a um alto-falante, ocorrem vibrações em seu cone flexível que produzem ondas de pressão que são interpretadas como som (BARBOSA, 1999).

Um sinal $x[n]\,x[n]$ é um sinal de tempo discreto se estiver definido somente em instantes de tempo específicos, em que x pertence ao conjunto dos números inteiros. Esse sinal $x[n]$ pode ser construído por meio da amostragem de um sinal continuo $x(t)$ em uma taxa uniforme (DINIZ; SILVA; NETTO, 2014).

Ao passar pelo processo de amostragem e quantização, um sinal de tensão passa a ser representado por uma sequência de números digitalizados. Os sinais digitais são formados por um conjunto de *bits*. Cada *bit* pode assumir apenas dois valores, ou nível alto (1), ou nível baixo (0). Portanto, quanto maior a quantidade de *bits*, melhor é a precisão da conversão desse sinal. Para os sinais analógicos serem processados digitalmente eles devem antes passar por um processo de conversão conhecido com amostragem e quantização (BARBOSA, 1999).

2.1.1 Processo de Amostragem

No processo de amostragem a informação de amplitude do sinal contínuo é obtida

em instantes de tempo específicos, indicados pelo intervalo ou período de amostragem T_a. Cada uma dessas informações é uma amostra da sequência discreta resultante, como pode ser visto na Figura 1b. O inverso do período de amostragem, dado por $F_a = \frac{1}{T_a}$, é conhecido por frequência de amostragem (NALON, 2009).

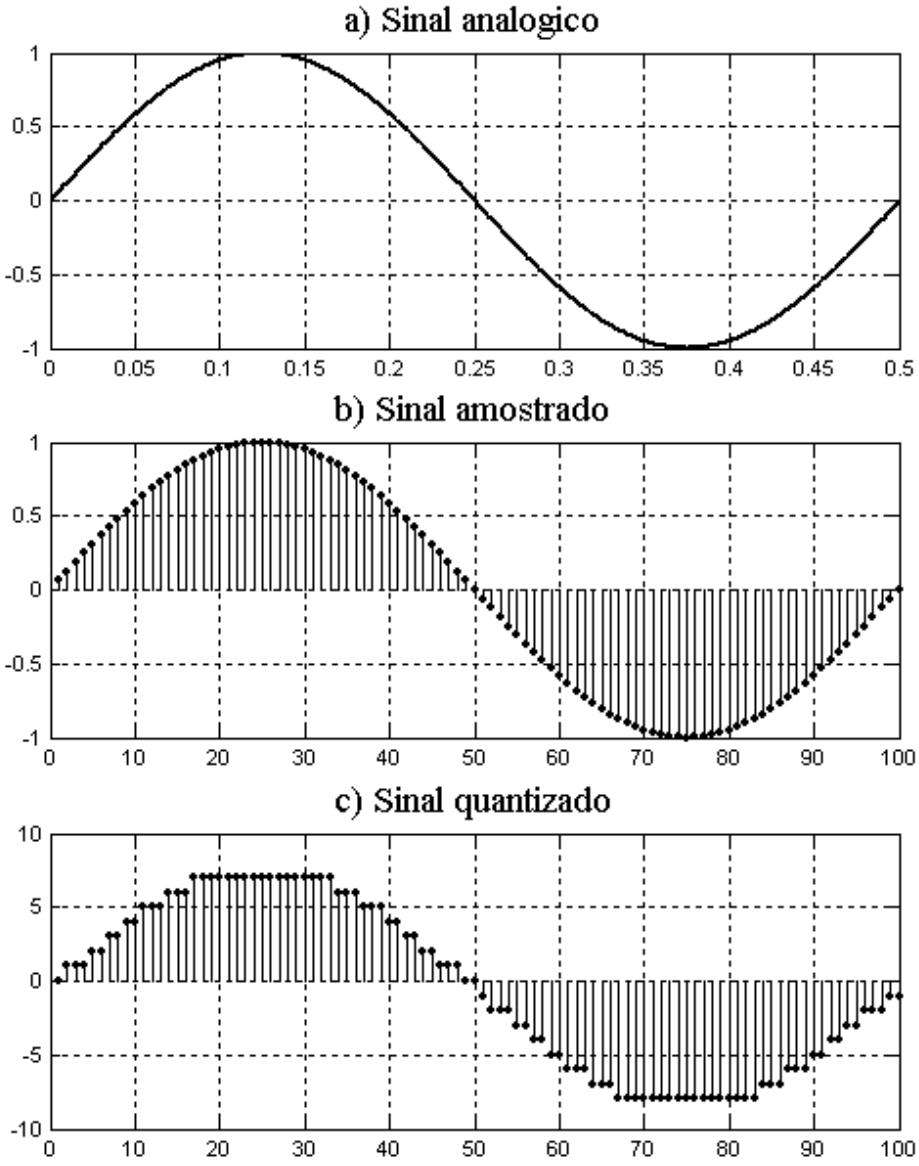


Figura 1 - Sinal original (a) Sinal amostrado (b) Sinal quantizado (c)

Fonte: Herrera (2004, p. 29).

No processo de conversão do sinal é desejável preservar o máximo das informações do sinal original (Figura 1a). Portanto, no processo de amostragem a taxa com que as informações do sinal discreto serão obtidas, ou seja, a taxa com que esse sinal será amostrado, deve ser escolhida baseada em um critério, o Teorema de Nyquist (BODANESE, 2008).

De acordo com o Teorema de Nyquist, um sinal de banda finita pode ser reconstruído, se for amostrado uniformemente a uma taxa mínima que corresponde a duas vezes o valor da maior componente de frequência do sinal (LATHI, 2007). Como na equação:

$$f_s \geq 2f_{sinal}. \tag{1}$$

Por exemplo, em um sinal original com maior componente de frequência de 2 kHz, a taxa de amostragem mínima deve ser de 4 kHz. No processo de gravação de CD de áudio a taxa de amostragem utilizada é de 44,1 kHz, mais que o dobro da máxima frequência que o ouvido humano pode captar, cerca de 20 kHz. Já na gravação de áudio para DVD é utilizada a frequência de amostragem de 48 kHz (BODANESE, 2008).

Fisicamente a amostragem é feita por meio de um conversor analógico digital. O processo para a obtenção das amostras de um sinal contínuo é dado pela Equação 2 em que x_c é o sinal que está sendo amostrado, n é a variável de tempo discreto e T_a é o período de amostragem (DINIZ; SILVA; NETTO, 2014).

$$x[n] = x_c(nT_a). \tag{2}$$

2.1.2 Processo de Quantização

Segundo Nalon (2009), é impossível fazer uma representação com precisão completa para números arbitrários através de números inteiros. Isso significa que a amplitude do sinal também deve ser discretizada. Essa discretização de amplitude é chamada quantização.

O processo de quantização deve acontecer porque a amplitude de cada amostra do sinal discreto é um número real, que para ser representado necessita de uma precisão infinita, mas pode ser representado por um número inteiro (NALON, 2009).

A quantização representa a resolução de uma amostra, se V_{max} é o valor máximo representável e n é o número de *bits* usado por amostra, o passo de quantização q é dado pela equação:

$$q = \frac{V_{max}}{2^n}. \tag{3}$$

A operação de quantização não é reversível, ou seja, após uma amostra ser quantizada é impossível obter sua amplitude original, pois mapeia um número infinito

(contínuo) de valores de entrada num número finito (discreto) de valores de saída. Os valores que não podem ser representados com a precisão requerida são arredondados. A diferença entre a amostra original e a amostra quantizada é chamada de erro de quantização (NALON, 2009). Esse erro pode ser minimizado com o aumento da resolução do número de *bits* usados. Para obter uma resolução equivalente à de um sistema de CD de áudio, são necessários 16 *bits*, o que significa que haverá 65536 combinações numéricas possíveis (BODANESE, 2008).

A Figura 2 apresenta um sinal de -8 V a +8 V, já compensado para passar pelo processo de conversão, ou seja, somado uma componente CC de 8 V ao sinal, com isso não possui valores negativos em sua amplitude. Para a representação desse sinal com uma precisão de 3 *bits*, conclui-se, aplicando os dados na Equação (3), que o passo de quantização é de 2 V. Portanto, é impossível representar o valor de 1 V, sendo necessário o arredondamento.

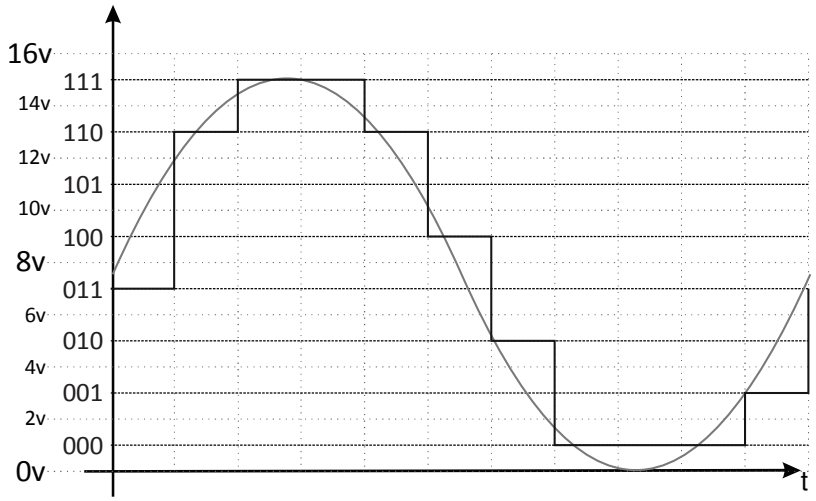


Figura 2 - Erro processo quantização

Fonte: Adaptado de Bodanese (2008, p. 11).

2.2 FILTROS

Uma das representações mais usuais de um filtro é um sistema linear, que modifica um sinal, removendo partes não desejadas do sinal, como o ruído, ou extraindo partes úteis do sinal, como determinadas componentes de frequência, de forma a obter um sinal de saída com as características desejadas para uma dada aplicação (ROSA, 2012).

Segundo Nalon (2009), um filtro é um sistema que seleciona características específicas e desejadas de um sinal. Se o filtro é linear, as características selecionadas, ou seja, filtradas, consistem nas componentes em frequência do sinal de entrada.

O processo de filtragem pode ser definido matematicamente, no domínio do tempo, como uma convolução. A integral de convolução entre duas funções no domínio do tempo t, $x_1(t)$ e $x_2(t)$, é simbolicamente representada por $x_1(t)^* x_2(t)$ e é definida por

$$x_1(t)^* x_2(t) \equiv \int_{-\infty}^{\infty} x_1(\tau) x_2(t-\tau) d\tau. \tag{4}$$

Através da Transformada de Laplace, a convolução dos sinais no domínio do tempo pode ser efetuada pelo produto dos sinais no domínio transformado, como apresentado na Equação 5.

$$\mathcal{L}[x_1(t)^* x_2(t)] = X_1(s).\ X_2(s). \tag{5}$$

Sendo $Y(s)$ a resposta do processo de filtragem, $H(s)$ a função de transferência do filtro e $X(s)$ o sinal a ser filtrado, tem-se:

$$Y(s) = H(s)X(s). \tag{5}$$

Um filtro pode ser definido a partir de uma função de transferência, relacionando à saída e a entrada do sistema do domínio da frequência.

$$H(s) = \frac{Y(s)}{X(s)}. \tag{7}$$

2.2.1 Tipos de Filtros

A utilidade dos filtros é selecionar de um sinal de entrada as frequências que devem estar presentes no sinal de saída. O tipo de filtro varia de acordo com a aceitação ou rejeição da faixa de frequência selecionada. A banda que contém as frequências aceitas é chamada banda de passagem e a que contém as frequências rejeitadas é denominada banda de rejeição (NALON, 2009).

Um filtro possui como resposta em frequência um gráfico do seu ganho de tensão *versus* frequência (MALVINO; BATES, 2007). As subseções a seguir apresentam a resposta

em frequência ideal de cada um dos seguintes filtros: passa-baixas, passa-altas, passa-faixa e rejeita-faixa.

2.2.1.1 Filtro Passa-Baixas

Os filtros que permitem a passagem das frequências abaixo da frequência de corte $F_c F_c$, e eliminam as componentes de frequências que estão acima desse limiar são chamados filtros passa-baixas. A Figura 3 representa esse filtro na sua forma ideal.

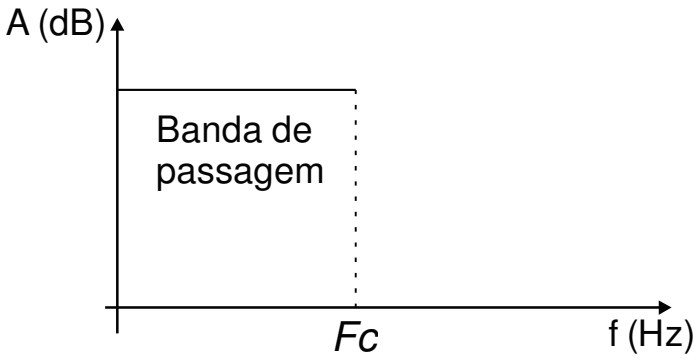


Figura 3 - Filtro Passa-Baixas ideal

Fonte: Adaptado de Malvino, Bates (2007, p. 223).

Nos equalizadores esse tipo de filtro permite a passagem das frequências mais baixas, atenuando ou eliminando as frequências mais altas.

2.2.1.2 Filtro Passa-Altas

O funcionamento desse filtro é similar ao filtro passa-baixas, porém atua atenuando as frequências que estão abaixo da frequência de corte F_c, permitindo a passagem das frequências que estão acima desse limiar. A Figura 4 representa um filtro passa-altas ideal.

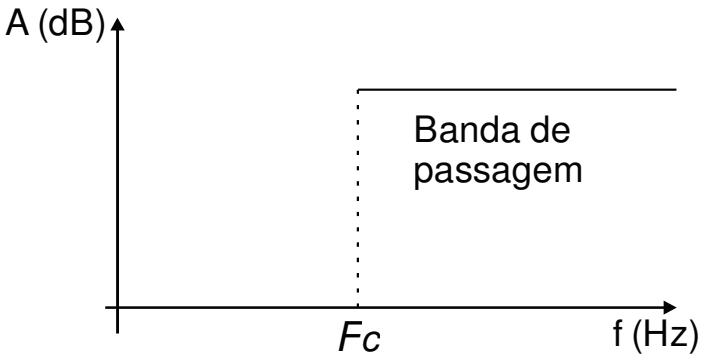


Figura 4 - Filtro Passa-Altas ideal

Fonte: Adadptado de Malvino, Bates (2007, p. 224).

Nos equalizadores esse tipo de filtro permite a passagem das frequências mais altas, atenuando ou eliminando as frequências mais baixas.

2.2.1.3 Filtro Passa-Faixa

São filtros que permitem a passagem das componentes de frequência dentro da largura da banda de passagem, definida por BW . As componentes de frequência fora dessa faixa são atenuadas ou eliminadas. A Figura 5 representa um filtro passa-faixa ideal.

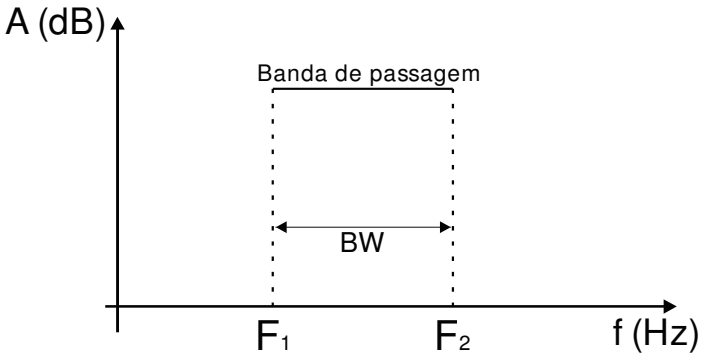


Figura 5 - Filtro Passa-Faixa ideal

Fonte: Adaptado de Malvino, Bates (2007, p. 224).

A largura de banda, BW , de um filtro passa-faixa é a diferença entre as frequências de 3dB de corte superior e inferior (MALVINO; BATES, 2007).

$$BW = f_2 - f_1. \tag{8}$$

A frequência central é simbolizada por f_0 f_0e é dada pela média geométrica entre as duas frequências de corte:

$$f_0 = \sqrt{f_1 f_2}. \quad (9)$$

O fator Q de um filtro passa-faixa é definido como a frequência central dividida pela largura de banda:

$$Q = \frac{f_0}{BW}. \quad (10)$$

Um exemplo de aplicação desse tipo de filtro ocorre quando é conhecida a faixa de frequência de um determinado instrumento musical de uma banda e com isso é possível atenuar os outros instrumentos e manter a faixa de frequência selecionada.

2.2.1.4 Filtro Rejeita-Faixa

Esse filtro trabalha de maneira similar ao passa faixa, porém ele impede a passagem das componentes de frequência dentro da faixa entre $F_{c1} F_{c1}$ e F_{c2}, que é o inicio e o fim da banda de rejeição. A Figura 6 representa um filtro rejeita-faixa ideal.

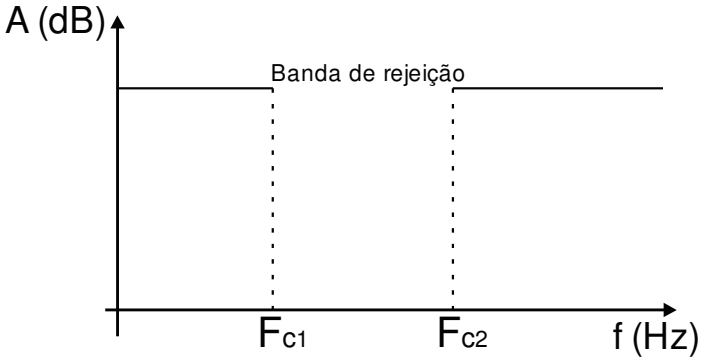


Figura 6 - Filtro Rejeita-Faixa ideal

Fonte: Adaptado de Malvino, Bates (2007, p. 225).

Como no exemplo anterior, sabendo a faixa de frequência de um determinado instrumento musical de uma banda, pode-se eliminar o som produzindo por esse instrumento.

2.2.2 Filtro Digital

Filtros que operam com sinais que passaram pelo processo de discretização, ou seja, operam em tempo discreto, são chamados de filtros digitais. Para operar com esse tipo de sinal é necessário o uso de conversores na entrada e na saída do sistema (ROSA, 2012) como mostrado na Figura 7.

A Figura 7 apresenta um diagrama de blocos que ilustra o funcionamento básico de um filtro digital. O bloco Conversor Analógico para Digital (ADC do inglês *Analog to Digital Converter*) é responsável pela conversão do sinal analógico para digital e o bloco Conversor Digital para Analógico (DAC do inglês *Digital to Analog Converter*) é responsável pela conversão do sinal digital para analógico. Já o bloco Processador é responsável pelo processamento do sinal, executando os algoritmos para filtragem do sinal.

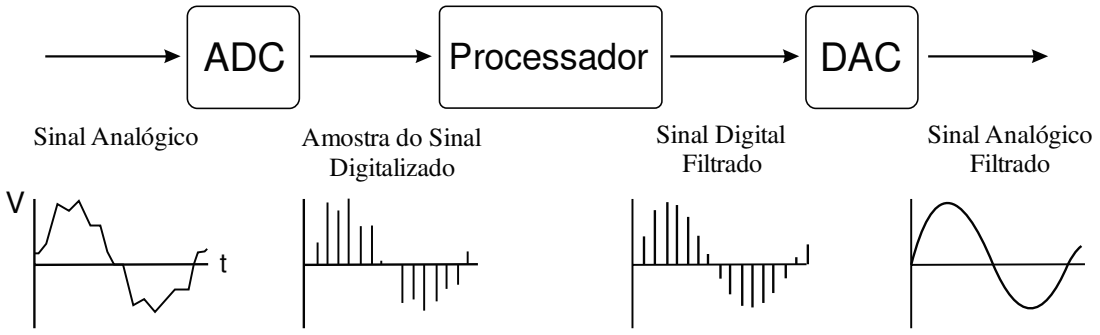


Figura 7 - Processo de Filtragem Digital

Fonte: Adaptado de Bodanese (2008, p. 14).

De acordo com Shenoi (2006), os filtros digitais possuem algumas vantagens em relação aos filtros analógicos:

- Um filtro digital é reprogramável, assim, as especificações do filtro podem ser alteradas sem a necessidade de alterar o circuito.
- Componentes dos filtros analógicos sofrem alterações em suas características por influência da temperatura, precisão dos componentes e envelhecimento. Esses fatores não influenciam nos filtros digitais.
- Alta precisão e confiabilidade. Aumentando a quantidade de *bits* para representar os coeficientes do filtro e do sinal de entrada, faz com que o filtro tenha uma melhor precisão.
- Os valores dos indutores, capacitores e parâmetros dos amplificadores operacionais usados nos filtros analógicos não possuem precisão elevada.

Porém, nos filtros digitais a representação desses componentes pode ser de alta precisão.

Em relação à resposta ao impulso, os filtros digitais são classificados em dois grupos. Se o sistema é implementado com equações de diferença que não contém termos recursivos, a saída não realimenta a entrada, então o filtro pode ser implementado diretamente por meio da convolução. Para ser implementável, a resposta ao impulso deve ser de duração finita. Esse tipo de filtro é conhecido por filtro com resposta ao impulso de duração finita (FIR do inglês *Finite-duration Impulse Response*) (NALON, 2009).

Já o outro tipo de filtro possui realimentação, ou seja, a saída do filtro depende em maior ou menor grau de suas amostras passadas. Filtros com essas características são conhecidos por filtros com resposta ao impulso de duração infinita (IIR do inglês *Infinite-duration Impulse Response*) (NALON, 2009).

2.2.2.1 Filtro Resposta Finita ao Impulso (FIR)

Os filtros de resposta finita ao impulso, também conhecido como filtros não recursivos, são os mais fáceis de serem implementados. Tais filtros não possuem realimentação, ou seja, as saídas do sistema dependem somente da entrada presente e de um número finito de amostras passadas (ROSA, 2012). Filtros FIR podem ser representados a partir da seguinte equação de diferença.

$$y[n] = \sum_{k=-\infty}^{\infty} b_k x[n-k]. \tag{11}$$

Em que $x[n]$ é o sinal de entrada deslocado a cada iteração do somatório em k unidades, $y[n]$ é o sinal de saída e b_k b_k são os coeficientes do filtro. A Equação 11 representa o processo de filtragem por meio da convolução do sinal de entrada pelos coeficientes do filtro, que são ao mesmo tempo a resposta ao impulso do sistema (ROSA, 2012).

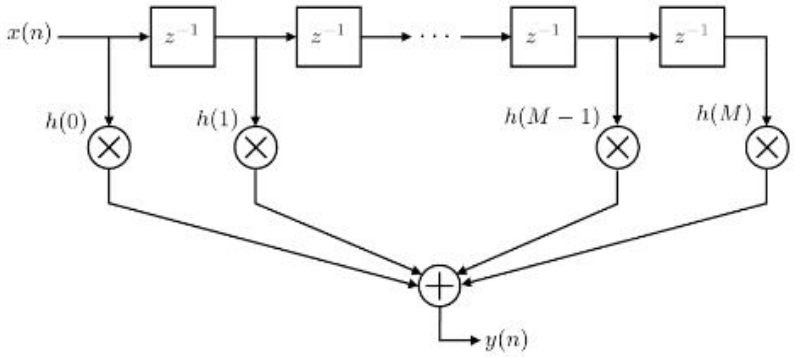


Figura 8 - Estrutura forma direta de um Filtro FIR
Fonte: Diniz, Silva e Netto (2014, p. 244).

A realização mais simples de um filtro FIR possui a estrutura apresentada na Figura 8, é chamada de realização na forma direta, pois os seus coeficientes, $h(n)h(n)$, são obtidos diretamente da função de transferência do filtro (DINIZ; SILVA; NETTO, 2014).

Quando um filtro FIR for de ordem elevada sua realização pode ser na forma de cascata de filtros FIR de ordem inferior a esse. Quando a ordem for par, a realização ocorre por meio de filtros FIR de segunda ordem. Se ímpar, a realização ocorre em cascata de filtros de segunda ordem e um filtro de primeira ordem (SHENOI, 2006) como apresentado na Figura 9.

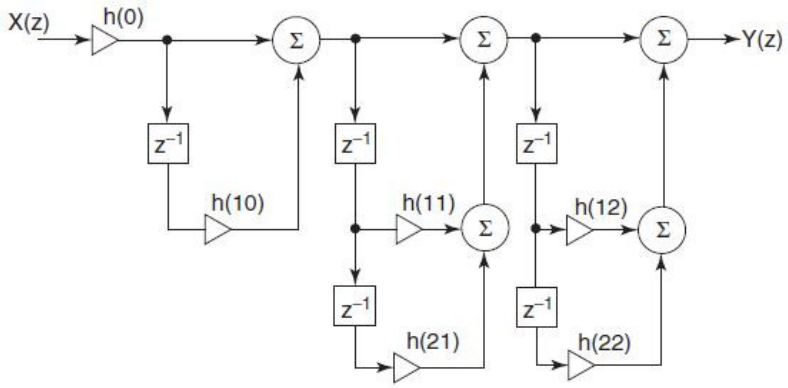


Figura 9 - Estrutura forma cascata de um Filtro FIR
Fonte: Shenoi (2006, p. 306).

O filtro FIR possui algumas desvantagens, dentre elas a necessidade da utilização de vários coeficientes na sua resposta ao impulso, podendo assim, resultar em sequências longas. Isso faz que seja necessário o uso de uma memória adicional, um número maior de operações e mais tempo para execução da filtragem (NALON, 2009).

Segundo Shenoi (2006) os filtros digitais FIR possuem algumas vantagens em relação aos filtros IIR:

- Filtros FIR podem ser facilmente projetados para se obter uma resposta em magnitude necessária de forma que ele atinja um atraso de grupo constante. Nesse tipo de filtro todas as frequências são transmitidas com a mesma quantidade de atraso. Portanto, não haverá distorção de fase, o sinal de entrada será atrasado por uma constante de tempo.
- O filtro FIR é sempre estável.

2.2.2.2 Filtro Resposta Infinita ao Impulso (IIR)

Os filtros de resposta infinita ao impulso, também conhecidos como filtros recursivos, possuem a realimentação como diferença para os filtros não recursivos. Por conta da realimentação para o valor de saída $y[n]$ ser calculado é necessário considerar um número finito k de amostras passadas deste sinal, ou seja, $y[n-k]$ convolvidos com os coeficientes $a_k\,a_k$ do filtro, além de considerar a convolução dos coeficientes $b_k\,b_k$ do filtro com o sinal de entrada $x[n]$ como nos filtros não recursivos (ROSA, 2012). Um filtro IIR possui a seguinte equação de diferenças:

$$y[n] = \sum_{k=0}^{N} b_k x[n-k] + \sum_{k=1}^{N} a_k y[n-k]. \tag{12}$$

A partir da Equação de diferenças (12) é possível chegar à relação entre entrada e saída, ou seja, na função de transferência desse filtro, que é dada pela equação:

$$\frac{Y(z)}{X(z)} = \frac{\sum_{n=0}^{M} b_n z^{-n}}{1 + \sum_{n=1}^{N} a_n z^{-n}}. \tag{13}$$

A Equação 13 apresenta uma combinação de polos e zeros, uma função de transferência de baixa ordem em forma fracionaria. Por conta dos polos a implementação digital dessa função de transferência necessita de procedimentos recursivos. Em vista de uma implementação de um algoritmo de um filtro com complexidade computacional econômica,

os filtros recursivos, devido a sua ordem inferior, conseguem um menor tempo de computação (ZÖLZER, 2008).

A aproximação de filtros digitais recursivos pode ser realizada por meio da conversão de uma função de transferência de tempo contínuo. Primeiramente deve-se projetar um filtro analógico a fim de satisfazer determinadas especificações. Esse processo é realizado por meio de uma aproximação padrão de filtro analógico. Posteriormente, a função de transferência de tempo discreto é obtida pela transformação do plano complexo s no plano complexo z, utilizando uma técnica de aproximação, como, a bilinear (ANTONIOU, 2016). Essa transformação é definida conforme a Equação 14, em que T_a T é o período de amostragem.

$$s = \frac{2}{T_a}\left(\frac{z-1}{z+1}\right). \tag{14}$$

Supondo que $H_a(s)\,H_a(s)$ seja uma função de transferência de um filtro analógico, substituindo a transformação bilinear da Equação 14 em $H_a(s)$ será obtida a função de transferência do filtro digital correspondente (HAYKIN; VEEN, 2001), apresentada na Equação 15.

$$H(z) = H_a(s)\Big|_{s=\left(\frac{2}{T_a}\right)\left(\frac{z-1}{z+1}\right)}. \tag{15}$$

2.3 EQUALIZADOR

Um dos equipamentos mais importantes utilizados no processamento de sinais de áudio é o equalizador. Os que possuem filtros mais complexos são utilizados em estúdio de áudio, porém em um nível mais básico, se encontram em quase todos os equipamentos de consumo, como, rádios automotivos, amplificadores, aparelhos de som portáteis, entre outros. Esses possuem filtros mais simples para equalizar o som (ZÖLZER, 2008).

Segundo Juth (2016) um equalizador possui duas funções. Pode ser utilizado para moldar o tom de um som de forma criativa, ou seja, ajustar o som de maneira que seu timbre seja totalmente diferente do original. Essa função é utilizada quando o usuário não está preocupado em manter a fidelidade do som. O equalizador também pode ser utilizado para alterar e corrigir o espectro de frequência do som, ou seja, atenuar ou aumentar certas bandas

de frequência, tudo isso para, por exemplo, retirar harmônicos indesejados ou para aumentar regiões mais desejadas.

Deve-se levar em consideração a resposta em frequência de sistemas de captura e reprodução sonora, pois uma música pode ser mal executada por um sistema de reprodução, por mais que ela seja bem gravada e devidamente equalizada durante o processo de gravação e mixagem. As características no domínio da frequência do sinal gerado a partir desse sistema de reprodução são diferentes se comparadas com a mesma música sendo executada em um sistema de reprodução com resposta em frequência ideal. Entende-se por sistema com resposta em frequência ideal aquele que não acrescenta ou retira energia em alguma faixa de frequência. O mesmo pode ocorrer com sistemas de captura com resposta de frequência diferente da ideal, ou seja, o sinal gravado por esse sistema possui um espectro de frequência diferente do sinal emitido originalmente (SILVA, 2005).

Sendo assim quando um sistema de som for configurado, primeiramente o sistema como um todo deve ser equalizado para compensar a resposta em frequência dos equipamentos de reprodução utilizados e do ambiente de audição. Após isso os canais dos microfones e dos instrumentos musicais são equalizados (REED, 2000). Com isso, pode-se observar que o processo de equalização pode ser utilizado para diversos fins. Por mais que todo o sistema já esteja devidamente compensado, pode-se ainda fazer uso do equalizador por uma simples questão de gosto do ouvinte.

Segundo Herrera (2004), a equalização é um método de correção utilizado para ajustar um sistema de áudio. Seu objetivo é tornar a resposta em frequência do sistema a mais plana possível (*flat*), o que significa ouvir um áudio transparente, fiel ou equilibrado.

Com isso, tem-se que o objetivo de aparelhos de captação sonora (microfones), reprodução de áudio (amplificadores e alto falantes), armazenamento e transmissão (cabos) de sinais de áudio é apresentar uma resposta em frequência plana (HERRERA, 2004). Tal comportamento pode ser limitado por diversos fatores, como um projeto ineficaz ou até mesmo características físicas intrínsecas dos elementos utilizados, gerando, assim, distorções na resposta em frequência. Os equalizadores são encontrados nos processos de distribuição de sinais de áudio, por exemplo, em uma gravação em estúdio de som e na reprodução do sinal de áudio (ROSA, 2012).

Herrera (2004) destaca que a qualidade de um equalizador é determinada por alguns fatores, como a coerência entre os parâmetros configurados no equalizador e a sua real atuação do sinal e a fácil utilização.

No mercado são encontrados dois principais tipos de equalizadores: os equalizadores gráficos, que são mais comuns, e os equalizadores paramétricos, que são mais complexos.

2.3.1 Equalizador Gráfico

Devido a sua baixa complexidade e menor preço, os equalizadores gráficos são mais utilizados no mercado. A fim de cobrir todo o espectro auditivo, esse tipo de equalizador é composto por vários filtros do tipo passa banda, porém suas frequências centrais e largura de banda são fixas. O equalizador gráfico é útil quando o usuário necessita manipular várias frequências ao mesmo tempo (BODANESE, 2008).

A largura de banda será determinada pela quantidade de bandas do equalizador. A largura de banda normalmente é classificada em oitavas, sendo elas, 1/3, 1/2, 2/3 ou 1/1, correspondendo a 31, 20, 15 ou 10 filtros de pico respectivamente, e igualmente espaçados entre si para cobrir todo o espectro de áudio (TERAHATA, 2003).

O equalizador do fabricante DBX (Figura 10) é um exemplo comercial de equalizador gráfico. Ele possui trinta e uma frequências centrais, que admite apenas variar o ganho de cada frequência, não permitindo assim uma maior flexibilidade ao usuário.

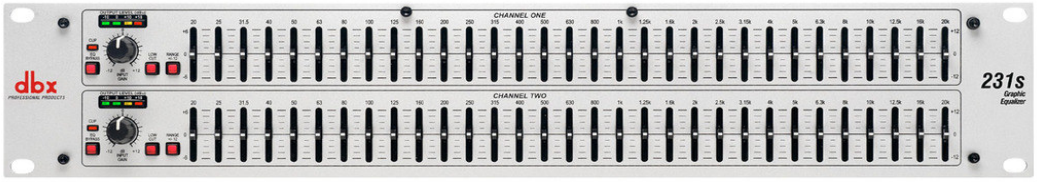


Figura 10 - Equalizador gráfico comercial da DBX

Fonte: Dbx (s.d.).

Já a Figura 11 exibe os controles de um canal do equalizador gráfico CGE 2101S da marca CICLOTRON. Esse equalizador possui dois canais, dez bandas de uma oitava centradas de 31Hz a 16kHz, com filtros de Q-constante (CICLOTRON, s.d).

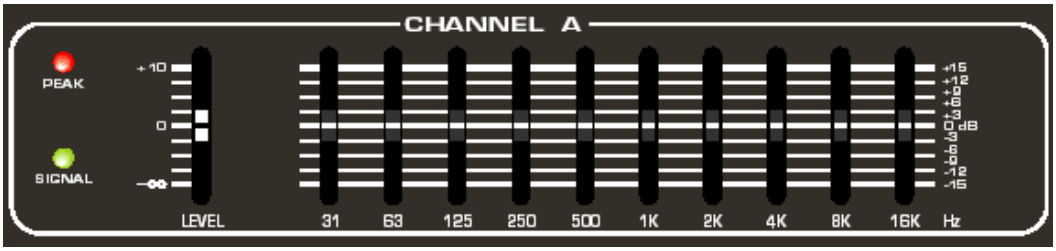


Figura 11 - Equalizador gráfico CGE 2102S - Ciclotron

Fonte: Ciclotron (s.d.).

2.3.2 Equalizador Paramétrico

A principal característica dos equalizadores paramétricos é a sua flexibilidade de configuração, que permite ao usuário obter os sinais de saída como desejado (ROSA, 2012).

Usualmente esses equalizadores apresentam de uma a quatro bandas, atuando de forma independente sobre os três principais parâmetros de um filtro (NEIVA, 2013):

a) Frequência central;

b) Largura de banda passante ou fator de seletividade Q;

c) Quantidade de reforço ou atenuação aplicada ao sinal.

Devido a essa flexibilidade, é possível obter praticamente qualquer efeito de equalização, por meio da associação em cascata de quatro ou cinco desses filtros parametrizáveis (ROSA, 2012).

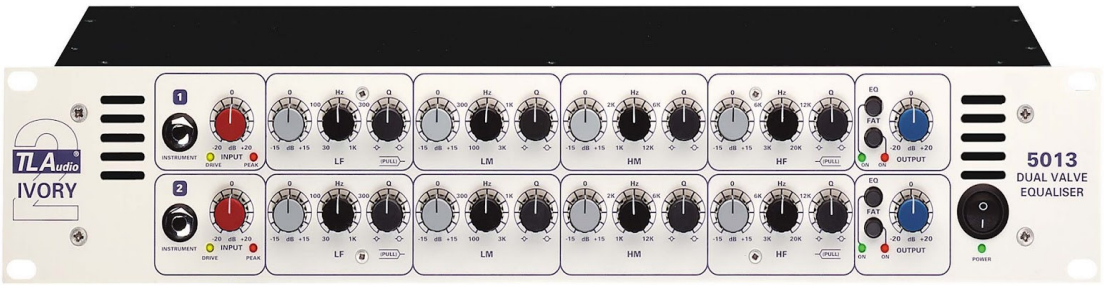


Figura 12 - Equalizador paramétrico

Fonte: Rocha (2012).

A Figura 12 exibe um modelo de equalizador paramétrico disponível no mercado, o qual possui, em média, um custo cinco vezes mais elevado se comparado ao equalizador gráfico. Através da imagem é possível verificar que o equalizador paramétrico possui uma interface que permite ao usuário configurar a largura de banda, ganho e frequência central. Isso possibilita uma melhor personalização do sistema, a fim de obter a resposta desejada.

Figura 13 - Ilustração de equalizador paramétrico

Fonte: Fernandes (s.d.).

A Figura 13 mostra parte de um equalizador paramétrico profissional. Como pode-se observar, na banda 1 apresentada, o usuário pode selecionar a frequência central entre 100 Hz a 2 kHz. Seu valor de Q varia entre 0,03 e 2 oitavas, e o ganho de –15 a +12 dB.

Os conceitos necessários para a implementação de um equalizador paramétrico foram apresentados nesse capítulo, tendo em vista que um equalizador é um método para modificar sinais de áudio a partir de filtros, que selecionam características específicas e desejadas dos sinais. Ainda, um equalizador é composto por um número finito de filtros, sendo que seus parâmetros podem ser ajustados pelo usuário a fim de obter a resposta desejada.

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método utilizados no desenvolvimento do projeto.

3.1 MATERIAIS

Para a execução do trabalho proposto foi utilizado um *kit* de desenvolvimento STM32F746G Discovery da STMicroelectronics. Este *kit* é equipado com o microcontrolador STM32F746NGH6, que possui um núcleo ARM Cortex-M7 de 32-*bits* operando com frequência máxima de 216 MHz. Os processadores Cortex-M7 possuem instruções para processamento digital de sinais e uma unidade de ponto flutuante. Esse microcontrolador também possui quatro I^2Cs, seis SPIs, três I^2Ss, quatro UARTs, três ADCs de 12-*bits*, dois DACs de 12-*bits*, tela sensível ao toque de 4,3 polegadas com resolução de 480x272, 1 Mbyte de memória Flash, 320 Kbyte de SRAM (STMICROELECTRONICS, 2017).

O *kit* de desenvolvimento STM32F746G Discovery conta também com um *codec* de áudio WM8994ECS/R da Cirrus Logic que está conectado à interface SAI (*Serial Audio Interface*) do microcontrolador STM32F746NGH6, e se comunica com o mesmo pelo barramento I^2C compartilhado com o módulo da câmera (STMICROELECTRONICS, 2017).

O WM8994 é um *codec* de alta fidelidade e de consumo de energia extremamente baixo, projetado para ser usado em *smartphones* e dispositivos portáteis com recursos multimídia, tais como: *tablets*, *ebooks*. Esse *codec* possui um equalizador totalmente programável que pode ser usado nos caminhos dos ADCs e DACs, a fim de manter o sinal em um nível constante e proteger os alto-falantes de sobrecargas (CIRRUS LOGIC, 2018).

Segundo Cirrus Logic (2018) as características do *codec* são:

- 4 canais DACs e dois canais ADCs de alta fidelidade;
- Resolução dos DACs e ADCs de até 24 *bits*;
- Amplificador para alto-falantes de classe D e AB estéreo;
- Equalizador paramétrico 5 bandas ReTune;
- Taxas de amostragem de 8 kHz a 96 kHz.

Além do microcontrolador também foi necessário o uso de *softwares* de simulação, Ambiente de Desenvolvimento Integrado (*Integrated Development Environment* - IDE) eclipse e *software* para desenvolvimento da interface gráfica.

3.2 MÉTODOS

A implementação do equalizador paramétrico foi realizada em 5 etapas: projeto e simulação dos filtros digitais, configuração do *kit* de desenvolvimento, implementação dos filtros, implementação da interface gráfica e realização de testes para validação dos resultados.

Na etapa de projeto dos filtros, foram escolhidas as faixas de frequências em que cada banda do equalizador atuará. Após, foram projetados todos os filtros com o auxilio do Matlab, os *peak* usando a função *designParamEQ* e o *low shelf* e *high shelf* usando os códigos das Listagens 1 e 2 respectivamente. Ainda nessa etapa os filtros projetados foram simulados a fim de verificar seu funcionamento e comprovar que todos os filtros foram projetados de forma correta.

Na segunda etapa todas as configurações necessárias para o correto funcionamento do microcontrolador e do *codec* foram realizadas. O *codec* foi configurado para utilizar uma resolução de 24 *bits*, com o objetivo de obter uma melhor qualidade na representação do sinal de áudio e uma taxa de amostragem de 48 kHz.

Com os filtros projetados e o *kit* de desenvolvimento devidamente configurado, a próxima etapa foi a implementação do filtro. O equalizador foi implementado de duas formas, por *software,* utilizando filtros do tipo IIR, e por *hardware,* utilizando o equalizador paramétrico móvel *retune* do *codec* contido no *kit* de desenvolvimento.

A seguir foi desenvolvida a interface gráfica para que o usuário possa interagir com o equalizador e ajustar os parâmetros desejados. Na última etapa foram realizados testes para verificar se os resultados obtidos na prática condizem com os apresentados na teoria.

4 DESENVOLVIMENTO

Este capítulo apresenta o desenvolvimento do projeto final. Inicialmente o projeto e a simulação dos filtros digitais, posteriormente a configuração do *kit* de desenvolvimento e por fim a implementação do equalizador e da interface gráfica.

4.1 PROJETO E SIMULAÇÃO DOS FILTROS DIGITAIS

Antes dos filtros serem projetos foi necessário definir as faixas de frequências em que cada banda do equalizador atuará. O equalizador desse projeto é composto de cinco bandas, a primeira do tipo *low shelf,* seguida de três do tipo *peak* e por fim uma do tipo *high shelf.* Foi definida também a faixa do fator Q dos filtros do tipo *peak* e a faixa do ganho de todos os filtros.

Os filtros passa altas e passa-baixas são muito úteis para o corte de sinais acima ou abaixo de uma determinada frequência. No entanto, em tais filtros ocorre mudança de ganho/atenuação progressivo nos sinais fora da banda de interesse (banda de transição e banda de rejeição). Por essa razão, a configuração de tons de alta ou baixa frequência tende a ser feita usando um filtro *shelf*. Filtros *shelving* são projetados para aplicar uma mudança de ganho igual a todas as frequências além de uma frequência selecionada pelo projetista, em vez de aplicar uma mudança de ganho progressiva além de um ponto de corte. Tais filtros requerem não apenas um controle para selecionar a frequência *shelving*, mas também para selecionar a quantidade de corte ou ganho aplicado na banda de interesse.

Embora os filtros *shelving* tendam a ser úteis para um ajuste de tons geral e suave, eles são menos úteis para aquelas aplicações em que é necessário atingir bandas de frequência específicas com maior precisão. Para afetar uma banda de frequência que não esteja em nenhum extremo do espectro de frequência é necessário um filtro de pico (*peak filter*). Esse filtro permite enfatizar ou atenuar seletivamente uma faixa limitada do espectro de áudio e geralmente oferece pelo menos dois controles, um para definir o ganho aplicado na banda de interesse e outro para especificar a frequência central da banda a ser tratada. Os filtros *low* e *high shelving* e os filtros de pico são baseados em filtros passa baixas, passa altas e passa banda, respectivamente, e um caminho direto do sinal.

A Tabela 1 apresenta a faixa de valores (valor inicial ao valor final) e o valor que será incrementado a cada passo. Por exemplo, a faixa de frequência que a banda *peak* 1 atua é

de 250 Hz até 2650 Hz, com a frequência variando de 200 em 200 Hz. O fator Q possui uma faixa de atuação de 0,5 a 5 com passos de 0,5, o ganho tem sua faixa ajustável de +/- 12 dB, com passos 1 dB.

Tabela 1 - Faixa de atuação de cada banda

Banda	Valor Inicial	Valor Final	Passos	Tamanho Passos
Low Shelf	50 Hz	400 Hz	14	25 Hz
Peak 1	250 Hz	2650 Hz	12	200 Hz
Peak 2	500 Hz	6500 Hz	12	500 Hz
Peak 3	1600 Hz	16000 Hz	12	1200 Hz
High Shelf	12000 Hz	18000 Hz	12	500 Hz
Q	0,5	5	9	0,5
Ganho	-12 dB	12 dB	24	1 dB

O filtro do tipo FIR possui a necessidade da utilização de vários coeficientes na sua resposta ao impulso, o que resulta em sequências longas, ou seja, ordem mais elevada se comparado ao mesmo filtro do tipo IIR. Ademais, filtros FIR resultam em maior uso de memória e um número elevado de operações matemáticas, o que resulta em maior tempo para sua execução. Essas características limitam o uso desse tipo de filtro em muitas aplicações baseadas em microcontroladores.

Assim, optou-se pela utilização de filtros do tipo IIR, de oitava ordem. Tais filtros foram projetados em seções de segunda ordem (S*econd Order Sections* - SOS) transposta.

Os projetos dos filtros digitais foram realizados utilizando o *software* MatLab desenvolvido pela MathWorks. Os coeficientes para todos os filtros foram calculados e armazenados em vetores. Para os filtros do tipo *peak* foi utilizada a função *designParamEQ*, que calcula os coeficientes a partir de parâmetros informados na função, os quais são: ordem, ganho, frequência central e largura de banda. O retorno dessa função são duas matrizes, numerador e denominador, respectivamente, cujas suas colunas correspondem a filtros SOS em cascata.

Para os filtros do tipo *low shelf* e *high shelf* calcularam-se os coeficientes com base no equacionamento apresentado em Zölzer (2008), porém com uma adaptação para filtros de oitava ordem, utilizando os códigos das Listagens 1 e 2 respectivamente.

```
V = powf(10, fabsf( gain )/20 );
K = tanf( M_PI*( freq/48000.0 )  );

if( gain >= 0 ){
      norm = 1/( 1 + 1/Q(o) * K + K * K );
      b0 = ( 1 + sqrt(V)/Q(o) * K + V * K * K ) * norm;
      b1 = 2 * ( V * K * K - 1 ) * norm;
      b2 = ( 1 - sqrt(V)/Q(o) * K + V * K * K ) * norm;
      a1 = 2 * ( K * K - 1) * norm;
      a2 = (1 - 1/Q(o) * K + K * K ) * norm;
}else{
```

```
      norm = 1/( 1 + sqrt(V)/Q(o) * K + V * K * K );
      b0 = (1 + 1/Q(o) * K + K * K ) * norm;
      b1 = 2 * ( K * K - 1) * norm;
      b2 = ( 1 - 1/Q(o) * K + K * K ) * norm;
      a1 = 2 * ( V * K * K - 1 ) * norm;
      a2 = ( 1 - sqrt(V)/Q(o) * K + V * K * K ) * norm;
}
```

Listagem 1 - Cálculo coeficientes filtro *low shelf*

```
V = powf(10, fabsf( gain )/ 20 );
K = tanf( M_PI*( freq/48000.0 )  );

if( gain >= 0 ){
      norm = 1 / ( 1 + 1/Q(o) * K + K * K );
      b0 = ( V + sqrt(V)/Q(o) * K + K * K ) * norm;
      b1 = 2 * ( K * K - V ) * norm;
      b2 = ( V - sqrt(V)/Q(o) * K + K * K ) * norm;
      a1 = 2 * ( K * K - 1 ) * norm;
      a2 = ( 1 - 1/Q(o) * K + K * K ) * norm;
}else{
      norm = 1/( V + sqrt(V)/Q(o) * K + K * K );
      b0 = ( 1 + 1/Q(o) * K + K * K ) * norm;
      b1 = 2 * ( K * K - 1 ) * norm;
      b2 = ( 1 - 1/Q(o) * K + K * K ) * norm;
      a1 = 2 * ( K * K - V ) * norm;
      a2 = ( V - sqrt(V)/Q(o) * K + K * K ) * norm;
}
```

Listagem 2 - Cálculo coeficientes filtro *high shelf*

A resposta em frequência dos filtros foi obtida a partir da função *fvtool* do MatLab, de forma a verificar se os mesmos estavam de acordo com os requisitos de projeto. Com o ganho variando com degraus de 1 dB, cada filtro tem um total de 25 configurações diferentes, sendo 12 faixas de *cut* (atenuação nas frequências do filtro), 12 faixas de *boost* (ganho nas frequências do filtro) e a condição sem ganho.

Levando em consideração que há um total de 10450 filtros projetados e a inviabilidade de apresentar a simulação de todos, optou-se por apresentar apenas um. Tomando como exemplo as frequências centrais 250, 1000, 4000 Hz para os filtros *peak* e as frequências de corte 50 e 12000 Hz para os filtros *low shef* e *high shelf*, respectivamente, há um total de 125 configurações possíveis para os cinco filtros de oitava ordem que compõem o equalizador proposto. A resposta em frequência dos filtros nessas configurações é apresentada da Figura 14.

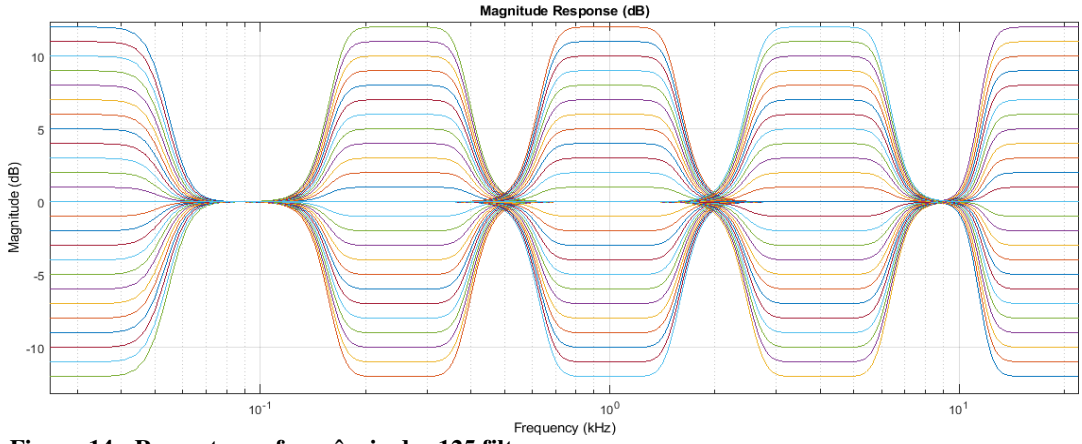


Figura 14 - Resposta em frequência dos 125 filtros

Fonte: Autoria própria.

4.2 IMPLEMENTAÇÃO DO EQUALIZADOR

Para a implementação do equalizador paramétrico, os filtros foram arranjados em série, que também é conhecida como estrutura em cascata, sendo a saída do primeiro filtro conectada à entrada do segundo filtro, e assim sucessivamente. O equalizador foi configurado de tal forma que o sinal de entrada passe primeiramente por um filtro *low shelf*, seguido de três filtros *peak*, finalizando em um filtro *high shelf*, como é apresentado em forma de diagrama de blocos na Figura 15.

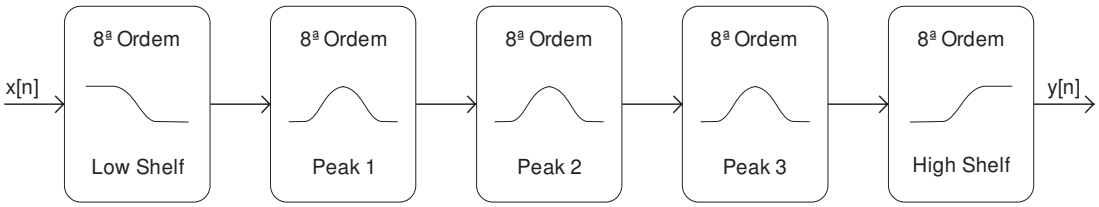


Figura 15 - Estrutura dos filtros do equalizador paramétrico

Fonte: Autoria própria.

4.2.1 Implementação dos Filtros Digitais

A biblioteca DSP do *Cortex Microcontroller Software Interface Standard* (CMSIS) foi utilizada para a implementação dos filtros digitais, pois é otimizada para processadores ARM Cortex-M. Com essa biblioteca, podem-se utilizar diferentes estruturas de filtros digitais, como a Forma Direta I e a Forma Direta II transposta. A estrutura na Forma Direta I

aceita dados em ponto flutuante e em ponto fixo no formato Q15 e Q31. Já a estrutura na Forma Direta II transposta aceita apenas dados em ponto flutuante. Apesar disso, tal estrutura requer apenas duas variáveis de estado por biquad, $d1$ e $d2$, ou seja, metade se comparado às quatro variáveis de uma estrutura na forma direta I.

Devido à economia de memória e processamento, por haver menos variáveis de estado, optou-se pela implementação dos filtros digitais usando estrutura na Forma Direta II transposta, em cascata de seções de segunda ordem.

A Figura 16 apresenta a estrutura de um filtro biquad na Forma Direta II transposta, em que $x[n]$ é o sinal de entrada, b_0, b_1 e b_2 são os coeficientes do filtro que multiplicam $x[n]x[n]$, também conhecidos como ganhos *feedforward,* $y[n]$ é o sinal de saída, a_1 e a_2 são os coeficientes que multiplicam o sinal $y[n]y[n]$, também conhecido como ganho *feedback* e d_1 e d_2 são as variáveis de estado. Para o uso dessa biblioteca os coeficientes a_1 e a_2 devem ter seus valores invertidos. A Listagem 3 apresenta o algoritmo correspondente a essa estrutura.

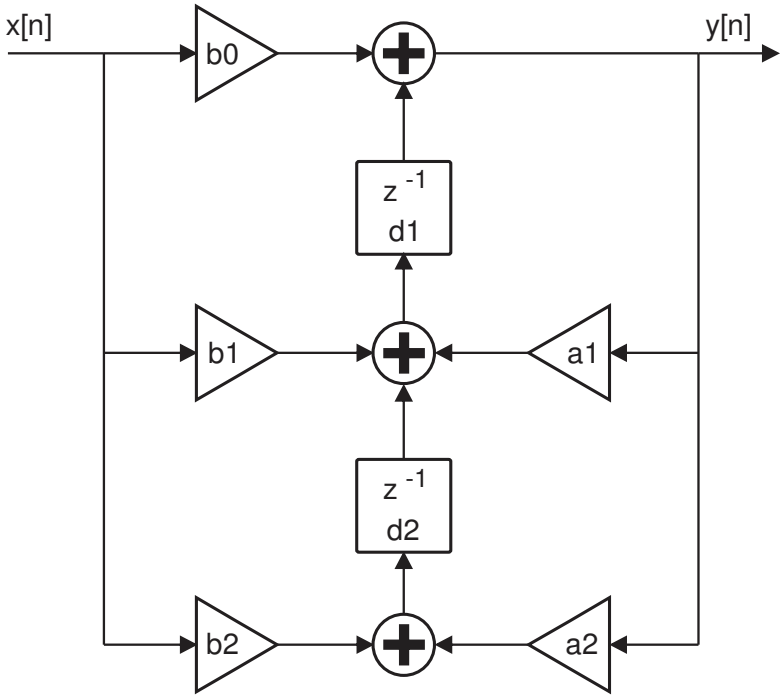


Figura 16 - Estrutura Forma Direta II transposta de um filtro digital IIR

Fonte: Adaptado de Shenoi (2006, p. 315).

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

Listagem 3 - Algoritmo do filtro para um biquad

Para utilizar a biblioteca DSP do CMSIS, primeiro é necessário inicializar a estrutura do filtro a partir da função de inicialização, que define os valores dos campos da estrutura interna, *arm_biquad_cascade_df2T_instance*, a partir dos outros parâmetros da função, ou seja, o número de estágios biquad do filtro, *numStages*, um ponteiro que aponta para os coeficientes do filtro, **pCoeffs* e por último um ponteiro apontando para o *buffer* de estado, **pState*. A Listagem 4 apresenta o código da função de inicialização do filtro.

```
void arm_biquad_cascade_df2T_init_f32 ( arm_biquad_cascade_df2T_instance_f32   * S,
                                        uint8_t                               numStages,
                                        float32_t                             * pCoeffs,
                                        float32_t                             * pState);
```

Listagem 4 - Função de inicialização *arm_biquad_cascade_df2T_init*

O vetor *pCoeffs* deve conter um total de coeficientes cinco vezes maior que *numStages* na seguinte ordem: $\{b_{10}, b_{11}, b_{12}, a_{11}, a_{12}, b_{20}, b_{21}, b_{22}, a_{21}, a_{22}, \ldots\}$, onde b_{1x} e a_{1x} b_{1x} são os coeficientes para o primeiro estágio, b_{2x} e a_{2x} são os coeficientes para o segundo estágio e, assim, sucessivamente. O vetor *pState* deve ser capaz de armazenar um total de valores duas vezes maior que *numStages* e devem ser armazenados na seguinte ordem: primeiro as duas variáveis de estado do estagio um, em seguida as do estagio dois e assim por diante.

Com a estrutura do filtro inicializada, para o áudio ser processado é necessário chamar a função *arm_biquad_cascade_df2T_f32*, sendo que cada chamada processa um bloco de dados, trecho do áudio, por meio do filtro. Essa função é apresentada no código da Listagem 5, onde **S* aponta para a estrutura de dados de instância do filtro, **pSrc* aponta para o bloco de dados de entrada, **pDst* aponta para o bloco de dados de saída e *blockSize* é o número de amostras a serem processadas. Tanto o bloco de dados de entrada quanto o bloco de dados de saída contêm *blockSize* valores.

```
void arm_biquad_cascade_df2T_f32 ( const arm_biquad_cascade_df2T_instance_f32  * S,
                                   float32_t                                   * pSrc,
                                   float32_t                                   * pDst,
                                   uint32_t                                  blockSize);
```

Listagem 5 - Função *arm_biquad_cascade_df2T*

Os coeficientes de todos os filtros digitais IIR projetados foram salvos na memória *flash*. Por conta desse tipo de memória ser mais lenta se comparado a memória RAM, a cada interação do usuário para ajustar algum parâmetro do equalizador, os coeficientes do filtro alterado são copiados da memória *flash* para a memória RAM, assim o equalizador executará de forma mais eficiente.

4.2.2 Equalizador Paramétrico Móvel *Retune*

A equalização utilizando *codec* de áudio possui um custo computacional mais baixo, pois utiliza *hardware* específico para essa aplicação. O *codec* foi utilizado nesse projeto por possuir uma resolução mais alta em seus conversores se comparado com os do *kit* de desenvolvimento.

Não se faz necessário o uso de codec de áudio para essas aplicações, pois filtros para equalizadores digitais de áudio também podem ser implementados por *software*, ou seja, em qualquer microcontrolador, mesmo sem *codec*, contando ADC e DAC, pois para ser processado é necessário que o sinal de áudio seja convertido em uma palavra digital.

O equalizador paramétrico móvel *retune* é um circuito que pode ser ativado no caminho de reprodução das interfaces de áudio digital do codec WM8994, que fornece três circuitos estéreos de equalização, que estão associados com as interface de áudio digital *AIF1 timeslot 0, AIF1 timeslot 1* e *AIF2* respectivamente. O equalizador é habilitado usando o *bit* 0 do registrador correspondente a cada caminho, conforme apresenta a Tabela 2.

Tabela 2 - Controle EQ em AIF1 *Timeslot* 0

Endereço Registrador	Bit	Nome	Default	Descrição
R1152 (0480h) AIF1 DAC1 EQ Gain (1)	0	AIF1DAC1_EQ_ENA	0	Habilitar EQ em AIF1DAC1 (AIF1, Timeslot 0) 0 = Desabilitado 1 = Habilitado
	15:11	AIF1DAC1_EQ_B1_GAIN [4:0]	01100 (0dB)	AIF1DAC1 (AIF1, Timeslot 0) Ganho EQ Banda 1 -12dB até +12dB com passos de 1dB
	10:06	AIF1DAC1_EQ_B2_GAIN [4:0]	01100 (0dB)	AIF1DAC1 (AIF1, Timeslot 0) Ganho EQ Banda 2 -12dB até +12dB com passos de 1dB
	05:01	AIF1DAC1_EQ_B3_GAIN [4:0]	01100 (0dB)	AIF1DAC1 (AIF1, Timeslot 0) Ganho EQ Banda 3 -12dB até +12dB com passos de 1dB
R1153 (0481h) AIF1 DAC1 EQ Gain (2)	15:11	AIF1DAC1_EQ_B4_GAIN [4:0]	01100 (0dB)	AIF1DAC1 (AIF1, Timeslot 0) Ganho EQ Banda 4 -12dB até +12dB com passos de 1dB
	10:06	AIF1DAC1_EQ_B5_GAIN [4:0]	01100 (0dB)	AIF1DAC1 (AIF1, Timeslot 0) Ganho EQ Banda 5 -12dB até +12dB com passos de 1dB

O equalizador pode ser configurado para dois modos de operação, *Default* ou *ReTune Mobile*. No modo *Default* as frequências centrais e as larguras de banda são fixas, o ganho

pode ser alterado dentro de uma faixa de +/- 12*dB*. Para ajustar o ganho do equalizador deve-se alterar o valor do registrador correspondente a banda a ser alterada, associado à AIF utilizada, conforme a Tabela 2. Os registradores têm um tamanho de cinco *bits*, seu valor varia de 00000, que corresponde a -12*dB*, até 11000, correspondente a +12dB. A frequência central de cada banda, para um taxa de amostragem de 48 kHz, é apresentada na Tabela 3.

Tabela 3 - Frequência Central cada Banda EQ modo Default

Banda	Frequência Central
1	100 Hz
2	300 Hz
3	875 Hz
4	2400 Hz
5	6900 Hz

No modo *ReTune Mobile*, habilitar o equalizador e ajustar o ganho de cada banda funciona como descrito para o modo *Default*. Porém nesse modo além do ajuste do ganho, a frequência central e largura de banda também podem ser ajustados para cada uma das cinco bandas do equalizador. Para realizar esses ajustes nesse modo, alguns registradores precisam ser configurados, registradores R1154 a R1171 para AIF1DAC1, R1186 a R1203 para AIF1DAC2 e R1410 a R1427 para AIF2. Os coeficientes que devem ser configurados nesses registradores são gerados utilizando ferramentas do *software* de controle WISCE passando os parâmetros do filtro.

A Tabela 4 apresenta os registradores para o equalizador *retune* no AIF1DAC1 com seus respectivos valores padrão.

Tabela 4 - Registradores EQ Retune

Endereço	Registrador	Padrão
R1154	AIF1 DAC1 EQ Banda 1 A	0FCAh
R1155	AIF1 DAC1 EQ Banda 1 B	0400h
R1156	AIF1 DAC1 EQ Banda 1 PG	00D8h
R1157	AIF1 DAC1 EQ Banda 2 A	1EB5h
R1158	AIF1 DAC1 EQ Banda 2 B	F145h
R1159	AIF1 DAC1 EQ Banda 2 C	0B75h
R1160	AIF1 DAC1 EQ Banda 2 PG	01C5h
R1161	AIF1 DAC1 EQ Banda 3 A	1C58h
R1162	AIF1 DAC1 EQ Banda 3 B	F373h
R1163	AIF1 DAC1 EQ Banda 3 C	0A54h
R1164	AIF1 DAC1 EQ Banda 3 PG	0558h
R1165	AIF1 DAC1 EQ Banda 4 A	168Eh
R1166	AIF1 DAC1 EQ Banda 4 B	F829h

R1167	AIF1 DAC1 EQ Banda 4 C	07ADh
R1168	AIF1 DAC1 EQ Banda 4 PG	1103h
R1169	AIF1 DAC1 EQ Banda 5 A	0564h
R1170	AIF1 DAC1 EQ Banda 5 B	0559h
R1171	AIF1 DAC1 EQ Banda 5 PG	4000h

4.2.3 Configuração STM32F746G Discovery

Para capturar e reproduzir o áudio utilizando esse *kit* de desenvolvimento, alguns dispositivos necessitam ser configurados. Para a configuração desses dispositivos foi utilizado o *STM32F746G-Discovery Audio BSP driver* disponibilizado junto ao STM32Cube, uma biblioteca que auxilia na configuração dos dispositivos do *kit*. O código da Listagem 6 apresenta essas configurações.

```
/**
  * Inicializa a gravação e reprodução em paralelo.
  * Parâmetros: InputDevice:  Dispositivo de entrada do áudio(microfone ou linha)
  *             OutputDevice: Dispositivo de saída do áudio(alto-falante, fone de
ouvido ou ambos)
  *             AudioFreq: Frequência do áudio a ser configurada para o periférico
SAI.
  *             BitRes:    Resolução a ser configurada.
  *             ChnlNbr:   Número de canais.
  * Retorno:  AUDIO_OK se as configurações estiverem corretas, diferente se
estiverem incorretas
  *
  * uint8_t BSP_AUDIO_IN_OUT_Init(InputDevice, OutputDevice, AudioFreq, BitRes,
ChnlNbr)
  */
if (BSP_AUDIO_IN_OUT_Init(INPUT_DEVICE_INPUT_LINE_1, OUTPUT_DEVICE_HEADPHONE,
      AUDIO_FREQUENCY_48K, 16, DEFAULT_AUDIO_IN_CHANNEL_NBR) == AUDIO_OK)
{
      Status_audio_init(1);
      audio_init = pdTRUE;
}
else
{
      Status_audio_init(0);
}

// Inicializa buffers SDRAM
memset((uint16_t*)AUDIO_BUFFER_IN, 0, AUDIO_BLOCK_SIZE*2);
memset((uint16_t*)AUDIO_BUFFER_OUT, 0, AUDIO_BLOCK_SIZE*2);
```

Listagem 6 - Configuração STM32F746G Discovery

A função *BSP_AUDIO_IN_OUT_Init* faz todas as configurações necessárias para o correto funcionamento dos dispositivos a serem utilizados, como configurar o *clock* do periférico *Serial Audio Interface* (SAI) com base na taxa de amostragem, inicializar o *codec*

de áudio e o configurar para os dispositivos de entrada e saída de áudio passados como parâmetros para a função. O retorno dessa função é AUDIO_OK se a configuração foi realizada com sucesso e um valor diferente se a configuração falhar.

As configurações utilizadas para os dispositivos são as seguintes: entrada de linha, saída de fone de ouvido, taxa de amostragem de 48 kHz, dois canais de áudio e resolução de 16 bits, devido a limitação do *driver*.

4.2.4 Tarefa de Áudio

Para que o áudio fosse processado em tempo real, sem que houvesse atrasos mesmo interagindo com a interface gráfica, foi necessário utilizar um sistema operacional de tempo real (*Real Time Operating System* - RTOS) e criar uma tarefa de áudio usando a prioridade *RealTime*, a prioridade mais alta disponível no sistema. Fez-se o uso do RTOS da CMSIS. Para adicionar uma tarefa nesse RTOS são utilizadas duas funções, *osThreadDef* e *osThreadCreate*. A função *osThreadDef* define os atributos de uma *thread*, passando os parâmetros nome, prioridade, instâncias e tamanho pilha para essa função. O parâmetro instância define o número de vezes que *osThreadCreate* pode ser chamado para o mesmo *osThreadDef*. Na Listagem 7 está a função *osThreadDef* com mais detalhes.

```
/*
 * osThreadDef(name, thread, priority, instances, stacksz);
 *
 * name               nome da Thread
 * priority           prioridade inicial da thread
 * instances          número de instâncias possíveis
 * stacksz            tamanho da pilha (em bytes) para a Thread
 *
 */

osThreadDef(osAudio_Thread, Audio_Thread,  osPriorityRealtime, 0, 512);
```

Listagem 7 - Função *osThreadDef*

Para criar a tarefa é necessário chamar a função *osThreadCreate* passando os parâmetros, *osThreadDef* definido anteriormente, e *argument*, ponteiro passado para a *thread* como argumento inicial. Essa função está definida detalhadamente no código da Listagem 8.

```
/*
 * osThreadId osThreadCreate ( const osThreadDef_t * thread_def, void * argument )
 *
 * Parâmetros: thread_def -> definição da thread referenciada com osThreadDef
 *             argument   -> ponteiro passado para a Thread como argumento inicial
 * Retorno:    osThreadId -> Id da Thread para referência por outras funções ou
NULL em caso de erro
 *
 */

AudioThreadId = osThreadCreate (osThread(osAudio_Thread),
```

Listagem 8 - Função *osThreadCreate*

4.3 IMPLEMENTAÇÃO DA INTERFACE GRÁFICA

Para que houvesse interação entre o usuário e o equalizador foi necessário o desenvolvimento de uma interface gráfica. Para que essa interação ocorra sem a interrupção do processamento e reprodução do áudio, é necessário criar uma tarefa para a interface. Essa tarefa foi criada da mesma forma que a tarefa de áudio, porém com uma prioridade mais baixa, foi utilizada a Prioridade Normal.

Por meio dessa Interface Gráfica do Utilizador (do inglês *Graphical User Interface* - GUI) pode-se alterar os parâmetros do equalizador, para isso foram criadas três telas. A primeira é a tela principal, apresentada na Figura 17a, onde pode-se ajustar o volume geral, e por meio de um botão rádio escolher entre o equalizador por filtros IIR, equalizador *retune*, ou *bypass*, em que a equalização é desligada.

Na tela do equalizador *retune*, demonstrada a Figura 17b, há uma *multipage* com quatro abas, uma para cada banda, e um gráfico que plota a saída do sistema para a configuração atual. A frequência, a largura da banda e a amplitude de cada banda podem ser ajustadas por meio de *sliders*. Também contém três botões, sendo possível restaurar para as configurações anteriores aos ajustes, salvar ou cancelar os ajustes realizados. A tela de equalização IIR, demonstrada Figura 17c, tem as mesmas funções da tela *retune*, porém os ajustes são realizados por meio de knobs.

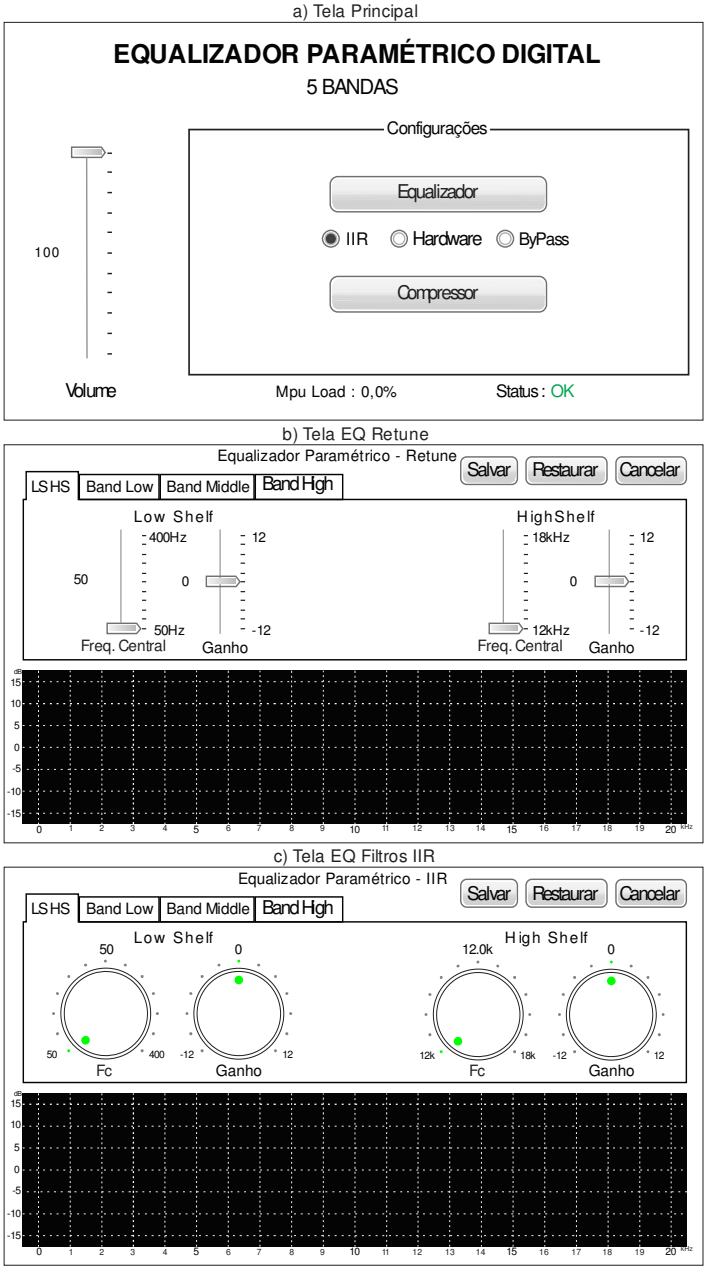


Figura 17 - GUIs do equalizador

Fonte: Autoria própria.

Para o desenvolvimento da GUI foi utilizado a ferramenta GUIBuilder disponível gratuitamente na biblioteca STemWin, uma solução fornecida pela STMicroelectronics devido a uma parceria com a SEGGER Microcontroller GmbH. Essa solução é baseada na SEGGER emWin.

A biblioteca STemWin possibilita a criação de GUIs com qualquer STM32, monitor e controlador LCD/TFT. Ela também possui um conjunto de *widgets*, como botões, caixas de seleção, calendários e medidores. Essa biblioteca aproveita a aceleração de *hardware* sempre

que possível e é fornecida gratuitamente para qualquer cliente STM32.

Para criar uma caixa de diálogo são requeridos dois objetos, uma tabela de recursos, que define os *widgets* incluídos e um procedimento de diálogo, que define os valores iniciais dos *widgets*. Após os dois itens serem criados, é necessário chamar a função *GUI_CreateDialogBox,* apresentada de maneira genérica no código da Listagem 9, para criar de fato a caixa de diálogo.

```
WM_HWIN GUI_CreateDialogBox ( const GUI_WIDGET_CREATE_INFO             * paWidget,
                              int                                     NumWidgets,
                              WM_CALLBACK                                   * cb,
                              WM_HWIN                                  hParent,
                              int                                           x0,
                              int                                           y0);
```

Listagem 9 - Função genérica *GUI_CreateDialogBox*

Primeiramente deve-se definir uma tabela de recursos, especificando os *widgets* que serão inclusos do diálogo, esses devem ser criados indiretamente chamando a função *<WIDGET>_CreateIndirect* como é apresentado no código da Listagem 10. Após a criação da caixa de diálogo, todos os *widgets* estarão visíveis, mas vazios, pois o procedimento de diálogo ainda não possui um código que inicializa os elementos. Os valores iniciais, as ações causadas e as interações entre os *widgets* precisam estar definidos no procedimento de diálogo.

O segundo passo é inicializar os *widgets* com seus valores iniciais, isso é feito no procedimento de diálogo como uma reação à mensagem *WM_INIT_DIALOG*, como é apresentado do código da Listagem 10.

```
static const GUI_WIDGET_CREATE_INFO _aDialogCreate[] = {
  { WINDOW_CreateIndirect, "PageInicial", ID_WINDOW_0, 0, 0, 480, 272, 0, 0x0, 0 },
  { BUTTON_CreateIndirect, "Equalizador", ID_BUTTON_EQ, 222, 105, 150, 25, 0, 0x0, 0 },
  { BUTTON_CreateIndirect, "Compressor", ID_BUTTON_COMP, 222, 173, 150, 25, 0, 0x0, 0 },
  { TEXT_CreateIndirect, "Txt_Mpu_load", ID_TEXT_LOAD, 241, 242, 35, 20, 0, 0x64, 0 },
  { TEXT_CreateIndirect, "Txt_Status", ID_TEXT_STATUS, 377, 242, 90, 20, 0, 0x64, 0 },
  { RADIO_CreateIndirect, "Radio_IIR", ID_RADIO_IIR, 215, 140, 35, 20, 0, 0x1401, 0 },
  { RADIO_CreateIndirect, "Radio_ReTube", ID_RADIO_Hard, 262, 140, 65, 20, 0, 0x1401, 0
},
  { SLIDER_CreateIndirect, "Slider_Vol", ID_SLIDER_VOL, 43, 84, 30, 147, 8, 0x0, 0 },
  { TEXT_CreateIndirect, "Txt_Vol", ID_TEXT_VOL, 13, 147, 25, 20, 0, 0x64, 0 },
  { RADIO_CreateIndirect, "Radio_ByPass", ID_RADIO_ByPass, 335, 140, 60, 20, 0, 0x1401,
0 },
};

static void _cbDialog(WM_MESSAGE * pMsg) {
  WM_HWIN hItem;
  GUI_RECT Rect;
  int  NCode;
  int     Id;
  char buf[6];
  int aux;
  uint8_t tmp;
```

```
  switch (pMsg->MsgId) {
  case WM_INIT_DIALOG:

    hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_LOAD);    //
    TEXT_SetTextAlign(hItem, GUI_TA_LEFT | GUI_TA_VCENTER);// Ini. 'Txt_Mpu_load'
    TEXT_SetText(hItem, "0%");                             //

    hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_STATUS);   //
    TEXT_SetText(hItem, "");                                // Ini. 'Txt_Status'
    TEXT_SetTextAlign(hItem, GUI_TA_LEFT | GUI_TA_VCENTER); //

    hItem = WM_GetDialogItem(pMsg->hWin, ID_RADIO_IIR);    //
    RADIO_SetText(hItem, "IIR", 0);                        // Ini. 'Radio_IIR'
    RADIO_SetGroupId(hItem,1);                             //

    hItem = WM_GetDialogItem(pMsg->hWin, ID_RADIO_Hard);   //
    RADIO_SetText(hItem, "Hardware", 0);                   // Ini. 'Radio_Hardware'
    RADIO_SetGroupId(hItem,1);                             //

    hItem = WM_GetDialogItem(pMsg->hWin, ID_RADIO_ByPass); //
    RADIO_SetText(hItem, "ByPass", 0);                     // Ini. 'Radio_ByPass'
    RADIO_SetGroupId(hItem,1);                             //

    hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_VOL);       //
    TEXT_SetText(hItem, buf);                               // Ini. 'Txt_Vol'
    TEXT_SetTextAlign(hItem, GUI_TA_RIGHT | GUI_TA_VCENTER); //
    TEXT_SetFont(hItem, GUI_FONT_8_1);                      //

    // Inicialização Slider Volume
    hItem = WM_GetDialogItem(pMsg->hWin,ID_SLIDER_VOL);
    SLIDER_SetNumTicks(hItem,100);
    SLIDER_SetRange(hItem,-100,0);
    SLIDER_SetFocusColor(hItem, GUI_STCOLOR_DARKBLUE);
    SLIDER_SetBkColor (hItem, GUI_WHITE);
    SLIDER_SetWidth(hItem, 0);
    SLIDER_SetSTSkin(hItem);
    SLIDER_SetValue(hItem, -64);
    aux = SLIDER_GetValue(hItem);
    aux *= -1;
    sprintf(buf,"%d",aux);
    break;

  default:

    WM_DefaultProc(pMsg);
    break;
  }
}

void CreateWin_Inicial(void) {

  GUI_CreateDialogBox(_aDialogCreate, GUI_COUNTOF(_aDialogCreate), _cbDialog,
WM_HBKWIN, 0, 0);

}
```

Listagem 10 - Algoritmo de criação da GUI

5 RESULTADOS

Com o objetivo de validar o equalizador foram realizados testes comparando a respostas dos filtros implementados com a resposta teórica. Os testes teóricos foram realizados com o auxílio do *software* Matlab e os práticos com um gerador de função e um osciloscópio.

Para isso foram gerados sinais com frequências pré-definidas, os quais foram processados com o Matlab, usando os mesmos filtros implementados no *kit* de desenvolvimento. Para os testes práticos, o gerador de função foi configurado para gerar os mesmos sinais teóricos, que foram inseridos no *kit* de desenvolvimento pelo *line-in*. O osciloscópio foi utilizado para medir tanto o sinal original quanto o processado, para uma melhor comparação. O primeiro canal do osciloscópio foi ligado junto com a saída do gerador de função para medir o sinal original e o segundo canal foi ligado na saída do *kit* de desenvolvimento para medir o sinal processado.

O processamento e os sinais dos testes foram realizados e gerados pelo Matlab a partir da Listagem 11, em que a variável SOS são os coeficientes que se alteram para cada filtro. As frequências utilizadas foram as seguintes: 350 Hz para o filtro *low shelf,* 18 kHz para o filtro *high shelf* e 3 kHz para o filtro *peak*.

```
Fs = 48000;

fsignal = 350;

t = linspace(0,1,Fs);

x = 0.125*cos(2*pi*fsignal*t);

out = sosfilt(SOS,x);
```

Listagem 11 - Algoritmo de criação e processamento do sinal

O primeiro teste do equalizador foi realizado para testar o filtro *low shelf,* o qual foi configurado para uma amplificação de 4 *dB* em 400 Hz. A Figura 18 apresenta o diagrama de bode para esse filtro.

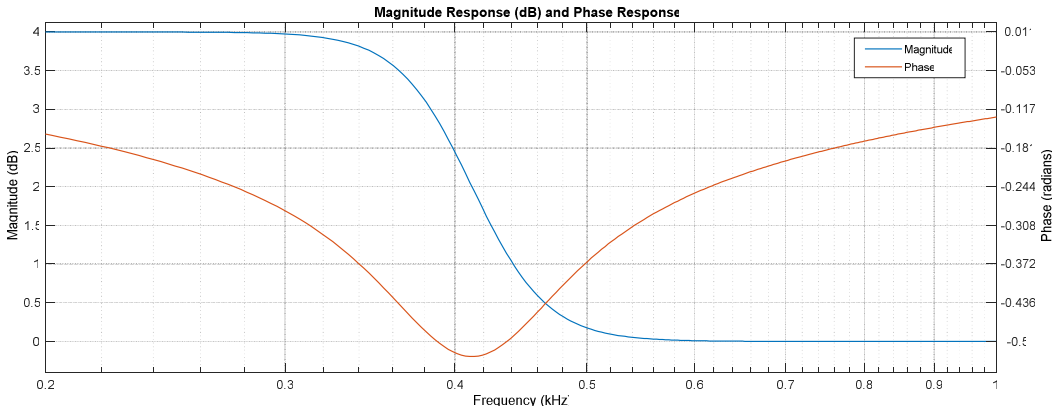


Figura 18 - Diagrama Bode filtro *low shelf* 400 Hz

Fonte: Autoria própria.

O resultado teórico para a filtragem com um sinal de 350 Hz é apresentado na Figura 19.

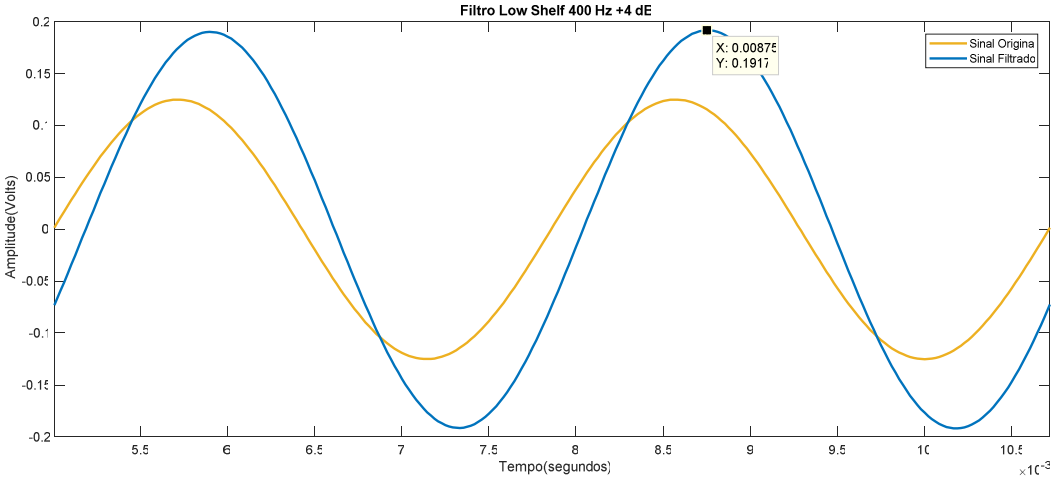
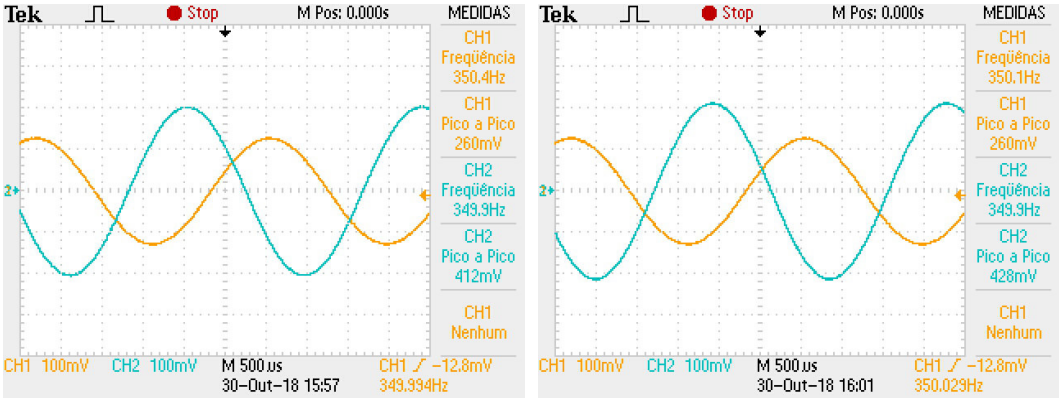


Figura 19 - Resultado teórico filtro *low shelf* 400 Hz

Fonte: Autoria própria.

Já a Figura 20 apresenta o resultado de um sinal de 350 Hz após ser processado pelo equalizador que foi configurado para uma amplificação de 4 *dB* na frequência 400 Hz com o restante das bandas em 0 *db*.



a) Equalizador IIR b) Equalizador Móvel *Retune*

Figura 20 - Resultado prático filtro low shelf 400 Hz

Fonte: Autoria própria.

Os sinais gerados tanto no Matlab quanto no gerador de funções foram configurados para uma amplitude de 250 mVpp. Porem, o sinal medido pelo canal 1 do osciloscópio, correspondente ao sinal original, apresentou uma amplitude de 260 mVpp. Após o processamento, teoricamente, esperava-se uma amplitude de 412 mVpp, o que condiz com a prática, conforme apresentado na Figura 20a, sinal medido pelo segundo canal do osciloscópio, saída do equalizador. Já o sinal processado pelo equalizador móvel *retune*, possui uma amplitude de 428 mVpp, como apresentado na Figura 20b, apresentando um erro de aproximadamente 3,88%.

Posteriormente foi realizado o teste para o filtro *high shelf* configurado para uma atenuação de 5 *dB* em 12 kHz. A Figura 21 apresenta o diagrama de bode para esse filtro.

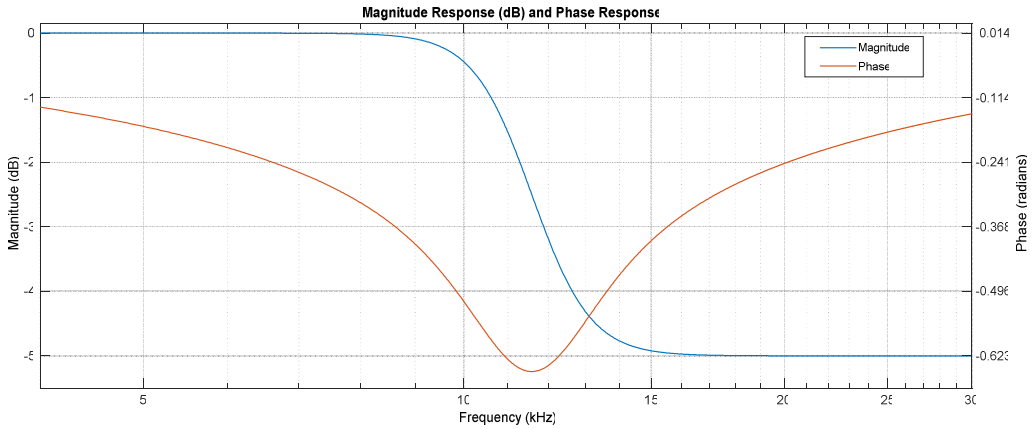


Figura 21 - Diagrama Bode filtro *high shelf* 12 kHz

Fonte: Autoria própria.

O resultado teórico para a filtragem com um sinal de 18 kHz é apresentado na Figura 22.

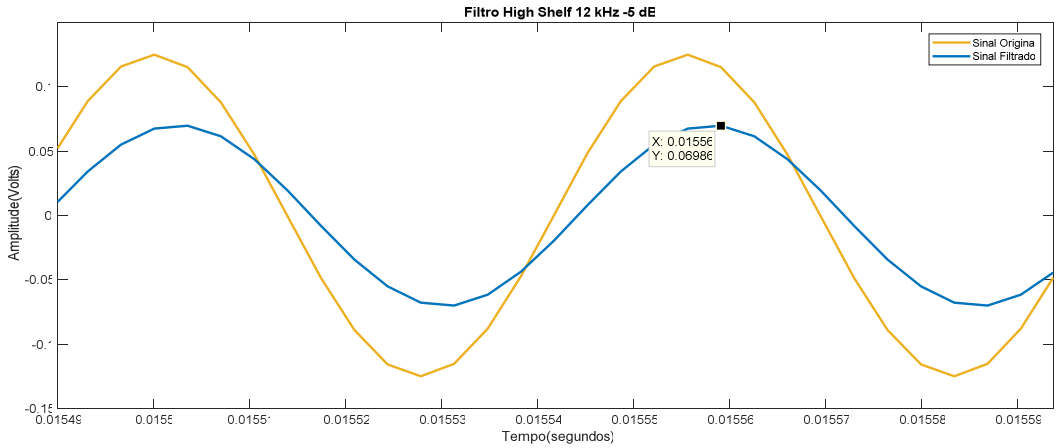
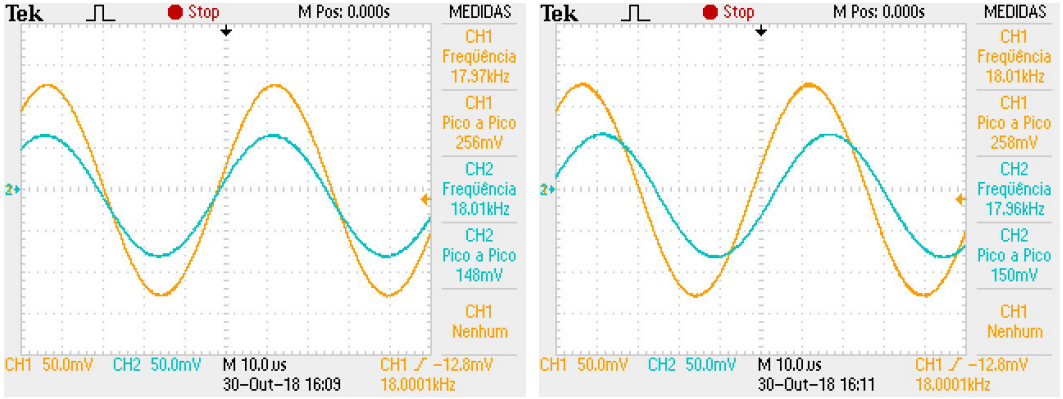


Figura 22 - Resultado teórico filtro *high shelf* 12 kHz

Fonte: Autoria própria.

Já a Figura 23 apresenta o resultado de um sinal de 18 kHz após ser processado pelo equalizador que foi configurado para -5 *dB* na frequência 12 kHz com o restante das bandas em 0 *db*.



a) Equalizador IIR b) Equalizador Móvel *Retune*

Figura 23 - Resultado prático filtro *high shelf* 12 kHz

Fonte: Autoria própria.

Ao realizar o teste do filtro *high shelf* no equalizador por filtros IIR, o sinal medido pelo canal 1 do osciloscópio, correspondente ao sinal original, apresentou uma amplitude de 256 mVpp. Após o processamento, teoricamente, esperava-se uma amplitude de 144 mVpp, porém o sinal medido pelo segundo canal do osciloscópio, saída do equalizador, apresentou uma amplitude de 148 mVpp, conforme apresentado na Figura 23a, com um erro de aproximadamente 2,77%.

Já para o teste do equalizador móvel *retune* o sinal original apresentou uma amplitude de 258 mVpp. Após o processamento, teoricamente, esperava-se uma amplitude de

145 mVpp, porém na teoria apresentou uma amplitude de 150 mVpp, como apresentado na Figura 23b, com um erro de aproximadamente 3,44%.

Para finalizar foi realizado o teste para o filtro *peak* que foi configurado com um ganho de 2 *dB* em 3 kHz com o Q em 1. O diagrama de bode desse filtro é apresentado na Figura 24.

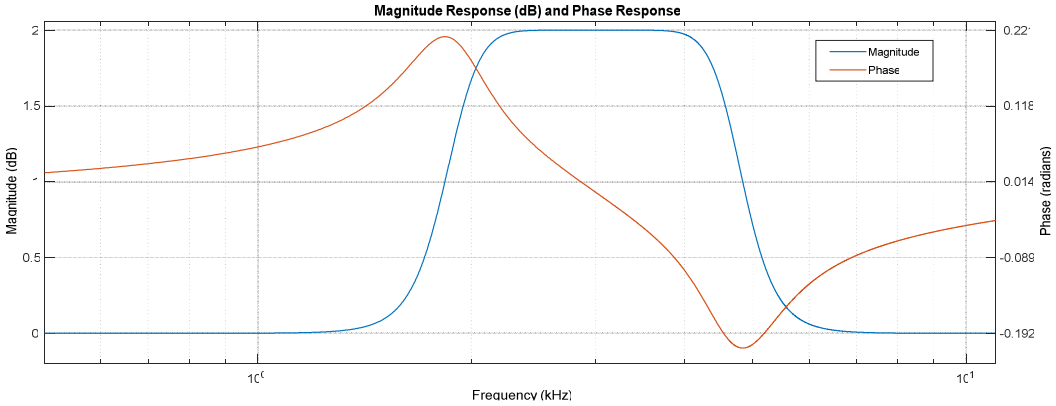


Figura 24 - Diagrama de Bode filtro *peak* 3 kHz Q 1

Fonte: Autoria própria.

A Figura 25 apresenta o resultado teórico para a filtragem com um sinal de 3 kHz.

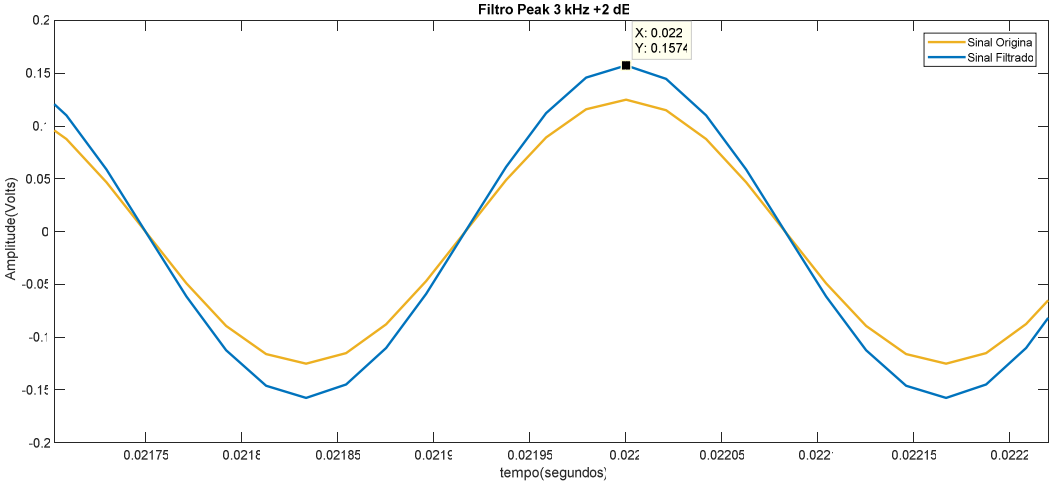
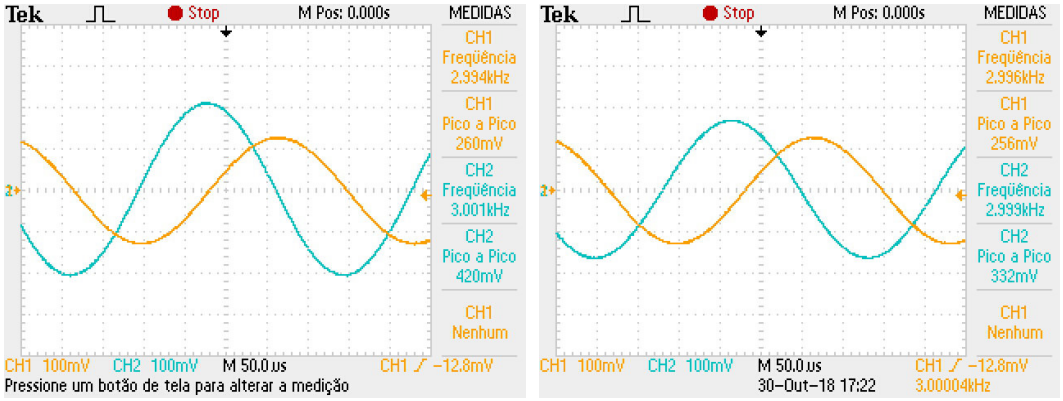


Figura 25 - Resultado teórico filtro *peak* 3 kHz Q 1

Fonte: Autoria própria.

Já a Figura 26 apresenta o resultado de um sinal de 3 kHz após ser processado pelo equalizador que foi configurado para 2 *dB* na frequência 3 kHz com Q igual a 1, com o restante das bandas em 0 *db*.



a) Equalizador IIR b) Equalizador Móvel *Retune*

Figura 26 - Resultado prático filtro Peak 3 kHz Q 1

Fonte: Autoria própria.

Ao realizar o teste do filtro *peak* no equalizador por filtros IIR, o sinal medido pelo canal 1 do osciloscópio, correspondente ao sinal original, apresentou uma amplitude de 260 mVpp. Após o processamento, teoricamente, esperava-se uma amplitude de 327 mVpp, porém o sinal medido pelo segundo canal do osciloscópio, saída do equalizador, apresentou uma amplitude de 420 mVpp, conforme apresentado na Figura 26a, com um erro de aproximadamente 28,44%.

Já para o teste do equalizador móvel *retune* o sinal original apresentou uma amplitude de 256 mVpp. Após o processamento, teoricamente, esperava-se uma amplitude de 322 mVpp, porém na teoria apresentou uma amplitude de 332 mVpp, como apresentado na Figura 26b, com um erro de aproximadamente 3,10%.

Analizando a amplitude dos resultados obtidos com o uso do Matlab e do osciloscopio, observou-se que os resultados teóricos e práticos são muito próximos, validando a implementação de um equalizador digital através da biblioteca DSP do CMSIS em um *kit* de desenvolvimento contendo um microcontrolador com unidade de ponto flutuante com precisão simples e *codec* de áudio.

6 CONCLUSÃO

Neste trabalho foi apresentado o desenvolvimento de um equalizador paramétrico digital de cinco bandas. A implementação foi realizada utilizando um *kit* de desenvolvimento de sistemas microcontrolados com CPU de 32 *bits*, *codec* de áudio e *display* sensível ao toque. O dispositivo possui uma interface gráfica por meio da qual é possível alterar, em tempo real, os parâmetros de equalização.

Para desenvolvimento do trabalho foi necessário realizar estudo sobre representação discreta de sinal, sua amostragem e processo de quantização, além de filtros e seus tipos e, ainda, sobre equalizadores gráfico e paramétrico.

O manual da emWin da SEGGER Microcontroller GmbH é completo e apresenta diversos *widgets* que possibilita a criação de janelas simples mas eficazes. Nele também explica o funcionamento da ferramenta GUIBuilder utilizada para o desenvolvimento da interface gráfica. Para a configuração do *kit* de desenvolvimento, e seu correto funcionamento, a STM disponibiliza um drive (*STM32F746G-Discovery Audio BSP)* que auxilia na configuração dos dispositivos do *kit*.

Para verificar o funcionamento dos filtros digitais implementados, foram realizados testes através do Matlab, gerador de funções e osciloscópio. Os resultados obtidos nos testes de respostas dos filtros do equalizador paramétricos foram satisfatórios quando comparados com os resultados teóricos, validando assim o equalizador paramétrico digital de cinco bandas quando implementado no *kit* de desenvolvimento STM32F746G Discovery.

A escolha de qualquer frequência dentro do espectro auditivo, 20 Hz a 20 kHz, não foi possível, devido a complexidade do cálculo dos coeficientes dos cinco filtros de oitava ordem, aquisição, processamento e reprodução do áudio em tempo real. Fez-se necessário estabelecer frequências pré-definidas e armazenar os coeficientes pré-calculados.

A quantidade de memória necessária para armazenar os coeficientes de todos os filtros projetados é de 4,13 Mbytes, que ultrapassa o espaço de memória *flash* do *kit* de desenvolvimento que é de 1 Mbyte, assim foi necessário armazenar o coeficientes na memória *flash* Quad-SPI de 16 Mbytes também contido no *kit*.

Uma possível abordagem em um trabalho futuro seria a tentativa de calcular os coeficientes sem atraso para que fosse possível a escolha de qualquer frequência não sendo necessário armazenar os coeficientes pré-calculados, e reconfigurar o *driver* para utilizar a resolução em 24 *bits*.

REFERÊNCIAS

ANTONIOU, Andreas. **Digital signal processing**: signals, systems and filters. United States: McGraw-Hill, 2006.

BARBOSA, Álvaro M. **Edição digital de som:** uma abordagem aos fundamentos da escultura sonora orientada para criadores. Escola das artes som e imagem – Universidade Católica Portuguesa. 1999.

BODANESE, João Paulo. **Implementação de um equalizador de áudio em DSP**. Florianópolis, 2008.

BRAGA, Newton C. **A História do DSP.** 2014. Disponível em: <http://www.newtoncbraga.com.br/index.php/eletronica/61-historia-da-eletronica/5894-a-historia-do-dsp-hist039>. Acesso em: 15 mai. 2017.

CICLOTRON. **Equalizadores gráficos**. S.d. Disponível em: < http://www.ciclotron.com.br/produtos/EQUAL_SG/CGE2101-SG/CGE-2101-SG.html>. Acesso em: 18 mai. 2017.

CIRRUS LOGIC. **Multi-channel Audio Hub CODEC for Smartphones**, 2018. Rev. 4.6.

DBX. **Equalizer 231s**. Disponível em: <http://dbxpro.com/en/products/231s>. Acesso em: 18 mai. 2017.

DINIZ, Paulo S, R; SILVA, Eduardo A. B. da; NETTO, Sergio L. Processamento Digital de Sinais: Projeto e Análise de Sistemas. 2. ed. Porto Alegre: Bookman, 2014.

FERNANDES, David. **Equalização III – equalizadores paramétricos**. S.d. Disponível em: < https://musicaeadoracao.com.br/25595/equalizacao-iii-equalizadores-parametricos >. Acesso em: 18 mai. 2017.

GRYTZ, Felipe. **Equalizadores**. Unicamp, 2003. Disponível em: < http://www.iar.unicamp.br/disciplinas/am005_2003/equalizadores.pdf>. Acesso em: 12 mar. 2017

HAYKIN, Simon; VEEN, Barry V. **Sinais e sistemas**. Porto Alegre: Bookman, 2001.

HERRERA, Christian G. **Projeto de sistemas de processamento digital de sinais de áudio utilizando metodologia científica.** 2004. 89 f. Dissertação (Mestre em Engenharia Elétrica) – Programa de Pós Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, 2004.

JUTH, Thomas. **The art of equalization:** how to create better sounding mixes. Berlin, 2016

LATHI, B. P. **Sinais e sistemas lineares.** 2 ed. Porto Alegre: Bookman, 2007.

MALVINO, Albert; BATES, David J. **Eletrônica**: volume 2. 7 ed. Porto Alegre: McGraw Hill, 2007.

NALON, José A. **Introdução ao processamento digital de sinais**. Rio de janeiro: LTC, 2009.

NEIVA. Álvaro C. de. **Filtros e equalizadores**. 2013. Disponível em: <http://www.alvaroneiva.site.br.com/filteq2.htm>. Acesso em: 18 mai. 2017.

NETO, José C. Música Instrumental é tocada para os pacientes do Hospital São Sebastião. **Portal Ternura**, Ibitinga, jun. 2018. Disponível em: <https://www.portalternurafm.com.br/noticias/regional/musica-instrumental-e-tocada-para-os-pacientes-do-hospital-sao-sebastiao/35876>. Acesso em: 08 out. 2018.

REED, Dale. **A perceptual assistant to do sound equalization**. In: 5th international conference on Intelligent user interfaces. Chicago, p. 212-218, 2000.

ROCHA, Adriel da. **Equalizadores e tabela de frequências**. 2012. Disponível em: <http://louvormoria.blogspot.com.br/2012/05/equalizadores-e-tabela-de-frequencias.html>. Acesso em: 18 mai. 2017.

ROSA, Ivon E. E. **Projeto de equalizador paramétrico avançado para aplicações em MP3**. 2012. 69 f. Monografia (Tecnólogo em Sistemas de Telecomunicações) - Curso Superior de Tecnologia em Sistemas de Telecomunicações, Instituto Federal de Santa Catarina, São José, 2012.

SHENOI, B, A. **Introduction to digital signal processing and filter design**. New Jersey: John Wiley, 2006

SILVA, Marco A. G. **Filtros digitais aplicados em sinais de áudio.** 2005. 62 f. Monografia (Ciência da Computação) – Curso de Ciência da Computação, Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora, Juiz de Fora, 2005.

STMICROELECTRONICS. **32F746GDISCOVERY board: user manual**. 2017. Disponível em:< http://www.st.com/resource/en/user_manual/dm00190424.pdf >. Acesso em: 07 jul. 2017.

TERAHATA, T. **Equilibrando frequências**. 2003. Disponível em: <http://www.territoriodamusica.com/preproducao/?c=69>. Acesso em: 17 mai. 2017.

ZÖLZER, Udo. **digital audio signal processing**. 2 ed. John Wiley, 2008.