



**Universidade Federal da Bahia
Escola Politécnica
Colegiado do Curso de Eng. Elétrica**



Gabriel Mendes de Lima Gomes

Desenvolvimento de Unidade de Efeitos de áudio para Guitarra destinado ao Estudo de Processamento Digital de Sinais

Orientador(a): Prof. Dr. Antônio Carlos Lopes Fernandes Júnior

Salvador-Ba - Brasil
12 de dezembro de 2023

Gabriel Mendes de Lima Gomes

**Desenvolvimento de Unidade de Efeitos de áudio para
Guitarra destinado ao Estudo de Processamento Digital de
Sinais**

Monografia apresentada ao Curso de Graduação em Engenharia Elétrica da Universidade Federal da Bahia como parte dos requisitos para a obtenção do grau de Engenheiro(a) Eletricista.

Orientador(a): Prof. Dr. Antônio Carlos Lopes Fernandes Júnior

Salvador-Ba - Brasil

12 de dezembro de 2023

Gabriel Mendes de Lima Gomes

**Desenvolvimento de Unidade de Efeitos de áudio para
Guitarra destinado ao Estudo de Processamento Digital de
Sinais**

Monografia apresentada ao Curso de Graduação em Engenharia Elétrica da Universidade Federal da Bahia como parte dos requisitos para a obtenção do grau de Engenheiro(a) Eletricista.

Trabalho aprovado. Salvador-Ba - Brasil, 12 de dezembro de 2023:

Orientador(a): Prof. Dr. Antônio Carlos Lopes
Fernandes Júnior

Prof. Dr. Paulo César Machado de Abreu Farias

Prof. Dr. Tiago Trindade Ribeiro

Salvador-Ba - Brasil
12 de dezembro de 2023

Este trabalho é dedicado a minha Mãe, minha família e ao meu Avô, onde quer que esteja.

Agradecimentos

Agradeço à minha mãe, por sempre cuidar de mim, estando perto ou não, nas melhores e piores condições, todo o seu esforço me proporcionou as oportunidades necessárias para eu chegar até aqui, tenho muito orgulho de ser seu filho. Aos meus avós, dona Nina e seu Aristóteles, obrigado pelo amor e carinho, quem sou hoje diz muito a respeito de como vocês me criaram. Aos meus padrinhos, meus tios, minha mamãe Marisa e meu Papai Anivaldo, em resumo, à minha família, por sempre me acolher e cuidar, posso dizer de boca cheia que sou um filho de muitas mães e pais, não poderia ter pessoas mais maravilhosas ao meu redor. A Elisabeth, minha companheira, amiga de todas as horas, confidente e meu grande amor, muito obrigado por nunca soltar da minha mão, não consigo mensurar em palavras o quanto você é importante pra mim. Aos meus grandes amigos da vida, Edson, Gustavo e Rocha, pelas risadas, delírios, discussões e parceria ao longo desses anos. Ao PET Elétrica, que me concedeu tantas experiências maravilhosas, me proporcionar amizades que levei pra vida e me apresentar a melhor professora do DEEC e o espelho do que almejo me tornar como pessoa e profissional, Luciana. Ao CTAI, onde conheci pessoas incríveis e enfrentei grandes desafios, não podendo deixar de agradecer a Márcio por acreditar no meu trabalho e valorizá-lo. Aos amigos que fiz na Braskem, pessoas tão carinhosas e verdadeiras, além de profissionais excepcionais, em especial, a Renatinha e Nai, pelas orientações e direcionamentos. Agradeço ao meu orientador, Antônio, por acreditar na minha ideia e nas minhas capacidades. Não posso deixar de agradecer também ao Professor Wellington, quem me auxiliou fortemente neste projeto. Enfim, gostaria de poder citar todas as pessoas que amo e me importo, passaria meses falando sobre cada um e como foi e é importante na minha vida. Se aquela criança desajeitada, que se tornou um adolescente relapso chegou até aqui, muito se deve ao carinho que recebi de todos vocês.

*“Eu sigo naquela fé, que talvez não mova montanhas, mas arrasta multidões e esvazia
camburões, preenche salas de aula e corações vazios”*
(Djongá)

Resumo

A ideia central deste projeto está baseada na utilização da placa de desenvolvimento STM32F407DISCOVERY como uma unidade de efeitos, também denominada pedaleira de guitarra, para processar o áudio com qualidade e reproduzir efeitos clássicos como Distortion, Delay, Tremolo e Flanger. Inicialmente é contextualizado o cenário em que surgiram os pedais de efeito, sua importância e como se transformaram com a evolução tecnológica. Em seguida, conceitos básicos do processamento digital de sinais são discutidos, para fornecer os conhecimentos necessários para realização deste projeto. No que diz respeito ao desenvolvimento da pedaleira, são tratados separadamente os blocos de Hardware necessário para o processamento do áudio e em seguida, é introduzido o método por trás da aquisição de dados através da pedaleira, bem como os efeitos que serão implementados via software. Após toda discussão teórica, é apresentado o processo realizado na confecção deste projeto, passando pela metodologia escolhida, as dificuldades encontradas e o levantamento do custo total. Por fim, é feita a avaliação dos resultados obtidos, fazendo a leitura do sinal proveniente da pedaleira desenvolvida, observando a qualidade do processamento e fidedignidade dos efeitos quando comparados com os seus pares teóricos.

Palavras-chave: STM32F407DISCOVERY, Processamento digital de sinais, efeitos de guitarra, PCM1802, PCM5102.

Abstract

The central idea of this project is based on using the STM32F407DISCOVERY development board as an effects unit, also known as a guitar pedal, to process audio with quality and reproduce classic effects such as Distortion, Delay, Tremolo and Flanger. Initially, the scenario in which effects pedals emerged, their importance and how they have changed with technological evolution is put into context. Next, basic concepts of digital signal processing are discussed, to provide the knowledge needed to carry out this project. As far as the development of the pedalboard is concerned, the hardware blocks needed for audio processing are dealt with separately, and then the method behind data acquisition via the pedalboard is introduced, as well as the effects that will be implemented via software. After all the theoretical discussion, the process of making this project is presented, including the methodology chosen, the difficulties encountered and the total cost survey. Finally, the results obtained are evaluated by reading the signal from the pedalboard developed, observing the quality of the processing and the reliability of the effects when compared to their theoretical counterparts.

Keywords: STM32F407DISCOVERY, Digital signal processing, guitar effects, PCM1802, PCM5102.

Listas de ilustrações

Fig. 1 – Sinal Discreto no Tempo.	29
Fig. 2 – Sistema de tempo discreto.	30
Fig. 3 – Amostragem de Sinal de Tempo Contínuo.	35
Fig. 4 – Representação no Domínio da Frequência da Amostragem.	36
Fig. 5 – Reconstrução do Sinal a Partir de Suas Amostras.	37
Fig. 6 – Transformada de Fourier do Sinal Amostrado. (a) $\Omega_s > 2\Omega_N$. (b) $\Omega_s < 2\Omega_N$.	37
Fig. 7 – Processamento Digital de Sinais Analógicos.	38
Fig. 8 – Representação de um Sample-and-Hold Ideal.	39
Fig. 9 – Curva que Relaciona Entrada e Saída Quantizada.	40
Fig. 10 – Barramento de Comunicação I2C.	41
Fig. 11 – Formas de Onda do Protocolo I2S.	42
Fig. 12 – Transferência de dados Utilizando DMA.	42
Fig. 13 – Diagrama de Blocos do Pedal.	43
Fig. 14 – Circuito de Alimentação e Polarização para os Amplificadores Operacionais.	44
Fig. 15 – Circuito de Alimentação Microcontrolador.	44
Fig. 16 – Circuito de Pré-Amplificação.	45
Fig. 17 – Circuito Amplificador Não Inversor.	46
Fig. 18 – Resposta em Frequência de um Filtro Passa-Baixa Butterworth.	47
Fig. 19 – Circuito Filtro Passa-Baixa Sallen And Key.	48
Fig. 20 – Circuito Filtro Anti-Aliasing.	49
Fig. 21 – Resposta em Frequência: Teórico x Simulação.	50
Fig. 22 – Módulo Conversor Analógico para Digital PCM1802.	51
Fig. 23 – Módulo Conversor Digital para Analógico PCM5102.	52
Fig. 24 – Circuito Filtro de Reconstrução.	53
Fig. 25 – Placa de Desenvolvimento STM32F4DISCOVERY.	54
Fig. 26 – Módulo LCD com Comunicação I2C.	55
Fig. 27 – Diagrama de Comunicação da Pedaleira.	56
Fig. 28 – Fluxo de Dados do Ping-Pong Buffer.	57
Fig. 29 – Buffer Circular e Interrupção DMA.	58
Fig. 30 – Curva Característica do Overdrive.	59
Fig. 31 – Espectro do Overdrive.	60
Fig. 32 – Curva Característica do Distortion.	61
Fig. 33 – Espectro do Distortion.	61
Fig. 34 – Diagrama de blocos do Delay FIR.	62
Fig. 35 – Resposta em Frequência do Delay FIR.	63
Fig. 36 – Diagrama de blocos do Tremolo.	63

Fig. 37 – Resposta Típica do Efeito de Tremolo.	64
Fig. 38 – Diagrama de blocos do Flanger.	65
Fig. 39 – Resposta Típica do Efeito de Flanger.	65
Fig. 40 – Esquemático de Ligação do STM32 e Conversores A/D e D/A.	68
Fig. 41 – Circuito da Pedaleira Montado na Protoboard.	69
Fig. 42 – Layout da Pedaleira.	69
Fig. 43 – Placa de circuito impresso.	70
Fig. 44 – Pedaleira Desenvolvida.	70
Fig. 45 – Comparação dos Sinais de Entrada e Saída.	73
Fig. 46 – Forma de Onda do Sinal de Entrada.	74
Fig. 47 – Espectrograma do Sinal de Entrada.	74
Fig. 48 – Forma de Onda do Distortion.	75
Fig. 49 – Espectrograma do Distortion.	75
Fig. 50 – Forma de Onda do Delay.	76
Fig. 51 – Espectrograma do Delay.	77
Fig. 52 – Forma de Onda do Tremolo.	78
Fig. 53 – Espectrograma do Tremolo.	78
Fig. 54 – Forma de Onda do Flanger.	79
Fig. 55 – Espectrograma do Flanger.	79
Fig. 56 – Esquemático Fonte de Alimentação.	87
Fig. 57 – Esquemático Pré-Amplificador.	88
Fig. 58 – Esquemático Filtro Anti-Aliasing.	89
Fig. 59 – Esquemático Filtro de Reconstrução.	90
Fig. 60 – Esquemático Microcontrolador e Periféricos.	91
Fig. 61 – Layout Final da Pedaleira.	92

Lista de tabelas

Tab. 1 – Tabela de Conexões PCM1802.	51
Tab. 2 – Tabela de Conexões PCM5102.	52
Tab. 3 – Tabela de Conexões LCD.	55
Tab. 4 – Tabela de Custos da Pedaleira	71

Lista de abreviaturas e siglas

A/D	Analógico para Digital
BCK	Bit Clock Line
CA	Corrente Alternada
CC	Corrente Contínua
C/D	Contínuo para Discreto
CD	Compact Disc
DSP	Digital Signal Processor
D/A	Digital para Analógico
DOUT	Data Out
DIN	Data In
DMA	Direct Memory Access
FPU	Floating Point Unit
FIR	Finite Impulse Response
HAL	Hardware Abstraction Layer
I2S	Inter-IC Sound
I2C	Inter-Integrated Circuit
HIR	Infinite Impulse Response
LRCLK	Left-Right Clock
LCD	Liquid Crystal Display
LFO	Low Frequency Oscillation
LIT	Lineares e Invariantes no Tempo
PCI	Placa de Circuito Impresso
SDA	Serial Data Line
SCL	Serial Clock Line
SCK	Serial Clock
SD	Serial Data
TFT	Thin Film Transistor
WS	Word Select

Listas de símbolos

Ω_s	Frequência de Amostragem (rad/s)
ω_s	Frequência de Amostragem (rad/amostra)
T	Período de Amostragem (s)
f_s	Frequência de Amostragem (Hz)
f_c	Frequência de Corte (Hz)
Δ	Passo de Quantização
R	Resistor (Ω)
C	Capacitor (F)
Z	Impedância (Ω)
Ω_0	Frequência de Ressonância (rad/s)
Q	Fator de Qualidade
K_0	Ganho na Banda de Passagem
D	Atraso em Amostras
a	Amplitude
e	Número de Euler
s	Frequência Complexa
z	Frequência Angular Complexa

Sumário

1	INTRODUÇÃO	25
1.1	Objetivos	26
1.1.1	Objetivos Gerais	26
1.1.2	Objetivos Específicos	26
1.2	Estrutura do Trabalho	27
2	FUNDAMENTAÇÃO TEÓRICA	29
2.1	Sinais de Tempo Discreto	29
2.2	Sistemas de Tempo Discreto	30
2.3	Sistemas Lineares e Invariantes no Tempo (LIT)	30
2.4	Propriedades dos Sistemas Lineares Invariantes no Tempo	31
2.4.1	Conexão em Cascata	32
2.4.2	Conexão em Paralelo	32
2.4.3	Estabilidade Bounded-Input/Bounded-Output (BIBO)	32
2.4.4	Causalidade	33
2.5	Resposta em Frequência de Sistemas em Tempo Discreto	33
2.6	Amostragem Periódica	34
2.6.1	Representação em Frequência	34
2.6.2	Reconstrução de um Sinal de Banda Limitada a Partir de Suas Amostras	35
2.6.3	Teorema de Nyquist-Shannon	36
2.6.4	Efeito de Aliasing	37
2.7	Processamento Digital de Sinais Analógicos	38
2.7.1	Conversão Analógico para Digital (A/D)	38
2.7.2	Conversão Digital para Analógico (D/A)	40
2.7.3	Protocolos de Comunicação: Inter-Integrated Circuit (I2C)	40
2.7.4	Protocolos de Comunicação: Inter-IC Sound (I2S)	41
2.7.5	Acesso Direto à memória (DMA)	42
3	COMPOSIÇÃO DO PROJETO	43
3.1	Hardware	43
3.1.1	Alimentação do Circuito	43
3.1.2	Pré-Amplificador	44
3.1.2.1	Impedância de Entrada	45
3.1.2.2	Ganho de Tensão	45
3.1.2.3	Resposta em Frequência	46
3.1.3	Filtro Anti-aliasing	47

3.1.4	Conversor Analógico para Digital - PCM1802	50
3.1.5	Conversor Digital para Analógico - PCM5102	52
3.1.6	Filtro de Reconstrução	52
3.1.6.1	Impedância de Saída	53
3.1.7	Microcontrolador - STM32F407DISCOVERY	54
3.1.8	Módulo LCD	55
3.2	Software	55
3.2.1	Fluxo de Código	55
3.2.2	Ping-Pong Buffer	56
3.3	Efeitos de Áudio	58
3.3.1	Distorções	58
3.3.1.1	Overdrive	59
3.3.1.2	Distortion	60
3.3.2	Delay	61
3.3.3	Tremolo	63
3.3.4	Flanger	64
4	METODOLOGIA DE IMPLEMENTAÇÃO	67
4.1	Processo de Confecção	67
4.2	Custo do Projeto	71
5	RESULTADOS EXPERIMENTAIS	73
5.1	Distortion	75
5.2	Delay	76
5.3	Tremolo	77
5.4	Flanger	78
6	CONSIDERAÇÕES FINAIS	81
	REFERÊNCIAS	83
	APÊNDICE A – CIRCUITO DESENVOLVIDO	87
	APÊNDICE B – CÓDIGO IMPLEMENTADO	93

1 Introdução

O processo de criação de novos instrumentos é resultado das relações dinâmicas entre construtores, músicos e suas necessidades e demandas ([CASTRO, 2007](#)). O surgimento da guitarra elétrica, bem como dos pedais de efeito, se enquadra perfeitamente neste caso, visto que, nascem da busca por novos sons e maior expressividade musical. A guitarra elétrica aliada aos pedais de efeito, desempenharam papéis significativos na evolução da música ao longo dos anos, moldando estilos musicais e permitindo com que músicos explorassem territórios sonoros antes impensados.

Comumente atribui-se a criação da guitarra à Adolph Rickenbacker e George Beauchamp, em 1934, com inspirações vindas diretamente das guitarras havaianas que naquele momento eram muito populares nos Estados Unidos ([SOUZA, 2002](#)). Entretanto, em outros lugares do mundo, também foram desenvolvidos projetos de guitarra elétrica, no Brasil, mais especificamente na Bahia, por volta de 1940, Dodô criava o “Pau Elétrico”, inspirado no violão elétrico utilizado pelo violonista clássico Benedito Chaves. O instrumento desenvolvido por Dodô possuía um braço de cavaco unido a um pedaço de madeira com um captador elétrico, sendo crucial para criação do gênero musical frevo baiano ([VARGAS, 2014](#)).

Mesmo com toda a sua importância e versatilidade, após anos de sua criação, a guitarra elétrica apenas ganhou status e notoriedade na cultura pop, no século XX, e muito em função da icônica performance do guitarrista Jimi Hendrix no Monterey International Pop Festival de 1967, na Califórnia, onde imortalizou o ritual de atear fogo na sua Fender Stratocaster ([SOUZA, 2002](#)).

A partir da década de 1960, os pedais de efeito como conhecemos hoje, se tornaram a principal fonte de criação e refinamento de novos sons, trazendo maior versatilidade à sonoridade extraída da guitarra pelos instrumentistas. As primeiras unidades de efeito desenvolvidas na década de 1960 foram os pedais de distorção, que adicionavam “sujeira”, granulação e aumentavam o volume do som da guitarra ([HATSCHEK; MONLEY, 2016](#)). Com o desenvolvimento dos circuitos digitais e subsequente proliferação do microprocessador, passou-se a vislumbrar a possibilidade da implementação DE sistemas capazes de realizar o processamento em tempo discreto de sinais, mesmo que os primeiros microprocessadores fossem muito lentos. A partir da década de 1980, a tecnologia de circuito integrado passou a permitir a confecção de microcomputadores com arquiteturas especialmente projetadas para o desenvolvimento de algoritmos de processamento discreto de sinais de tempo contínuo ([OPPENHEIM; SCHAFER, 2014](#)).

O processamento de sinais em tempo discreto pode ser descrito como a manipulação

de sequências de valores definidos apenas em um conjunto inteiro de instantes de tempo, em vez de funções de um argumento contínuo ([OPPENHEIM; SCHAFER, 2014](#)). hoje, com o estabelecimento das técnicas de processamento digital de sinais e evolução tecnológica, é possível desenvolver, de maneira mais acessível, visto que há uma facilidade maior na aquisição de componentes e o maior compartilhamento de conhecimento, pedais que reproduzam o interesse do músico, introduzindo mais personalidade ao som, além de tornar possível não apenas reproduzir os efeitos de áudio analógico, como também a implementação de mais de um efeito por pedal, nas chamadas pedaleiras.

A partir da década de 1980, muitos guitarristas começaram a mudar de pedais analógicos para as chamadas unidades multi-effects (pedaleiras). A mudança ocorreu principalmente por causa do interesse em ter vários efeitos presentes em uma unidade compacta que pudesse conter predefinições personalizadas de tons ([KAREN, 2020](#)). E com o passar dos anos, a sua versatilidade tem se tornado ainda maior, passando além de abranger uma vasta gama de efeitos, agora também possibilitando a modelagem e simulação de amplificadores, gabinetes, afinadores embutidos e opções de programação avançada, justificando a preferência de muitos músicos, dos mais variados estilos e níveis de habilidade.

1.1 Objetivos

1.1.1 Objetivos Gerais

Com este trabalho, busca-se a confecção de uma pedaleira digital para guitarra, de código aberto, no que diz respeito a software e Hardware, a partir da plataforma de desenvolvimento STM32F407DISCOVERY, por possuir um ambiente de programação amigável ao usuário, reduzindo o tempo de programação de periféricos, podendo focar no estudo e uso da teoria de processamento digital de sinais. A solução deverá ser capaz de replicar de maneira fidedigna, efeitos como Distortion, Delay, Tremolo e Flanger, em alta qualidade, seja de maneira isolada ou em conjunto, assim como em uma pedaleira comercial.

1.1.2 Objetivos Específicos

Os objetivos específicos listados para esta monografia são:

- Realizar revisão teórica dos principais conceitos acerca do processamento digital de sinais, necessários para o desenvolvimento do projeto.
- Confeccionar hardware da etapa de condicionamento de sinais, que contempla o estudo e projeto dos filtros analógicos. Esses filtros serão responsáveis pela adequação do

sinal aos padrões necessários para leitura do conversor Analógico para Digital ([A/D](#)) e adequação do sinal transmitido do conversor Digital para Analógico ([D/A](#)) para o amplificador.

- Estabelecimento da comunicação entre os conversores [A/D](#) e [D/A](#) com o Digital Signal Processor ([DSP](#)). Será realizado o estudo e implementação do protocolo de comunicação Inter-IC Sound ([I2S](#)) para possibilitar a troca de informação entre os dispositivos.
- Implementar os efeitos digitais Distortion, Delay, Tremolo e Flanger. Fazer a leitura do sinal proveniente da pedaleira a partir da placa de áudio de um notebook e avaliá-los quanto à fidedignidade com os efeitos originais teóricos, observando os espectrogramas de cada efeito.

1.2 Estrutura do Trabalho

Esta monografia está organizada de modo a favorecer o entendimento do projeto desenvolvido, facilitando possíveis reproduções por terceiros. No capítulo 2, percorre-se pela base teórica básica necessária para o entendimento do projeto. No capítulo 3, são abordados os blocos que constituem o hardware da pedaleira, apresentando suas principais características, além da descrição do fluxo de código para o processamento digital de áudio e estudo dos efeitos de áudio que se desejam implementar. O capítulo 4, por sua vez, contém o processo de confecção da pedaleira. Já no capítulo 5 são discutidos os resultados experimentais obtidos. Encerrando o trabalho, as considerações finais e as propostas para trabalhos futuros são apresentados no capítulo 6.

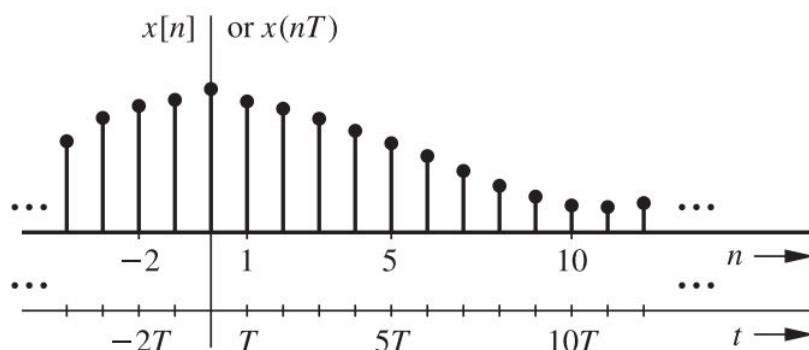
2 Fundamentação Teórica

O princípio básico que rege o desenvolvimento deste projeto é a possibilidade de um sinal em tempo contínuo ser processado, a partir de suas amostras, idealmente por um sistema que opera em tempo discreto, ou seja, onde a variável independente apresenta valores discretos, normalmente representados como uma sequência de número e em seguida expandindo esse conceito para os sistemas digitais onde tratamos de sinais discretos tanto na amplitude quanto no tempo. Visando o entendimento de como isso é possível, nesta seção será realizada uma revisão dos conceitos mais importantes por trás do processamento digital de sinais analógicos.

2.1 Sinais de Tempo Discreto

De maneira básica, descreve-se um sinal de tempo discreto como sendo uma sequência de números. Tais sinais podem ser representados, por exemplo, como $x[n]$, no qual a variável n assume valores inteiros e $x[n]$ representa o n -ésimo número da sequência x ([LATHI, 2007](#)). Ou seja, diferentemente dos sinais de tempo contínuo que são definidos para todos os instantes de tempo, os sinais de tempo discreto só podem ser definidos para instantes múltiplos de n , como pode ser visto na Figura 1.

Figura 1 – Sinal Discreto no Tempo.



Fonte: ([LATHI, 2007](#)).

Os sinais discretos podem ter sua origem a partir de situações que possuem características discretas, como a contegem de eventos ou até mesmo, modelos de renda nacional, porém, o objeto de estudo aqui se restringe às sequências que surgem da amostragem periódica de sinais de tempo contínuo $x_a(t)$. Representando $x_a(t)$ como uma sequência discreta, o n -ésimo número desta sequência, presente no instante nT , será:

$$x[n] = x_a(nT), \quad -\infty < n < \infty \quad (2.1)$$

Dentro da teoria de sinais de tempo discreto, existem diversas sequências básicas muito importantes, dentre elas, uma que irá se destacar aqui é a sequência amostra unitária ou impulso, (2.2), definida como:

$$\delta[n] = \begin{cases} 0, & n \neq 0 \\ 1, & n = 0 \end{cases} \quad (2.2)$$

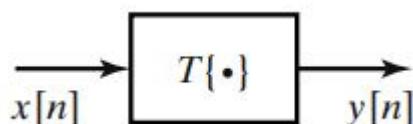
Qualquer sequência arbitrária pode ser representada a partir da sequência amostra unitária, (2.3), bastando apenas realizar o somatório dos impulsos devidamente ponderados e deslocados (OPPENHEIM; SCHAFER, 2014).

$$\begin{aligned} x[n] &= \dots + a_{-1}\delta[n+1] + a_0\delta[n] + a_1\delta[n-1] + a_2\delta[n-2] + \dots \\ x[n] &= \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \end{aligned} \quad (2.3)$$

2.2 Sistemas de Tempo Discreto

Sistemas de tempo discreto podem ser definidos como funções matemáticas capazes de transformar uma sequência de entrada $x[n]$ em uma nova sequência $y[n]$ de saída, onde $y[n] = T\{x[n]\}$, sendo $T\{\cdot\}$ a transformação ou sistema (OPPENHEIM; SCHAFER, 2014). Na Figura 2 está a representação simplificada de um sistema de tempo discreto.

Figura 2 – Sistema de tempo discreto.



Fonte: (OPPENHEIM; SCHAFER, 2014).

2.3 Sistemas Lineares e Invariantes no Tempo (LIT)

Uma classe muito importante dos sistemas de tempo discreto e que merece uma atenção especial, são aqueles definidos como Lineares e Invariantes no Tempo (LIT), visto que várias propriedades e características que facilitam a análise e o projeto de sistemas podem ser aplicadas a eles. Os sistemas lineares são definidos pelo princípio da superposição, que é formado pelas propriedades da aditividade e homogeneidade (OPPENHEIM; SCHAFER, 2014), apresentados nas equações (2.4) e (2.5) respectivamente. Se $y_1[n]$ e $y_2[n]$ são as respostas de um sistema para as entradas $x_1[n]$ e $x_2[n]$, então o sistema será tido como linear se e somente se:

$$\begin{aligned} T\{x_1[n] + x_2[n]\} &= T\{x_1[n]\} + T\{x_2[n]\} \\ T\{x_1[n] + x_2[n]\} &= y_1[n] + y_2[n] \end{aligned} \quad (2.4)$$

e

$$T\{ax[n]\} = aT\{x[n]\} = ay[n] \quad (2.5)$$

A invariância no tempo define os sistemas cujos parâmetros não são alterados com o tempo, ou seja, se a entrada aplicada a um sistema invariante no tempo for atrasada por T segundos, a saída seguirá o mesmo comportamento, sendo atrasada pelos mesmos T segundos (LATHI, 2007). Supondo que um sistema transforme a sequência de entrada $x[n]$ na sequências de saída $y[n]$, o sistema será dito invariante no tempo se para todo atraso n_0 , $x_1[n] = x[n - n_0]$ produz $y_1[n] = y[n - n_0]$. As propriedades de linearidade e invariância no tempo combinadas levam a representações especialmente convenientes para os sistemas (OPPENHEIM; SCHAFER, 2014), nas suas representações e durante a manipulação, especialmente quando tratamos de aplicações direcionadas para o Processamento Digital de Sinais.

Seja $h[n] = T\{\delta[n]\}$, também conhecido como resposta ao impulso, lembrando uma sequência qualquer pode ser representada como uma combinação linear de impulsos atrasados, equação (2.3), a resposta do sistema a qualquer entrada $x[n]$ é:

$$y[n] = T \left\{ \sum_{k=-\infty}^{\infty} x[k] \delta[n-k] \right\} \quad (2.6)$$

Aplicando as propriedades da linearidade para a distribuição da transformação do sistema em cada um dos termos da equação (2.6), observando também que pela invariância no tempo $T\{\delta[n-k]\} = h[n-k]$, a saída de um sistema pode ser completamente caracterizada por:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] h[n-k] \quad (2.7)$$

Da equação (2.7), fica evidente que se conhecemos a resposta impulsiva do sistema, podemos obter a sua saída para qualquer outra entrada. A Equação (2.7) é chamada de soma de convolução, e é representada pela notação $y[n] = x[n] * h[n]$.

2.4 Propriedades dos Sistemas Lineares Invariantes no Tempo

Por serem totalmente descritos pela soma de convolução, os sistemas LIT, tem como propriedades, as mesmas aplicadas a soma de convolução (OPPENHEIM; SCHAFER,

2014):

- Comutatividade:

$$y[n] = x[n] * (h_2[n] * h_1[n]) = (x[n] * h_2[n]) * h_1[n] \quad (2.8)$$

- Distributividade:

$$x[n] * (h_1[n] + h_2[n]) = x[n] * h_1[n] + x[n] * h_2[n] \quad (2.9)$$

- Associatividade:

$$y[n] = (x[n] * h_1[n]) * h_2[n] = x[n] * (h_1[n] * h_2[n]) \quad (2.10)$$

2.4.1 Conexão em Cascata

Conectando dois sistemas com resposta ao impulso $h_1[n]$ e $h_2[h]$, a saída do primeiro sistema é a entrada do segundo sistema e vice versa, podendo expandir essa lógica para um número maior de sistemas se necessário. Essa implicação surge da aplicação das propriedades de comutatividade e associatividade, respectivamente equações (2.8) e (2.10) (OPPENHEIM; SCHAFER, 2014). A resposta ao impulso total equivalente $h[n]$ é:

$$h[n] = h_1[n] * h_2[n] = h_2[n] * h_1[n] \quad (2.11)$$

2.4.2 Conexão em Paralelo

Se dois sistemas LIT com resposta ao impulso $h_1[n]$ e $h_2[h]$ são conectados em paralelo, suas saídas são somadas. Esse resultado surge da aplicação da propriedade distributiva da convolução, equação (2.9), onde a conexão de dois sistemas em paralelo é equivalente a um único sistema cuja resposta ao impulso é a soma das respostas ao impulso individuais (OPPENHEIM; SCHAFER, 2014).

$$h[n] = h_1[n] + h_2[n] \quad (2.12)$$

2.4.3 Estabilidade Bounded-Input/Bounded-Output (BIBO)

Um sistema pode ser considerado estável externamente, se e somente se, para qualquer sequência de entrada limitada em sua amplitude, produzir uma sequência de saída também limitada (LATHI, 2007). Seja $x[n]$ com amplitude limitada, $|x[n]| \leq B_x < \infty$, lembrando que a saída de um sistema LIT é dada por (2.6), chegamos ao seguinte resultado:

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] \quad (2.13)$$

$$|y[n]| \leq \sum_{k=-\infty}^{\infty} |h[k]| |x[n-k]| \leq B_x \sum_{k=-\infty}^{\infty} |h[k]|$$

Portanto, a saída é limitada se o somatório do lado direito da equação (2.13) for limitado.

2.4.4 Causalidade

Para sistemas causais, não é possível usar valores futuros para calcular a saída num dado instante de tempo. Ou seja, a causalidade implica que a saída de um sistema no instante n_0 , $y[n_0]$, não depende de qualquer entrada que ocorra após n_0 , sendo $x[n]$ válido apenas para $n \leq n_0$ (DINIZ; SILVA, 2014).

2.5 Resposta em Frequência de Sistemas em Tempo Discreto

Considerando a entrada $X[n] = e^{j\omega_0 n}$, chamada de exponencial complexa discreta de frequência ω_0 , e aplicando à um sistema LIT:

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k]$$

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]e^{j\omega_0(n-k)} \quad (2.14)$$

$$y[n] = e^{j\omega_0 n} \left(\sum_{k=-\infty}^{\infty} h[k]e^{-j\omega_0 k} \right)$$

Com o resultado obtido na equação (2.14), pode-se definir a equação (2.15), conhecida como resposta em frequência, que descreve a mudança na amplitude de um sinal do tipo exponencial complexa com frequência qualquer ω , quando aplicado a um sistema LIT (OPPENHEIM; SCHAFER, 2014).

$$H(e^{j\omega}) = \sum_{k=-\infty}^{\infty} h[k]e^{-j\omega k} \quad (2.15)$$

É possível reescrever a equação (2.14) em termos da resposta em frequência do sistema, sendo então $y[n] = H(e^{j\omega_0})e^{j\omega_0 n}$. Fica claro então que a saída do sistema é outra exponencial complexa na mesma frequência da entrada, cuja magnitude e fase são modificadas pela função $H(e^{j\omega})$ na frequência $\omega = \omega_0$ (OPPENHEIM; SCHAFER, 2014).

2.6 Amostragem Periódica

Um sinal discreto $x[n]$ pode ser gerado a partir da amostragem de um sinal $x_a(t)$ de tempo contínuo, isto é, $x[n] = x_a(nT)$, onde T é o intervalo de tempo em que o sinal é amostrado, ou seja, o período de amostragem. Seu inverso, f_s , é denominado frequência de amostragem ou taxa de amostragem. Se nossa intenção é processar um sinal de tempo contínuo com um sistema de tempo discreto, é necessário antes de mais nada, realizar a amostragem (DINIZ; SILVA, 2014). O processo de amostragem ideal consiste nas etapas de modulação da entrada com um trem de impulsos, seguido da conversão dos impulsos em amostras (OPPENHEIM; SCHAFER, 2014). A função trem de impulsos periódico é:

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \quad (2.16)$$

De maneira geral, a amostragem não é reversível, sendo assim, não é possível reconstruir o sinal de entrada $x_a(t)$, a partir de suas $x[n]$, exceto caso o conteúdo em frequência do sinal seja limitado em banda (OPPENHEIM; SCHAFER, 2014). Para contornar este inconveniente, define-se $x_c(t)$ como a versão de banda limitada de $x_a(t)$. Pode-se representar o sinal amostrado de tempo contínuo, $x_i(t)$, através do produto de $s(t)$ e $x_c(t)$, visto na equação (2.17).

$$\begin{aligned} x_i(t) &= x_c(t) \cdot s(t) = x_c(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) \\ x_i(t) &= \sum_{n=-\infty}^{\infty} x_c(nT) \delta(t - nT) \end{aligned} \quad (2.17)$$

A Figura 3 representa um sinal de tempo contínuo $x_a(t)$ e o resultado da sua amostragem com um trem de impulsos com período $T = 1/f_s$.

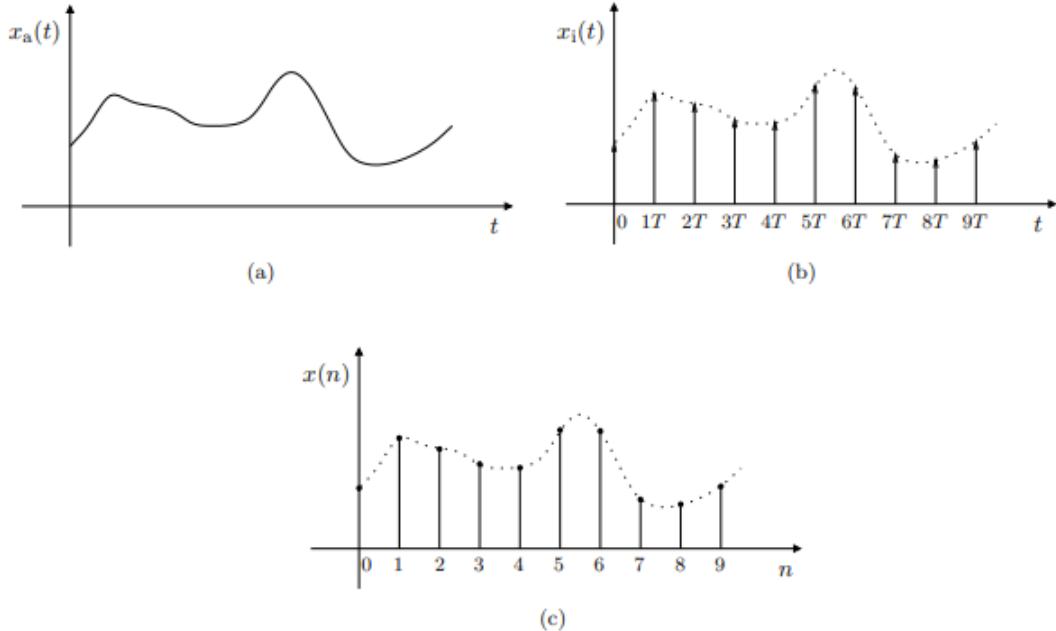
2.6.1 Representação em Frequência

Para obter a relação no domínio da frequência entre a entrada e a saída de um conversor Contínuo para Discreto (C/D) ideal, devemos partir da transformada de Fourier do sinal $s(t)$.

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \leftrightarrow S(j\Omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta(j\Omega - k\Omega_s) \quad (2.18)$$

Na equação (2.18), Ω_s é denominado frequência de amostragem, em radianos por segundo, sendo relacionada com o período de amostragem a partir de $\Omega_s = 2\pi/T$. Das propriedades da transformada de Fourier, deve-se lembrar que a multiplicação no domínio do tempo equivale a convolução no domínio da frequência, sendo assim, o produto de $x_c(t)$

Figura 3 – Amostragem de Sinal de Tempo Contínuo.



Fonte: ([DINIZ; SILVA, 2014](#)).

e $s(t)$, equivale a convolução das transformadas de Fourier $X_c(j\Omega)$ e $S(j\Omega)$, a então a transformada de Fourier de $x_i(t)$ ([OPPENHEIM; SCHAFER, 2014](#)).

$$\begin{aligned} x_i(t) &= x_c(t) \cdot s(t) \Leftrightarrow X_i(j\Omega) = \frac{1}{2\pi} X_c(j\Omega) * S(j\Omega) \\ X_i(j\Omega) &= \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c(j\Omega - k\Omega_s) \end{aligned} \quad (2.19)$$

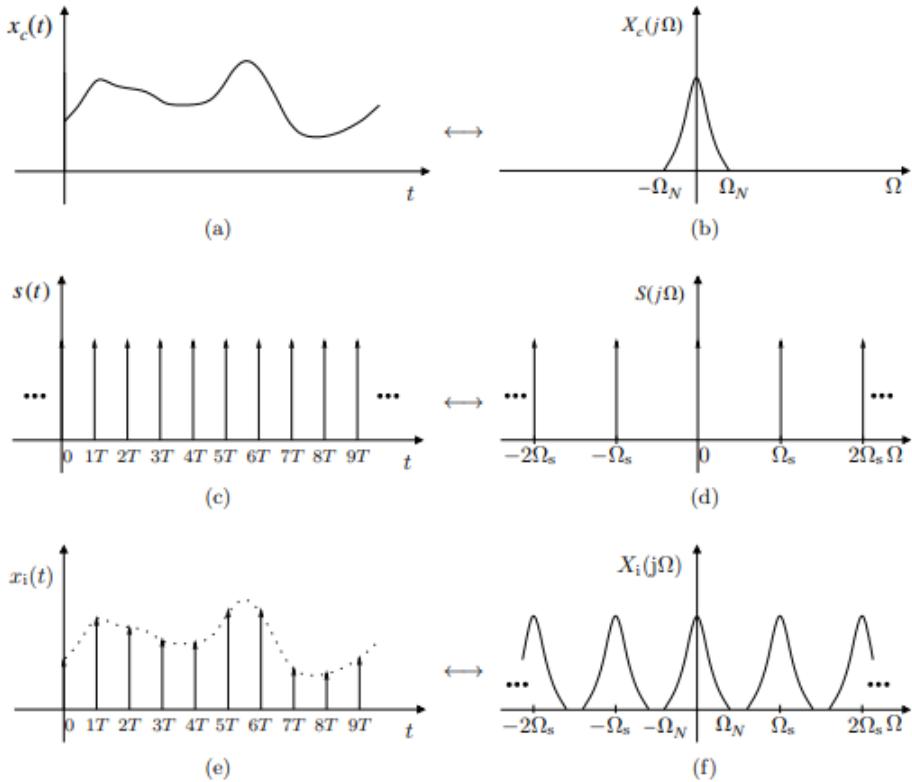
A equação (2.19) apresenta a relação entre as transformadas de Fourier da entrada e saída do modulador de trem de impulsos, desse resultado, entende-se que o espectro do sinal amostrado, $X_i(j\Omega)$, consiste de cópias do espectro do sinal $X_c(j\Omega)$, repetidas a cada Ω_s rad/s e escalonadas por $1/T$, como pode ser visto na Figura 4.

2.6.2 Reconstrução de um Sinal de Banda Limitada a Partir de Suas Amostras

Segundo o teorema da amostragem, se um sinal $x_c(t)$ de tempo contínuo tem largura de banda limitada, isto é, sua transformada de Fourier é definida como $X_c(j\Omega) = 0$ para $|\Omega| > \Omega_N$, ao ser amostrado, $x_c(t)$ pode ser completamente recuperado a partir do sinal de tempo discreto $x[n] = x_c(nT)$, se a frequência de amostragem Ω_s satisfizer a condição $\Omega_s > 2\Omega_N$ ([DINIZ; SILVA, 2014](#)).

O processo de reconstrução do sinal amostrado, Figura 5, se dá a partir da conversão das amostras novamente em um trem de impulsos e sua filtragem com um filtro passa-baixa apropriado. Se o sinal atende às condições do teorema da amostragem e o processo de

Figura 4 – Representação no Domínio da Frequênciada Amostragem.



Fonte: ([DINIZ; SILVA, 2014](#)).

reconstrução é realizado, a transformada de Fourier de saída do filtro será idêntica à transformada de Fourier do sinal de tempo contínuo original $x_c(t)$.

2.6.3 Teorema de Nyquist-Shannon

Nas sessões anteriores, foram definidas quais condições precisam ser atendidas para a correta recuperação do sinal de tempo contínuo a partir de suas amostras. Essas condições são estabelecidas pelo teorema de Nyquist-Shannon. Este teorema determina que um sinal de banda limitada $x_c(t)$, será unicamente descrito por suas amostras $x[n] = x_c(nT)$, $-\infty < n < +\infty$, se e somente se:

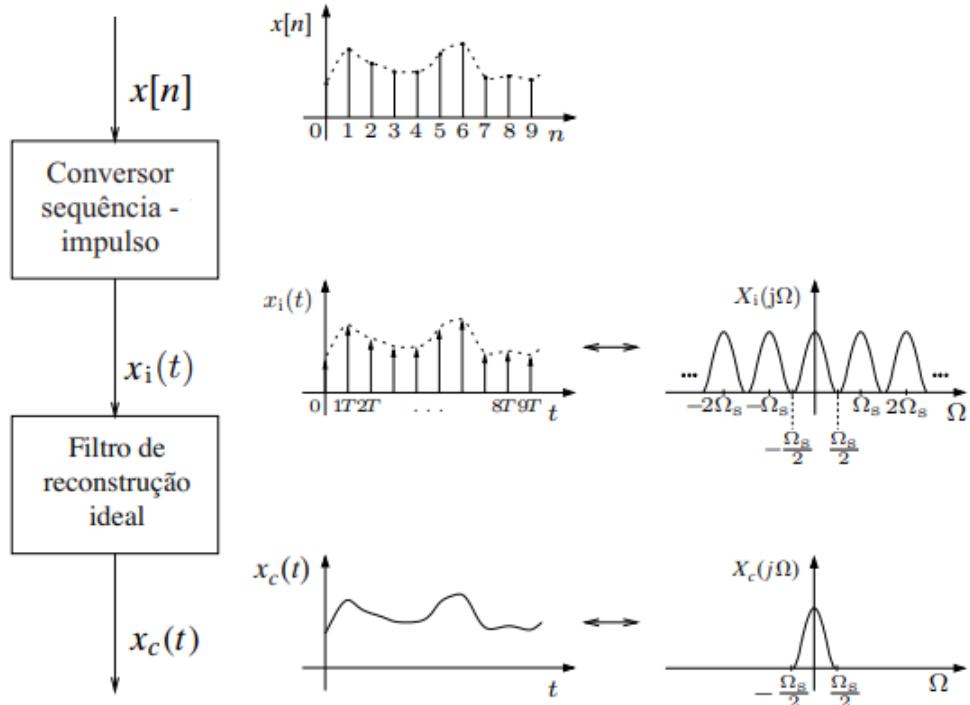
$$\Omega_s - \Omega_N > \Omega_N \quad (2.20)$$

Ou seja,

$$\Omega_s = \frac{2\pi}{T} > 2\Omega_N \quad (2.21)$$

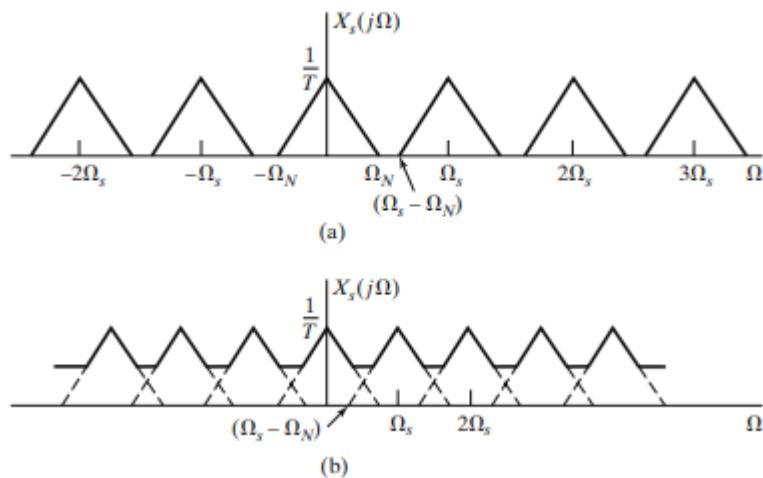
Fica evidente que para ser possível recuperar o sinal original limitado em banda, a frequência de amostragem deve ser maior que duas vezes a frequência máxima do sinal amostrado, também chamada de Frequência de Nyquist ([OPPENHEIM; SCHAFER, 2014](#)).

Figura 5 – Reconstrução do Sinal a Partir de Suas Amostras.

Fonte: ([DINIZ; SILVA, 2014](#)).

2.6.4 Efeito de Aliasing

Durante o processo de amostragem, caso a frequência de amostragem não seja suficientemente alta, ao menos duas vezes maior que a frequência máxima do sinal ou se o sinal de tempo contínuo não for limitado em banda de frequência, teremos como consequência a sobreposição do espectro, Figura 6, que é chamado de aliasing. Nessa situação componentes de alta frequência no sinal são erroneamente interpretadas como componentes de baixa frequência ([LATHI, 2007](#)).

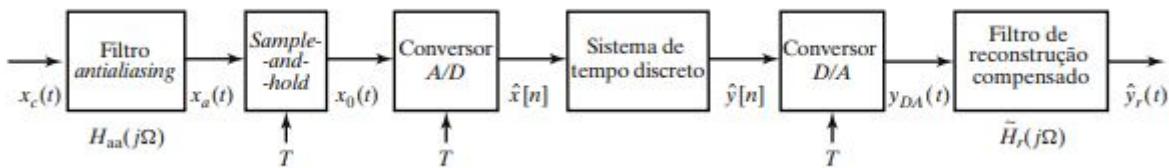
Figura 6 – Transformada de Fourier do Sinal Amostrado. (a) $\Omega_s > 2\Omega_N$. (b) $\Omega_s < 2\Omega_N$.Fonte: ([OPPENHEIM; SCHAFER, 2014](#)).

A solução para o aliasing é alcançada com a implementação de um filtro passa-baixa antes da aquisição dos dados, que tem por função limitar a máxima frequência do sinal à metade da frequência de amostragem. Este filtro é chamado de filtro Anti-Aliasing ([LATHI, 2007](#)).

2.7 Processamento Digital de Sinais Analógicos

Como discutido nas seções anteriores, sinais de tempo contínuo reais, não são exatamente limitados em banda, além disso, os filtros ideias utilizados na limitação de banda ou na reconstituição do sinal amostrado não podem ser realizados. Por fim, em uma próxima etapa da etapa de processamento, a conversão do tempo contínuo para o discreto é ainda limitada por um número finito de bits disponíveis para o sinal digital, fazendo com que as amostras tenham valores iguais ou ligeiramente diferentes dos reais nos instantes de amostragem. Na prática, os processos de conversão de sinais do tempo contínuo para o tempo discreto e vice-versa, podem ser aproximados apenas por dispositivos chamados conversores [A/D](#) e [D/A](#), nesta seção iremos nos debruçar um pouco mais sobre esses conversores. A Figura 7 representa um modelo mais realista do processamento digital de sinais analógicos ([OPPENHEIM; SCHAFER, 2014](#)).

Figura 7 – Processamento Digital de Sinais Analógicos.



Fonte: ([OPPENHEIM; SCHAFER, 2014](#)).

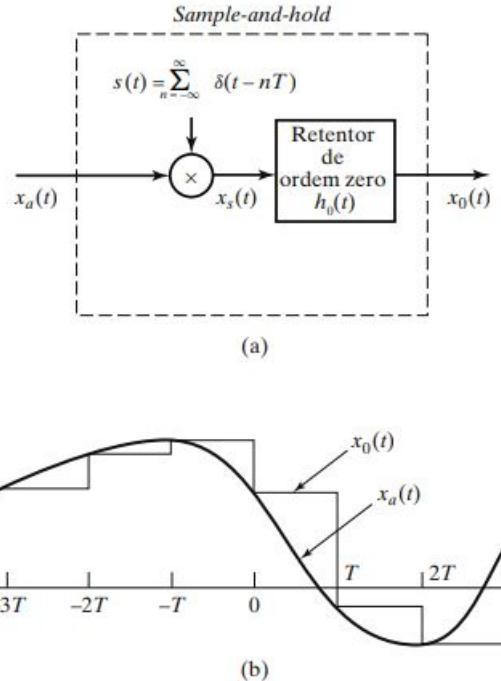
2.7.1 Conversão Analógico para Digital (A/D)

Um sinal analógico é caracterizado pelo fato de sua amplitude poder assumir qualquer valor em uma faixa contínua, apenas a sua amostragem não irá resultar em um sinal digital, pois a amostra de um sinal analógico pode assumir, ainda, qualquer valor em uma faixa contínua. O processo de conversão em um sinal digital se dá através da amostragem (Sample and Hold) e arredondamento (Quantização/Codificação) ([LATHI, 2007](#)).

A etapa de Sample-and-Hold, amostra o sinal $x_a(t)$ nos instantes $t = nT$ e mantém o valor obtido por T segundos, isto é, até que o valor referente ao próximo intervalo de tempo seja amostrado. Mais precisamente, $x_0(t) = x_a(nT)$ para $nT \leq t < (n + 1)T$

(DINIZ; SILVA, 2014). Na Figura 8 temos a representação do processo de amostragem com Sample-and-hold.

Figura 8 – Representação de um Sample-and-Hold Ideal.



Fonte: (OPPENHEIM; SCHAFER, 2014).

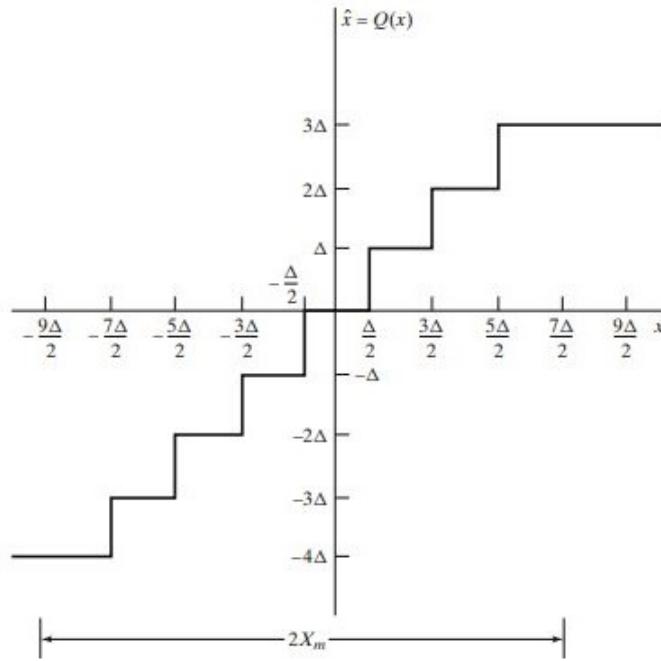
Na etapa de quantização, é realizada a conversão do valor de cada amostra $x_0(t)$ de saída do bloco de sample-and-hold para $nT \leq t < (n + 1)T$ em um número $x[n]$. O número obtido como resultado será enviado para um hardware digital, e por conta disso, deve ser representado por um número finito de bits através da quantização e codificação. Essa operação pode introduzir um erro no sinal, tão menor quanto maior o número de bits empregados na representação (DINIZ; SILVA, 2014). O quantizador atua convertendo um sinal de tempo discreto em um sinal digital finito de valores predeterminados (OPPENHEIM; SCHAFER, 2014). Essa operação é representada pela equação (2.22).

$$\hat{x} = Q(x[n]) \quad (2.22)$$

Na Figura 9 é possível observar as características de um quantizador uniforme com três bits, que arredonda os valores de amostras $x[n]$ para o nível mais próximo de digitalização.

Após o sinal ser quantizado, passamos para a etapa de codificação, que relaciona a amplitude do sinal quantizado com um código binário. A relação entre as palavras de código e os níveis do sinal quantizado depende do parâmetro X_m , esse parâmetro determina o nível de fundo de escala do conversor A/D. Se o conversor A/D possibilitar uma excursão do sinal de entrada entre os limites X_m e $-X_m$, considerando um quantizador de $B + 1$

Figura 9 – Curva que Relaciona Entrada e Saída Quantizada.



Fonte: ([OPPENHEIM; SCHAFER, 2014](#)).

bits e o código em complemento de dois, os passos de quantização terão sua amplitude dada pela equação (2.23).

$$\Delta = \frac{2X_m}{2^{B+1}} = \frac{X_m}{2^B} \quad (2.23)$$

2.7.2 Conversão Digital para Analógico (D/A)

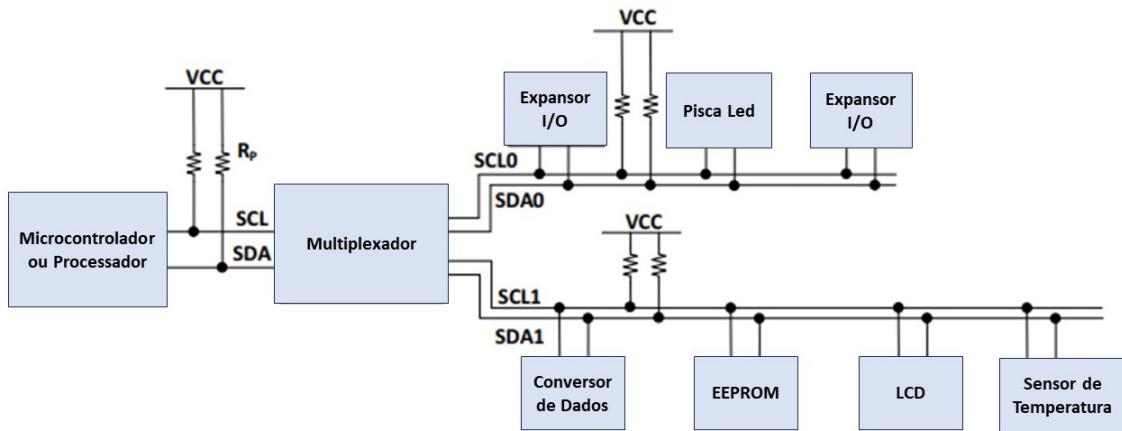
Após o processamento digital do sinal, o conversor D/A toma uma sequência de palavras de código binárias como entrada e produz uma saída de tempo contínuo na forma de um trem de pulsos $y_{DA}(t)$. Este trem de pulsos é então processado por um filtro analógico passa-baixa não ideal (Filtro de Reconstrução), necessário para eliminar as repetições do espectro contidas em $y_{DA}(t)$, a fim de recuperar o sinal analógico correspondente a $\hat{y}[n]$.

2.7.3 Protocolos de Comunicação: Inter-Integrated Circuit (I2C)

A comunicação utilizando o barramento Inter-Integrated Circuit (I2C) é muito popular e poderosa, tendo como principal benefício o fato de permitir que múltiplos dispositivos possam ser conectados a um mesmo barramento e possibilitar que eles troquem dados. A comunicação I2C é bidirecional, porém half-duplex, sendo que um dispositivo é designado como mestre e os demais dispositivos são considerados subordinados. O mestre envia uma solicitação de dados para o subordinado desejado, e ele responde enviando os dados solicitados. Isso é possível graças ao uso de endereços únicos para cada dispositivo

subordinado, que permitem que o mestre saiba exatamente qual dispositivo deve ser acessado. A interface física I₂C é constituída por duas linhas de transmissão, a Serial Data Line (**SDA**) responsável pela transmissão de dados bidirecionalmente e a Serial Clock Line (**SCL**) com o clock gerado pelo mestre (VALDEZ; BECKER, 2015). Uma possível configuração de barramento I₂C é apresentada na Figura 10.

Figura 10 – Barramento de Comunicação I₂C.



Fonte: Adaptado de (VALDEZ; BECKER, 2015).

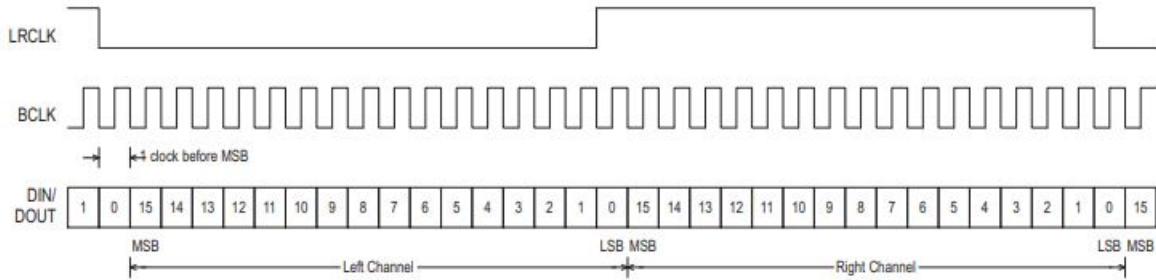
2.7.4 Protocolos de Comunicação: Inter-IC Sound (I₂S)

O protocolo I₂S é o formato de áudio digital mais comumente utilizado na transferência de dados de áudio entre dois dispositivos de áudio digital. Um barramento I₂S normalmente faz uso de três linhas para transferência de dados (LEWIS, 2012):

- Serial Clock (**SCK**): Utilizado para a sincronização entre transmissor e receptor dos dados. É conhecido também como Bit Clock Line (**BCK**).
- Word Select (**WS**): Responsável por indicar se os dados transmitidos referem-se ao canal direito (quando em nível alto) ou esquerdo (quando em nível baixo). Pode ser denominado como Left-Right Clock (**LRCLK**).
- Serial Data (**SD**): Contém os dados multiplexados transacionado entre os dispositivos de áudio. Também pode ser chamado de Data Out (**DOUT**) ou Data In (**DIN**) a depender do tipo do dispositivo.

Cada amostra do sinal de áudio transmitida no protocolo I₂S contém um par de amostras dos canais esquerdo e direito. O canal esquerdo é transferido durante meio período de **WS**, enquanto ele está em nível baixo, já o canal direito, é transferido durante a outra parcela do meio período, agora quando **WS** está em nível alto. Consequentemente, a frequência **WS** é equivalente à taxa de amostragem de áudio (ST, 2008). A Figura 11, apresenta um exemplo de transmissão de uma palavra de 16 bit com o protocolo I₂S.

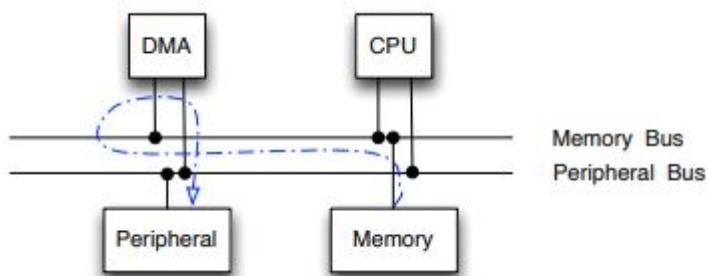
Figura 11 – Formas de Onda do Protocolo I2S.

Fonte: ([BURNS, 2010](#)).

2.7.5 Acesso Direto à memória (DMA)

O controlador Direct Memory Access (**DMA**) tem por função transferir dados de um endereço para outro, entre periféricos e a memória principal, sem a intervenção do processador. O uso de **DMA** possibilita a transferência de dados em alta velocidade entre periféricos e memória, bem como de memória para memória. Os dados podem ser movidos rapidamente pelo controlador **DMA** sem nenhuma ação da CPU. Isso mantém os recursos da CPU livres para outras operações. Por exemplo, o controlador DMA pode mover dados de um periférico, como o conversor **A/D**, para a memória RAM diretamente. O controlador **DMA** é implementado em processadores com dispositivos de hardware dedicado ([INSTRUMENTS, 2018](#)). Nestes dispositivos o barramento de memória e de periféricos são compartilhado com o processador. No exemplo ilustrado na Figura 12, é feita a leitura da memória a partir do barramento de memória, em seguida o dado lido é gravado pelo controlador DMA em um periférico através do barramento de periféricos ([BROWN, 2016](#)).

Figura 12 – Transferência de dados Utilizando DMA.

Fonte: ([BROWN, 2016](#)).

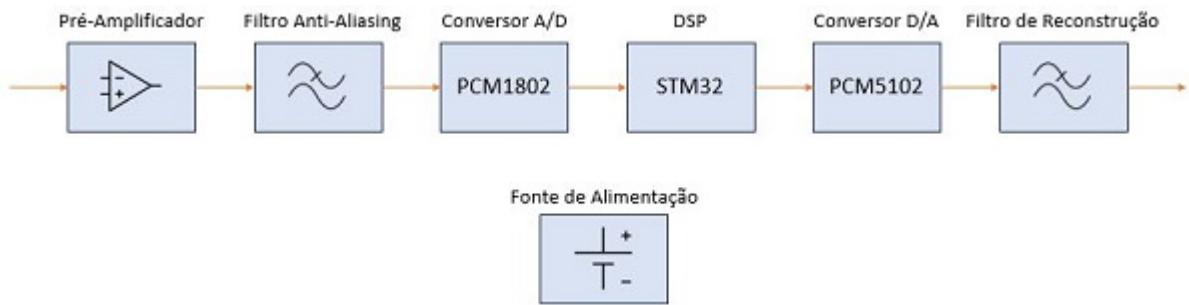
3 Composição do Projeto

Este capítulo é destinado ao entendimento dos principais elementos que compõem o projeto, o hardware e o software. O hardware é responsável pela implementação das funcionalidades do pedal, condicionamento e processamento do sinal proveniente da guitarra, já o software, produz os efeitos de áudio e permite a interação com o usuário.

3.1 Hardware

Antes da realização do processamento de áudio, são necessárias algumas etapas responsáveis por adequar o sinal da guitarra às melhores condições para o **DSP** e amplificador. Essa etapa é comumente chamada de condicionamento de sinal. O Hardware pode ser dividido nos blocos apresentados na Figura 13.

Figura 13 – Diagrama de Blocos do Pedal.



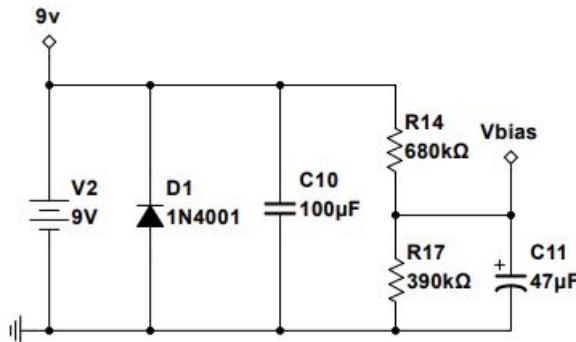
Fonte: Autor.

3.1.1 Alimentação do Circuito

A pedaleira é composta por diversos circuitos alimentados com tensões distintas, entretanto a fonte de alimentação utilizada fornece apenas *9 Volts*, que é um padrão comercial amplamente utilizados nos pedais e pedaleiras atuais. Sendo assim, dois circuitos foram projetados para converter a tensão de *9 Volts* em tensões de *5 e 3,3 Volts*. No primeiro circuito, apresentado na Figura 14, os capacitores de C_{10} e C_{11} são responsáveis por remover toda ondulação da tensão de alimentação, já o diodo $D1$ protege o pedal contra inversões de polaridade durante a alimentação do circuito. Por questões de praticidade, os resistores R_{14} e R_{17} compõem um divisor de tensão que produz os *3,3 Volts* necessários para a polarização dos amplificadores operacionais.

Para a alimentação do microcontrolador, foi desenvolvido de maneira experimental um circuito capaz de fornecer uma tensão de *5 Volts* estabilizada, além de proporcionar a corrente adequada para o seu funcionamento e dos periféricos utilizados. No circuito da

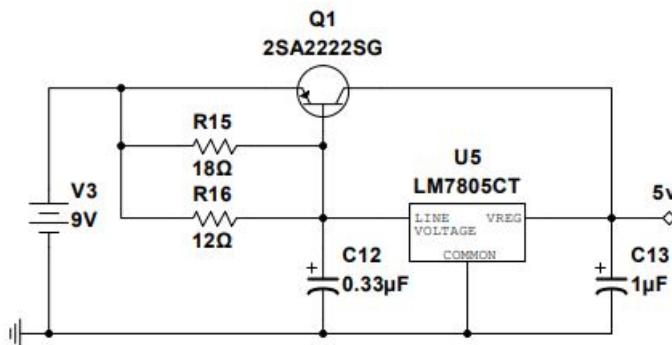
Figura 14 – Circuito de Alimentação e Polarização para os Amplificadores Operacionais.



Fonte: Autor.

Figura 15, os capacitores C_{12} e C_{13} também atuam atenuando ondulações provenientes da tensão de alimentação, já o LM7805 é o regulador de tensão linear responsável por fornecer os 5 Volts de interesse. Por último e não menos importante, o circuito composto pelo transistor NPN $Q1$ e os resistores R_{15} e R_{16} , fornece a corrente solicitada pelo microcontrolador, reduzindo o esforço solicitado ao LM7805 e diminuindo o aquecimento do circuito de alimentação.

Figura 15 – Circuito de Alimentação Microcontrolador.



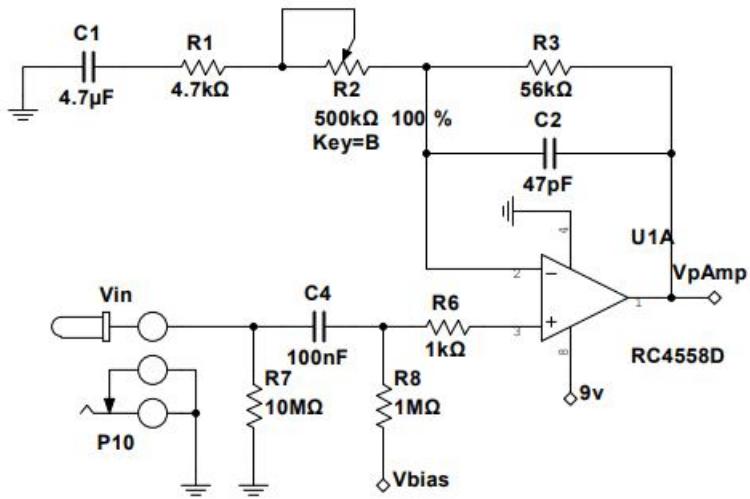
Fonte: Autor.

3.1.2 Pré-Amplificador

O circuito de Pré-Amplificador fornece alta impedância de entrada para o pedal, ganho de tensão, além da primeira filtragem ao sinal lido.

No circuito da Figura 16, o amplificador operacional está configurado na topologia não inversora, onde os resistores R_1 , R_2 e R_3 são responsáveis por definir o ganho de tensão. Já os resistores R_6 , R_7 e R_8 , contribuem com a impedância de entrada do pedal, sendo que o resistor R_8 também fornece a tensão contínua, necessária para a polarização do amplificador operacional. Por fim, os capacitores C_1 , C_2 e C_4 basicamente irão filtrar o sinal.

Figura 16 – Circuito de Pré-Amplificação.



Fonte: Autor.

3.1.2.1 Impedância de Entrada

A partir de uma rápida análise de Corrente Alternada ([CA](#)) dos circuitos, normalmente utilizada na análise de circuitos de áudio, onde as fontes de Corrente Contínua ([CC](#)) devem ser consideradas como terra e os capacitores como curtos-circuitos, fica visível que a impedância de entrada é dada pela equação [\(3.1\)](#). Para a impedância de entrada do *RC4558* foi utilizado o valor típico de $5M\Omega$ definido no datasheet do componente ([INSTRUMENTS, 2014](#)).

$$Z_{In} = (R_7 // R_8) // (R_6 + Z_{InAmpOp}) \quad (3.1)$$

$$Z_{In} \approx 770k\Omega$$

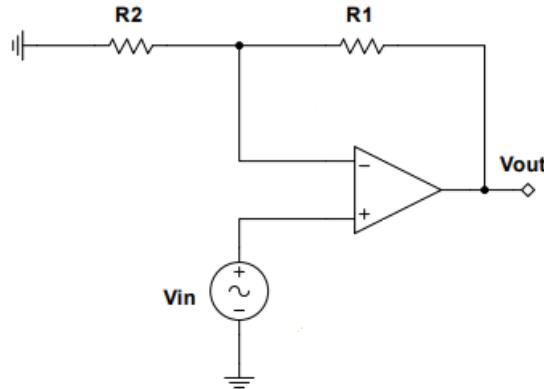
O valor obtido para Z_{In} foi próximo à impedância de entrada de pedais clássicos como MXR MicroAmp da Dunlop ([DUNLOP, 1984](#)), que é de $1M\Omega$, ou seja, o resultado obtido aqui para a impedância de entrada parece ser alta o suficiente para proteger os captadores da guitarra, evitando que correntes maiores do que os limites suportados por eles sejam solicitadas.

3.1.2.2 Ganho de Tensão

O amplificador na topologia não-inversora, ilustrado na Figura [17](#), é composto por um amplificador operacional e um divisor de tensão que transporta até o seu terminal não inversor, uma fração da tensão de saída ([RAZAVI, 2010](#)).

Considerando o amplificador operacional ideal e aplicando análise nodal no circuito da Figura [17](#), fica evidente que o ganho de tensão depende apenas da razão entre os valores dos resistores [\(3.2\)](#).

Figura 17 – Circuito Amplificador Não Inversor.



Fonte: Autor.

$$G_v = 1 + \frac{R_1}{R_2} \quad (3.2)$$

Para o circuito da Figura 16, o ganho de tensão é dado pela relação entre os resistores R_3 e R_1 e o potenciômetro R_2 . Aproveitando o resultado da equação (3.2), define-se a faixa de variação do ganho do estágio de pré-amplificação (3.3).

$$G_v = 1 + \frac{R_3}{R_1 + R_2} \approx \begin{cases} 1,11 & (0,91dB), & R_2 = 500k\Omega \\ 12,91 & (22,21dB), & R_2 = 0 \end{cases} \quad (3.3)$$

Ou seja, considerando um sinal de entrada de $200mV$ pico a pico, a etapa de pré-amplificação pode elevar o sinal de entrada até em torno de $2,58$ Volts com o ajuste do valor do potenciômetro.

3.1.2.3 Resposta em Frequência

O objetivo do Pré-amplificador é ajustar a amplitude do sinal de entrada sem modificar drasticamente o conteúdo harmônico. Entretanto, se faz necessária uma filtragem do sinal, seja para limitá-lo em banda, seja para remover o ruído de baixa frequência (zumbido). O capacitor C_4 em conjunto com R_6 , R_8 e a impedância de entrada do amplificador operacional compõe um filtro passa-alta, que remove o nível CC do sinal da guitarra, (3.4).

$$f_{Cinput} = \frac{1}{2\pi(R_8/(R_6 + Z_{InAmpOp}))C_4} \approx 1,91Hz \quad (3.4)$$

Após a filtragem inicial para remover o nível CC do sinal de entrada, o pré-amplificador atua como um filtro passa-faixa, porém não produz efeito significativo na banda de frequência do sinal da guitarra, visto que esse não é o seu objetivo. O corte superior desse filtro passa-faixa é definido por C_2 e R_3 , na equação (3.5).

$$f_{Cmax} = \frac{1}{2\pi R_3 C_2} \approx 60,47kHz \quad (3.5)$$

Já C_1 , R_1 e R_2 , determinam o corte inferior, (3.6).

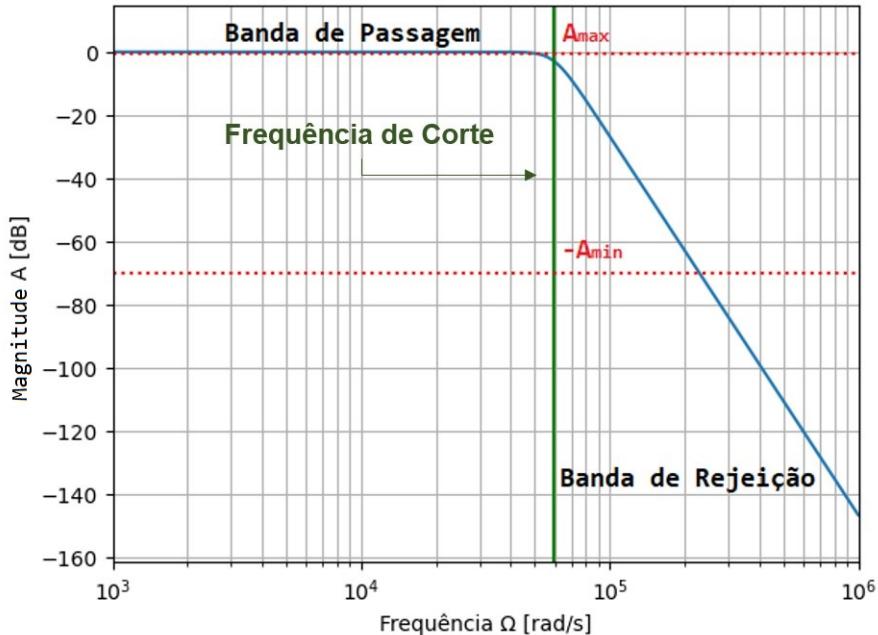
$$f_{Cmin} = \frac{1}{2\pi(R_1 + R_2)C_1} \approx \begin{cases} 0,07Hz, & R_2 = 500k\Omega \\ 7,20Hz, & R_2 = 0 \end{cases} \quad (3.6)$$

3.1.3 Filtro Anti-aliasing

Com o intuito de limitar a banda de passagem do sinal de entrada, respeitando o teorema de Nyquist-Shannon durante o processo de amostragem do sinal com o conversor A/D, o áudio será processado por um filtro passa-baixa Butterworth de 6ª ordem na topologia Sallen and Key. O filtro foi projetado considerando uma atenuação máxima na banda de passagem de $A_{max} = 0,5dB$ para uma frequência de 0 a $8kHz$, e uma atenuação mínima na banda de rejeição de $A_{min} = 70dB$, em torno de $44,1kHz$.

O filtro passa-baixa Butterworth foi escolhido pelo fato de possuir uma resposta plana, não contendo nenhum tipo de ondulação na banda de passagem, além disso, a sua resposta máxima plana acontece próxima à frequência $\Omega = 0$ (JUNIOR, 2007). Na Figura 18, é apresentada uma resposta típica de um filtro passa-baixa Butterworth.

Figura 18 – Resposta em Frequência de um Filtro Passa-Baixa Butterworth.



Fonte: Autor.

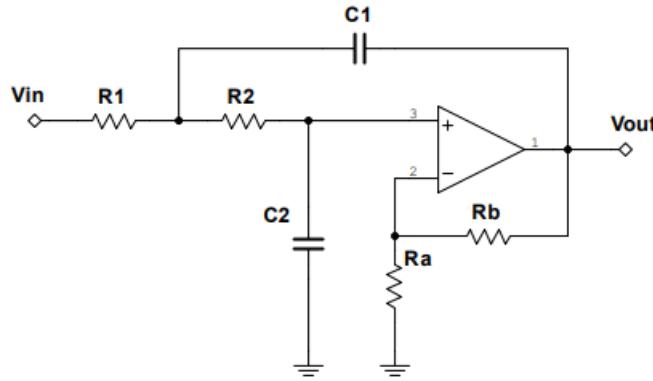
O projeto do filtro foi realizado utilizando as funções *Buttord* e *Butter* da biblioteca *Scipy* do Python. A função *Buttord*, a partir das definições de limite superior da banda de passagem (Ω_p), limite inferior da banda de rejeição (Ω_s), máxima atenuação da banda

de passagem (A_{max}) e mínima atenuação da banda de rejeição (A_{min}) retorna a ordem mínima e a frequência de corte (Ω_n), onde o sinal de entrada cai a 3dB. Já a função *Butter* é utilizada para encontrar os coeficientes do numerador e denominador da função de transferência do filtro, ela recebe como parâmetros de entrada a ordem mínima e a frequência natural. A função de transferência de sexta ordem do filtro passa-baixa obtida com o uso das funções do Python é apresentada na equação (3.7).

$$H(s) = \frac{4,62 \cdot 10^{28}}{s^6 + 2,31e \cdot 10^5 s^5 + 2,68 \cdot 10^{10} s^4 + 1,96e \cdot 10^{15} s^3 + 9,61 \cdot 10^{19} s^2 + 2,98 \cdot 10^{24} s + 4,62 \cdot 10^{28}} \quad (3.7)$$

Para realizar esse filtro fisicamente foi escolhida a topologia Sallen and Key por ser relativamente simples. Na Figura 19 é apresentado circuito capaz de representar um filtro passa-baixa de segunda ordem (também conhecido com biquad).

Figura 19 – Circuito Filtro Passa-Baixa Sallen And Key.



Fonte: Autor.

A função de transferência do circuito da Figura 19 é definida na equação (3.8), onde $k = 1 + \frac{R_b}{R_a}$.

$$T(s) = \frac{k / (R_1 R_2 C_1 C_2)}{s^2 + (\frac{1}{R_1 C_1} + \frac{1}{R_2 C_1} + \frac{1-k}{R_2 C_2})s + \frac{1}{R_1 R_2 C_1 C_2}} \quad (3.8)$$

Para auxiliar no cálculo dos componentes do circuito de interesse, vale relembrar a notação comum para a função de transferência de um filtro passa-baixa de segunda ordem, definida na equação (3.9).

$$T_{LP}(s) = \frac{K_0 \Omega_0}{s^2 + \frac{\Omega_0^2}{Q} s + \Omega_0^2} \quad (3.9)$$

Comparando as equações (3.8) e (3.9), encontramos os valores de Ω_0 , Q e K_0 , que se referem respectivamente a frequência de ressonância, fator de qualidade e ganho na banda de passagem.

$$\Omega_0 = \sqrt{\frac{1}{R_1 R_2 C_1 C_2}} \quad (3.10)$$

$$Q = \frac{\Omega_0}{\left(\frac{1}{R_1 C_1} + \frac{1}{R_2 C_1} + \frac{1-k}{R_2 C_2}\right)} \quad (3.11)$$

$$K_0 = 1 + \frac{R_b}{R_a} \quad (3.12)$$

Como a equação do filtro passa-baixa obtido em (3.7) é de sexta ordem, e a topologia sallen and key discutida aqui é capaz de implementar apenas funções de transferências de segunda ordem, logo, para dar continuidade ao cálculo dos componentes, é necessário dividir a função de transferência da equação (3.7) em três biquads.

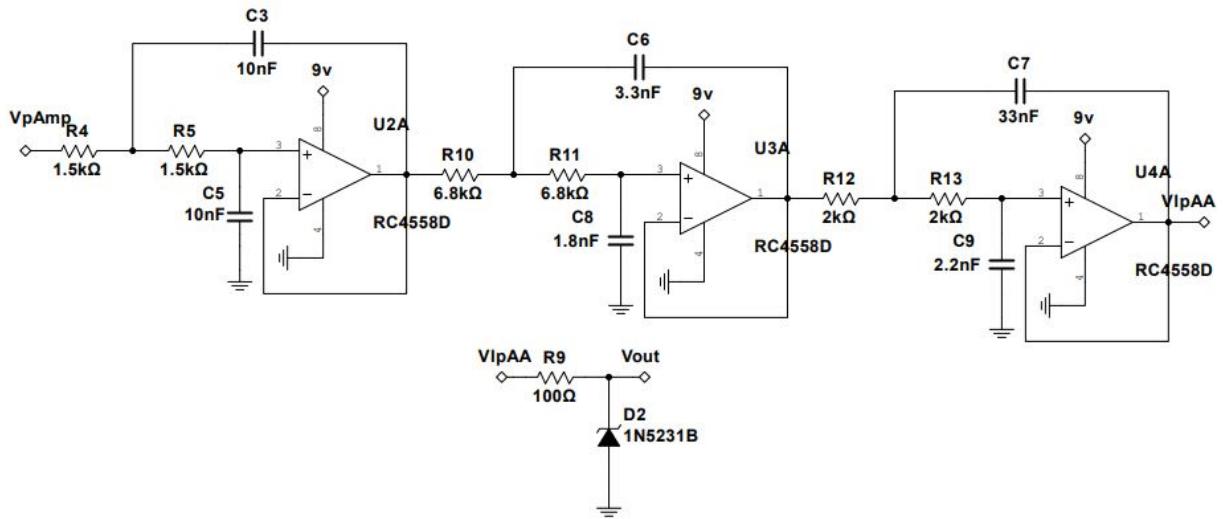
$$H_{b1}(s) = \frac{3,59 \cdot 10^9}{s^2 + 1,16 \cdot 10^5 s + 3,59 \cdot 10^9} \quad (3.13)$$

$$H_{b2}(s) = \frac{3,59 \cdot 10^9}{s^2 + 8,47 \cdot 10^4 s + 3,59 \cdot 10^9} \quad (3.14)$$

$$H_{b3}(s) = \frac{3,59 \cdot 10^9}{s^2 + 3,10 \cdot 10^4 s + 3,59 \cdot 10^9} \quad (3.15)$$

De posse dos biquads, considerando $k = 1$ e utilizando as equações (3.9) a (3.11), é possível chegar ao circuito da Figura 20, onde todos os componentes já foram ajustados para os seus valores comerciais mais próximos.

Figura 20 – Circuito Filtro Anti-Aliasing.

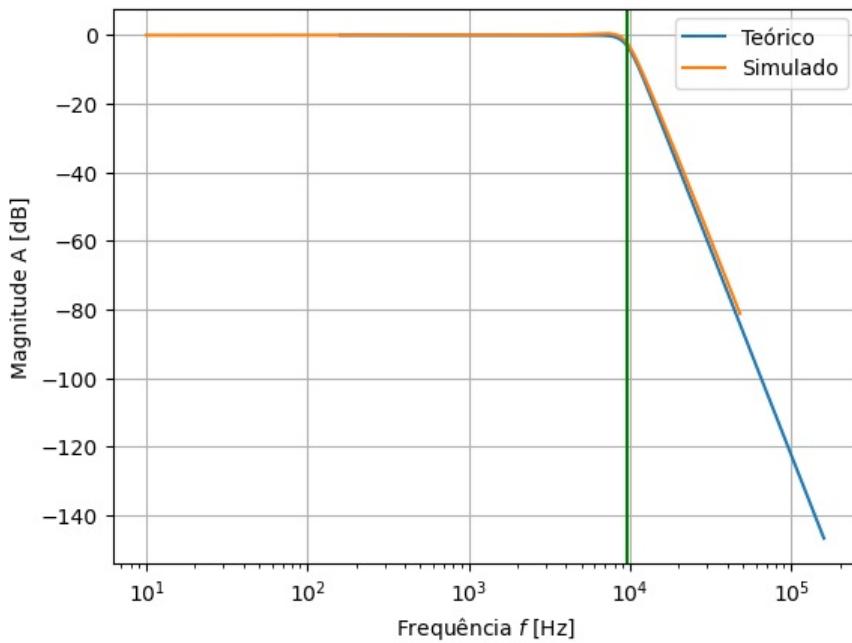


Fonte: Autor.

A Figura 21 retrata a comparação entre as respostas em frequência obtidas a partir da equação (3.7) e do circuito da Figura 20 considerando apenas os três biquads. Para essa

comparação o circuito foi implementado e simulado no software LTspice. Fica evidente que as aproximações necessárias para ajustar os componentes para os valores comerciais mais próximos causam uma pequena diferença entre as curvas teórica e simulada, porém nada significativo.

Figura 21 – Resposta em Frequência: Teórico x Simulação.



Fonte: Autor.

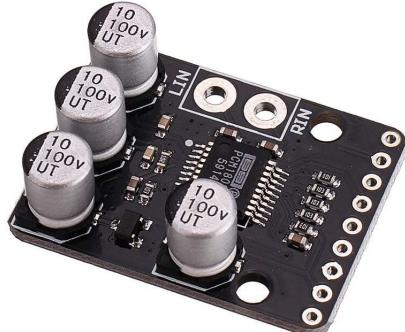
Nas seções anteriores foi possível verificar que circuito de Pré-Amplificação pode elevar o sinal de entrada em até 12,91 vezes, pensado nisso, como forma de proteger o conversor [A/D](#), na saída do filtro anti-aliasing, Figura 20, foi adicionado um circuito limitador de tensão, composto por um diodo zener D_2 e um resistor R_9 . Os limitadores são amplamente utilizados para proteger circuitos sensíveis contra picos de tensão indesejados, ou seja, à medida que o nível de entrada excede um limite, a saída deve permanecer constante ([RAZAVI, 2010](#)). Para proteger o conversor [A/D](#), o limite máximo do sinal na saída desta estapa é definido em 5,1 Volts, valor estabelecido na documentação do PCM1802.

3.1.4 Conversor Analógico para Digital - PCM1802

Devido a limitação da resolução presente no conversor [A/D](#) da placa de desenvolvimento STM32F407DISCOVERY, que é de 12 bits, optou-se por adicionar um módulo capaz de suprir essa necessidade, realizando a leitura do sinal proveniente da guitarra e o transmitindo ao [DSP](#). O módulo escolhido foi o conversor [A/D](#) Delta-Sigma PCM1802, que converte um sinal de áudio analógico em um sinal de áudio digital com alta precisão e baixo ruído. O PCM1802 possui uma interface serial de áudio digital, [I2S](#) que permite a

transferência de dados digitais em alta velocidade. Ele suporta taxas de amostragem de $32kHz$ a $96kHz$ com resolução de até 24 bits ([INSTRUMENTS, 2016](#)).

Figura 22 – Módulo Conversor Analógico para Digital PCM1802.



Fonte: ([AMAZON, 2023](#)).

Por padrão industrial, o Compact Disc ([CD](#)), uma aplicação de alta fidelidade, requer uma largura de faixa do sinal de áudio de até $20kHz$, sendo necessária uma taxa de amostragem de $40kHz$, entretanto, usualmente é definido o padrão de $44,1kHz$. O sinal é então amostrado a uma resolução de 16 bits, para reduzir o erro de quantização ([LATHI, 2007](#)). Pensando nisso, optou-se por utilizar uma frequência de amostragem de $96kHz$, além da resolução de 24 bits para o dado amostrado oferecida pelo PCM1802, especificações mais do que suficientes para um processamento de áudio em alta qualidade.

Na tabela 1, é apresentada a configuração utilizada para realizar a comunicação entre o PCM1802 e o STM32.

Tab. 1 – Tabela de Conexões PCM1802.

PCM1802	STM32
PDWN	3,3V
BYPAS	GND
FSYNC	3,3V
MODE0	GND
MODE1	GND
FMT0	3,3V
FMT1	GND
LRCK	PB12
BCK	PB10
SCK	PC6
DOUT	PC2

Fonte: Autor.

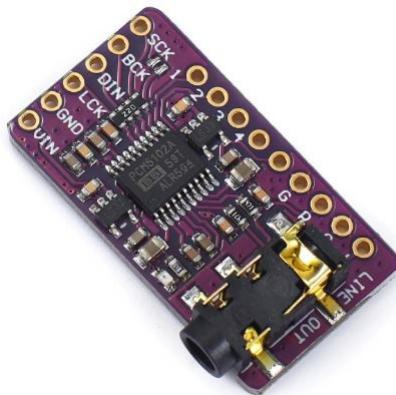
Os terminais MODE0 e MODE1 são responsáveis por definir que o PCM1802 irá atuar como subordinado ao STM32, já FMT0 e FMT1 especificam o formato em que os dados de áudio serão convertidos serão transmitidos ao STM32, ou seja, [I2S](#) de 24 bits. Por fim, ainda em relação a configuração de hardware da comunicação, os terminais PDWN, FSYNC e BYPAS habilitam respectivamente o conversor [A/D](#), a conversão do sinal de

áudio e o filtro passa-alta interno, que evita a conversão do nível **CC** contido no sinal de entrada. Toda a transmissão do áudio é realizada através de **BCK**, **LRCLK**, **SCK** e **DOUT**.

3.1.5 Conversor Digital para Analógico - PCM5102

O PCM5102 é um conversor **D/A** fabricado pela Texas Instruments, que foi projetado para converter sinais digitais de áudio em sinais analógicos de alta fidelidade, visto que, pode trabalhar com dados em resoluções de 16 a 32 bits e taxas de amostragem de até $384kHz$.

Figura 23 – Módulo Conversor Digital para Analógico PCM5102.



Fonte: ([ALIEXPRESS, 2023](#)).

A configuração do PCM5102 é mais simples que a do PCM1802. A única configuração necessária é a definição do formato desejado, realizada pelo terminal FMT, bastando apenas deixá-lo em nível baixo, selecionando o padrão **I2S**. Na Tabela 2 temos as conexões entre os terminais do conversor **D/A** e o **DSP**.

Tab. 2 – Tabela de Conexões PCM5102.

PCM5102	STM32
LRCK	PB12
BCK	PB10
SCKL	PC6
DIN	PC3

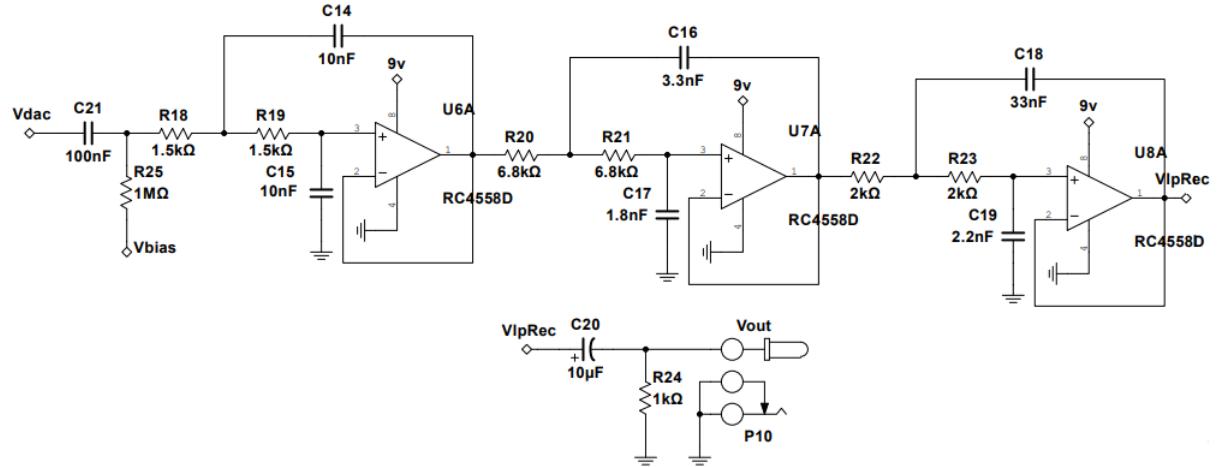
Fonte: Autor.

3.1.6 Filtro de Reconstrução

O processo de reconstrução de um sinal em tempo contínuo $x(t)$ a partir de suas amostras também é chamado de interpolação e essa reconstrução é feita passando o sinal amostrado através de um filtro passa-baixa ([LATHI, 2007](#)). Sendo assim, foi reaproveitado

o filtro Anti-Aliasing da seção anterior, para realizar o último processamento do sinal de áudio antes de ser transmitido ao amplificador.

Figura 24 – Circuito Filtro de Reconstrução.



Fonte: Autor.

Observando atentamente o circuito da Figura 24, pode-se notar que existem apenas duas diferenças em comparação com o filtro da Figura 20. Ao sinal proveniente do PCM5102 é adicionado um nível CC a partir do resistor R_{25} , polarizando a entrada não inversora do amplificador operacional e possibilitando o processamento do sinal pelo circuito. Já o capacitor C_{21} está ali para evitar que a tensão de polarização seja aplicada a saída do conversor D/A, o chamado capacitor de acoplamento. Na saída desse filtro, é adicionado também um filtro passa-alta de primeira ordem, responsável por remover a tensão de polarização e transmitir apenas o sinal de áudio para o amplificador. Na equação (3.16) é definida a frequência de corte deste filtro.

$$f_C = \frac{1}{2\pi R_{24} C_{20}} \approx 15,91 \text{ Hz} \quad (3.16)$$

3.1.6.1 Impedância de Saída

A impedância de saída do pedal é definida pela associação paralela entre o resistor R_{24} e a impedância de saída do amplificador operacional, representado na equação (3.17).

$$\begin{aligned} Z_{Out} &= (R_{24}/Z_{OutAmpOp}) \\ Z_{Out} &\approx 1 \text{ k}\Omega \end{aligned} \quad (3.17)$$

Idealmente a impedância de saída do pedal deve ser baixa, a resistência R_{24} faz com que a impedância de saída do pedal fique em torno de $1 \text{ k}\Omega$, que é uma boa impedância de saída, mantendo a fidelidade do sinal.

3.1.7 Microcontrolador - STM32F407DISCOVERY

A placa de desenvolvimento STM32F407DISCOVERY é o núcleo da pedaleira de Guitarra, sendo responsável por controlar e coordenar as operações de cada elemento do circuito. Este microcontrolador funciona a uma velocidade de clock máxima de $168MHz$, permitindo muitas instruções por amostra de áudio. A placa contém um microcontrolador STM32F407VGT6 com um processador ARM Cortex-M4 de 32 bits, uma Floating Point Unit (**FPU**), que permite cálculos rápidos e complexos de algoritmos para **DSP**, além de uma memória Flash de $1Mbyte$ e $192kbytes$ de RAM ([ST, 2020](#)), especificações mais do que suficientes para a implementação dos efeitos escolhidos para este projeto.

Outro benefício que justifica a escolha deste microcontrolador diz respeito ao conjunto de ferramentas de desenvolvimento oferecidos pela STMicroelectronics, que auxiliam na programação, como a STM32CubeIDE, um ambiente de desenvolvimento projetado especificamente para programar, depurar e desenvolver projetos de firmware. O STM32CubeIDE possui a biblioteca Hardware Abstraction Layer (**HAL**), que proporciona a abstração do hardware, isentando o desenvolvedor das peculiaridades do hardware e tornando mais rápida a confecção dos códigos, ou seja, acelera a curva de aprendizado com o microcontrolador e possibilita que o foco do estudo seja totalmente direcionado a confecção dos efeitos e aplicação da teoria de processamento digital de sinais.

Figura 25 – Placa de Desenvolvimento STM32F4DISCOVERY.



Fonte: ([ST, 2020](#)).

A placa STM32F407DISCOVERY pode ser usada para uma ampla variedade de projetos, desde controle de dispositivos simples até aplicações de alto desempenho, como processamento de sinais que é o caso deste projeto.

3.1.8 Módulo LCD

Como forma de facilitar a interação do usuário com a pedaleira, foi adicionado um Liquid Crystal Display (**LCD**), Figura 26, para apresentar a interface gráfica com o nome dos efeitos e parâmetros a serem ajustados. A conexão entre o display e o microcontrolador é realizada utilizando o protocolo de comunicação serial **I2C**, o que reduz o número de terminais em uso e simplifica o circuito.

Figura 26 – Módulo LCD com Comunicação I2C.



Fonte: ([ROBOTICA, 2023](#)).

A tabela 3 faz referência a configuração da conexão entre o módulo **LCD** e o microcontrolador.

Tab. 3 – Tabela de Conexões LCD.

LCD	STM32
SDA	PB7
SCL	PB6

Fonte: Autor.

3.2 Software

Com o Hardware necessário desenvolvido e em pleno funcionamento, o próximo passo é a aquisição do sinal da guitarra e seu processamento digital. Nesta seção está descrito o funcionamento da rotina principal do projeto, além das funções destinadas ao processamento do áudio e aplicação dos efeitos sonoros.

3.2.1 Fluxo de Código

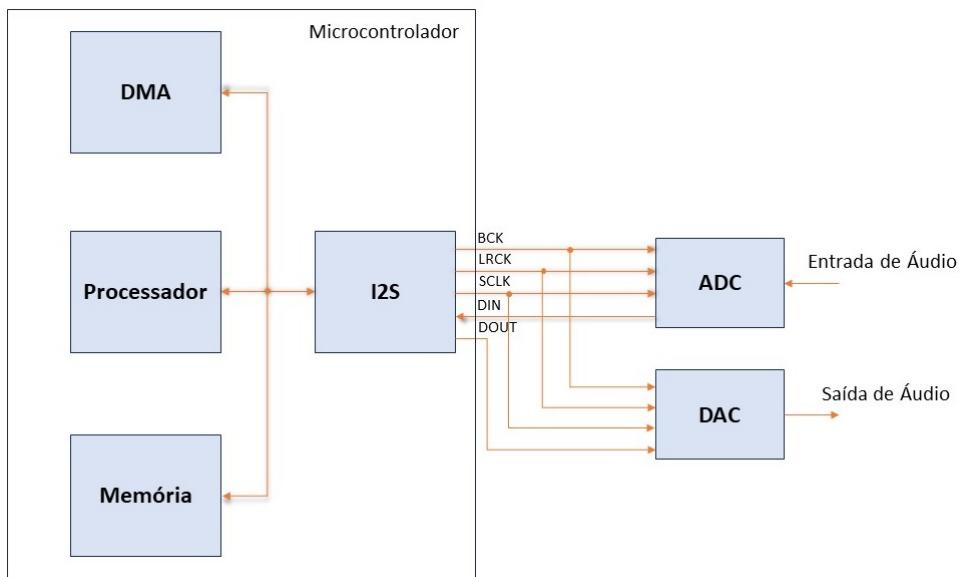
O desenvolvimento do software envolve a programação de todos os periféricos e dos efeitos digitais de áudio. O microcontrolador recolhe continuamente amostras da

entrada de áudio a uma frequência de $96kHz$ e resolução de 24 bits utilizando o Conversor A/D PCM1802, os dados convertidos e depois processados pelo DSP são enviados para o conversor D/A PCM5102 que reconstitui a informação analógica. Ao receber os dados, o programa aplica quaisquer efeitos selecionados pelo usuário, podendo aplicar um ou mais efeitos simultaneamente. Toda a etapa de aquisição e processamento de áudio é tratado por interrupções, que nada mais são do que uma forma de sinalizar ao processador a ocorrência de eventos importantes relacionados aos periféricos, ou seja, na ocorrência da interrupção é solicitado ao processador que pare o que está fazendo para atendê-la. As interrupções são uma maneira eficiente de lidar com eventos periódicos e otimizar o desempenho do sistema. Já a interface com o usuário, por trabalhar com eventos esporádicos e sem prioridade, são implementados na rotina principal do programa, o que ajuda também na separação desta etapa de código da parte destinada ao processamento de áudio.

3.2.2 Ping-Pong Buffer

Na Figura 27 temos a representação das conexões internas e externas entre os conversores e os elementos do microcontrolador. Nesta configuração o Controlador DMA trabalha transferindo os dados das amostras entre os periféricos e a memória principal RAM, ou seja, todo o processamento é baseado na manipulação individual amostra por amostra proveniente do conversor A/D. A interação entre o DMA, processador, memória e os periféricos é definido pela técnica de Ping-Pong Buffer aqui descrita.

Figura 27 – Diagrama de Comunicação da Pedaleira.

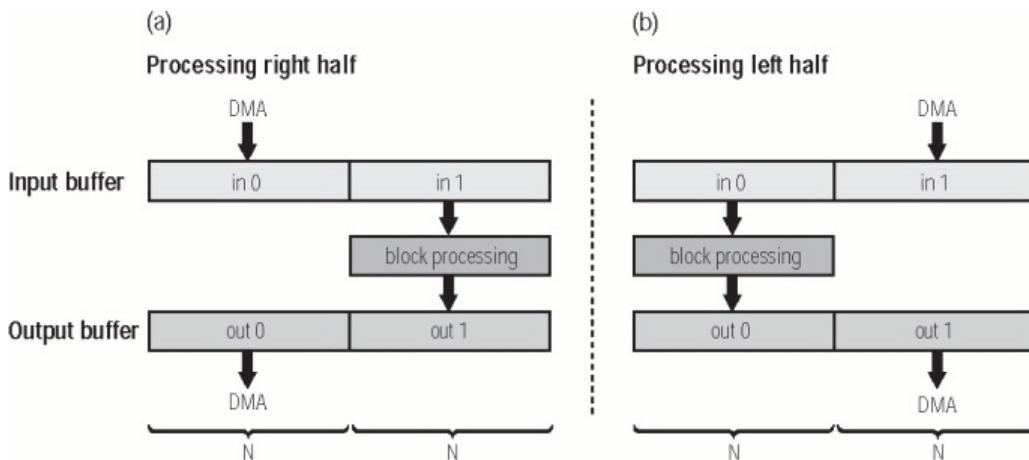


Fonte: Autor.

A técnica de Ping-Pong Buffer realiza o processamento de áudio de maneira eficiente, onde basicamente, existe um espaço de memória reservado (também chamado de vetor ou buffer), que é dividido em duas partes iguais, sendo acessadas pelo DMA e pelo processador

independente. Dessa forma, enquanto uma parcela do vetor recebe uma cadeia de bytes até que seu espaço de memória seja totalmente preenchido, bytes provenientes de amostras vindas do conversor A/D, a outra parcela do vetor é acessada pelo processador. Por se tratar de um sistema que atua lendo e transmitindo dados, é necessário a utilização de dois vetores com o dobro do tamanho necessário para armazenar uma amostra lida. Isso é ilustrado na Figura 28, enquanto o processador faz a leitura da parcela *in 1* do vetor de entrada e armazena o resultado na parcela *out 1* do vetor de saída, o controlador DMA está preenchendo *in 0* e transmitindo os dados de *out 0* (KATZ; GENTILE; LUKASIAK, 2007). Esse fluxo altamente eficiente aproveita os periféricos do STM32 para fazer todo o processo de entrada/saída de áudio, reduzindo o esforço que seria necessário do processador e permitindo que ele dedique todos os seus recursos a processar os algoritmos programados.

Figura 28 – Fluxo de Dados do Ping-Pong Buffer.

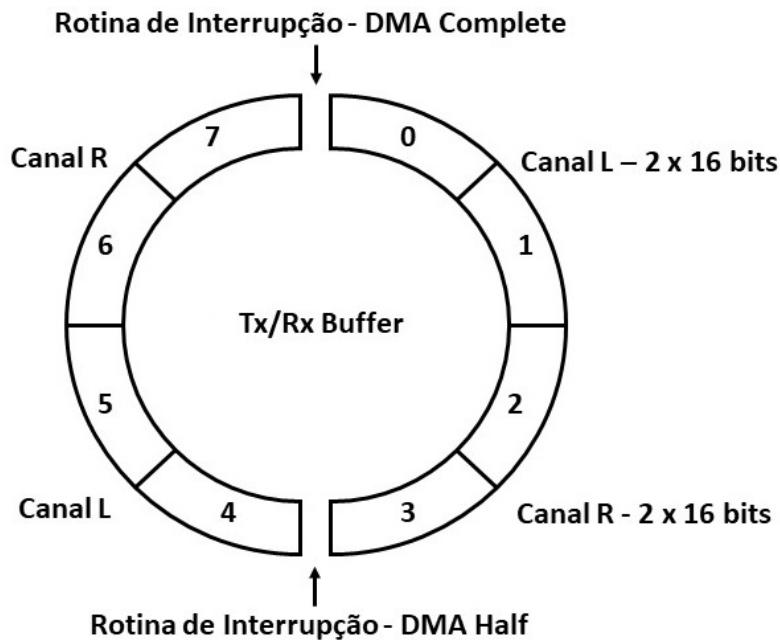


Fonte: (KATZ; GENTILE; LUKASIAK, 2007).

Outro ponto importante da técnica de Ping-Pong Buffer, é utilização dos buffers de maneira circular, ou seja, os dados inseridos no buffer são organizados na ordem que chegam, e sua retirada é feita nessa mesma ordem. Caso o vetor de dados seja totalmente preenchido, o DMA passa a inserir novos dados a partir do começo do vetor, sobrepondo as leituras mais antigas (KATZ; GENTILE; LUKASIAK, 2007). Neste projeto, os conversores A/D, D/A atuam transmitindo e recebendo amostras com resolução de 24 bits. As amostras de 24 bits são enviadas em quadros de 32 bits especificados pelo protocolo I2S, cada quadro contém um par de amostras dos canais esquerdo e direito, 16 bits para cada canal, pois a função do STM32 responsável por receber e transmitir as informações dos conversores, trabalha apenas com variáveis de 16 bits. Após o recebimento dos pares de amostras de 16 bits, é feita a reconstituição da amostra de 24 bits. Para aplicar o Ping-Ping Buffer são necessários dois vetores (Recepção e Transmissão) de 16 bits com 8 posições. As quatro primeiras posições desse vetor são preenchidas pelo DMA e disponibilizada em sua interrupção "DMA Half Complete" quando recebe 2 pares de amostras do periférico I2S. Enquanto essas duas amostras estão sendo manipuladas pelo

processador e em seguida enviadas para o conversor D/A através do periférico I₂S, uma segunda rotina de interrupção é gerada, denominada de "*DMA Complete*" que disponibiliza na outra metade do vetor mais 2 pares de amostras para serem processadas, e esse ciclo se repete continuamente para proporcionar o processamento digital de áudio em tempo real (CONCEIÇÃO, 2022). A Figura 29 ilustra o preenchimento do buffer de dados. No Apêndice B está implementado todo o código utilizado para replicar a técnica descrita nesta seção.

Figura 29 – Buffer Circular e Interrupção DMA.



Fonte: Autor.

3.3 Efeitos de Áudio

Nos tópicos seguintes serão descritos os efeitos implementados digitalmente na pedaleira aqui desenvolvida, descrevendo suas equações e funcionamento.

3.3.1 Distorções

O efeito de distorção do som de áudio surge da inserção de harmônicos ao sinal original. A quantidade e a proporção de harmônicos irão mudar a depender da topologia de implementação, ou seja, depende de como a saturação foi gerada, se obtida com amplificadores operacionais, transistores, diodos ou válvulas. Dentro da família de efeitos de distorção, podemos citar o Overdrive e o Distortion. O Overdrive tenta simular o efeito de amplificadores valvulados saturados, a sua região de operação está tanto na região linear quanto na região não linear, devido a seu recorte suave no sinal de entrada, o chamado "*Soft Clipping*". Já o Distortion opera principalmente na região não linear e produz uma

saída com maior amplitude, sendo gerado a partir de recorte simétrico e abrupto do sinal de áudio, denominado "Hard Clipping"(GONTIJO; PAVEI; NOCETI, 2016).

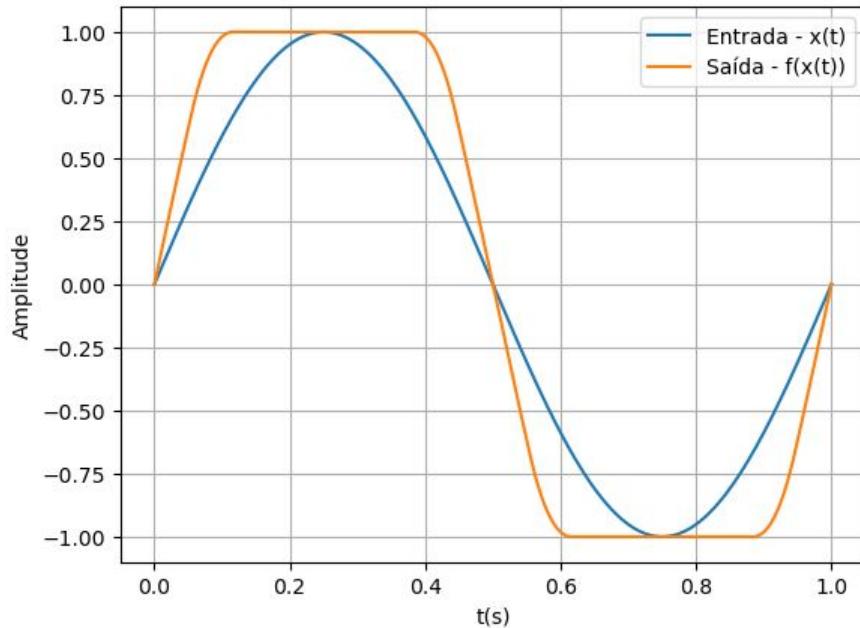
3.3.1.1 Overdrive

A implementação do efeito de Overdrive pode ser realizada a partir de um recorte suave dos valores de entrada. Uma abordagem possível para a geração de uma não linearidade de saturação suave é dada por (ZOLZER, 2011):

$$f(x) = \begin{cases} 2x & , \quad 0 \leq |x| \leq 1/3 \\ 1 - \frac{(2-3x)^2}{3}, & 1/3 \leq |x| \leq 2/3 \\ 1, & 2/3 \leq |x| \leq 1 \end{cases} \quad (3.18)$$

Na Figura 30 podemos observar a curva características da equação do efeito de Overdrive, enunciada na equação (3.18). Pode-se notar a suave transição entre as regiões de linearidade e não linearidade, o que identifica este tipo de efeito.

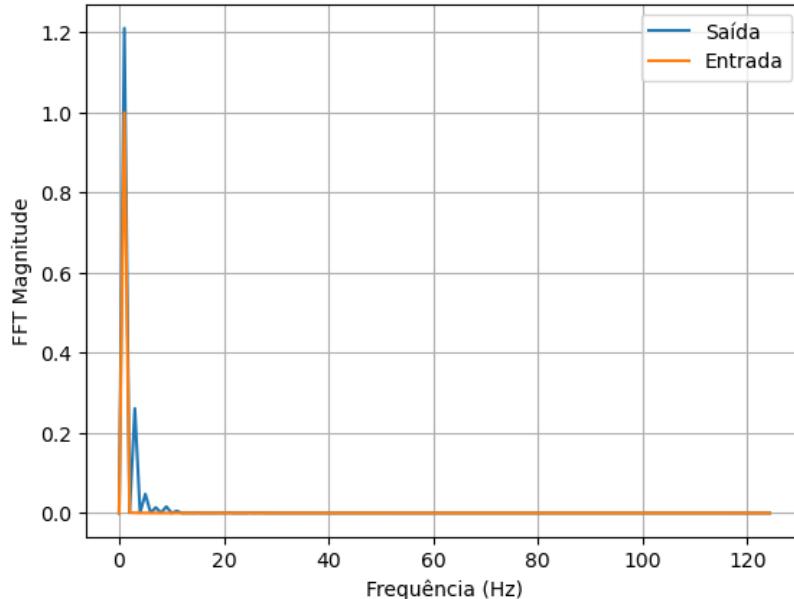
Figura 30 – Curva Característica do Overdrive.



Fonte: Autor.

A Figura 31 representa o espectro de frequência dos sinais. Na entrada temos uma senoide com uma frequência de 1Hz e após a aplicação do efeito de Overdrive passamos a observar o surgimento de harmônicos na saída.

Figura 31 – Espectro do Overdrive.



Fonte: Autor.

3.3.1.2 Distortion

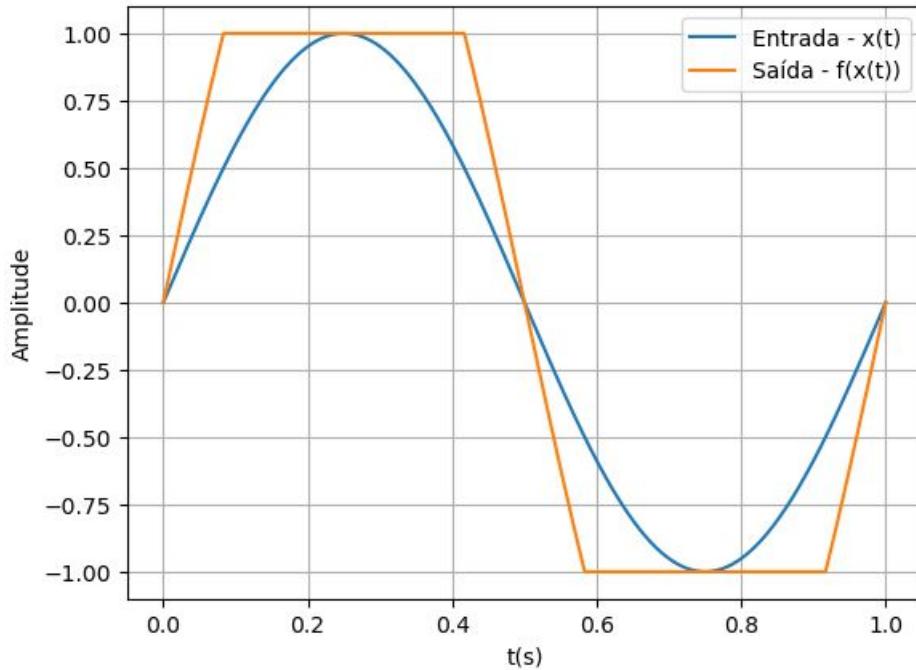
Para a implementação do efeito de Distortion, precisamos de uma equação que forneça uma não linearidade bem pronunciada. A equação (3.19) possibilita o surgimento de harmônicos com muito mais potência na saída que o efeito de Overdrive.

$$f(x) = \begin{cases} -1 & , \quad x \leq -1 \\ x, & -1 < x < 1 \\ 1, & x \geq 1 \end{cases} \quad (3.19)$$

Na Figura 32 podemos observar a curva característica da equação do efeito de Distortion. Por conta do recorte não suave do sinal de entrada, temos um som mais repleto de harmônicos do que no caso anterior, o que é popularmente descrito como um som mais "quente".

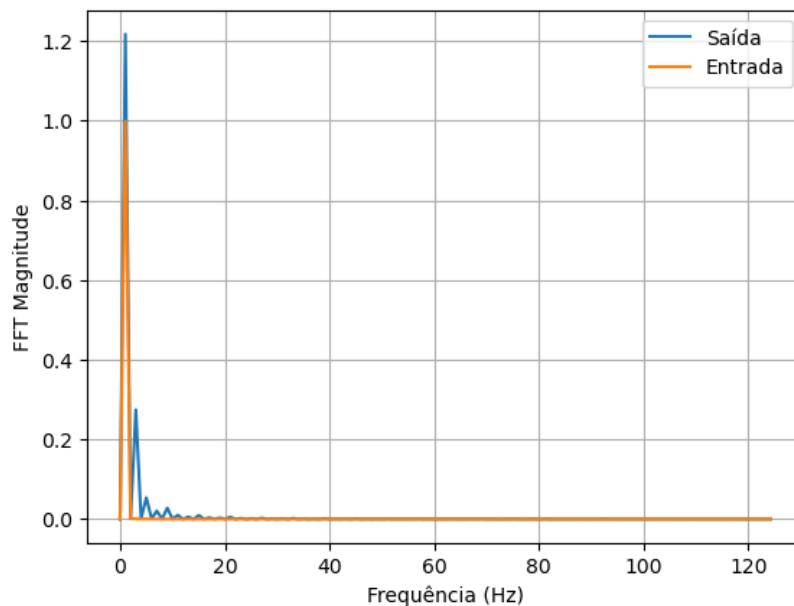
Ainda considerando uma senoide com frequência de 1Hz e após a aplicação do efeito de Distortion, na Figura 33, passamos a observar o surgimento de harmônicos na saída, porém mais pronunciados que no caso do Overdrive.

Figura 32 – Curva Característica do Distortion.



Fonte: Autor.

Figura 33 – Espectro do Distortion.



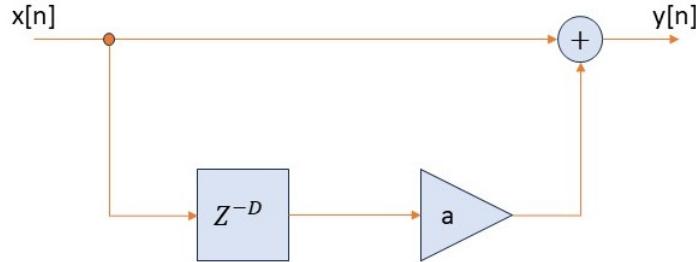
Fonte: Autor.

3.3.2 Delay

O Delay é considerado como o elemento básico na implementação de outros efeitos de áudio como Chorus, Reverb, Flanger e etc. Um efeito de delay consiste em atrasar o som original em um determinado intervalo de tempo, aplicar um ganho específico no sinal atrasado e então adicioná-lo ao som original. O tipo de Delay implementado depende de qual sinal será atrasado, caso o sinal de entrada seja atrasado, teremos um Delay do tipo

Finite Impulse Response (**FIR**), se atrasarmos o sinal de saída teremos então um Delay do tipo Infinite Impulse Response (**IIR**) ([GONTIJO; PAVEI; NOCETI, 2016](#)). Neste trabalho iremos focar no Delay **FIR**, representado pelo diagrama de blocos da Figura 34.

Figura 34 – Diagrama de blocos do Delay FIR.



Fonte: Autor.

A partir do diagrama de blocos podemos retirar a equação que possibilita a implementação deste tipo de Delay. Este efeito apresenta dois parâmetros a serem ajustados pelo usuário, o ganho do sinal atrasado a e a quantidade de atraso D .

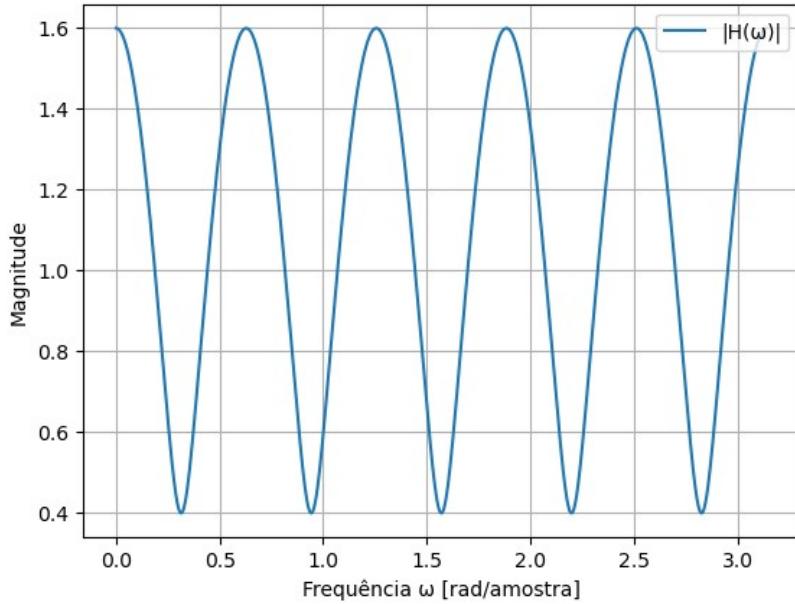
$$y[n] = x[n] + a \cdot x[n - D] \quad (3.20)$$

Da equação (3.20) podemos determinar a resposta em frequência do efeito de Delay, (3.21), onde ω representa a frequência normalizada em radianos por amostra.

$$H(e^{j\omega}) = 1 + a \cdot e^{-j\omega D} \quad (3.21)$$

O Delay **FIR** apresenta a resposta em frequência vista na Figura 35, considerando um ganho a positivo. Podemos notar um comportamento interessante, o filtro atua amplificando todas as componentes de frequências que são múltiplas de $2\pi/D$, e atenua todas as frequências intermediárias, um comportamento típico do chamado "Comb Filter" ([ZOLZER, 2011](#)).

Figura 35 – Resposta em Frequência do Delay FIR.

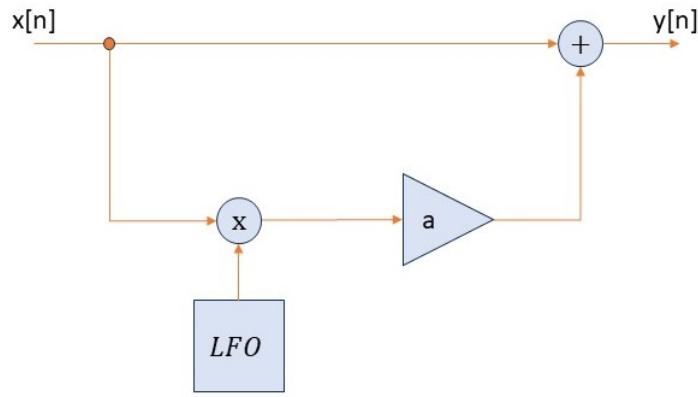


Fonte: Autor.

3.3.3 Tremolo

O efeito de Tremolo é realizado a partir da multiplicação do sinal de áudio com uma senoide de baixa frequência, gerada por um Low Frequency Oscillation (LFO). Essa operação equivale a modular em amplitude o sinal proveniente da guitarra (GONTIJO; PAVEI; NOCETI, 2016). Na Figura 36 temos o diagrama de blocos que representa o efeito de Tremolo.

Figura 36 – Diagrama de blocos do Tremolo.



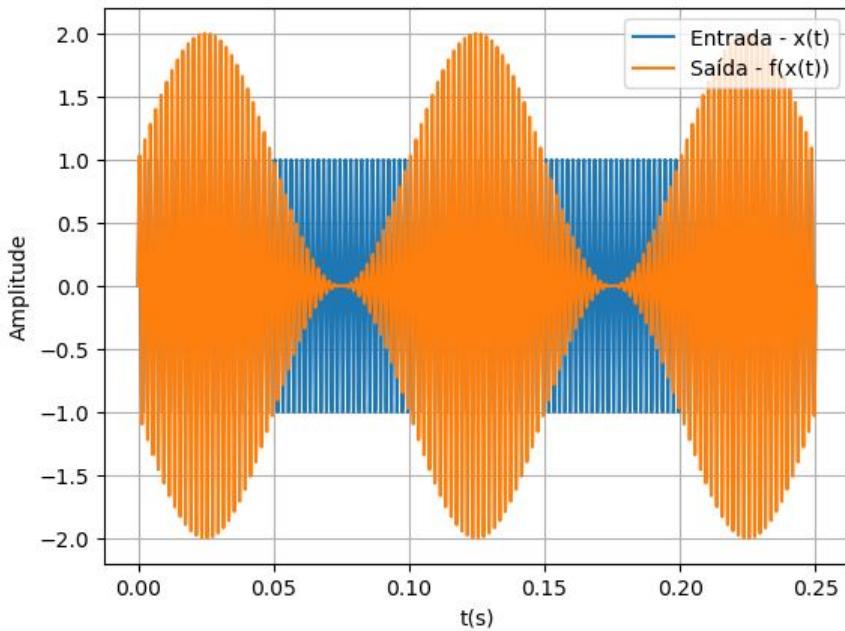
Fonte: Autor.

Analisando o diagrama de blocos da Figura 36 podemos chegar a equação a diferenças que relaciona o sinal de entrada e saída, (3.22).

$$y[n] = x[n] \cdot \left(1 + a \cdot \operatorname{sen}\left(2\pi \frac{f_c}{F_s} n\right)\right) \quad (3.22)$$

Em sua grande maioria, os efeitos de tremolo apresentam dois possíveis controles, a velocidade da oscilação, responsável por definir a frequência das variações de volume, sendo ajustado pelo parâmetro f_c . O segundo controle é o de profundidade, que é usado para definir o quanto profunda é a mudança de volume, controlado através do ganho a . Na Figura 37 é possível observar o comportamento do efeito de Tremolo. Para este exemplo é realizada a modulação de uma senoide de $500Hz$, amostrada a $44.1kHz$, com um LFO ajustado em $10Hz$ e um ganho de $a = 1$.

Figura 37 – Resposta Típica do Efeito de Tremolo.



Fonte: Autor.

3.3.4 Flanger

O último efeito a ser implementado é o Flanger, que é a junção das características dos efeitos de Delay e Tremolo. O Flanger é produzido a partir da criação de interferências construtivas e destrutivas (ZOLZER, 2011). As interferências são geradas somando o som original com uma versão sua atrasada, porém diferente do efeito de Delay, o atraso é variável e definido por um LFO. O diagrama de blocos do Efeito de Flanger é apresentado na Figura 38.

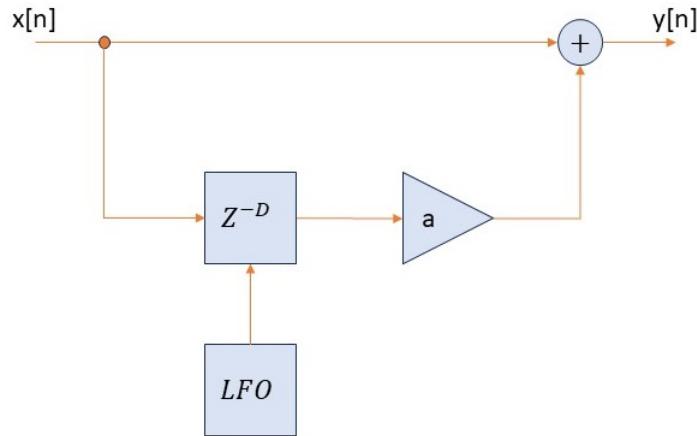
Na equação (3.23) temos descrito matematicamente o efeito de Flanger.

$$y[n] = x[n] + a \cdot x[n - d[n]] \quad (3.23)$$

Sendo que o valor de $d[n]$ é definido pela equação (3.24).

$$d[n] = \frac{D}{2} \left(\operatorname{sen}(2\pi \frac{f_c}{F_s} n) \right) \quad (3.24)$$

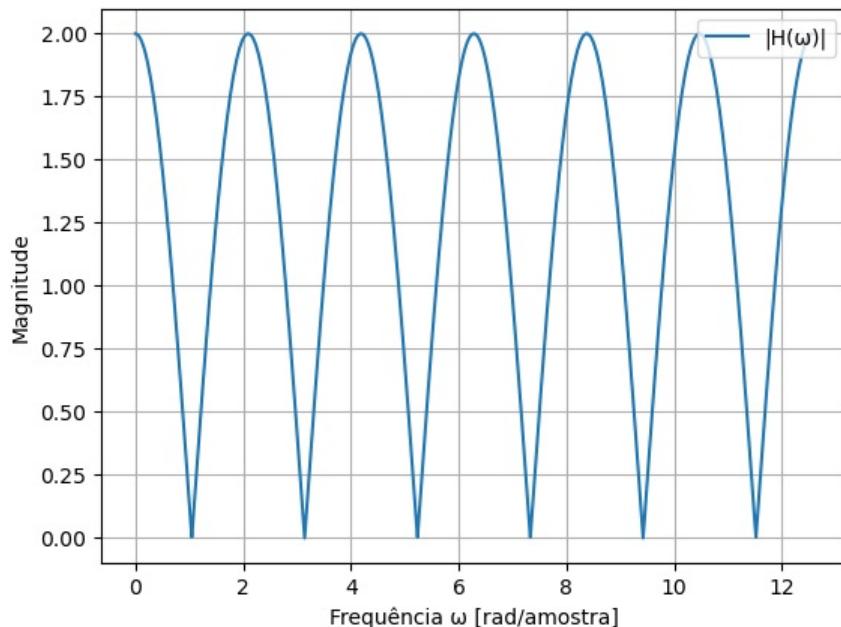
Figura 38 – Diagrama de blocos do Flanger.



Fonte: Autor.

Nomralmente no Flanger, o atraso $d[n]$ é menor do que $10ms$ e a frequência do **LFO** é menor ou igual a $5Hz$ ([GONTIJO; PAVEI; NOCETI, 2016](#)). Os principais parâmetros de ajustes deste efeito, são o tempo máximo de atraso, definido a partir da modificação de D , a frequência de oscilação do **LFO**, ajustada por f_c , além do ganho a que determina quanto do sinal atrasado é adicionado a saída. Na Figura 39 é apresentada a resposta em frequência do Flanger, que tem um comportamento muito semelhante a resposta do Delay, entretanto, a variação de $d[n]$ modifica constantemente a faixa das componentes de frequência que são amplificadas ou atenuadas, em múltiplos de $2\pi/d[n]$.

Figura 39 – Resposta Típica do Efeito de Flanger.



Fonte: Autor.

4 Metodologia de Implementação

Nas próximas seções deste capítulo serão discutidos o processo e a metodologia aplicados na fabricação da pedaleira, além do levantamento dos custos envolvidos.

4.1 Processo de Confecção

O presente trabalho tomou como base metodológica a pesquisa do tipo bibliográfica e experimental. A pesquisa bibliográfica é feita a partir do levantamento de referências teóricas já analisadas, e publicadas por meios escritos e eletrônicos, como livros, artigos científicos, páginas de web sites ([FONSECA, 2002](#)). Já a pesquisa experimental, envolve algum tipo de experimento. Consiste em determinar um objeto de estudo, selecionar as variáveis e definir as formas de controle e de observação dos efeitos ([SILVA, 2015](#)).

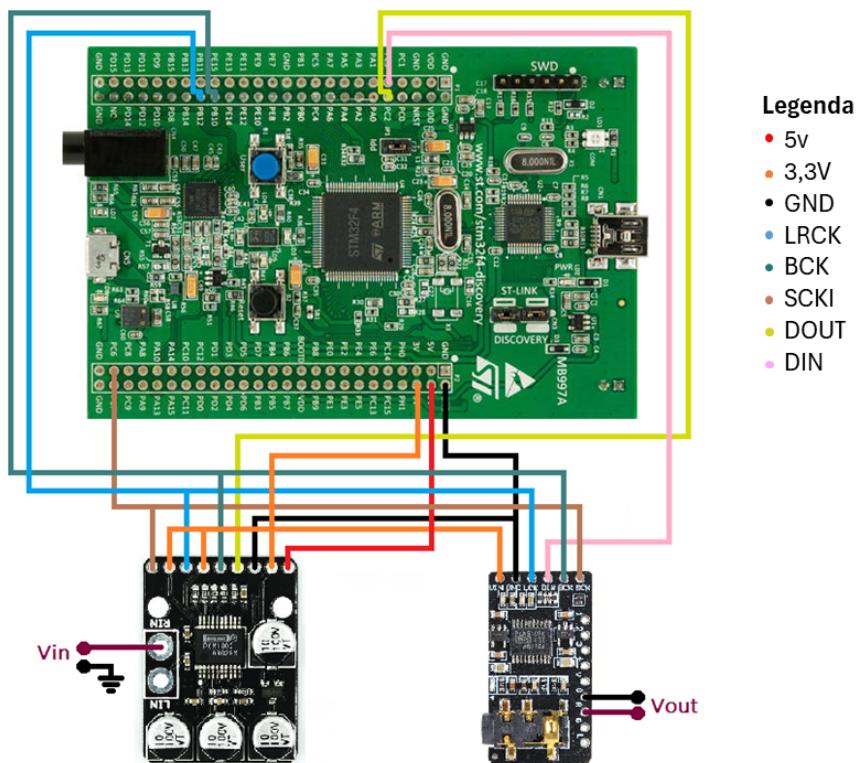
Durante o processo de desenvolvimento deste projeto, foram consultados livros, trabalhos de conclusão de curso e artigos que pudessem embasar o estudo e implementação de efeitos de áudio para guitarra em microcontroladores, revisitando conceitos da teoria de processamento digital de sinais, filtros analógicos, sistemas microprocessados e etc. A partir da base teórica estabelecida, foi então escolhido para a implementação do projeto a placa de desenvolvimento STM32F407DISCOVERY, pelo seu valor reduzido quando comparado a um [DSP](#) e por conta das suas ferramentas e bibliotecas que reduzem o tempo destinado ao entendimento e programação dos periféricos. Entretanto, essa placa apresenta limitações na resolução dos conversores [A/D](#) e [D/A](#), visto que, apresetam uma resolução de 12 bits. Pensando em obter maior fidedignidade entre o sinal original e o sinal processado, se fez necessário a utilização de conversores externos com maiores resoluções. Trabalhos como "*Pedal de Efeitos Digital Destinado a Guitarra Elétrica*" ([CONCEIÇÃO, 2022](#)) e "*Construção de uma Unidade Reprogramável de Efeito Tremolo para Guitarra Elétrica*" ([FERREIRA; FAGUNDES, 2022](#)) contribuíram na escolha dos conversores PCM1802 e PCM5102 para substituírem os conversores originais do microcontrolador.

Antes da realização da leitura do sinal de áudio pelos conversores e microcontrolador, a etapa inicial consistiu em projetar e implementar os circuitos destinados ao pré-processamento e reconstituição do sinal de áudio. O projeto dos filtros tomou como base bibliográfica o livro "*Principles of Active Network Synthesis and Design*" ([DARYANANI, 1976](#)), que desenvolve os princípios fundamentais da síntese de redes ativas e passivas a partir de considerações práticas de projeto. Com os filtros projetados, a simulação foi realizada utilizando o software LTspice para verificação da resposta em frequência. Já a etapa de Pré-Amplificação foi replicada de topologias já conhecidas em pedais comerciais. No que diz respeito a alimentação da pedaleira, as fontes [CC](#) comerciais geralmente

fornecem uma tensão de *9 Volts*, porém, devido ao STM32F407DISCOVERY operar com no máximo *5 Volts*, foi necessário utilizar um regulador de tensão para produzir os *5 Volts*, neste caso, o LM7805, junto a um transistor 2SA2222SG destinado a suprir a corrente exigida.

Com os circuitos analógicos devidamente projetados, simulados e com os componentes eletrônicos definidos, o passo seguinte foi realizar o teste em protoboard antes de partirmos para a confecção da Placa de Circuito Impresso (PCI). Um ponto muito importante na montagem do circuito é a comunicação entre os conversores A/D, D/A e o microcontrolador, aos quais são os principais elementos responsáveis pelo processamento digital e que exigem atenção nas conexões realizadas. Na Figura 40, é apresentado o esquema de como deve ser efetuada a ligação entre o microcontrolador e os periféricos.

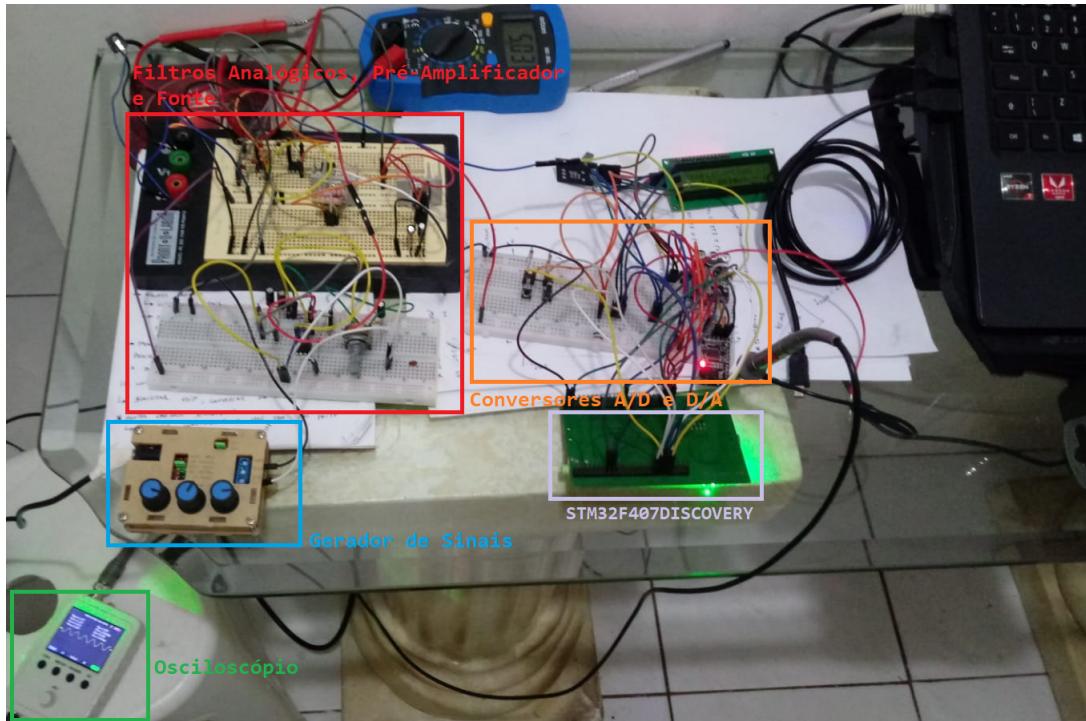
Figura 40 – Esquemático de Ligação do STM32 e Conversores A/D e D/A.



Fonte: Autor.

Com o entendimento das conexões necessárias entre todos os elementos do projeto, partiu-se para implementação do circuito da pedaleira em protoboard. Ao realizar essa etapa foi notada a necessidade da utilização de dissipadores de calor no circuito da fonte, para diminuir o aquecimento do regulador LM7805 e do transistor 2SA2222SG, por conta do consumo de corrente solicitado pelo microcontrolador e periféricos, que ficou em torno de $220mA$. Os dissipadores foram instalados nos dois componentes com a utilização de pasta térmica e micas isolantes, reduzindo substancialmente a temperatura da fonte durante a operação da pedaleira. A Figura 41 registra a montagem final do circuito em protoboard.

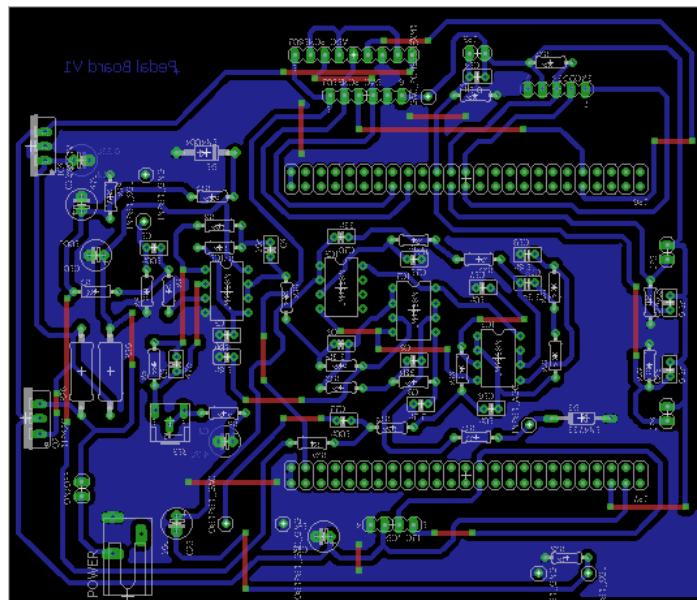
Figura 41 – Circuito da Pedaleira Montado na Protoboard.



Fonte: Autor.

Após a validação do circuito, foi então desenvolvido o layout da PCI no software CS Eagle, uma aplicação destinada ao design de PCI. A Figura 42, apresenta o layout final da pedaleira.

Figura 42 – Layout da Pedaleira.

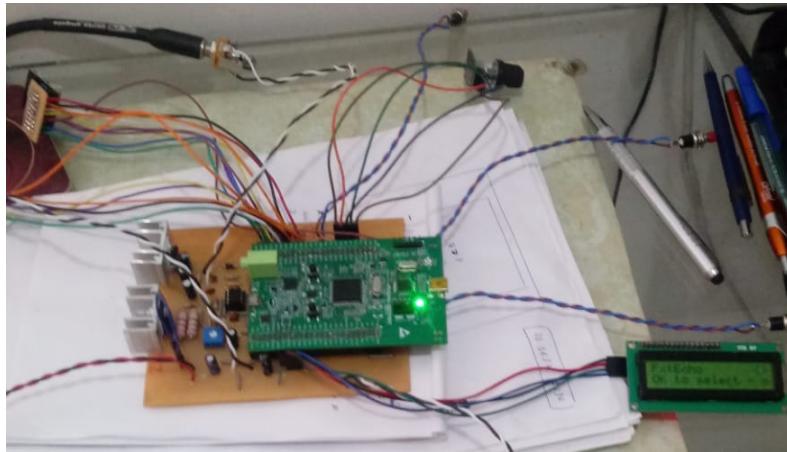


Fonte: Autor.

Por fim, a etapa final consistiu na confecção da placa de circuito, conforme indicado na Figura 43. Todos os componentes foram soldados e mais um teste foi executado para verificar se todas as funcionalidades da placa estavam operacionais. Toda a sua programação

foi realizada utilizando a plataforma STM32CubeIDE, por facilitar a programação dos periféricos do microcontrolador, deixando sob responsabilidade do desenvolvedor apenas a definição das especificações desejadas para operação dos periféricos e a programação da lógica de interesse.

Figura 43 – Placa de circuito impresso.



Fonte: Autor.

Com o circuito devidamente testado e todas as funcionalidades validadas, a placa foi então colocada em uma caixa de alumínio para proteção e blindagem eletromagnética. A Figura 44 apresenta o resultado final do projeto. Para a interface com o usuário, além do display LCD já apresentado, a pedaleira consta com quatro botões programáveis que podem assumir as funcionalidades desejadas pelo usuário, sendo que um deles trata-se de um encoder decoder rotacional.

Figura 44 – Pedaleira Desenvolvida.



Fonte: Autor.

4.2 Custo do Projeto

O valor final do projeto está majoritariamente associado ao custo das peças utilizadas, porém a mão de obra empregada, ferramentas utilizadas e os custos associados a transporte não deixam de ser relevantes. Optou-se por representar na Tabela 4.2 apenas os valores relacionados aos componentes utilizados, já que os outros custos são muito mais variáveis e subjetivos.

Tab. 4 – Tabela de Custos da Pedaleira

Componente	Quantidade	Preço por Unidade (R\$)
Conversor ADC PCM1802	1	25,54
Conversor DAC PCM5102	1	34,40
Placa STM32F407DISCOVERY	1	183,60
Amplificador Operacional RC4558	4	4,50
Diodo Zener 1N5231	1	0,30
Regulador de Tensão LM7805	1	5,50
Transistor 2SA2222	1	2,50
Encoder Decoder Rotacional KY-040	1	13,90
Jack P10 Mono	2	3,70
Jack P4 DC Femea	1	3,50
20m Cabo Flexível 1mm ²	5	39,00
Chave Gangorra 12 volts	1	15,00
Push Button NA	3	2,00
Caixa Metálica 200x60x200mm	1	109,00
Display LCD 16x2 I2C	1	25,00
Knobs Potenciômetro	1	3,80
Placa de Fenolite 23x20cm	1	20,00
Dissipador de Calor	2	4,50
Mica Isolante	2	0,45
Pasta Térmica	1	8,90
Soquete CI de 8 pinos	4	0,70
Barra de pinos fêmea 2x40	2	8,00
Fonte de 9 volts 1A	1	31,50
Diodo 1N4007	1	0,50
Resistor 120 Ohms 3W	1	1,70
Resistor 180 Ohms 3W	1	1,70
Resistor 390K Ohms 1/4W	1	0,20
Resistor 1K Ohms 1/4W	2	0,20

Resistor 10K Ohms 1/4W	4	0,20
Resistor 4,7K Ohms 1/4W	1	0,20
Resistor 56K Ohms 1/4W	1	0,20
Resistor 1M Ohms 1/4W	2	0,20
Resistor 10M Ohms 1/4W	1	0,20
Resistor 1,5K Ohms 1/4W	4	0,20
Resistor 6,8K Ohms 1/4W	4	0,20
Resistor 2K Ohms 1/4W	4	0,20
Resistor 100 Ohms 1/4W	1	0,20
Trimpot 500K Ohms	1	4,50
Capactior Eletrolítico 0,1uF	1	0,50
Capactior Eletrolítico 0,33uF	1	0,50
Capactior Eletrolítico 10uF	1	1,00
Capactior Eletrolítico 47uF	1	0,50
Capactior Eletrolítico 100uF	1	1,00
Capactior Eletrolítico 4,7uF	1	0,50
Capactior Cêramico 100nF	2	0,50
Capactior Cêramico 47pF	1	0,50
Capactior Cêramico 10nF	7	0,50
Capactior Cêramico 3,3nF	2	0,50
Capactior Cêramico 1,8nF	2	0,50
Capactior Cêramico 33nF	2	0,50
Capactior Cêramico 2,2nF	2	0,50
Total		763,94

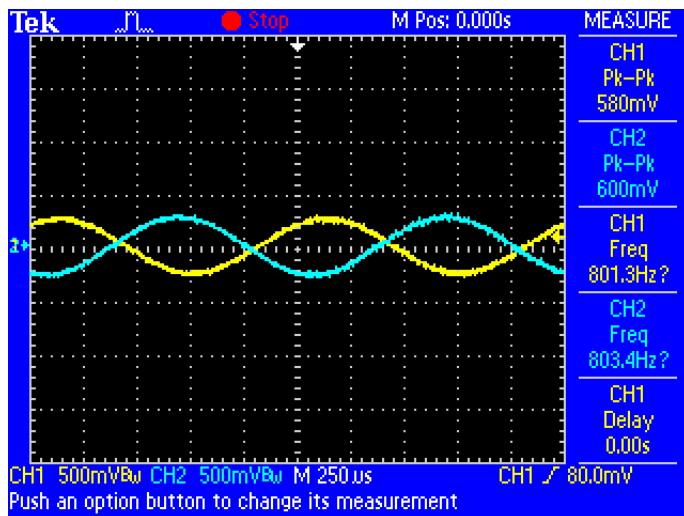
Fonte: Autor.

O custo final do projeto, se comparado com as pedaleiras mais simples do mercado acaba se tornando elevado, entretanto, vale ressaltar que o resultado final ainda não está otimizado, este preço poderia ser drasticamente reduzido, a partir da escolha de outros componentes e algumas modificações na sua arquitetura, além da compra de componentes em grande quantidade.

5 Resultados Experimentais

Este capítulo tem por objetivo tratar dos resultados obtidos a partir de testes realizados na pedaleira, a fim de validar seu correto funcionamento e comparar os resultados práticos com a teoria desenvolvida nos capítulos anteriores. Antes de iniciar a validação dos efeitos implementados, é interessante promover a comparação entre entrada e saída da pedaleira, para a verificação dos possíveis impactos ocasionados pelo processamento digital. Essa comparação é realizada na Figura 45.

Figura 45 – Comparaçāo dos Sinais de Entrada e Saída.



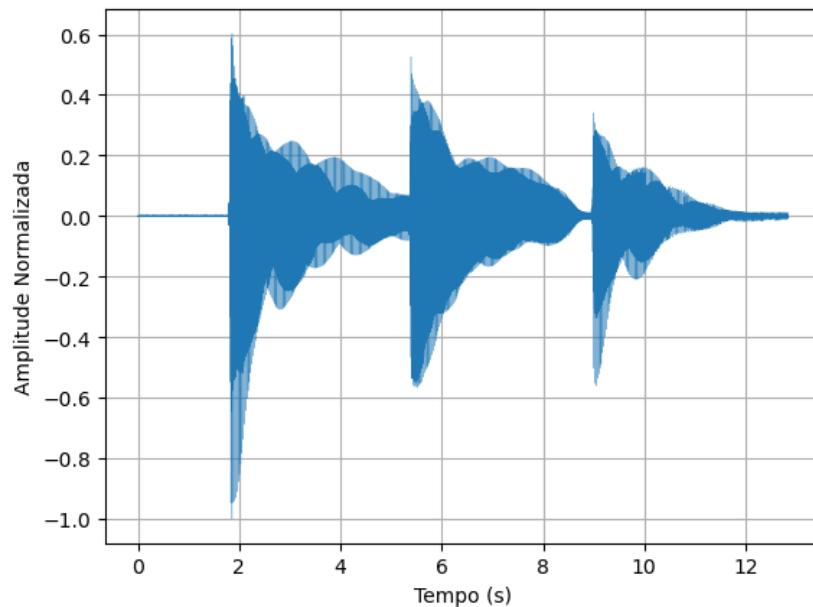
Fonte: Autor.

No teste realizado, foi introduzida uma senoide de 560mV de pico a pico, com uma frequência de aproximadamente 800Hz, vale ressaltar que a pequena diferença entre as frequências de entrada e saída se deve ao ruído imposto pelo meio e ferramentas utilizadas na medição. Ao processar o sinal de entrada, a pedaleira introduziu um atraso de 560μs no sinal de saída, além de um ganho de $A = \frac{600mV}{580mV} = 1,0345$. O atraso produzido pela pedaleira por ser muito pequeno acaba sendo imperceptível, já o ganho introduzido, está próximo do que foi calculado no projeto da etapa do Pré-Amplificador.

Nas próximas seções serão apresentados os resultados obtidos com cada um dos efeitos implementados com seus parâmetros no máximo, para avaliarmos a máxima influência de cada um dos efeitos digitais no sinal da guitarra, no seguinte repositório, [GitHub](#), podem ser encontrados todos os resultados apresentados nesta seção. Para a avaliação dos resultados produzidos pela pedaleira, foi feita a gravação da execução do power chord de mi (E5), tocado três vezes consecutivas. A gravação foi realizada através do software Audacity, conectando diretamente a saída da pedaleira a entrada de áudio de um notebook. Nas Figuras 46 e 47, são apresentadas as formas de onda do sinal de

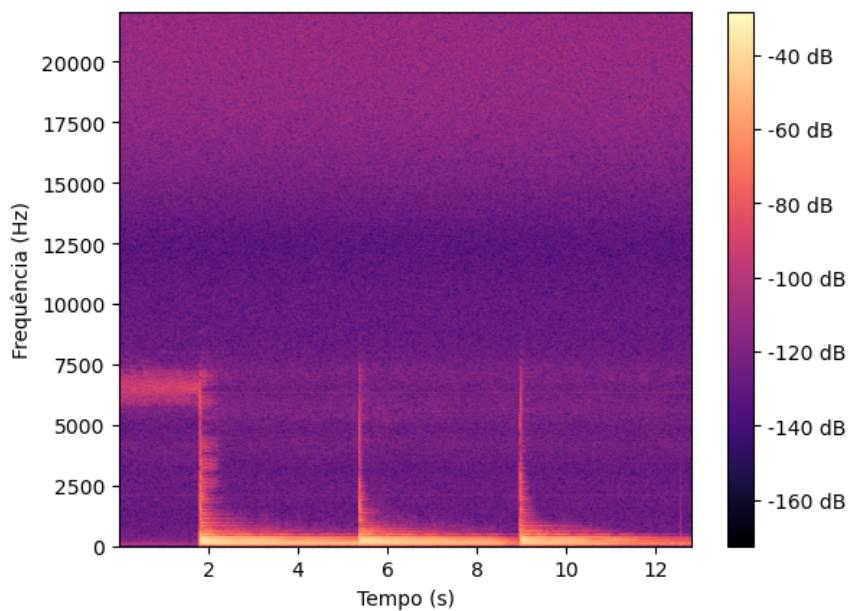
entrada e seu espectrograma.

Figura 46 – Forma de Onda do Sinal de Entrada.



Fonte: Autor.

Figura 47 – Espectrograma do Sinal de Entrada.

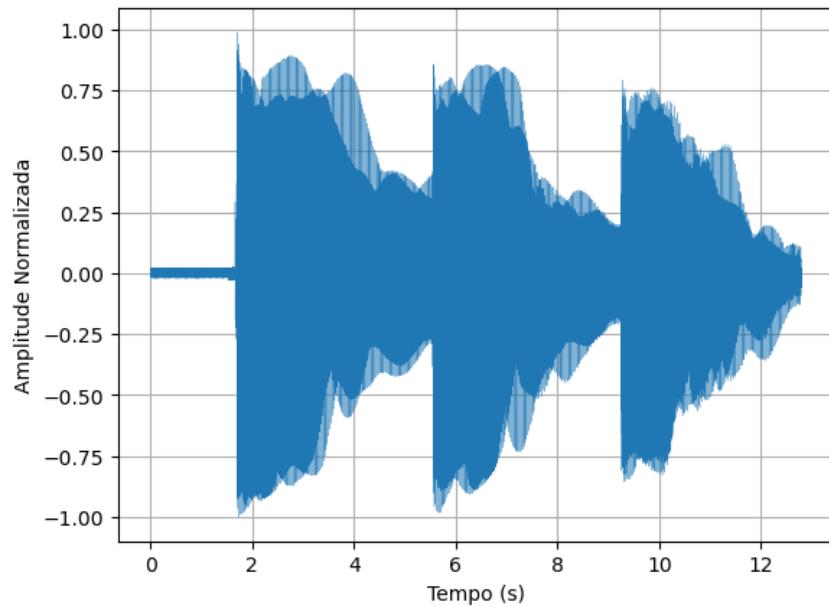


Fonte: Autor.

5.1 Distortion

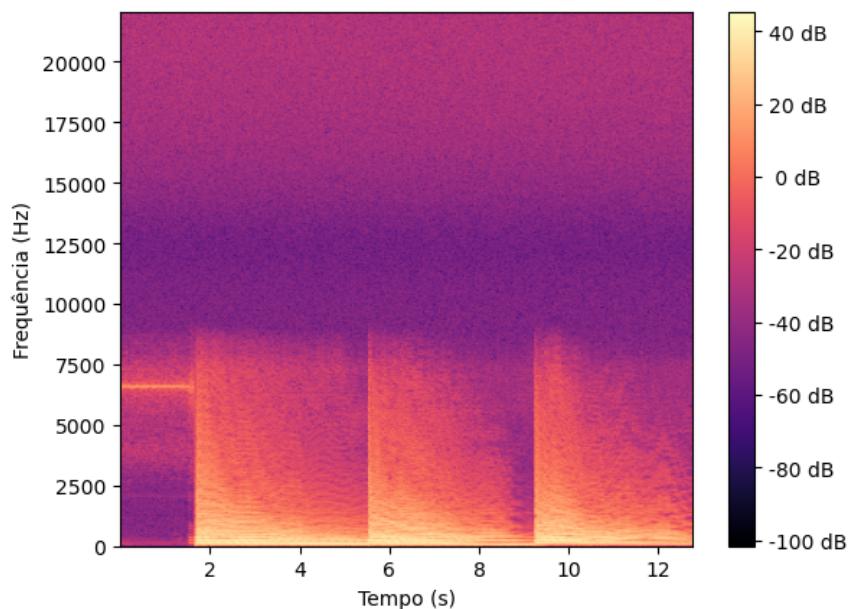
Para a implementação do efeito Distortion, tomou-se a liberdade de realizar uma combinação entre as equações (3.18) e (3.19), usando a ideia do corte não suave do Distortion e a amplificação dos valores intermediários do Overdrive, o que proporciona um som mais limpo para volumes mais baixos e um som mais distorcido para volumes mais altos. O resultado da aplicação deste efeito está nas Figuras 48 e 49.

Figura 48 – Forma de Onda do Distortion.



Fonte: Autor.

Figura 49 – Espectrograma do Distortion.



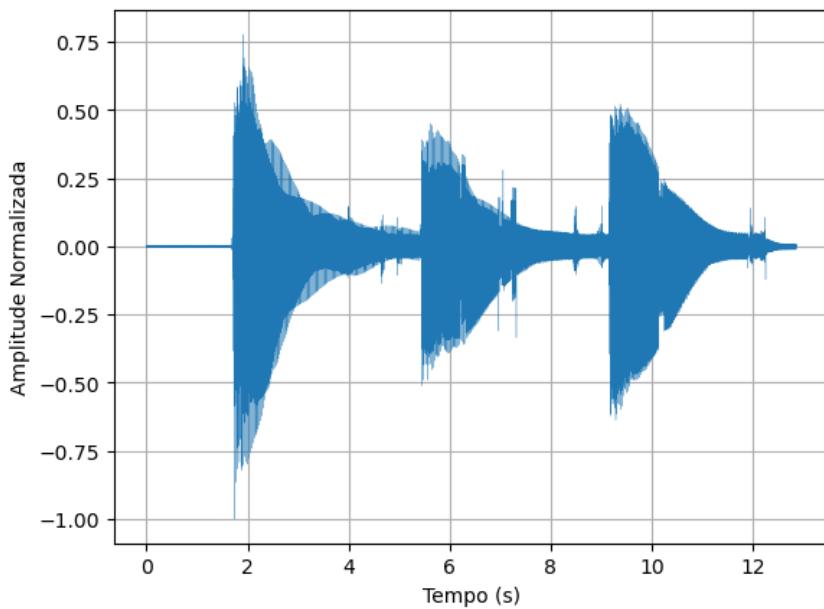
Fonte: Autor.

Analizando o sinal da pedaleira no domínio do tempo, fica evidente que o sinal da guitarra foi amplificado, mas não de maneira linear. Direcionando a atenção para o espectrograma da Figura 49, nota-se o surgimento de harmônicos que diminuem gradualmente. Proporcionando um som mais denso e com uma sustentação maior.

5.2 Delay

Na implementação do Delay foi utilizado um atraso de $D = 100ms$ e o ganho da parcela atrasada foi mantido como $a = 1$.

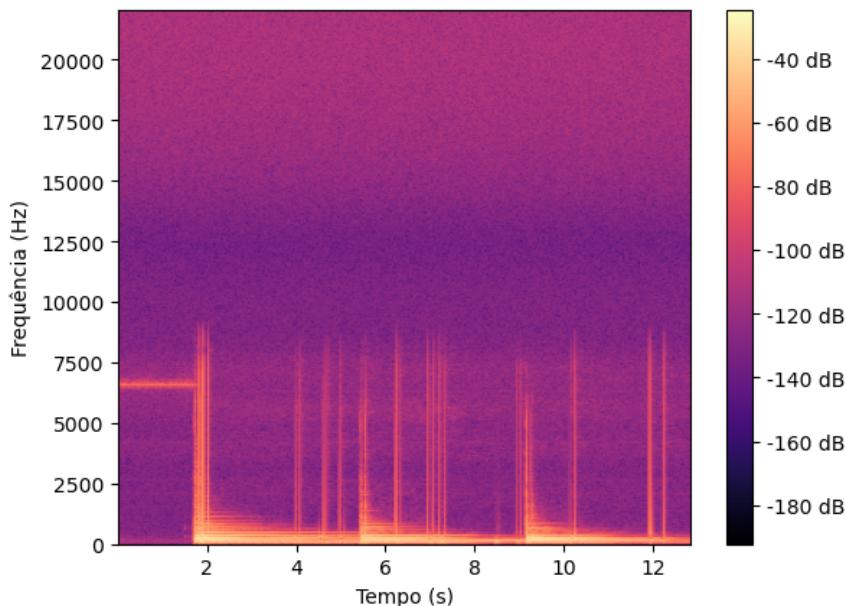
Figura 50 – Forma de Onda do Delay.



Fonte: Autor.

Por conta da adição das repetições ao sinal original, no domínio do tempo, Figura 50, temos o preenchimento de algumas lacunas presentes no som original, já ao verificar o espectrograma da Figura 51, visualiza-se um reforço em algumas componentes de frequência, enquanto outras são atenuadas. Nestas configurações obtém-se como resultado a repetição das notas tocadas quase que instantaneamente. Diminuindo a amplitude da parcela atrasada, é possível dar profundidade espacial ao som, como se o som estivesse sendo tocando em uma sala grande.

Figura 51 – Espectrograma do Delay.



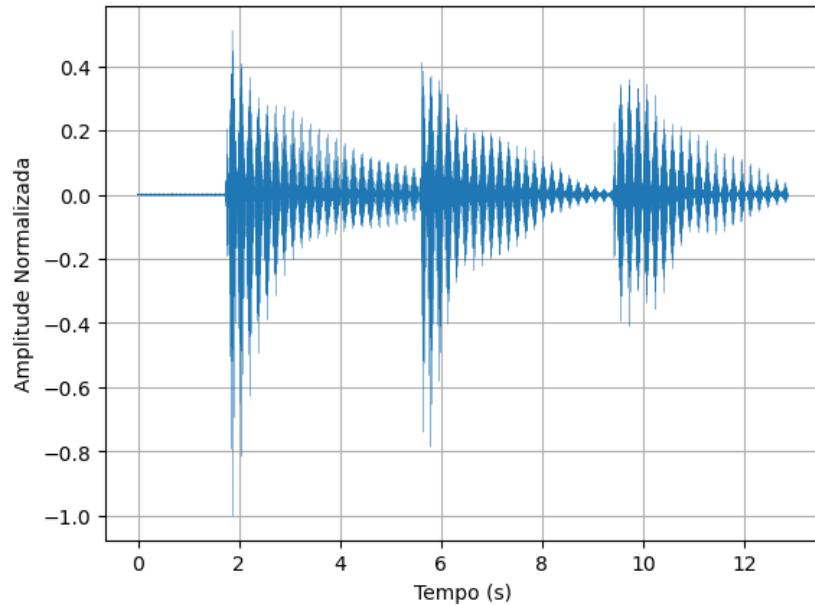
Fonte: Autor.

5.3 Tremolo

Durante a implementação do efeito de Tremolo, como forma de reduzir o esforço computacional exigido do processador, visto que, ao utilizar a função seno no código e habilitar um outro efeito de maneira simultânea com o Tremolo, o microcontrolador travava e perdia suas funcionalidades. Sendo assim, ao invés de realizar a modulação do sinal de entrada com uma onda senoidal como discutido anteriormente, optou-se por fazer a modulação com uma onda triangular, produzindo o mesmo efeito sem diferença audível notável ou qualquer tipo de travamento no microcontrolador. No teste apresentado nas Figuras 52 e 53, foi utilizada uma onda triangular com frequência de $f_c = 10\text{Hz}$ e amplitude $a = 1$.

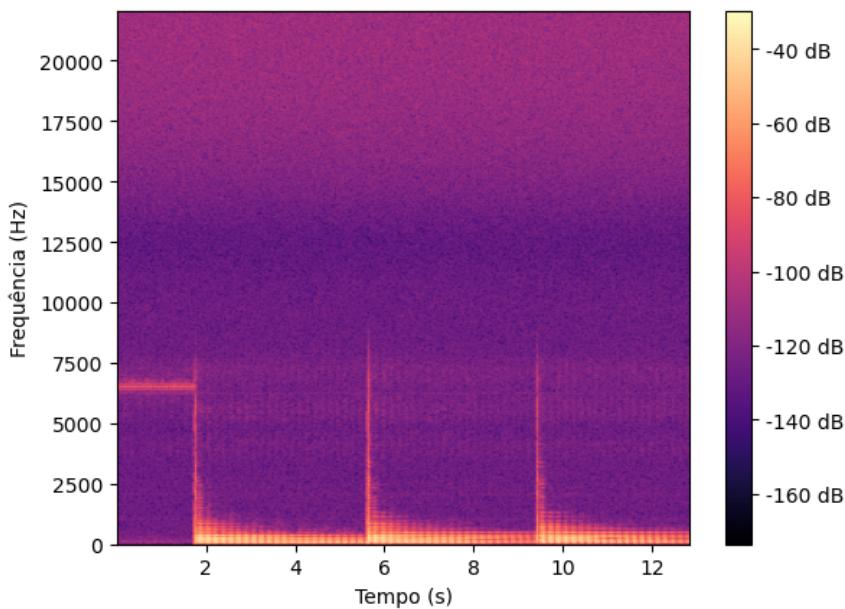
Como consequência da aplicação deste efeito, no domínio do tempo é possível observar a variação da amplitude do sinal da guitarra, resultando no aumento e diminuição constante do volume, gerando um som vibrante. No espectrograma, o efeito da variação da amplitude do sinal da guitarra fica ainda mais pronunciado, com a densidade espectral de potência sendo fortemente atenuada em alguns instantes e inalterada em outros.

Figura 52 – Forma de Onda do Tremolo.



Fonte: Autor.

Figura 53 – Espectrograma do Tremolo.



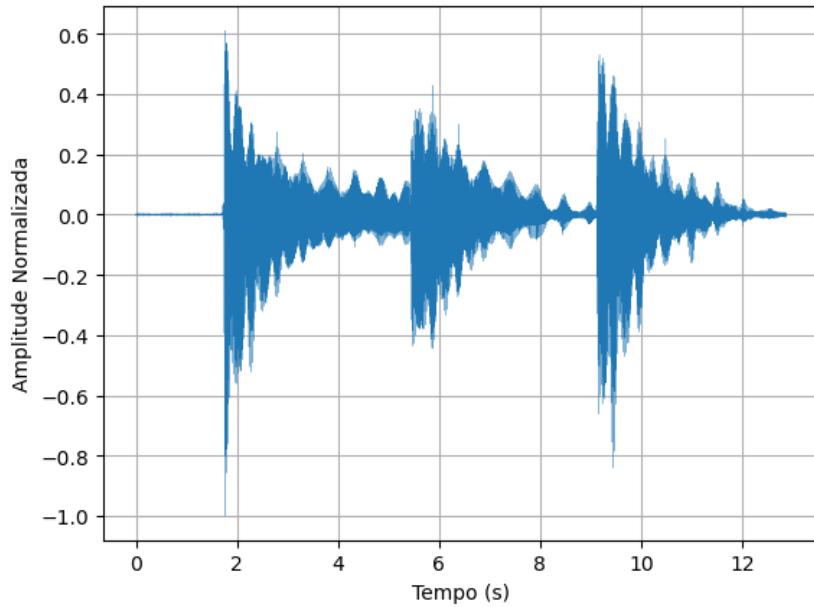
Fonte: Autor.

5.4 Flanger

Assim como na implementação do efeito de Tremolo, a modulação da quantidade de atrasado do Flanger também foi implementada utilizando uma onda triangular. Para o teste das Figuras 54 e 55, foi utilizada uma onda triangular de $f_c = 2\text{Hz}$ com um ganho do sinal em atraso de $a = 0,8$. Nesta implementação o valor de D será mantido fixo em 5ms .

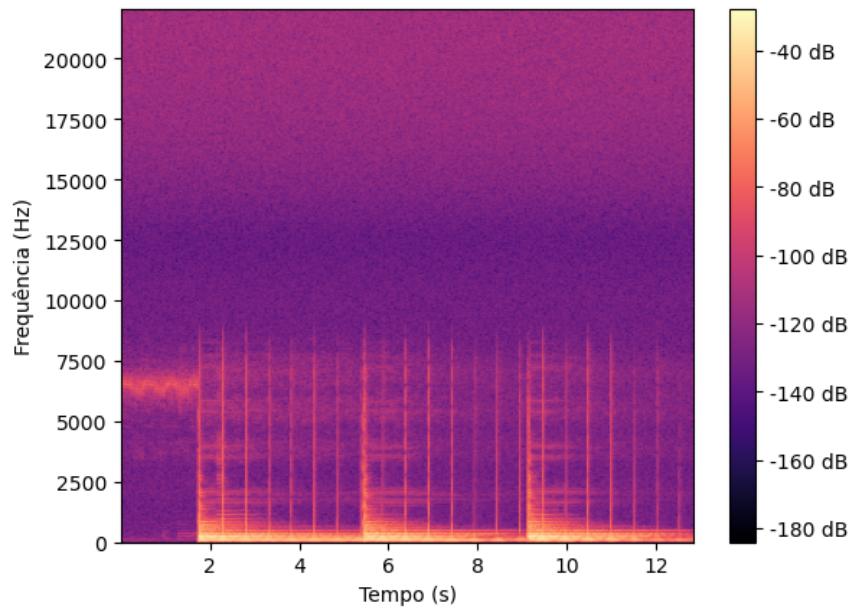
Por conta do atraso variável, no domínio do tempo observa-se uma oscilação

Figura 54 – Forma de Onda do Flanger.



Fonte: Autor.

Figura 55 – Espectrograma do Flanger.



Fonte: Autor.

constante na amplitude do sinal, porém mais suave que o Tremolo. No que diz respeito ao domínio da frequência, temos o efeito de ondulação no espectro do sinal, devido às interferências construtivas e destrutivas entre o sinal original e sua cópia atrasada. Este efeito produz ressonâncias no som da guitarra, dando mais profundidade, brilho e vibração.

6 Considerações Finais

A proposta deste trabalho foi o desenvolvimento de um hardware capaz de realizar o processamento digital de sinais e a sua utilização para implementar digitalmente os efeitos de guitarra Distortion, Delay, Tremolo e Flanger. O produto final cumpre as especificações desejadas e realiza tudo o que se pretendia. Os quatro efeitos programados se comportaram como discutido teoricamente, no que diz respeito a domínio do tempo e frequência, o mesmo pode ser dito em relação à sonoridade, onde ficou claro que sua qualidade não deixa a desejar mesmo com as limitações de hardware.

Todo o desenvolvimento deste projeto foi desafiador, necessitando muita resiliência e comprometimento. Dentre os grandes obstáculos enfrentados, o que mais se destaca é a falta de materiais ou aplicações práticas com a placa de desenvolvimento STM32F407DISCOVERY no processamento de sinais, além de documentações sobre a utilização dos conversores PCM1802 e PCM5102. Outro grande desafio encontrado foi durante os testes do circuito em protoboard, onde foi notado o super aquecimento do regulador responsável por alimentar o microcontrolador e os periféricos, sendo necessário estudar topologias de circuitos para fonte de alimentação que pudessem suprir a tensão e corrente necessárias, com baixo aquecimento do circuito. Mesmo com esses desafios e outros que surgiram ao longo do caminho, foi possível desenvolver o circuito da pedaleira e implementar os efeitos propostos.

Ainda que o projeto proposto tenha atendido aos objetivos, para futuros trabalhos, novas versões do pedal poderiam melhorar uma série de características como o aquecimento a partir da substituição do regulador linear por regulador chaveado, como um conversor Buck. Substituição do display LCD por um display Thin Film Transistor (TFT) que permitiria a apresentação de informações adicionais de uma forma esteticamente agradável. A inclusão de uma memória externa também seria uma grande melhoria para a pedaleira, possibilitando a implementação de Delays com atrasos maiores e outros efeitos como Reverb. Além disso, novos efeitos podem ser implementados, como Phaser, Wah-Wah, Whammy e etc.

Em suma, o desenvolvimento desta pedaleira é uma excelente forma de pôr em prática e exercitar conceitos desenvolvidos no estudo da teoria de processamento digital de sinais e eletrônica analógica, no que diz respeito ao desenvolvimento de filtros analógicos.

Referências

- ALIEXPRESS. Interface i2s pcm5102a dac decodificador gy-pcm5102 i2s. *ALIEXPRESS*, 2023. Acesso em: 10 out 2023. Disponível em: <<https://pt.aliexpress.com/item/1005002898278583.html>>. Citado na página 52.
- AMAZON. Pcm1802 24bit 105db audio stereo a/d converter adc decoder amplifier module. *AMAZON*, v. 7, 2023. Acesso em: 8 out 2023. Disponível em: <<https://www.amazon.com.br/PCM1802-Stereo-Converter-Decoder-Amplifier/dp/B0C9JSH1BP>>. Citado na página 51.
- BROWN, G. Discovering the stm32 microcontroller. *Bloomington: Indiana University*, p. 179–181, Jun 2016. Acesso em: 28 jun 2023. Disponível em: <<https://legacy.cs.indiana.edu/~geobrown/book.pdf>>. Citado na página 42.
- BURNS, M. Interfacing an i2s device to an msp430 device. *Texas Instruments*, Mar 2010. Acesso em: 26 out 2023. Disponível em: <<https://www.st.com/resource/en/application-note/cd00179121-connecting-i2s-audio-devices-to-the-str7-str9-mcu-stmicroelectronics.pdf>>. Citado na página 42.
- CASTRO, G. A. D. Guitarra elÉtrica: Entre o instrumento e a interface. *CONGRESSO DA ANPPOM*, 2007. Acesso em: 20 ago 2023. Disponível em: <https://www.anppom.org.br/anais/anaiscongresso_anppom_2007/sonologia/sonolog_GASCastro.pdf>. Citado na página 25.
- CONCEIÇĀO, E. A. D. ConceiÇĀo, emerson andrey da. pedal de efeitos digital destinado a guitarra elÉtrica. *TCC (Graduação) - Curso de Engenharia Elétrica, Universidade do Estado de Santa Catarina, Joinville*, Dec 2022. Acesso em: 12 jul 2023. Disponível em: <https://github.com/Emersonandrey11/Digital-Guitar-Stompbox/blob/main/Relat_TCC_2_.pdf>. Citado 2 vezes nas páginas 58 e 67.
- DARYANANI, G. Principles of active network synthesis and design. *John Wiley & Sons Ltd*, 1976. Citado na página 67.
- DINIZ, P.; SILVA, E. D. Processamento digital de sinais: Projeto e análise de sistemas. *Bookman*, v. 2, 2014. Citado 6 vezes nas páginas 33, 34, 35, 36, 37 e 39.
- DUNLOP, J. Product manual: M133 micro amp. *Dunlop*, 1984. Acesso em: 30 out 2023. Disponível em: <<https://www.jimdunlop.com/content/manuals/M133.pdf>>. Citado na página 45.
- FERREIRA, G. S.; FAGUNDES, F. D. ConstruÇĀo de uma unidade reprogramÁvel de efeito tremolo para guitarra elÉtrica. *Conferênciade Estudos em Engenharia Elétrica, Uberlândia*, v. 1, n. 20, p. 1–6, Dec 2022. Acesso em: 7 jul 2023. Disponível em: <https://www.peteletricaufu.com.br/static/ceel/artigos/artigo_720.pdf>. Citado na página 67.
- FONSECA, J. J. S. Metodologia da pesquisa científica. *UECE - Universidade Estadual do Ceará*, 2002. Acesso em: 18 nov 2023. Disponível em: <<http://www.ia.ufrrj.br/ppgea/conteudo/conteudo-2012-1/1SF/Sandra/apostilaMetodologia.pdf>>. Citado na página 67.

GONTIJO, W. A.; PAVEI, A. H.; NOCETI, S. Implementações de efeitos de Áudio utilizando arduino due e pedalshield. *Sociedade de Engenharia de Áudio Palestra em Congresso*, May 2016. Acesso em: 1 nov 2023. Disponível em: <<https://www.linse.ufsc.br/~sidnei/artigos.php>>. Citado 4 vezes nas páginas 59, 62, 63 e 65.

HATSCHEK, K.; MONLEY, J. Guitar effects pedals and the evolution of music – part 1. *Conservatory Of Music Faculty Articles, Stockton*, Feb 2016. Acesso em: 26 jun 2023. Disponível em: <<https://scholarlycommons.pacific.edu/com-facarticles/18/>>. Citado na página 25.

INSTRUMENTS, T. Rc4558 dual general-purpose operational amplifier. *Texas Instruments*, Oct 2014. Acesso em: 10 ago 2023. Disponível em: <<https://www.ti.com/lit/ds/symlink/rc4558.pdf>>. Citado na página 45.

INSTRUMENTS, T. Pcm1802 single-ended analog-input 24-bit, 96-khz stereo a/d converter. *Texas Instruments*, Dec 2016. Acesso em: 21 jun 2023. Disponível em: <https://www.ti.com/lit/ds/symlink/pcm1802.pdf?ts=1687113889917&ref_url=https%253A%252F%252Fwww.google.com%252F>. Citado na página 51.

INSTRUMENTS, T. Direct memory access (dma) controller module. *Texas Instruments*, Aug 2018. Acesso em: 3 nov 2023. Disponível em: <https://www.ti.com/lit/ug/slau395f/slau395f.pdf?ts=1698812502374&ref_url=https%253A%252F%252Fwww.google.com%252F>. Citado na página 42.

JUNIOR, A. P. Amplificadores operacionais e filtros ativos. *Bookman*, v. 8, p. 161, 2007. Citado na página 47.

KARREN, C. Analog versus digital guitar pedals, shaping guitar tones and sparking debates. *CAPSTONE PROJECTS AND MASTER'S THESES*, May 2020. Acesso em: 11 set 2023. Disponível em: <https://digitalcommons.csumb.edu/cgi/viewcontent.cgi?article=1793&context=caps_thes_all>. Citado na página 26.

KATZ, D.; GENTILE, R.; LUKASIAK, T. Fundamentals of embedded audio, part 3. *EETimes*, Sep 2007. Acesso em: 24 out 2023. Disponível em: <<https://www.eetimes.com/fundamentals-of-embedded-audio-part-3/>>. Citado na página 57.

LATHI, B. P. Sinais e sistemas lineares. *Bookman*, v. 2, 2007. Citado 7 vezes nas páginas 29, 31, 32, 37, 38, 51 e 52.

LEWIS, J. Common inter-ic digital interfaces for audio data transfer. *Analog Devices*, Jan 2012. Acesso em: 25 out 2023. Disponível em: <<https://www.analog.com/media/en/technical-documentation/technical-articles/ms-2275.pdf>>. Citado na página 41.

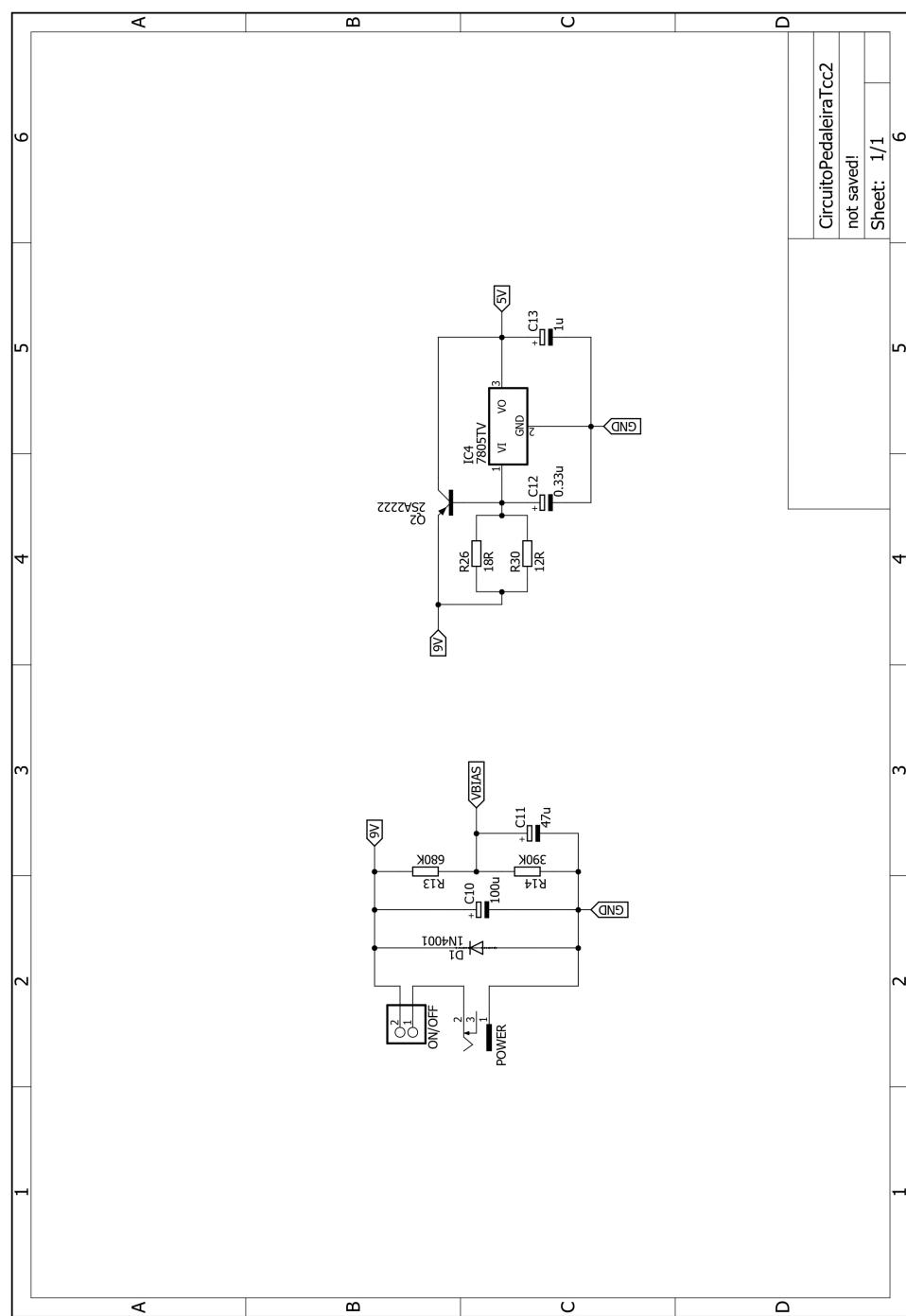
OPPENHEIM, A. V.; SCHAFER, R. W. Processamento em tempo discreto de sinais: Sinais de tempo discreto. *Pearson Education do Brasil*, v. 3, 2014. Citado 13 vezes nas páginas 25, 26, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39 e 40.

RAZAVI, B. Fundamentos de microeletrônica. *LTC — Livros Técnicos e Científicos Editora S.A.*, v. 7, Oct 2010. Citado 2 vezes nas páginas 45 e 50.

- ROBOTICA, C. D. Display lcd azul 16x2 com módulo i2c soldado serial 16x02 1602. *CASA DA ROBOTICA*, 2023. Acesso em: 17 nov 2023. Disponível em: <<https://www.casadarobotica.com/display/display/display-lcd-azul-16x2-com-modulo-i2c-serial-ja-soldado-16x02-1602>>. Citado na página 55.
- SILVA, A. M. D. Metodologia da pesquisa. *Ed UAB/UECE*, v. 2, 2015. Acesso em: 18 nov 2023. Disponível em: <https://educapes.capes.gov.br/bitstream/capes/432206/2/Livro_Metodologia%20da%20Pesquisa%20-%20Comum%20a%20todos%20os%20cursos.pdf>. Citado na página 67.
- SOUZA, N. D. Guitarra elétrica: um ícone na cultura pop do século xx. *Revista Vernáculo, Universidade Federal do Paraná*, v. 1, n. 5, p. 33–45, Jun 2002. Acesso em: 17 jun 2023. Disponível em: <<https://revistas.ufpr.br/vernaculo/article/viewFile/18452/12000>>. Citado na página 25.
- ST. Connecting i2s audio devices to the str7/str9 mcu. *ST*, Jan 2008. Acesso em: 25 out 2023. Disponível em: <https://www.st.com/resource/en/application_note/cd00179121-connecting-i2s-audio-devices-to-the-str7-str9-mcu-stmicroelectronics.pdf>. Citado na página 41.
- ST. Discovery kit with stm32f407vg mcu - user manual. *ST*, v. 7, Oct 2020. Acesso em: 16 ago 2023. Disponível em: <https://www.st.com/resource/en/user_manual/um1472-discovery-kit-with-stm32f407vg-mcu-stmicroelectronics.pdf>. Citado na página 54.
- VALDEZ, J.; BECKER, J. Understanding the i2c bus. *Texas Instruments*, Jun 2015. Acesso em: 18 jun 2023. Disponível em: <https://www.ti.com/lit/an/slva704/slva704.pdf?ts=1698727206402&ref_url=https%253A%252F%252Fwww.google.com%252F>. Citado na página 41.
- VARGAS, A. A guitarra baiana: Tradição e desuso. *XI CONGRESO IASPMAL*, p. 387–393, 2014. Acesso em: 18 jun 2023. Disponível em: <<https://iaspmal.com/index.php/2016/03/02/actas-xi-congreso/?lang=pt>>. Citado na página 25.
- ZOLZER, U. Dafx: Digital audio effects. *John Wiley & Sons Ltd*, v. 2, Jan 2011. Citado 3 vezes nas páginas 59, 62 e 64.

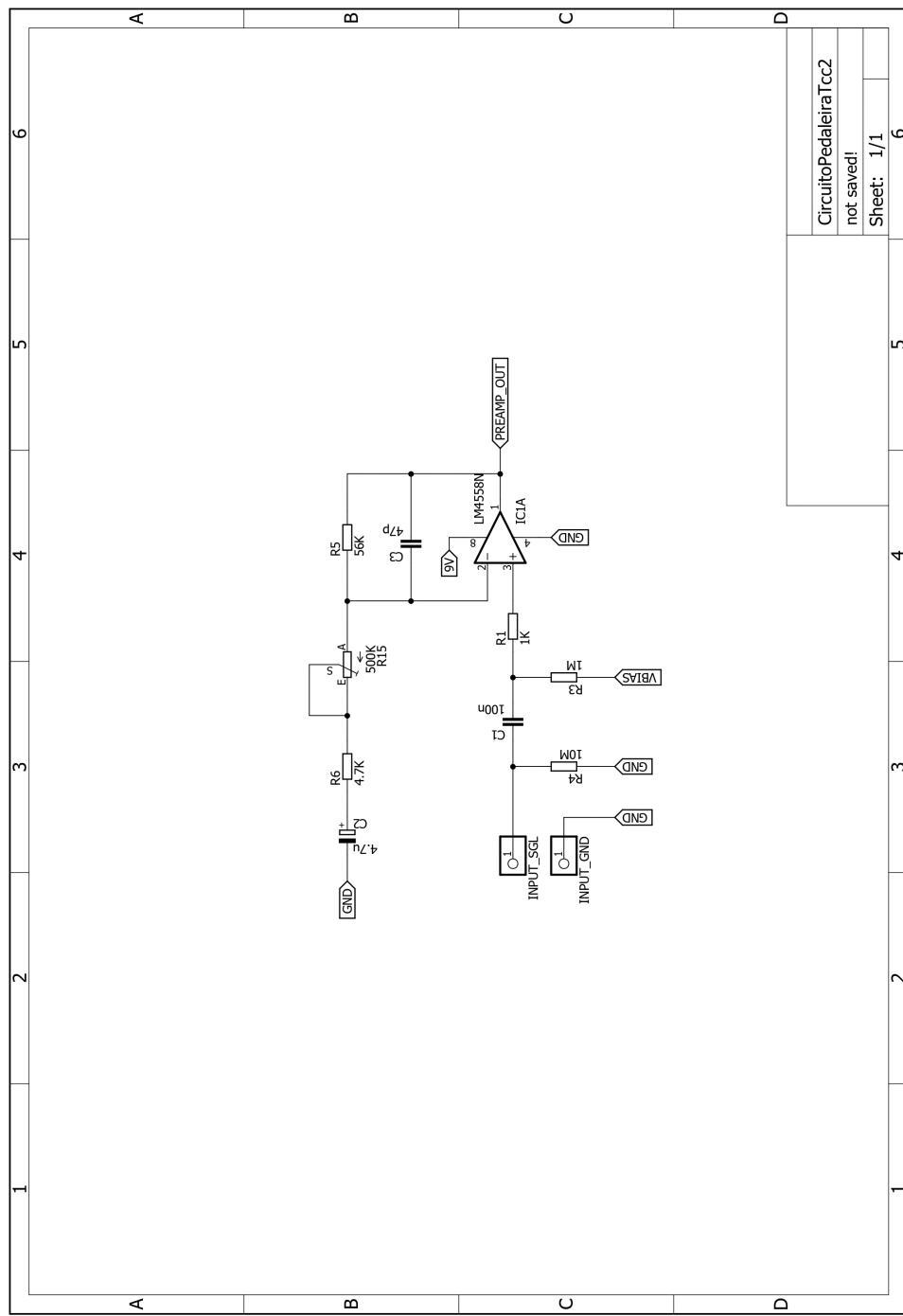
APÊNDICE A – Circuito Desenvolvido

Figura 56 – Esquemático Fonte de Alimentação.



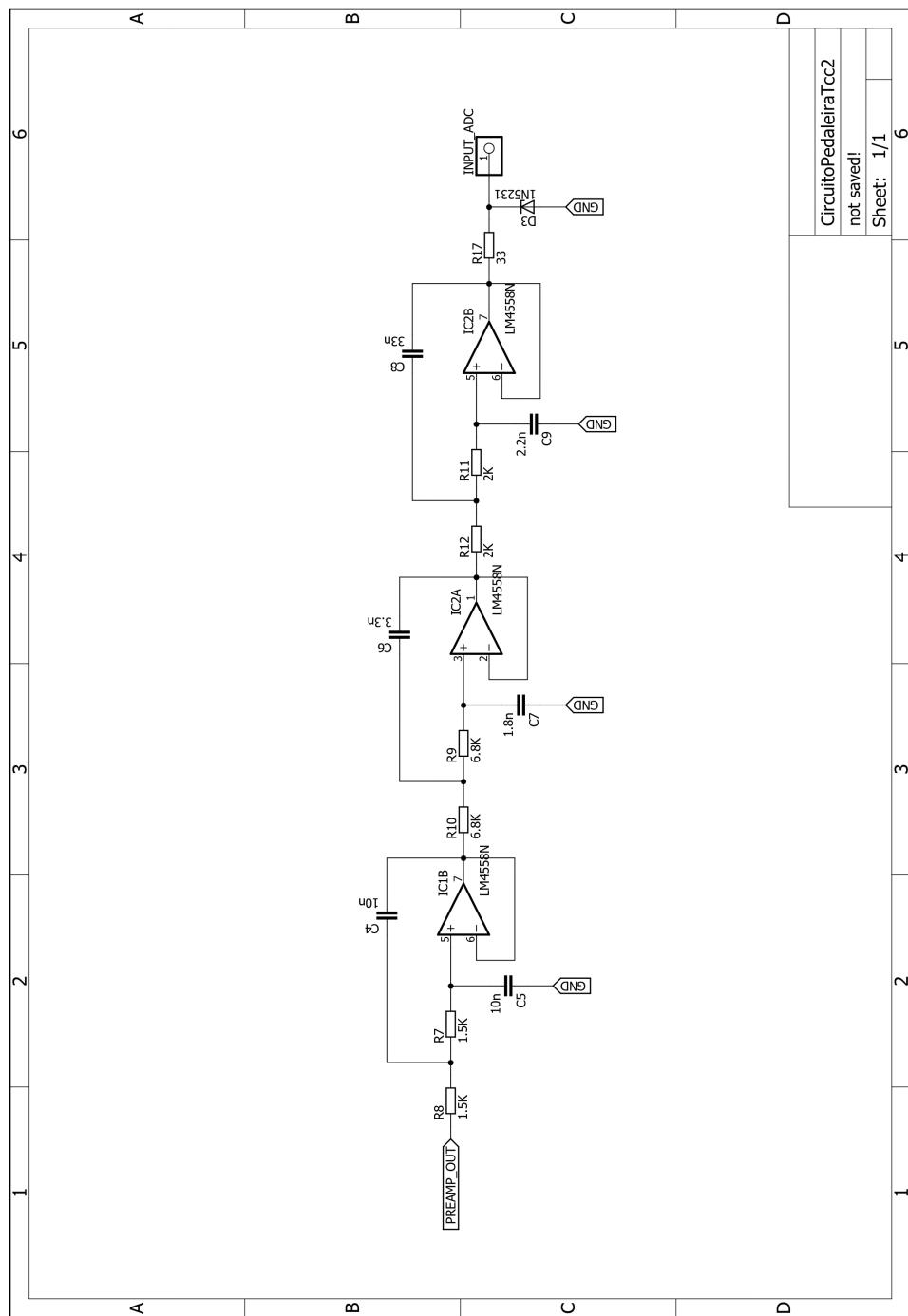
Fonte: Autor.

Figura 57 – Esquemático Pré-Amplificador.



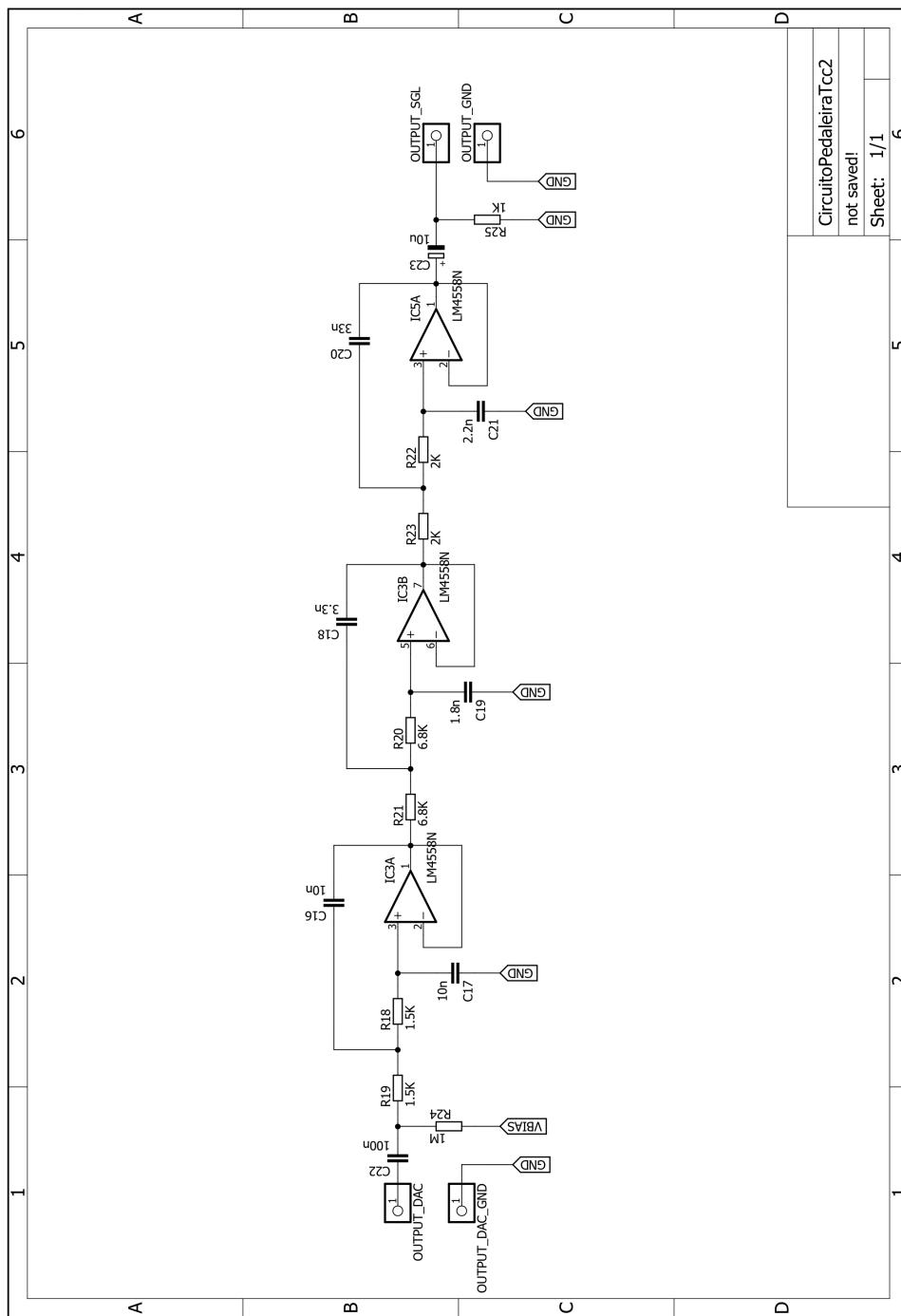
Fonte: Autor.

Figura 58 – Esquemático Filtro Anti-Aliasing.



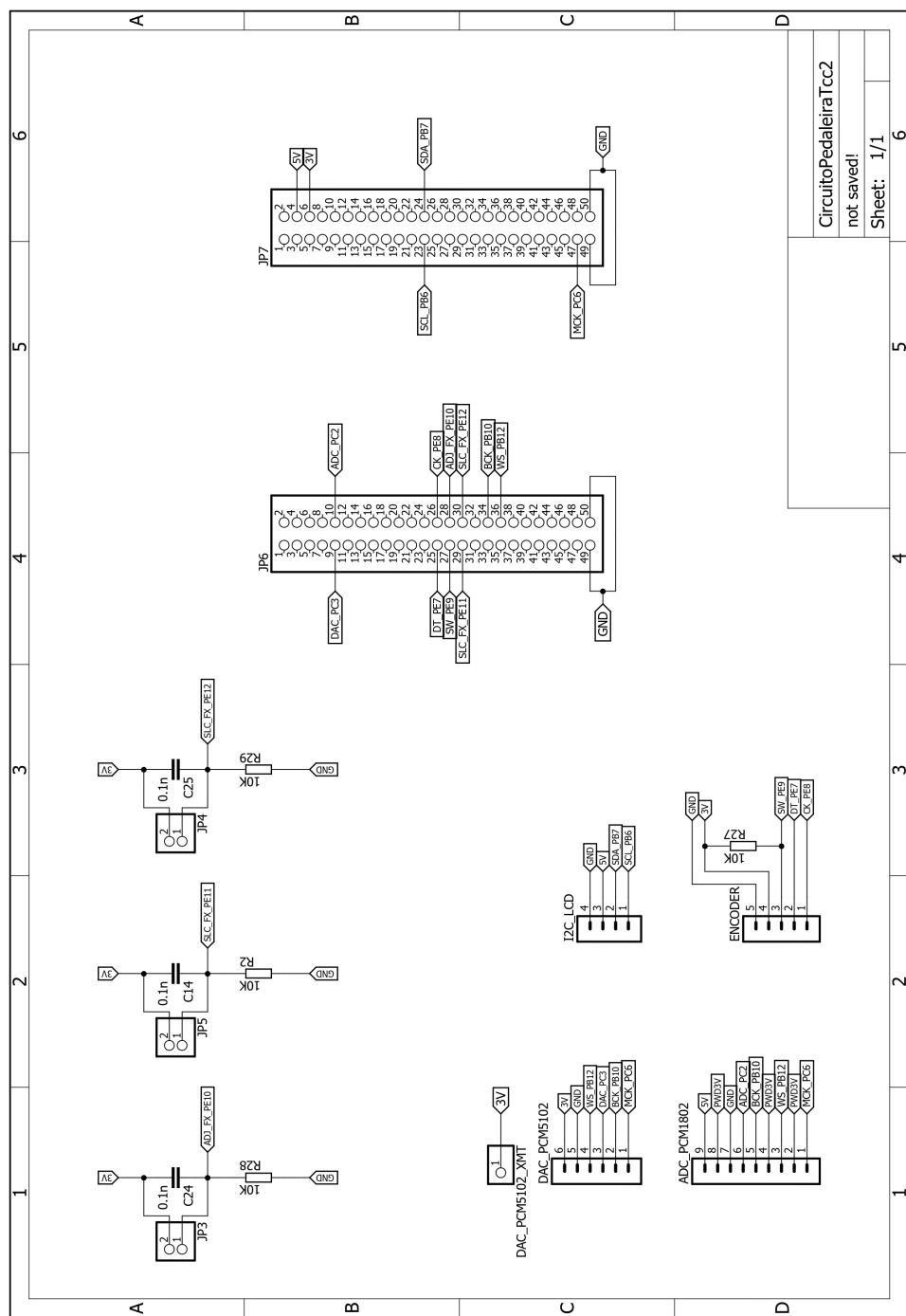
Fonte: Autor.

Figura 59 – Esquemático Filtro de Reconstrução.



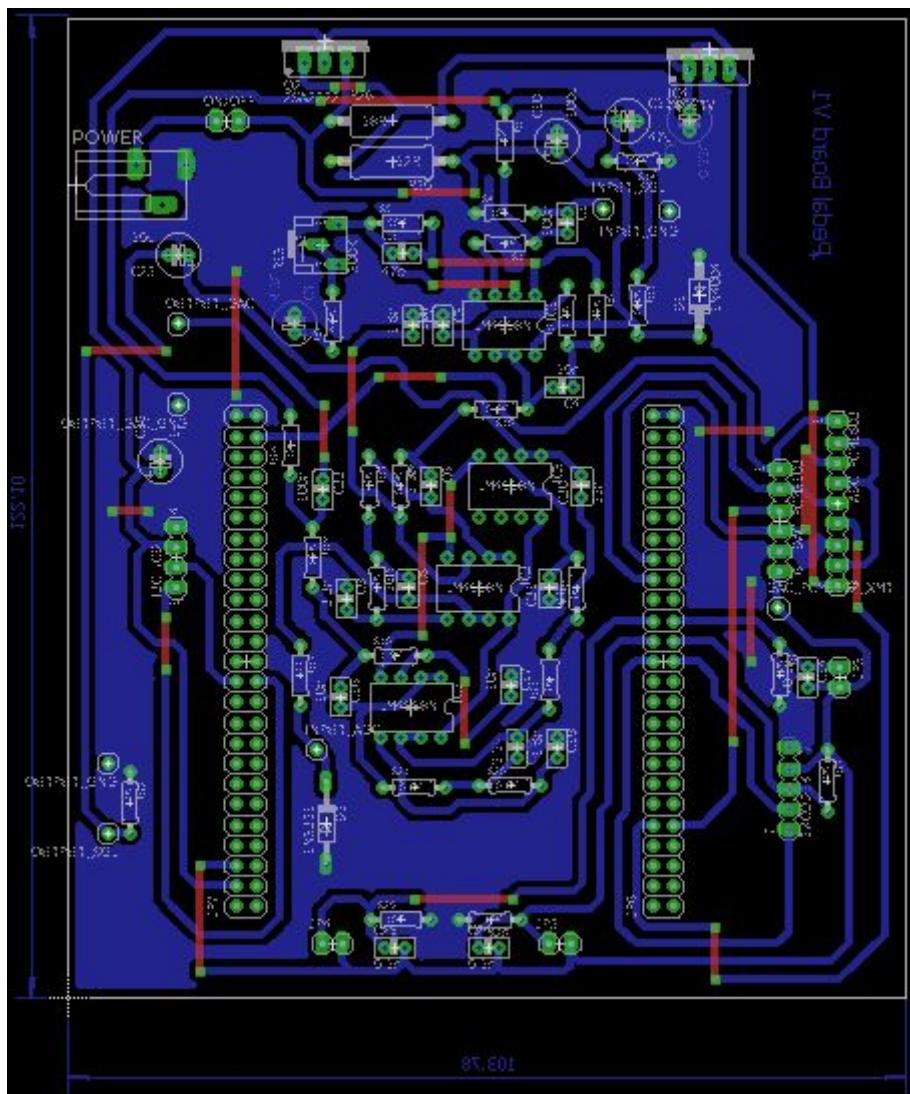
Fonte: Autor.

Figura 60 – Esquemático Microcontrolador e Periféricos.



Fonte: Autor.

Figura 61 – Layout Final da Pedaleira.



Fonte: Autor.

APÊNDICE B – Código Implementado

```

1  /* Includes _____*/
2 #include "main.h"
3
4  /* Private includes _____*/
5 /* USER CODE BEGIN Includes */
6
7 #define ARM_MATH_CM4
8 #include "math.h"
9 #include "arm_math.h"
10 #include "stdio.h"
11 #include "i2c-lcd.h"
12 /* USER CODE END Includes */
13
14 /* Private typedef _____*/
15 /* USER CODE BEGIN PTD */
16
17 /* USER CODE END PTD */
18
19 /* Private define _____*/
20 /* USER CODE BEGIN PD */
21 #define Fs 96000.0f
22 #define MAX_DELAY 9600
23 #define MAX_DELAY_Flg 480
24 #define threshold 62500000.0f
25
26 /* USER CODE END PD */
27
28 /* Private macro _____*/
29 /* USER CODE BEGIN PM */
30
31 /* USER CODE END PM */
32
33 /* Private variables _____*/
34 I2C_HandleTypeDef hi2c1;
35 I2S_HandleTypeDef hi2s2;
36 DMA_HandleTypeDef hdma_i2s2_ext_rx;
37 DMA_HandleTypeDef hdma_spi2_tx;
38
39 /* USER CODE BEGIN PV */
40 /* USER CODE END PV */
41
42 /* Private function prototypes _____*/
43 void SystemClock_Config(void);

```

```
44 static void MX_GPIO_Init( void );
45 static void MX_DMA_Init( void );
46 static void MX_I2S2_Init( void );
47 static void MX_I2C1_Init( void );
48 /* USER CODE BEGIN PFP */
49
50 // Funções do Menu
51 void text_effect( char *txt, uint8_t n );
52 void text_prm( char *txt, int8_t r, int8_t c, int8_t num );
53 int8_t encoder_cnt( int8_t prm, int8_t prm_mx );
54
55 // Funções Efeitos
56 int effects( float32_t sample );
57 float32_t distortion( float32_t sample );
58 float32_t echo( float32_t sample );
59 float32_t tremolo( float32_t sample );
60 float32_t flanger( float32_t sample );
61 /* USER CODE END PFP */
62
63 /* Private user code -----*/
64 /* USER CODE BEGIN 0 */
65 // Buffers de recepção e transmissão de dados
66 uint16_t rxBuf[8];
67 uint16_t txBuf[8];
68 float32_t yout, youtFx = 0, y_lp = 0, y_1 = 0, x_1 = 0;;
69
70 // flags para navegação no menu e leitura de botões
71 uint8_t flag = 0, flag_enc = 0, menu = 0, count = 0, effect = 0, prm = 0,
    prm_mx = 0, n1 = 0, n2 = 0;
72 GPIO_PinState State = GPIO_PIN_RESET, LastState = GPIO_PIN_RESET;
73
74 // Parâmetros dos efeitos
75 float32_t vol = 0.5;
76 uint8_t glb_lv = 99;
77 uint8_t flg_gn = 50, dst_gn = 50, ech_gn = 70;
78 uint8_t ech_dp = 80, trm_dp = 80;
79 uint8_t flg_fq = 40, trm_fq = 50;
80
81 // Echo
82 float32_t buffer[MAX_DELAY];
83 uint16_t oldest = 0;
84
85 // Tremolo
86 float32_t trgWave_trm = 0;
87 int8_t inc_trm = 1;
88
89 // Flanger
```

```

90 float32_t BufferFlg[MAX_DELAY_Flg];
91 float32_t trgWave_flg = 0;
92 int8_t inc_flg = 1;
93 uint16_t oldestFlg = 0, currentFlg = 0, Dint = 0;
94
95 /* USER CODE END 0 */
96
97 /**
98 * @brief The application entry point.
99 * @retval int
100 */
101 int main(void)
102 {
103     /* USER CODE BEGIN 1 */
104
105     /* USER CODE END 1 */
106
107     /* MCU Configuration
108
109     * Reset of all peripherals, Initializes the Flash interface and the
110     * Systick.
111     */
112     HAL_Init();
113
114     /* USER CODE BEGIN Init */
115
116     /* Configure the system clock */
117     SystemClock_Config();
118
119     /* USER CODE BEGIN SysInit */
120
121     /* USER CODE END SysInit */
122
123     /* Initialize all configured peripherals */
124     MX_GPIO_Init();
125     MX_DMA_Init();
126     MX_I2S2_Init();
127     MX_I2C1_Init();
128     /* USER CODE BEGIN 2 */
129     lcd_init();
130     HAL_I2SEx_TransmitReceive_DMA(&hi2s2, txBuf, rxBuf, 4);
131     LastState = HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_7);
132
133     lcd_put_cur(0, 0);
134     lcd_send_string("Inicializando...");
```

```
135 HAL_Delay(500);
136 lcd_put_cur(1, 0);
137 lcd_send_string ("HDX - DSP");
138 HAL_Delay(1500);
139 lcd_clear();
140 /* USER CODE END 2 */
141
142 /* Infinite loop */
143 /* USER CODE BEGIN WHILE */
144
145 while (1)
{
146     /* USER CODE END WHILE */
147
148     /* USER CODE BEGIN 3 */
149
150
151 // Selecao de Efeitos
152 vol = (0.005)*glb_lv; // Ajuste de volume, comum a todos os efeitos
153 if (menu == 0){
154     prm = 0;
155     effect = encoder_cnt(effect, 4);
156     // Catalogo de efeitos
157     switch (effect) {
158         case 0:
159             text_effct("Clean      >", effect);
160             prm_mx = 0;
161             break;
162         case 1:
163             text_effct("Distortion <>", effect);
164             prm_mx = 2;
165             break;
166         case 2:
167             text_effct("Echo      <>", effect);
168             prm_mx = 3;
169             break;
170         case 3:
171             text_effct("Tremolo    <>", effect);
172             prm_mx = 3;
173             break;
174         case 4:
175             text_effct("Flanger   <", effect);
176             prm_mx = 3;
177             break;
178     }
179
180     count++;
181     if (count > 10){
```

```

182     count = 0;
183     // Botao de ajuste de efeito
184     if ((HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_10)) && (flag == 0)) {
185         flag = 1;
186         menu = 1;
187     } else if (!HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_10)) {
188         // Botao que define o efeito , ate 2 efeitos
189         if ((HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_12)) && (flag == 0)) {
190             flag = 1;
191             if (n1 == 0 && n2 == 0) {
192                 n1 = effect;
193             } else if (n1 == effect) {
194                 n1 = 0;
195             } else if (n2 == 0 && n1 != 0) {
196                 n2 = effect;
197             } else if (n2 == effect) {
198                 n2 = 0;
199             }
200         } else if (!HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_12)) {
201             flag = 0;
202         }
203     }
204 }
205
206 }
207
208 // Selecao de parametros
209 if (menu == 1){
210     switch (effect) {
211     case 0:
212         glb_lv = encoder_cnt(glb_lv, 99);
213         text_prm("Lv:", 1, 0, glb_lv);
214         break;
215     case 1:
216         if (prm == 0) {
217             glb_lv = encoder_cnt(glb_lv, 99);
218             text_prm("Lv:", 1, 0, glb_lv);
219         } else {
220             dst_gn = encoder_cnt(dst_gn, 99);
221             text_prm("Gn:", 1, 0, dst_gn);
222         }
223         break;
224     case 2:
225         if (prm == 0) {
226             glb_lv = encoder_cnt(glb_lv, 99);
227             text_prm("Lv:", 1, 0, glb_lv);
228         } else if (prm == 1){

```

```

229         ech_gn = encoder_cnt(ech_gn, 99);
230         text_prm("Gn:", 1, 0, ech_gn);
231     } else {
232         ech_dp = encoder_cnt(ech_dp, 99);
233         text_prm("Dp:", 1, 0, ech_dp);
234     }
235     break;
236 case 3:
237     if (prm == 0) {
238         glb_lv = encoder_cnt(glb_lv, 99);
239         text_prm("Lv:", 1, 0, glb_lv);
240     } else if (prm == 1){
241         trm_dp = encoder_cnt(trm_dp, 99);
242         text_prm("Dp:", 1, 0, trm_dp);
243     } else {
244         trm_fq = encoder_cnt(trm_fq, 99);
245         text_prm("Fq:", 1, 0, trm_fq);
246     }
247     break;
248 case 4:
249     if (prm == 0) {
250         glb_lv = encoder_cnt(glb_lv, 99);
251         text_prm("Lv:", 1, 0, glb_lv);
252     } else if (prm == 1){
253         flg_gn = encoder_cnt(flg_gn, 99);
254         text_prm("Gn:", 1, 0, flg_gn);
255     } else {
256         flg_fq = encoder_cnt(flg_fq, 99);
257         text_prm("Fq:", 1, 0, flg_fq);
258     }
259     break;
260 }
261
262 count++;
263 if (count >= 10){
264     count = 0;
265     // Botao de ajuste de efeito
266     if ((HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_10)) && (flag == 0)) {
267         flag = 1;
268         menu = 0;
269         lcd_clear();
270     } else if (!HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_10)) {
271         // botao do Encoder (SW)
272         if ((!HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_9)) && (flag == 0)) {
273             flag = 1;
274             if (prm < prm_mx) {
275                 prm++;

```

```

276         } else {
277             prm = 0;
278         }
279     } else if (HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_9)) {
280         flag = 0;
281     }
282 }
283 }
284 }
285 }
286 /* USER CODE END 3 */
287 }
288
289 /**
290 * @brief System Clock Configuration
291 * @retval None
292 */
293 void SystemClock_Config(void)
294 {
295     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
296     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
297
298     /** Configure the main internal regulator output voltage
299     */
300     __HAL_RCC_PWR_CLK_ENABLE();
301     __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
302
303     /** Initializes the RCC Oscillators according to the specified parameters
304     * in the RCC_OscInitTypeDef structure.
305     */
306     RCC_OscInitStruct.OscillatorType = RCC OSCILLATORTYPE_HSE;
307     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
308     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
309     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
310     RCC_OscInitStruct.PLL.PLLM = 4;
311     RCC_OscInitStruct.PLL.PLLN = 168;
312     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
313     RCC_OscInitStruct.PLL.PLLQ = 4;
314     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
315     {
316         Error_Handler();
317     }
318
319     /** Initializes the CPU, AHB and APB buses clocks
320     */
321     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
322                         |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;

```

```
323     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
324     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
325     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
326     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
327
328     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
329     {
330         Error_Handler();
331     }
332 }
333
334 /**
335 * @brief I2C1 Initialization Function
336 * @param None
337 * @retval None
338 */
339 static void MX_I2C1_Init(void)
340 {
341
342     /* USER CODE BEGIN I2C1_Init_0 */
343
344     /* USER CODE END I2C1_Init_0 */
345
346     /* USER CODE BEGIN I2C1_Init_1 */
347
348     /* USER CODE END I2C1_Init_1 */
349     hi2c1.Instance = I2C1;
350     hi2c1.Init.ClockSpeed = 100000;
351     hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
352     hi2c1.Init.OwnAddress1 = 0;
353     hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
354     hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
355     hi2c1.Init.OwnAddress2 = 0;
356     hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
357     hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
358     if (HAL_I2C_Init(&hi2c1) != HAL_OK)
359     {
360         Error_Handler();
361     }
362     /* USER CODE BEGIN I2C1_Init_2 */
363
364     /* USER CODE END I2C1_Init_2 */
365
366 }
367
368 /**
369 * @brief I2S2 Initialization Function
```

```

370 * @param None
371 * @retval None
372 */
373 static void MX_I2S2_Init( void )
374 {
375     /* USER CODE BEGIN I2S2_Init_0 */
376
377     /* USER CODE END I2S2_Init_0 */
378
379     /* USER CODE BEGIN I2S2_Init_1 */
380
381     /* USER CODE END I2S2_Init_1 */
382     hi2s2.Instance = SPI2;
383     hi2s2.Init.Mode = I2S_MODE_MASTER_TX;
384     hi2s2.Init.Standard = I2S_STANDARD_PHILIPS;
385     hi2s2.Init.DataFormat = I2S_DATAFORMAT_24B;
386     hi2s2.Init.MCLKOutput = I2S_MCLKOUTPUT_ENABLE;
387     hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_96K;
388     hi2s2.Init.CPOL = I2S_CPOL_LOW;
389     hi2s2.Init.ClockSource = I2S_CLOCK_PLL;
390     hi2s2.Init.FullDuplexMode = I2S_FULLDUPLEXMODE_ENABLE;
391     if (HAL_I2S_Init(&hi2s2) != HAL_OK)
392     {
393         Error_Handler();
394     }
395     /* USER CODE BEGIN I2S2_Init_2 */
396
397     /* USER CODE END I2S2_Init_2 */
398
399 }
400
401 /**
402 * Enable DMA controller clock
403 */
404
405 static void MX_DMA_Init( void )
406 {
407
408     /* DMA controller clock enable */
409     __HAL_RCC_DMA1_CLK_ENABLE();
410
411     /* DMA interrupt init */
412     /* DMA1_Stream3_IRQHandler interrupt configuration */
413     HAL_NVIC_SetPriority(DMA1_Stream3_IRQn, 0, 0);
414     HAL_NVIC_EnableIRQ(DMA1_Stream3_IRQn);
415     /* DMA1_Stream4_IRQHandler interrupt configuration */
416     //HAL_NVIC_SetPriority(DMA1_Stream4_IRQn, 0, 0);

```

```

417 //HAL_NVIC_EnableIRQ(DMA1_Stream4_IRQn);
418
419 }
420
421 /**
422 * @brief GPIO Initialization Function
423 * @param None
424 * @retval None
425 */
426 static void MX_GPIO_Init( void )
427 {
428     GPIO_InitTypeDef GPIO_InitStruct = {0};
429
430     /* GPIO Ports Clock Enable */
431     __HAL_RCC_GPIOH_CLK_ENABLE();
432     __HAL_RCC_GPIOC_CLK_ENABLE();
433     __HAL_RCC_GPIOE_CLK_ENABLE();
434     __HAL_RCC_GPIOB_CLK_ENABLE();
435     __HAL_RCC_GPIOA_CLK_ENABLE();
436
437     /*Configure GPIO pins : PE7 PE8 PE9 PE10
438                               PE11 PE12 */
439     GPIO_InitStruct.Pin = GPIO_PIN_7|GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10
440                           |GPIO_PIN_11|GPIO_PIN_12;
441     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
442     GPIO_InitStruct.Pull = GPIO_NOPULL;
443     HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
444
445 }
446
447 /* USER CODE BEGIN 4 */
448
449 // _____ Funcoes de tratamento do audio _____
450
451 // Leitura do sinal de audio
452 void HAL_I2SEx_TxRxHalfCpltCallback(I2S_HandleTypeDef *hi2s)
453 {
454     // Restaura a amostra de 24 bit sinalizada a partir dos buffers de 16 bit
455     int lSample = (int) (rxBuf[0]<<16)|rxBuf[1];
456     int rSample = (int) (rxBuf[2]<<16)|rxBuf[3];
457
458     // Divide as amostra por 2 -> -3dB por amostra
459     lSample = lSample>>1;
460     rSample = rSample>>1;
461
462     // Filtro digital passa-baixa
463     y_lp = (0.9366f)*y_1 + (0.0634f)*x_1;

```

```

464     x_1 = rSample + lSample;
465     y_1 = y_lp;
466
467     // Aplicacao dos efeitos
468     youtFx = effects ((y_lp));
469
470     // Restaura o buffer para transmissao
471     txBuf[0] = (youtFx>>16)&0xFFFF;
472     txBuf[1] = youtFx&0xFFFF;
473     txBuf[2] = (youtFx>>16)&0xFFFF;
474     txBuf[3] = youtFx&0xFFFF;
475 }
476
477 void HAL_I2SEX_TxRxCpltCallback(I2S_HandleTypeDef *hi2s)
478 {
479     // Restaura a amostra de 24 bit sinalizada a partir dos buffers de 16 bit
480     int lSample = (int) (rxBuf[4]<<16)|rxBuf[5];
481     int rSample = (int) (rxBuf[6]<<16)|rxBuf[7];
482
483     // Divide as amostra por 2 -> -3dB por amostra
484     lSample = lSample>>1;
485     rSample = rSample>>1;
486
487     // Filtro digital passa-baixa
488     y_lp = (0.9366f)*y_1 + (0.0634f)*x_1;
489     x_1 = rSample + lSample;
490     y_1 = y_lp;
491
492     // Aplicacao dos efeitos
493     youtFx = effects ((y_lp));
494
495     // Restaura o buffer para transmissao
496     txBuf[4] = (youtFx>>16)&0xFFFF;
497     txBuf[5] = youtFx&0xFFFF;
498     txBuf[6] = (youtFx>>16)&0xFFFF;
499     txBuf[7] = youtFx&0xFFFF;
500 }
501
502 // _____ Selecao de Efeitos _____
503 int effects (float32_t sample) {
504     if (n1 == 1 || n2 == 1){
505         sample = distortion (sample);
506     }
507     if (n1 == 2 || n2 == 2){
508         sample = echo (sample);
509     }
510     if (n1 == 3 || n2 == 3){

```

```

511     sample = tremolo(sample);
512 }
513 if (n1 == 4 || n2 == 4){
514     sample = flanger(sample);
515 }
516
517 return sample*(vol);
518 }
519
520 // Efeitos
521 float32_t distortion (float32_t sample) {
522     float32_t gain = 1 + (0.02*dst_gn);
523
524     if ((gain*sample) > threshold){
525         yout = threshold;
526     } else if ((gain*sample) < -threshold){
527         yout = -threshold;
528     } else {
529         yout = gain*sample;
530     }
531
532     return yout;
533 }
534
535 float32_t echo(float32_t sample){
536     float32_t delayed = buffer[oldest];
537     uint16_t delayDP = ((MAX_DELAY * (ech_dp + 1))/100);
538     float32_t alpha = (0.01*ech_gn);
539
540     yout = (sample + (alpha * delayed));
541
542     buffer[oldest] = sample;
543     if (oldest++ >= delayDP) {
544         oldest = 0;
545     }
546
547     return yout;
548 }
549
550 float32_t tremolo(float32_t sample){
551     float32_t freq = (0.1*trm_fq); // 0 a 10Hz
552     float32_t alpha = (0.01*trm_dp); // 0 a 1
553
554     trgWave_trm = trgWave_trm + inc_trm*((2*freq)/Fs);
555     if(trgWave_trm >= 1 && inc_trm == 1){
556         inc_trm = -1;
557     } else if(trgWave_trm <= 0){

```

```

558     inc_trm = 1;
559 }
560
561 yout = sample*(1.0f - (alpha*trgWave_trm));
562
563 return yout;
564 }
565
566 float32_t flanger(float32_t sample){
567     float32_t freq = (0.05*flg_fq); // 0 a 5Hz
568     float32_t alpha = (0.01*flg_gn); // 0 a 1
569
570     BufferFlg[oldestFlg] = sample;
571     oldestFlg = (oldestFlg + 1);
572     if (oldestFlg >= MAX_DELAY_Flg){
573         oldestFlg = 0;
574     }
575
576     Dint = MAX_DELAY_Flg*trgWave_flg;
577
578     trgWave_flg = trgWave_flg + inc_flg*((2*freq)/Fs);
579     if(trgWave_flg >= 1 && inc_flg == 1){
580         inc_flg = -1;
581     } else if(trgWave_flg <= 0){
582         inc_flg = 1;
583     }
584
585     currentFlg = (oldestFlg + MAX_DELAY_Flg - Dint)%MAX_DELAY_Flg;
586     yout = sample + (alpha*BufferFlg[currentFlg]);
587
588     return yout;
589 }
590
591 // _____ Funcoes do LCD _____
592 void text_effect(char *txt, uint8_t n){
593     lcd_put_cur(0, 0);
594     lcd_send_string("Fx:");
595     lcd_put_cur(0, 3);
596     lcd_send_string(txt);
597     lcd_put_cur(1, 0);
598     if (n1 == n || n2 == n) {
599         lcd_send_string("OK to select - x");
600     } else {
601         lcd_send_string("OK to select - o");
602     }
603 }
```

```

605 void text_prm(char *txt, int8_t r, int8_t c, int8_t num){
606     int dezena = num / 10;
607     int unidade = num % 10;
608     char crtDezena = '0' + dezena;
609     char crtUnidade = '0' + unidade;
610
611     lcd_put_cur(r, c);
612     lcd_send_string(txt);
613     lcd_put_cur(r, c+3);
614     lcd_send_data(crtDezena);
615     lcd_put_cur(r, c+4);
616     lcd_send_data(crtUnidade);
617     lcd_put_cur(r, c+6);
618     lcd_send_string("        ");
619 }
620
621 // _____ Funcoes de controle do Encoder _____
622 int8_t encoder_cnt(int8_t prm, int8_t prm_mx){
623
624     State = HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_7);
625     if ((State != LastState) && (!State)){
626         /* Se o estado do DT for diferente do estado do CK, isso significa que
627         o Encoder esta girando no sentido horario */
628         if (HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_8) != State) {
629             if (prm < prm_mx){
630                 prm++;
631             }
632             else {
633                 if (prm > 0){
634                     prm--;
635                 }
636             }
637             LastState = State; // Atualiza o estado anterior do CK com o estado atual
638
639             return prm;
640     }
641 // _____
642
643 /* USER CODE END 4 */
644
645 /**
646 * @brief This function is executed in case of error occurrence.
647 * @retval None
648 */
649 void Error_Handler(void)
650 {

```

```
651 /* USER CODE BEGIN Error_Handler_Debug */
652 /* User can add his own implementation to report the HAL error return
   state */
653 __disable_irq();
654 while (1)
655 {
656 }
657 /* USER CODE END Error_Handler_Debug */
658 }

659
660 #ifdef USE_FULL_ASSERT
661 /**
662  * @brief Reports the name of the source file and the source line number
663  * where the assert_param error has occurred.
664  * @param file: pointer to the source file name
665  * @param line: assert_param error line source number
666  * @retval None
667 */
668 void assert_failed(uint8_t *file, uint32_t line)
669 {
670 /* USER CODE BEGIN 6 */
671 /* User can add his own implementation to report the file name and line
   number,
   ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
   line) */
672 /* USER CODE END 6 */
673 }
674
675 #endif /* USE_FULL_ASSERT */
```