UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA
CURSO DE ENGENHARIA ELÉTRICA

GEOVANI CARDOZO ALVES

PROJETO DE UNIDADE DE EFEITO SONORO PARA GUITARRAS

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2018

GEOVANI CARDOZO ALVES

PROJETO DE UNIDADE DE EFEITO SONORO PARA GUITARRAS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de Engenheiro Eletricista.

Orientador: Dr. Marcelo de Oliveira Rosa
UTFPR

CURITIBA
2018

Geovani Cardozo Alves

Projeto de Unidade de Efeito Sonoro para Guitarras

Este Trabalho de Conclusão de Curso de Graduação foi julgado e aprovado como requisito parcial para a obtenção do Título de Engenheiro, do curso de Engenharia Elétrica do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná (UTFPR).

Curitiba, de 14 junho de 2018.

_____
Prof. Antonio Carlos Pinho, Dr.
Coordenador de Curso
Engenharia Elétrica

_____
Profa. Annemarlen Gehrke Castagna, Ma.
Responsável pelos Trabalhos de Conclusão de Curso
de Engenharia Elétrica do DAELT

ORIENTAÇÃO

_____
Marcelo de Oliveira Rosa, Dr.
Universidade Tecnológica Federal do Paraná
Orientador

BANCA EXAMINADORA

_____
Marcelo de Oliveira Rosa, Dr.
Universidade Tecnológica Federal do Paraná

_____
Adriano Ruseler, Dr.
Universidade Tecnológica Federal do Paraná

_____
Antonio Carlos Pinho, Dr.
Universidade Tecnológica Federal do Paraná

A folha de aprovação assinada encontra-se na Coordenação do Curso de Engenharia Elétrica

Ao Curso de Engenharia Elétrica na Universidade Tecnológica Federal do Paraná e às pessoas com quem convivi ao longo desses anos. As experiências e oportunidades que pude compartilhar com amigos nesses espaços foram a melhor fase da minha formação acadêmica.

AGRADECIMENTOS

A esta universidade, seu corpo docente, direção e administração que oportunizaram meu crescimento profissional e acadêmico enquanto absorvia conhecimento para meu campo de estudo.

Ao meu orientador Marcelo de Oliveira Rosa, pelo suporte no pouco tempo que lhe coube, pelas suas correções e incentivos não somente neste trabalho mas, também, no artigo que foi desenvolvido a partir deste.

Aos meus pais, pelo amor, incentivo e apoio incondicional.

À minha namorada pelo apoio e suporte durante os momentos mais difíceis entre trabalho e faculdade, além de auxílio para correção nas questões de ortografia ligadas ao trabalho.

Aos meus colegas de faculdade, também pelo apoio, incentivo e, principalmente, por me distraírem no momento certo a fim de ter momentos de descanso durante essa jornada.

Por fim, a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

I know nothing, because I know too much, and understand not nearly enough and never will. (RICE, Anne).

RESUMO

ALVES, Geovani C. PROJETO DE UNIDADE DE EFEITO SONORO PARA GUITARRAS. 2018. 66 f. Trabalho de Conclusão de Curso – Curso de Engenharia Elétrica, Universidade Tecnológica Federal do Paraná. Curitiba, 2018.

Neste trabalho é apresentado uma unidade de efeito sonoro digital para guitarras a partir de um microcontrolador. São usados conversores de 16 bits, tanto analógico para digital quanto de digital para analógico, operando a 48 kHz onde transferem informações para o microcontrolador, no caso do conversor AD, e do microcontrolador no caso do DA, sendo isso possível através de uma interface serial (SPI). São abordadas as decisões do projeto para a conexão de todos os componentes, o projeto de filtros passa baixa para redução de *aliasing*, e recursos adicionais úteis para músicos. Por fim, são mostrados os resultados obtidos através do dispositivo e melhorias futuras são discutidas.

Palavras-chave: Unidade de efeito. Processamento de Sinais. Pedaleira.

ABSTRACT

ALVES, Geovani C. A Micro-Controlled Digital Effect Unit for Guitars. 2018. 66 f. Trabalho de Conclusão de Curso – Curso de Engenharia Elétrica, Universidade Tecnológica Federal do Paraná. Curitiba, 2018.

Here is presented a micro-controlled digital effect unit for guitars. Different from other undergraduate projects, it is used a high-quality 16-bit Analog-to-Digital (A/D) and Digital-to-Analog (D/A) converters operating at 48 kHz that respectively transfer data to and from a micro-controller through serial peripheral interfaces (SPIs). It is discussed the design decisions for interconnecting all these components, the project of anti-aliasing (low-pass) filters, and additional features useful for players. Finally, some results obtained from this device are shown, and discuss future improvements.

Keywords: Digital Effect Unit. Signal Processing. Stomp Box.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

A/D	Analógico/Digital
ADC	*Analog to Digital Converter*
CI	Circuito Integrado
D/A	Digital/Analógico
DC	*Direct Current*
DAC	*Digital to Analog Converter*
GND	*Ground*
IDE	*Integrated Development Enviroment*
LFO	*Low Frequency Oscillator*
LSB	*Least Significant Bit*
MOSI	*Master Out Slave In*
MISO	*Master In Slave Out*
MSB	*Most Significant Bit*
μC	Microcontrolador
PCB	*Printed Circuit Board*
PDS	Processamento Digital de Sinais
PTH	*Pin-Through Hole*
SAR	*Successive Approximation Register*
SMD	*Surface-Mount Devices*
SSI	*Synchronous Serial Interface*
TI	Texas Instruments

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

1.1 TEMA

Efeitos de áudio vêm sendo essenciais para a personalização da música desde o início do século XX. Nesta década instrumentos como o teremim e o órgão *Hammond* foram inventados e ambos já possuíam efeitos intrigantes: o primeiro é um instrumento tocado com suas mãos próximas a duas antenas separadas dando volume e frequência ao som emitido pelo mesmo, e o último, quando criado em 1934, já possuía alguns efeitos de áudio originados de forma mecânica, como por exemplo um alto-falante giratório no qual dava o efeito tremolo ao som. Com isso, guitarristas também buscaram efeitos para seus instrumentos, sendo tais efeitos obtidos mecanicamente inserindo furos em alto-falantes ou inserindo os mesmos dentro de grandes caixas d'água vazias (TARQUIN, 2015).

Como consequência, fabricantes de amplificadores de áudio, com o intuito de atingir esses efeitos sem a necessidade de um estúdio ou de equipamentos mecânicos, começaram a implementar sistemas analógicos de processamento de sinais em amplificadores valvulados, diminuindo a complexidade e o número de equipamentos necessários para atingir o som desejado e, posteriormente, dando origem às pedaleiras ou unidades de efeito. Ao fim dos anos 60, efeitos sonoros para guitarra tinham se tornado uma parte indispensável para a música pop (THOMPSON, 1997).

Um dos primeiros efeitos a se conseguir foi o *overdriving*, que se define como um efeito de saturação do amplificador em seus níveis de tensão, gerando a distorção do áudio de entrada. Além do *overdriving*, surgiram também outros efeitos como os de modulação, ambiência e tonalidade, entre outros. Os efeitos de modulação alteram não só a amplitude do sinal como também a sua frequência e fase. O *chorus* é um tipo de efeito de modulação que gera no sinal de entrada um atraso de 20 a 30 milissegundos, dando a impressão de dois instrumentistas estarem tocando o mesmo acorde com um pequeno atraso entre eles. Já os efeitos de ambiência simulam a reverberação e o eco, como se o instrumentista estivesse em ambientes acústicos que produzissem tais efeitos, como uma sala grande e vazia. Por fim, os efeitos de tonalidade alteram a equalização (graves, médios e agudos) do sinal de entrada, encontrados em pedaleiras do tipo Paramétrico e do tipo Gráfico. Foram citados aqui alguns tipos de unidades de efeito de som, no entanto, além de outros efeitos que alteram outras propriedades do áudio, há também a possibilidade de combinação de várias pedaleiras, gerando efeitos únicos e originais.

Na Figura 1 é mostrado um exemplo de pedaleira do tipo *overdrive*, possuindo 3 regulagens: *Level*, *Drive* e *Tone*. O *level* ajusta o volume do sinal de saída, o drive

ajusta o quanto de efeito será aplicado ao sinal, e o tone define o corte de algumas frequências do áudio. Além das regulagens, há também o botão que aciona a pedaleira (parte preta escrita "*Boss*") quando pisada pelo guitarrista, e os conectores do tipo P1 de entrada (*input*) e saída (*output*).

Figura 1 – Pedaleira *Overdrive*

Fonte: Tarquin (2015)

Cada pedaleira gerava apenas um tipo de efeito, existindo, ainda, a necessidade de utilização de várias pedaleiras em série para que efeitos múltiplos fossem permitidos de maneira simultânea.

A partir dos anos 90, tornou-se comum o uso de unidades de efeito digital, ou pedaleiras digitais, que, além de serem mais compactas e portáteis, também conseguiam produzir diferentes efeitos simultaneamente. Esse tipo de pedaleira não é produzido unicamente com sistema analógico, pois também faz o uso de microcontrolador na parte de processamento digital do sinal de entrada, que, muitas vezes, também atua no pré-condicionamento de sinal e sua amostragem.

O uso de microcontroladores revolucionou o campo da produção musical, ao passo que esse tipo de tecnologia permitiu aos guitarristas maior flexibilidade e possibilidades de efeitos de áudio, gerando novos estilos musicais com efeitos sonoros que antes não eram possíveis.

Com esse advento, não era mais necessário o desenvolvimento de placas exclusivamente dedicadas à aplicação dos projetos, o que implicava muitas vezes em equipamentos maiores e de difícil manutenção.

Com o passar dos anos e com os avanços da tecnologia, os microcontroladores tornaram-se uma das melhores relações custo/benefício (MARTINS, 2005), sendo um dos dispositivos mais empregados para soluções na indústria musical atualmente, pois além da sua instalação em um sistema ser mais simples do que o desenvolvimento de

um novo circuito, a modificação de qualquer projeto também se torna acessível, uma vez que só é preciso reprogramar o microcontrolador, não havendo, assim, a necessidade de alterar significativamente o *hardware* adjacente.

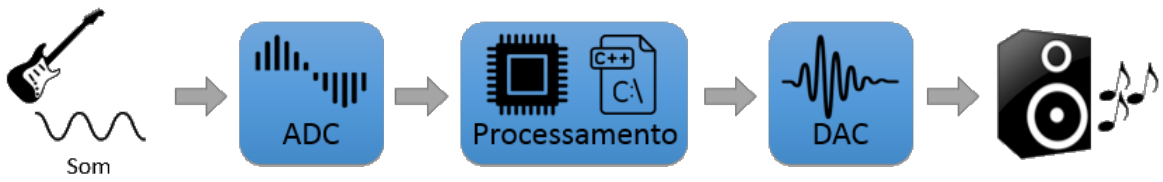
> Um microcontrolador pode efetuar várias funções que necessitariam de um grande número de outros componentes. Assim, aprender a programar microcontroladores significa aprender a resumir circuitos em um único componente. (MARTINS, 2005, p. 14).

A principal vantagem da pedaleira digital em relação à analógica é sua versatilidade quanto ao uso e deslocamento da mesma. Porém, a qualidade do efeito é, ainda hoje, questionada entre os músicos, podendo ser chamado de "som robotizado" devido ao fato do resultado da amostragem de sinal não ser idêntico ao sinal analógico de entrada.

1.1.1 Delimitação do Tema

Esse estudo tem por objetivo a construção de um protótipo de unidade de efeito para guitarras a partir de um microcontrolador da Texas Instruments em conjunto com conversores de sinais. Este protótipo é uma unidade pré-programada com a finalidade de distorcer o som em várias intensidades e timbres. Esta unidade é acionada em tempo real pelo músico, tendo a mesma que possuir uma resposta imediata à entrada de sinal. Terá uma entrada física suportando pino P1, e saída física do mesmo gênero. A Figura 2 mostra o diagrama de blocos da solução.

Figura 2 – Diagrama de Blocos



Fonte: O Autor

Também faz parte do escopo do trabalho o desenvolvimento de código para o microcontrolador a fim de se conseguir efeitos para a guitarra, tais como o *overdrive* (efeito de distorção), *delay* e *looper* (efeitos de tempo) e *tremolo* (efeito de modulação).

Apesar do condicionamento do sinal de entrada não ser o objetivo principal deste trabalho, foi necessário a implementação de alguns filtros analógicos, os quais têm por finalidade inibir ruídos indesejáveis provenientes de fontes externas, condicionar o sinal adequadamente para processamento pelo microcontrolador e, por fim, prepará-los para o estágio de amostragem e reprodução.

1.2 PROBLEMAS E PREMISSAS

A premissa deste projeto é realizar a montagem e operação de um equipamento de montagem simples com um conversores de sinais robustos.

Foi avaliado o uso do conversor analógico para digital (A/D) do microcontrolador TIVA™ ou a necessidade de implementar um conversor A/D externo, o que também implicou na montagem de equipamentos que não estariam disponíveis em um primeiro momento, como circuitos integrados (CIs), *sockets* e periféricos da Texas Instruments (fabricante do microcontrolador).

Independente dos conversores utilizados, deve atentar-se à frequência de amostragem do conversor, a fim de minimizar a perda de informação do sinal por *aliasing*. Considerou-se, também, que algumas partes desse protótipo (como por exemplo o pré-tratamento do sinal) foram montadas primeiramente em uma *protoboard*, ou matriz de pontos, e, portanto, pôde ocorrer mau contato entre os elementos do circuito, além de interferências eletromagnéticas provenientes de fontes externas que fossem processadas posteriormente pelo equipamento. Sendo assim, se fez necessário o uso de alguns filtros analógicos para o condicionamento do sinal de entrada.

Também foi avaliado a mecânica do botão de acionamento da pedaleira, uma vez que o mesmo será acionado pelo pé do guitarrista e terá que aguentar certa pressão em seus contatos, possuindo dessa forma, uma boa resistência mecânica.

Por fim, a partir da necessidade de um sistema com funcionamento em tempo real, a velocidade com que o processador executa o efeito é fundamental para o correto funcionamento do sistema. Não é desejado ter uma defasagem perceptível aos ouvidos ao utilizar o equipamento, ou seja, o músico quer escutar o que está tocando naquele exato instante de tempo. Outro ponto importante foi a sincronização da informação de entrada com a de saída, reduzindo, assim, o efeito *jitter* sobre o sinal, efeito que causa perda de informações sobre o sinal amostrado.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

A construção de um protótipo de uma unidade digital de efeito sonoros programável para guitarra, conhecido popularmente como pedal ou pedaleira de guitarra, a partir de um microcontrolador e filtros analógicos para condicionamento de sinais.

1.3.2 Objetivos Específicos

- Revisar a literatura;
- Definir os componentes a serem usados para o desenvolvimento da pedaleira;

- Estudo de processamento de sinais digitais com a finalidade de se representar matematicamente os efeitos de distorção sonora que serão implementados;
- Condicionamento do sinal de entrada a partir de circuito externo;
- Implementação e teste do conversor analógico/digital;
- Implementação e teste dos filtros de efeitos sonoros;
- Implementação e teste do conversor digital/analógico para transmissão do sinal ao alto-falante.

1.4 JUSTIFICATIVA

Levando em consideração o crescente número de entusiastas pela música que buscam equipamentos financeiramente acessíveis, é interessante obter um equipamento no qual o próprio instrumentista poderia montar e selecionar os efeitos conforme desejado, obtendo uma pedaleira versátil para uso musical e, assim, destacar a viabilidade construtiva do equipamento a partir de componentes que já estão no mercado.

Além disso, uma das principais vantagens da pedaleira digital em relação à analógica é a capacidade de ter vários efeitos em um único equipamento, dispensando o uso de diversas pedaleiras que seriam necessárias para obter um efeito diferente, ou personalizado, dos comumente usados pelos músicos atualmente.

Finalmente, aplicar em um problema real o conhecimento adquirido das disciplinas de Sinais e Sistemas na aplicação de filtros analógicos e processamento de sinais; Eletrônica Analógica/Digital, levando em consideração a montagem do circuito e teste do mesmo, e por fim, Sistemas Microcontrolados escrevendo o código a ser usado no microcontrolador.

1.5 PROCEDIMENTOS METODOLÓGICOS

A elaboração deste trabalho será dividida em três etapas principais: a definição e delimitação do alvo de estudo, a pesquisa e determinação dos componentes a serem usados no projeto e teste dos mesmos, síntese do código do microcontrolador e teste prático da peladeira para a produção dos efeitos sonoros a partir do sinal de entrada.

A primeira etapa corresponde à proposta em questão, onde já está delimitado o tema do trabalho. Na segunda será apresentado os componentes escolhidos e a topologia dos filtros e circuitos implementados na solução completa. Por fim, apresentar os resultados obtidos através dos algoritmos desenvolvidos e avaliar o desempenho do sistema.

Será utilizado o microcontrolador TIVA^{TM} EK-TM4C1294XL para o desenvolvimento da solução aqui proposta (TEXAS INSTRUMENTS, 2014c).

Na terceira parte, será visualizada a forma de onda da saída do captor da guitarra, para dessa forma compreender seus parâmetros e posteriormente passar este sinal à entrada do conversor A/D, podendo assim ser realizado o teste prático do mesmo. Será ainda definido o método de distorção aplicado ao microcontrolador, a modelagem matemática de cada efeito sonoro a partir da teoria de PDS (Processamento Digital de Sinais), a escrita do programa com linguagem previamente definida e teste do algoritmo a partir do sinal do conversor A/D.

Por fim, na quarta seção, será implementado o conversor D/A, para a transformação do sinal discreto no tempo em um sinal contínuo no tempo (analógico), e teste de todos os componentes funcionando em conjunto.

Finalmente, após averiguado que a unidade de efeito sonoro está funcionando de acordo com o planejado, expressar os resultados de forma de onda obtidos na saída da pedaleira comparando-os com os da entrada da mesma, e verificar se está de acordo com as teorias apresentadas em sala de aula.

1.6 ESTRUTURA DO TRABALHO

O trabalho de conclusão de curso foi estruturado em quatro capítulos principais. O primeiro baseia-se nos motivos pelos quais este projeto está sendo executado e comenta os principais benefícios das pedaleiras digitais.

O segundo capítulo, apresenta um breve histórico do processamento digital de sinais e em quais equipamentos esse método é aplicado atualmente. Também são ressaltados os embasamentos teóricos referentes à aquisição e processamento de sinais que, posteriormente, serão utilizadas na prática e, também, a apresentação dos *pseudocodes* que serão implementados na unidade de efeito. Ainda nesse capítulo, foi apresentado como foi feita a escolha dos conversores de sinais e o porquê da escolha dos mesmos, além disso, também foi abordado o teste prático de cada conversor que participará da unidade de efeito.

O terceiro capítulo trata sobre o sinal de saída do instrumento para o qual esta unidade de efeito está sendo desenvolvida, a guitarra. O desenvolvimento da placa para a captação deste sinal, os filtros que nela foram implementados e os componentes eletrônicos integrantes da mesma.

O quarto capítulo versa sobre as formas de ondas resultantes da aplicação de cada efeito, conclusão do trabalho a partir da análise sobre o correto funcionamento do pedal elétrico e comentários finais sobre o projeto.

2 REVISÃO DE LITERATURA

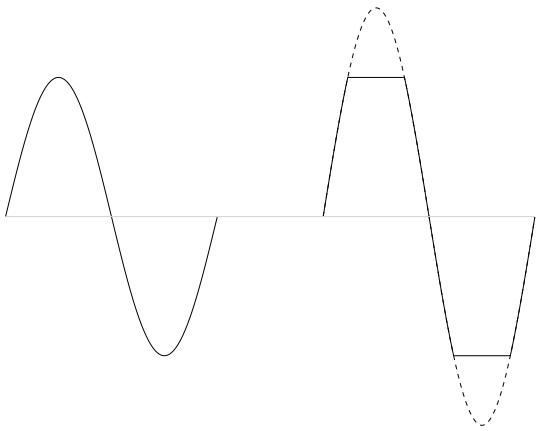
2.1 PROCESSAMENTO DE SINAIS DIGITAIS

O processamento de sinais se faz necessário em inúmeras aplicações utilizadas atualmente, desde áreas como a da medicina, militar, exploração do espaço, comunicação, assim como em equipamentos do nosso cotidiano: celulares, computadores, som de alta definição, etc.

Mas o que vem a ser Processamento de Sinais propriamente dito? De acordo com Oppenheim (1999), o processamento de sinais pode ser descrito como "a representação, transformação e manipulação dos sinais e as informações que eles contêm". Portanto, ele pode ser utilizado no caso de uma operação onde se queira, por exemplo, separar dois sinais que, de alguma forma, foram previamente combinados, ou aumentar/diminuir a intensidade de um sinal.

Tratando o som como uma função variável contínua no tempo, pode-se então a partir das teorias de processamento de sinais alterar suas características (altura, intensidade e timbre) de modo a obter um som diferente na saída de um sistema de acordo com o desejo do projetista. Como um exemplo, no caso das pedaleiras de distorção, a intenção é elevar a intensidade do som de modo que o pré-amplificador ficasse saturado, não conseguindo aumentar toda a intensidade do sinal, mas sim apenas parte do mesmo, fazendo com que os picos da onda sejam cortados (*clipping*), como exemplifica a Figura 3.

Figura 3 – Onda de Distorção



Fonte: O Autor

Para realizar este efeito, circuitos eletrônicos eram dispostos em uma configuração específica de componentes, pois nos primórdios da eletrônica o campo do microprocessamento ainda estava dando os primeiros passos e, por isso, não havia

memória e/ou processamento suficiente para fazer operações algébricas tão complexas para processamento de sinais. Logo, as soluções eram desenvolvidas para sinais contínuos no tempo através de circuitos analógicos, os quais muitas vezes ocupavam espaço considerável e estavam sujeitos a variações de temperatura, umidade, etc. O que resultava num circuito com imprecisão em seus resultados (PUHLMANN, 2014).

A partir do avanço da microeletrônica, em meados de 1971, processadores mais potentes e uma maior capacidade de armazenamento de dados foram desenvolvidos. Esse avanço em conjunto com o desenvolvimento de teorias para sinais discretos no tempo como a da transformada rápida de Fourier, eram as barreiras necessárias a serem quebradas para o início do processamento digital de sinais e, posteriormente, das primeiras pedaleiras a partir de circuitos digitais e microcontrolados. (OPPENHEIM, 1999).

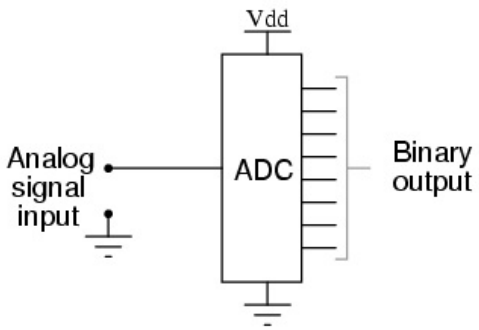
2.2 CONVERSORES DE SINAIS

Conversores Analógicos/Digitais, convertem um sinal analógico e o transformam em um sinal digital a uma determinada taxa de amostragem definida pelo projetista. Esta taxa é o principio fundamental da amostragem de sinais (SEDRA et al., 2007). Quanto maior a taxa de amostragem maior será o número de amostras obtidas do sinal, ou seja, mais similar ao original (analógico) será o resultado.

2.2.1 Conversor Analógico/Digital (A/D)

Antes do sinal ser efetivamente processado gerando o efeito desejado pela pedaleira, é necessário transformar esse sinal do mundo real (sinal contínuo no tempo) para o mundo digital (sinal discreto no tempo), logo se fez necessário um conversor A/D para realizar tal operação. A Figura 4 mostra o esquemático de um conversor A/D.

Figura 4 – Conversor A/D



Fonte: (KUPHALDT, 2006)

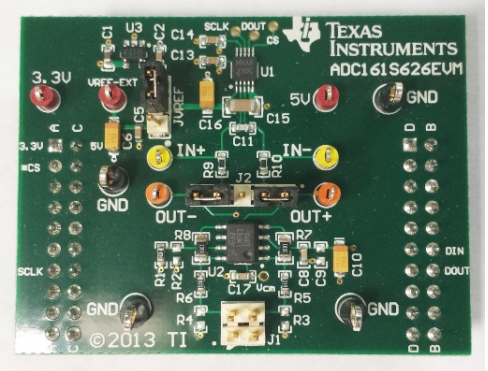
Posteriormente, este sinal é convertido para um um número binário de N bits proporcional ao valor da amostra de sinal em relação a uma tensão de referência. Este

número expressa a resolução do conversor, em outras palavras, quanto maior o número de bits do conversor, menor será o erro deste processo de quantização do sinal. Com a fórmula abaixo pode-se obter o valor do sinal a ser utilizado pelo microcontrolador após a conversão.

$$\frac{2^{\text{Resolução ADC}} - 1}{V_{Ref}} = \frac{Leitura\ ADC_{Decimal}}{V_{Medida}}$$

Para este projeto, foi arbitrada uma resolução de 16 bits para o sinal de entrada e saída da pedaleira. O µC em questão possui um ADC de apenas 12 bits, logo, a alternativa foi obter um periférico externo que substitua o conversor A/D atual. Foi escolhido um conversor de 16 bits da Texas Instruments já inserido no BoosterPack (circuito pronto para ser acoplado no microcontrolador). O conversor, de modelo ADC161S626EVM, está ilustrado na Figura 5.

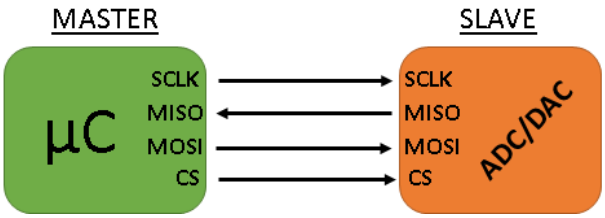
Figura 5 – ADC161S626EVM



Fonte: (TEXAS INSTRUMENTS, 2015a)

Esse conversor de topologia SAR (*Successive Approximation Register*) utiliza a interface de comunicação SSI (*Synchronous Serial Interface*) a fim de enviar dados para o microcontrolador da Texas. Essa interface requer o uso de no mínimo 3 pinos do microcontrolador sendo um o CS (*Chip Select*) o qual seleciona o periférico a ser usado e, para o conversor ADC em questão, quando este pino muda para o estado lógico baixo é dado o inicio da conversão de sinal (TEXAS INSTRUMENTS, 2008). Outro pino necessário é o SCL (*Serial Clock*) o qual fornece o clock para o periférico selecionado a fim de sincronizar a informação a ser enviada/recebida. Por fim, o SSTx (*Synchronous Serial Transmitter*) que é usado para transmitir os dados convertidos pelo ADC para o microcontrolador. A Figura 6 ilustra as conexões entre o microcontrolador e o ADC através da SSI.

Figura 6 – Diagrama de Blocos SSI



Fonte: O Autor

Através do pino MISO (*Master In Slave Out*), o microcontrolador manda informações para o periférico. Já com o pino MOSI (*Master Out Slave In*) ocorre o contrário, o microcontrolador recebe informações do periférico como no caso do ADC.

Para teste desse conversor foi utilizado um algoritmo relativamente simples, o mesmo fica lendo a entrada do ADC e, através da função printf da bilioteca *Standard Input Output* (stdio) do C, o resultado é mostrado para o usuário. A parte mais crítica deste processo é a configuração das portas do microcontrolador para que o mesmo se comunique corretamente com o ADC. Esta configuração será mostrada posteriormente na Seção 2.3.

Quanto ao resultado da conversão do ADC, esse é recebido pelo μC em notação de complemento de dois. A mesma que vem a ser uma notação binária criada a fim de acabar com a duplicidade de representação para o zero, além de facilitar a representação de números negativos, no qual o primeiro bit (MSB - *Most Significant Bit*) representa o sinal do número binário, 0 se for positivo e 1 para negativo.

$$0_{DEC} = 00000000_{BIN} = 11111111_{BIN}$$

Na notação de complemento de 2, se o número for positivo a conversão pode ser feita da forma direta (ou complemento de 1) para decimal, como é exemplificado com o número de 8 bits abaixo:

$$01101001_{BIN} = 105_{DEC}$$

Já para um número negativo deve-se fazer a conversão conforme a regra de complemento de 2: primeiramente, inverte-se o número binário (nega o vetor de bits):

$$not(01101001)_{BIN} = 10010110_{BIN}$$

A partir deste resultado soma-se 1:

$$10010111_{BIN} = -105_{DEC}$$

Logo, no caso do sinal recebido do ADC estar com o MSB setado, o mesmo deve ser convertido para decimal a partir da conversão por complemento de 2. Como

consequência se faz necessário a elaboração de um algoritmo de conversão para o caso em que o sinal de entrada seja deste tipo.

Algoritmo 1: Exemplo de Conversao Complemento de 2

```
Input: Numero binario a ser recebido do ADC pelo microcontrolador
Output: Numero convertido
while 1 do
    adcValue → Valor do ADC
    if adcValue & 0x8000 == 0x8000 then
        adcValue = not(adcValue) + 1;
    end
end
```

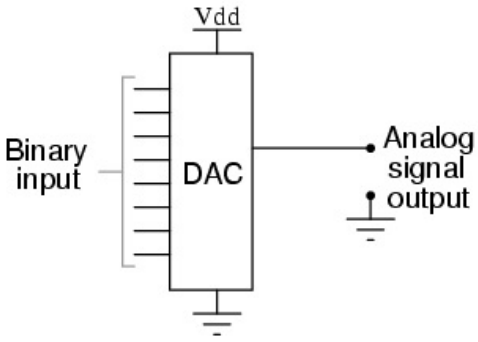
Para o ADC, a SSI foi configurada de modo a obter 16 bits a cada amostragem e a uma frequência de $5\,\text{MHz}$, portanto, quando a função de leitura do A/D for chamada a mesma irá obter 16 bits do conversor. Entretanto, na folha de dados desse componente está descrito que nos primeiros dois ciclos de *clock* da SSI o mesmo enviará bits nulos, logo, esses devem ser descartados. Isso faz com que seja necessário obter duas amostras da SSI, onde a primeira irá obter os 14 primeiros bits (do MSB para o LSB) e a segunda irá receber os últimos dois restantes. Dessa forma, se faz necessário agrupar essas duas amostras de modo que resulte nos 16 bits para formar o valor correto. Portanto, o total de bits a serem enviados pela SSI é 32, o que faz com o que o tempo total da conversão, desprezando o tempo do processamento do código pelo μC e levando em consideração a frequência da SSI, seja de $6{,}4\,\mu\text{s}$.

Após o sinal ser processado pelo microcontrolador, esse deve ser enviado ao alto-falante, porém o mesmo se encontra na forma digital e deve ser convertido para analógico novamente, este processo é feito pelo conversor de operação inversa A/D.

2.2.2 Conversor Digital/Analógico (D/A)

Este conversor é recíproco ao apresentado na seção anterior, isto é, realiza a operação contrária de converter o sinal digital em analógico. O processo de conversão é similar ao do A/D, não há taxa de amostragem propriamente dita, contudo, o sinal deve ser convertido na mesma velocidade que foi feito com o conversor D/A a fim de não haver perdas de dados ou a modificação do sinal de saída em relação ao original (*jitter*). O conversor basicamente recebe a entrada digital e a transforma numa saída analógica com a tensão proporcional à da fonte de referência em que o D/A está inserido. Na Figura 7 encontra-se o esquemático de um conversor D/A.

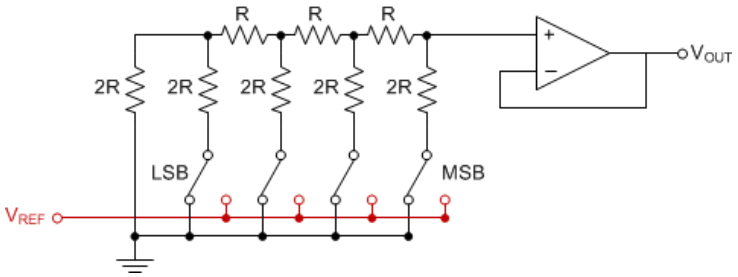
Figura 7 – Conversor D/A



Fonte: (KUPHALDT, 2006)

Na Figura 9 consta o conversor D/A utilizado nesse projeto, também da fabricante Texas Instruments modelo DAC161S055EVM, que possui uma topologia de conversão do tipo matriz de resistores, ilustrada na Figura 8. Nessa topologia cada bit configura um registrador (representado por uma chave *on/off* no esquemático) que permite ou não a passagem de corrente pelo resistor ligado à chave. Quando o registrador está configurado para permitir a passagem de corrente, há uma queda de tensão no resistor nele conectado. Essa diferença de potencial será observada pelo seguidor de tensão mais a frente e, com isso, dependendo da sua posição, do bit mais significativo (MSB) ao menos significativo (LSB), a tensão de saída será proporcional à tensão de referência do conversor e ao número de bits ativos (TEXAS INSTRUMENTS, 2010).

Figura 8 – Topologia Matriz de Resistores



Fonte: (TEXAS INSTRUMENTS, 2016)

Da mesma forma que o conversor anterior, ele possui 16 bits de resolução e se comunica através da SSI. Neste caso, porém, ao invés de transmitir sinal ao microcontrolador ele irá receber sinal para gerar a tensão de saída através do pino SSIRx (*Synchronous Serial Receiver*) no conversor, ou como mostra a Figura 6 pelo pino MOSI. O conversor DAC utilizado é ilustrado na Figura 9.

Figura 9 – DAC161S055EVM

Fonte: (TEXAS INSTRUMENTS, 2014a)

Neste *BoosterPack*, além do próprio conversor, consta também um regulador de tensão, o qual por padrão é usado como referência de tensão elétrica para o circuito do DAC. Pode-se utilizar uma tensão de referência externa efetuando a mudança do *jumper* na placa, porém esta tensão não deve exceder o limite de $6{,}3\,\mathrm{V}$, conforme especificado na folha de dados do componente a fim de não danificar as portas do CI do conversor.

Na configuração da SSI desse módulo, foi utilizado uma frequência de $20\,\mathrm{MHz}$ e 8 bits a cada amostragem, pois, ao contrário do ADC, esse conversor possui registradores internos que são configurados a partir de comandos enviados ao mesmo. Esses comandos possuem 8 bits cada e, além de configurar os registradores, também fazem a escrita na porta do D/A. Logo, o modo correto de enviar o valor para o DAC é escrevendo na interface um vetor que possui ao todo 24 bits, onde 8 são do comando de escrita e os 16 últimos representam o valor a ser convertido em forma de tensão. Desprezando o tempo de processamento de código, o tempo para a conversão do módulo D/A é de $1{,}2\,\mu\mathrm{s}$.

Ambos os conversores são alimentados pela placa do microcontrolador, com tensões de $3{,}3\,\mathrm{V}$ e $5\,\mathrm{V}$, que representam as tensões de nível lógico e de alimentação, respectivamente. Vale lembrar também que enquanto no A/D é preciso apenas mudar o estado lógico do pino CS para ele começar a enviar informações para o microcontrolador, no conversor digital/analógico há alguns registradores internos que precisam ser configurados de forma que o mesmo funcione corretamente, após isso deve-se enviar o comando de escrita para posteriormente transmitir o valor a ser convertido em analógico. A função no código que faz a configuração da memória interna do conversor é descrita no Apêndice C.

2.3 O MICROCONTROLADOR TIVA™

A fim de aprender a criar código para esse microcontrolador, assim que o kit foi adquirido, foi desenvolvido um algoritmo simples a fim de testar funções básicas como acender um LED, apertar um botão e piscar outro LED, etc.

Figura 10 – EK-TM4C1294XL

Fonte: (TEXAS INSTRUMENTS, 2015b)

É possível perceber que no kit há encaixes para duas "placas de expansão", chamadas pela TI de *Boosterpacks*. Esses dois encaixes serão usados para conectar os conversores de sinais que foram tratados na Seção 2.2.

O IDE utilizado neste trabalho foi o da própria fabricante do kit, Texas Instruments, uma vez que a mesma recomenda seu uso e torna mais fácil o desenvolvimento de projetos através do mesmo. O IDE é o *Code Composer Studio* fornecido gratuitamente quando usado em conjunto com produtos da TI.

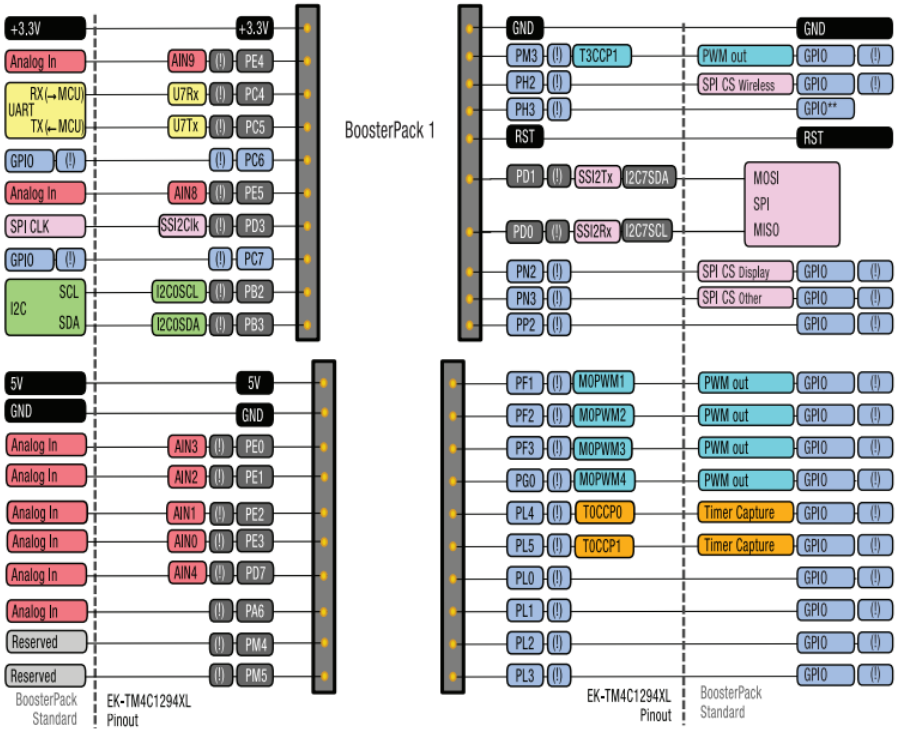
Para os primeiros testes com os conversores, foi necessária a correta configuração das portas do microcontrolador de modo a fazer a comunicação entre eles. Na Figura 11 é mostrada a disposição dos pinos do kit que encaixarão no Boosterpack 1 (placa que é conectada logo acima do microcontrolador fixado no kit de desenvolvimento).

No caso da configuração das portas do ADC, os pinos utilizados foram o PE4, PD0, PD1 e PD3. Foram configurados da seguinte forma:

- **PE4**: Esta porta é configurada como o CS da SSI, portanto ela é do tipo lógica e é deixada em nível alto até a parte em que o programa irá solicitar o início da conversão, então ela deve ser alterada para nível lógico baixo dando início à conversão;
- **PD0 e PD1**: Esses pinos são configurados com a função MOSI e MISO, respectivamente. Equivalendo a SSITx e SSIRx como mostrado na Figura 11;
- **PD3**: Esse pino é configurado como o *clock* da SSI a fim de sincronizar as informações enviadas/recebidas.

Apesar de o ADC não receber nenhuma informação do microcontrolador, o pino de envio de sinal deve ser configurado, pois o controlador só irá iniciar sinal de *clock* da SSI quando alguma informação é enviada através da interface - isso está especificado na seção 17 da folha de dados do TIVA™ (TEXAS INSTRUMENTS, 2014d), onde é explicado o funcionamento da SSI. Dessa forma, alguma informação, apesar de não seja usada, deve ser enviada a fim de iniciar o *clock* para sincronizar as informações enviadas pelo conversor.

Figura 11 – Pinagem Boosterpack 1



Fonte: (TEXAS INSTRUMENTS, 2014b)

No momento em que conectou-se as duas peças, ADC e TIVA^{TM}, notou-se que o pino de CS (*Chip Select*) na placa do ADC não corresponde à mesma posição do encaixe do kit onde deveria ser o CS. O CS da placa do ADC encaixa no pino PE4 do kit, logo é necessário escrever a sua função no código (mudança de estado lógico para iniciar/interromper conversão) manualmente para que a amostragem ocorra de maneira correta. Com exceção do PE4, os outros pinos são comandados a partir da biblioteca disponibilizada pela TI, chamada de *TIVAWare*, também fornecida gratuitamente pela fabricante.

Para o teste do conversor DAC, foi utilizada uma configuração de pinos semelhante à apresentada para o ADC. Porém, nessa etapa, ao invés do microcontrolador receber informações do conversor, ele envia o sinal digital a ser convertido para analógico. Como ocorreu no anterior, o pino CS da placa do conversor DAC também não coincidiu com o pino CS do μC, uma vez que o segundo encaixe para as placas expansoras possui uma pinagem quase igual à primeira. Portanto, também foi necessário alterar o estado do pino manualmente através do código com o propósito de dar início/término à conversão.

2.4 ALGORITMOS DE PROCESSAMENTO DE SINAIS

Como efeito mais simples e provavelmente o mais usado, a distorção, o efeito que realmente dá o som de guitarra ao instrumento, pois sem ele o som de guitarra

seria o de um violão elétrico. Esse efeito, já explicado anteriormente e também ilustrado pela Figura 3 no começo desse capítulo, ceifa os extremos inferior e superior da onda. A quantidade do pico da onda a ser cortado irá determinar o nível de distorção, quanto mais ceifada a onda, mais distorcido será o som. O Algoritmo 2 ilustra a ideia do código a ser escrito para esse efeito.

Algoritmo 2: Algoritmo de Distorção

```
Input: Sinal do ADC
Output: Sinal para o DAC
while 1 do
    Sinal = Sinal_ADC
    if Sinal >= limite_de_distorção then
        Sinal = limite_de_distorção
    end
    DAC_Saída = (Sinal)
end
```

Outro efeito que foi desenvolvido para este protótipo foi o *delay*. Como a tradução já sugere, esse efeito causa o atraso do sinal em uma certa quantidade de tempo estabelecida. Essa quantidade depende do quanto de memória está disponível para armazenar as amostras: se o tempo requerido for grande a ponto de não haver memória suficiente no microcontrolador, deve-se utilizar armazenamento externo.

No Algoritmo 3, nota-se um vetor de valores que irá armazenar as amostras. Para se obter o atraso, primeiramente, inicia-se o vetor de *delay* com zeros e, posteriormente, escreve o primeiro valor do vetor no DAC, e após, é feita a amostragem no ADC gravando esse valor também no vetor de *delay*. Quando terminar de percorrer todo o vetor, os valores previamente amostrados já terão preenchido todo o vetor, e portanto serão reproduzidos na sequencia que irá iniciar novamente o *array*. Portanto, para obter quanto tempo de atraso em segundos do sinal terá, basta dividir o número do tamanho do vetor pela frequência de amostragem: $N_{\text{Tamanho\_Vetor}}/f_S$.

Já através do efeito *Looper* o músico tem a opção de gravar parte do que está tocando e posteriormente escutá-la. Isso tem a finalidade do músico poder gravar acordes de base para solar[1] sem a necessidade de uma outra pessoa tocando mais um instrumento.

Nesse efeito, o algoritmo deve guardar as amostras de sinais até que, com o vetor todo preenchido, o sinal gravado comece a ser reproduzido. Como no *delay* este efeito também depende da memória do microcontrolador para guardar as amostras, portanto, provavelmente, o mesmo poderá gravar apenas poucos segundos de áudio.

O *Tremolo*, muitas vezes confundido com o *vibrato*, de acordo com Ratton (2004) é um efeito que realiza a variação periódica da amplitude de um sinal (a oscilação

[1] Parte da música onde o instrumentista se destaca com uma passagem melódica.

Algoritmo 3: Algoritmo do Efeito *Delay*

```
Input: Sinal do ADC
Output: Sinal para o DAC
for i = 0 até i = Tamanho_Vetor do
    Sinal_Atrasado[i] = 0
end
i = 0
while 1 do
    DAC_Saída = Sinal_Atrasado[i]
    Sinal_Atrasado[i] = ADC_Entrada
    i = i++
    i = i % Tamanho_Vetor
end
```

Algoritmo 4: Algoritmo do Efeito *Loop*

```
Input: Sinal do ADC
Output: Sinal para o DAC
while Gravando do
    Sinal[ i ] = Sinal_ADC
    i++
end
i_max = max( i )
while Reproduzindo do
    for i=0 ; i = i_max ; i++ do
        Sinal_DAC = Sinal [ i ]
    end
end
```

da amplitude em geral é menos de 20 Hz), dando a impressão que alguém está variando a intensidade do som no alto-falante, aumentando-a e diminuindo sucessivamente.

O jeito mais comum de se implementar tal efeito em circuitos analógicos, é a utilização de um LFO (*Low Frequency Oscillator* — Oscilador de Baixa Frequência). É esse oscilador que irá variar a amplitude do som, portanto ele modula o som de entrada e passa este sinal à saída da unidade de efeito.

No microcontrolador, é necessário criar um sinal que varie com o tempo para que o mesmo, multiplicado ao sinal de entrada, molde a amplitude do sinal recebido do ADC e, posteriormente, envie esse para o DAC.

Algoritmo 5: Algoritmo do Efeito *Tremolo*

```
Input: Sinal do ADC
Output: Sinal para o DAC
while 1 do
    Sinal_Variável = Sv
    DAC_Saída = (Sv * (Sinal_ADC))
end
```

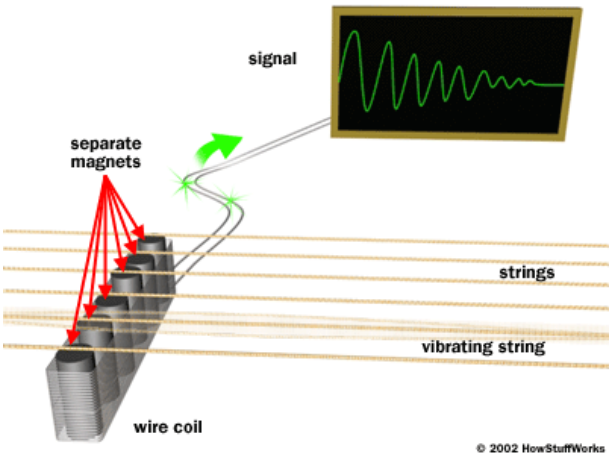
3 DESENVOLVIMENTO

3.1 O SINAL DA GUITARRA

O propósito dessa unidade de efeito sonoro é receber os sinais provindos da guitarra, alterá-los para o efeito desejado e reproduzi-los no alto-falante. Portanto é necessário que o sinal do instrumento seja compatível com as configurações do ADC, pois se a amplitude do sinal for muito pequena, sua manipulação será dificultada, não só pelas questões de ruídos mas também pela pequena faixa de números inteiros que ele iria produzir após passar pelo conversor A/D. Já no caso do sinal ser maior do que o suportado pelo ADC, o que é pouco provável, seria necessário que reduzir o seu nível de tensão para a posterior conversão do sinal.

Feita uma breve pesquisa sobre o sistema de captação de som de guitarras, percebe-se que o sinal de saída é pequeno quando comparado com os valores de tensão de entrada/saída dos conversores. Um esquemático do sistema de captação é mostrado na Figura 12.

Figura 12 – Esquemático do Sistema de Captação da Guitarra



Fonte: (BRAIN, 2002)

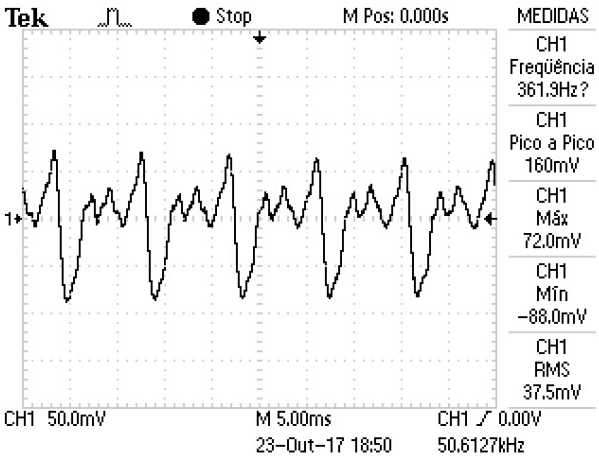
O sistema é embasado nas leis do eletromagnetismo. A ponte de captação é um ímã permanente que atrai levemente as cordas de aço que, devido ao campo magnético do ímã, geram por consequência seu próprio campo magnético. Ao redor dos ímãs da ponte há várias espiras de cobre formando uma bobina, que tem a terminação de seus fios em um resistor interno à guitarra. Esse, por sua vez, é conectado paralelamente ao conector de saída de áudio. Quando uma corda é tocada, sua vibração faz com que o campo magnético ao seu redor oscile na mesma frequência na qual é afinada. Pela lei de Faraday (BASTOS, 2004), uma bobina inserida em um campo magnético variável apresentará uma corrente nela induzida, essa corrente passando por um resistor irá

produzir uma tensão proporcional à intensidade da mesma, conforme a lei de Ohm (BRAIN, 2002).

Nota-se, portanto, que o valor da tensão de saída é proporcional à força de atração dos ímãs nas cordas. Essa tensão tem sua faixa de tensão começando em $100\,mV_{PP}$ e pode chegar a $1\,V_{PP}$ em captadores com ímãs de campo magnético mais intenso, como os de neodímio.

Conectando a saída da guitarra diretamente no osciloscópio obteve-se uma amostra do sinal da mesma e com isso se identificou o nível de tensão, conforme mostra a Figura 13.

Figura 13 – Tensão de Saída da Guitarra



Fonte: O Autor

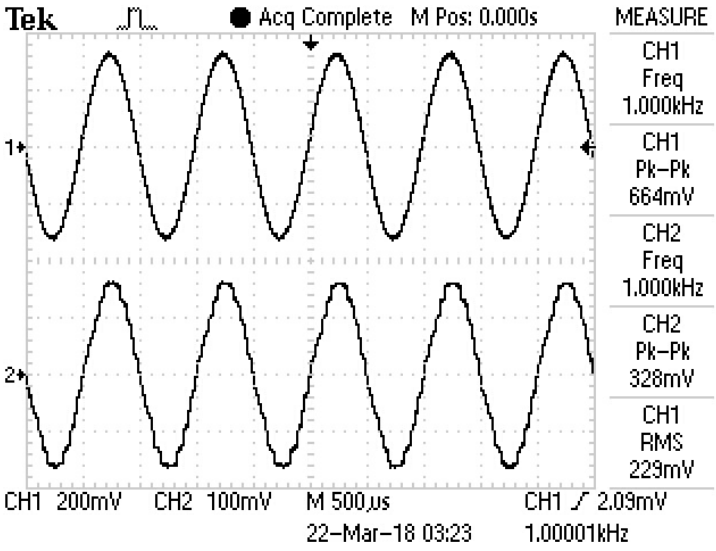
O sinal dessa guitarra não é maior que $200\,mV_{PP}$, como já era esperado, um nível de tensão muito pequeno para a entrada do ADC. Torna-se necessário o uso de um amplificador a fim de aumentar esse nível para pelo menos $1\,V_{PP}$, ampliando a faixa de tensão para se trabalhar com o ADC para a manipulação do áudio, e ao mesmo tendo o cuidado para não prejudicar as entradas do conversor devido à sobretensão. Esse amplificador será posicionado na primeira etapa da placa de captura de sinais desenvolvida e será explicado posteriormente na Seção 3.4.

3.2 SINAL DE SAÍDA DO DAC

Após o funcionamento dos conversores, um código simples para enviar o sinal de entrada diretamente para a saída – portanto sem haver qualquer alteração do sinal – foi implementado a fim de monitorar através do osciloscópio o comportamento dos conversores e do μC diante da alteração da frequência, essa que por sua vez deve ter uma faixa de $20\,Hz$ a $20\,kHz$ (faixa de frequência audível para o ser humano). Em baixas frequências obteve-se um bom resultado da forma de onda de saída, isso pode ser averiguado pela figura Figura 14.

Antes de prosseguir, a título de informação, nessa e nas próximas sessões estarão ilustradas a captura de imagem do osciloscópio, a fim de evidenciar a diferença entre os sinais de entrada e saída do ADC e DAC, respectivamente. A fim de não haver confusões, o seguinte padrão será estabelecido quando houver dois sinais na imagem: o sinal de entrada do ADC será o CH1 e o de saída do DAC será o CH2.

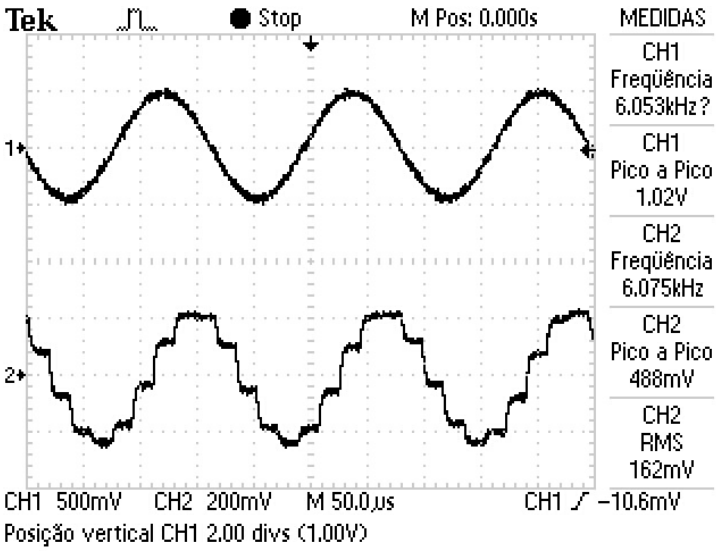
Figura 14 – *Aliasing* na Saída do DAC



Fonte: O Autor

Para 1 kHz, a onda de saída é praticamente igual à onda de entrada. No entanto, a partir de cerca de 6 kHz, foi identificado o efeito de *sample-and-hold* na onda de saída do conversor D/A, conforme mostrado na figura Figura 15.

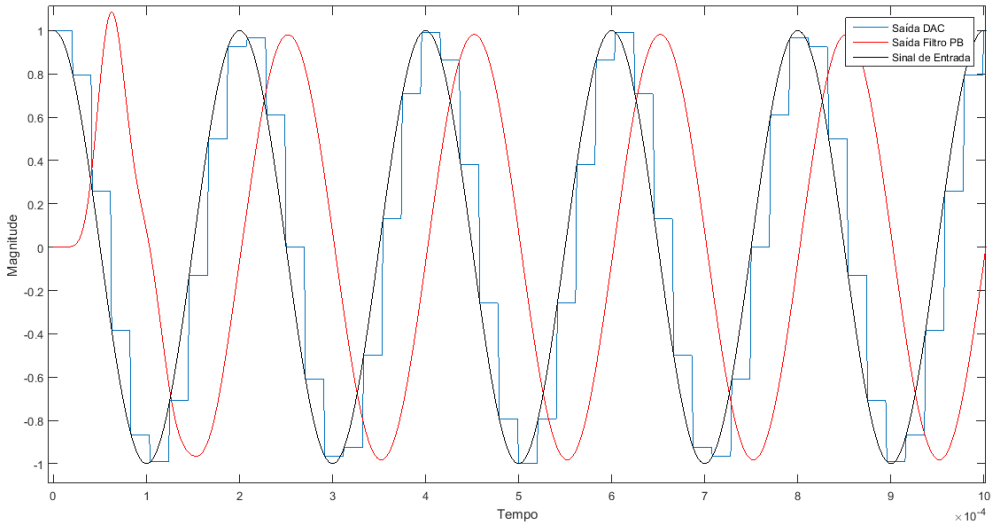
Figura 15 – *Aliasing* na Saída do DAC



Fonte: O Autor

Em um primeiro momento pensou-se que o problema era com a configuração dos conversores, e a solução para esse problema foi assumida por bastante tempo até que uma simulação no MATLAB revelou que a configuração dos conversores estava certa e, que na verdade, o problema era outro. Na simulação, uma onda de entrada senoidal foi gerada na entrada no conversor A/D e estimada a saída do conversor D/A com os *steps* resultante do processo de amostragem. Averiguou-se que, quando se utilizava um filtro passa-baixas sobre a onda de saída do D/A (função *butter* no MATLAB), os degraus desapareciam dessa, conforme ilustrado na Figura 16. Com isso, ficou evidente a necessidade do uso de um filtro adicional na saída do D/A do tipo passa-baixas, uma vez que no circuito do *BoosterPack* do D/A já possuía um passa-baixas de alta frequência de corte. O dimensionamento e a topologia escolhida dos filtros serão tratadas na Subseção 3.4.2.

Figura 16 – Simulação MATLAB



Fonte: O Autor

A fim de diminuir possíveis ruídos no sinal de entrada, pensou-se também em adicionar o mesmo filtro usado no D/A para a entrada de sinal no A/D. Portanto, foi necessário dimensionar um circuito contendo o amplificador de áudio para o sinal da guitarra, e ambos os filtros de entrada e saída.

3.3 O CÓDIGO PARA A UNIDADE DE EFEITO

Todo o código desenvolvido para o μC foi estruturado em pequenas funções onde cada uma possui sua finalidade específica, como escrever certo número na porta de saída do D/A ou configurar a inicialização da SSI a fim do microcontrolador se comunicar com os periféricos. Cada função possui um comentário em cima da mesma o qual identifica sua finalidade. O código na íntegra está descrito no Apêndice D.

Além disso, é interessante ressaltar que, como foi especificado no começo do projeto, a taxa de amostragem deveria ser de $48\,\text{kHz}$. Para atingir essa sincronia entre os dois conversores, foi utilizada as funções de interrupção por tempo do controlador. Essa interrupção foi configurada para ser periódica e para disparar a cada $1 \div 48\,\text{kHz}$. Com isso garante-se a sincronia dos conversores e evita a perda de dados (*jitter*).Por fim, alguns LEDs foram configurados para indicar qual efeito estava em operação.

3.4 PLACA DE AMPLIFICAÇÃO E FILTRAGEM DE SINAL

Com todas as soluções de contorno propostas na seção anterior, tornou-se indispensável o projeto uma placa de circuito impresso para alocar todos esses componentes e conectá-la diretamente na placa de desenvolvimento da TI (*LaunchPad*) ilustrada anteriormente na Figura 10.

O objetivo foi que o dispositivo funcionasse com o nível de tensão que a *LaunchPad* fornece, isto é, $5\,\text{V}$. Portanto os amplificadores operacionais escolhidos deveriam ser capazes de trabalhar com esse nível de tensão e conseguir amplificar o sinal de entrada até o nível máximo de alimentação, característica *rail-to-rail*. Além disso, uma vez que a porta USB (*Universal Serial Bus*) só consegue fornecer no máximo $500\,\text{mA}$ de corrente, o amplificador deveria drenar pouca corrente e portanto ser de baixa potência. Por fim, uma outra restrição foi o encapsulamento - que devia ser do tipo DIP (*Dual In-line Package*) - para que fosse possível testá-lo na *protoboard*. Dessa maneira, foi escolhido o amplificador da TI OPA344 (TEXAS INSTRUMENTS, 2000).

A partir da definição desse amp-op pode-se desenvolver os dispositivos necessários para a aquisição e tratamento de sinal: amplificação e filtro ativo.

3.4.1 Amplificador de Áudio

Como já explicado anteriormente, o sinal da guitarra tem uma amplitude muito baixa em comparação com a faixa de tensão que o ADC opera. Portanto um amplificador foi projetado para elevar essa tensão. Contudo, foi necessário atentar-se à configuração do amp-op, uma vez que não há alimentação negativa e, logo, a parte negativa do sinal não seria reproduzida na saída desse amplificador. Foi decidido usar uma configuração na qual o amplificador polariza a tensão de entrada em $V_S/2$, ou seja, metade da tensão de alimentação. Dessa forma, o sinal oscilaria em torno de $2{,}75\,\text{V}$ e, com a configuração escolhida, somente a parte AC do sinal seria amplificada. A configuração escolhida é ilustrada na Figura 17.

Os resistores R_1 e R_2 fazem a polarização do sinal em $V/2$. Já o capacitor de entrada de sinal, C_2, realiza o acoplamento entre o sinal de entrada e a tensão presente na porta não-inversora do amp-op bloqueando, dessa forma, qualquer tensão DC ali presente. A princípio, pensou-se em aumentar os valores dos resistores de polarização

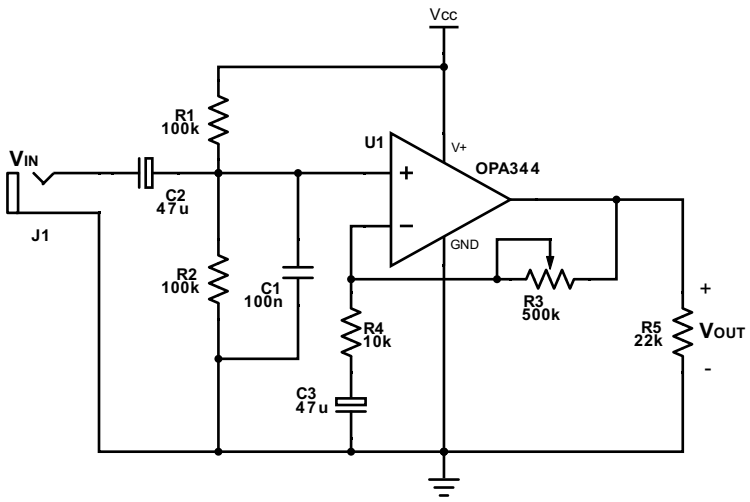
para que a corrente drenada pelos mesmos fosse ainda mais baixa, consequentemente diminuindo a potência neles dissipada. Porém, quando essa possibilidade foi testada com resistores de $1\,\mathrm{M}\Omega$, o amp-op não funcionava. O não funcionamento foi devido ao amplificador sair de sua operação linear, uma vez que a corrente de polarização do terminal não-inversor diminuiu a um nível que o deixava inoperante. Na teoria, considera-se amplificadores com corrente nula de entrada, tanto para o terminal não-inversor, quanto para o terminal inversor. Na prática, mesmo muito pequena (na ordem de nano a pico-amperes), essa corrente dos terminais de entrada do amplificador deve ser considerada.

O objetivo do capacitor presente entre o resistor de ganho (R_4) e o GND é fazer com que o amp-op não amplifique a diferença de tensão DC que estaria presente se o R_4 fosse diretamente ligado ao terra. Logo, o ganho desse amplificador em malha fechada fica determinado pela Equação (1).

$$G = 1 + \frac{R_3}{R_4} \tag{1}$$

Na prática, como os sinais de áudio da saída de guitarra variam de modelo para modelo, foi utilizado um potenciômetro de alta resistência (ex. $500\,\mathrm{k}\Omega$) para que o usuário ajuste o ganho necessário para seu instrumento.

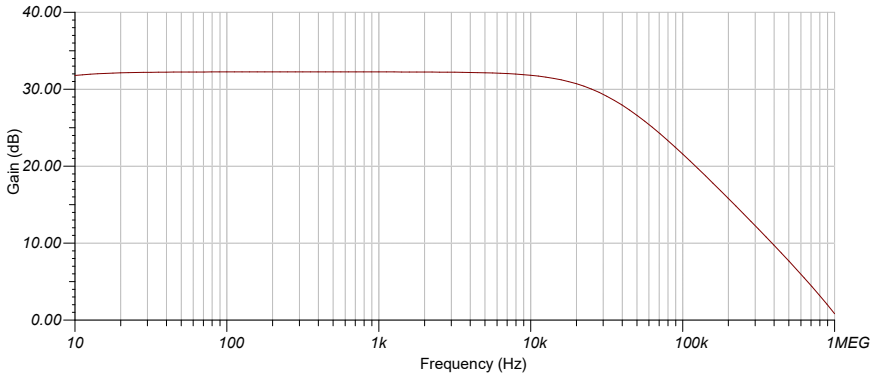
Figura 17 – Configuração do Amplificador Operacional



Fonte: O Autor

Além dos cuidados de tensão e corrente, também fez-se necessário a análise do ganho do amplificador de acordo com a frequência de entrada, uma vez que essa grandeza, por se tratar de um sinal sonoro, não é constante. A fim de simplificar a análise e não depender de deduções analíticas, foi feita uma simulação com o software Tina (TEXAS INSTRUMENTS, 2007) da própria TI a fim de avaliar o OPA344 na simulação considerando seus parâmetros internos. O resultado é mostrado na figura Figura 18.

Figura 18 – Resposta à Variação de Frequência do OPA344



Fonte: O Autor

É visto que, na faixa de frequência de operação do amp-op para o projeto (20 Hz a 20 kHz), o gráfico mantém-se praticamente constante, algo que é desejável uma vez que não é interessante ter variações de ganho para diferentes frequências se tratando de um dispositivo que tem como entrada sinais de frequência variável. Do contrário, haveria notas do instrumento que possuiriam mais intensidade sonora que as outras. Observando o gráfico não é possível dizer ao certo o ganho do amplificador, porém o mesmo é obtido pela expressão do cálculo do decibel mostrado na Equação (2).

$$G = 20 \times \log_{10}\left(\frac{V_O}{V_I}\right) [dB] \tag{2}$$

O ganho V_O/V_I do amp-op na configuração usada no projeto, mostrada na Figura 17, é dado pela Equação (1), portanto, substitui-se a mesma na Equação (2). O valor encontrado está de acordo com o gráfico da Figura 18. Para a simulação foi utilizado um R_f = 40 kΩ. Logo, tem-se:

$$G = 20 \times \log_{10}\left(1 + \frac{R_3}{R_4}\right) = 20 \times \log_{10}\left(1 + \frac{40 \times 10^3}{1 \times 10^3}\right) \cong 32\ dB \tag{3}$$

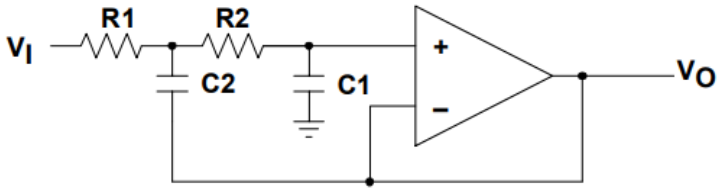
3.4.2 Filtros de Sinal

3.4.2.1 Filtro Passa-Baixas

Para ambas a entrada e saída de sinal do μC é necessário filtros passa-baixas. A começar pela entrada, onde o sinal provém da guitarra e já é amplificado pelo circuito anterior ao filtro, este sinal pode apresentar muitos ruídos devido ao fato de que o mesmo provém de cabos expostos a um meio eletricamente poluído. Já na saída do DAC, o objetivo é eliminar efeito *sample-and-hold* presente no sinal, como mostrado na Seção 3.2.

Para os dois casos, fez-se o uso de um filtro passa-baixas ativo, para facilitar acoplamento com outros circuitos, diferente de filtros passivos. Implementando uma topologia conhecida no meio acadêmico, optou-se por um filtro Sallen-Key o qual é mostrado na Figura 19.

Figura 19 – Filtro Sallen-Key



Fonte: (TEXAS INSTRUMENTS, 1999)

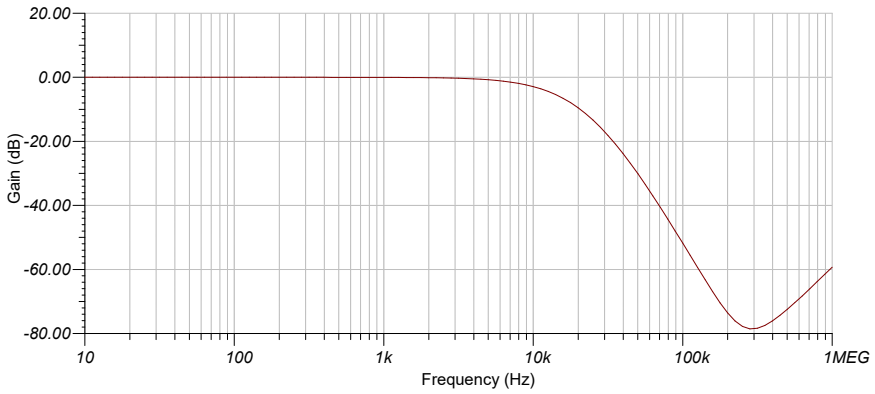
O principio básico de funcionamento se dá pelos capacitores C1 e C2: quando o sinal for de baixa frequência, a reatância desses capacitores será maior e, portanto, a maior parte da tensão da onda de entrada irá seguir para V_{OUT}. No caso onde a frequência da onda de entrada seja alta, esse sinal será drenado pelos capacitores até o GND.

Para fazer com que a inclinação da reta do ganho na frequência de corte seja no mínimo $-40\,\mathrm{dB/dec}$ (objetivando ter um filtro passa-baixa mais próximo do ideal), adotou-se um filtro de quarta ordem. Uma vez que o filtro Sallen-Key é de segunda ordem, projetou-se uma topologia em cascata de dois desses filtros atendendo o critério estabelecido. Para esse filtro também foi utilizado o CI do estágio de amplificação, OPA344.

A frequência de corte foi escolhida respeitando o teorema de Nyquist (OPPENHEIM, 1999). Portanto, com uma frequência de amostragem de $48\,\mathrm{kHz}$ os valores dos componentes adotados para os filtros passa-baixa foram selecionados de forma que, combinados a com a topologia usada, obtenham uma frequência de corte do filtro de $24\,\mathrm{kHz}$. Os valores dos componentes são mostrados no circuito do Apêndice A e foram dimensionados com auxílio do artigo da Texas Instruments (1999) e do site de desenvolvimento de filtros

Simulando o circuito com o mesmo *software* utilizado anteriormente é possível obter a curva dos filtros PB utilizados no projeto. O resultado, mostrado na Figura 20, é a combinação de dois filtros em cascata, e, como já era esperado, a partir da metade da frequência de amostragem ($f_C = f_S/2 = 24\,\mathrm{kHz}$) o filtro começa a ter um ganho negativo, ou seja, atenua o sinal de entrada.

Figura 20 – Resposta à Variação de Frequência do Filtro Sallen-Key

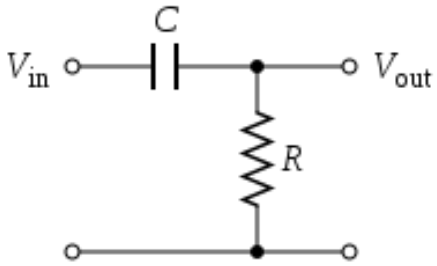


Fonte: O Autor

3.4.2.2 Filtro Passa-Altas

Em todos os estágios do circuito final, desde a entrada do amplificador até a saída no segundo filtro passa-baixas, o sinal possui um valor DC, requisito de projeto necessário para possibilitar o uso de amplificadores sem a necessidade de uma fonte de tensão negativa. Contudo, para reproduzi-lo num alto-falante ou conectá-lo a um amplificador de som, esse sinal não pode apresentar nenhuma componente contínua em sua composição. Para eliminá-la sem a utilização de nenhum filtro sofisticado como o passa-baixas, o qual tinha uma importância maior na filtragem de sinais, fez-se o uso da topologia de filtro passa-altas RC, ilustrado na Figura 21.

Figura 21 – Topologia de Filtro Passivo Passa-Altas



Fonte: O Autor

O funcionamento desse filtro é bem mais simples do que o apresentado anteriormente, para qualquer componente DC do sinal o capacitor série irá atuar como um circuito aberto, porém componentes alternadas, a partir da frequência de corte, serão observadas na saída desse filtro.

A frequência de corte para essa etapa deve ser a menor possível, uma vez que esse filtro só irá eliminar a componente de corrente contínua do circuito, ou seja, a componente de frequência zero. Logo, todas as outras frequência a partir da zero

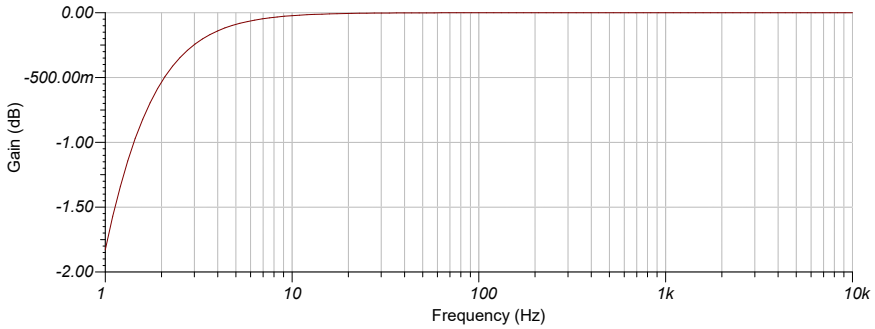
devem ser constatadas na saída do filtro. A partir da Equação (4) pode-se determinar a frequência de corte do filtro.

$$f_C = \frac{1}{2\pi RC} \tag{4}$$

O procedimento que geralmente é feito para escolher os valores de R e C, é determinar um valor comercial para um dos componentes e, a partir disso, já possuindo f_C e um dos elementos passivos previamente escolhido, calcula-se o outro. Como o valor provavelmente será fracionado devido às divisões, escolhe-se o valor comercial mais próximo e com ele recalcula-se a frequência de corte agora usando os valores comerciais determinados.

Para o projeto em questão foi determinado um capacitor de $10\,\mu F$ e um resistor de $22\,k\Omega$, resultando em uma frequência de corte igual a $0{,}72\,Hz$. Portanto, esse filtro praticamente deixará passar por ele todas as frequências a partir de aproximadamente $1\,Hz$ e atenuando as frequências abaixo desse valor, especialmente a componente CC ($f_C = 0\,Hz$). Simulando esse circuito, a resposta à frequência obtida é observada através da Figura 22.

Figura 22 – Resposta à Variação de Frequência do Passa-Altas RC



Fonte: O Autor

Como já era esperado, o filtro atenua todas as frequências muito próximas a $0\,Hz$ e tem ganho nulo para todas as outras, dessa forma, deixando passar apenas a parte oscilante do sinal que, no caso desse projeto, se trata de um sinal de áudio sem *offset*.

3.4.3 O *Hardware*

Todos os componentes utilizados no projeto (inclusive o amp-op já citado OPA344) possuem encapsulamento do tipo PTH (*Pin-Through Hole*). Tal encapsulamento é o mais comum de ser achado nas lojas de eletrônica de varejo e mais simples de serem soldado, além de conseguir encaixá-lo na *protoboard* sem maiores dificuldades. Já componentes do tipo SMD (*Surface-Mount Devices*) necessitam de

uma maior experiência e de equipamentos mais sofisticados para serem adicionados à placa, porém são mais fáceis de serem encontrados em lojas virtuais dos fabricantes e distribuidores. Isso fez o que a busca por CIs específicos para áudio fosse um tanto trabalhosa, uma vez que a maior parte desses componentes em sites de fabricantes são do tipo SMD.

Antes da confecção da PCB (*Printed Circuit Board* – Placa de circuito impresso), foram testados o comportamento dos amplificadores operacionais e os componentes adjacentes a eles na *protoboard*, uma vez que o resultado era o esperado, fez-se o uso do *software* Proteus para o projeto da placa, uma vez que o mesmo era de mais fácil acesso e uso que outros programas do mesmo gênero.

Tendo em mente os dispositivos necessários a serem dispostos na PCB, mencionados nas seções anteriores, foi desenhado o esquemático ilustrado no Apêndice A. Com o esquemático feito, era necessário, agora, traçar as rotas das trilhas no projeto da PCB que posteriormente seriam reproduzidas na placa. Essa etapa foi realizada manualmente pelo autor no mesmo *software* em que foi feito o esquemático. As trilhas desenhadas estão ilustradas no Apêndice B.

O *layout* foi utilizado para a fabricação da placa através do processo de corrosão no qual, através de uma solução de percloreto de ferro, a placa é corroída deixando somente as trilhas previamente estampadas na placa. O processo de estampagem se deu pela impressão das trilhas do circuito em papel coche 115 e, com qualquer dispositivo que tenha uma superfície com alta temperatura, justaposição do papel coche com a placa (com as trilhas viradas para a mesma) aquecendo a parte que não está encostando no papel. Realizado esse procedimento por cerca de 15 minutos, lavou-se a placa com água até a remoção de todo o papel e, com as trilhas sobre a placa, prosseguiu-se para o processo de corrosão explicado anteriormente.

Figura 23 – Placa do Circuito de Captação de Áudio

(a) Trilhas por baixo da placa

(b) Circuito com componentes montados

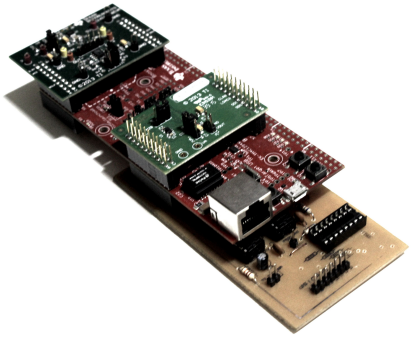
Fonte: O Autor

O resultado desse trabalhoso processo de desenvolvimento do esquema eletrônico, desenho da placa e confecção da mesma é ilustrado na Figura 23, onde na imagem à esquerda mostra a placa logo após o processo de corrosão, já na da direita todos os componentes estão montados. Foi escolhido o uso de soquetes de CIs ao

invés de soldá-los diretamente na placa, a fim de facilitar a rápida troca dos mesmo caso necessário fosse.

Na Figura 24 é possível observar todo o conjunto: placa de desenvolvimento e placa de aquisição e filtragem de sinais. Tomou-se o cuidado no momento da confecção da placa com a dimensão e distância entre as barras pinos que iriam ser responsáveis pelo encaixe da placa de aquisição na placa de desenvolvimento.

Figura 24 – A Unidade de Efeito



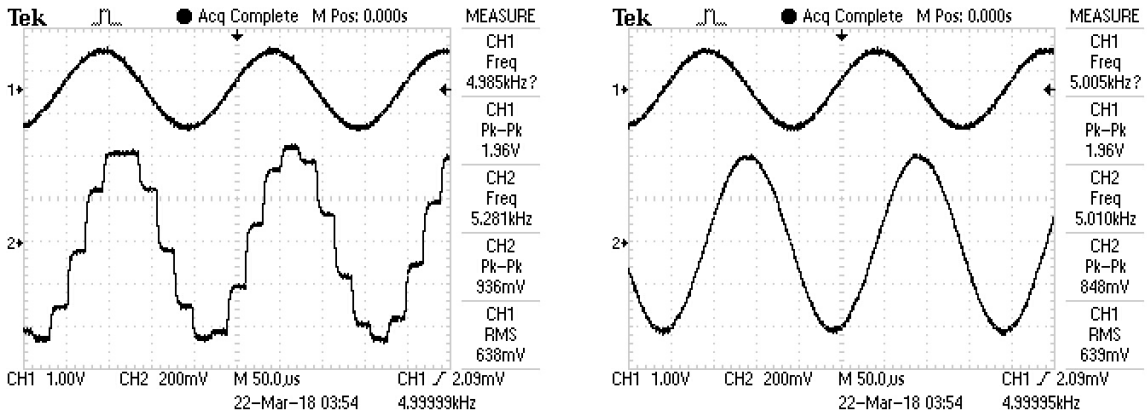
Fonte: O Autor

4 RESULTADOS

4.1 RESULTADO DOS CÓDIGOS DE EFEITO

Antes de testar os códigos de efeito, é necessário observar qual o resultado dos filtros Sallen-Key utilizados para resolver o problema de *aliasing* presente na saída do DAC. Para isso, foi utilizado um osciloscópio juntamente com um gerador de funções e, na entrada de sinal do ADC, foi aplicado uma senoide de 5 kHz, valor no qual o DAC começava a apresentar distorções de onda.

Figura 25 – Comparação entre Pré e Pós Filtro



(a) Antes do Filtro (b) Depois do Filtro

Fonte: O Autor

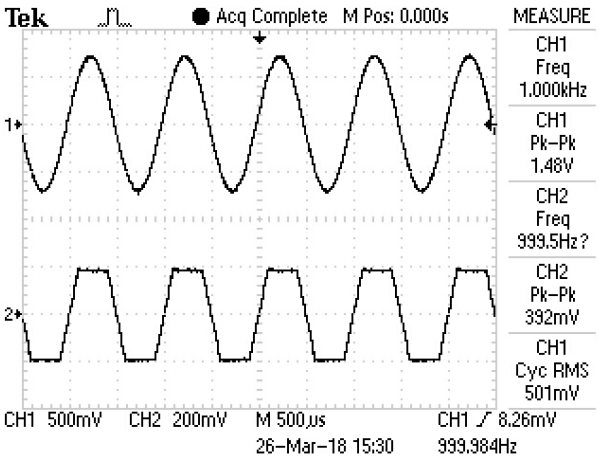
O resultado observado na Figura 25 mostra o antes e o depois da passagem do sinal pelo filtro. Fica evidente a necessidade desse dispositivo a fim de se obter uma fidelidade de sinal, e portanto do áudio, no projeto. Nota-se, entretanto, um pequeno aumento no atraso do sinal de saída, Figura 25b. Esse atraso é de cerca de 50 µs para um frequência de 5 kHz, um valor admissível para este trabalho, pois dificilmente será notado por ouvidos humanos.

4.1.1 Distorção

Os algoritmos já mostrados previamente nesse trabalho foram a base de desenvolvimento dos códigos que de fato foram implementados no microcontrolador.

Como pôde-se perceber na Seção 2.4, o código mais simples de ser feito é o efeito de distorção, e por isso se começou por ele. Basicamente uma tomada de decisão lógica *if/else* faz com que parte da onda seja cortada realizando o que já se era esperado.

Figura 26 – Resultado Algoritmo de Distorção



Fonte: O Autor

O resultado, mostrado na Figura 26, é bastante satisfatório e condiz com o algoritmo proposto. Para o teste desse algoritmo também foi usada uma onda senoidal oriunda do gerador de funções. O valor para o qual o código deveria saturar a onda foi configurado em 3000 de uma escala que vai de 0 a 2^{15} - 1 $\Rightarrow$ 0 a $32\,767$, o que resulta em cerca de 9% da saturação da onda. Lembrando que quanto mais saturada (achatada) a onda, mais "rasgado"será o som.

4.1.2 *Delay*

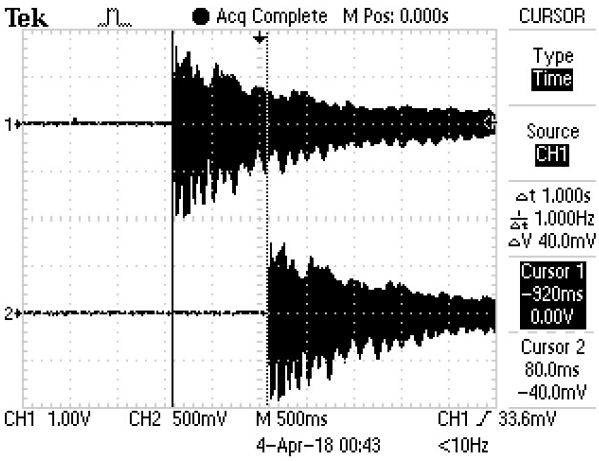
Nesse efeito, o som é simplesmente deslocado para trás no tempo, ou seja, no sentido de atraso. Como foi mostrado no Algoritmo 2, o programa inicializa o vetor com zeros onde será guardado temporariamente o sinal e, após a inicialização, o efeito propriamente dito começa. O código lê o primeiro valor do vetor recém inicializado, envia esse valor o DAC, incrementa a posição do vetor em uma unidade, adquire um sinal do ADC e o grava na posição incrementada. Esse procedimento se repete indefinidamente.

Um parâmetro importante desse efeito é o quanto de *delay* será realizado, pois quanto mais longo o atraso maior será o vetor para guardar todos os valores dos sinais recebidos, além de, é claro, aumentar o período de inicialização do vetor de atraso. Para decidir o tamanho do vetor, deve-se tomar como base a quantidade disponível da SRAM (*Static Random Access Memory* – Memória Estática de Acesso Randômico), pois é nesse tipo de memória que ficarão alocados os valores dos sinais captados pelo ADC neste efeito. No tipo de microcontrolador utilizado o valor da SRAM é de $256\,\mathrm{MB}$, insuficiente para guardar períodos mais longos que dois segundos e meio de áudio. Se tratando de um protótipo, foi utilizado o valor de atraso de um segundo, no qual se faz necessário o uso de um vetor com $48\,000$ posições de 16 bits cada, uma vez que a frequência de amostragem é $48\,\mathrm{kHz}$.

Neste efeito, foi utilizado a captação da guitarra de modo a ter uma melhor

visualização do atraso. Com o osciloscópio preparado para a captura do sinal de saída, reduziu-se a escala de tempo de modo que alguns segundos fossem capturados. O resultado, ilustrado na Figura 27, comprova a eficácia do algoritmo para o efeito de *delay*, o áudio captado pelo osciloscópio no instante em que foi emitido pela guitarra (sinal superior) teve um atraso de um segundo quando saiu da unidade de efeito (sinal inferior), como mostra os cursores posicionados sobre o início de cada sinal (Δt = 1000 s).

Figura 27 – Resultado Algoritmo de *Delay*



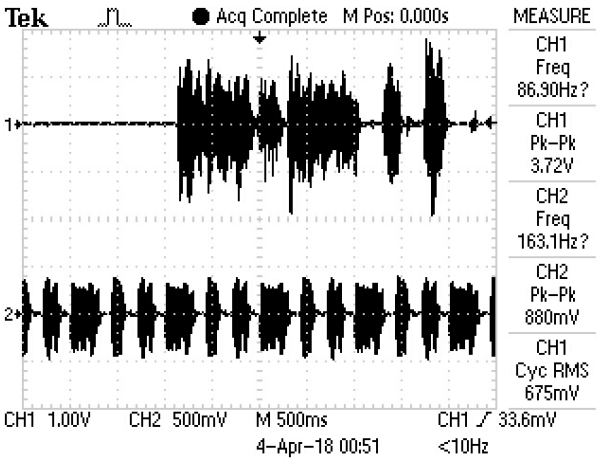
Fonte: O Autor

4.1.3 *Loop*

Esse efeito é muito similar ao anterior, uma vez que também utiliza um laço de repetição sobre um vetor. Porém, ao invés de enviar o sinal ao DAC e posteriormente gravar o próximo valor de áudio, o efeito apenas grava o sinal enquanto o vetor não fica cheio de valores. Uma vez que o *array* de valores de áudio está completo, o algoritmo, agora, apenas envia os valores previamente guardados de modo que sempre volta ao início do vetor, por isso o nome *loop*, em outras palavras, o efeito repete o que foi gravado por indefinidas vezes.

A fim de diferenciar o sinal de entrada do de saída, uma vez que é esperado que a saída fique repetindo o sinal gravado enquanto o sinal da guitarra na entrada do dispositivo pode ter qualquer valor, o áudio a ser repetido indefinidamente foi gravado e, somente após a gravação, foi extraída a imagem do osciloscópio. Da mesma forma que foi utilizado para o efeito *delay*, aqui também utilizou-se o sinal de captação da guitarra diminuindo a escala de tempo.

Figura 28 – Resultado Algoritmo de *Loop*



Fonte: O Autor

O resultado da Figura 28 reflete o que era esperado. O sinal 1 (superior) é um áudio qualquer gerado pela guitarra, enquanto o canal 2 (inferior) é o sinal repetido várias vezes pelo μC, onde nele se percebe o padrão que o mesmo gera na tela do osciloscópio pelo fato do sinal ser repetido.

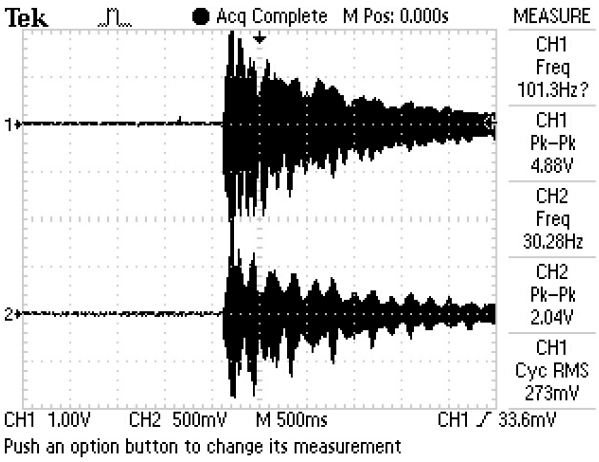
4.1.4 *Tremolo*

De todos os efeitos o tremolo se destaca por ser mais elaborado e, consequentemente, mais trabalhoso. No Algoritmo 5 o efeito de tremolo se mostra bem simples, contudo na simplificação desse código foi omitida o sinal necessário (S_V) para variar a amplitude do sinal da saída. Na pratica esse sinal, que se chamará no código de sinal de tremolo, deve de ser sintetizado no *software* MATLAB a partir de valores resultantes de uma senoide.

Como já explicado na Seção 2.4, o *tremolo* é um efeito que varia a amplitude da onda de entrada com uma frequência baixa. Nas pedaleiras convencionais esse valor varia entre $0{,}5$ Hz e 10 Hz. A escolha dessa frequência de oscilação irá definir quantas amostras serão extraídas do sinal de tremolo no MATLAB, uma vez que ele deve ser gerado com a mesma frequência de amostragem usada na pedaleira, 48 kHz. A título de simplificação e número exato de amostras escolhe-se $4{,}8$ Hz o que resulta em um sinal de oscilação de $10\,000$ amostras. O algoritmo construído para obter as amostras do sinal de *tremolo* é descrito no Apêndice C.

Depois de geradas as amostras, cria-se um vetor no código da unidade do efeito com esses valores para serem usados posteriormente. Quando acionado o efeito, uma amostra é coletada do ADC e multiplicada com o primeiro valor do vetor sinal de tremolo, após isso o resultado da multiplicação é então passado ao DAC, por fim a posição do vetor sinal de tremolo é incrementada.

Figura 29 – Resultado Algoritmo de *Tremolo*



Fonte: O Autor

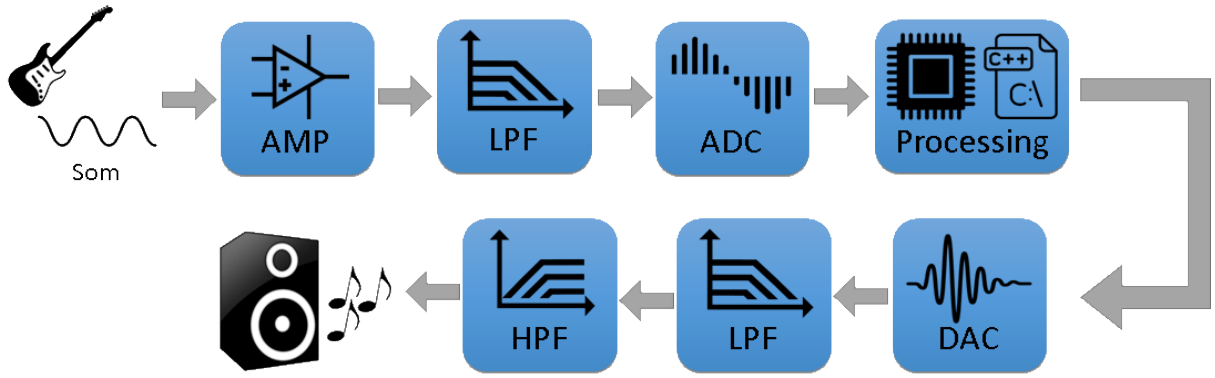
Neste efeito também foi reduzida a escala de tempo no osciloscópio de modo que a variação da amplitude seja mais aparente como mostra a Figura 29. O próprio sinal de entrada por natureza já é um tanto oscilante, mas ainda assim a variação da amplitude do sinal de saída é facilmente observada comprovando o funcionamento do código.

5 CONCLUSÃO

Neste trabalho de conclusão de curso foi apresentada uma solução prática para um dispositivo que realiza efeitos sonoros de sinais provindos de um violão elétrico ou uma guitarra, a qual foi implementada com um microcontrolador e conversores A/D e D/A externos operando a $48\,\text{kHz}$. Circuitos adicionais para amplificadores e filtros passa-baixa foram desenvolvidos cobrindo os limites estabelecidos pela teoria de Nyquist e alguns efeitos foram implementados para demonstrar o uso da unidade de efeito. Interrupções de tempo foram utilizadas a fim de amostrar o sinal de entrada a cada $1 \div 48\,000$ segundos com precisão, antes de enviá-lo para o conversor D/A minimizou o nível de *jitter*. Todo o sistema foi alimentado com uma tensão de $5\,\text{V}$ e apresentou um consumo de $(0{,}47\,\text{W})$, o qual é suprido pela porta USB do kit de desenvolvimento.

Ao final deste projeto, com todos os problemas encontrados durante a parte prática, notou-se que o diagrama mostrado na Figura 2 teve de ser reestruturando para o que de fato foi implementado, sendo este último ilustrado na Figura 30, com a adição de todas as soluções de contorno.

Figura 30 – Diagrama de Blocos Final da Unidade de Efeito



Fonte: O Autor

Embora o haja atraso gerado pelos filtros analógicos e os algoritmos implementados são em geral simplistas, esse protótipo acadêmico da unidade de efeito funcionou como era esperado.

5.1 TRABALHOS FUTUROS

A fim de implementar uma unidade de efeitos digitais de tempo real, ou seja, uma pedaleira que possa ser reparametrizada no instante em que a mesma esteja em uso, além de melhorar a qualidade de áudio, os próximos passos podem ser tomados:

- Reduzir o *jitter* causado por filtros digitais usando interrupções independentes (respectivamente para as rotinas do ADC e do DAC) e respectivos *buffers* de dados que operem em paralelo.
- Criar um servidor na internet (TCP/IP) dentro do microcontrolador que transfira todas as amostras para um cliente remoto. Isso permitiria a criação de um *mixer* virtual onde o número de canais seria limitado à capacidade do computador
- Criar um servidor na internet (TCP/IP) dentro do microcontrolador que receba algoritmos de efeitos de clientes remotos. Isso permitiria que a unidade de efeito fosse reprogramada remotamente durante concertos.

5.2 CONSIDERAÇÕES FINAIS

Muitos dos desafios encontrados durante a realização desse trabalho sequer foram mencionados em sala de aula, pois não abordavam o conteúdo programático vigente, isso devido à alta carga de tópicos a serem cobertos por cada disciplina e que, portanto, devem ser selecionados de acordo com a relevância para o curso, não podendo o mesmo cobrir tudo de todas as disciplinas, algo que é absolutamente compreensível, logo, não sendo este fato culpa das instituições de ensino.

Contudo, isso revela que o conhecimento adquirido durante a graduação é aquele meramente necessário para ter-se noção de assuntos relacionados ao curso escolhido, que neste caso é engenharia elétrica. Dessa maneira, como um trabalho de encerramento de curso, isto na verdade se trata apenas do começo de uma busca constante por conhecimento, conhecimento este que depende da carreira escolhida, gostos por determinados assuntos, etc. Para muitos, essa busca provavelmente jamais acabará visto que ela é essencial para crescermos como profissionais da área e como pessoas que visam construir uma sociedade melhor.

Referências

BASTOS, J. P. A. **Eletromagnetismo para engenharia: estática e quase-estática**. [S.l.]: Ed. da UFSC, 2004. Citado na página 30.

BRAIN, M. **How Electric Guitars Work**. 2002. Disponível em: <https://entertainment.howstuffworks.com/electric-guitar1.htm>. Acesso em: 23 de Outubro de 2017. Citado 2 vezes nas páginas 30 e 31.

KUPHALDT, T. R. Lessons in electric circuits, volume iv – digital. **Vol. Fifth Edition. Open Book Project**, 2006. Citado 2 vezes nas páginas 20 e 24.

MARTINS, N. A. **Sistemas microcontrolados: uma Abordagem com o Microcontrolador PIC 16F84**. [S.l.]: Novatec Editora. São Paulo, 2005. Citado 2 vezes nas páginas 14 e 15.

OPPENHEIM, A. V. **Discrete-time signal processing**. [S.l.]: Pearson Education India, 1999. Citado 3 vezes nas páginas 19, 20 e 37.

PUHLMANN, H. **Processamento Digital de Sinais - DSP**. 2014. Disponível em: <www.embarcados.com.br/introducao-ao-processamento-digital-de-sinais-dsp-parte-1/>. Acesso em: 27 de Maio de 2017. Citado na página 20.

RATTON, M. **Dicionário de áudio e tecnologia musical**. [S.l.: s.n.], 2004. Citado na página 28.

SEDRA, A. S. et al. **Microeletrônica**. [S.l.]: Pearson Prentice Hall, 2007. Citado na página 20.

TARQUIN, B. **Stomp on this! The Guitar Pedal Effects Guidebook**. [S.l.]: Cengage Learning, 2015. Citado 2 vezes nas páginas 13 e 14.

TEXAS INSTRUMENTS. **Alalysis Of The Sallen-Key Architecture**: Application report. [S.l.], 1999. 18 p. Disponível em: <http://www.ti.com.cn/cn/lit/an/sloa024b/sloa024b.pdf>. Acesso em: 23 de Outubro de 2017. Citado na página 37.

TEXAS INSTRUMENTS. **Low Power, Single-Supply, Rail-to-Rail Operational Amplifiers**: microamplifier™ series. [S.l.], 2000. 30 p. Disponível em: <http://www.ti.com/lit/ds/symlink/opa2345.pdf>. Acesso em: 01 de Maio de 2018. Citado na página 34.

TEXAS INSTRUMENTS. Getting started with tina-ti. **TINA-TI SPICE Simulator Quick Start Guide (SBOU052)**, 2007. Citado na página 35.

TEXAS INSTRUMENTS. **ADC161S626 16-Bit, 50 to 250 kSPS, Differential Input, MicroPower ADC**. [S.l.], 2008. 33 p. Disponível em: <www.ti.com/lit/ds/symlink/adc161s626.pdf>. Acesso em: 03 de Janeiro de 2016. Citado na página 21.

TEXAS INSTRUMENTS. **DAC161S055 Precision 16-Bit, Buffered Voltage-Output DAC**. [S.l.], 2010. 30 p. Disponível em: <www.ti.com/lit/ds/symlink/dac161s055.pdf>. Acesso em: 15 de Janeiro de 2016. Citado na página 24.

TEXAS INSTRUMENTS. **16-bit 1 Channel Digital to Analog Converter Evaluation Module**. 2014. Disponível em: <http://www.ti.com/tool/DAC161S055EVM>. Acesso em: 29 de Maio de 2018. Citado na página 25.

TEXAS INSTRUMENTS. **Meet The TIVA™ C Series TM4C1294**: Connected launchpad evaluation kit. [S.l.], 2014. 3 p. Disponível em: <http://www.ti.com/lit/ml/spmz858/spmz858.pdf>. Acesso em: 24 de Outubro de 2017. Citado na página 27.

TEXAS INSTRUMENTS. **TIVA™ C Series TM4C1294 Connected LaunchPad Evaluation Kit: EK-TM4C1294XL**: User's guide. [S.l.], 2014. 38 p. Disponível em: <www.ti.com/lit/ug/spmu365c/spmu365c.pdf>. Acesso em: 10 de Dezembro de 2016. Citado na página 17.

TEXAS INSTRUMENTS. **TIVA™ TM4C1294NCPDT Microcontroller**: Data sheet. [S.l.], 2014. 1890 p. Disponível em: <www.ti.com/lit/ds/symlink/tm4c1294ncpdt.pdf>. Acesso em: 12 de Dezembro de 2016. Citado na página 26.

TEXAS INSTRUMENTS. **16-bit 1 Channel Differential Input SAR ADC Evaluation Module**. 2015. Disponível em: <http://www.ti.com/tool/adc161s626evm>. Acesso em: 30 de Maio de 2018. Citado na página 21.

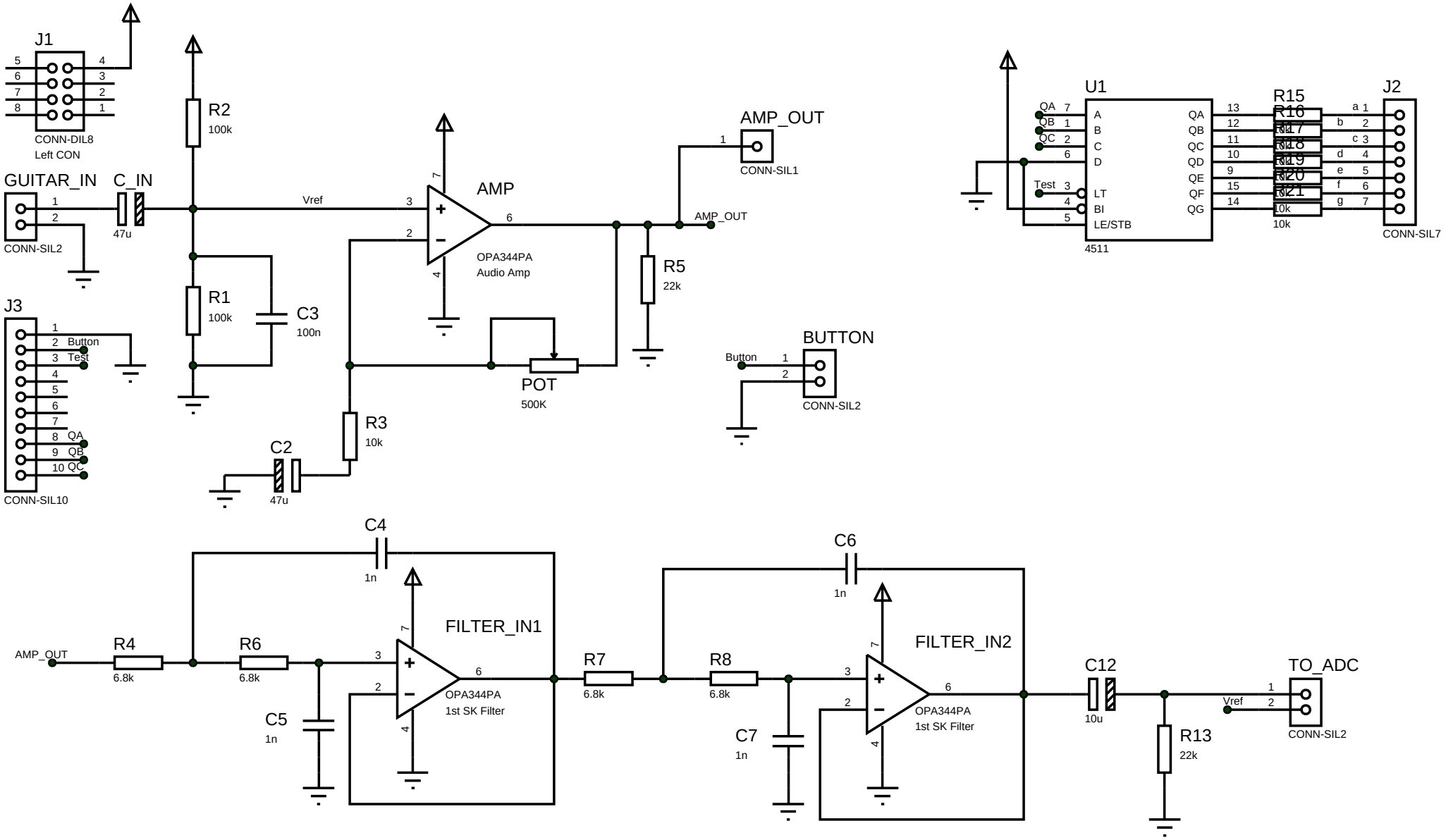
TEXAS INSTRUMENTS. **ARM® Cortex®-M4F-Based MCU TM4C1294 Connected LaunchPad™ Evaluation Kit**. 2015. Disponível em: <http://www.ti.com/tool/EK-TM4C1294XL>. Acesso em: 01 de Junho de 2018. Citado na página 26.

TEXAS INSTRUMENTS. **Trimming a Digital-to-Analog Converter to Improve Accuracy**. 2016. Disponível em: <https://www.edn.com/Home/PrintView?contentItemId=4442636>. Acesso em: 29 de Maio de 2018. Citado na página 24.

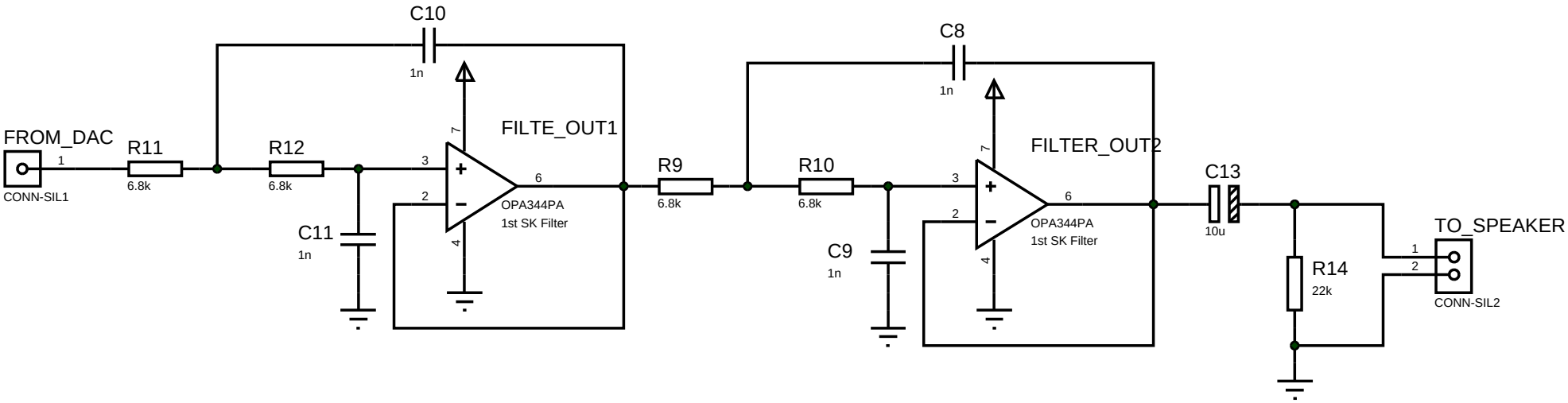
THOMPSON, D. **The Stompbox A History of Guitar Fuzzes, Flangers, Echoes and Wahs**. [S.l.]: Miller Freeman, 1997. Citado na página 13.

Apêndices

APÊNDICE A – Circuito Eletrônico da Unidade de Efeito

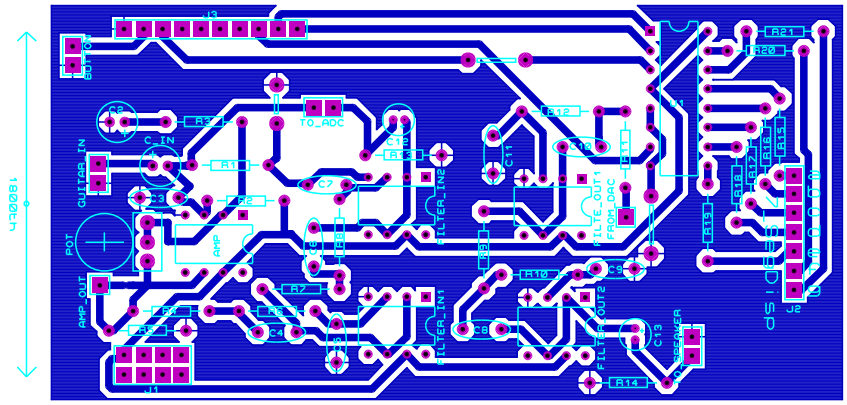


FILE NAME: **Guitar Pedal**	DATE: **5/22/2018**
DESIGN TITLE: **Senior Project Board**	PAGE: **1 of 2**
ORGANIZATION: UTFPR	
BY: **Geovani Cardozo Alves** REV: **01**	TIME: **8:37:20 PM**



FILE NAME: **Guitar Pedal**	DATE: **5/22/2018**
DESIGN TITLE: **Senior Project Board**	PAGE: **2 of 2**
ORGANIZATION: UTFPR	TIME: **8:37:20 PM**
BY: **Geovani Cardozo Alves**	REV: **01**

APÊNDICE B – *Layout* Placa de Circuito Impresso



APÊNDICE C – Código para Gerar a Onda de *Tremolo* no MATLAB

```
% MATLAB CODE FOR THE TREMOLO WAVE SAMPLES
% By Geovani Cardozo Alves
% Advisor: Marcelo de Oliveira Rosa
% Senior project of Universidade Tecnologica Federal do Parana
% Curitiba - 2018

% sample frequency
fs = 48000;
% sample period
Ts = 1/fs;

% tremolo frequency
ft = 4.8;
% tremolo period
Tt = 1/ft;

% calculating time variable
t = 0 : Ts : Tt - Ts;

% calculating tremolo wave
tremolo_wave = 0.75 + 0.25*sin(2*pi*ft*t);
```

APÊNDICE D – Código da Unidade de Efeito

```
// SOUND EFFECT UNIT
// By Geovani Cardozo Alves
// Advisor: Marcelo de Oliveira Rosa
// Senior project of Universidade Tecnologica Federal do Parana
// Curitiba – 2018


#include <stdbool.h>
#include <stdio.h>
#include <stdint.h>
#include <math.h>
#include "inc/hw_memmap.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/ssi.h"
#include "driverlib/sysctl.h"
#include "inc/hw_types.h"
#include "driverlib/interrupt.h"
#include "driverlib/timer.h"
#include "inc/tm4c1294ncpdt.h"

// Defining Variables
#define SAMPLE_FREQ 48000
#define MAX_DISTORTION 3000


// Number of effects + Default = 5
volatile uint8_t ui8Count=0;
// ui16Index for the LOOP effect
volatile uint16_t ui16Index=0, ui16TmloCount=0, ui16Delay=0;
volatile bool recording = false;
uint32_t pui32DataRx[3], ui32SysClkFreq, ui32Period;
int16_t convertedValue, soundWave[120000];

// Tremolo wave declaration. Values were
// omited in order to no poluate the document
const float tremolo[10000];

// Function Prototype
void Effects(void);

// Timer interrupt in order to provide a sample frequency of 48 kHz
void InitTimerInt(){
```

```
44
45    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
46
47    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
48
49    ui32Period = ui32SysClkFreq / SAMPLE_FREQ;
50
51    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period - 1);
52
53    IntEnable(INT_TIMER0A);
54    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
55    IntMasterEnable();
56 }
57
58 // Handler code for the interruption call of PortJ
59 // When called the program changes de effect by changing the ui8Count
      number
60 void PortJIntHandler(void) {
61
62    uint16_t i;
63
64    GPIOIntClear(GPIO_PORTJ_BASE, GPIO_INT_PIN_0);
65    TimerDisable(TIMER0_BASE, TIMER_A);
66
67    ui8Count++;
68    if (ui8Count > 4)
69      ui8Count = 0;
70
71    ui16Index = 0;
72
73    // Reset all LEDs
74    GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0 | GPIO_PIN_1, 0);
75    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0 | GPIO_PIN_4, 0);
76
77    // Choose the LEDs which corresponds to the effect selected
78    switch (ui8Count){
79
80    case 1:
81      // DISTORTION
82      // D3 on
83      GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4);
84
85      break;
86
87    case 2:
88      // DELAY / ECHO
89      // D2 on
```

```
90
91      for(i=0 ; i < SAMPLE_FREQ ; i++){
92        soundWave[i]=0;
93      }
94
95      ui16Index = 0;
96      ui16Delay = 0;
97
98      GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0);
99
100     break;
101
102   case 3:
103     // LOOP
104     // D2 and D3 on
105     GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0);
106     GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4);
107
108     // Show that it's recording
109     // Light LED D4
110     GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);
111     recording = true;
112
113     break;
114
115   case 4:
116     //TREMOLO
117     // D1 and D3 on
118     GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, GPIO_PIN_1);
119
120     break;
121
122   default:
123     // No Effect
124     // Both LED off
125
126     GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0 | GPIO_PIN_1,
127         GPIO_PIN_0 | GPIO_PIN_1);
128     GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0 | GPIO_PIN_4,
129         GPIO_PIN_0 | GPIO_PIN_4);
130   }
131
132   // Button bounce delay
133   SysCtlDelay(ui32SysClkFreq / 2);
134
135   TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period - 1);
136   TimerEnable(TIMER0_BASE, TIMER_A);
```

```
137 }
138
139 void Timer0IntHandler(void) {
140 // Clear the timer interrupt
141   TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
142
143   Effects();
144 }
145
146 // Initialization of PORTJ PIN0, the button for the above interruption
147 void InitButton(void) {
148
149   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOJ);
150
151   while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOJ)) {
152   }
153
154   GPIOPinTypeGPIOInput(GPIO_PORTJ_BASE, GPIO_PIN_0);
155
156   GPIOPadConfigSet(GPIO_PORTJ_BASE, GPIO_PIN_0, GPIO_STRENGTH_2MA,
157   GPIO_PIN_TYPE_STD_WPU);
158   GPIOIntTypeSet(GPIO_PORTJ_BASE, GPIO_PIN_0, GPIO_FALLING_EDGE);
159   GPIOIntRegister(GPIO_PORTJ_BASE, PortJIntHandler);
160   GPIOIntEnable(GPIO_PORTJ_BASE, GPIO_INT_PIN_0);
161
162 }
163
164 // Initializing PORTD ==> FSS for the DAC
165 void InitPORTD2(void) {
166
167   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
168
169   while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOD)) {
170   }
171
172   GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, GPIO_PIN_2);
173
174   GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2, GPIO_PIN_2);
175 }
176
177 // Initializing PORTM0 --> DAC Clear
178 // This port maintains the Clear pin of the DAC at High level
179 void InitPORTM0(void) {
180
181   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOM);
182
183   while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOM)) {
```

```
184     }
185
186     GPIOPinTypeGPIOOutput(GPIO_PORTM_BASE, GPIO_PIN_0);
187
188     GPIOPinWrite(GPIO_PORTM_BASE, GPIO_PIN_0, GPIO_PIN_0);
189 }
190
191 // Initializing PORTD ==> FSS for the ADC
192 void InitPORTE4(void) {
193     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
194
195     while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOE)) {
196     }
197
198     GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, GPIO_PIN_4);
199
200     GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_4, GPIO_PIN_4);
201 }
202
203 void InitADCSSI() {
204
205     // The SSI0 peripheral must be enabled for use.
206     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
207     SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI2);
208
209     // Configure the pin muxing for SSI2 functions on port D0, D1, and D3.
210     GPIOPinConfigure(GPIO_PD3_SSI2CLK);
211     //GPIOPinConfigure(GPIO_PD2_SSI2FSS);  Function held by PORTE4 (InitPORTE
        )
212     GPIOPinConfigure(GPIO_PD0_SSI2XDAT1);
213     GPIOPinConfigure(GPIO_PD1_SSI2XDAT0);
214
215     GPIOPinTypeSSI(GPIO_PORTD_BASE, GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_3);
216
217     SSIConfigSetExpClk(SSI2_BASE, ui32SysClkFreq, SSI_FRF_MOTO_MODE_3,
218     SSI_MODE_MASTER, 5000000, 16);
219
220     //SSIAdvModeSet(SSI2_BASE, SSI_ADV_MODE_LEGACY);
221
222     // Enable the SSI0 module.
223     SSIEnable(SSI2_BASE);
224
225     // Emptying the FIFO
226     while (SSIDataGetNonBlocking(SSI2_BASE, &pui32DataRx[0])) {
227     }
228
229 }
```

```
230
231 void InitDACSSI(void) {
232
233   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOQ);
234   SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI3);
235
236   GPIOPinConfigure(GPIO_PQ0_SSI3CLK);
237   GPIOPinConfigure(GPIO_PQ3_SSI3XDAT1);
238   GPIOPinConfigure(GPIO_PQ2_SSI3XDAT0);
239
240   GPIOPinTypeSSI(GPIO_PORTQ_BASE, GPIO_PIN_0 | GPIO_PIN_2 | GPIO_PIN_3);
241
242   SSIConfigSetExpClk(SSI3_BASE, ui32SysClkFreq, SSI_FRF_MOTO_MODE_0,
243   SSI_MODE_MASTER, 20000000, 24);
244
245   SSIEnable(SSI3_BASE);
246
247   while (SSIDataGetNonBlocking(SSI3_BASE, &pui32DataRx[0])) {
248   }
249
250 }
251
252 // Initializing the DAC
253 void InitDAC(void) {
254
255   // Commands to initialing the DAC
256   uint8_t command[6] = { 0x01, 0x08, 0x18, 0x28, 0x08 };
257   uint8_t first_data[6] = { 0x00, 0x40, 0x00, 0x00, 0xFF };
258   uint8_t second_data[6] = { 0x00, 0x00, 0x01, 0xFE, 0xFF };
259   uint8_t uintIndex;
260
261   for (uintIndex = 0; uintIndex < 2; uintIndex++) {
262
263     GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2, 0);
264
265     SSIDataPut(SSI3_BASE, command[uintIndex]);
266     while (SSIBusy(SSI3_BASE)) {
267     }
268
269     SSIDataPut(SSI3_BASE, first_data[uintIndex]);
270     while (SSIBusy(SSI3_BASE)) {
271     }
272     SSIDataPut(SSI3_BASE, second_data[uintIndex]);
273     while (SSIBusy(SSI3_BASE)) {
274     }
275
276     GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2, GPIO_PIN_2);
```

```
277   }
278 }
279
280 // Reads value from the ADC
281 void ReadADC(void) {
282
283   uint8_t ui8Index;
284   uint16_t adcRX[2];
285   uint32_t dummy;
286
287   while (SSIDataGetNonBlocking(SSI2_BASE, &pui32DataRx[0])) {}
288
289   GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_4, 0);
290
291   for (ui8Index = 0; ui8Index < 2; ui8Index++) {
292
293     SSIDataPut(SSI2_BASE, 0x00);
294
295     SSIDataGet(SSI2_BASE, &dummy);
296
297     adcRX[ui8Index] = (dummy);
298   }
299
300   GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_4, GPIO_PIN_4);
301
302   convertedValue = (((adcRX[0] << 2) | (adcRX[1] >> 14)) & 0xFFFF);
303 }
304
305 // Function to Write to DAC output
306 void WriteDAC(uint16_t data) {
307
308   GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2, 0);
309
310   // Write Command to SPI
311   SSIDataPut(SSI3_BASE, 0x08);
312   while (SSIBusy(SSI3_BASE)) {
313   }
314   SSIDataPut(SSI3_BASE, data >> 8);
315   while (SSIBusy(SSI3_BASE)) {
316   }
317   SSIDataPut(SSI3_BASE, data & 0xFF);
318   while (SSIBusy(SSI3_BASE)) {
319   }
320
321   GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2, GPIO_PIN_2);
322
323 }
```

```
324
325 // Initializes the LED that will show the user which effect was selected
326 // PORTs N (D0, D1) and F (D3, D4)
327 void InitLED(){
328   // PORTN
329   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);
330
331   while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPION)) {}
332
333   GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_0 | GPIO_PIN_1);
334
335   GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0 | GPIO_PIN_1, 0);
336
337   // PORTF
338   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
339
340   while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOF)) {}
341
342   GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0 | GPIO_PIN_4);
343
344   GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0 | GPIO_PIN_4, 0);
345 }
346
347 int main(void) {
348
349   ui32SysClkFreq = SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ |
350   SYSCTL_OSC_MAIN | SYSCTL_USE_PLL |
351   SYSCTL_CFG_VCO_480), 120000000);
352
353   // Call all the initialization codes
354
355   InitPORTD2();
356
357   InitPORTM0();
358
359   InitPORTE4();
360
361   InitADCSSI();
362
363   InitLED();
364
365   InitDACSSI();
366
367   InitButton();
368
369   InitDAC();
370
```

```
371   InitTimerInt();
372
373   PortJIntHandler();
374
375   while (1) {}
376 }
377
378 void Effects(void){
379
380   switch (ui8Count) {
381
382   case 1:
383     //First Effect
384     //DISTORTION
385
386     ReadADC();
387
388     if (convertedValue > MAX_DISTORTION )
389       convertedValue = MAX_DISTORTION;
390     else if (convertedValue < -MAX_DISTORTION)
391       convertedValue = -MAX_DISTORTION;
392
393     WriteDAC(convertedValue + 32768);
394
395     break;
396
397   case 2:
398     //Second Effect
399     //DELAY
400
401     ReadADC();
402
403     WriteDAC(soundWave[ui16Index] + 32768);
404     soundWave[ui16Index] = convertedValue;
405
406     ui16Index++;
407     ui16Index = ui16Index % SAMPLE_FREQ;
408
409     break;
410
411   case 3:
412     //Third Effect
413     //LOOP
414
415     // If recording then...
416     if (recording){
417
```

```
418         ReadADC();
419         soundWave[ui16Index] = convertedValue;
420
421         ui16Index++;
422
423         if (ui16Index == SAMPLE_FREQ) {
424           recording = false;
425           ui16Index = 0;
426           GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0);
427         }
428       }
429
430       // If not recording then...
431       if (!recording) {
432
433         WriteDAC(soundWave[ui16Index] + 32768);
434
435         ui16Index++;
436         ui16Index = ui16Index % SAMPLE_FREQ;
437       }
438
439       break;
440
441     case 4:
442       //Fourth Effect
443       //TREMOLO
444
445       ReadADC();
446       WriteDAC((uint16_t) (convertedValue*tremolo[ui16TmloCount] + 32768));
447
448       ui16TmloCount++;
449       ui16TmloCount = ui16TmloCount % 10000;
450
451       break;
452
453     default:
454
455       ReadADC();
456       WriteDAC(convertedValue + 32768);
457
458     }
459 }
```