Simulador de amplificador de áudio

Paulo Augusto M. F. de Souza
Estudante de Engenharia Eletrônica
Faculdade Gama - FGA UnB
Email: pauloaugustomiguelfonseca@gmail.com

Tiago Martins de Brito
Estudante de Engenharia Eletrônica
Faculdade Gama - FGA UnB
Email: tiagomartinsbrito879@gmail.com

***Resumo*—Com o processamento de sinais de áudio, é possivel modificar as características originais do sinal. Por ser de áudio, pode-se ver o seu comportamento em faixas de frequência diferente e com isso equalizar um sinal da maneira pretendida através de filtros. Também é possível amplificar ele de uma maneira que o torne mais intenso ou menos intenso, modificando a sua amplitude.**

***P*alavras-chaves – Frequência, filtro, amplificador, equalizador.**

I. Introdução

A. Justificativa

Para tratamento de sinais de áudio com equipamentos digitais ou analógicos, ainda há um elevado custo, então como esse projeto tem como motivação tentar reduzir esses custos, mas tendo em vista manter uma certa qualidade. Com esse custo reduzido, poderia ser usado para o aprendizado para quem ainda está aprendendo e não quer gastar muito com equipamentos de áudio.

B. Objetivos

O objetivo deste trabalho, é mostrar como pode ser feita a equallização de áudio, com o Raspberry Pi através da amostragem de um sinal.

C. Requisitos

A priori está sendo estudada a possibilidade de usar um conversor AD, de pelo menos 16 bits e uma frequência de amostragem de 40kHz, ou no lugar dele colocar uma interface de áudio, que tem uma taxa de amostragem de 48kHz e de 24 bits. Isso se deve pois o Raspberry não possui conversor AD, então é preciso usar um externo.

Sensores serão usados para colocar algum tipo de efeito durante o processamento do sinal pelo Raspberry, como por exemplo um sensor de distância para colocar a intensidade do efeito. Também pode ser adicionado um VU-Meter, para mostrar as bandas de frequência do sinal.

Para todos os cálculos e processamentos dos sinais, será usado o Raspberry Pi 3. Ele será usado pois, tem uma grande capacidade de processamento, e com uma boa velocidade. O uso dele permite criar sistemas embarcados para essa aplicação de processamento de sinais de áudio.

II. Revisão Bibliográfica

A. Teorema da amostragem

Para o processamento computacional de um sinal analógico, antes é preciso fazer a amostragem do sinal. Isso consiste em discretizar o sinal, ou seja, torna-lo digital e com isso fazer o seu processamento em um computador, ou no Raspberry Pi. Amostrar é definir para um certo intervalo de tempo, um valor para o sinal. Onde esse ele apenas possui um valor definido em um intervalo de tempo específico.

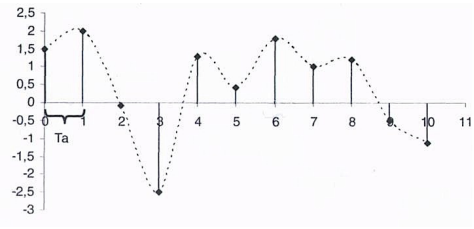

Figura 1. Amostragem

Na Figura 1, é possível ver justamente, onde para cada intervalo de tempo é que se tem uma amostra do sinal, um valor definido apenas em um intervalo de tempo específico. Em intervalos de tempo não definido, o sinal recebe zero como seu valor.

Um meio para as discretização, é usar o critério de Nyquist para amostragem, que diz que a frequência de amostragem deve ser de pelo menos duas vezes a frequência do sinal. Isso se deve para que se possa retomar o sinal original com o mínimo de distorções possíveis. Na Figura 2 é possivel ver justamente esse critério, pois quando a frequência de amostragem é duas vezes menor, há uma sobreposição do sinal, e isso não permite voltar ao sinal original. [**?**]

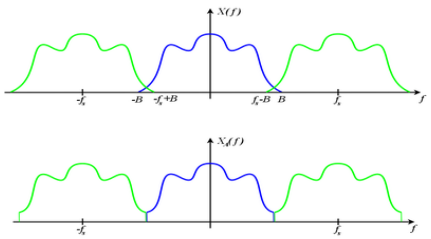

Figura 2. Nyquist

Fonte: https://pt.wikipedia.org/wiki/Teorema_da_amostragem_de_Nyquist-Shannon

B. Processamento de sinais

Em áudio a largura de banda corresponde ao espectro audível, compreendido entre 20 e 20kHz. Então, a partir de

dispositivos de captação sonora, o objetivo é trabalhar com faixas de frequência compreendidas na parte audível. Com isso, se busca então trabalhar com a resposta em frequência dessa faixa. [2]

O processamento desses sinais, podem ser feitos com a aplicação de filtros, que delimitam uma faixa específica para se trabalhar. Esses filtros, que podem ser, passa baixas, passa altas ou passa bandas, servem para fazer a equalização de um sinal de áudio. Além dessa equalização, também podem ser incorporados controles para ajuste da amplitude dos sinais antes e após o processamento. [1]

Filtros passa baixas, são dispositivos que permitem a passagem de um sinal até uma certa frequência máxima. Passa altas, são os que permitem a passagem a partir de uma frequência mínima. E passa bandas, são os que permitem a passagem a partir de uma frequência mínima até uma máxima. As figuras abaixo mostram justamente a resposta em frequência de cada filtro.

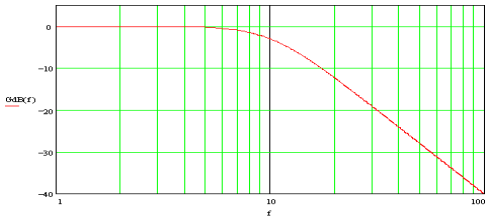

Figura 3. Passa baixas

Fonte:
http://www.alvaroneiva.site.br.com/filteq2_arquivos/image011.gif

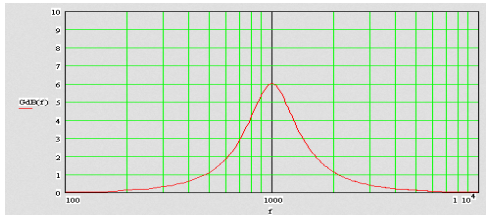

Figura 4. Passa banda

Fonte:
http://www.alvaroneiva.site.br.com/filteq2_arquivos/image015.gif

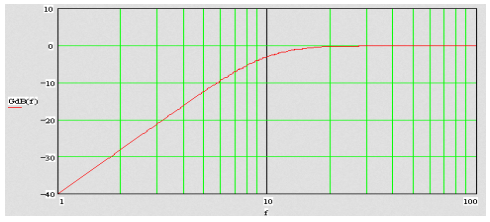

Figura 5. Passa alta

Fonte:
http://www.alvaroneiva.site.br.com/filteq2_arquivos/image013.gif

III. Desenvolvimento

A primeira parte do desenvolvimento, foi conectar a interface de áudio Roland DuoCapture UA11 ao Linux. Essa conectividade foi possível com a entrada P10 da interface, pois quando tentamos usar o P2 não foi possível gravar um arquivo de áudio. A gravação, foi possível usando um software para esse tipo de tarefa, o Ardour, e também um que já vem na Raspberry Pi, o Alsamixer.

Após a gravação, é possível ouvir o áudio gravado, com o Ardour, na saída da interface. Quando a gravação é feita usando o Alsamixer é possível ouvir o som na saída P2 da Raspberry Pi.

Após a leitura, o próximo passo então é fazer o processamento desse sinal de áudio, e adicionar efeitos e aplicar filtros para tratamento sinal recebido na Raspberry. Esses filtros podem ser desenvolvidos na da Raspberry e com isso modificar o sinal para que possa ser percebido as diferenças na saída.

A. *Descrição de Hardware*

A ligação da interface com a Raspberry Pi é feita através do cabo USB que permite a sua comunicação. O instrumento é conectado na interface de áudio para que seja feito uma conversão AD para que seja possível a leitura e gravação dos dados.

A conecção do instrumento pode ser feita com o conector de entrada P10 da interface permitindo assim que ela faça a conversão AD. Como essa interface possui uma taxa de amostragem de 48KHz, a qualidade do sinal audível possui uma boa qualidade com isso é possível fazer uma boa análise do áudio e consequentemente o seu tratamendo através de filtros.

Figura 6. Interface de áudio

Fonte: https://www.roland.com/us/products/duo-capture/

A Figura 6 traz a interface de áudio usada para nosso projeto. Nela é possível que ela contém duas entradas e duas saídas. E atrás dela possui o conector para se ligar o cabo USB e se fazer sua conexão com o computador ou a Raspberry Pi.

Na Figura 7 é possível ver como ele pode ser ligado ao computador ou ao Raspberry Pi. Esse tipo de interface permite que não apenas um tipo de instrumento ou microfone possa ser ligado a ela.

Esse tipo de implementação de hardware permite que se faça então a leitura do PC ou da Raspberry do sinal que pode ser tratado e então enviado para saídas de áudio e então ser escutado por uma pessoa através de um fone ou auto falante.

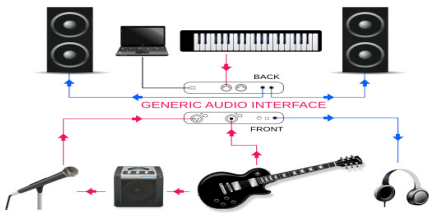

Figura 7. Conecção da interface e o PC

Fonte: http://libremusicproduction.com/sites/default/files/articles/Bedroom%20musician%20setup.png

B. Descrição de Software

Implementação do Filtro passa-baixas

Foi realizada uma tentativa de implementar um filtro passa baixa usando média móvel. Para isso abriu-se um arquivo de áudio dentro de um programa em C, copiou-se as amostras deste arquivo e essas amostras foram coladas em um outro arquivo de audio com o filtro implementado. Contudo esse tipo de implementação não é a esperada para o projeto que visa uma tentativa de fazer o processamento do sinal em tempo real. Esse tipo de tentativa foi feita pelo motivo de não conseguir ler as amostras vindas da interface de audio via usb. O código usado para o filtro passa-baixas com médias móveis é da Figura 8:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{

        FILE *fp;
        FILE *fp1;
        char c[10];
        int i = 0;
        float media = 0.0;
        fp = fopen("Corvette_pass.wav", "r");
        fp1 = fopen("out.wav", "w");

        if (!fp){ printf("deu ruim"); exit(0);}
        if(!fp1){printf("deu ruim"); exit(1);}
        while((c[i] = fgetc(fp)) != EOF){
                media += c[i];
                i++;
                if(i==10){
                        media = media/10.0;
                        putc(media, fp1);
                        media = 0.0;
                        i = 0;}
                        }


        fclose(fp);
        fclose(fp1);

        return 0;

}
```

Figura 8. Código usado como tentantiva de implementação

O código começa abrindo um arquivo de áudio .wav, apenas para leitura e outro para escrita. Foi feito a leitura desse arquivo posição por posição e então gravado em um caractere "c". Realizou-se a soma desses caracteres para calcular a média entre eles com 10 caracteres. Essa foi a lógia usada para fazer um filtro com médias móveis. No entanto, o problema apresentado aqui é que se faz necessário gravar primeiro o arquivo de áudio (um sinal de áudio) e depois processar o arquivo. Mas o propósito do nosso projeto é tentar implementar esse processo em tempo real, criando assim um equalizador. Porém, esta foi a tentativa realizada nesse ponto de controle.

REFERÊNCIAS

[1] Henrique Bartoski Ferreira. Processamento de sinais digitais de áudio no raspberry pi e beaglebone black. B.S. thesis, Universidade Tecnológica Federal do Paraná, 2014.

[2] Christian Gonçalves Herrera. *Projeto de Sistemas de Processamento Digital de Sinais de Áudio Utilizando Metodologia Científica*. PhD thesis, Universidade Federal de Minas Gerais, 2004.