UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

LUCAS NEDEL KIRSTEN

Implementação digital da caixa Leslie em espaço livre

Porto Alegre

2018

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

LUCAS NEDEL KIRSTEN

Implementação digital da caixa Leslie em espaço livre

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para Graduação em Engenharia Elétrica

Orientador: Prof. Dr. Adalberto Schuck Jr.

Porto Alegre
2018

Dedico este trabalho ao meu
avô Olmiro Emílio Nedel.

AGRADECIMENTOS

Agradeço aos meus pais por todo o apoio e dedicação que tiveram comigo durante todos os anos da graduação.

A minha irmã, Camila, e a toda minha família por sempre acreditarem em minha capacidade.

A minha namorada, Diandra, por estar ao meu lado durantes esses anos de altos e baixos.

Ao Lucas Santolin, por me ajudar na confecção da placa de circuito impresso do pedal.

Aos amigos e colegas de curso, pelos momentos de descontração.

Aos professores que estavam dispostos a dar seu melhor.

Ao professor Adalberto Schuck pelos anos de orientação e dedicação.

Se enxerguei mais longe, foi porque me apoiei sobre os ombros de gigantes.

Isaac Newton

Resumo

O presente trabalho aborda os fundamentos teóricos da descrição e discretização dos efeitos gerados pela caixa Leslie, a fim de se produzir um sistema embarcado capaz de simula-lá. Os efeitos trazidos pela Leslie são: a distorção harmônica, a separação entre as frequências graves e agudas entorno dos 800Hz, o efeito Doppler e a modulação em amplitude. Tais efeitos foram quantificados e discretizados em forma de algoritmos. Esses algoritmos foram simulados no MATLAB 2012b e transcritos para linguagem C++, a fim de se programar o sistema dentro da placa de desenvolvimento Teensy 3.6. Projetou-se circuitos periféricos para adequação dos sinais de entrada e saída, e desenvolveu-se um circuito capaz de fornecer tensão simétrica nos níveis de -5V, 0V e 5V para alimentar o pedal. A sonoridade resultante foi comparada a do pedal RT-20 da BOSS em diferentes configurações com senoides e ondas triangulares de entrada. Apesar das sonoridades resultantes se aparentarem, o formato dos sinais de saída para frequências acima da frequência de 800Hz se mostraram bastante distintos, enquanto que sinais abaixo dessa frequência tiveram maior semelhança. As discrepâncias foram atribuídas ao fato do sistema implementar aproximações dos efeitos Doppler e da modulação em amplitude, e não implementar as múltiplas reflexões do som que ocorrem dentro da caixa. Entretanto, agregada às múltiplas combinações de variações dos parâmetros dos efeitos, tal pedal seria capaz de simular aspectos não considerados de outros modelos e outros efeitos que se baseiam nos da Leslie.

Palavras-chave: Engenharia Elétrica. Processamento Digital de Sinais. Caixa Leslie. Pedal digital.

Abstract

This paper intends to approach the theoretical foundations of the description and discretization of Leslie cabinet's effects, in order to produce an embedded system capable of simulate it. The Leslie's effects over an input sound are: the harmonic distortion, the crossover between the treble and the bass sounds over 800Hz, the Doppler effect and the amplitude modulation. The effects were quantified and discretized into algorithms. These algorithms were simulated in MATLAB 2012b and converted to C++ in the Teensy 3.6 development board. An input and output circuits were designed to suit the signals, and a symmetric supply circuit were also designed to provide -5V, 0V and 5V voltage levels. The resulting sounds of the pedal were compared to the BOSS' RT-20 pedal in different configurations with sine and triangular waves input signals. Although the resulting sonority was similar, the shape of the output signals to frequencies above 800Hz were very distinct, while the signals under this frequency had more similarities. Those nonconformities were attributed to the approximations in the Doppler effect and the amplitude modulation, joined to the multiple feedback non-inplementation of the sound in the Leslie's interior. However, joined to the multiple combinations of the parameters' variations, the pedal would be capable of simulate non-considered aspects of other models and effects based on the Leslie.

Keywords: Electrical Engeneering. Digital Signal Processing. Leslie Cabinet. Digital Pedal.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

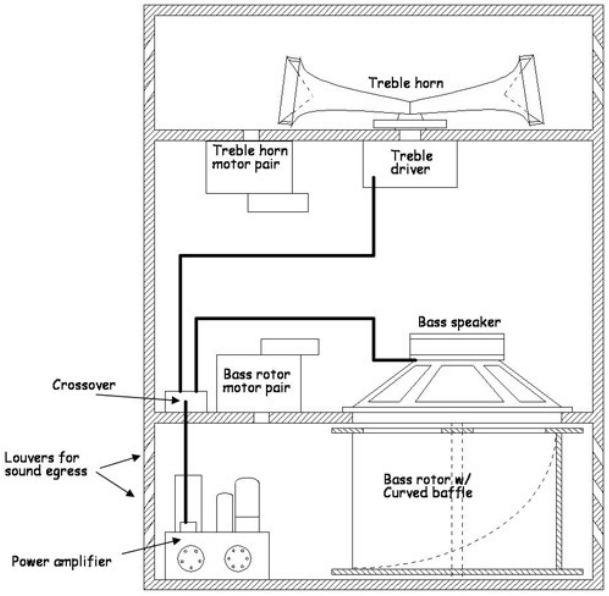
1.1 A caixa Leslie

A caixa Leslie é um equipamento de produção musical utilizada para amplificar e modificar o som do instrumento que está conectado a ela. Sua caraterística principal é a presença de alto-falantes rotatórios que, por conta disso, faz surgir dois efeitos no som que chega ao ouvinte: a variação da tonalidade e da intensidade do sinal. Sua construção interna está ilustrada na Figura 1. Ela se baseia em um alto-falante para as frequências agudas (ou chamado *compression driver*) acoplado a duas cornetas, um alto-falante para as frequências graves (também denominado *woofer*), um tambor (que se encontra abaixo do *woofer* com uma abertura superior para a entrada do som do *woofer* e uma abertura lateral para direcionar o som a medida que o tambor rotaciona), dois motores para a rotação do tambor e das cornetas, o circuito elétrico de amplificação (feito com válvulas termoiônicas) e filtragem do sinal, e um gabinete (ou caixa) de madeira que envolve todos os componentes e faz o som que sai dos alto-falantes refletir em seu interior. Um esquema simplificado desses componentes é mostrado na Figura 2. Além disso, há uma chave ou pedal externo que permite ao músico controlar a velocidade de rotação das cornetas e do tambor através da seleção entre os modos lento e rápido de rotação (também chamados modos *chorale* e *tremolo*, respectivamente) (THE THEATRE ORGAN, 2018; CULT JAZZ, 2018; STRYMON, 1981).

Figura 1 – Construção interna da caixa Leslie.

Fonte: CAPTAIN FOLDBACK'S, 2018.

Figura 2 – Esquemático dos componentes integrantes da caixa Leslie.



Fonte: North Suburban HAMMOND ORGAN Society, 2018.

Seu nome deriva de seu inventor, Donald Leslie, que iniciou seus trabalhos em 1930 na tentativa de obter um alto-falante capaz de melhor simular órgãos tubulares. Leslie fez sua primeira caixa em 1941, recebendo exposição nacional nos Estados Unidos, e tornando-se um sucesso na crítica e comércio. Entre os anos de 1960 e 1970 a caixa Leslie se tornou extremamente popular entre diversos artistas, sendo utilizada por bandas de grande renome como Deep Purple, The Beatles, The Beach Boys e Pink Floyd (THE THEATRE ORGAN, 1981). Grandes guitarristas que também usaram a Leslie foram, por exemplo, Mick Hayes, Slash, Eric Clapton, Frankie Sullivan e David Gilmour (HAMMOND USA, 2018).

A Leslie possui uma linha grande de modelos de caixas acústicas bastante diferentes umas das outras. Diversos modelos foram criados até chegarem nos atuais modelos 122 e 147 que ainda são produzidos pela Hammond. Os modelos mais antigos são o 21H, 22H, 30, 31H, 760, 825 e 830. Cada modelo se diferenciava dos outros pelas opções de velocidade dos alto-falantes, a divisão de canais de áudio, o uso ou não das válvulas termoiônicas, o material de revestimento da caixa, dentre outros aspectos (CULT JAZZ, 2018). A Figura 3 apresenta alguns dos modelos citados.

De forma simplificada, as etapas de funcionamento da caixa Leslie são: (1) o sinal de entrada é pré-amplificado e amplificado com válvulas termoiônicas (gerando uma leve distorção no sinal); (2) o sinal resultante é filtrado por dois filtros (um passa-baixas e outro passa-altas) com frequências de corte em torno de 800Hz - sendo que a parte do sinal com frequência acima de 800Hz é enviada para as cornetas, e a parte com frequência abaixo de 800Hz é enviada ao *woofer*; (3) o sinal é enviado aos alto-falantes. Devido ao

Figura 3 – Modelos de caixa Leslie.

(a) Modelo 122. (b) Modelo 760. (c) Modelo 830.

Fontes: 3a - J-ROY, 2018; 3b - CAVALLI MUSICA, 2018; 3c - CAPTAIN FOLDBACK'S, 2018.

movimento rotatório dos alto-falantes, o som que chega ao ouvinte terá tanto uma variação de altura (devido ao efeito Doppler, ou modulação em frequência) quanto uma variação na intensidade (efeito denominado como modulação em amplitude). Ou seja, a medida que os alto-falantes se aproximam ou se afastam do ouvinte, esse perceberá aumentos e diminuições tanto no tom do som quanto em sua intensidade.

Pela marcante variação da sonoridade dos instrumentos musicais provocada pela caixa Leslie, é natural que ela, ainda hoje, seja muito utilizada por vários músicos. Por vezes, surgiram implementações de pedais analógicos capazes de simular efeitos separados da caixa Leslie, como é o caso dos pedais de *phaser* e *vibrato* (que simula o desvio de frequência), do pedal de trêmolo (capaz de simular o efeito da variação em amplitude) e o pedal de distorção (que simula a distorção harmônica provocada pelas válvulas termoiônicas). Contudo, com a revolução das aplicações digitais, os novos músicos têm trocado os aproximados 100kgs da caixa Leslie, ou os diversos pedais analógicos necessários para a simular, por implementações digitais compactas dela. Diversos pedais digitais foram desenvolvidos com o propósito de simular a caixa Leslie na íntegra, cada um desses com suas peculiaridades na discretização da caixa. Alguns dos mais famosos pedais de simulação da Leslie são: Electro Harmonix Lester G (ELECTRO-HARMONIX, 2018), Strymon Lex Rotary (STRYMON, 2018), Neo Instruments Ventilator II (NEO INSTRUMENTS, 2018) e RT-20 da BOSS (BOSS, 2018). Na Figura 4 é mostrado o pedal RT-20 da BOSS, que é projetada para guitarristas e tecladistas, oferecendo ajustes de tempo de rotação, mixagem dos rotores de grave e agudo, efeito *overdrive*, e outros. Por ser um dos pedais mais reconhecidos pela fidelidade de simulação dos efeitos da caixa Leslie, ele será utilizado como referência nesse trabalho.

Figura 4 – Pedal RT-20 da BOSS.

Fonte: BOSS, 2018.

1.2 Objetivos e Metodologia

Esse trabalho consiste, portanto, na implementação digital da caixa Leslie modelo 122 em espaço livre microfonada com microfones à 180°. Para tanto, desenvolveu-se algoritmos capazes de simular os efeitos sonoros separados produzidos pela caixa Leslie, sendo eles: a distorção harmônica, a separação das frequências graves e agudas através do filtro de *crossover*, o efeito Doppler e a variação de intensidade sonora. Esses algoritmos foram implementados em um microcontrolador capaz de executar tais instruções com o uso de parâmetros variáveis definidos pelo usuário para variação da sonoridade dos efeitos. Os sinais de aquisição da guitarra e envio do microcontrolador foram adequados através de dois circuitos de filtragem na entrada e saída do sistema. Desenvolvendo-se, ao final, um protótipo de pedal digital embarcado e portátil.

Os algoritmos foram simulados no programa MATLAB 2012b e comparados com um sinal quantizado a 12bits e reduzido a ponto flutuante para verificar a portabilidade do sistema para um microcontrolador com tais especificações. Com a validação da portabilidade dos efeitos, esses foram convertidos para linguagem C++ e implementados dentro da placa de desenvolvimento Teensy 3.6.

Devido a indisponibilidade de uma caixa Leslie, o som obtido com o pedal foi comparado ao do pedal RT-20 da BOSS. Tal pedal é considerado referência no mercado em simulação da caixa Leslie e, portanto, foi utilizado como base de comparação qualitativa. Isso foi feito adquirindo-se os sinais de saída dos pedais no programa Audacity 2.1.3 e os analisando tanto no domínio do tempo, quanto no domínio da frequência para diferentes sinais de entrada e opções dos pedais.

2 Fundamentos Teóricos

Nessa seção serão apresentados e descritos detalhadamente os efeitos gerados pela caixa Leslie. Serão abordados aspectos físicos da caixa, assim como a discretização do sistema.

2.1 Características de funcionamento da caixa Leslie

Inicialmente, é importante salientar que a caixa Leslie foi concebida como um dispositivo para modificação de som. Assim, sua finalidade é ser parte integrante do instrumento musical (THE THEATRE ORGAN, 1981).

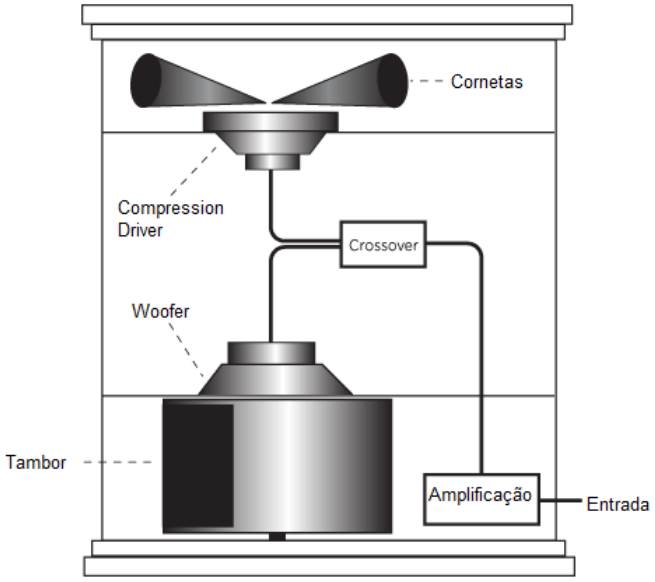
De forma geral, o funcionamento da caixa Leslie pode ser resumido como:

1. O sinal de entrada é pré-amplificado e amplificado por válvulas termiônicas;
2. O sinal resultante é filtrado por um filtro passa-baixas e outro passa-altas com frequências de corte em aproximadamente 800Hz;
3. Os sons agudos são enviados para as cornetas e os graves para o *woofer*;
4. Tanto as cornetas quanto o tambor giram e emitem o sinal sonoro resultante dentro da caixa de madeira que envolve todo o sistema físico da caixa Leslie.

A Figura 5 mostra um esquemático simplificado dos principais componentes no funcionamento da caixa Leslie, e a Figura 6 mostra a representação da rotação dos alto-falantes com identificação dos componentes, sendo: (1) cornetas, (2) *compression driver*, (3) motor para as cornetas, (4) filtragem do sinal de entrada, (5) motor para o tambor, (6) *woofer*, (7) abertura do tambor, (8) tambor, e (9) gabinete de madeira.

Devido ao uso de válvulas termoiônicas e suas não-linearidades, toda a amplificação da caixa cria distorção harmônica. Além disso, tal distorção também é atribuída as não-linearidades dos filtros usados. Contudo, o principal aspecto da caixa Leslie é o fato dela possuir alto-falantes rotatórios. Na maioria dos modelos, ela possui duas opções de velocidade para a rotação dos alto-falantes: *chorale* e *tremolo*, sendo o controle dessas velocidades feito através de uma chave ou pedal (dependendo do modelo) como ilustra a Figura 7. Devido a inércia do tambor ser maior que a das cornetas, há uma diferença nas velocidades de rotação dos mesmos. Enquanto que as cornetas giram entorno de 50rpm a 400rpm nos modos *chorale* e *tremolo*, respectivamente, o tambor gira entorno de 40rpm a 340rpm e em sentido oposto ao das cornetas (HARMONY CENTRAL, 2018). Um fato importante é que uma das cornetas é muda, ou seja, ela não emite nenhum som, e é

Figura 5 – Esquemático mecânico ilustrativo da caixa Leslie.



Fonte: modificado de STRYMON, 2011.

utilizada apenas para manter o momento de inércia do sistema e evitar instabilidades na rotação.

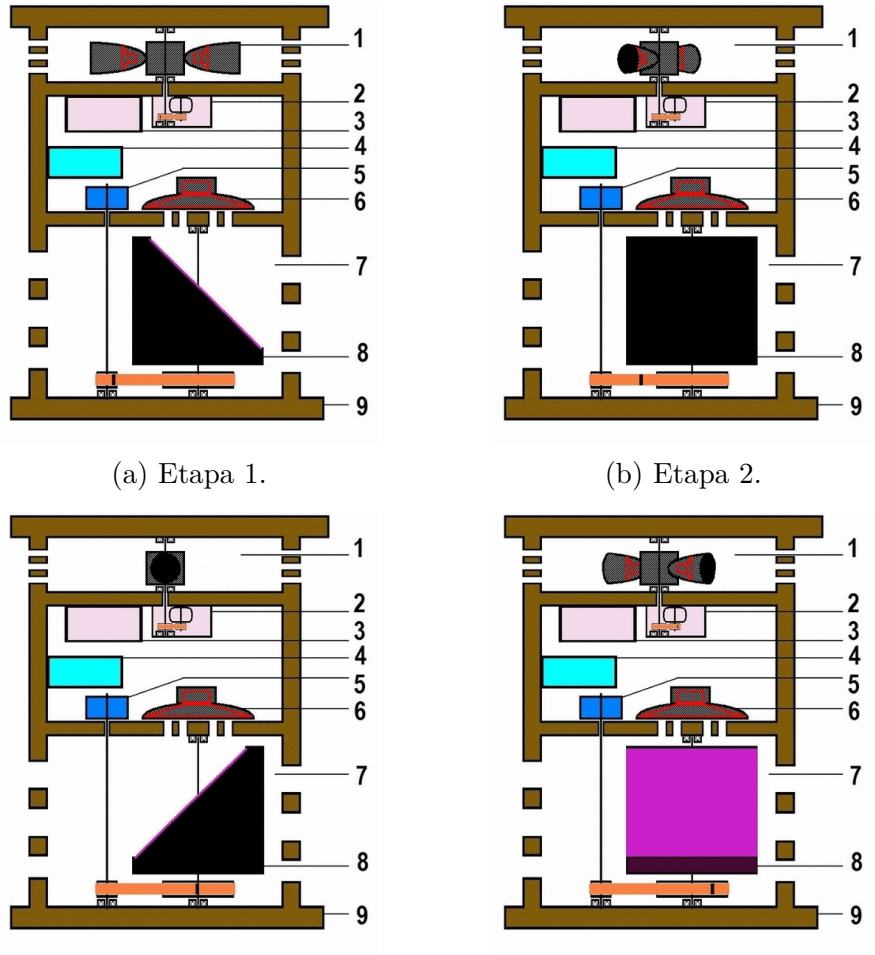
Por conta da rotação dos alto-falantes, a caixa Leslie pode ser considerada uma fonte sonora direcional com rotação em velocidade constante em torno de um ponto fixo. A intensidade do som irá aumentar a medida que a fonte rotatória se aproximar do ouvinte, e diminuir a medida que ela se afastar do mesmo - tendo seu máximo e mínimo quando o alto-falante estiver na direção do ouvinte (THE THEATRE ORGAN, 1981). O efeito resultante é uma modulação em amplitude (ou modulação AM), conhecido entre os músicos como o efeito trêmolo (ZÖLZER, 2011).

Além disso, a medida que o alto-falante rotaciona em direção ao ouvinte, o tom do sinal aumenta e, a medida que ele se move para longe, a tonalidade diminui. Esse efeito também é conhecido por efeito Doppler e é conhecido entre os músicos como efeito vibrato (THE THEATRE ORGAN, 1981; ZÖLZER, 2011). Qualitativamente, o efeito Doppler é mais evidenciado no som agudo emitido pelas cornetas, enquanto que, a modulação em amplitude é mais evidenciada no som grave emitido pelo *woofer*.

2.2 Modelo físico da caixa Leslie

Nessa seção será apresentada a modelagem física da caixa Leslie e seus efeitos. Para tanto, a Figura 8 apresenta o diagrama de blocos geral de funcionamento da caixa Leslie e seus efeitos descritos até então.

Figura 6 – Etapas de funcionamento da caixa Leslie para uma rotação dos alto-falantes.



(a) Etapa 1. (b) Etapa 2.

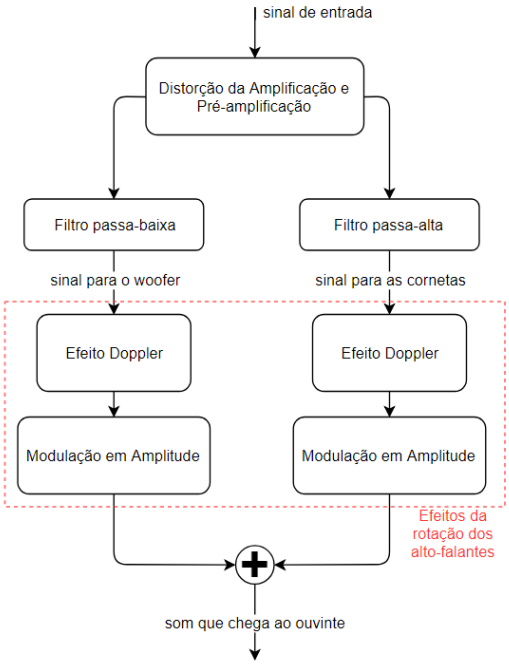
(c) Etapa 3. (d) Etapa 4.

Fonte: WIKIPEDIA, 2018.

Figura 7 – Chave de seleção do modo de velocidade.

Fonte: WIKIPEDIA, 2018.

Figura 8 – Diagrama de blocos geral dos efeitos da caixa Leslie sob o sinal de entrada.



Fonte: Própria, 2018.

2.2.1 Pré-amplificação e amplificação

O circuito responsável pela pré-amplificação e amplificação do sinal de entrada na caixa Leslie utiliza de válvulas termoiônicas (no caso, triodos e pentodos). As válvulas termoiônicas são dispositivos utilizados para amplificar sinais de tensão de baixa magnitude. Dispositivos com três terminais são denominados triodos e são usados principalmente em circuitos pré-amplificadores. Dispositivos com quatro ou cinco terminais (denominados tetrodos e pentodos, respectivamente) são usados principalmente na amplificação da potência de saída de um alto-falante. A Figura 9a apresenta exemplos reais de válvulas termoiônicas. Para sinais de baixa intensidade, o sinal de entrada é amplificado de forma linear pelas válvulas. Contudo, para sinais de alta intensidade, as válvulas tendem a saturar e, com isso, limitar o sinal de saída, gerando uma distorção não-linear (PAKARINEN e YEH, 2009).

As não-linearidades trazidas pelas válvulas, entretanto, não são vistas como introdução de ruído no sistema quando trata-se de termos musicais. Ao contrário do que se pensaria, as não-linearidades são capazes de acrescentar harmônicos muito sonoros. Há duas principais formas de distorção geradas pelas válvulas: distorção simétrica e assimétrica. O tipo de válvula utilizada para a amplificação do sinal é o que determina se o sinal sofrerá distorção simétrica ou assimétrica, sendo que, de forma geral, os pentodos introduzem distorção simétrica, enquanto que, os triodos introduzem distorção assimétrica. Para ilustrar os efeitos de cada uma dessas distorções sobre um sinal senoidal, veja a

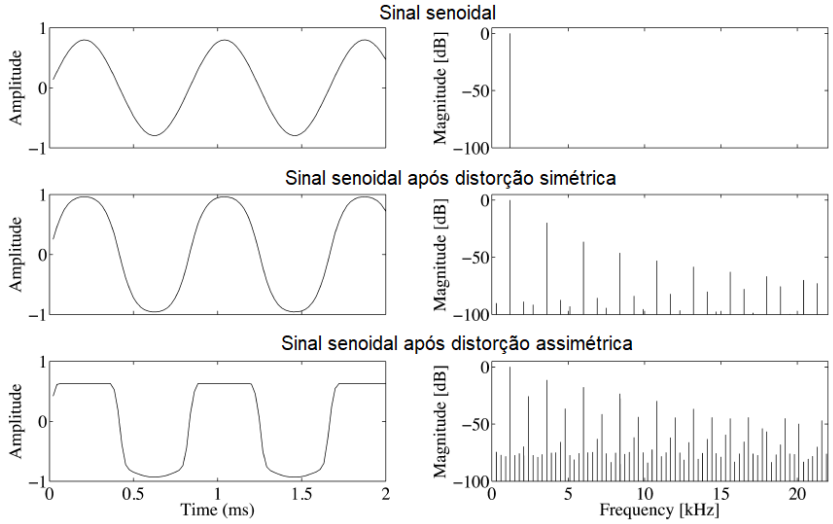
Figura 10. Através dessa figura é possível perceber que a distorção simétrica introduz, em sua maioria, harmônicos ímpares; enquanto que, a distorção assimétrica introduz tanto harmônicos ímpares, quanto harmônicos pares.

Figura 9 – Exemplos de válvulas termoiônicas.

(a) Exemplos reais de válvulas.

Fonte: THE NATIONAL MUSEUM OF COMPUTING, 2018.

Figura 10 – Efeitos da distorção simétrica e assimétrica sobre uma senoide.



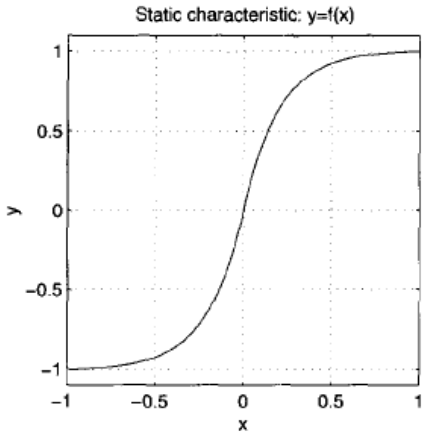
Fonte: modificado de PAKARINEN, 2009.

Diversos autores focaram seus esforços em obter funções capazes de melhor descrever os efeitos das válvulas. Apesar de as funções propostas serem diferentes, a maioria deles concordam que a melhor classe de funções que simulam as válvulas são as funções sigmoides. Dempwolf, Holters e Zölzer (2010) e Yeh (2008), por exemplo, propõem o uso da sigmoide arco seno hiperbólico; enquanto que, Bendiksen (1997) propõe uma função baseada na exponencial capaz de simular a distorção simétrica. Tal função é descrita por:

$$y(x) = \frac{x}{|x|}\left(1 - e^{dist \cdot x^2/|x|}\right) , \tag{1}$$

em que *dist* é o nível de distorção. Tal equação faz com que, para níveis baixos de sinal, o sistema se comporte quase linearmente; enquanto que, para níveis altos de sinal, o sistema limite a saída. A relação entrada e saída de tal sistema é ilustrada na Figura 11.

Figura 11 – Relação entrada e saída para distorção simétrica com $dist = 1$.



Fonte: modificado de ZÖLZER, 2011.

2.2.2 Filtros de *crossover*

O circuito responsável por separar os sons agudos dos graves e, assim, fazer as duas filtragens do sinal de entrada é apresentado na Figura 12. Um parâmetro importante na análise dos filtros que não é evidenciado na Figura 12 é o valor da impedância dos alto-falantes, que, para ambos, é de 16Ω. O filtro passa-baixas é conectado em série com o *woofer* e o filtro passa-altas é conectado em série com o *compression driver*. Tal topologia de circuito é chamada de rede dividida (ou *dividing network*) e é aplicada em sistemas acoplados, pois esse arranjo faz com que as frequências baixas sejam enviadas para um alto-falante de baixas frequências (o *woofer*), enquanto que as altas frequências sejam enviadas para um alto-falante de altas frequências (as cornetas). A frequência cuja potência enviada para os dois alto-falantes é igual é chamada de frequência de *crossover* - para a caixa Leslie, essa frequência é de 800Hz (TERMAN, 1943).

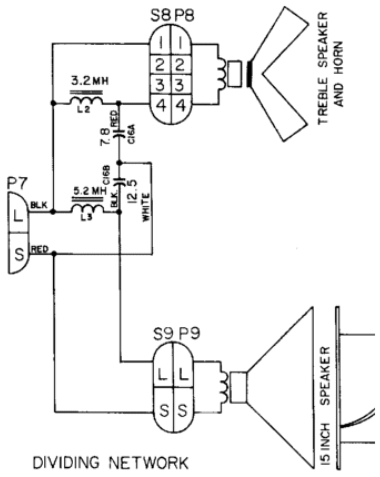
Para melhor entendimento e análise dos dois filtros da Figura 12, o circuito é redesenhado na Figura 13. Nessa figura, R_o representa a impedância dos alto-falantes. Dessa forma, em relação as entradas e saídas de sinal nos filtros da Figura 13, é possível estabelecer a seguinte função de transferência para o filtro passa-altas:

$$H_a(s) = \frac{s^2}{s^2 + \frac{s}{C_1 R_o} + \frac{1}{L_1 C_1}}\,, \tag{2}$$

e para o filtro passa-baixas:

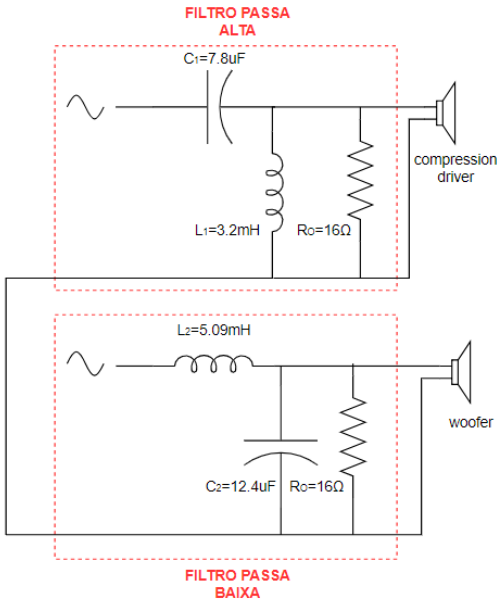
$$H_b(s) = \frac{1}{s^2 L_2 C_2 + s\frac{L_2}{R_o} + 1}\,. \tag{3}$$

Figura 12 – Circuitos de filtragem (*Dividing Network*) na íntegra.



Fonte: modificado de BENTON ELECTRONICS, 2018.

Figura 13 – Circuitos de filtragem simplificados.



Fonte: Própria, 2018

Note que ambos os filtros são de segunda ordem, o que já era esperado devido ao fato da filtragem ser feita por um capacitor e um indutor nos dois circuitos.

2.2.3 Efeito Doppler ou modulação em frequência

O Efeito Doppler, também referido como modulação em frequência, faz com que o tom de uma fonte sonora pareça estar aumentando e/ou diminuindo devido ao movimento da fonte e/ou do ouvinte em relação um ao outro (SMITH, 2002).

De acordo com Zölzer (2011), a modulação em frequência de um sinal de áudio se refere à modificação direta da frequência do sinal de áudio pelo controle do sinal modulador $m(t)$. O sistema capaz de efetuar a modulação em frequência é dado por

$$h(t) = \delta(t - \phi(t)) \,, \tag{4}$$

em que h é a resposta ao impulso do sistema, δ é a função delta de Dirac e ϕ é a função que faz a variação do tempo. A função ϕ é definida em relação ao sinal modulador $m(t)$ na forma

$$\phi(t) = 2\pi \cdot k_{FM} \cdot \int_{-\infty}^{t} m(\tau) \, \mathrm{d}\tau \,, \tag{5}$$

com k_{FM} sendo uma constante da modulação. Tal equação pode ser reescrita em relação ao sinal modulador $m(t)$ da seguinte forma (considerando $k_{FM} = \frac{1}{2\pi}$ sem perda de generalidade):

$$m(t) = \dot{\phi}(t) \,. \tag{6}$$

Tem-se, portanto, que o sinal modulado em frequência será dado por

$$y(t) = x(t) * h(t) = x(t - \phi(t)) \,. \tag{7}$$

De acordo com Smith (2002), o sinal de saída provocado por uma entrada do tipo senoidal com frequência angular ω_s na forma

$$x(t) = e^{j\omega_s t} \tag{8}$$

no sistema definido pela equação 7 é

$$y(t) = x(t - \phi(t)) = e^{j\omega_s \cdot (t - \phi(t))} \,. \tag{9}$$

O valor da fase instantânea do sinal é definido como

$$\theta(t) = \angle y(t) = \omega_s \cdot (t - \phi(t)) \,. \tag{10}$$

Tal equação pode ser diferenciada, resultando na frequência instantânea, ω_l, descrita por

$$\omega_l = \omega_s \cdot [t - \dot{\phi}(t)] \,. \tag{11}$$

A frequência instantânea trata-se, portanto, da frequência do som que chega ao ouvinte. Dessa forma, a taxa de crescimento ou decrescimento do atraso relativo à frequência será

$$\dot{\phi}(t) = \frac{\omega_s - \omega_l}{\omega_s} \,. \tag{12}$$

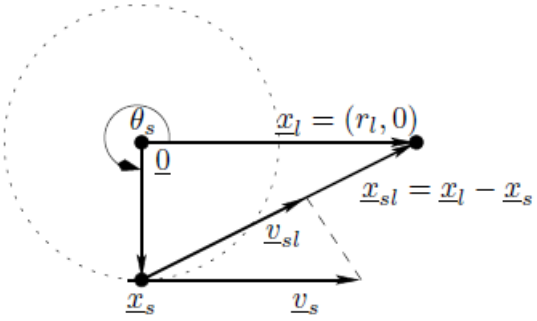
Para se chegar ao efeito Doppler gerado pela caixa Leslie, parte-se da formulação física da frequência que o ouvinte escuta para uma fonte e/ou ouvinte em movimento que é descrita por:

$$\omega_l = \omega_s \cdot \frac{1 + \frac{v_{ls}}{c}}{1 - \frac{v_{sl}}{c}} \,, \tag{13}$$

em que v_{ls} é a velocidade linear do ouvinte relativa ao meio de propagação em direção a fonte, v_{sl} é a velocidade linear da fonte relativa ao meio de propagação em direção ao ouvinte, e c é a velocidade do som no meio. Note que todas as quantidades na equação 13 são escalares.

Para se determinar as velocidades v_{ls} e v_{sl} utiliza-se a Figura 14 como base, tendo em vista que $\vec{x_l}$ é o vetor posição do ouvinte (que vai do centro de rotação do alto-falante ao ouvinte), $\vec{x_s}$ é o vetor posição da fonte (que vai do centro de rotação do alto-falante ao alto-falante - fonte), $\vec{x_{sl}}$ é o vetor posição da fonte ao ouvinte, $\vec{v_s}$ é o vetor velocidade linear da fonte, $\vec{v_{sl}}$ é o vetor velocidade linear relativa entre a fonte e o ouvinte, e r_l é a distância do centro de rotação ao ouvinte.

Figura 14 – Geometria do movimento de rotação de um alto falante em relação ao ouvinte.



Fonte: SMITH, 2002

Conforme a Figura 14, pode-se quantificar $\vec{x_s}$ e $\vec{x_l}$ como

$$\vec{x_s}(t) = \begin{bmatrix} r_s \cos(\omega_m t) \\ r_s \operatorname{sen}(\omega_m t) \end{bmatrix}, \tag{14}$$

onde r_s é o raio de rotação do alto-falante e ω_m é a velocidade angular de rotação do mesmo. E $\vec{x_l}$ como

$$\vec{x_l}(t) = \begin{bmatrix} r_l \\ 0 \end{bmatrix}. \tag{15}$$

Supondo que o ouvinte está em repouso, $\vec{v_l}$ é nulo. O vetor velocidade da fonte é

$$\vec{v_s}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\vec{x_s}(t) = \begin{bmatrix} -r_s\omega_m \operatorname{sen}(\omega_m t) \\ r_s\omega_m \cos(\omega_m t) \end{bmatrix}. \tag{16}$$

A velocidade relativa entre a fonte e o ouvinte, $\vec{v_{sl}}$, pode ser determinada ao se projetar $\vec{v_s}$ sob $\vec{x_s}$, que, por serem ortogonais, se simplifica em:

$$\vec{v_{sl}} = \frac{\vec{v_s} \cdot \vec{x_l}}{||\vec{x_l} - \vec{x_s}||^2}(\vec{x_l} - \vec{x_s}). \tag{17}$$

Substituindo as equações 14, 15 e 16 em 17 obtém-se que:

$$\vec{v_{ls}}(t) = \frac{-r_l r_s \omega_m \operatorname{sen}(\omega_m t)}{r_l^2 + 2 r_l r_s \cos(\omega_m t) + r_s^2} \begin{bmatrix} r_l - r_s \cos(\omega_m t) \\ -r_s \operatorname{sen}(\omega_m t) \end{bmatrix} . \tag{18}$$

Considerando que a distância do centro de rotação ao ouvinte é muito maior que a da fonte (ou seja, $r_l \gg r_s$) a equação acima se reduz a

$$\vec{v_{sl}} \approx -r_s \omega_m \operatorname{sen}(\omega_m t) \begin{bmatrix} 1 \\ 0 \end{bmatrix} . \tag{19}$$

Por fim, substituindo a equação 19 em 13 - lembrando que o ouvinte está em repouso (v_l nulo), obtém-se que o desvio de frequência provocado pelo efeito Doppler da caixa Leslie é dado por

$$\omega_l = \frac{\omega_s}{1 + r_s \omega_m \operatorname{sen}(\omega_m t)/c} \approx \omega_s \left[1 - \frac{r_s \omega_m}{c} \operatorname{sen}(\omega_m t) \right] . \tag{20}$$

Essa aproximação é válida para pequenas variações de frequência provocadas pelo efeito Doppler com os alto-falantes rotacionando em espaço livre. Assim, para um ouvinte distante, os alto-falantes causam, aproximadamente, uma variação senoidal na frequência do som emitido (SMITH, 2002). Precisa-se, agora, determinar a forma do sinal modulador, $m(t)$.

Substituindo a equação 20 em 12, obtém-se que o sinal modulador é:

$$\dot{\phi}(t) = m(t) = \frac{r_s \omega_m}{c} \operatorname{sen}(\omega_m t) . \tag{21}$$

Conclui-se, assim, que a função moduladora para o efeito Doppler também varia senoidalmente ao longo do tempo, sendo proporcional ao raio de rotação - r_s, e a velocidade angular de rotação do alto-falante - ω_m, e a velocidade do som no meio - c. Integrando-se a equação 21, chega-se a expressão da função ϕ (responsável pelo atraso no sinal de entrada). Assim, ϕ é dada por

$$\phi(t) = K - \frac{r_s}{c} \cos(\omega_m t) , \tag{22}$$

com K sendo uma constante de integração, determinada ao se supor que não há desvio de frequência no sinal $x(t)$ no instante inicial $t = 0$. Dessa forma

$$\phi(0) = 0 = K - \frac{r_s}{c} \cos(\omega_m \cdot 0) . \tag{23}$$

E a constante K é

$$K = \frac{r_s}{c} . \tag{24}$$

Portanto, a função do atraso para o sinal de entrada é expresso por

$$\phi(t) = \frac{r_s}{c} \cdot [1 - \cos(\omega_m t)] . \tag{25}$$

Note que o argumento r_s/c determina um certo atraso temporal constante, fazendo com que a expressão 25 possa ser reescrita na forma

$$\phi(t) = t_a \cdot [1 - \cos(\omega_m t)]\,, \tag{26}$$

em que t_a representa o tempo de atraso do efeito Doppler dado por

$$t_a = \frac{r_s}{c}\,. \tag{27}$$

O valor de t_a é responsável por aumentar ou diminuir a intensidade do desvio de frequência e possui valores entre 0.2ms e 1ms, conforme Anderton (1985).

2.2.4 Modulação em amplitude

A modulação em amplitude (ou modulação AM) faz com que o sinal varie sua intensidade sonora ao longo do tempo. Um sinal $x(t)$ modulado em amplitude por uma moduladora $m(t)$ é descrito como:

$$y(t) = [1 + \alpha m(t)] \cdot x(t)\,, \tag{28}$$

em que α é uma grandeza adimensional que representa o coeficiente de modulação, definido para valores entre 0 e 1, sendo que, para $\alpha = 0$ não se terá modulação em amplitude, enquanto que, para $\alpha = 1$ a modulação em amplitude será máxima (ZÖLZER, 2011). Para se chegar a modulação AM provocada pela caixa Leslie ao sinal que chega ao ouvinte, parte-se da definição de intensidade sonora dada por

$$I = \frac{P_s}{A}\,, \tag{29}$$

onde P_s é a potência sonora emitida pela fonte e A é a área da esfera que tem seu centro na posição do alto-falante e raio até a posição do ouvinte. O raio dessa esfera é dado por

$$r = \sqrt{||\vec{x_s}(t) - \vec{x_l}(t)||} = \sqrt{[r_l - r_s\cos(\omega_m t)]^2 + r_s^2\,\mathrm{sen}^2(\omega_m t)}\,, \tag{30}$$

sendo $\vec{x_s}$, $\vec{x_l}$, r_l, r_s e ω_m as mesmas grandezas definidas na seção 2.2.3 e descritas na Figura 14. Isto posto, supondo que os alto-falantes são emissores sonoros pontuais de potência constante, a intensidade sonora recebida pode ser descrita como

$$I = \frac{P_s}{4\pi r^2}\,. \tag{31}$$

Substituindo-se a equação 30 em 31, tem-se que:

$$I = \frac{k}{r_l^2 + r_s^2\cos^2(\omega_m t) - 2r_l r_s\cos(\omega_m t) + r_s^2\,\mathrm{sen}^2(\omega_m t)}\,, \tag{32}$$

em que $k = P_s/4\pi$. Simplificando e organizando os termos da equação acima, obtém-se que a intensidade sonora é dada por

$$I = \frac{k}{r_l^2} \cdot \frac{1}{1 + (\frac{r_s}{r_l})^2 - 2\frac{r_s}{r_l}\cos(\omega_m t)}\,. \tag{33}$$

Supondo que a distância do ouvinte ao centro de rotação é muito maior que a distância do alto-falante em relação ao centro de rotação, ter-se-á que $r_s/r_l \approx 0$. Aproximando-se o termo que multiplica k/r_l^2 na equação 33 por uma expansão em série de Taylor em torno de $r_s/r_l = 0$, tem-se que:

$$\frac{1}{1 + (\frac{r_s}{r_l})^2 - 2\frac{r_s}{r_l}\cos(\omega_m t)} \approx 1 + 2\frac{r_s}{r_l}\cos(\omega_m t) + \left(\frac{r_s}{r_l}\right)^2 \cdot (2\cos(\omega_m t) + 1) + \dots . \tag{34}$$

Eliminando os termos de ordem acima de 2 (pois seu valor será muito pequeno) na equação 34, e substituindo em 33, obtém-se que a intensidade sonora é aproximadamente

$$I \approx \frac{P_s}{4\pi r_l^2} \cdot \left[1 + 2\frac{r_s}{r_l}\cos(\omega_m t)\right] . \tag{35}$$

Através da equação 35 é possível perceber que a intensidade sonora que chega ao ouvinte varia senoidalmente ao longo do tempo e depende da relação entre o raio de rotação da fonte e a distância do ouvinte ao centro de rotação (r_s/r_l). Tal conclusão indica que o sinal da moduladora, $m(t)$, deverá também variar senoidalmente com frequência angular igual ao de rotação do alto-falante, sendo descrito por:

$$m(t) = \cos(\omega_m t) . \tag{36}$$

E o coeficiente de modulação será dado por

$$\alpha = \frac{2r_s}{r_l} . \tag{37}$$

Outra conclusão importante se dá ao comparar as equações 21 e 35, pois é possível perceber que o máximo desvio de frequência do sinal estará defasado em 90° da máxima intensidade do sinal que o ouvinte recebe. Tal conclusão também é descrita em Zölzer (2011) e está ilustrada na Figura 15.

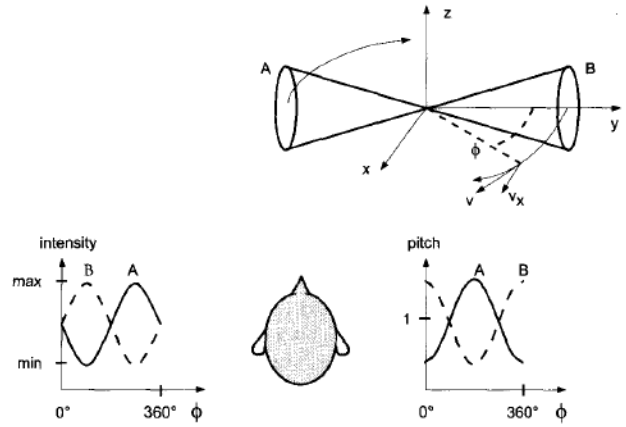
2.2.5 Microfonação

A microfonoção do sistema refere-se a forma com que a captação do sinal de saída de uma caixa Leslie real seria feita. A forma mais utilizada é usando dois microfones posicionados à 180° um do outro. Isso é feito justamente para que haja o contraste dos picos e vales do efeito Doppler e da modulação em amplitude ocorrendo em cada um dos microfones e, dessa forma, o ouvinte tenha a sensação que o som está "girando".

2.3 Discretização do sistema

Nessa seção serão apresentadas as técnicas de discretizações dos efeitos descritos pela caracterização física da caixa Leslie. Para tanto, inicialmente, apresenta-se uma breve introdução aos sinais discretos e, sequencialmente, a discretização dos efeitos.

Figura 15 – Representação da defasagem de 90° entre o máximo desvio de frequência e a máxima intensidade do sinal que chega ao ouvinte.



Fonte: ZÖLZER, 2011.

2.3.1 Sinais discretos

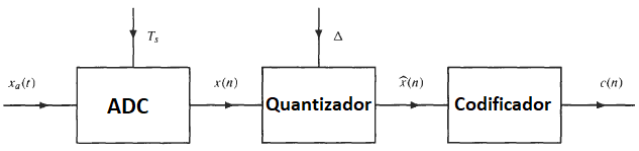
De acordo com Hayes (1998), um sinal discreto corresponde a uma sequência indexada de números reais ou complexos. Por conta disso, um sinal discreto é função de uma variável com valor inteiro, n, e pode ser denotado na forma $x[n]$.

Para se obter a discretização de um sinal contínuo (ou analógico), $x_a(t)$, é necessário que esse seja amostrado através de um ADC (conversor analógico/digital - *analog-digital converter*). O processo da amostragem refere-se a: (1) conversão analógica/digital (A/D), (2) quantização do sinal amostrado e (3) codificação do sinal quantizado. A Figura 16 ilustra os componentes do conversor ADC.

A conversão analógica/digital amostra valores do sinal analógico a intervalos constantes de tempo, denominado período de amostragem, T_s - seu inverso é a frequência de amostragem ($F_s = 1/T_s$). Dessa forma, o sinal amostrado, $x[n]$, se relaciona com o sinal contínuo, $x_a(t)$, que o originou na forma:

$$x[n] = x_a(nT_s)\,. \tag{38}$$

Figura 16 – Componentes do conversor ADC relacionando as etapas da conversão do sinal contínuo $x_a(t)$ no sinal discreto $c(n)$.



Fonte: modificado de HAYES, 1998.

Pela formulação acima, é possível descrever o sinal amostrado, $x_s(t)$, na forma:

$$x_s(t) = \sum_{n=-\infty}^{\infty} x_a(nT_s) \cdot \delta(t - nT_s)\,, \tag{39}$$

em que δ representa a função delta de Dirac. Ou seja, o sinal amostrado corresponde ao sinal contínuo original multiplicado por um trem de impulsos que forçam o valor do sinal amostrado a ser diferente de zero apenas nos valores múltiplos inteiros de T_s. A transformada contínua de Fourier do sinal $x_s(t)$ é

$$X_s(j\Omega) = \sum_{n=-\infty}^{\infty} x_a(nT_s) \cdot e^{-jn\Omega T_s} = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} X_a(j\Omega - jk\Omega_s)\,, \tag{40}$$

onde Ω é a frequência angular contínua, Ω_s é a frequência de amostragem angular ($\Omega_s = 2\pi \cdot F_s$) e X_a é o sinal da transformada contínua de Fourier de $x_a(t)$. Verifica-se, assim, que o sinal amostrado no domínio da frequência será formado por k repetições centradas em Ω_s do sinal $X_a(j\Omega)$. A transformada discreta de Fourier do sinal $x[n]$ é dada por

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] \cdot e^{-jn\omega} = \sum_{n=-\infty}^{\infty} x_a(nT_s) \cdot e^{-jn\omega}\,, \tag{41}$$

sendo ω a frequência angular discreta. Comparando as equações 40 e 41 obtém-se que o sinal discreto no domínio da frequência é dado por

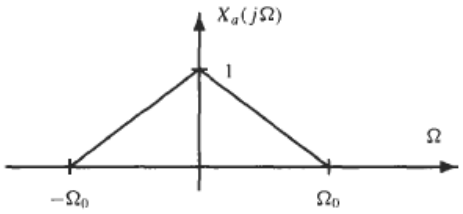
$$X(e^{j\omega}) = X_s(j\Omega)|_{\Omega=\omega/T_s} = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} X_a\left(j\frac{\omega}{T_s} - j\frac{2\pi k}{T_s}\right)\,. \tag{42}$$

Assim, $X(e^{j\omega})$ é versão escalada de $X_s(j\Omega)$, com escala dada por: $\omega = \Omega T_s$; e possui atenuação proporcional ao período de amostragem, T_s. Além disso, o sinal $X(e^{j\omega})$ é periódico com período 2π. A Figura 17 mostra, então, os efeitos da amostragem de um sinal contínuo com largura de banda limitada, a fim de esclarecer os conceitos apresentados.

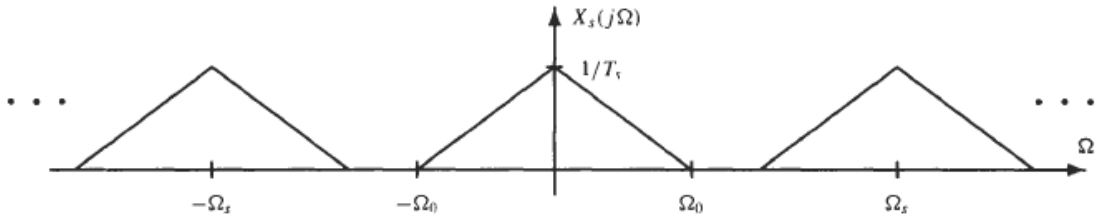
É possível observar, pela análise da Figura 17c, que, quando a frequência de amostragem é menor que duas vezes a maior componente de frequência do sinal contínuo a ser amostrado, ocorre a superposição de frequências. Esse fenômeno é chamado *aliasing* e deve ser evitado ao máximo com o uso de filtros passa-baixas, pois ele modifica severamente o sinal original adicionando ruído irremovível no sistema digital. Por conta disso, a frequência de amostragem deve ser sempre maior que o dobro da maior componente de frequência do sinal a ser amostrado. Tal condição é chamada critério de Nyquist, ou teorema da amostragem.

Dando continuidade as etapas da amostragem feita por um ADC, Oppenheim (1999) descreve que a quantificação trata-se de um sistema não linear cujo propósito é transformar o sinal amostrado de entrada em um dos valores finitos possíveis dentro da memória do sistema que armazenará a informação. Quantificadores podem ser definidos com níveis de quantização uniformemente ou não uniformemente espaçados; contudo,

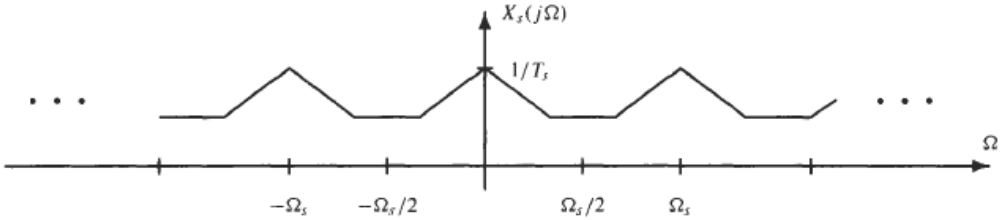
Figura 17 – Efeitos da amostragem de um sinal contínuo, $x_a(t)$, com largura de banda limitada.



(a) Sinal contínuo $x_a(t)$ com largura de banda limitada no domínio da frequência.



(b) Sinal de amostragem, $x_s(t)$, de $x_a(t)$ no domínio da frequência para frequência de amostragem $\Omega_s \geq 2\Omega_o$.



(c) Sinal de amostragem, $x_s(t)$, de $x_a(t)$ no domínio da frequência para frequência de amostragem $\Omega_s < 2\Omega_o$.

Fonte: HAYES, 1998.

quando serão feitos cálculos numéricos sob as amostras, o quantificador usualmente é de passos uniformes.

Uma característica importante do quantificador é o tamanho do seu passo, ou seja, a distância entre dois valores amostrados que geram um mesmo valor de sinal quantizado. O tamanho do passo de um quantificador é calculado como:

$$\Delta = \frac{X_{max}}{2^B}\,, \tag{43}$$

em que X_{max} é o valor máximo de entrada do ADC e B é o número de bits utilizado na quantização. Como os valores amostrados são restringidos a uma faixa de valores finitos, a quantização acaba retornando um sinal com certo erro em relação ao sinal amostrado. Tal erro é dado por:

$$e[n] = x_Q[n] - x[n]\,, \tag{44}$$

sendo $x_Q[n]$ o sinal quantizado e $x[n]$ o sinal amostrado. O valor desse erro para um

quantificador de $(B+1)$ bits com Δ dado pela equação 43 satisfaz

$$-\frac{\Delta}{2} \leq e[n] \leq \frac{\Delta}{2}\,, \tag{45}$$

desde que o sinal amostrado satisfaça

$$-\left(X_{max} + \frac{\Delta}{2}\right) < x[n] \leq \left(X_{max} - \frac{\Delta}{2}\right)\,. \tag{46}$$

Caso o sinal amostrado possua valores fora da escala de valores da equação acima, o erro de quantização pode ser maior em magnitude do que o expresso pela equação 45.

Por fim, o codificador é o sistema que converte o sinal elétrico de saída do quantizador no valor correspondente em bits a ser armazenado (HAYES, 1998). Tendo abordado esses aspectos iniciais sobre sistemas discretos, procede-se às técnicas de discretização dos efeitos da caixa Leslie.

2.3.2 Filtros de *crossover*

Há diversas técnicas para se converter um filtro analógico para o domínio digital. A técnica escolhida nesse trabalho foi o uso da Transformada Bilinear. Essa técnica refere-se a uma transformação entre as variáveis s e z que mapeia todo o eixo $j\Omega$ no plano s para uma revolução do círculo unitário no plano z (OPPENHEIM, 1999). A conversão do domínio s para o domínio z é descrito pela equação

$$s = 2F_s \cdot \frac{z-1}{z+1}\,, \tag{47}$$

em que F_s é a frequência que o sinal é amostrado. Por conta dessa transformação não ser linear, a relação de transformação entre a frequência analógica $j\Omega$ (que compreende valores entre $-\infty$ a ∞) e a frequência digital ω (que compreende valores entre $-\pi$ a π) é definida por:

$$\omega = 2 \cdot \tan^{-1}\left(\frac{\Omega}{2F_s}\right)\,. \tag{48}$$

Para muitas aplicações é interessante que o valor da função de transferência de um filtro para uma determinada frequência no domínio analógico seja igual ao valor no domínio discreto. Para isso, a transformada bilinear pode ser descrita através de uma frequência F_c de pré-deformação (*prewarp*). Com esse intuito, a relação entre as variáveis s e z torna-se:

$$s = \frac{2\pi F_c}{\tan(\pi F_c/F_s)} \cdot \frac{z-1}{z+1}\,. \tag{49}$$

E a relação entre as frequências fica sendo descrita por

$$\omega = 2 \cdot \tan^{-1}\left(\frac{\Omega \cdot \tan(\pi F_c/F_s)}{2F_s}\right)\,. \tag{50}$$

Para o presente contexto, é interessante que o valor da função de transferência na frequência de *crossover* dos filtros digitais seja a mesma do domínio analógico (para se garantir que o valor de atenuação dos dois filtros nessa frequência seja aproximadamente a mesma). Assim, através da relação da equação 49, é possível discretizar a função de transferência do filtro analógico passa-altas apresentado na equação 2 na forma

$$H_a[z] = K_f^2 \cdot \frac{1 - 2z^{-1} + z^{-2}}{\left(K_f^2 + \frac{K_f}{C_1 R_o} + \frac{1}{L_1 C_1}\right) + \left(\frac{2}{L_1 C_1} - 2K_f^2\right) z^{-1} + \left(K_f^2 - \frac{K_f}{C_1 R_o} + \frac{1}{L_1 C_1}\right) z^{-2}}\,, \quad (51)$$

onde $K_f = \frac{2\pi F_c}{\tan(\pi F_c / F_s)}$. E o filtro passa-baixas apresentado na equação 3 é discretizado como

$$H_b[z] = \frac{1 + 2z^{-1} + z^{-2}}{L_2(K_f^2 C_2 + K_f/R_o) + (2 - 2K_f^2 L_2 C_2) z^{-1} + (K_f^2 L_2 C_2 - K_f L_2/R_o + 1) z^{-2}}\,. \quad (52)$$

Através da análise das funções de transferência discretas é possível concluir que tratam-se de filtros do tipo IIR (*infinite impulse response*). Ou seja, tomando-se a transformada inversa dos filtros digitais, os mesmos acabam por satisfazer a seguinte equação de diferenças:

$$y[n] - \sum_{k=1}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]\,, \quad (53)$$

em que, para os filtros das equações 51 e 52, $N = M = 2$, e os coeficientes a_k e b_k são dados respectivamente pelos valores do denominador e numerador das funções de transferência.

2.3.3 Efeito Doppler ou modulação em frequência

O sistema discreto capaz de fazer a modulação em frequência de um sinal discreto $x[n]$ é análogo àquele apresentado em 2.2.3, sendo, dessa forma, descrito por

$$h[n] = \delta[n - \phi[n]]\,, \quad (54)$$

em que $\phi[n]$ é o sinal discretizado da equação 26. Sua discretização resulta em

$$\phi[n] = D_t \cdot [1 - \cos(\omega_m n T_s)]\,, \quad (55)$$

onde D_t representa o número de amostras atrasadas para determinado tempo de atraso t_a (definido na equação 27), sendo

$$D_t = t_a \cdot F_s\,. \quad (56)$$

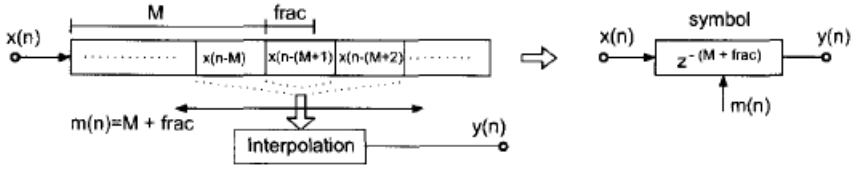
O sinal discreto modulado em frequência pode ser escrito, portanto, como

$$y[n] = x[n] * h[n] = x[n - \phi[n]]\,. \quad (57)$$

Como $\phi[n]$ não resulta apenas em valores inteiros para cada instante de tempo discreto n, tal tipo de atraso no sinal deve ser decomposto em sua parte inteira e fracionária. A parte inteira é implementada por uma série de M unidades de atrasos, enquanto que a parte fracionária é aproximada por algum filtro de interpolação - como interpolação linear, *spline* ou *allpass* (ZÖLZER, 2011). Tal técnica de atraso no sistema é chamada Atraso Fracionário (ou *Fractional Delay*). A representação em diagramas de bloco de seu funcionamento, juntamente à simbologia usada para descrição de tal tipo de atraso, são mostradas na Figura 18. Usando a técnica da interpolação linear, sendo M a parte inteira de $\phi[n]$ e f a parte fracionária, o sinal $y[n]$ oriundo do efeito Doppler será obtido na forma

$$y[n] = x[n - (M + 1)] \cdot f + x[n - M] \cdot (1 - f)\,. \tag{58}$$

Figura 18 – Representação em diagrama de blocos do atraso fracionário.



Fonte: ZÖLZER, 2011

2.3.4 Modulação em amplitude

O sistema capaz de efetuar a modulação em amplitude de um sinal discreto $x[n]$ é análogo àquele apresentado em 2.2.4 e é descrito por:

$$y[n] = (1 + \alpha m[n]) \cdot x[n]\,, \tag{59}$$

onde o valor de α é dado pela equação 37, e o sinal modulador, $m[n]$, é dado pela discretização da equação 36 como:

$$m[n] = \cos(\omega_m n / F_s)\,. \tag{60}$$

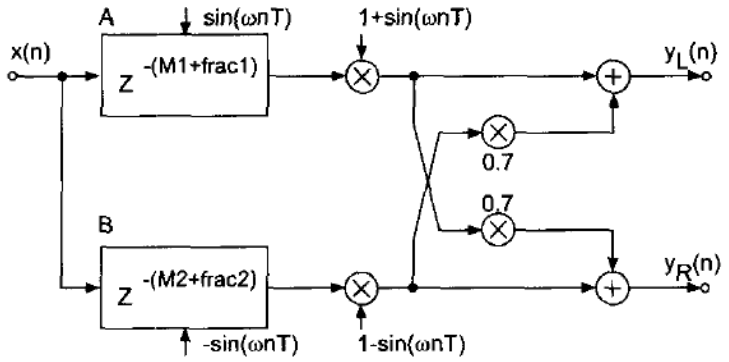
O sinal de saída da modulação em amplitude é dado, então, por

$$y[n] = (1 + \alpha \cos(\omega_m n T_s)) \cdot x[n]\,. \tag{61}$$

2.3.5 Microfonação

Utilizando-se a ideia de dois microfones posicionados à 180° um do outro obtém-se a saída estéreo do sistema de simulação da caixa Leslie. Para tal implementação, utilizou-se como base o sistema proposto em (ZÖLZER, 2011) e mostrado na Figura 19. Note que

Figura 19 – Diagrama de blocos parcial do sistema para a saída estéreo.



Fonte: ZÖLZER, 2011.

apenas a parte do efeito Doppler e da modulação em amplitude são evidenciados, pois apenas esses efeitos são alterados pelo uso da saída estéreo.

A diferença imposta pela posição dos microfones está justamente na fase dos sinais moduladores. Ou seja, esses se apresentam com sinal trocado em cada canal de saída. Há também a presença do trecho que "mistura" os canais por um fator de 0.7. Esse trecho faz, então, que parte do sinal que irá ser enviado para o canal esquerdo seja somado ao canal direito (e vice-versa), pois a saída de um canal sempre terá influência sobre a saída do outro.

2.4 Reconstrução de sinais discretos

O sinal processado discretamente pelo microcontrolador será enviado para o DAC (*digital-analog converter*) que converterá os valores em bits do sinal digital em valores analógicos de tensão. Lembrando que a relação entre o sinal amostrado, $x_s(t)$, e o sinal discreto, $x[n]$, é dada por

$$x_s(t) = \sum_{n=-\infty}^{\infty} x[n] \cdot \delta(t - nT_s)\,, \tag{62}$$

é necessário, portanto, encontrar uma função $h_r(t)$ capaz de converter o sinal x_s (representado como um trem de impulso) em um sinal analógico contínuo. Se o sinal foi amostrado respeitando o critério de Nyquist, tal função h_r existe e o sinal amostrado pode ser recuperado de sua versão discreta. Idealmente, essa função deverá ter ganho T_s (para compensar o termo $1/T_s$ que surge da amostragem - equação 42) e frequência de corte Ω_c entre Ω_o e $\Omega_s - \Omega_o$, sendo Ω_o a maior componente de frequência do sinal contínuo amostrado (como ilustrado na Figura 17a) e Ω_s a frequência angular de amostragem. Uma escolha comum para a frequência de corte é utilizar $\Omega_c = \Omega_s/2 = \pi/T_s$, pois, assim, para qualquer relação de Ω_s e Ω_o é garantido o critério de Nyquist ($\Omega_s \geq 2\Omega_o$) (OPPENHEIM, 1999). A função

h_r pode ser, portanto, um filtro passa-baixas definido como

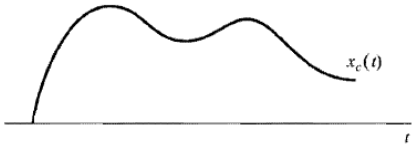
$$h_r(t) = \frac{\text{sen}(\pi t/T_s)}{\pi t/T_s}\,. \tag{63}$$

Dessa forma, o sinal reconstruído será dado por

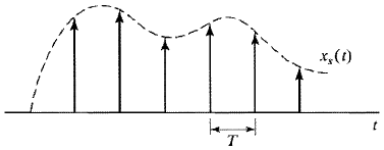
$$x_r(t) = \sum_{n=-\infty}^{\infty} x[n] \cdot h_r(t - nT_s) = \sum_{n=-\infty}^{\infty} x[n] \cdot \frac{\text{sen}[\pi(t - nT_s)/T_s]}{\pi(t - nT_s)/T_s}\,. \tag{64}$$

Tal equação indica que h_r serve como função interpoladora dos valores discretos da função $x[n]$. A Figura 20 ilustra a reconstrução de um sinal contínuo de entrada, x_c, que foi amostrado gerando o sinal x_s e reconstruído através da função h_r.

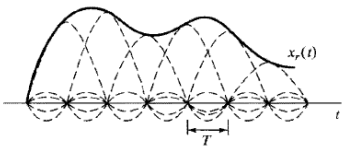
Figura 20 – Reconstrução do sinal x_c contínuo amostrado.



(a) Sinal contínuo $x_c(t)$.



(b) Sinal amostrado de x_c, gerando $x_s(t)$, sendo $T = T_s$.



(c) Representação da interpolação feita entre o sinal reconstrutor h_r e o sinal amostrado x_s, sendo $T = T_s$.

Fonte: OPPENHEIM, 1999.

2.5 Projeto de filtros passa-baixas

Como visto nas seções anteriores, é necessário que haja dois filtros analógicos no sistema: um na entrada para evitar o efeito *aliasing* e outro na saída para reconstruir o sinal processado de suas amostras digitais. Para tanto, é recorrente o projeto de filtros passa-baixas utilizando arquiteturas como Sallen-Key ou MFB (*Multiple-Feedback*) e topologias como a Butterworth, Bessel ou Chebyshev. Baseando-se no artigo de Karki (2000), é apresentado as etapas de projeto para filtros passa-baixas de arquitetura Sallen-Key e topologia Butterworth.

A função de transferência de um filtro passa-baixas de segunda ordem pode ser expressa na forma

$$H(f) = -\frac{K}{\left(\frac{f}{FSF \cdot f_c}\right)^2 + \frac{1}{Q}\frac{jf}{FSF \cdot f_c} + 1}, \tag{65}$$

em que f é a frequência variável, f_c é a frequência de corte do filtro, FSF é o fator de escala da frequência e Q é o fator de qualidade. Esses parâmetros determinam a equação genérica para qualquer topologia de filtro passa-baixas e são utilizados para se determinar a equação final (numérica) do filtro, sendo os valores de FSF e Q tabelados para cada estágio e variam para cada topologia. Em filtros Butterworth, a equação é otimizada para que haja ganho quase constante na banda de passagem, atenuação de -3dB na frequência de corte e, acima da frequência de corte, atenuação de -20dB/década/ordem (ou -6dB/oitava/ordem).

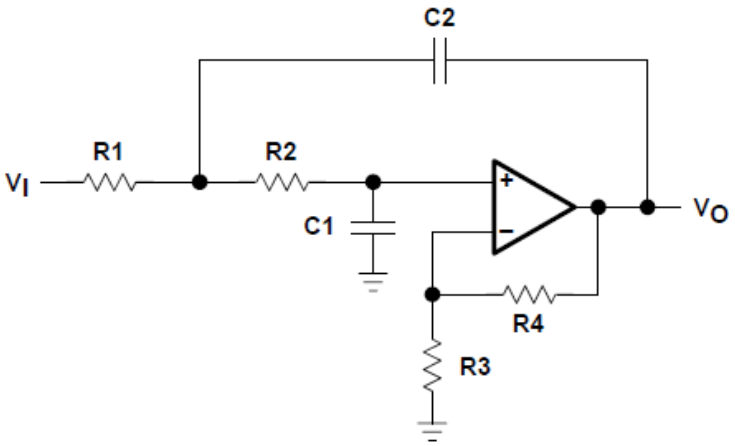
O circuito para a arquitetura Sallen-Key de um filtro passa-baixas de segunda ordem é mostrado na Figura 21. Sua função de transferência é dada por:

$$H(f) = \frac{\frac{R3+R4}{R3}}{(j2\pi f)^2(R1R2C1C2) + j2\pi f(R1C1 + R2C1 - R1C2R4/R3) + 1}. \tag{66}$$

Esse circuito representa um estágio do filtro. Os parâmetros como descritos na equação 65 são dados por:

- Ganho do filtro: $K = \frac{R3+R4}{R3}$;
- Frequência de corte do filtro: $f_c = \frac{1}{2\pi\sqrt{R1R2C1C2}}$;
- Fator de qualidade: $Q = \frac{\sqrt{R1R2C1C2}}{R1C1+R2C1+R1C2(1-K)}$.

Figura 21 – Topologia Sallen-Key de filtros ativos.



Fonte: KARKI, 2000.

Para a maioria das aplicações, é comum a necessidade de filtros de ordens mais altas. Para se obter isso, recorre-se ao cascateamento de n estágios (n par) de filtros

passa-baixas de segunda ordem. Isso é feito adicionando-se em série $n/2$ estágios de filtros passa-baixas de segunda ordem para se obter o filtro da ordem desejada. Caso se queira obter um filtro de ordem ímpar, porém, utiliza-se um filtro de primeira ordem em série com um *buffer* antes do primeiro estágio. Para cada estágio do filtro, deve-se ater ao valor de FSF e Q que devem ser utilizados para que a função de transferência final corresponda a topologia escolhida. Em filtros Butterworth, o valor de FSF para todos os estágios (independentemente da ordem) é igual a 1, o que permite maior facilidade no projeto. A tabela 1 mostra valores de Q para cada estágio de um filtro passa-baixas Butterworth de até sexta ordem.

Tabela 1 – Valores de Q para cada estágio de um filtro Butterworth de até sexta ordem.

Ordem do filtro	Estágio 1	Estágio 2	Estágio 3
2	0.7071		
3	1		
4	0.5412	1.3065	
5	0.6180	1.6181	
6	0.5177	0.7071	1.9320

Fonte: modificado de KARKI, 2000.

Para facilitar o projeto, diversas técnicas de simplificação são apresentadas por Karki (2000) ; porém, mesmo assim, o projeto costuma ser difícil e complexo. Por conta disso, é recorrente o uso de algoritmos e programas já implementados com essa finalidade.

3 Materiais e Métodos

Nessa seção serão apresentados os procedimentos para a elaboração do pedal digital. Inicialmente é apresentada a implementação algorítmica dos efeitos discretizados em MATLAB para simulação e definição dos parâmetros e suas faixas de valores que poderão ser alterados pelo usuário final. A seguir, é feita a validação de tais algoritmos para o cenário de quantização dos dados (simulando, assim, as características de funcionamento mais encontradas em microcontroladores comerciais de baixo custo).

O microcontrolador escolhido para executar os algoritmos é apresentado em sequência, juntamente à adaptação dos códigos em MATLAB para linguagem C++ a fim de se obter melhor agilidade e otimização dos códigos. Além disso, são explanadas as configurações internas do microcontrolador (como ADCs, DACs, *timers*, entre outros).

Sucessivamente, é apresentado o projeto do *hardware* periférico ao microcontrolador e o projeto de confecção da placa de circuito impresso (PCI) do pedal. Por fim, é proposto o teste de comparação entre o pedal desenvolvido e o pedal RT-20 da BOSS para diferentes configurações e sinais de entrada.

3.1 Implementação

O diagrama de blocos do sistema completo proposto para simular os efeitos da caixa Leslie microfonada com microfones à 180° é apresentado na Figura 22. A partir desse diagrama será feito o algoritmo final do simulador. Lembre-se que o tambor e as cornetas estão girando em sentidos opostos e, por isso, suas funções moduladoras estarão defasadas em 180°.

3.1.1 Distorção harmônica

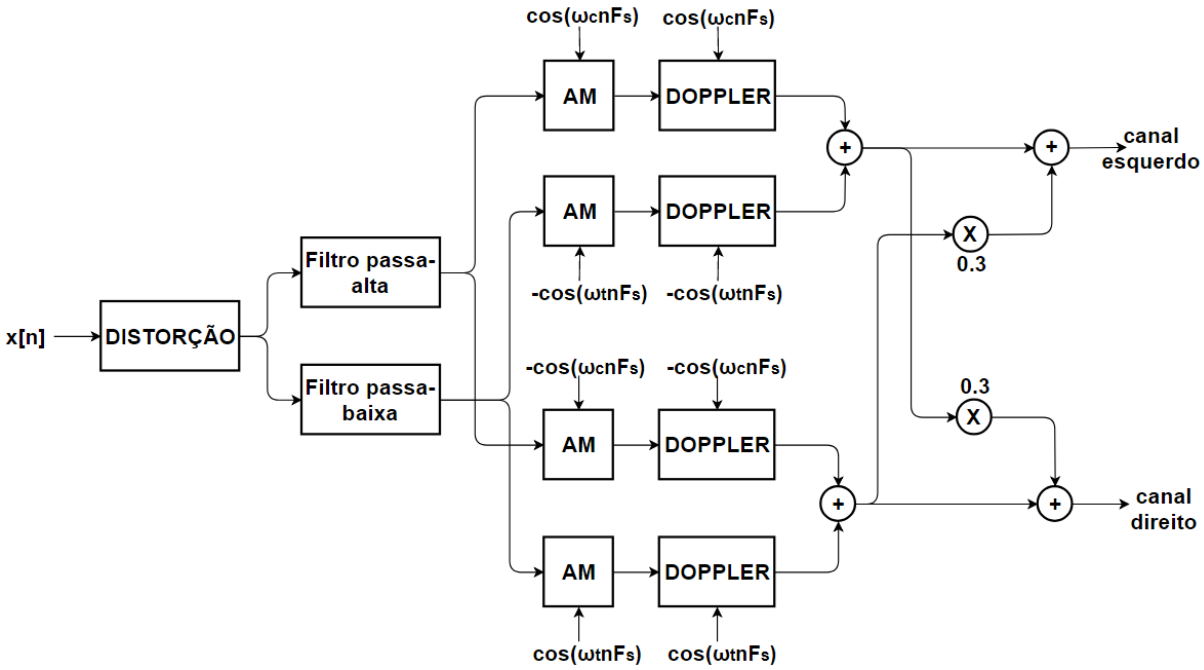
Para o algoritmo da distorção harmônica, a equação 1 é reescrita na forma:

$$y(x) = \begin{cases} 1 - e^{-dist \cdot x} & x \geq 0 \\ -1 + e^{dist \cdot x} & x < 0 \end{cases}, \tag{67}$$

onde *dist* é o nível de distorção. Assim, diminui-se o custo computacional de calcular a relação $\frac{x}{|x|}$, tornando a resposta do sistema mais rápida.

Para a simulação dos efeitos da distorção foi utilizado o código presente no APÊNDICE A.1 com uma senoide de 100Hz de entrada. Gerou-se, então, os gráficos da saída do sinal distorcido com a função *plot()* e suas representações no domínio da frequência com a função *fft()*, para averiguação dos harmônicos introduzidos, com valores de *dist* de 1 à 10.

Figura 22 – Diagrama de blocos do sistema completo para simulação da caixa Leslie.



Fonte: Própria, 2018.

Foi utilizada essa faixa de valores para *dist*, pois a caixa Leslie tem como característica a introdução de apenas uma leve distorção no sinal de entrada.

3.1.2 Filtros de *crossover*

Substituindo os valores dos componentes dos filtros da Figura 13 nas equações 2 e 3, obteve-se as funções de transferência dos filtros filtros analógicos. Armazenando em vetores os coeficientes do numerador e denominador das funções de transferência e utilizando a função *tf()* é possível definir dentro do MATLAB tais funções. Assim, torna-se fácil trabalhar e converter tais filtros para o domínio digital. Usando os vetores para o numerador e denominador dos filtros analógicos, a frequência de amostragem e a frequência de *crossover* na função *bilinear()*, converte-se os filtros analógicos em filtros digitais - sendo o retorno dessa função o numerador e denominador do filtro digital. Os vetores contendo o numerador e denominador dos filtros analógicos foram utilizados, então, na função *bode()* para se obter o diagrama de Bode analógico; e o vetor contendo o numerador e denominador dos filtros digitais foram utilizados na função *freqz()* para se obter o diagrama de Bode digital dos filtros.

Para se fazer a filtragem digital de um sinal já amostrado a frequência F_s usou-se a função *filter()* - que faz a convolução entre a função de transferência do filtro com o sinal amostrado. A função *filter()* recebe os vetores do numerador e denominador do filtro discreto, juntamente com sinal amostrado para se obter o sinal de saída já filtrado. Tal

função foi usada, então, para se obter os sinais filtrados digitalmente. O código que faz a filtragem de um sinal de entrada em suas componentes de baixa e alta frequência em torno de uma frequência de *crossover* encontra-se no (APÊNDICE A.2).

Outra forma de se implementar os filtros digitais é utilizando as equações de diferenças que descrevem os mesmos. Para as simulações em MATLAB, o uso de tais equações não se torna prático, pois as funções pré-definidas tornam a simulação mais versátil, permitindo pouca codificação e praticidade no manipulação dos dados.

3.1.3 Efeito Doppler

Como já descrito em seção anterior, o efeito Doppler é implementado utilizando a técnica do Atraso Fracionário, por intermédios da interpolação linear. O algoritmo capaz de efetuar o efeito Doppler para um sinal já amostrado é descrito pelas seguintes etapas (ZÖLZER, 2011):

1. Lê o valor do tempo de atraso (valores entre 0.2 e 1ms);
2. Lê o valor da velocidade do alto-falante, em rpm (valores entre 40 a 340rpm para o tambor, e 50 a 400rpm para cornetas);
3. Calcula o número de amostras para o atraso (equação 56);
4. Converte a velocidade para Hz (divide o valor em rpm por 60);
5. Calcula o tamanho da linha de atraso (a linha de atraso será uma estrutura de dados, do tipo lista, por exemplo, que conterá as amostras passadas do sinal de entrada);
6. Inicializa a linha de atraso (ou seja, coloque todos os valores da lista em zero);
7. Lê a amostra atual;
8. Calcula o valor da função moduladora (equação 55);
9. Separa a parte inteira e fracionária da função moduladora;
10. Insere o sinal de entrada na linha de atraso;
11. Calcula a interpolação linear (equação 58) usando a linha de atraso;
12. Envia o sinal calculado pela interpolação linear para a saída do sistema;
13. Volta para o item 7 enquanto não tiver percorrido todas as amostras do sinal de entrada.

A linha de atraso descrita acima refere-se a uma estrutura de dados do tipo lista capaz de armazenar informações de forma análoga a uma *array* em linguagem C. Para

esse algoritmo, a inserção dos elementos foi feita através da técnica *push front*, ou também denominada *insert*, em que novos elementos são adicionados no início da lista, fazendo com que o índice de posição dos demais elementos aumentem em 1 a cada nova inserção. Esse método torna simples a iteração sobre os valores passados de entrada, mas não é um método eficiente, pois é necessário o realocamento de memória de todos os valores da linha de atraso a cada iteração do algoritmo - o que acaba sendo lento. Uma forma de se agilizar esse processo é armazenando o local de memória das novas inserções através de ponteiros. O tamanho da linha de atraso será dado por:

$$L = \max(\phi[n]) = 2 \cdot D_t \,, \tag{68}$$

onde D_t é o número de amostras atrasadas calculado pela equação 56.

No APÊNDICE A.3 está o código descrito por esse algoritmo com a possibilidade de variação dos parâmetros do tempo de atraso e da velocidade de rotação do alto-falante. A simulação do efeito Doppler foi feita utilizando uma entrada do tipo senoidal com frequência de 500Hz e variando-se dois parâmetros: o tempo de atraso e a velocidade de rotação do alto-falante.

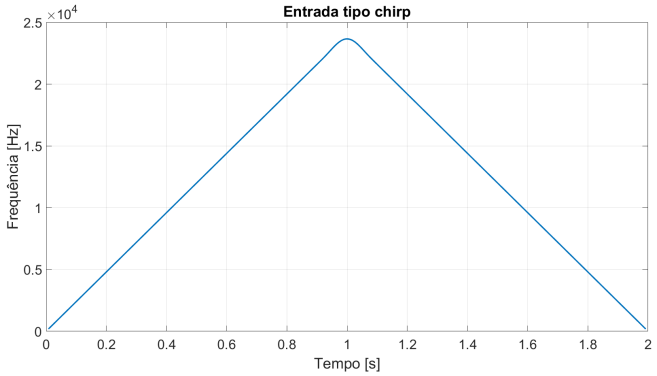
3.1.4 Modulação em amplitude

A modulação AM foi implementada exatamente como a equação 61. A simulação da modulação em amplitude foi feita com a variação de dois parâmetros: o coeficiente de modulação (definido entre 0 e 1) e a velocidade de rotação do alto-falante. No APÊNDICE A.4 está o código referente a modulação em amplitude com a possibilidade de variação desses parâmetros.

3.2 Validação para quantização de 12bits e formato *float*

Até então todos os algoritmos propostos foram feitos usando valores contínuos com a precisão *double* de 64bits do MATLAB. Contudo, muitos microcontroladores empregam uma quantização de 12bits dos valores do sinal de entrada e saída e implementam números em formato *float* de 32bits. Para validar a funcionalidade dos algoritmos para esse cenário, simulou-se a quantização de 12 bits e diminuiu-se a precisão dos valores para 32bits no MATLAB na simulação dos efeitos. Utilizou-se como entrada do sistema a função *chirp()*, que faz uma varredura linear de todas frequências até uma certa frequência máxima. Delimitou-se, então, a frequência máxima de 24kHz (considerando amostragem a 48kHz e o critério de Nyquist). A Figura 23 mostra a variação da frequência do sinal para uma entrada do tipo *chirp* de 2 segundos. Como é possível perceber, a frequência do sinal cresce linearmente até 24kHz, em 1 segundo, e, após, decresce linearmente também, fazendo toda a varredura de frequências possível do sinal de entrada.

Figura 23 – Variação na frequência ao longo do tempo de uma entrada do tipo *chirp*.



Fonte: Própria, 2018.

Para as simulações, considerou-se o sistema completo para saída mono, utilizando-se nível de distorção de $dist = 10$, tempo de atraso de $t_a = 1$ms para o efeito Doppler, velocidades dos alto-falantes em 340rpm para o tambor e 400rpm para as cornetas, e coeficiente de modulação em amplitude de $\alpha = 0.8$. Simulando-se o sistema com precisão *double*, e *float* com 12bits de quantização, subtraiu-se os sinais de saída para se obter o sinal de erro e calculou-se, através da função *immse()*, o erro quadrático médio entre os sinais (Statistics How To, 2018).

3.3 Teensy 3.6

Para a elaboração desse trabalho, escolheu-se utilizar o kit de desenvolvimento Teensy 3.6, desenvolvido pela PJRC (2018), usado em aplicações microcontroladas de alto desempenho. As especificações do kit mais relevantes ao trabalho estão na Tabela 2. A Figura 24 apresenta o Teensy 3.6 e sua pinagem das vistas superior e inferior da placa.

O uso do núcleo Cortex-M4F (ARM Cortex-M4 32b, 2018) faz com que seja possível executar tarefas de alta performance, incluindo processamento via *hardware* para ponto flutuante de 32bits - o que faz sua execução ser até 30 vezes mais rápida em relação àqueles cujo processamento é via *software*, além de agregar instruções DSP (PJRC, 2018).

Os 12bits de resolução dos ADCs e DACs permitem uma discretização de 4096 valores de entrada e saída. Dessa forma, considerando a excursão de sinal máxima de $3,3V$, haverá resolução de $0,81mV/bit$. Além disso, por haver dois DACs, o Teensy 3.6 possibilita a saída de áudio em estéreo (dois canais).

Agregado a isso, o uso do Teensy 3.6 facilita a programação por ser compatível com a IDE do Arduino através da extensão Teensyduino , permitindo a programação em linguagem C++ do microcontrolador e a utilização de funções pré-programadas.

De acordo com CIRCUITAR (2018), ao se utilizar a IDE Teensyduino, é necessário

Tabela 2 – Especificações técnicas do Teensy 3.6.

Característica	Especificação
Processador	MK66FX1M0VMD18
Núcleo do processador	Cortex-M4F
Velocidade de processamento	180Mhz
Memória Flash	1024kbytes
RAM	256kbytes
EEPROM	4096bytes
Entradas analógicas	58 pinos
Conversores (ADCs)	2
Resolução dos ADCs	12bits
Tensão de entrada máxima	3.3V
Saídas analógicas (DACs)	2 pinos
Resolução do DAC	12 bits
Tensão de saída máxima	3.3V
Corrente de saída máxima	10mA
Timers	19 (6 com 16bits)

Fonte: PJRC, 2018.

programar duas funções: a *setup()* e a *loop()*. A função *setup()* será executada apenas uma vez no início do fluxo do código. Ela é utilizada, então, para se fazer as configurações iniciais. A função *loop()* será executada após a função *setup()* e ficará se repetindo. Nela são adicionados, então, os trechos de código que devem ser executados repetidamente.

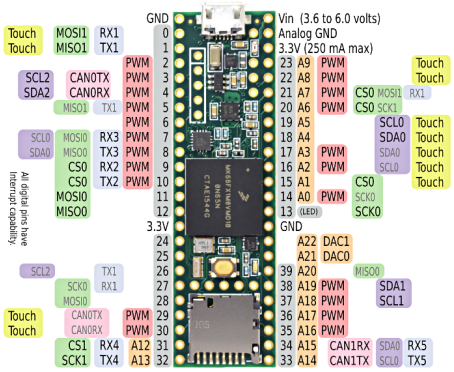
3.3.1 Implementação em C++

Para converter os algoritmos propostos em MATLAB para C++ a fim de serem usados dentro de um microcontrolador é necessário que alguns aspectos da programação sejam alteradas para se obter mais velocidade de processamento e melhor alocamento da memória do microcontrolador.

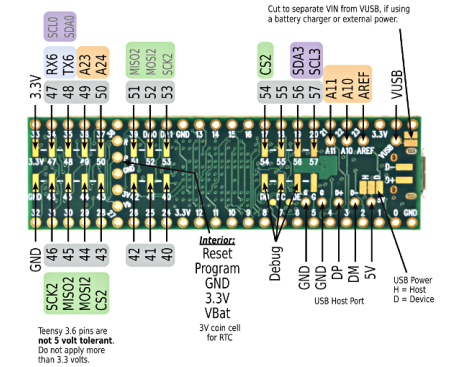
3.3.1.1 Linha de atraso

Em diversos momentos do código é necessário que sejam armazenados os valores passados de entrada e saída dos efeitos - como na implementação dos filtros (que será melhor abordada na seção 3.3.1.2) e do efeito Doppler. Como explanado anteriormente, a forma de implementação *push front* utilizada no algoritmo em MATLAB é demasiadamente lenta para se aplicar a um sistema de áudio. Por conta disso, utiliza-se a técnica da *linha de atraso*. Sua codificação baseia-se no uso de uma *array* armazenando os valores, um ponteiro de escrita "apontando" para essa mesma *array* (ou seja, armazenando a posição de memória da mesma), uma função de escrita na *array* e uma função de busca dos valores atrasados (isso para cada variável que necessite armazenamento de seus valores

Figura 24 – Pinagem do kit Teensy 3.6.



(a) Vista superior.



(b) Vista inferior.

Fonte: PJRC, 2018.

passados). O ponteiro serve, então, como referencia de onde foi armazenado o último valor dentro da *array*.

A função de escrita recebe apenas o valor que será armazenado na *array*. Ela inicia verificando se a posição de escrita do ponteiro ainda está nos limites da *array*. Caso não esteja, o ponteiro deve voltar a posição inicial (índice 0 da *array*). Após isso, o valor de entrada é escrito na posição de memória do ponteiro e esse é acrescido para pular uma posição de memória. Note que essa função não possui retorno, e serve apenas para armazenar o valor da amostra atual dentro da *array* e atualizar a posição de memória na escrita da *array* em relação ao seu ponteiro.

A função de leitura é um pouco mais complexa, pois ela precisa lidar com o fato dessa implementação tornar a *array* cíclica pela forma como a função de escrita foi feita. Sua implementação se baseia, então, em identificar dentro da *array* o índice correspondente ao atraso solicitado. Essa identificação é feita verificando se a posição atual do ponteiro dentro da *array* é maior que o número de amostras atrasadas solicitada. Caso seja, a função retorna o valor da *array* na posição do ponteiro menos o número de amostras

atrasadas. Contudo, se a posição atual do ponteiro dentro da *array* for menor que o número de amostras atrasadas, a função retorna o valor da *array* na posição dada por:

$$p = TAM - (n - ptr)\,, \tag{69}$$

onde TAM é o tamanho da *array*, n é o número de amostras atrasadas e ptr é a posição relativa do ponteiro dentro da *array* (índice da *array* que o ponteiro aponta).

3.3.1.2 Equações de diferenças para os filtros de *crossover*

A implementação dos filtros em código C++ se deu através da transcrição das equações de diferenças dos mesmos. Para tanto, foram armazenados os coeficientes das equações de diferenças (equação 53) dentro de um *array* do tipo *const float* e os valores atrasados de entrada e saída dentro de um *array* do tipo *volatile float* com implementação na forma de *linha de atraso*. O uso do prefixo *const* faz com que os valores da *array* de coeficientes sejam alocados no local de memória estático (ou seja, não possibilitando que eles sejam alterados no fluxo do código); enquanto que, o uso do prefixo *volatile* informa ao compilador que aquela variável poderá ser alterada sem o conhecimento do programa principal (nesse caso, dentro das interrupções) e, por esse motivo, ela não deve ser otimizada.

3.3.1.3 Aproximações numéricas

Para os algoritmos da distorção, do efeito Doppler e da modulação em amplitude é necessário o uso de funções que não são implementadas via *hardware* na maioria dos microcontroladores. São essas funções: a exponencial para a distorção e o cosseno para o efeito Doppler e a modulação em amplitude. Por conta disso, é necessário se aproximar essas funções com o uso de polinômios. Há diversas técnicas para isso, como o uso das séries de Taylor e de Maclaurin, ou dos polinômios de Chebyshev. Nesse trabalho, no entanto, utilizou-se a função *polyfit()* presente no MATLAB, que permite a aproximação polinomial de uma função em um polinômio de ordem n.

Para se usar a função *polyfit()* definiu-se o intervalo de valores que cada uma das funções deveria ser aproximada (de -1 a 1 para a exponencial, e de $-\pi/2$ a $\pi/2$ para o cosseno) e o tamanho do passo utilizado (supondo-se quantificação de 12bits, utilizou-se passo de $\frac{1}{2^{12}} = 2.44 \cdot 10^{-4}$). Com isso, cria-se os vetores para a variável independente e o vetor contendo as funções aplicadas a tal variável. Entrando com esses dois vetores e a ordem do polinômio na função *polyfit()* obtém-se os coeficientes do polinômio que melhor aproxima a função. A ordem escolhida para os polinômios foi $n = 7$, pois, para ordens maiores, o custo e velocidade computacional podem ser afetados. As aproximações foram, então, comparadas com as funções originais a fim de se obter o erro absoluto e o erro quadrático médio.

3.3.2 Configuração dos ADCs

Os dois ADCs (chamados ADC0 e ADC1) foram configurados utilizando a biblioteca *ADC.h*, própria do Teensy, instanciando um objeto "adc" da classe implementada por essa biblioteca. Estabeleceu-se que o ADC0 (referenciado como *ADC_0* dentro do código) seria utilizado apenas para a aquisição dos parâmetros dos efeitos e o ADC1 (referenciado como *ADC_1* dentro do código) seria utilizado apenas para a aquisição dos sinais recebidos pela guitarra. Dentro da biblioteca do Teensy, é possível se configurar diversos aspectos dos ADCs. As configurações feitas em cada ADC estão presentes na Tabela 3. Nela é apresentado o nome do método utilizado no objeto "adc", sua especificação e o parâmetro passado.

Tabela 3 – Método, especificação e parâmetro utilizado para a configuração dos ADCs.

Método	Especificação	ADC0	ADC1
.setReference()	Referência de tensão	3.3V	3.3V
.setSamplingSpeed()	Velocidade de amostragem	VHS[1]	VHS
.setConversionSpeed()	Velocidade de conversão	VHS	VHS
.setResolution()	Resolução	8bits	12bits
.setAveraging()	Médias feitas na amostragem	32	4

Fonte: Própria, 2018.

Além disso, foi utilizada a função *pinMode()* para identificar os pinos de entrada, passando a ela o número do pino e a constante *INPUT*. Essas configurações foram implementadas dentro de uma função própria criada apenas com essa finalidade e chamada dentro da função *setup()*. A Figura 25 mostra, então, os pinos referentes a cada um dos ADCs.

3.3.3 Configuração dos DACs

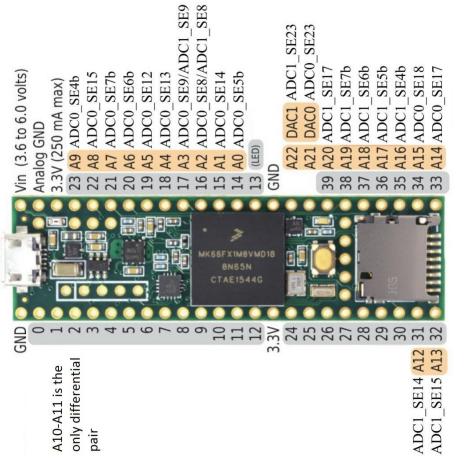
A configuração dos DACs é bastante simples. Para tanto, utilizou-se a função *pinMode()* para identificar os pinos de saída passando a constante *OUTPUT* como segundo parâmetro. Acrescido disso, usou-se a função *analogWriteResolution()* com parâmetro 12 para utilizar os 12bits de resolução dos DACs. Essas configurações foram implementadas dentro de uma função própria criada apenas com essa finalidade e chamada dentro da função *setup()*.

3.3.4 Configuração do *timer*

O *timer* foi configurado utilizando a classe *IntervalTimer* , já implementada no Teensyduino, para se garantir constante a frequência de amostragem. Sua configuração foi

[1] VERY_HIGH_SPEED

Figura 25 – Pinagem dos ADCs do Teensy 3.6.



Fonte: FORUM PJRC, 2017.

feita instanciando um objeto "amostrador" da classe *IntervalTimer*. Utilizando o método *.begin()* é possível estabelecer a função que será chamada pela interrupção do *timer* e o período de chamada da interrupção (que é o período de amostragem). O período de amostragem utilizado foi de $\frac{1}{48kHz} = 20.83\mu s$. Por fim, usando o método *.priority()* com parâmetro "0" define-se que essa interrupção terá máxima prioridade. Essas configurações foram implementadas dentro de uma função própria criada apenas com essa finalidade e chamada dentro da função *setup()*.

3.3.5 Rotina de interrupção

A rotina de interrupção se refere a função chamada pelo *timer* a cada $20.83\mu s$ (período de amostragem). Dentro dessa rotina será feita: a aquisição do sinal da guitarra, o envio dos sinais processados, e o processamento do sinal amostrado através dos efeitos da caixa Leslie descritos com base no diagrama de blocos da Figura 22.

Inicia-se a rotina com a aquisição do sinal da guitarra - utilizando o método *.analogRead()* sob o objeto "adc" passando o pino de entrada (A13) e o ADC (*ADC_1*) como parâmetros. A leitura do ADC retorna um valor que vai de 0 à 4095 ($2^{12} - 1 = 4095$). Logo, é necessário que se converta esse valor para o intervalo de -1 à 1 subtraindo-se o valor médio (2048) e dividindo o resultado por 4096.

Após isso, faz-se a escrita dos sinais processados na última execução da função com o uso da função *analogWrite()* passando o pino do canal (A21 e A22) e o valor do sinal processado para esse canal como parâmetros. Como os valores do sinal processado estão na faixa de -1 à 1, é necessário retorná-los para valores de 0 à 4095 a fim de que eles possam ser enviados para o DAC. Isso é feito somando-se um ao sinal processado e multiplicado pelo valor médio (2048). A rotina tem seguimento, então, com a implementação dos efeitos

da caixa Leslie sob o novo sinal amostrado.

3.3.6 Leitura dos parâmetros para os efeitos

A leitura dos parâmetros para os efeitos foi feita dentro da função *loop()*. Nessa função, há inicialmente um atraso de 500ms implementado usando a função *delay()* para estabilidade do algoritmo e, após, os parâmetros são lidos com o uso do método *.analogRead()* sob o objeto "adc" passando o número do pino que deve ser lido (A5 ao A9) e o ADC0 (ADC_0) como parâmetros. Os valores lidos foram, então, mapeados entre seus valores máximos e mínimos tendo em vista que a leitura do ADC0 retorna valores entre 0 à 255 ($2^8 - 1 = 255$). A Tabela 4 aglutina todos os parâmetros, com suas faixas de valores na unidade de referência e a função de mapeamento usada, além do nome impresso no pedal. Nessa tabela, POT refere-se ao valor retornado pelo método *.analogRead()* sob o objeto "adc" e Fs é a frequência de amostragem. Observe que o mapeamento de alguns parâmetros são normalizados ou multiplicados pela frequência de amostragem, a fim de que esses valores sejam ajustados corretamente para o domínio discreto.

Tabela 4 – Mapeamento dos parâmetros.

Parâmetro	Nome no pedal	Faixa	Função de mapeamento
Distorção	DISTORTION	1 à 10	$POT/255 \cdot 9 + 1$
Velocidade da corneta	HORN_SPEED	50 à 400rpm	$(1.5 + 5.17 \cdot POT/255)/Fs$
Velocidade do tambor	DRUM_SPEED	40 à 340rpm	$(0.7 + 5 \cdot POT/255)/Fs$
Tempo de atraso	DELAY	0.1 à 1ms	$(0.1 + 0.9 \cdot POT/255) \cdot Fs$
Coeficiente de modulação	MODULATION	0 à 1	$POT/255$

Fonte: Própria, 2018.

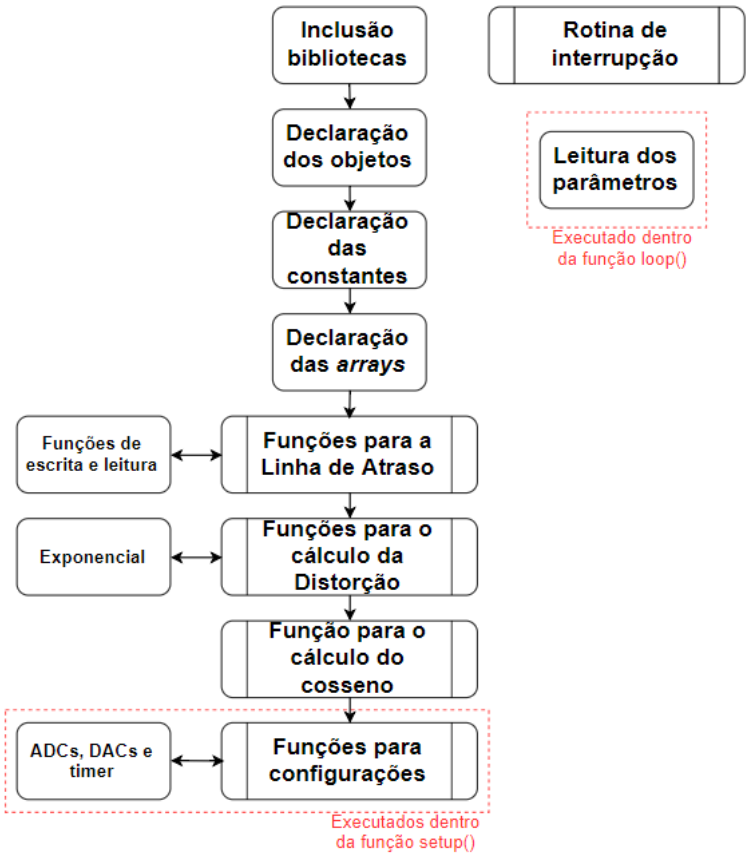
Outro parâmetro importante se refere a escolha entre os modos *chorale* e *tremolo*, feito através da chave 3PDT. Sem pressionar a chave, a velocidade dos alto-falantes será a velocidade mínima possível; enquanto que, ao se pressionar a chave, a velocidade será trocada para a velocidade escolhida pelo usuário. Isso foi feito inicializando o pino digital com a função *pinMode()* (informando o pino de ligação e a constante $INPUT$), e seu valor lido com a função *digitalRead()* (passando apenas o pino digital a ser lido), que retorna o valor booleano da leitura.

3.3.7 Organização do código

Visto os aspectos mais específicos de conversão dos efeitos para a linguagem C++ e das configurações de uso do microcontrolador, essa seção detém-se a apresentar a organização do código em forma de blocos sequenciais com a explicação de cada um deles. Para tanto, observe o diagrama de blocos da Figura 26 que mostra sequencialmente a

escrita do código dentro do microcontrolador. O código completo em C++ se encontra no (APÊNDICE B).

Figura 26 – Diagrama da organização do código.



Fonte: Própria, 2018.

Abaixo, então, é explicado cada um dos blocos da Figura 26:

- **Inclusão bibliotecas**: trecho de código que inclui a biblioteca para lidar com os ADCs.
- **Declaração dos objetos**: declaração dos objetos para os ADCs e para o *timer* ("adc" e "amostrador", respectivamente).
- **Declaração das constantes**: declaração de todos os valores que serão constantes dentro do código, sendo eles: tamanho das *array*s para as amostras de entrada, para os filtros e para o efeito vibrato; frequência de amostragem; período de amostragem; *array*s contendo os coeficientes do numerador e denominador das equações de diferenças para os filtros; valor de π para o cálculo do cosseno.
- **Declaração das *array*s**: declaração de todas as *array*s implementadas como linha de atraso e seus ponteiros, sendo elas: para os valores de entrada de amostras, para valores passados dos filtros e para valores do efeito vibrato em suas componentes graves e agudas para cada um dos canais.

- **Funções para Linha de Atraso**: trecho de código que implementa as funções de escrita e leitura para cada uma das *array*s declaradas no item anterior.
- **Funções para o cálculo de distorção**: implementação da função de aproximação da exponencial e da função que introduz a distorção.
- **Função para o cálculo do cosseno**: função que aproxima o cosseno de acordo com a aproximação polinomial.
- **Rotina de interrupção**: função chamada pelo *timer* a cada período de amostragem com a execução da rotina de interrupção descrita na seção 3.3.5, com o uso da Figura 8 para a implementação dos efeitos.
- **Funções para configurações**: funções criadas para configuração dos ADCs, DACs e *timer* (como descritas nas seções 3.3.2, 3.3.3 e 3.3.4 respectivamente) e chamadas dentro da função *setup()*.
- **Leitura dos parâmetros**: rotina que faz a leitura dos parâmetros e a conversão do valor lido para a faixa de valores correspondentes de acordo com a Tabela 4 dentro da função *loop()*.

3.4 Projeto do *hardware*

Para se obter a integração do sistema digital para simular a caixa Leslie com o sistema físico que envolve a entrada do sinal de uma guitarra e a saída para um amplificador, é necessário se projetar um circuito analógico periférico ao microcontrolador. Tal sistema analógico tem como funções, portanto, fazer a alimentação do sistema, ajustar o sinal da guitarra que chega ao microcontrolador e ajustar o sinal de saída do microcontrolador para o amplificador. Agregado a isso, é necessário interligar os componentes responsáveis pelo ajuste dos parâmetros dos efeitos: os potenciômetros lineares e a chave de seleção do tipo 3PDT. O circuito do pedal completo com todas as ligações encontra-se no (APÊNDICE C).

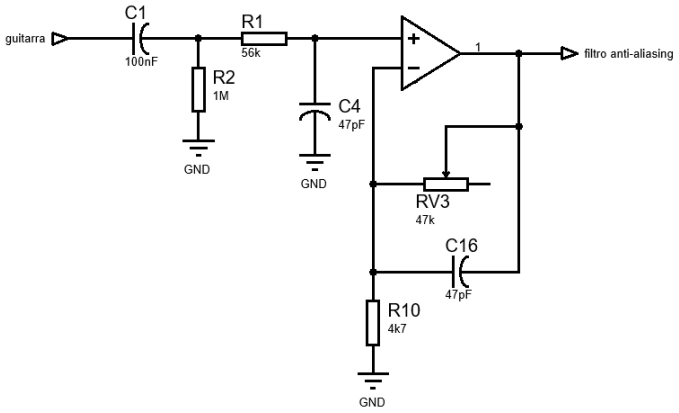
3.4.1 Filtro de entrada

O filtro de entrada constitui os seguintes blocos de circuitos respectivamente: (1) casamento de impedância com a entrada; (2) amplificação do sinal; (3) filtragem; e (4) ajuste do nível do sinal para a entrada do ADC.

O nível de tensão proveniente de uma guitarra varia desde os 200mVpp até os 800mVpp (dependendo do tipo de captador) (DAILEY, 2012) e possui uma impedância de saída em torno de 20kΩ à 40kΩ (ProSoundWeb, 2018). Para se aproveitar a maior faixa dinâmica do ADC (3.3Vpp) é necessário que esse sinal seja amplificado e, a fim de

que a maior parte do sinal que sai da guitarra seja transmitida para o circuito, precisa-se também que esse tenha uma impedância de entrada maior que a impedância da guitarra. Uma impedância de entrada do circuito 10 vezes maior que a impedância da guitarra, por exemplo, fará com que 90% do sinal seja de fato transmitido para o circuito. O bloco de circuito projetado para essa função é mostrado na Figura 27. O capacitor C1 elimina eventuais níveis DC que o sinal de entrada tenha, enquanto que o resistor R2 faz o casamento de impedância para que a maior parte da tensão esteja sob ele e seja enviada, assim, aos estágios seguintes. O restante do circuito é a topologia clássica de um amplificador não inversor com ganho dado por $G = \left(1 + \frac{RV3}{R10}\right)$. Ou seja, o ganho de entrada pode variar de 1 à 11 para que o pedal atenda diferentes modelos de captadores.

Figura 27 – Circuito para o casamento de impedância e amplificação do sinal de entrada.

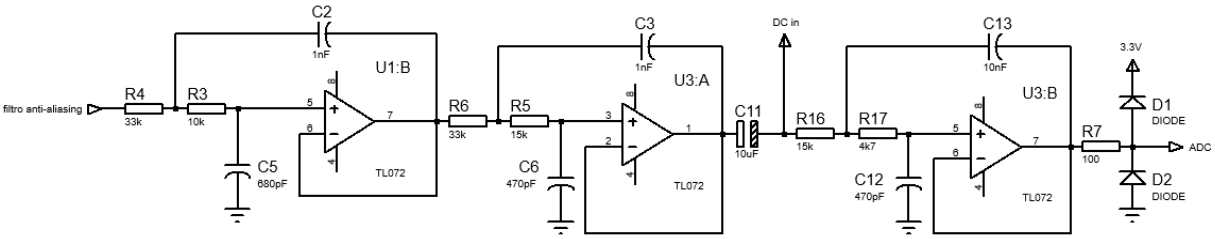


Fonte: Própria, 2018.

Para a filtragem do sinal foi projetado um filtro Butterworth de sexta ordem com frequência de corte em 10kHz e de topologia Sallen-Key utilizando o *web site* OKAWA Electric Design . Com essas especificações, garante-se, então, uma atenuação acima de -40dB em 24kHz (conforme o critério de Nyquist). Nesse *web site* é possível selecionar a frequência de corte e o valor do fator de qualidade do filtro. Utilizando a Tabela 1, projetou-se os três estágios para resultar num filtro de sexta ordem com ganho unitário. O circuito projetado está apresentado na Figura 28. A entrada desse circuito é conectada em série à saída do circuito para o casamento de impedância e amplificação, e sua saída é conectada à entrada do conversor ADC.

O nó identificado como *DC in* na Figura 28 provém do circuito apresentado na Figura 29. Esse circuito é responsável por adicionar nível DC ao sinal de entrada. Trata-se de um divisor de tensão capaz de fazer o sinal excursionar em torno dos 1.65V. Isso é necessário, pois o ADC do Teensy 3.6 só pode receber valores na faixa dos 0V a 3.3V. Os 3.3V são obtidos diretamente dos pinos do Teensy 3.6. O capacitor C11 entre o segundo e terceiro estágio do filtro passa-baixas serve, então, para remover um possível nível DC

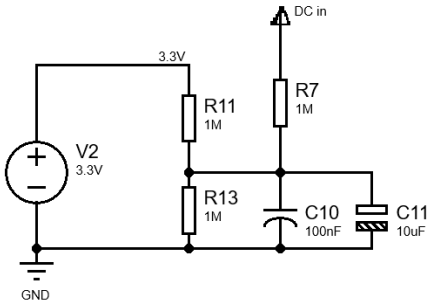
Figura 28 – Filtro *anti-aliasing* Butterworth de sexta ordem com arquitetura Sallen-Key.



Fonte: Própria, 2018.

adicionado pelos *offset*s dos operacionais usados.

Figura 29 – Circuito que adiciona nível DC ao sinal de entrada.



Fonte: Própria, 2018.

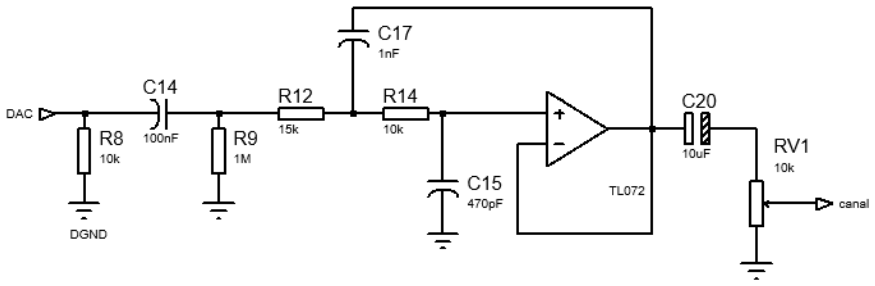
Por fim, o nó de saída do circuito do filtro liga-se ao resistor R7 que limita a corrente na entrada do ADC, e os dois diodos servem como proteção para que o sinal no pino não supere os 3.3V ou tenha valores negativos de tensão. Como são necessários 4 amplificadores operacionais para o circuito de entrada, foi utilizado o integrado TL074, que possui os 4 operacionais necessário em seu encapsulamento.

3.4.2 Filtro de saída

O filtro de saída tem como função reconstruir o sinal enviado pelo DAC e filtrar os harmônicos de mais alta ordem que possam surgir. A saída do DAC do Teensy 3.6 excursiona de 0V a 3.3V e, por esse motivo, é necessário que seja removido o nível DC e atenuado o sinal para que ele possa ser enviado a um amplificador (que é projetado para receber sinais com mesmo nível de tensão que sai de uma guitarra). O circuito foi projetado novamente utilizando o *web site* OKAWA Electric Design (analogamente ao filtro passa-baixas na entrada do circuito) com base na arquitetura Sallen-Key para um filtro passa-baixas de segunda ordem com frequência de corte em 18kHz do tipo Butterworth. Para esse projeto, é importante se ater que só há um valor para o fator de qualidade (como mostra a Tabela 1).

O circuito projetado está apresentado na Figura 30. O resistor R8 e o capacitor C14 formam um filtro passa-altas que elimina flutuações de nível DC do sinal enviado pelo DAC; o resistor R9 faz o casamento de impedância com a saída do DAC; o capacitor C20 remove o nível DC do sinal que pode ter sido adicionado pelo *offset* do amplificador operacional; e o potenciômetro RV1 é responsável por atenuar o sinal de saída (a fim de se adequar ao nível de tensão que seria enviado pelos captadores de uma guitarra). Lembrando que o sistema possui saída estéreo, haverá, então, dois filtros de saída, um para cada canal. Por conta disso, foi utilizado o circuito integrado TL072 que possui em seu encapsulamento dois amplificadores operacionais.

Figura 30 – Filtro de saída para reconstrução do sinal enviado pelo DAC.

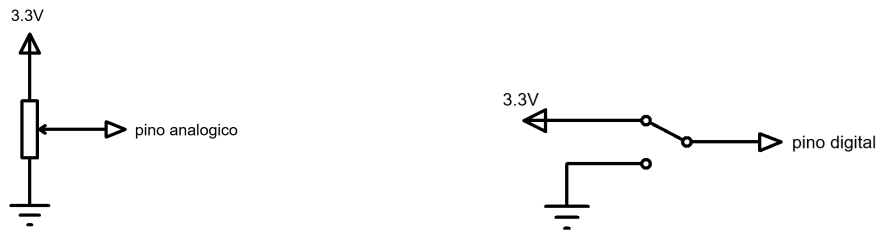


Fonte: Própria, 2018.

3.4.3 Componentes para seleção de parâmetros

Para a seleção dos parâmetros dos efeitos, foram utilizados cinco potenciômetros lineares de 10kΩ. Cada um deles teve os pinos das extremidades ligados aos 3.3V e GND fornecidos pelo Teensy 3.6, enquanto que o pino central foi ligado ao pino analógico do microcontrolador como mostra a Figura 31a. Para alternar entre os modos *chorale* e *tremolo* foi utilizada uma chave 3PDT ligada ao pino digital do Teeensy 3.6 chaveando entre os 3.3V e GND como mostra a Figura 31b.

Figura 31 – Ligações dos componentes para variação dos parâmetros.



(a) Ligação dos potenciômetros. (b) Ligação da chave 3PDT.

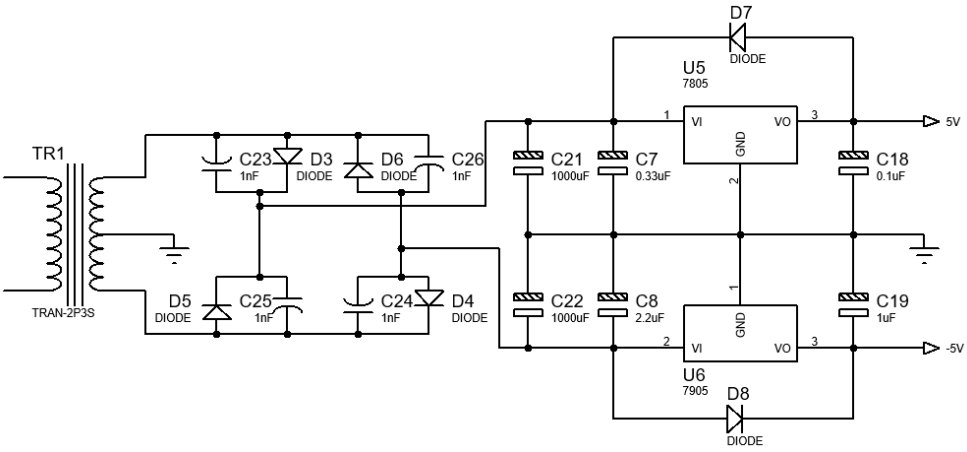
Fonte: Própria, 2018.

3.4.4 Alimentação do circuito

Para alimentar o Teensy 3.6 é necessário uma tensão entre 3.6V e 6V (como ilustrado na Figura 24). Conforme o *datasheet* dos TL07x em (TEXAS INSTRUMENTS, 1978), há uma diminuição na excursão máxima do sinal de saída relativa a alimentação deles. Ou seja, não é possível que o valor de saída do operacional seja igual ao valor de sua alimentação (sendo esse efeito simétrico tanto para a alimentação positiva quanto a negativa). Dessa forma, como se terá um sinal de entrada com excursão entorno dos 1.65V e com 3.3Vpp máximo para a entrada do ADC, optou-se por desenvolver um circuito capaz de fornecer tensão de forma simétrica nos níveis de 5V, 0V e -5V.

Para se obter a fonte simétrica, inicialmente, o circuito de alimentação deverá rebaixar a tensão da rede e converte-lá para três níveis de tensão. Isso é feito utilizando um transformador com três fios no secundário, pois, dessa forma, é possível obter dois níveis de tensão com fases opostas e um nível de referência (que será o 0V). A conversão da tensão AC para DC é feita com um retificador monofásico de onda completa e filtrado por um capacitor de alta capacitância a fim de tornar o sinal retificado em um nível de tensão quase DC. Feito isso, é necessário se estabilizar essa tensão DC gerada utilizando algum regulador de tensão. Escolhendo-se os reguladores uA7805 (TEXAS INSTRUMENTS, 1976) e LM7905 (NATIONAL SEMICONDUCTOR, 1994), é possível estabilizar os níveis de tensão positiva e negativa para 5V e -5V, respectivamente. O projeto do circuito completo é mostrado, então, na Figura 32.

Figura 32 – Circuito para alimentação simétrica.



Fonte: Própria, 2018.

Os capacitores conectados em paralelo aos diodos da ponte monofásica foram colocados para caso as tensões de saída não se mostrarem estáveis. Os capacitores em paralelo com os pinos de entrada e GND, e saída e GND dos reguladores de tensão seguem a recomendação dos *datasheets* dos mesmos. Por fim, os diodos paralelos aos terminais de entrada e saída dos reguladores são utilizados para impedir que os reguladores sejam da-

nificados por possíveis correntes em sentido oposto ao que o regulador fornece (no caso do uA7805) ou consome (no caso do LM7905).

3.5 Confecção das placas em circuito impresso

Com o projeto dos circuitos finalizados, procedeu-se o trabalho com a confecção dos mesmo em placa de circuito impresso (PCI), a fim de tornar o circuito mais estável e resistente. Optou-se por separar o circuito da alimentação do restante do pedal para que não houvesse perigo do usuário ter contato com a tensão da rede. As duas placas foram desenvolvidas no programa EasyEDA , uma plataforma *online* que permite tanto a simulação de circuitos quanto a confecção de PCIs de forma gratuita. Ambas as placas foram confeccionadas em face única e seguem a nomenclatura de componentes conforme os circuitos do APÊNDICE C projetados dentro do EasyEDA.

3.6 Comparação do pedal com a RT-20

Como explanado anteriormente, o pedal RT-20 da BOSS é um dos pedais referência no mercado em simulação da caixa Leslie atualmente. Por isso e devido a indisponibilidade de uma Leslie, optou-se por usar esse pedal como a base de comparação qualitativa. Para tanto, utilizou-se uma configuração fixa para alguns dos parâmetros da RT-20 descritos na Tabela 5. Optou-se por usar o valor mínimo de distorção harmônica, pois o modo de operação I já inclui o que o fabricante informa como a "distorção natural da caixa Leslie".

Tabela 5 – Configurações dos parâmetros fixos da RT-20.

Parâmetro	Descrição	Configuração
INPUT	Entrada de áudio	A (MONO)
OUTPUT	Saída de áudio	A (MONO)
MODE	Modo de operação	I
RISE TIME	Tempo da troca de velocidade	Indiferente
EFFECT	Quanto do efeito terá a saída	Máximo
DIRECT	Quanto do sinal limpo terá na saída	Mínimo
OVERDRIVE	Nível de distorção harmônica	Mínimo
LOW	Velocidade no modo lento	Meio (Padrão)
FAST	Velocidade no modo rápido	Meio (Padrão)

Fonte: Própria, 2018.

Nos testes utilizou-se entradas do tipo senoidal em duas variações de frequência (uma com frequência abaixo de 800Hz e outra acima, a fim de validar o circuito de *crossover*) e onda triangular com frequência de 220Hz para simular uma nota Lá Maior com harmônicos. A Tabela 6 reúne as configurações da RT-20, do tipo e da frequência dos

sinais utilizados para os testes. O parâmetro *BALANCE* permite que o usuário escolha o quanto do sinal oriundo do tambor ou das cornetas seja enviado a saída. Alterando entre os modos *BASS* e *HORN* é possível, então, analisar individualmente o efeito de cada alto-falante.

Tabela 6 – Especificações dos sinais de entrada e parâmetros da RT-20 para os testes realizados.

Teste	Sinal usado	Frequência	Parâmetro RT-20	
			BALANCE	Velocidade
1	Senoide	250Hz	BASS	Lenta
2	Senoide	250Hz	BASS	Rápida
3	Senoide	250Hz	HORN	Lenta
4	Senoide	250Hz	HORN	Rápida
5	Senoide	2.5kHz	BASS	Lenta
6	Senoide	2.5kHz	BASS	Rápida
7	Senoide	2.5kHz	HORN	Lenta
8	Senoide	2.5kHz	HORN	Rápida
9	Triangular	220Hz	Meio	Lenta
10	Triangular	220Hz	Meio	Rápida

Fonte: Própria, 2018.

Conforme é possível inferir das Tabelas 5 e 6, a RT-20 não permite a configuração dos efeitos individuais do efeito Doppler e da modulação em amplitude. Além disso, o pedal desenvolvido nesse trabalho não possui opção análoga ao *BALANCE*. Dessa forma, a fim de ser possível a comparação, fez-se a configuração manual do código já implementado para possibilitar o teste usando apenas o tambor ou a corneta, e testou-se a sonoridade resultante para diferentes valores do coeficiente de modulação (*MODULATION*) e do tempo de atraso (*DELAY*) até que as sonoridades estivessem parecidas. Foi utilizada a velocidade de 50rpm para as cornetas e 40rpm para o tambor no modo lento, 400rpm para as cornetas e 340rpm para o tambor no modo rápido, valor de $dist = 1$ para a distorção e $\alpha = 0.8$ para a modulação em amplitude em todos os testes. O valor do tempo de atraso usado foi de $t_a = 0.5ms$ para o modo lento, e $t_a = 1ms$ no modo rápido para o efeito Doppler.

Os sinais foram adquiridos usando o programa Audacity 2.1.3 com uma frequência de amostragem de 48kHz e usando apenas um canal. Propôs-se quatro comparações para cada conjunto igual de velocidade e frequência da senoide usada:

- Comparação entre os modos *BASS* e *HORN* para o sinal da RT20;
- Comparação entre os modos simulados *BASS* e *HORN* para o sinal do pedal;
- Comparação no tempo do sinal da RT20 com o sinal do pedal;

- Comparação na frequência do sinal da RT20 com o sinal do pedal.

As duas primeiras comparações visam verificar o efeito do *crossover*. A comparação no tempo e frequência do sinal do pedal com o da RT20 foram variados de acordo com a frequência da senoide: para a frequência de 250Hz se comparou os sinais do tambor; enquanto que, para a frequência de 2.5kHz, comparou-se os sinais das cornetas. Nessas últimas duas comparações, os sinais foram normalizados e adicionou-se um *offset* no sinal do pedal a fim de melhor visualizar as formas de onda. Por fim, os sinais de onda triangular foram comparados no tempo e frequência.

4 Resultados

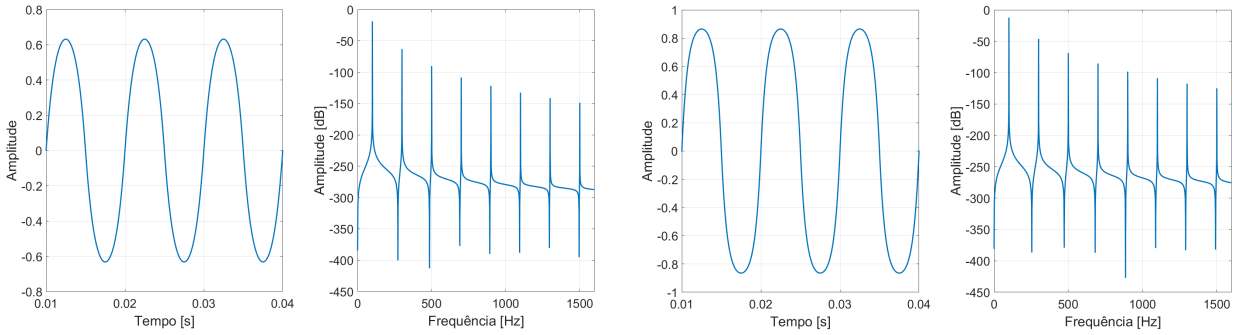
4.1 Simulações dos efeitos

Nessa seção são apresentados os resultados das simulações dos efeitos da caixa Leslie com a variação dos parâmetros em MATLAB.

4.1.1 Distorção harmônica

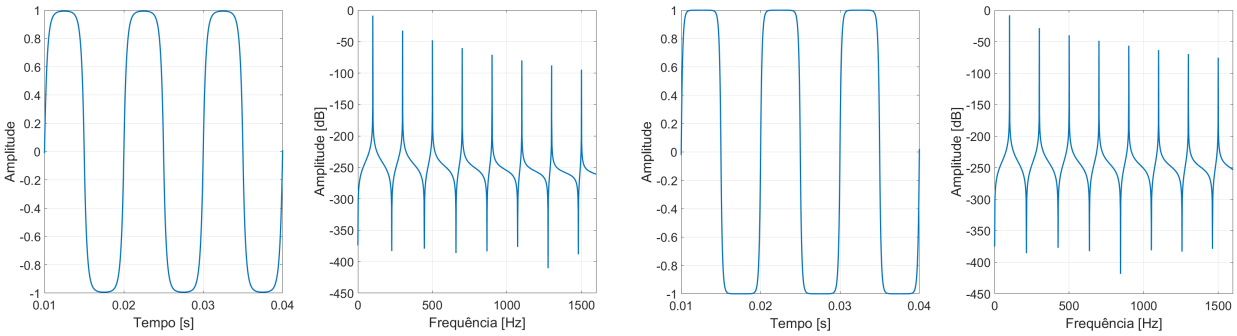
Os gráficos no domínio tempo e frequência para distorção harmônica simétrica de uma senoide de 100Hz são apresentados na Figura 33. Como é possível observar, o aumento do parâmetro *dist* faz com que a limitação do sinal se torne mais evidente, produzindo harmônicos mais intensos no sinal de saída.

Figura 33 – Gráficos dos sinais distorcidos simetricamente para diferentes valores do parâmetro *dist*.



(a) $dist = 1$.

(b) $dist = 2$.



(c) $dist = 5$.

(d) $dist = 10$.

Fonte: Própria, 2018.

4.1.2 Filtros de *crossover*

A função de transferência do filtro passa-altas analógico é dada por:

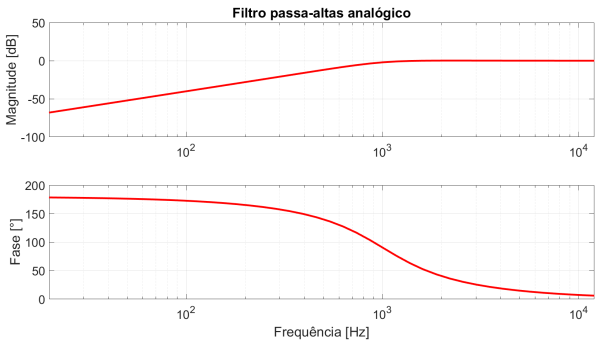
$$H_a(s) = \frac{s^2}{s^2 + 8.014 \cdot 10^3 s + 4.005 \cdot 10^{10}}\,, \tag{70}$$

e para o filtro passa-baixas:

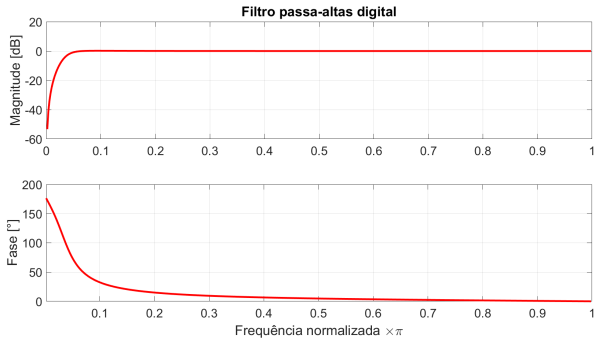
$$H_b(s) = \frac{1}{6.313 \cdot 10^{-8} s^2 + 3.148 \cdot 10^{-4} s + 1}\,. \tag{71}$$

Os diagramas de Bode dos filtros definidos pelas equações 70 e 71 são mostrados nas Figuras 34a e 34c, respectivamente. Nota-se que ambos os diagramas estão coerentes com o esperado pelos filtros, apresentando suas frequências de corte em aproximadamente 800Hz e um decaimento entorno dos -50db/década a partir dessa frequência.

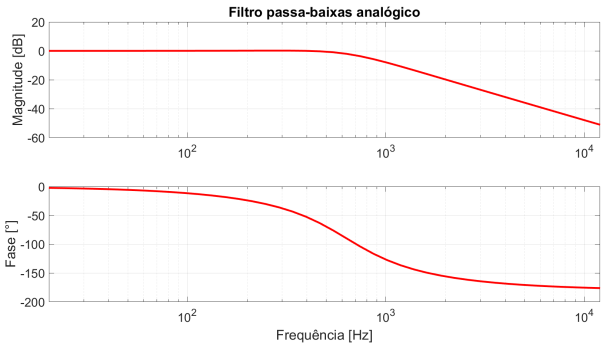
Figura 34 – Diagramas de Bode para filtros analógicos e suas discretizações.



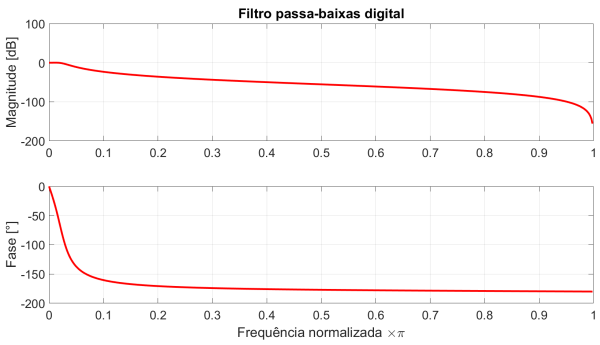
(a) Filtro passa-altas analógico.



(b) Filtro passa-altas digital.



(c) Filtro passa-baixas analógico.



(d) Filtro passa-baixas digital.

Fonte: Própria, 2018.

A função de transferência do filtro passa-altas digital é dada por:

$$H_a[z] = \frac{1 - 2z^{-1} + z^{-2}}{1.088 - 1.99z^{-1} + 0.92z^{-2}}\,, \tag{72}$$

e para o filtro passa-baixas:

$$H_b[z] = \frac{1 + 2z^{-1} + z^{-2}}{611.99 - 1158.507z^{-1} + 550.979z^{-2}}\,. \tag{73}$$

Os diagramas de Bode dos filtros digitais apresentados nas equações 72 e 73 são mostrados nas Figuras 34b e 34d, respectivamente.

Por fim, as equações de diferenças dos filtros digitais são descritas por

$$1.088 \cdot y_a[n] - 1.99 \cdot y_a[n-1] + 0.92 \cdot y_a[n-2] = 1 - 2 \cdot x[n-1] + x[n-2] \tag{74}$$

para o filtro passa-altas, e

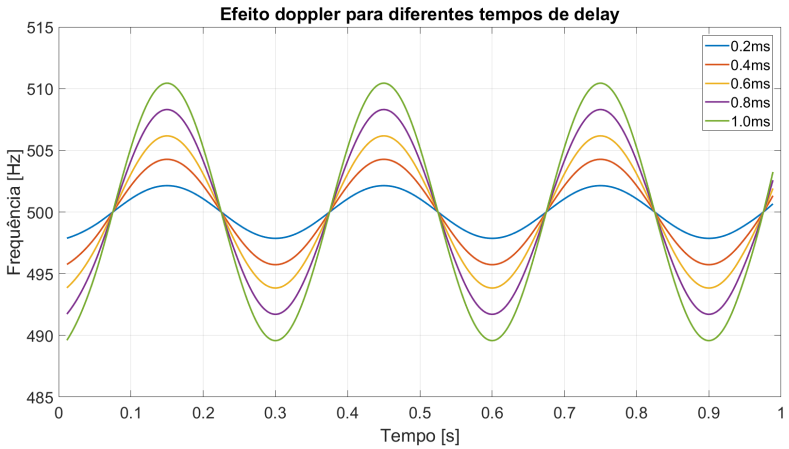
$$611.99 \cdot y_b[n] - 1158.507 \cdot y_b[n-1] + 550.979 \cdot y_b[n-2] = 1 + 2 \cdot x[n-1] + x[n-2]\,, \tag{75}$$

para o filtro passa-baixas.

4.1.3 Efeito Doppler

Os resultados da variação em frequência para um senoide de 500Hz em relação a variação no tempo de atraso estão mostrados na Figura 35. Atenta-se ao fato da frequência do sinal de saída variar senoidalmente em torno da frequência fundamental da senoide. Nota-se, também, que o tempo de atraso é proporcional a variação de frequência do sinal de saída, fazendo com que maiores valores de atraso resultem em uma excursão maior de frequência na saída.

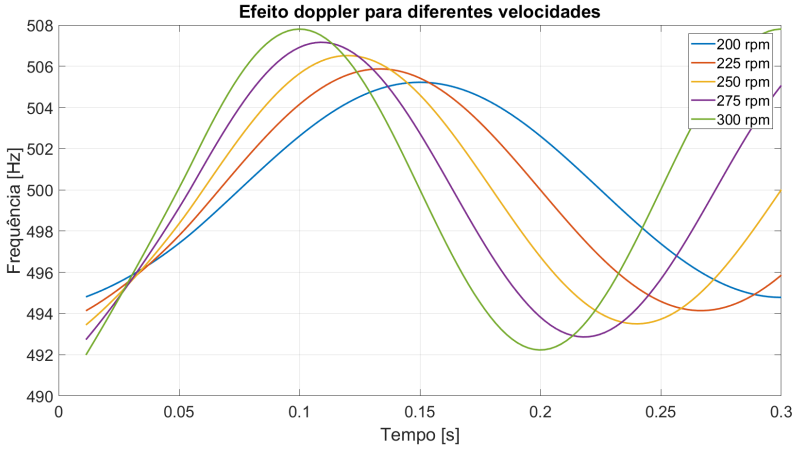
Figura 35 – Variação na frequência de uma senoide de 500Hz devido a variação no tempo de atraso para o Efeito Doppler.



Fonte: Própria, 2018.

Variando-se a velocidade do alto-falante, é possível se obter o gráfico da Figura 36 para a excursão da frequência do sinal de saída para uma entrada senoidal de 500Hz. Verifica-se que a velocidade de rotação do alto-falante é proporcional a variação de frequência assim como ao período dessa variação. Ou seja, a medida em que a velocidade do alto-falante aumenta, a excursão de frequência na saída aumenta e o período em que essa excursão ocorre diminui.

Figura 36 – Variação na frequência de uma senoide de 500Hz devido a variação da velocidade do alto-falante para o Efeito Doppler.

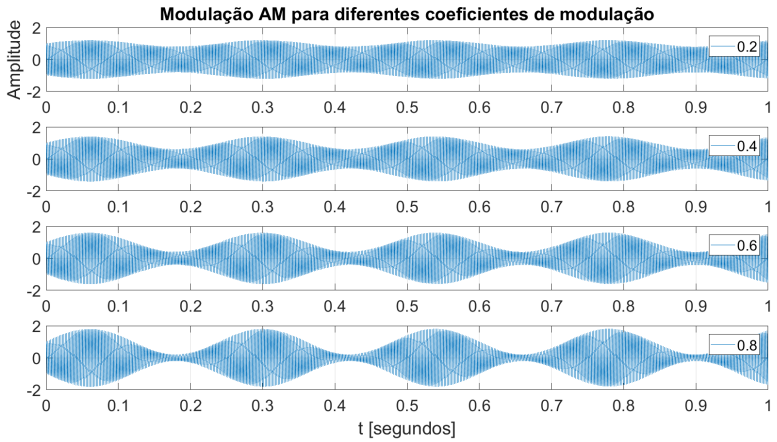


Fonte: Própria, 2018.

4.1.4 Modulação em amplitude

Os efeitos da variação do coeficiente de modulação para uma entrada senoidal de 500Hz estão mostrados na Figura 37. Verifica-se que a variação do coeficiente de modulação se reflete na intensidade dos vales e picos do sinal de saída.

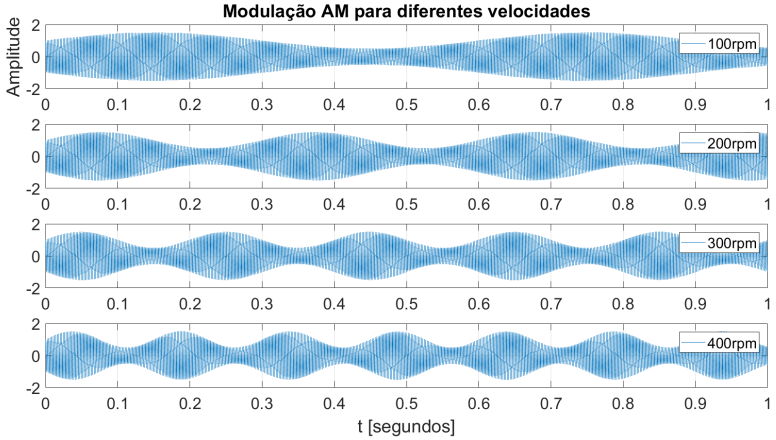
Figura 37 – Variação no sinal de saída para entrada senoidal de 500Hz devido a variação do coeficiente de modulação para a modulação em amplitude.



Fonte: Própria, 2018.

Os efeitos da variação da velocidade de rotação do alto-falante estão apresentados na Figura 38. Nota-se que a variação da velocidade altera a frequência de surgimento dos vales e picos no sinal de saída.

Figura 38 – Variação no sinal de saída para entrada senoidal de 500Hz devido a variação da velocidade do alto-falante para a modulação em amplitude.

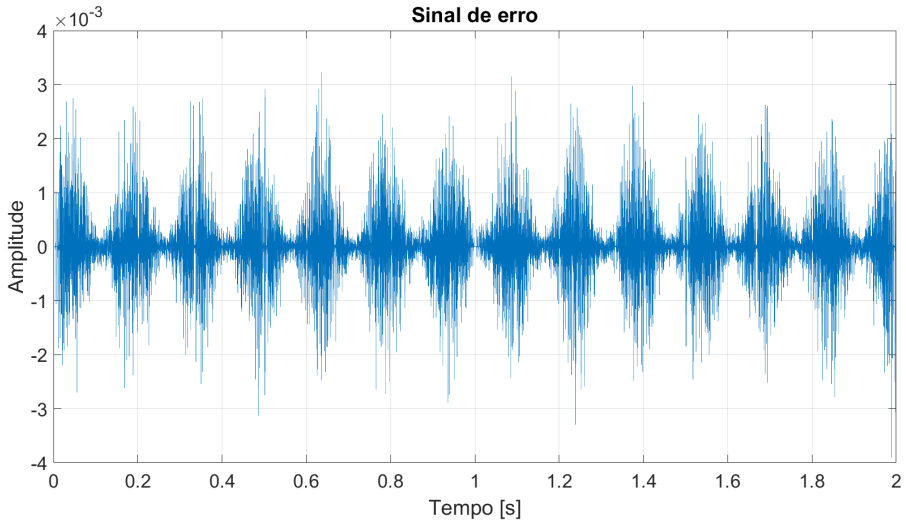


Fonte: Própria, 2018.

4.2 Simulação da quantização de 12bits e formato *float*

O sinal de erro obtido pela diferença da simulação em MATLAB com valores contínuos e formato numérico *double* em relação a um microcontrolador com quantização de 12bits e formato numérico *float* de 32bits é apresentado na Figura 39. O fato do sinal de erro parecer oscilar é devido a modulação em amplitude acentuada utilizada na simulação do sistema completo. Verificou-se que o erro absoluto máximo foi de $3.90 \cdot 10^{-3}$ e o erro quadrático médio teve valor $5.84 \cdot 10^{-8}$.

Figura 39 – Sinal de erro gerado pela subtração entre os sinais de precisão *double* e *float* com quantização de 12bits no sistema completo.



Fonte: Própria, 2018.

4.3 Validação das aproximações polinomiais

O polinômio que melhor aproxima a função exponencial no intervalo [-1,1] é dado por:

$$e^x \approx -0.0002x^7 + 0.014x^6 - 0.008x^5 + 0.041x^4 + 0.17x^3 + 0.5x^2 + x + 1\,. \tag{76}$$

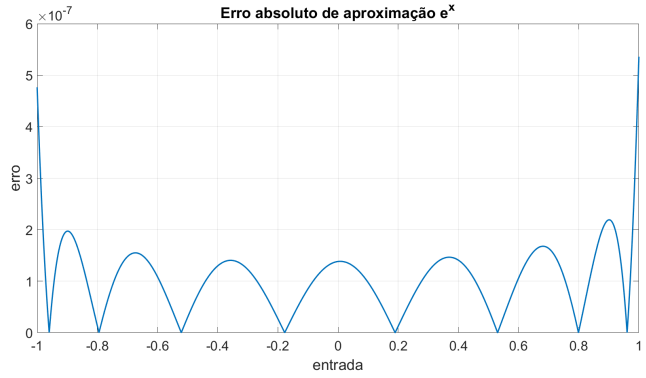
A Figura 40a apresenta o gráfico do erro absoluto da diferença entre a função exponencial implementada pelo MATLAB e sua aproximação. O valor máximo do erro absoluto foi de $5.36 \cdot 10^{-7}$ e o erro quadrático médio foi de $1.51 \cdot 10^{-14}$.

O polinômio que melhor aproxima o cosseno no intervalo $[-\frac{\pi}{2}, \frac{\pi}{2}]$ é dado por:

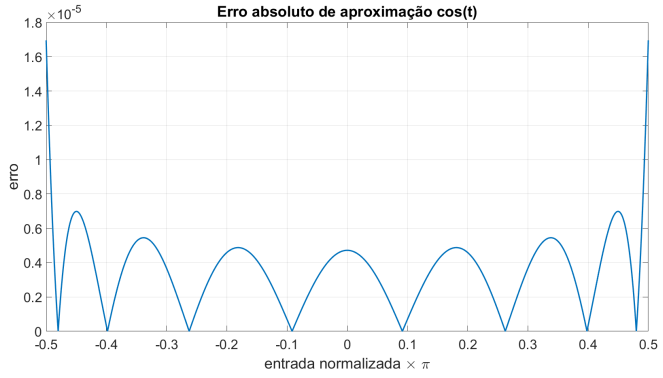
$$\cos(x) \approx -0.0013x^6 + 0.0415x^4 - 0.4999x^2 + 1\,. \tag{77}$$

A Figura 40b apresenta o gráfico do erro absoluto da diferença entre o cosseno implementado pelo MATLAB e sua aproximação. O valor máximo do erro absoluto foi de $1.69 \cdot 10^{-5}$ e o erro quadrático médio foi de $1.73 \cdot 10^{-11}$.

Figura 40 – Erro absoluto entre a diferença das funções implementadas em MATLAB e pela aproximação polinomial de ordem 7.



(a) Erro absoluto da aproximação da exponencial.



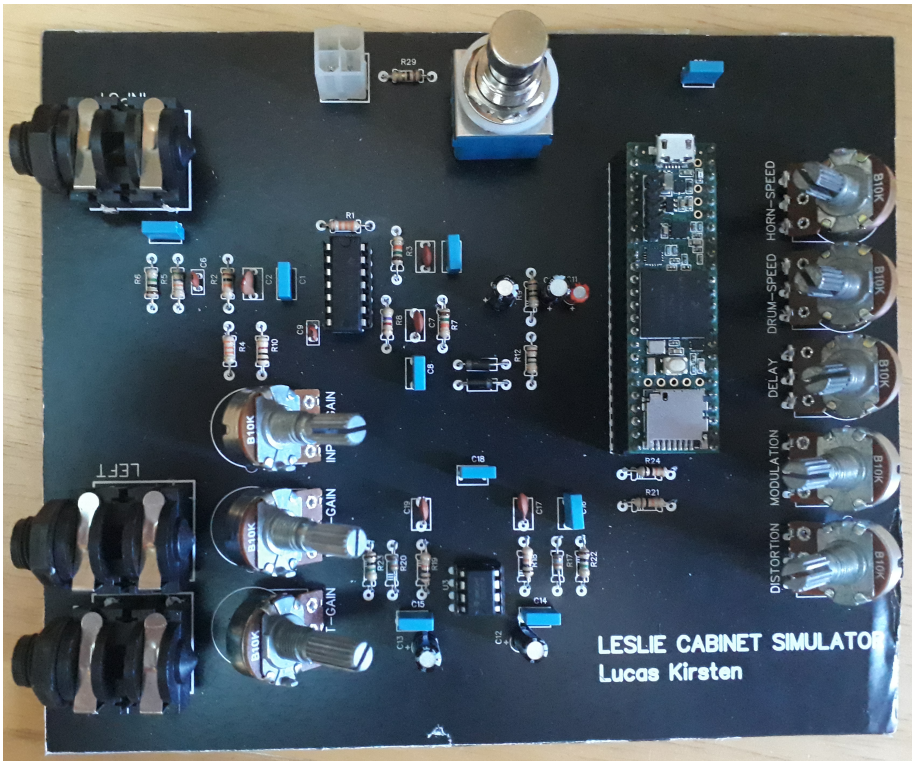
(b) Erro absoluto da aproximação do cosseno.

Fonte: Própria, 2018.

4.4 Placas de circuito desenvolvidas

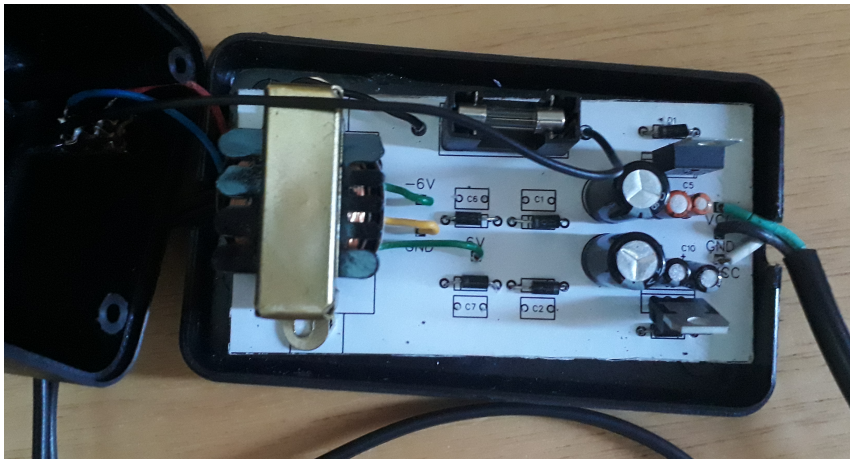
As Figuras 41 e 42 apresentam as placas desenvolvidas para o pedal e para a alimentação, respectivamente.

Figura 41 – Placa do pedal desenvolvido.



Fonte: Própria, 2018.

Figura 42 – Placa do circuito de alimentação do pedal.



Fonte: Própria, 2018.

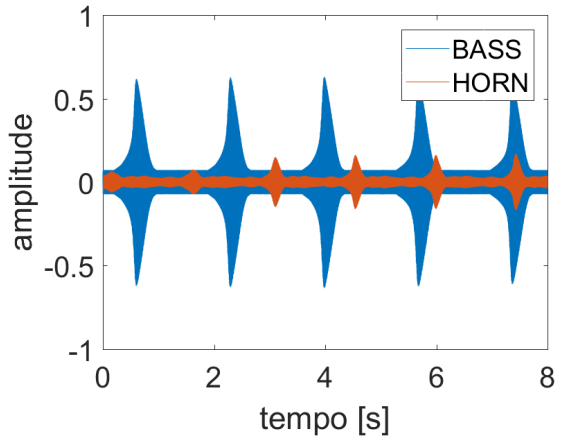
4.5 Comparação do pedal com a RT-20

A Figura 43 apresenta a comparação dos sinais da RT-20 e do pedal nos modos lento e rápido com senoide de 250Hz para o *BALANCE* em *BASS* e *HORN*. Na Figura 44 é apresentada a comparação dos sinais da RT-20 e do pedal oriundos do tambor no domínio do tempo e frequência.

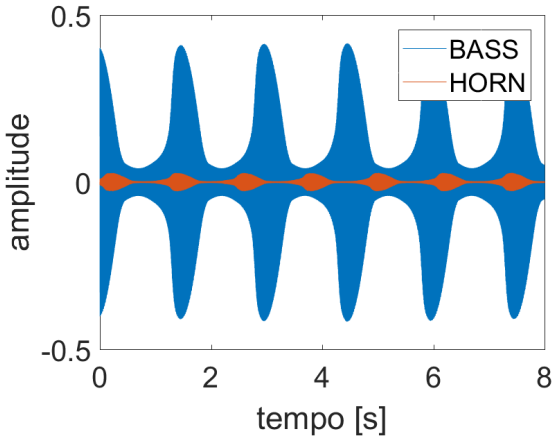
A Figura 45 apresenta a comparação dos sinais da RT-20 e do pedal nos modos lento e rápido com senoide de 2.5kHz para o *BALANCE* em *BASS* e *HORN*. Na Figura 46 é apresentada a comparação dos sinais da RT-20 e do pedal oriundos das cornetas no domínio do tempo e frequência.

Por fim, a Figura 47 apresenta a comparação dos sinais no domínio tempo e frequência da RT-20 e do pedal nos modos lento e rápido com onda triangular de 220Hz para o *BALANCE* no meio (ou seja, recebendo sinal de ambos os alto-falantes).

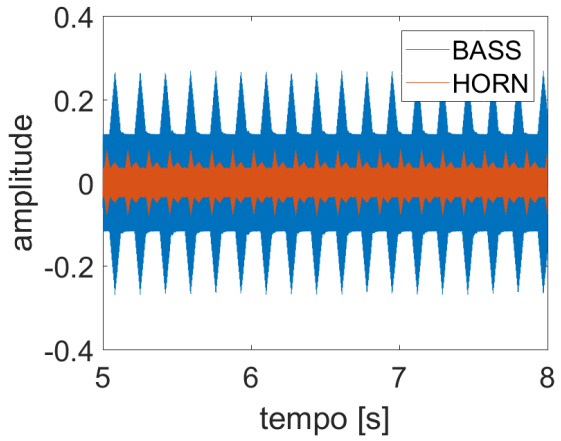
Figura 43 – Comparação dos sinais da RT-20 e do pedal nos modos lento e rápido com senoide de 250Hz.



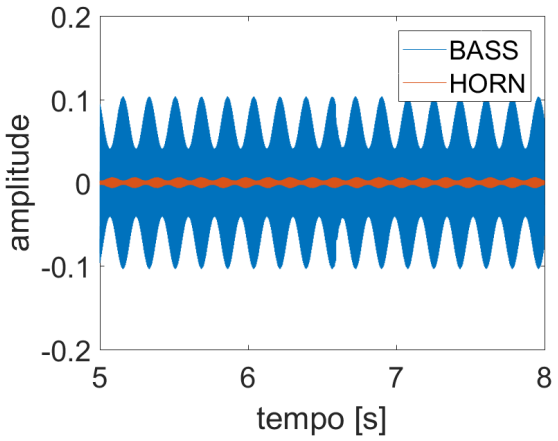
(a) Sinais da RT-20 no modo lento.



(b) Sinais do pedal no modo lento.



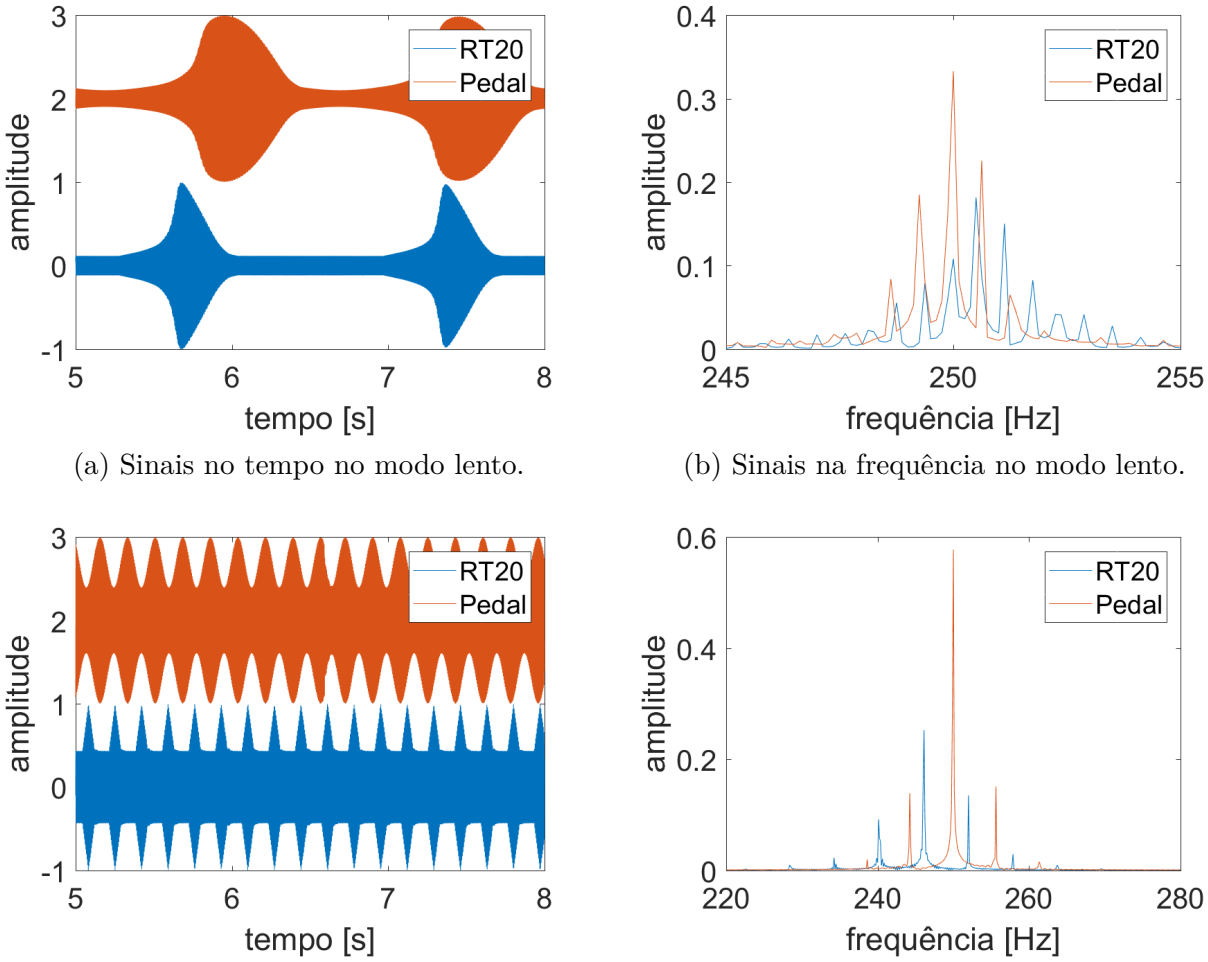
(c) Sinais da RT-20 no modo rápido.



(d) Sinais do pedal no modo rápido.

Fonte: Própria, 2018.

Figura 44 – Comparação dos sinais oriundos do tambor da RT-20 e do pedal nos modos lento e rápido com senoide de 250Hz no domínio tempo e frequência.



(a) Sinais no tempo no modo lento.

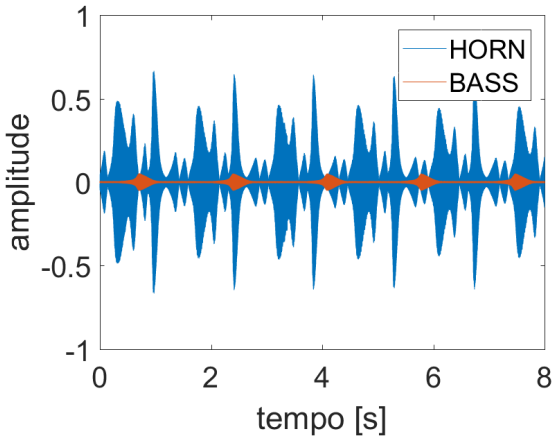
(b) Sinais na frequência no modo lento.

(c) Sinais no tempo no modo rápido.

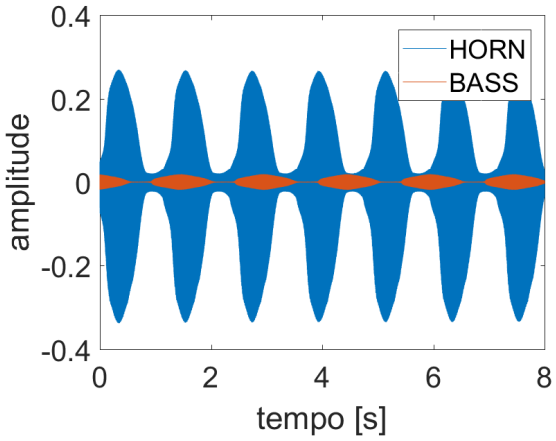
(d) Sinais na frequência no modo rápido.

Fonte: Própria, 2018.

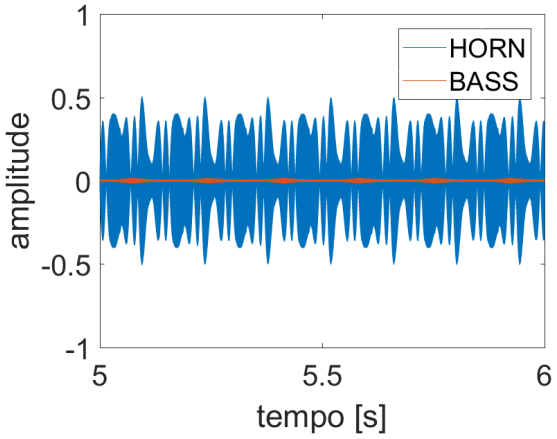
Figura 45 – Comparação dos sinais da RT-20 e do pedal nos modos lento e rápido com senoide de 2.5kHz.



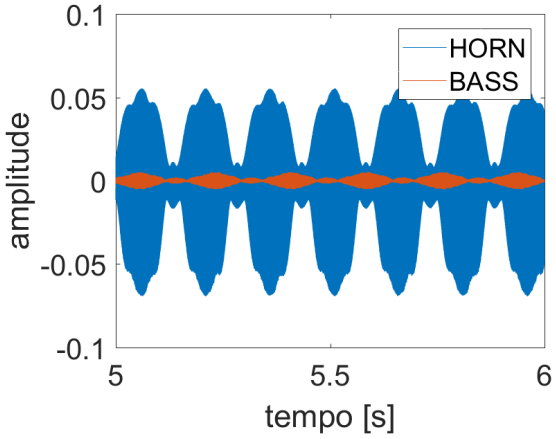
(a) Sinais da RT-20 no modo lento.



(b) Sinais do pedal no modo lento.



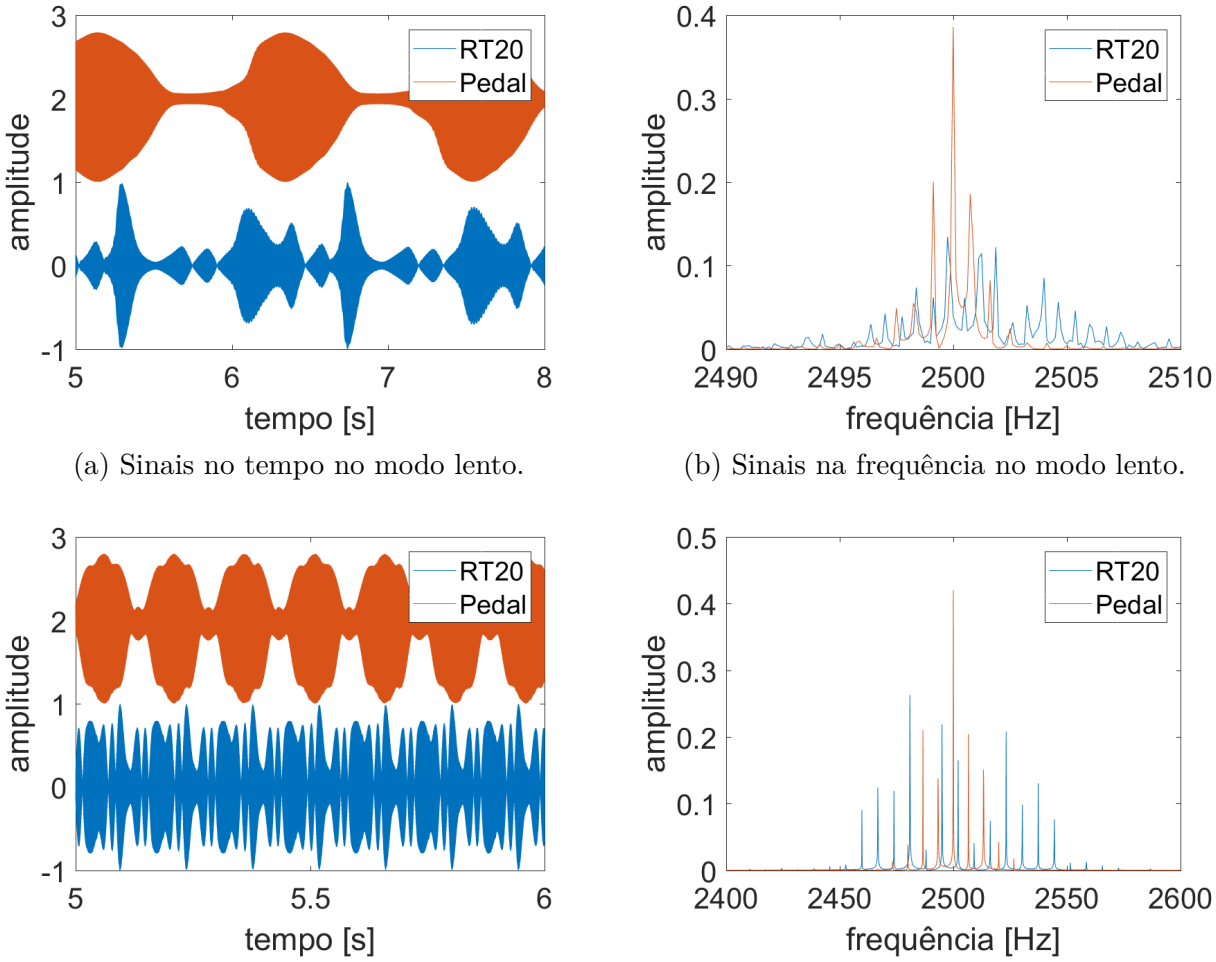
(c) Sinais da RT-20 no modo rápido.



(d) Sinais do pedal no modo rápido.

Fonte: Própria, 2018.

Figura 46 – Comparação dos sinais oriundos do tambor da RT-20 e do pedal nos modos lento e rápido com senoide de 2.5kHz no domínio tempo e frequência.



(a) Sinais no tempo no modo lento.

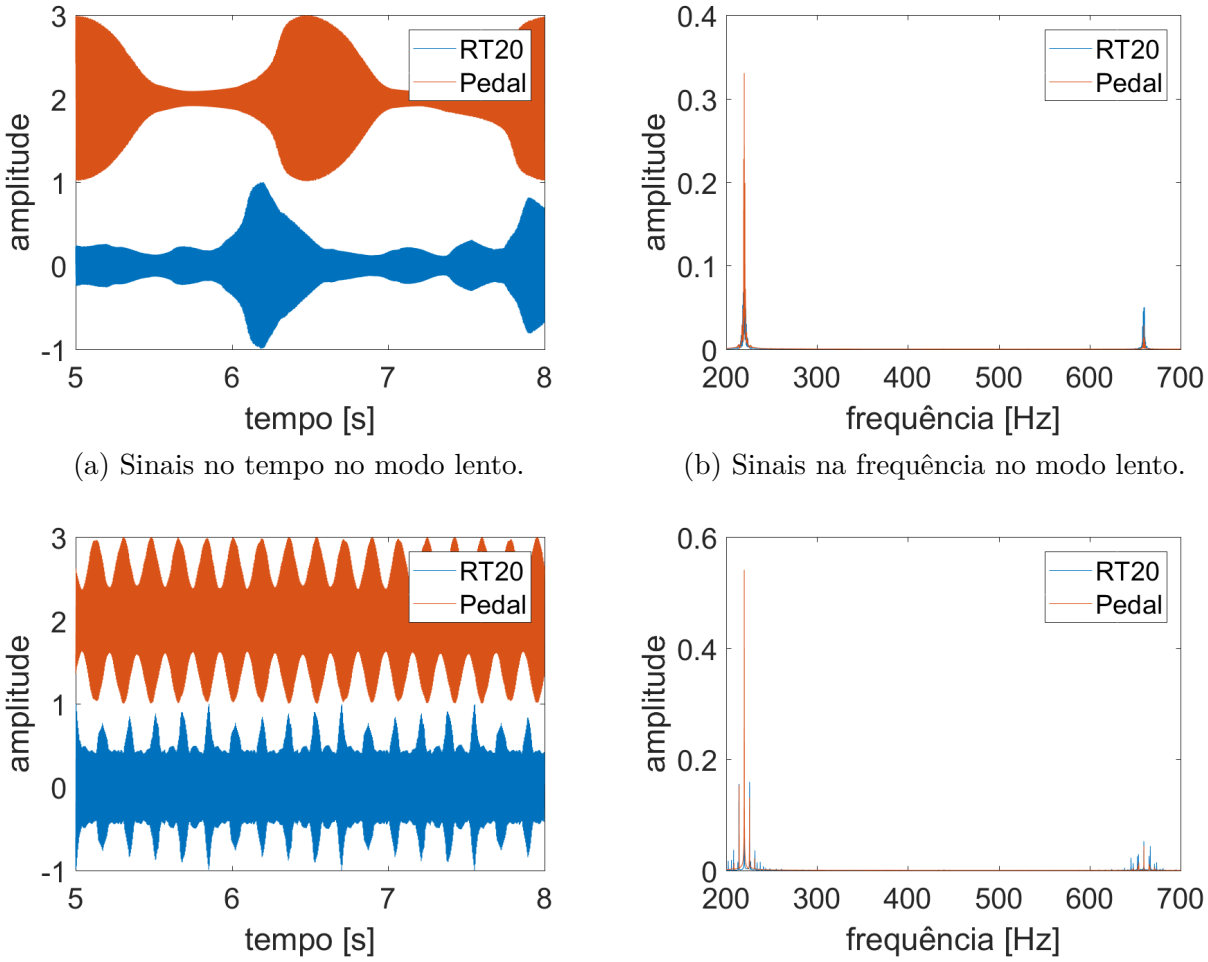
(b) Sinais na frequência no modo lento.

(c) Sinais no tempo no modo rápido.

(d) Sinais na frequência no modo rápido.

Fonte: Própria, 2018.

Figura 47 – Comparação dos sinais da RT-20 e do pedal nos modos lento e rápido com onda triangular de 220Hz no domínio tempo e frequência.



(a) Sinais no tempo no modo lento.

(b) Sinais na frequência no modo lento.

(c) Sinais no tempo no modo rápido.

(d) Sinais na frequência no modo rápido.

Fonte: Própria, 2018.

5 Discussões

A análise das simulações dos efeitos em MATLAB mostra a viabilidade de suas implementações - conforme esperado pela caracterização física e discretização dos mesmos. Agregado a isso, a simulação de tais efeitos com quantização de 12bits e formato *float* de 32bits ilustra que o erro será muito pequeno e não deverá causar um efeito significativo na sonoridade final ao serem executados pelo Teensy 3.6. O mesmo se conclui para as aproximações polinomiais da exponencial e do cosseno usadas para os cálculos da distorção, do efeito Doppler e da modulação em amplitude, respectivamente.

Com base nos comparativos dos efeitos produzidos pelo pedal com a senoide abaixo da frequência de *crossover* e outra com frequência acima, é possível verificar o correto funcionamento da discretização do circuito de filtragem, pois o sinal grave aparece atenuado nas cornetas e o sinal agudo aparece atenuado no tambor. Comparando-se esses sinais com os obtidos pela RT-20, notou-se que, para a senoide de 250Hz, eles se mostraram mais parecidos que os da senoide de 2.5kHz - verificando-se isso tanto nos sinais no domínio tempo quanto na frequência. Para a senoide de 2.5kHz obteve-se uma forma de onda muito diferente daquela do pedal, além do fato da implementação usada pela RT-20 adicionar uma quantidade maior de harmônicos. Entretanto, deve-se lembrar de dois aspectos na implementação proposta: (1) as funções para o efeito Doppler e modulação em amplitude foram aproximadas para situações em que o ouvinte (ou os microfones) se encontram relativamente longe da caixa; (2) a implementação foi feita considerando a caixa Leslie em espaço livre, ou seja, sem as reflexões de som que surgem dentro do gabinete. Para velocidade lenta de rotação dos alto-falantes e sinais abaixo da frequência de *crossover*, então, as aproximações tiveram mais proximidade. Essa conclusão também é verificada ao se analisar as formas de onda e o espectro de frequências para a entrada triangular (que simularia uma nota musical). Como a maior frequência fundamental de uma guitarra está na faixa dos 1kHz, é possível concluir, então, que o pedal consegue fazer a simulação de forma aproximada para a maior faixa de notas e suas discrepâncias surgem de forma mais notória para os harmônicos de mais alta ordem.

Escutando os sons produzidos pelo pedal e pela RT-20 sob os sinais de teste não foi possível encontrar muita diferença. Aparentemente, as duas implementações seriam capazes de entregar resultados muito semelhantes e as divergências se tornaram mais visíveis ao se analisar com mais detalhes os sinais produzidos. Como tratam-se de implementações diferentes, seria evidente que tais diferenças surgissem.

6 Conclusões

O presente trabalho teve como objetivo principal o desenvolvimento de um protótipo de pedal digital capaz de simular a caixa Leslie microfonada. Para atingir tal objetivo, foram caracterizados separadamente os efeitos marcantes da caixa, sendo eles: a distorção harmônica do circuito de pré-amplificação e amplificação, o filtro de *crossover* para separação dos sons agudos e graves entorno dos 800Hz, o efeito Doppler e a modulação em amplitude provocados pela rotação dos alto-falantes. Ao se discretizar tais efeitos, foi possível o desenvolvimento de algoritmos em MATLAB capazes de os reproduzir separadamente, e a definição dos parâmetros de interesse para que o usuário do pedal pudesse variá-los. Esses parâmetros são: o nível de distorção, a velocidade dos alto-falantes, o coeficiente de modulação em amplitude e o tempo de atraso para a intensidade do desvio de frequência.

O uso da placa de desenvolvimento Teensy 3.6 foi validada através da simulação do sistema para uma quantização dos dados e redução do número de bits para o cálculo numérico em relação ao sistema desenvolvido no MATLAB. Tal validação foi crucial, pois elucidou que tais limitações não causariam efeitos significativos para a sonoridade final do pedal. Esse tipo de validação também foi verificada nas aproximações polinomiais da função exponencial e do cosseno.

A comparação entre os sinais oriundos do pedal desenvolvido com o pedal RT-20 da BOSS demonstrou uma maior semelhança para aqueles sinais cujas frequências estejam abaixo da de *crossover* (800Hz). As desconformidades foram atribuídas, então, as aproximações utilizadas na caracterização do efeito Doppler e da modulação em amplitude, além do fato de não ter sido implementado as múltiplas reflexões que o som tem dentro do gabinete da Leslie. Dessa forma, acredita-se que ao se utilizar as equações não aproximadas para tais efeitos (equações 20 e 33), agregado a técnicas de implementações para as múltiplas reflexões - como propostas em Kronland-Martinet e Voinier (2008), e Smith et al. (2002), os sinais resultantes do pedal se mostrariam mais semelhantes aos do pedal RT-20.

Em conclusão, o pedal desenvolvido se mostrou funcional e robusto, da mesma forma que o projeto da fonte de alimentação. Apesar de haver diferenças entre ele e a RT-20, ainda assim sua sonoridade se mostrou muito parecida com a de uma caixa Leslie real. Agregado a isso, ao se alterar os parâmetros, torna-se possível simular efeitos de diferentes modelos de Leslie, além de permitir que o usuário obtenha outros efeitos oriundos dela. Por fim, as especificações técnicas do pedal e da fonte são apresentadas na Tabela 7.

Tabela 7 – Especificações do pedal e da fonte desenvolvida.

Pedal	
Impedância de entrada	1MΩ
Impedância de saída	10kΩ máximo
Nível de sinal de entrada	-17.78dBu
Nível de sinal de saída	3.56dBu máximo
Frequência de amostragem	48kHz
Número de bits	12bits
Fonte	
Entrada de tensão	Bivolt: 127V ou 220V
Níveis de tensão da saída	-5V, 0V e 5V
Consumo de corrente	200mA máximo

Fonte: Própria, 2018.

Referências

THE THEATRE ORGAN. Unearthing the mysteries of the Leslie cabinet. Disponível em: <http://www.theatreorgans.com/hammond/faq/mystery/mystery.html>. Acesso em: 17 de abril de 2018.

CULT JAZZ. A Caixa Mágica Leslie. Disponível em: <http://www.cultjazz.com.br/p/a-caixa-leslie.html>. Acesso em: 17 de abril de 2018.

STRYMON. Rotary Speaker Technology White Paper. Disponível em: <https://www.strymon.net/rotary-speaker-technology-white-paper/>. Acesso em: 17 de abril de 2018.

CAPTAIN FOLDBACK'S. 2nd generation speakers. Disponível em: <http://www.captain-foldback.com/Leslie_sub/chorale2.htm>. Acesso em: 8 de dezembro de 2018.

North Suburban HAMMOND ORGAN Society. LESLIE SPEAKERS. Disponível em: <http://www.nshos.com/LES4.htm>. Acesso em: 8 de dezembro de 2018.

HAMMOND USA. Leslie Guitar Artists. Disponível em: <http://hammondorganco.com/artists/guitar-leslie-artists/>. Acesso em: 7 de julho de 2018.

J-ROY. Instruments/Backline. Disponível em: <http://www.johnroysound.com/new-page-1/>. Acesso em: 10 de novembro de 2018.

CAVALLI MUSICA. AMPLIFICATORE LESLIE 760-N 90W 3 Speed Chorale/Stop/Tremolo. Disponível em: <https://www.cavallimusica.com/amplificatore-leslie-760-n-90w-3-speed-chorale-stop-tremolo.html>. Acesso em: 10 de novembro de 2018.

CAPTAIN FOLDBACK'S. Leslie 830. Disponível em: <http://www.captain-foldback.com/Leslie-_sub/leslie_830.htm>. Acesso em: 10 de novembro de 2018.

ELECTRO-HARMONIX. Lester G - Deluxe Rotary Speaker. Disponível em: <https://www.ehx.com/products/lester-g>. Acesso em: 7 de julho de 2018.

STRYMON. Lex Rotatory. Disponível em: <https://www.strymon.net/products/lex-/>. Acesso em: 7 de julho de 2018.

NEO INSTRUMENTS. Mini Vent 2. Disponível em: <http://www.neo-instruments.de/en>. Acesso em: 7 de julho de 2018.

BOSS. RT-20. Disponível em: <https://www.boss.info/br/products/rt-20/>. Acesso em: 7 de julho de 2018.

WIKIPEDIA. Leslie speaker. Disponível em: <https://en.wikipedia.org/wiki/Leslie_speaker>. Acesso em: 14 de dezembro de 2018.

HARMONY CENTRAL. Article: Rotatory Speaker Cabinet Miking. Disponível em: <http://www.harmonycentral.com/articles/rotary-speaker-cabinet-miking>. Acesso em: 20 de julho de 2018.

ZÖLZER, Udo (Ed.). DAFX: digital audio effects. John Wiley & Sons, 2011.

PAKARINEN, Jyri; YEH, David T. A review of digital techniques for modeling vacuum-tube guitar amplifiers. Computer Music Journal, v. 33, n. 2, p. 85-100, 2009.

THE NATIONAL MUSEUM OF COMPUTING. A temporary display of Thermionic Valves. Disponível em: <http://www.tnmoc.org/news/notes-museum/temporary-display-thermionic-valves>. Acesso em: 8 de dezembro de 2018.

DEMPWOLF, Kristjan; HOLTERS, Martin; ZÖLZER, Udo. Discretization of parametric analog circuits for real-time simulations. In: Proceedings of the 13th international conference on digital audio effects (DAFx'10). 2010.

YEH, David T. et al. Numerical methods for simulation of guitar distortion circuits. Computer Music Journal, v. 32, n. 2, p. 23-42, 2008.

BENDIKSEN, Ragnar. Digitale lydeffekter. Diplomoppgave i akustikk, Norges teknisk-naturvitenskapelige universitet, Institutt for teleteknikk, Trondheim, 1997.

TERMAN, Frederick Emmons. Radio engineers' handbook. 1943.

BENTON ELECTRONICS. Servicing the Leslie 122 Amplifier. Disponível em: <http://bentonelectronics.com/servicing-the-leslie-122-amplifier/>. Acesso em: 10 de novembro de 2018.

SMITH, Julius et al. Doppler simulation and the Leslie. In: Proceeding of the Workshop on Digital Audio Effects, DAFx-02, Hamburg, Germany. 2002. p. 188-191.

ANDERTON, Craig. The digital delay handbook. Music Sales Corporation, 1985.

HAYES, Monson H. Schaum's outline of digital signal processing. McGraw-Hill, Inc., 1998.

OPPENHEIM, Alan V. Discrete-time signal processing. Pearson Education India, 1999.

KARKI, Jim. Active low-pass filter design. Texas Instruments Application Report, 2000.

Statistics How To. Mean Squared Error Definition. Disponível em: <https://www.statisticshowto.datasciencecentral.com/mean-squared-error/>. Acesso em: 8 de dezembro de 2018.

PJRC. Teensy USB Development Board. Disponível em: <https://www.pjrc.com/store/teensy36.html>. Acesso em: 20 de julho de 2018.

PJRC. Teensy Technical Specifications. Disponível em: <https://www.pjrc.com/teensy/techspecs.html>. Acesso em: 10 de novembro de 2018.

PJRC. Teensy 3.6 Pins. Disponível em: <https://www.pjrc.com/teensy/pinout.html>. Acesso em: 10 de novembro de 2018.

ARM Cortex-M4 32b. Datasheet disponível em: <https://www.pjrc.com/teensy/K66P14-4M180SF5V2.pdf>. Acesso em: 09 de outubro de 2018.

PJRC. Teensyduino. Disponível em: <https://www.pjrc.com/teensy/teensyduino.html>. Acesso em: 20 de julho de 2018.

CIRCUITAR. Setup & Loop. Disponível em: <https://www.circuitar.com.br/projetos-/setup-loop/>. Acesso em: 8 de dezembro de 2018.

GITHUB. Teensy 3.x/LC ADC implementation. Disponível em <https://github.com/pedvide/ADC>. Acesso em: 30 de julho de 2018.

FORUM PJRC. Teensy 3.6 ADC resolution. Disponível em: <https://forum.pjrc.com/threads/41911-Teensy-3-6-ADC-resolution>. Acesso em: 10 de novembro de 2018.

PJRC. IntervalTimer. Disponível em: <https://www.pjrc.com/teensy/td_timing_-IntervalTimer.html>. Acesso em: 30 de julho de 2018.

DAILEY, Denton J. Electronics for guitarists. Springer Science & Business Media, 2012.

ProSoundWeb. Answers To Frequently Asked Questions About Impedance & Impedance Matching In Sound Systems. Disponível em: <https://www.prosoundweb.com/topics/audio/answers_to_frequency_asked_questions_about_impedance_impedance_matching_in_/>. Acesso em: 8 de dezembro de 2018.

OKAWA Electric Design. Sallen-Key Low-pass Filter Design Tool. Disponível em: <http://sim.okawa-denshi.jp/en/OPseikiLowkeisan.htm>. Acesso em: 10 de novembro de 2018.

TEXAS INSTRUMENTS. TL07xx Low-Noise JFET-Input Operational Amplifiers. Setembro de 1978 - revisado em julho de 2017.

TEXAS INSTRUMENTS. uA7800 SERIES POSITIVE-VOLTAGE REGULATORS. Maio de 1976 - revisado em maio de 2003.

NATIONAL SEMICONDUCTOR. LM79XX Series 3-Terminal Negative Regulators. Novembro de 1994.

EasyEDA. Disponível em: <https://easyeda.com/>. Acesso em: 30 de outubro de 2018.

Audacity. Disponível em: <https://www.audacityteam.org/>. Acesso em: 9 de dezembro de 2018.

KRONLAND-MARTINET, Richard; VOINIER, Thierry. Real-time perceptual simulation of moving sources: application to the Leslie cabinet and 3D sound immersion. EURASIP Journal on Audio, Speech, and Music Processing, v. 2008, p. 7, 2008.

APÊNDICE A – CÓDIGOS EM MATLAB

Encontram-se aqui os códigos separados dos efeitos da distorção, filtragem, efeito Doppler e modulação em amplitude em MATLAB para simulação dos mesmos.

A.1 Distorção

```
function [ y ] = distortion( x, dist )
% distorcao simetrica
% x : sinal de entrada
% dist : nivel de distorcao
% y : sinal de saida

x=x*dist;
L = length(x);
y = zeros(L,1);

for n=1:L
        if x(n)>=0
                y(n)=1-exp(-x(n));
        else
                y(n)=-1+exp(x(n));
        end
end

end
```

A.2 Filtros

```
function [y_baixa, y_alta] = filtros(x, Fs, Fc, n_baixa, d_baixa, num_alta, den_alta)
% y_baixa: sinal de saida aplicado ao filtro passa baixa
% y_alta: sinal de saida aplicado ao filtro passa alta
% x: sinal de entrada
% Fs: frequencia de amostragem
% Fc: frequencia de corte dos filtros (800Hz)
% n_baixa: numerador do filtro passa baixa
% d_baixa: denominador do filtro passa baixa
% n_alta: numerador do filtro passa alta
% d_alta: denominador do filtro passa alta

% Filtros
% Passa baixa
        % analogico
H_baixa = tf(n_baixa,d_baixa);
        % digital com bilinear
[num_baixa,den_baixa] = bilinear(n_baixa,d_baixa,Fs,Fc);

% Passa alta
        % analogico
H_alta = tf(n_alta, d_alta);
        % digital com bilinear
[num_alta,den_alta] = bilinear(n_alta,d_alta,Fs,Fc);
```

```
24
25 y_baixa = filter(num_baixa,den_baixa,x);
26 y_alta = filter(num_alta,den_alta,x);
27 end
```

A.3 Efeito Doppler

```
1  function [y] = doppler(x, vel, Width, Fs)
2  % x : sinal de entrada
3  % vel: velocidade de rotacao em rpm
4  % Width: tempo de delay
5  % Fs: frequencia de amostragem
6
7  L = length(x);
8
9  Modfreqc = vel/60; % velocidade em Hz
10 M = Modfreqc/Fs;
11
12 y = zeros(L,1);
13
14 % variaveis gerais para doppler
15 DELAY=round(Width*Fs);
16 LEN=2+DELAY*2;
17 Delayline=zeros(LEN,1);
18
19 % interpolacao linear
20 for n=1:L
21    MOD=sin(M*2*pi*n);
22    ZEIGER=1+DELAY*(1+MOD);
23    i=floor(ZEIGER);
24    frac=ZEIGER-i;
25    Delayline=[x(n);Delayline(1:LEN-1)];
26    y(n)=Delayline(i+1)*frac+Delayline(i)*(1-frac);
27 end
28 end
```

A.4 Modulação em amplitude

```
1  function [y] = am(x, vel, coef, Fs)
2  % x: sinal de entrada
3  % vel: velocidade em rpm
4  % coef: coeficiente de modulacao (0-1)
5  % Fs: frequencia de amostragem
6
7  Modfreqc = vel/60; % velocidade em Hz
8  M = Modfreqc/Fs;
9
10 L = length(x);
11 y = zeros(L,1);
12
13 for n=1:L
14    MOD=sin(M*2*pi*n);
15    y(n) = (1+coef*MOD)*x(n);
16 end
17 end
```

APÊNDICE B – CÓDIGO EM C++

Encontra-se aqui o código completo em C+++ implementado e programado através da biblioteca Teensyduino para o Teensy 3.6.

```
#include <ADC.h>

ADC *adc = new ADC(); // objeto do adc
IntervalTimer amostrador; //objeto do timer

// tamanho das arrays
#define IN 3      //entradada
#define OUTb 3    //saida passa baixas
#define OUTa 3    //saida passa altas
#define DELAY 100 //linhas de delay 2+round(1e-3*Fs)*2

// constantes
//essenciais
const float Fs = 48000;
const float Ts = 20.833333333;
//filtros
const float num_low[3] = {0.001633649788549, 0.003267299577098, 0.001633649788549};
const float den_low[3] = {1, -1.893773275652651, 0.900307874806846};
const float num_high[3] = {0.919203457701762, -1.838406915403524, 0.919203457701762};
const float den_high[3] =  {1, -1.830400301837787, 0.846413528969261};

// declaração das arrays
// entrada (atual e antigas)
float in[IN], *wptrIn = in;
// filtros
float outL[IN]={0}, *wptrOutL = outL;
float outH[IN]={0}, *wptrOutH = outH;
// linhas de delay
float Delayline_lowR[DELAY]={0}, *wptrLr = Delayline_lowR;
float Delayline_lowL[DELAY]={0}, *wptrLl = Delayline_lowL;
float Delayline_highR[DELAY]={0}, *wptrHr = Delayline_highR;
float Delayline_highL[DELAY]={0}, *wptrHl = Delayline_highL;

// ******************FUNçÕES PARA DELAYS******************
// coloca o valor de entrada na array
void putIn(float entrada){
        if ((wptrIn-in)>=IN) wptrIn-=IN;
        *(wptrIn++)=entrada;
}
void putOutL(float entrada){
        if ((wptrOutL-outL)>=IN) wptrOutL-=IN;
        *(wptrOutL++)=entrada;
}
void putOutH(float entrada){
        if ((wptrOutH-outH)>=IN) wptrOutH-=IN;
        *(wptrOutH++)=entrada;
}
void putDelayLowR(float entrada){
        if ((wptrLr-Delayline_lowR)>=DELAY) wptrLr-=DELAY;
        *(wptrLr++)=entrada;
```

```
51 }
52 void putDelayLowL(float entrada){
53         if ((wptrLl-Delayline_lowL)>=DELAY) wptrLl-=DELAY;
54         *(wptrLl++)=entrada;
55 }
56 void putDelayHighR(float entrada){
57         if ((wptrHr-Delayline_highR)>=DELAY) wptrHr-=DELAY;
58         *(wptrHr++)=entrada;
59 }
60 void putDelayHighL(float entrada){
61         if ((wptrHl-Delayline_highL)>=DELAY) wptrHl-=DELAY;
62         *(wptrHl++)=entrada;
63 }
64
65 // retorna o valor da array com o valor de delay setado na posicao
66 float getIn(int posicao){
67         if ( (wptrIn-in)-posicao < 0 ) posicao = IN+((wptrIn-in)-posicao);
68         else posicao = (wptrIn-in)-posicao;
69         return in[posicao];
70 }
71 float getOutL(int posicao){
72         if ( (wptrOutL-outL)-posicao < 0 ) posicao = IN+((wptrOutL-outL)-posicao);
73         else posicao = (wptrOutL-outL)-posicao;
74         return outL[posicao];
75 }
76 float getOutH(int posicao){
77         if ( (wptrOutH-outH)-posicao < 0 ) posicao = IN+((wptrOutH-outH)-posicao);
78         else posicao = (wptrOutH-outH)-posicao;
79         return outH[posicao];
80 }
81 float getDelayLowR(int posicao){
82         if ( (wptrLr-Delayline_lowR)-posicao < 0 ) posicao = DELAY+((wptrLr-
               Delayline_lowR)-posicao);
83         else posicao = (wptrLr-Delayline_lowR)-posicao;
84         return Delayline_lowR[posicao];
85 }
86 float getDelayLowL(int posicao){
87         if ( (wptrLl-Delayline_lowL)-posicao < 0 ) posicao = DELAY+((wptrLl-
               Delayline_lowL)-posicao);
88         else posicao = (wptrLl-Delayline_lowL)-posicao;
89         return Delayline_lowL[posicao];
90 }
91 float getDelayHighR(int posicao){
92         if ( (wptrHr-Delayline_highR)-posicao < 0 ) posicao = DELAY+((wptrHr-
               Delayline_highR)-posicao);
93         else posicao = (wptrHr-Delayline_highR)-posicao;
94         return Delayline_highR[posicao];
95 }
96 float getDelayHighL(int posicao){
97         if ( (wptrHl-Delayline_highL)-posicao < 0 ) posicao = DELAY+((wptrHl-
               Delayline_highL)-posicao);
98         else posicao = (wptrHl-Delayline_highL)-posicao;
99         return Delayline_highL[posicao];
100 }
101 // ************************************************
102
103 // **************CÁLCULO DISTORçÃO*********************
104 const float coefExp[8] =
        {0.999999861759424,0.999999926994482,0.500004968172645,0.166667735751766,
        0.041639414933251, 0.008329174011336, 0.001435883647857, 2.043272244298092e-4};
```

```
105 float exponencial(float x){
106         float en2=x*x; float en3=en2*x; float en4=en3*x;
107         float en5=en4*x; float en6=en5*x; float en7=en6*x;
108         return coefExp[0]+x*coefExp[1]+en2*coefExp[2]+en3*coefExp[3]+en4*coefExp[4]+en5*
                coefExp[5]+en6*coefExp[6]+en7*coefExp[7];
109 }
110
111 float distortion(float x){
112         if (x>=0) return 1-exponencial(-x);
113         else return -1+exponencial(x);
114 }
115 // ************************************************
116
117 // **************CÁLCULO COSSENO*****************************************
118 // angulos
119 const float pi = 3.14159265359;
120 const float pi_2 = pi/2;
121 const float pi3_2 = 3*pi/2;
122 const float pi2 = 2*pi;
123 // coeficientes
124 const float coefCos[4] = {-0.49999,0.0415,-0.0013};
125 // calcula o valor de cosseno por polinomio
126 float cosseno(float ang){
127         // valida o angulo
128         long mul = ang/pi2;
129         if (mul>0) ang = ang-mul*pi2;
130         if (ang>pi3_2) ang = ang-pi2;
131         else if (ang>pi_2) ang = pi-ang;
132         // calcula o cosseno com os coeficientes
133         float ang2=ang*ang, ang4=ang2*ang2, ang6=ang4*ang2;
134         return 1+ang2*coefCos[0]+ang4*coefCos[1]+ang6*coefCos[2];
135 }
136 // ************************************************************************
137
138 // ****************VIBRATO*************************
139 // potenciometros (valores iniciais)
140 //(velocidades cornetas e woofer, delay, coef. AM, distorção)
141 float Mc=400.0/(60*Fs), Mw=340.0/(60*Fs), Width=1.0/10000*Fs, coefMod=0.8, dist=1;
142
143 // variaveis auxiliares
144 volatile long it=0; //iterator que varia o cosseno
145 volatile int p;
146 volatile uint16_t sample;
147 volatile float insys, in1, in2, low, low1, low2, high, high1, high2;
148 volatile float outLr, outLl, outHr, outHl, MOD, ZEIGER, frac;
149 volatile float outVibR, outVibL, outLeft, outRight;
150 volatile float POT;
151 volatile long tInit,tEnd;
152
153 // faz aquisição de som
154 void getSample(void){
155         //tInit = micros();
156         sample = adc->analogRead(A13,ADC_1);
157         // escreve os valores para o efeito nos DACs
158         analogWrite(A21, (uint16_t)((outLeft+1)*2048));
159         analogWrite(A22, (uint16_t)((outRight+1)*2048));
160         // converte o valor amostrado para -1 e 1
161         insys = (float)(sample-2048.0)/4096.0;
162
163         // distorcao
```

```
164         insys = distortion(dist*insys);
165         putIn(insys);
166
167         // filtragem
168         in1 = getIn(2); in2= getIn(3);
169         low1 = getOutL(1); low2 = getOutL(2);
170         high1 = getOutH(1); high2 = getOutH(2);
171         low = den_low[0]*(num_low[0]*insys+num_low[1]*in1+num_low[2]*in2-den_low[1]*low1-
                den_low[2]*low2);
172         high = den_high[0]*(num_high[0]*insys+num_high[1]*in1+num_high[2]*in2-den_high
                [1]*high1-den_high[2]*high2);
173         putOutL(low);
174         putOutH(high);
175
176         // woofer
177         MOD = cosseno(Mw*pi2*it);
178         ZEIGER = 1+Width*(1+MOD);
179         p = (int)ZEIGER;
180         frac = ZEIGER-p;
181         putDelayLowR(low);
182         outLr = getDelayLowR(p+2)*frac+getDelayLowR(p+1)*(1-frac);
183         outLr = (1+coefMod*MOD)*outLr;
184
185         MOD = -MOD;
186         ZEIGER = 1+Width*(1+MOD);
187         p = (int)ZEIGER;
188         frac = ZEIGER-p;
189         putDelayLowL(low);
190         outLl = getDelayLowL(p+2)*frac+getDelayLowL(p+1)*(1-frac);
191         outLl = (1+coefMod*MOD)*outLl;
192
193         //  outLr = 0.0; outLl = 0.0;
194
195         // cornetas
196         MOD = -cosseno(Mc*pi2*it);
197         ZEIGER = 1+Width*(1+MOD);
198         p = (int)ZEIGER;
199         frac = ZEIGER-p;
200         putDelayHighR(high);
201         outHr = getDelayHighR(p+2)*frac+getDelayHighR(p+1)*(1-frac);
202         outHr = (1+coefMod*MOD)*outHr;
203
204         MOD = -MOD;
205         ZEIGER = 1+Width*(1+MOD);
206         p = (int)ZEIGER;
207         frac = ZEIGER-p;
208         putDelayHighL(high);
209         outHl = getDelayHighL(p+2)*frac+getDelayHighL(p+1)*(1-frac);
210         outHl = (1+coefMod*MOD)*outHl;
211
212         //  outHr = 0.0; outHl = 0.0;
213
214         it++; //iterator do cosseno
215
216         // saídas
217         outVibR = outHr+outLr; outVibL = outHl+outLl;
218         outRight = outVibR+0.3*outVibL; outLeft = outVibL+0.3*outVibR;
219         //tEnd = micros();
220 }
221 // ***********************************************
```

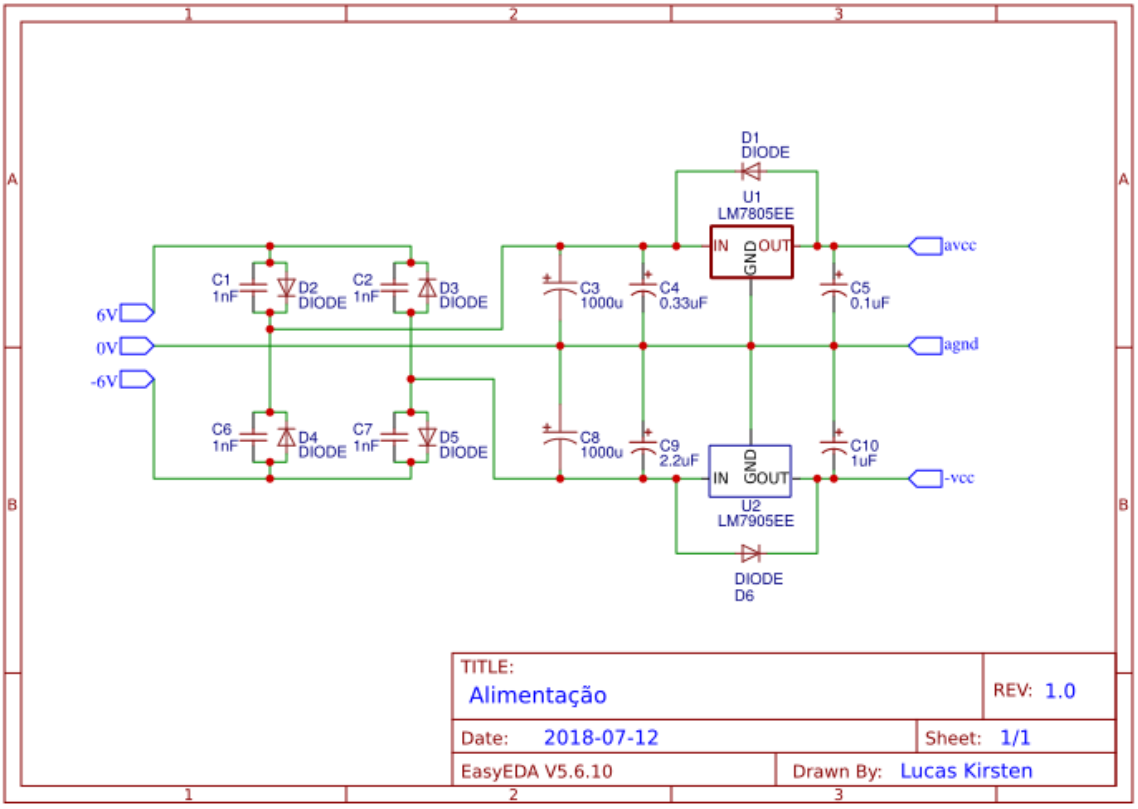
```
222
223 // ***************CONFIGURAçÃO ADC,DAC e TIMER**************
224 void configADC(){
225         // ADC_0
226         pinMode(A9, INPUT); pinMode(A8, INPUT); pinMode(A7, INPUT); pinMode(A6, INPUT);
227         adc->setReference(ADC_REFERENCE::REF_3V3, ADC_0);
228         adc->setSamplingSpeed(ADC_SAMPLING_SPEED::HIGH_SPEED, ADC_0);
229         adc->setConversionSpeed(ADC_CONVERSION_SPEED::HIGH_SPEED, ADC_0);
230         adc->setResolution(10, ADC_0);
231         adc->setAveraging(32, ADC_0);
232         // ADC_1
233         pinMode(A13, INPUT);
234         adc->setReference(ADC_REFERENCE::REF_3V3, ADC_1);
235         adc->setSamplingSpeed(ADC_SAMPLING_SPEED::VERY_HIGH_SPEED, ADC_1);
236         adc->setConversionSpeed(ADC_CONVERSION_SPEED::VERY_HIGH_SPEED, ADC_1);
237         adc->setResolution(12, ADC_1);
238         adc->setAveraging(4, ADC_1);
239         // PINO DIGITAL
240         pinMode(2,INPUT);
241 }
242
243 void configDAC(){
244         pinMode(A21, OUTPUT); pinMode(A22, OUTPUT);
245         analogWriteResolution(12);
246 }
247
248 // inicia o timer para fazer as amostragens
249 void initTimer(){
250         amostrador.begin(getSample, Ts);
251         amostrador.priority(0);
252 }
253 // ********************************************************
254
255 void setup(){
256         //sinaliza inicio das amostragens (pisca o led 3 vezes) e matém aceso
257         pinMode(LED_BUILTIN, OUTPUT);
258         for (int j=0; j<8; j++) {
259                 if (j%2==0) digitalWrite(LED_BUILTIN, LOW);
260                 else digitalWrite(LED_BUILTIN, HIGH);
261                 delay(200);
262         }
263
264         configADC();
265         configDAC();
266         initTimer();
267 }
268
269 void loop(){
270         delay(500);
271         POT = adc->analogRead(A9,ADC_0);
272         Mc=(0.8333+5.8333*POT/adc->getMaxValue(ADC_0))/Fs;
273         POT = adc->analogRead(A8,ADC_0);
274         Mw=(0.6667+5.0*POT/adc->getMaxValue(ADC_0))/Fs;
275         POT = adc->analogRead(A7,ADC_0);
276         Width=(0.1+0.9*POT/adc->getMaxValue(ADC_0))/1000*Fs;
277         POT = adc->analogRead(A6,ADC_0);
278         coefMod=POT/adc->getMaxValue(ADC_0);
279         POT = adc->analogRead(A5,ADC_0);
280         dist=POT/adc->getMaxValue(ADC_0)*9+1;
281         //verifica se está no modo lento
```

```
282             if(!digitalRead(2)){
283                     Mc = 0.8333/Fs;
284                     Mw = 0.6667/Fs;
285             }
286 }
```

APÊNDICE C – CIRCUITOS

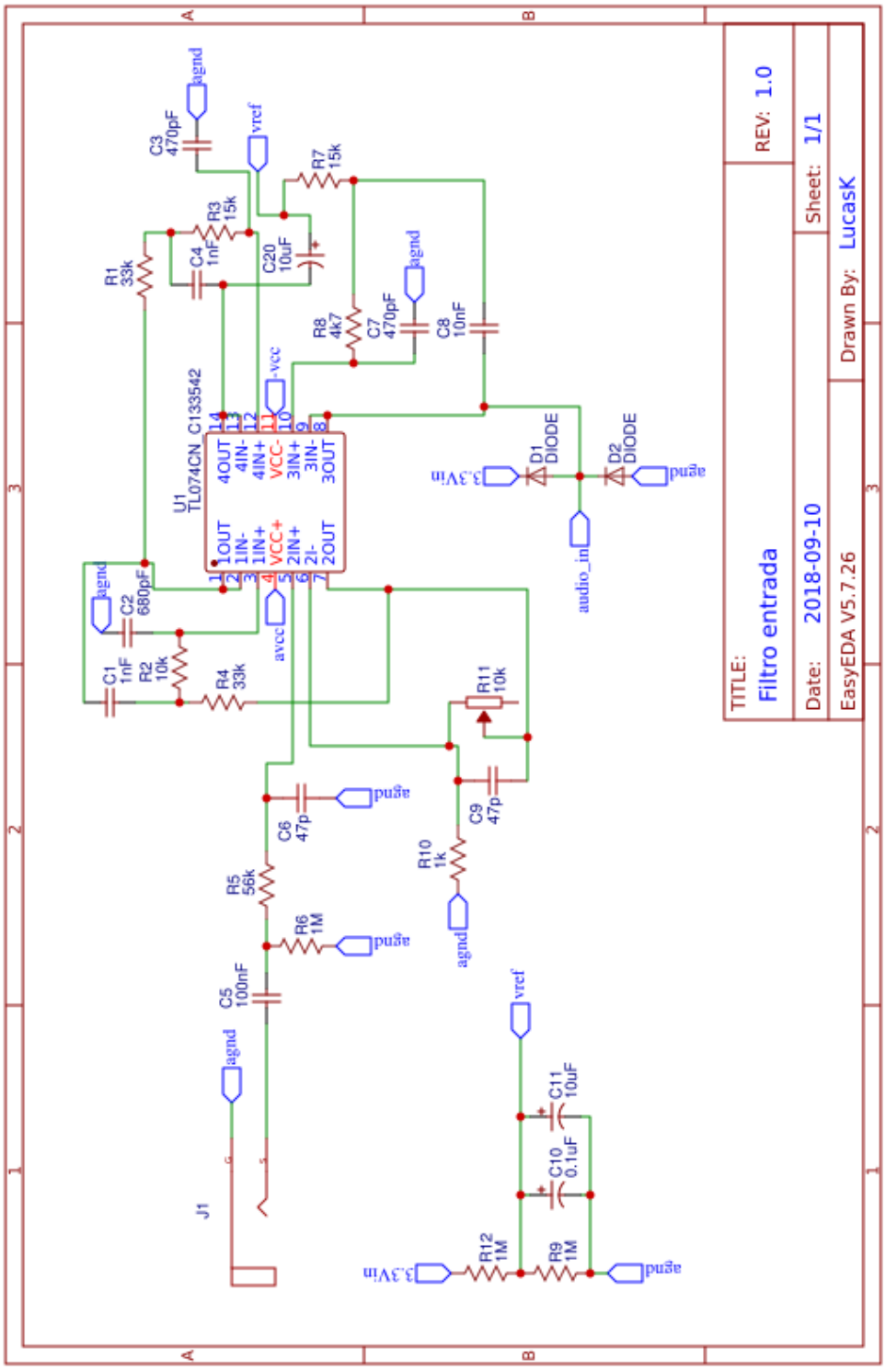
Encontram-se aqui todas as ligações feitas no programa EasyEDA dos componentes para os circuitos projetados, assim como as placas em circuito impresso.

Figura 48 – Ligações para alimentação simétrica.



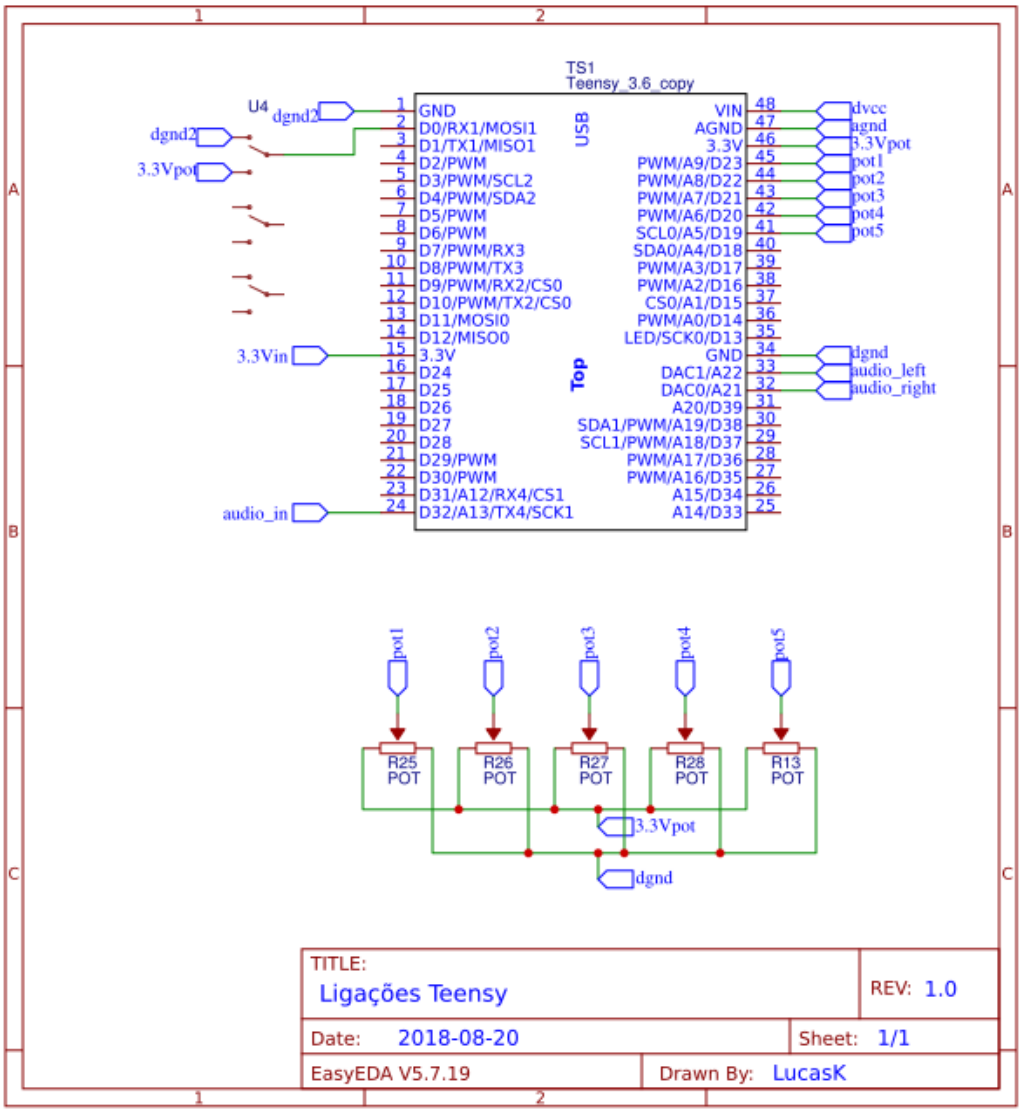
Fonte: Própria,2018.

Figura 49 – Ligações para o filtro de entrada.



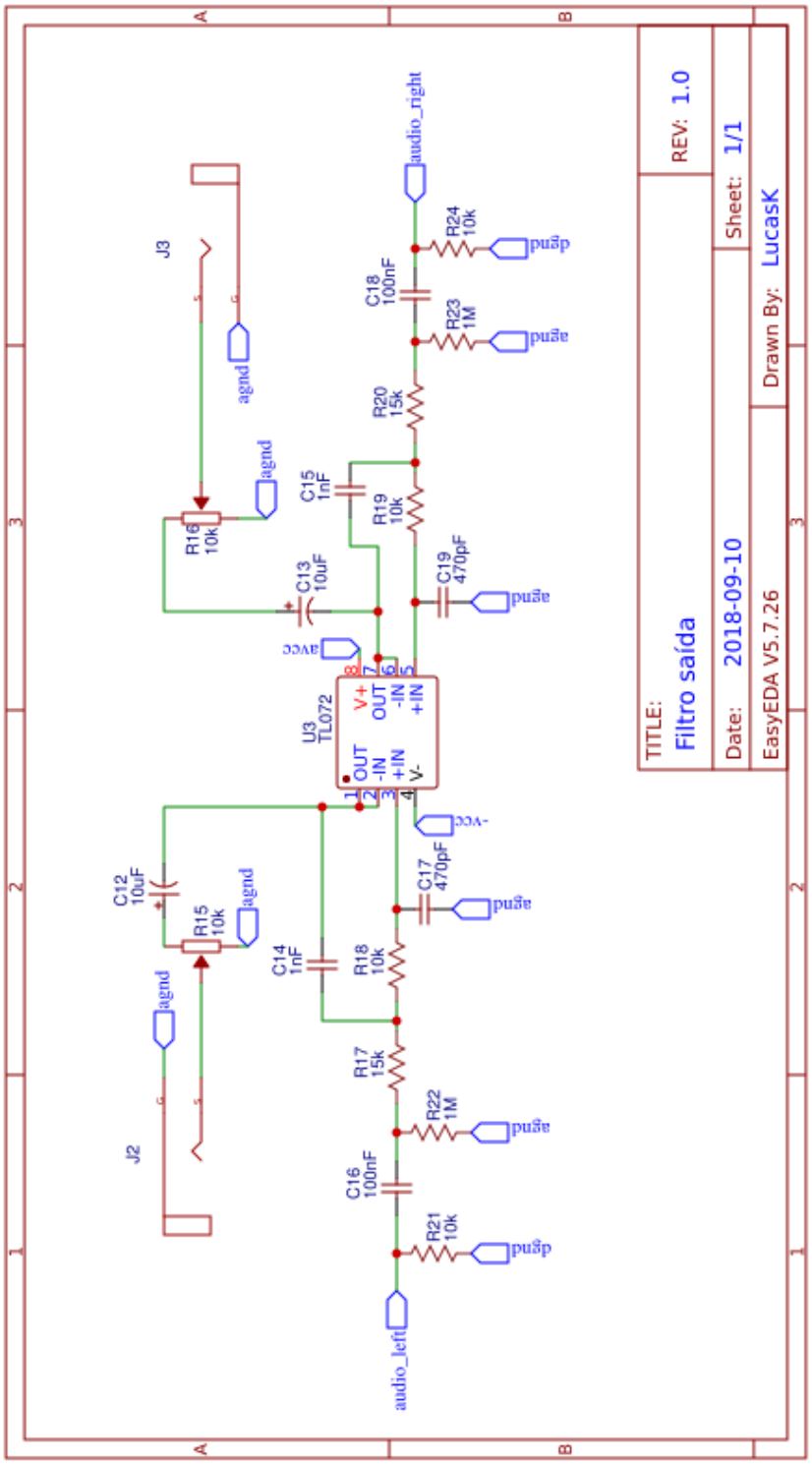
Fonte: Própria,2018.

Figura 50 – Ligações dos pinos do Teensy 3.6.



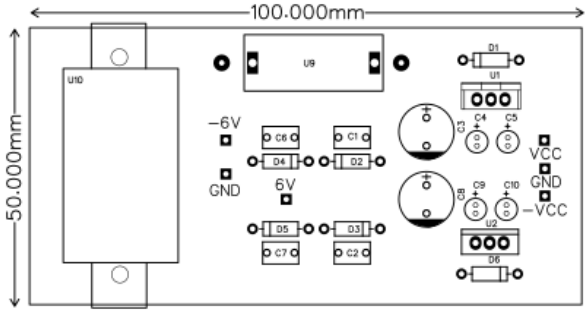
Fonte: Própria,2018.

Figura 51 – Ligações para o filtro de saída.

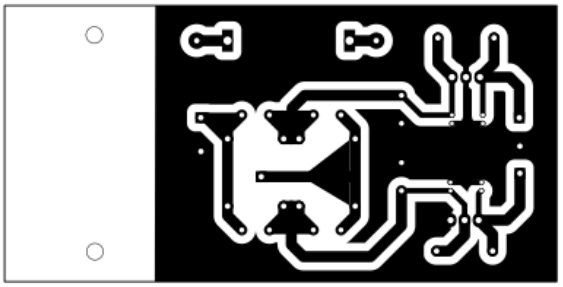


Fonte: Própria,2018.

Figura 52 – PCI do circuito de alimentação.



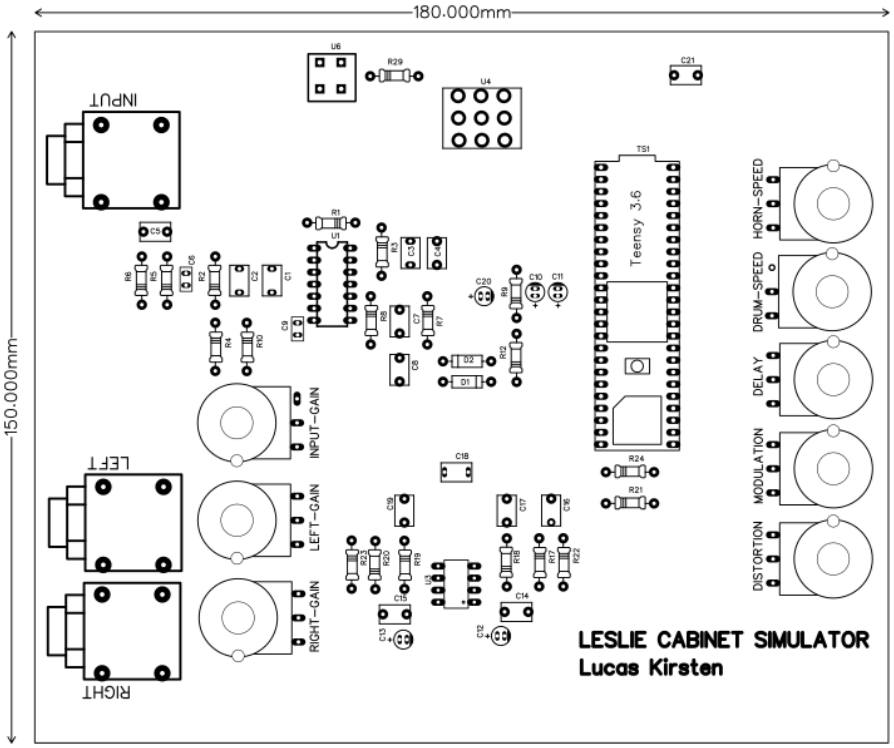
(a) Vista superior: disposição componentes.



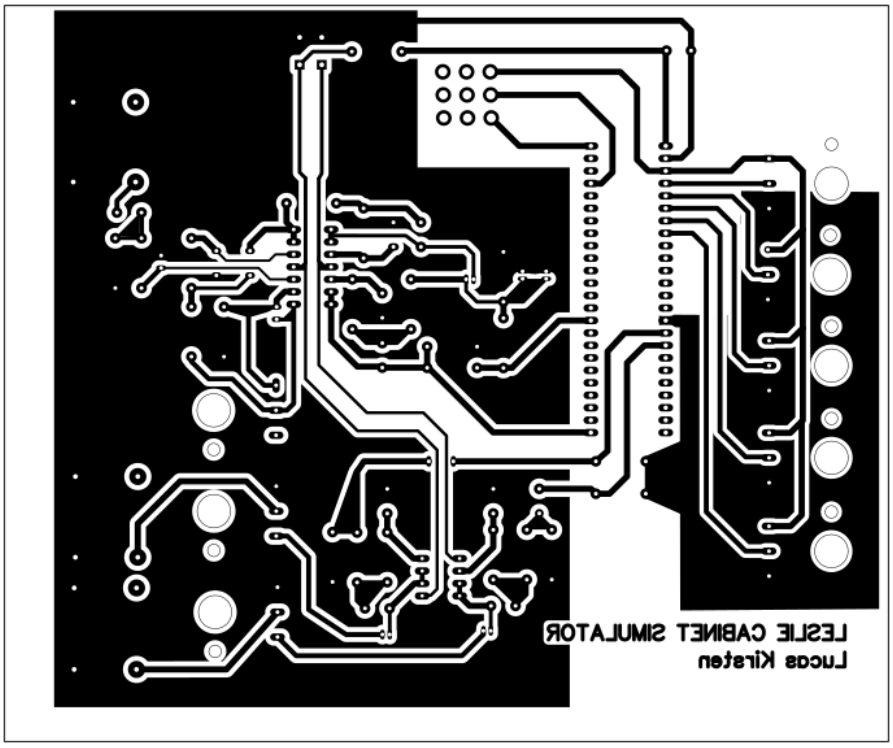
(b) Vista inferior: trilhas no cobre.

Fonte: Própria, 2018.

Figura 53 – PCI do circuito do pedal de simulação.



(a) Vista superior: disposição componentes.



(b) Vista inferior: trilhas no cobre.

Fonte: Própria, 2018.