**ELABORAÇÃO DE UM ARTIGO CIENTÍFICO: modelo**

J. V. Campos[[1]](#footnote-0) e J. S. Oliveira[[2]](#footnote-1)

**MODULAÇÃO DE ONDAS PARA A APLICAÇÃO DE EFEITOS EM**

**INSTRUMENTOS MUSICAIS**  (ESSE FOI O NOSSO TÍTULO DA ÚLTIMA VEZ)

**RESUMO:** Este trabalho apresenta o desenvolvimento de uma pedaleira de efeitos digital, utilizando um microcontrolador de plataforma aberta Franzininho WiFi como seu núcleo. A metodologia empregada integra a concepção de hardware robusto com o desenvolvimento de software otimizado em C++, focado na captura e processamento digital de sinais de áudio. O sistema implementa efeitos sonoros como *delay*, *overdrive*, *phaser* e *tremolo* em tempo real, visando oferecer uma solução versátil e de baixa latência para músicos. Espera-se que o protótipo funcional demonstre a viabilidade do uso de hardware nacional em aplicações de áudio digital, contribuindo para a democratização da tecnologia musical e fomentando o desenvolvimento de soluções abertas.

**Palavras-chave:** Processamento Digital de Sinais, Pedaleira Digital, Franzininho WiFi. Efeitos Sonoros, Hardware Aberto.

**1 INTRODUÇÃO**

A evolução das tecnologias digitais aplicadas à música tem promovido avanços significativos na criação, manipulação e reprodução de efeitos sonoros. Dentro desse cenário, as pedaleiras digitais surgem como uma solução versátil e eficiente para músicos que buscam personalizar sua sonoridade de forma compacta, acessível e interativa. Entretanto, muitos dispositivos comerciais ainda apresentam limitações, como dependência de hardware proprietário e baixa flexibilidade na customização, o que restringe o acesso a essas ferramentas, principalmente para músicos independentes e desenvolvedores de soluções abertas (MARTINS; RODRIGUES, 2020).

Sob a perspectiva da inovação tecnológica, observa-se um crescente interesse na utilização de microcontroladores de alto desempenho aplicados à área de áudio digital. Trabalhos como os de Santos et al. (2021) e Oliveira e Costa (2022) destacam o uso de plataformas embarcadas para processamento de áudio em tempo real, demonstrando que a integração de hardware acessível com algoritmos de efeitos digitais pode atender tanto às demandas sonoras quanto aos requisitos de latência e fidelidade. Entretanto, poucos estudos priorizam o uso de microcontroladores de produção nacional, como a Franzininho WiFi, que oferece recursos robustos baseados no chip ESP32, favorecendo iniciativas de desenvolvimento sustentável e open source no Brasil (SOUSA et al., 2023).

**Além da concepção do hardware, este projeto também propõe um software dedicado, programado em linguagem C++, responsável por capturar os sinais de entrada, aplicar algoritmos digitais de efeitos sonoros — como**  **— e gerar a saída com mínima latência. O sistema permite ajustes paramétricos em tempo real por meio de potenciômetros e** switches **físicos, e ainda possibilita a importação de arquivos de áudio para posterior modificação. Essa flexibilidade amplia o uso da pedaleira tanto em performances ao vivo quanto em ambientes de estúdio, oferecendo uma experiência híbrida e altamente customizável (ALMEIDA; PEREIRA; MELO, 2022).**

Frente a esse cenário, o presente trabalho justifica-se pela necessidade de desenvolver uma pedaleira digital personalizável e baseada em hardware livre, capaz de processar efeitos de áudio em tempo real com alta qualidade sonora e baixa latência. O problema de pesquisa que **fundamenta/ orienta** este estudo consiste na seguinte indagação: "É possível desenvolver uma pedaleira digital eficiente e baseada em hardware nacional, que ofereça processamento de efeitos sonoros em tempo real com qualidade comparável aos equipamentos comerciais?". **A hipótese que se busca testar é que, por meio da integração da Franzininho WiFi, conversores digital-analógico (DAC), potenciômetros, interruptores e um software próprio em C++, é viável atingir uma solução funcional, eficiente e de fácil replicação.**

Diante do que foi exposto, este artigo tem como objetivo principal desenvolver e validar um protótipo de pedaleira digital com processamento de efeitos de áudio em tempo real, utilizando a placa Franzininho Wi-Fi como núcleo do sistema. Pretende-se avaliar sua viabilidade técnica, desempenho sonoro e usabilidade no contexto musical. Como resultados esperados, busca-se obter um sistema com interface intuitiva e qualidade sonora satisfatória, que contribua para a democratização da tecnologia musical, além de fomentar o uso de hardware nacional e fortalecer o desenvolvimento de soluções abertas no campo do áudio digital.

**2 REFERENCIAL TEÓRICO**

**2.1 Modulação de Ondas e Efeitos Sonoros em Instrumentos Musicais**

Os pedais de efeito são ferramentas cruciais no universo musical, permitindo a transformação do timbre de instrumentos através da modulação de ondas sonoras. Essa diversidade possibilita a amplificação da emoção em uma peça musical (Reiss & McPherson, 2014). Tradicionalmente, pedais analógicos modulam as ondas eletricamente, garantindo alta fidelidade, porém, cada efeito requer um circuito dedicado. Essa limitação impõe aos músicos a aquisição de múltiplos pedais, gerando acúmulo de equipamentos e dificultando a versatilidade, especialmente em performances ao vivo (Krueger, 2008).

Como solução, surgem os pedais digitais, que consistem em um conjunto de hardware e software para modulação digital das ondas. O processo envolve a conversão do sinal analógico para digital, a aplicação do efeito e a reconversão para analógico. Embora essas conversões possam introduzir uma pequena perda na definição do áudio, a principal vantagem dos pedais digitais reside na capacidade de alternar ou combinar múltiplos efeitos em um único dispositivo. Isso oferece maior versatilidade e praticidade para os músicos (Marzochi, 2013).

**2.2 Microcontroladores e Hardware da Pedaleira Digital**

A base de uma pedaleira digital moderna reside em microcontroladores capazes de processar sinais em tempo real. O microcontrolador ESP32, que compõe a placa Franzininho WiFi, é um exemplo de hardware de plataforma aberta com recursos ideais para essa aplicação. Embora não seja projetada especificamente para áudio, a Franzininho Wi-Fi possui capacidade de processamento, GPIOs e conectividade Wi-Fi suficientes para executar rotinas de Processamento Digital de Sinais (DSP) com desempenho satisfatório (Matuck, 2005).

Além do microcontrolador, a arquitetura de hardware da pedaleira inclui componentes essenciais. Um Conversor Digital-Analógico (DAC) é fundamental para transformar o sinal digital processado de volta em áudio audível. Potenciômetros e interruptores são empregados para controle intuitivo dos parâmetros e seleção dos efeitos, respectivamente. As entradas e saídas de áudio são isoladas por circuitos analógicos para garantir a fidelidade do sinal. A montagem do sistema, que compreende a prototipagem em protoboard e a confecção de Placa de Circuito Impresso (PCI), reflete a integração de conceitos de eletrônica digital e analógica.

Estudos de projetos de código aberto como o Arduino Guitar Pedal (Arduino Guitar Pedal, s.d.) e o pedalSHIELD (PedalSHIELD, s.d.) forneceram embasamento técnico para a seleção e implementação desses componentes, garantindo a robustez e a qualidade do hardware.

**2.3 Software para Aplicação de Efeitos em Instrumentos Musicais**

A modularidade do código foi garantida pela criação de duas bibliotecas em C++. A *wavManipulation.hpp* é responsável pela manipulação direta de arquivos .wav, com funções para leitura, escrita e aplicação de efeitos. A estrutura de um arquivo .wav, dividida em seções "RIFF", "fmt" e "data", foi fundamental para o design dessa biblioteca. A segunda biblioteca, Effects.hpp, contém classes específicas para cada efeito (Delay, Overdrive, Phaser, Tremolo), herdando de uma classe abstrata AudioEffect. Essa arquitetura promove a adição de novos efeitos e o aprimoramento dos existentes sem interferência no código principal.

**2.3.1 Delay**

O efeito Delay é um efeito temporal que armazena o sinal de entrada em um buffer circular e o reproduz após um determinado intervalo de tempo. O delay pode ser configurado para misturar o sinal original com múltiplas repetições, controlando parâmetros como o tempo de atraso, o feedback (realimentação) e a mistura entre o sinal seco e o sinal processado. O *delayedSample* é lido do buffer na posição *writeIndex*. Em seguida, a nova amostra a ser escrita no buffer é calculada como a soma da sample de entrada com a *delayedSample* multiplicada pelo fator de feedback. O *writeIndex* é incrementado e reiniciado se atingir o final do buffer, garantindo o comportamento circular. Finalmente, o método retorna uma mistura controlada pelo usuário entre o sinal original e o sinal com delay.

**2.3.2 Overdrive**

O efeito Overdrive é um tipo de distorção de áudio que simula a saturação de amplificadores de válvulas. Este efeito adiciona harmônicos e sustain ao sinal, conferindo um timbre mais encorpado e "quente". A implementação do Overdrive geralmente envolve o corte e a compressão do sinal de áudio que excede um determinado limiar, utilizando funções não-lineares, como a função arctan ou o recorte hard-clipping. Essas funções distorcem a forma de onda do áudio de forma controlada, criando o efeito característico de overdrive.

**2.3.3 Phaser**

A manipulação da fase, orquestrada pelo Phaser, é alcançada por meio de uma intrincada cadeia de filtros do tipo “all-pass”, onde a frequência central é dinamicamente controlada por um oscilador de baixa frequência (LFO). Primeiramente, o método calcula o valor do LFO usando uma função seno, que varia entre -1 e 1. A fase do LFO é continuamente atualizada com base na frequência do LFO e na taxa de amostragem (sampleRate). Em seguida, a frequência central dos filtros é modulada pelo LFO, usando o parâmetro "depth". O sinal de entrada é então processado através de uma cascata de filtros all-pass, cada um com seu próprio buffer. O resultado de cada filtro é realimentado para o próximo, e o last armazena a saída do último filtro para o feedback na próxima amostra. Finalmente, a saída do efeito é uma mistura do sinal original e do sinal processado pelos filtros. Como elucidam Reiss e McPherson (2014), essa interação gera deslocamentos de fase que produzem cancelamentos e reforços periódicos no espectro, culminando em um som característico de varredura que evoca qualidades “espaciais” ou “psicodélicas”. A função reset() garante que a fase do LFO e os buffers dos filtros sejam reiniciados.

**2.3.4 Tremolo**

O efeito Tremolo consiste na modulação periódica da amplitude do sinal de entrada. O método *processSample* da classe *TremoloEffect* utiliza um oscilador de baixa frequência (LFO) para variar a intensidade do sinal ao longo do tempo, de acordo com a profundidade e a frequência especificadas. O método calcula o valor do LFO usando uma função seno, que é ajustada para variar entre 0 e 1, garantindo que a amplitude nunca seja completamente zerada (a menos que depth seja 1). A fase do LFO é atualizada de forma similar ao Phaser. O sinal de entrada é então multiplicado por um fator que é modulado pelo LFO e pelo “*depth*”. O tremolo não altera o conteúdo espectral do sinal, mas sim sua dinâmica, criando uma sensação de pulsação rítmica. O resultado é uma variação cíclica do volume, produzindo um efeito pulsante e vibrante, amplamente utilizado em instrumentos de corda e teclados. A função reset() reinicia a fase do LFO.

**2.4 Trabalhos Correlatos**

A área de processamento de áudio digital para instrumentos musicais tem sido objeto de diversos estudos e desenvolvimentos, muitos dos quais se beneficiam da natureza de código aberto de plataformas de microcontroladores, o que permite um avanço contínuo e colaborativo.

O projeto "Arduino Guitar Pedal" (Arduino Guitar Pedal, s.d.), por exemplo, demonstra a viabilidade da utilização da plataforma Arduino para a criação de pedais de efeito. Similarmente, o "PedalSHIELD" (PedalSHIELD, s.d.) explora as capacidades de microcontroladores para processamento de áudio em tempo real, fornecendo uma base de conhecimento prático para o desenvolvimento de soluções customizadas. Essas iniciativas de código aberto são cruciais, pois oferecem modelos e inspirações para novas implementações, como a presente proposta que visa o desenvolvimento de uma pedaleira utilizando o Franzininho Wi-Fi.

Além disso, trabalhos que abordam o tratamento de sinais sonoros no computador, como os estudos de Krueger (2008) sobre simulação de pedais virtuais, fornecem insights valiosos sobre os algoritmos e técnicas de DSP que podem ser adaptados para hardware embarcado. A pesquisa sobre análise e remoção de ruídos em sinal de voz, como a de Rocha (2011), e a remoção adaptativa de ruído em sinais de áudio, por Sarmento et al. (2012), embora focadas em voz, apresentam princípios de manipulação de sinais que são aplicáveis ao contexto de efeitos de guitarra, especialmente na busca por maior clareza e fidelidade sonora.

A viabilidade técnica de utilizar hardware nacional e de uso geral para o processamento de áudio, conforme abordado por Matuck (2005) em seu trabalho sobre redes neurais artificiais aplicadas ao processamento de sinais de voz, reforça a escolha da Franzininho Wi-Fi para este projeto. A pesquisa contínua e o desenvolvimento em Processamento Digital de Sinais de Áudio, conforme observado por Marzochi (2013) com o crescimento do mercado de instrumentos musicais e a busca por produtos de alta qualidade, indicam a relevância e a demanda por soluções inovadoras no campo dos efeitos digitais para instrumentos.

**3 METODOLOGIA**

**3.1 Especificação Arquitetônica**

O arcabouço da pedaleira digital, conforme proposto, busca oferecer uma plataforma de efeitos sonoros em tempo real, permitindo aos músicos personalizar o timbre de seus instrumentos. A arquitetura da aplicação é composta por componentes principais interconectados de forma modular, permitindo adaptabilidade e escalabilidade do sistema. O projeto consiste na implementação de um hardware integrado a um microcontrolador, que, por meio de programação, permite a implementação de efeitos para guitarra.

1. Entrada do Instrumento: Este componente é responsável pela captação do sinal de áudio analógico proveniente do instrumento musical. Para garantir a fidelidade do sinal, serão utilizadas entradas de áudio isoladas por meio de circuitos analógicos apropriados.
2. Processamento Digital de Sinais (DSP): Utilizando a placa Franzininho Wi-Fi, baseada no ESP32, este módulo processa o sinal de áudio capturado. Essa placa nacional, apesar de não ser específica para áudio, possui recursos suficientes (como GPIOs, conectividade Wi-Fi e capacidade de processamento) para executar rotinas de DSP com desempenho satisfatório. Algoritmos de DSP serão aplicados para simular efeitos como distorção, delay, overdrive, phaser e tremolo. A primeira etapa do desenvolvimento foi dedicada ao estudo e aplicação dos conceitos relacionados a Sinais e Processamento Digital de Sinais (Bianchi, 2012; Herrera, 2004).
3. Controle e Interface do Usuário: Este componente permite a interação do usuário com a pedaleira. Potenciômetros serão utilizados para a regulagem de parâmetros dos efeitos, enquanto interruptores de acionamento permitirão a troca e ativação dos efeitos em tempo real.
4. Conversão Digital-Analógica (DAC): Após o processamento do sinal digital, este módulo é responsável por converter o sinal modificado de volta para a forma analógica.
5. Saída do Áudio Processado: Este componente direciona o sinal de áudio analógico, agora com os efeitos aplicados, para o amplificador ou sistema de som.

**3.2 Diferenciais do Sistema**

O diferencial do sistema reside na sua capacidade de oferecer uma pedaleira digital funcional e personalizável, utilizando hardware nacional e plataformas abertas. Pedais digitais, ao contrário dos analógicos, convertem o sinal do instrumento em um sinal digital para aplicar o efeito e, após isso, o sinal é novamente convertido para analógico para reprodução. Embora essa conversão possa dificultar a obtenção da mesma qualidade dos pedais analógicos, pedaleiras digitais atuais se aproximam significativamente desse nível de qualidade.

O Processamento Digital de Sinais de Áudio tem sido impulsionado pelo ramo comercial, com empresas investindo em produtos de alta qualidade, como a BOSS GT-100 Amp Effects Processor e a POD HD500X. O projeto busca desenvolver um pedal de efeitos digital utilizando um hardware de plataforma aberta capaz de processar e modificar o som.

**3.3 Implementação Técnica**

A pedaleira digital será desenvolvida com base em uma arquitetura modular. O hardware será implementado com a placa Franzininho Wi-Fi, baseada no ESP32, como unidade central de processamento. Para o desenvolvimento do hardware, foram estudados projetos open-source já existentes, como o Arduino Guitar Pedal e o pedalSHIELD. A montagem será realizada inicialmente em protoboard para testes do circuito e da conversão Analógico-Digital. Em seguida, será feita a confecção da placa de circuito impresso (PCI), através da impressão do circuito e corrosão da mesma, perfuração para acomodar os componentes, soldagem e integração do hardware.

O software foi desenvolvido em linguagem C++ por meio da IDE Arduino, adaptada à Franzininho. O código será responsável por:

Capturar o sinal de áudio via entrada analógica.

Aplicar algoritmos de DSP, simulando efeitos como distorção, delay, overdrive, phaser e tremolo.

Realizar a saída do sinal modificado por meio do DAC.

Interpretar os comandos físicos dos potenciometros e switches para alterar os parâmetros dos efeitos em tempo real.

Além disso, serão desenvolvidas rotinas específicas de otimização para garantir uma comunicação eficiente entre o microcontrolador e os componentes periféricos, assegurando baixa latência e estabilidade durante a operação ao vivo. As primitivas básicas para a implementação dos efeitos no Arduino incluem a declaração inicial das variáveis, a configuração dos conversores Analógico-Digital e Digital-Analógico, a leitura e conversão dos dados, e a aplicação do Processamento Digital de Sinais (DSP).

**3.4 Modelagem do Arcabouço**

A modelagem do arcabouço da pedaleira digital é essencial para transformar as ideias em uma estrutura visual e funcional. Através da modelagem, é possível entender o funcionamento do sistema, seus componentes e a interação dos usuários. Serão definidos requisitos funcionais (ações que o sistema deve executar, como a aplicação de efeitos e a resposta aos controles) e não funcionais (qualidade sonora, desempenho em tempo real e estabilidade). Essa etapa de modelagem é crucial para transformar a ideia em uma solução prática e eficiente. A etapa de desenvolvimento dos efeitos envolveu um amplo estudo de Processamento Digital de Sinais e outras áreas relacionadas à manipulação e utilização de dados, empregando a linguagem de programação específica para o Arduino.

**4 DESENVOLVIMENTO**

A fase inicial do desenvolvimento do software concentrou-se na seleção dos efeitos a serem implementados, visando maximizar a variedade e a versatilidade da pedaleira. Com base nos conceitos de Processamento Digital de Sinais (DSP) abordados por Reiss e McPherson (2014), foram escolhidos os efeitos de delay, overdrive, phaser e tremolo.

Para a programação, a linguagem C++ foi a opção preferencial devido à sua otimização e capacidade de manipulação eficiente de dados, características cruciais para o processamento de áudio em tempo real. A compatibilidade nativa do C++ com a placa Franzininho Wi-Fi simplificou a integração entre hardware e software. O desenvolvimento inicial envolveu a criação de um programa para aplicar efeitos em áudio em tempo de execução, posteriormente adaptado para operação em tempo real com a inclusão de um buffer. A modulação das ondas ocorre através da iteração sobre cada ponto da onda, aplicando o efeito individualmente a cada valor.

Para simplificar o código final e promover a modularidade, foram criadas duas bibliotecas em C++, garantindo uma clara separação de responsabilidades. A primeira, *wavManipulation.hpp*, foi destinada à manipulação direta de arquivos wave.

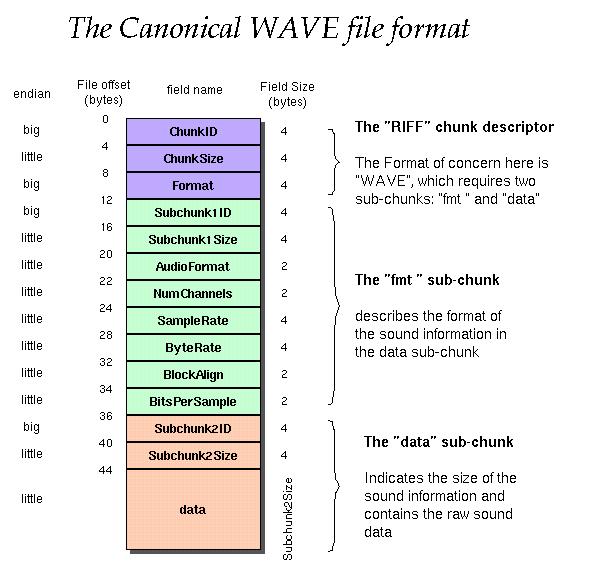


Figura 1 - Formato canônico do arquivo WAVE

A estrutura de um arquivo .wav é fundamental para entender o design da biblioteca wavManipulation.hpp. Ela é segmentada em três partes principais: "RIFF", "fmt" e "data".

A seção "*RIFF*" inicia o arquivo com 12 bytes e é subdividida em "*Chunk ID*" (identifica a seção), "*Chunk Size*" (indica o tamanho total do arquivo) e "*Format*" (confirma o formato "WAVE").

A segunda divisão, "fmt", é mais extensa e complexa, detalhando a formatação do som codificado. Com 24 bytes, ela é dividida em oito partes: "*Subchunk1ID*", "*Subchunk1Size*", "*AudioFormat*" (compressão utilizada), "*NumChannels*" (quantidade de canais), "*SampleRate*" (amostras por segundo), "*ByteRate*" (bytes por segundo), "*BlockAlign*" (tamanho em bytes de uma amostra) e "*BitsPerSample*" (bits por amostra).

A terceira e última parte, "data", é dividida em "*SubChunk2ID*", "*SubChunk2Size*" e a seção final "*data*", que armazena os dados de áudio. A biblioteca *wavManipulation.hpp* opera diretamente sobre o arquivo, utilizando uma *struct* WaveHeader para identificar o cabeçalho do arquivo wave e funções especializadas para leitura, escrita e aplicação dos efeitos de áudio.

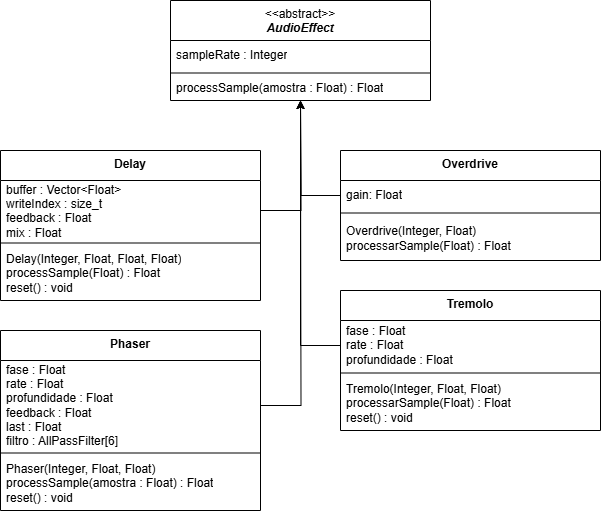
****

Figura 2 - Diagrama UML das classes

A segunda biblioteca implementada foi a Effects.hpp, que contém quatro classes, uma para cada efeito codificado, e uma classe abstrata denominada *AudioEffect*. Esta classe abstrata serve como base para as demais, estabelecendo uma interface comum para a aplicação de efeitos sonoros. Essa estrutura promove a adição de novos efeitos e o aprimoramento dos existentes sem interferência.

As classes concretas implementam as funcionalidades específicas para cada tipo de efeito, herdando as características e o comportamento definidos em *AudioEffect* e especializando-os para atender aos requisitos de processamento de áudio desejados. Isso permite que o projeto seja modular, facilitando a integração e a manutenção de diferentes algoritmos de efeito dentro de um único dispositivo digital. A aplicação do projeto se manifesta na capacidade de alternar ou combinar múltiplos efeitos em um único pedal digital, superando a limitação dos pedais analógicos que exigem um circuito dedicado para cada efeito.

**5. Síntese e Atividade prevista**

O presente trabalho dedicou-se ao desenvolvimento de uma pedaleira de efeitos digital, empregando um microcontrolador de plataforma aberta, a Franzininho Wi-Fi, como núcleo do sistema. A proposta integrou a concepção de um hardware robusto com o desenvolvimento de um software otimizado em C++, visando oferecer aos músicos uma solução versátil, eficiente e de baixo custo para personalização sonora. A estrutura do sistema foi projetada para capturar sinais de áudio analógicos, processá-los digitalmente aplicando efeitos como delay, overdrive, phaser e tremolo, e retornar o sinal modificado com mínima latência, conforme detalhado no projeto. A utilização de hardware nacional e a abordagem de código aberto buscam fomentar a democratização da tecnologia musical e a replicabilidade do projeto.

Como proposta de continuidade, prevê-se a expansão das funcionalidades do software, incluindo a implementação de uma interface de usuário mais intuitiva para seleção e ajuste de efeitos via tela LCD ou conexão sem fio (Wi-Fi), explorando a capacidade da Franzininho Wi-Fi. Além disso, busca-se aprimorar a qualidade sonora dos efeitos através de otimizações nos algoritmos DSP e a exploração de conversores digital-analógico (DAC) de maior resolução, visando minimizar a perda de definição de áudio e equiparar a qualidade do produto final a pedaleiras comerciais de alto desempenho. Futuras atividades também contemplam a experimentação com a adição de novos efeitos sonoros e a possibilidade de gravação e reprodução de *loops* de áudio diretamente no dispositivo.

REFERENCIAS

Arduino Guitar Pedal. Disponível em: http://www.instructables.com/id/Arduino-Guitar-Pedal/. Acesso em: 4 jul. 2016.

BIANCHI, A. J. Processamento de áudio em tempo real em plataformas computacionais de alta disponibilidade e baixo custo. 2012. Disponível em: https://www.ime.usp.br/~ajb/projeto/mestrado-ajb.pdf. Acesso em: 14 jun. 2025.

HERRERA, C. G. Projeto de sistemas de processamento digital de sinais de Áudio utilizando metodologia científica. 2004.

KRUEGER, D. Tratamento de sinais sonoros no computador simulando pedais virtuais. 2008.

MARZOCHI, R. Cresce a venda de instrumentos musicais. 2013.

MATUCK, G. R. Processamento de sinais de voz padrões comportamentais por redes neurais artificiais. INPE, 2005. (Tech. Rep.). Disponível em: http://mtc-m21c.sid.inpe.br/col/sid.inpe.br/mtc-m21c/2020/08.03.18.54/doc/processamento%20sinais.pdf. Acesso em: 14 jun. 2025.

OLIVEIRA, L. G. A.; DUARTE, L. T. Um estudo sobre separação cega de sinais de Áudio. [S. l.: s. n.], [s.d.]. Disponível em: https://www.prp.unicamp.br/pibic/congressos/xxicongresso/paineis/103024.pdf. Acesso em: 14 jun. 2025.

PedalSHIELD. Disponível em: http://www.electrosmash.com/pedalshield. Acesso em: 4 jul. 2016.

REISS, J. D.; MCPHERSON, A. P. Audio Effects: Theory, Implementation and Application. Boca Raton: CRC Press, 2014. Disponível em: https://www.taylorfrancis.com/books/mono/10.1201/b17593/audio-effects-joshua-reiss-andrew-mcpherson. Acesso em: 14 jun. 2025.

ROCHA, L. G. Análise e remoção de ruídos em sinal de voz através de técnica de atenuação de espectro. 2011. Disponível em: https://www2.ifmg.edu.br/sic/edicoes-anteriores/resumos-2016/desenvolvimento-de-uma.pdf. Acesso em: 14 jun. 2025.

SARMENTO, C. F.; REGIS, D.; ARAÚJO, W. Remoção adaptativa de ruído em sinais de Áudio. 2012. Disponível em: https://propi.ifto.edu.br/ocs/index.php/connepi/vii/paper/viewFile/2986/2540. Acesso em: 14 jun. 2025.

Observação: Não foi possível encontrar links diretos para o conteúdo completo de algumas das referências citadas (Herrera, Krueger, Marzochi, Rocha, Sarmento et al.) nos resultados da busca.

1. Graduando em Ciências da computação. E-mail: joao.madeiro@iesb.edu.br [↑](#footnote-ref-0)
2. Graduando em Engenharia da computação. E-mail: julia.s.oliveira@iesb.edu.br [↑](#footnote-ref-1)