Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

IoMusT: Efeitos Sonoros Interativos com ESP32

André Guilherme Aquino de Alencar

TRABALHO DE GRADUAÇÃO

ENGENHARIA ELÉTRICA

Brasília

2024

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

IoMusT: Efeitos Sonoros Interativos com ESP32

André Guilherme Aquino de Alencar

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro Eletricista

Orientador: Prof. Dr. Adson Ferreira Da Rocha

Brasília
2024

Aquino de Alencar, André Guilherme.
A672i IoMusT: Efeitos Sonoros Interativos com ESP32 / André Guilherme Aquino de Alencar; orientador Adson Ferreira Da Rocha. -- Brasília, 2024.
41 p.

Trabalho de Graduação (Engenharia Elétrica) -- Universidade de Brasília, 2024.

1. Internet das Coisas Musicais. 2. Engenharia de Áudio. 3. Pedais Inteligentes. 4. Interação Software-Pedais Analógicos. I. Ferreira Da Rocha, Adson, orient. II. Título

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

IoMusT: Efeitos Sonoros Interativos com ESP32

André Guilherme Aquino de Alencar

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro Eletricista

Trabalho aprovado. Brasília, 20 de Setembro de 2024:

Prof. Dr. Adson Ferreira da Rocha,
UnB/FT/ENE
Orientador

Prof. Dr. João Luiz Azevedo de Carvalho,
UnB/FT/ENE
Examinador Interno

Prof. Dr. Paulo Roberto de Lira Gondim,
UnB/FT/ENE
Examinador Interno

Brasília
2024

Este trabalho é dedicado àqueles cujas ideias são tão imensas que transbordam das próprias mentes.

Agradecimentos

Aos meus professores que me inspiraram e me aconselharam bastante durante toda a minha joranada acadêmica. Ao maior parceiro e incentivador da minha vida, Guilherme Veras, meu pai e melhor amigo que me acolhe nos meus melhores e piores momentos. Minha avó Maria da Glória, que sempre me apoiou em todos os meus projetos e que é minha grande mentora. Aos meus demais familiares, em especial: meu avô Reginaldo Alencar, meus tios paternos Reginaldo e Gisele Veras, minha mãe Henia Aquino, minha irmã Giovanna Schneider, meus avós queridos Raimundo e Maria Aquino, meus tios maternos Hellen Aquino e Julião Aquino e meu padrasto Diego. A todos os meus verdadeiros amigos por sempre me lembrarem que não estou sozinho e, por último mas não menos importante, meus companheiros do ramo musical, que me encorajam incessantemente a acompanhá-los em seus projetos.

“There’s faith and there’s sleep.
We need to pick one, please.
Because faith is to be awake
and to be awake is for us to think
and for us to think is to be alive.”
(Tyler Joseph)

Resumo

A música nada mais é do que um conjunto ordenado de ondas sonoras que se propagam em meios materiais de forma a provocar sensações diversas no cérebro. Desde a criação do primeiro instrumento musical, a música vem evoluindo até se tornar o que é hoje. Nos tempos atuais existe uma infinidade de tecnologias responsáveis por reproduzir os mais diversos sons possíveis e, dentre esses instrumentos, está a guitarra elétrica. Um dos fatores que diferencia em grande parte a guitarra elétrica dos demais instrumentos musicais é a sua versatilidade sonora, que se dá por meio da criação dos conhecidos pedais de efeitos sonoros que, hoje, são produzidos e comercializados em grande escala. Acontece que, de fato, os pedais ditos analógicos nada mais são do que circuitos eletrônicos que modificam o som originário da guitarra. Esse som, ao ser produzido na guitarra, é transformado posteriormente em ondas elétricas pelo processo de transdução, sofrendo as modificações para soar da forma desejada pelo músico. De fato, os pedais analógicos possuem o diferencial do dito "som orgânico", ou seja, diferente dos pedais digitais, que são fabricados com um software responsável pela modificação do som, os analógicos produzem um som que não passa necessariamente por um software. Entretanto, pedais analógicos geralmente possuem a desvantagem de precisarem ser ativados individualmente quando estão conectados em série com outros pedais de forma a produzir o timbre completo desejado pelo guitarrista. A partir desse problema, é possível utilizar os conhecimentos de Engenharia Elétrica de forma a melhorar a usabilidade desses pedais para favorecer o músico. Por meio da tecnologia da Internet das Coisas Musicais, é possível utilizar um servidor de rede, como o do sistema embarcado ESP32, para ativar e desativar esses pedais de forma mais fácil e dinâmica para o músico que está fazendo seu show ao vivo e de forma completamente automatizada. Essa ideia traz uma gama de possibilidades para que o músico não precise se empenhar em trocar os efeitos pisando fisicamente em cada um dos pedais, mas sim já estabelecendo quais pedais que ele gostaria de utilizar e já pré definindo suas combinações de efeitos favorita no próprio celular com apenas um simples toque. O que é mais comum para pedais digitais, agora pode ser implementado por meio da internet das coisas para os pedais analógicos de forma a produzir a melhor sonoridade possível que os músicos tanto desejam.

Palavras-chave: Internet das Coisas Musicais. Engenharia de Áudio. Pedais Inteligentes. Interação Software-Pedais Analógicos.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BPM	*Beats Per Minute*
ESP32	*Wi-Fi and Bluetooth Microcontroller*
HTTP	*HyperText Transfer Protocol*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*Internet Engineering Task Force*
IoMusT	*Internet of Musical Things*
IoT	*Internet of Things*
IP	*Internet Protocol*
PT2399	*Integrated Circuit for Echo Effects*
RFC	*Request For Comments*
RFID	*Radio Frequency Identification*
TRS	*Tip-Ring-Sleeve*
TS	*Tip-Sleeve*
VCO	*Voltage Controlled Oscillator*
X9C104S	*Digital Potentiometer*

Sumário

1 Introdução

1.1 Contextualização

Desde a origem da primeira guitarra elétrica, diversos efeitos de som surgiram para realizar o trabalho de estilizar as músicas. Desde 1962, por exemplo, com o lançamento de "(I can't get no) Satisfaction", o pedal de *fuzz* tem se popularizado demasiadamente entre os músicos. De acordo com o *site Whiplash*, que é o veículo de comunicação sobre *rock* mais importante do Brasil [11], sabe-se que o efeito de *fuzz* foi descoberto de forma involuntária, a partir de uma falha técnica, o som acabou agradando os ouvidos dos que estavam ali presentes e, assim, surgiu a ideia da criação de diversos efeitos para distorcer o som da guitarra elétrica, marcando de vez o instrumento na história da música como essencial.

Dado esse universo de possibilidades e, a partir do artigo científico feito por dois estudantes de Física da Universidade Federal de Uberlândia (UFU) [13], tem-se que o som da guitarra nada mais é do que uma soma ponderada de senoides com diferentes frequências e fases que, ao passar pela transdução, se transformam em um sinal elétrico. Assim, é possível criar circuitos eletrônicos para realizar as tais modificações e lapidações sonoras para a guitarra, e até mesmo para diversos instrumentos musicais em geral. Portanto, a partir dos conhecimentos adquiridos com a eletrônica moderna, é possível criar uma imensa gama de efeitos. Assim, enxerga-se o áudio gerado pela nota de uma guitarra como um sinal e o circuito de efeito como um sistema. Dessa forma, cada som desejado durante a música estará soando graças a uma configuração específica de componentes eletrônicos que é estabelecida previamente ao momento da musica em que ele deve soar.

A partir dessas informações, é possível ainda que esses componentes eletrônicos possam interagir com sistemas embarcados e com a gama de possibilidades que eles oferecem. Em suma, dessa forma estariamos lidando com a automação do som. Com isso, há ainda a possibilidade de controle desses sistemas embarcados via rede graças a microcontroladores como o ESP32. E isso, atualmente, é uma área conhecida como *Internet Of Musical Things (IoMusT)*, a Internet das Coisas Musicais. Um tema que já é explorado há alguns anos e já possui um artigo[1] publicado em um periódico do IEEE em 2018.

1.2 Descrição do Problema e Motivação

Uma forma de obtenção de renda para os musicistas brasileiros que estão iniciando sua jornada na cena musical é com a realização de apresentações de números musicais a partir da venda de ingressos. Isso é um fato, pois a partir do momento em que uma banda lança o seu projeto comercialmente, ela dificilmente obterá renda antes de ser conhecida pelo público, pois a banda precisa ser divulgada para ser conhecida. Dado esse problema, uma alternativa que faz com que o público possa se interessar pela banda é através de shows, que logicamente são uma forma de divulgar a arte do grupo musical visando fazer o público pesquisar as músicas nas plataformas de *streaming*. Juntamente a isso, claro, é possível aproveitar os eventos musicais para realizar a venda dos produtos das bandas. O chamado *Merchandise*.

Por conta disso, é de suma importância que um músico preze por realizar a melhor performance possível, de forma a atrair a atenção do maior número de pessoas e conquistar seu público. Isso já não é uma tarefa fácil para a maioria dos guitarristas, que precisam constantemente regular seus efeitos manualmente em seus pedais. Por mais que as pedaleiras ofereçam uma solução de armazenamento de *pre-sets*, a maioria esmagadora oferece apenas soluções em *software*. O que se torna um incômodo para muitos músicos que têm como objetivo preservar o som analógico através do uso de circuitos eletrônicos.

As possibilidades oferecidas para músicos portadores de pedais analógicos são muito escassas. O que força uma nova geração a aderir soluções digitais, uma vez que os pedais analógicos convencionais que estão acessíveis a musicistas novatos no mercado não oferecem soluções que oferecem *pre-sets*. Isto força os musicistas a se agacharem e configurarem os seus pedais no chão durante o número musical constantemente, podendo afetar a performance musical.

1.3 Objetivos

1.3.1 Objetivo Geral

Este trabalho tem como objetivo demonstrar a aplicabilidade da tecnologia de Internet das Coisas em circuitos eletrônicos para o auxílio de musicistas durante a execução de suas perfomances durante números musicais. Será demonstrada aqui a possibilidade e a eficiência de se controlar efeitos para guitarras através da construção de um circuito de controle de um gerador de ecos, que também é conhecido como *delay*, a partir de uma placa de reverberação *PT2399* e da conexão de um ESP32 interagindo com potenciômetros digitais do tipo *X9C104S* aplicados a essa placa.

1.3.2 Objetivos Específicos

Os objetivos específicos deste trabalho emergem da aplicabilidade de circuitos eletrônicos com ênfase na Internet das Coisas voltados para facilitar a interação dos músicos com seus equipamentos eletrônicos durante apresentações. São estes:

- Demonstrar como a implementação de potenciômetros digitais pode ser efetiva ao se manipular componentes eletrônicos com a finalidade de controlar sons;
- Discutir a aplicabilidade e a rentabilidade de se gerar efeitos com interessantes sonoridades de forma barata e acessível com o uso do conhecimento em eletrônica digital e internet das coisas;
- Argumentar acerca de maiores aplicações do mundo da internet das coisas musicais voltadas para diferentes efeitos para guitarras elétricas e até de automação baseada no chamado BPM de uma música afim de facilitar a performance dos músicos em seus shows.

2 Fundamentação Teórica

Primeiramente deve-se realizar a introdução teórica aos conceitos utilizados para um entendimento completo deste trabalho e uma descrição acerca do funcionamento dos componentes que foram utilizados para a sua confecção.

2.1 Internet das Coisas Musicais (IoMusT)

De acordo com o artigo sobre o projeto Sunflower [3], Internet das Coisas (IoT) por si só é um termo originário de 1999, utilizado por Kevin Ashton, pioneiro da tecnologia britânica cofundadora do Auto-ID center, no Massachussetts Institute of Technology. *IoT* é um termo que Kevin utilizou para referenciar uma tecnologia de *RFID* em cadeias de suprimento. O que ele não imaginava seria o grande potencial do termo que havia criado. Hoje, Internet das Coisas é um termo muito mais abrangente que descreve o funcionamento de equipamentos cotidianos de forma automatizada ou com o uso da internet para que esteja funcionando.

Quando o conceito de *IoT* é aplicado no mundo da música, o novo conceito passa a ser o de *Internet of Musical Things (IoMusT)*. De acordo com um artigo do *site Free 6G Training* [12], uma Coisa Musical é definida como um dispositivo computacional capaz de detectar, adquirir, processar ou atuar, e trocar dados com uma finalidade musical.

Já o conceito de Internet das Coisas Musicais, de acordo com a mesma referência [12], é o conjunto de interfaces, protocolos e representações de informações relacionadas à música que possibilitam serviços e aplicações com um propósito musical, baseados em interações entre humanos e Coisas Musicais, ou entre as próprias Coisas Musicais, em domínios físicos e/ou digitais.

O conceito de Internet das Coisas Musicais abre uma gama de aplicabilidades para o universo musical como um todo. Como, por exemplo, o controle de volume e de timbres de instrumentos musicais a partir de uma máquina central atuando como servidor, como visto na Figura 1. Em suma, equipamentos podem ser modificados, monitorados e até sincronizados via rede graças à existência do estudo do campo de *IoMusT*.

Figura 1 – Diagrama de exemplo de uma rede IoMusT. Fonte: ask.audio

Quando se fala de *IoMusT*, também refere-se a qualquer interação entre o mundo da Internet das Coisas e a sua aplicabilidade no mundo musical. Inclusive, a forma com que essas tecnologias podem ser utilizadas para facilitar a performance do músico que está executando sua apresentação em tempo real.

Existe uma variedade de protocolos que podem ser utilizados especificamente para internet das coisas como, por exemplo, o protocolo *ZigBee*. Esse protocolo é usado amplamente para comunicação entre dispositivos inteligentes em diversas situações devido a sua alta velocidade e alcance considerável, por exemplo em ambientes industriais. Porém, neste trabalho haverá um enfoque maior no protocolo HTTP e de uma rede sem fio proveniente de um ESP32.

2.1.1 Microcontrolador ESP32

O ESP32 é um microcontrolador, ou sistema embarcado, construido pela empresa Espress If, uma empresa chinesa criada em 2008, sediada em Xangai e especializada em Internet das Coisas. Como qualquer sistema embarcado, o ESP32 possui uma grande versatilidade para automatização de tecnologias do dia a dia das pessoas. Porém, a grande vantagem de se utilizar o ESP32 é a facilidade de realizar projetos que exigem conectividade com uma rede, pois esse microcontrolador vem acompanhado de um módulo Wi-fi e possui compatibilidade com várias tecnologias de rede como o bluetooth e o zigbee. Entretanto, o foco neste trabalho é o próprio wi-fi, isto é, por meio do protocolo IEEE 802.11.

O ESP32 Devkit v1 é capaz de ler códigos do próprio programa "Arduino IDE", a partir da linguagem *Sketch* do próprio programa. Além disso, ele possui uma IDE própria que não foi utilizada para a confecção desse trabalho, que é da própria empresa Espress If. A pinagem do ESP32 aparece de forma objetiva no esquema da Figura 2.

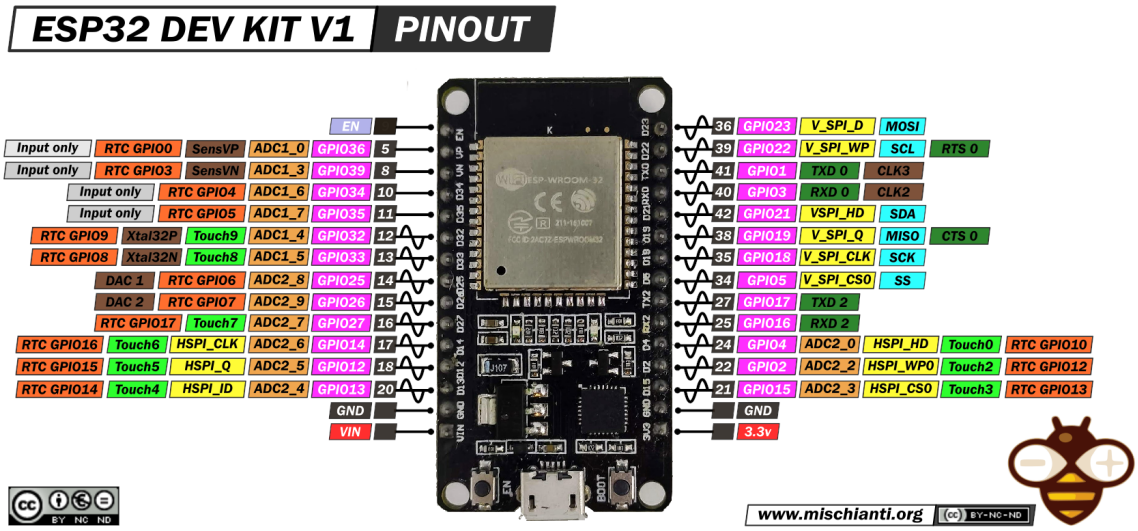

Figura 2 – Pinagem do ESP32 Devkit v1. Fonte:
https://mischianti.org/doit-esp32-dev-kit-v1-high-resolution-pinout-and-specs/

Dessa forma, é possível produzir um código capaz de criar um servidor de rede dentro do ESP32 a partir de duas principais bibliotecas. Elas estão disponíveis para serem utilizadas na linguagem .ino, que é a principal linguagem do Arduino compreendida também pelo ESP32. De acordo com a documentação das bibliotecas do Arduino [5], São elas:

- **WiFi.h:** Biblioteca responsável por permitir conexão local (ou com a Internet) usando o módulo Wi-fi do ESP32. Possibilitando instanciar servidores, clientes e a troca de pacotes na internet através da Wi-Fi.
- **WebServer.h:** Essa biblioteca é usada para criar propriamente o servidor Web dentro do ESP32, de forma que ele possa lidar diretamente com requisições de rede feitas através do protocolo HTTP.

Para entender um pouco melhor a conectividade do ESP32, deve-se explicar um pouco os conceitos de rede abordados neste projeto.

2.1.2 Conceitos básicos de Redes de Computadores

A tecnologia de rede escolhida para a realização deste trabalho foi a Wi-Fi, também conhecida como protocolo IEEE 802.11. Portanto, serão definidos alguns conceitos de redes de computadores para melhor compreensão da metodologia do projeto no Capítulo 3.

2.1.2.1 Protocolo HTTP

De acordo com o livro de Jim Kurose e Keith Ross "Redes de computadores e a Internet: uma abordagem top-down"[6], o Protocolo de Transferência de Hipertexto (*HyperText*

Transfer Protocol) é definido pela *Internet Engineering Task Force* pelos *RFCs* 1945, 7230 e 7540 como o principal protocolo da camada de aplicação da internet. Ele possui a função de transferir objetos entre clientes e servidores a partir de uma conexão segura realizada graças a um outro protocolo que está em sua essência, o TCP. Com o HTTP, é realizada a requisição de, por exemplo, páginas *web*, como a que foi utilizada neste trabalho para realizar o controle dos potenciômetros digitais.

2.1.2.2 IPs locais

Dentro do conceito de Internet das Coisas, deve ser bem entendido o que é o principal protocolo da Internet: o *Internet Protocol (IP)*. que é o protocolo responsável, resumidamente, por aferir endereços numéricos a sites na internet. O tipo de endereçamento IP utilizado neste trabalho é o IPv4. Desses endereços IP, há endereços tidos como privados. Isto é, endereços IP que existem apenas dentro de uma rede privada, não detectável por dispositivos de fora dessa rede. Esses endereços privados seguem a seguinte lógica:

Um endereço IPv4 basicamente é composto por 4 números de 0 a 255 separados por pontos. Ao se enxergar esses números na base numérica binária, são quatro números representáveis por 8 bits. Em algum local entre esses 4 octetos é definida uma divisória. Essa divisória delimita a parte do IP pertencente a rede e a parte que pertence à **subrede**. Todos os dispositivos conectados a uma mesma rede devem ter o mesmo prefixo de endereço IP e se diferenciam apenas pelo sufixo, que é a porção da subrede. Cada dispositivo terá seu endereço dentro da disponibilidade oferecida pelos bits restantes do endereço da rede.

Os endereços de IP ditos privados seguem um padrão mundial quando se trata do tipo de endereçamento IPv4. As faixas de endereços IP privados podem ser vistas na Tabela 1.

Classe	Faixa de Endereço	Máscara de Sub-rede Padrão	Número de Endereços
A	10.0.0.0 a 10.255.255.255	255.0.0.0 (ou /8)	16.777.216
B	172.16.0.0 a 172.31.255.255	255.240.0.0 (ou /12)	1.048.576
C	192.168.0.0 a 192.168.255.255	255.255.0.0 (ou /16)	65.536

Tabela 1 – Faixas de endereços IP privados

2.1.2.3 Linguagem de formatação HTML

HTML (HyperText Markup Language) é a linguagem padrão utilizada para criar e estruturar páginas *web*. Ela utiliza uma série de "*tags*"(marcadores) que indicam ao navegador como exibir o conteúdo, como textos, botões, *links* e outros elementos. O HTML permite a criação de páginas interativas e é a base para a construção de *websites*, formando a estrutura que, combinada com CSS (*Cascading Style Sheets*) e *JavaScript*, proporciona um conteúdo visualmente atraente e dinâmico. Neste trabalho, utilizou-se marcadores de cabeçalho como **<h1>** e **<h2>** e também botões que interagem com o código do Arduino IDE.

2.2 A Eletrônica no Mundo Musical

A história da eletrônica no mundo musical começou ha quase um século atrás. Como neste trabalho haverá o foco na guitarra elétrica, é importante salientar que, de acordo com o *site Mission Engineering* [9], especificamente no ano de 1931 com a criação do primeiro amplificador de guitarra pela empresa Electro String Company, surgiu uma enorme gama de ideias para aplicar os conhecimentos de eletrônica no mundo musical. Com isso, era de se esperar uma transformação da história da música para sempre.

A história da eletrônica digital e a da música estão intimamente conectadas desde então. Com a criação de diversos equipamentos como sintetizadores, *pads*, pedais digitais e, atualmente, com a internet. O que leva a uma gama de possibilidades ainda maior para o mercado musical explorar mais efeitos e uma performance ainda melhor do músico que domina essa tecnologia que está mais acessível nos dias atuais.

Pedais de guitarra também possuem uma lógica parecida na parte de pré-amplificação, para trabalhar com o áudio de forma mais detalhada e precisa, é possível aplicar diversas perturbações no áudio de forma a deixá-lo da forma desejada pelo engenheiro responsável.

2.2.1 A lógica do Jack P10

O primeiro componente a ser abordado para a confecção deste trabalho é o jack P10, ou conector TS, ou TRS. Basicamente, o formato P10 é um padrão de entrada para equipamentos musicais e sinais de áudio de forma geral, principalmente de instrumentos elétricos como a guitarra. É também o padrão de entrada que será utilizado nesse trabalho pra entrada e saída do som do instrumento no circuito de *delay*.

De forma simples, o som do jack P10 é lido por um componente do tipo fêmea semelhante ao da Figura 3, no qual deve ser colocado um componente do tipo macho. A lógica dele funciona da seguinte forma: há dois tipos principais de jacks P10, o TS (*Tip-Sleeve*) e o TRS (*Tip-Ring-Sleeve)*. Para entender a diferença entre eles, primeiramente deve ser compreendida a diferença entre som estéreo e som mono.

Figura 3 – Conector de entrada-base fêmea do tipo P10. Fonte: Cabos Golden.

De acordo com o artigo no site *Ditto Music* [10], o som mono é nada mais do que a junção de todas as partes de um som em um único sinal, de forma a condensar o som para uma leitura direta por um único canal. Já a tecnologia estéreo divide o som em duas partes para uma experiência mais completa do ouvinte, isto é, um canal positivo e outro negativo.

Como pode ser observado na Figura 4, no conector TS existe uma entrada do tipo *Tip*, da qual é lido o sinal de áudio propriamente dito de forma elétrica pelo componente de entrada P10. E também há uma entrada do tipo *Sleeve*, que deve estar aterrada. Já o conector TRS, há uma entrada intermediária chamada *Ring*, que é basicamente uma parte negativa no áudio, que é útil para o caso de sons estéreos.

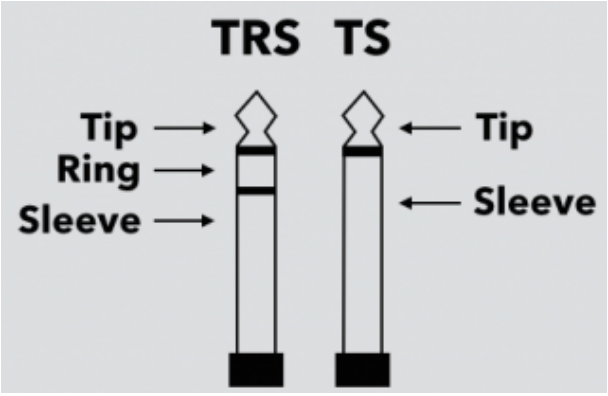

Figura 4 – Esquemático das entradas P10 TS e TRS. Fonte: electronicsartisanedynel.blogspot.com.

2.2.2 Placa de Ecos e Reverberação: PT2399

Será introduzido agora o principal componente responsável pelo efeito gerado nesse projeto. A placa de reverberação PT2399 inicialmente não é o produto final mostrado na Figura 5. Mas sim o circuito integrado que está na placa, ele é responsável pela geração de efeitos de ecos. O potenciômetro presente na placa vermelha do circuito é responsável pelo

volume desse efeito gerado. Quanto mais elevado, mais notável o efeito se torna aos ouvidos. Basicamente ele eleva a potência do efeito que sai da sessão responsável pelo efeito.

Figura 5 – Placa de Reverberação PT2399. Fonte: amazon.com.

O produto final carrega o circuito integrado que pode ser visto na Figura 6. Este sim, responsável pela geração dos efeitos. Cada pino carrega uma função única, como pode ser observado na própria imagem.

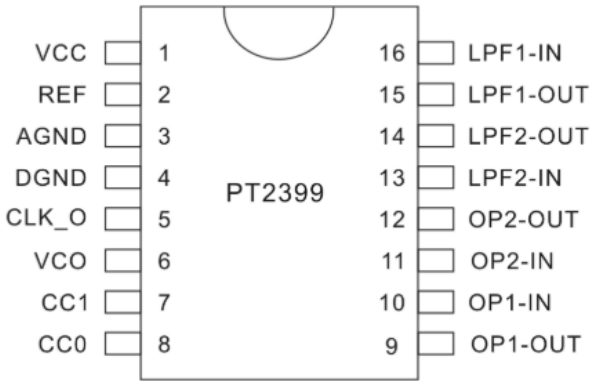

Figura 6 – Chip responsável pelos ecos da placa PT2399. Fonte: [2]

Para destacar o que é relevante, há um circuito específico responsável pela geração do efeito de eco, isto é, o *delay*. Uma configuração específica de componentes é soldada na placa vermelha presente na Figura 5. Após essa confecção a nível eletrônico, é possível tratar o circuito como uma caixa preta. Como pode ser observado na Figura 7.

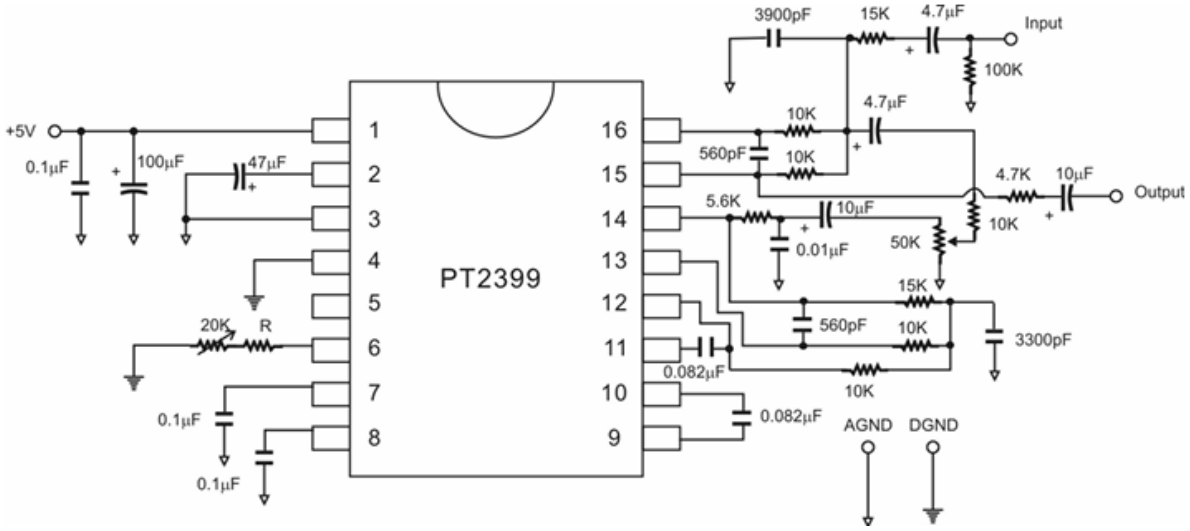

Figura 7 – Circuito completo gerador de Ecos com a PT2399. Fonte: [2]

É possivel observar, na Figura 7, o potenciômetro conectado em série com o resistor *R* na entrada 6. De acordo com o *datasheet* da placa [2], essa entrada é a reguladora da frequência do Oscilador Controlado por Voltagem *(Voltage Controlled Oscilator - VCO)*. Responsável pela geração do tempo entre os ecos. Basicamente, esse é o principal potenciômetro responsável pelo efeito de *delay*. Quanto maior a resistência de *R*, maior será a frequência do *VCO* e, como consequência, menor será o tempo entre os ecos gerados. Esse tempo entre os ecos se chama *feedback*.

Para este trabalho, os circuitos das Figuras 6 e 7 foram considerados caixas pretas. O circuito final pode ser visto como um sistema em malha fechada, como mostrado na Figura 8. Neste circuito, o sinal se retroalimenta dele mesmo com uma amplitude menor. O que é basicamente o que um circuito de *delay* faz.

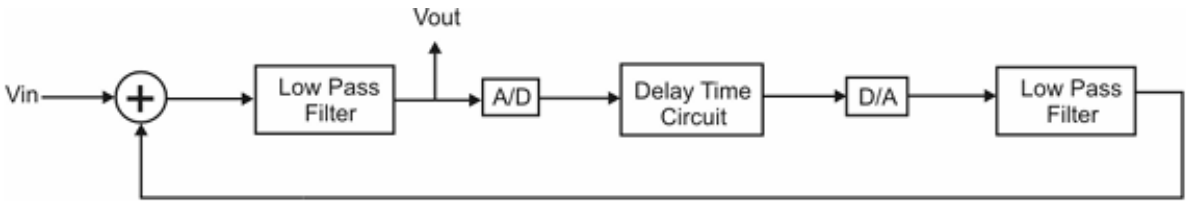

Figura 8 – Sistema linear equivalente do circuito completo gerador do efeito de ecos. Fonte: [2]

Com esse esquema é possível ter uma noção mais simplificada de como funciona o circuito. O sinal analógico primeiramente passa por um filtro passa-baixas, de forma a eliminar frequências indesejadas. Em seguida, passa pelo processo de quantização para se tornar um sinal digital. Por fim, seu volume é reduzido no circuito de *delay*, novamente é filtrado e retorna ao início para passar pelo mesmo filtro passa-baixas inicial e ser reproduzido. Esse processo é cíclico e decadente até que se torna nulo.

Quanto maior a resistência de *R*, maior será o tempo entre um eco e o outro. Isso ocorre porque a corrente que chega ao *VCO* é alterada, o que faz com que a tensão mude e assim, a frequência do gerador de funções oscilatórias também se torna diferente. Basicamente, existe uma relação proporcional entre o tempo entre os ecos e a resistência de *R*, que pode ser encontrada na página 8 do datasheet do CI PT2399 [2].

2.2.3 O Potenciômetro Digital X9C104S

Após a criação do potênciometro analógico, um dispositivo capaz alterar a resistência entre seus dois pontos a partir da rotação de uma chave, será apresentado agora um dispositivo que realiza exatamente isso de forma digital e possibilita essa alteração de forma remota. Esse dispositivo é conhecido como potenciômetro digital, e pode ser visualizado na Figura 9.

Figura 9 – O potenciômetro digital X9C104S (Visão superior e inferior). Fonte: Yonggan81's shop

Uma empresa especializada em potenciômetros chamada *Xicor* produziu um circuito integrado controlável via microcontroladores, um potenciômetro digital chamado *X9C104S*. Este potenciômetro pertence a uma linha de dispositivos chamada de *X9C102/103/104/503*, cada um desses potenciômetros possui uma resistência máxima diferente. No caso, o *X9C104S* possui uma resistência de até 100 kΩ.

De acordo com o *datasheet* do *X9C104S* [9], ele possui um decodificador de 1 para 100 em sua estrutura. Cada saída desse decodificador está conectada a um ponto de contato ligado a um arranjo de 100 resistores e a ativação desses pontos é controlada por meio de comandos específicos que são enviados a o potenciômetro digital, ativando os seus resistores. O diagrama funcional do potenciômetro digital *X9C104S* pode ser visto na Figura 10.

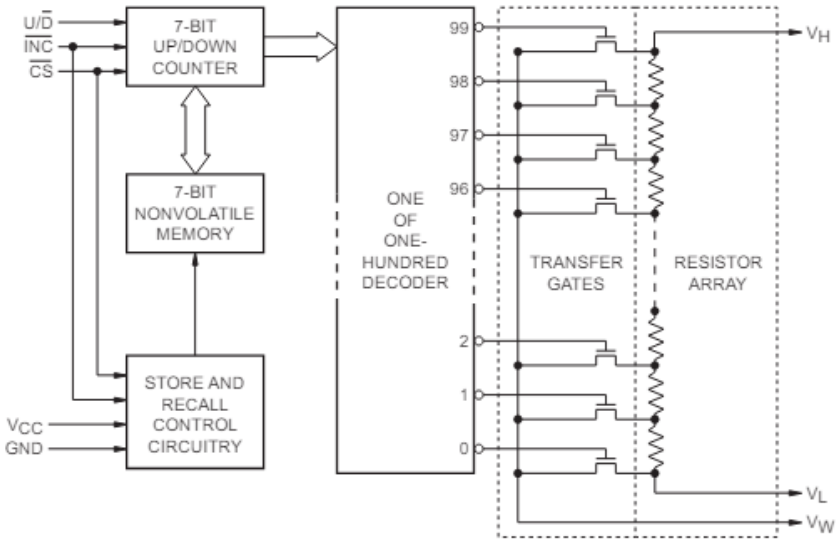

Figura 10 – Diagrama de funções do X9C104S. Fonte: [4]

Agora partindo para a pinagem desse circuito integrado, presente na Tabela 2 com suas respectivas funções.

Tabela 2 – Descrição dos terminais.

Símbolo	Descrição
RH	Terminal Alto do Potenciômetro
RW	Terminal do Limpador do Potenciômetro
RL	Terminal Baixo do Potenciômetro
GND	Aterramento
VCC	Tensão de Alimentação (5V)
U/D	Entrada Subida/Descida
INC	Indicador de Incremento
CS	Seleciona Chip para salvamento

As entradas **Terminal Alto**, **Terminal Baixo** e **Terminal do Limpador** são as tradicionais entradas de um potenciômetro analógico comum.

É possível então realizar o controle de aumento e diminuição da resistência do potenciômetro através da manipulação dos bits que estarão sendo lidos pelo dispositivo. Há 8 tipos entradas no potenciômetro digital. Representadas pelos símbolos da Tabela 2. Dessas entradas, três são ditas como de controle, duas como alimentação (GND e VCC) e três são para o potenciômetro em si (RH, RL e RW).

As entradas de controle são o *CS: Chip Select, U/D: Up/Down* e *INC: Incremento*. A mecânica funciona da seguinte forma: cada entrada dessa possui apenas duas opções de leitura, que são zero e um. O Chip Select deve estar com a entrada em zero e o Incremento deve estar com a entrada em um inicialmente. Em seguida, a leitura da entrada U/D representará, no momento, se o usuário deseja subir (valor 1) ou descer (valor 0) a resistência do potenciômetro.

Supondo que o usuário deseje elevar a resistência, a entrada U/D deve receber o valor 1. Após isso, a entrada INC deve receber o comando para descer de 1 para 0. Isso ocasionará o fechamento da chave logo acima da pré-definida anteriormente. Por exemplo, se o potenciômetro estiver marcando 50 kilo ohms, ao executar esses comandos ele subirá para 51 kilo ohms. Em seguida, ao elevar o CS para 1, esse valor de resistência será registrado na memória interna do potenciômetro. Um novo valor de resistência poderá ser definido após isso, elevando novamente o Incremento para o valor 1 e depois descendo novamente, o que fará com que a entrada U/D seja lida novamente e um novo incremento (ou decremento) seja realizado. A entrada CS é responsável unicamente pelo salvamento do valor de resistência na memória ao ir de 0 para 1. Um esquemático visual dessa mecânica pode ser visto na Figura 11.

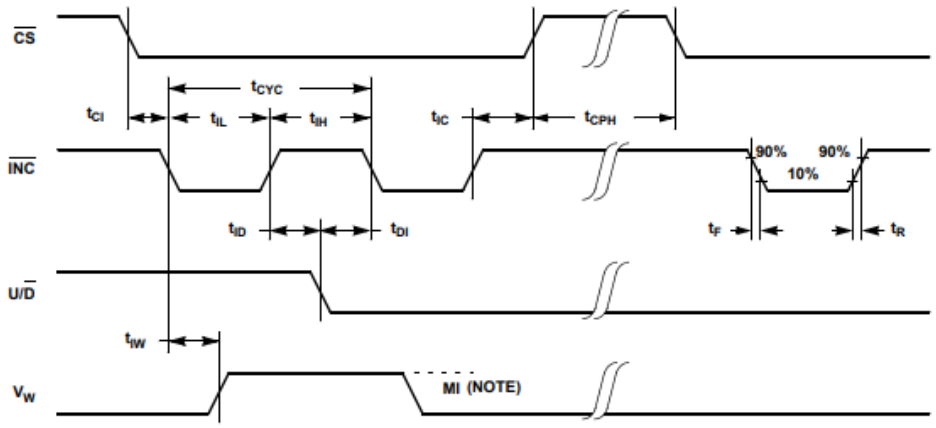

Figura 11 – Diagrama de tempo do X9C104S. Fonte: [4]

As demais entradas do potenciômetro são mais simples de compreender. Resumidamente o *VCC* é a alimentação de 5 volts; o *GND* é o referencial terra (0 volts); E, por fim, as entradas RH, RW e RL são respectivamente as entradas positiva, limpadora e negativa padrões de potenciômetros analógicos. De maneira resumida, este é o funcionamento geral do potenciômetro digital X9C104S, responsável pelo controle digital da resistência que será modificada via rede neste projeto. As informações aqui presentes foram todas retiradas do datasheet do potenciômetro X9C104S [9].

3 Metodologia

Neste capítulo será apresentada a forma como foi desenvolvida a parte experimental deste trabalho. Primeiramente, deve-se lembrar que o principal objetivo aqui é demonstrar a possibilidade, averiguar a performance e eficiência do controle de efeitos musicais analógicos por meio da tecnologia de Internet das Coisas, isto é, a Internet das Coisas Musicais.

3.1 Ajustes na placa de efeito e nos potenciômetros digitais

Primeiramente, deve-se ter a noção de como se dispõe inicialmente uma placa do tipo PT2399. Como mostra a Figura 12, há dois pinos para *Input* e dois para *Output*, um positivo e outro negativo. Os sinais positivos serão as leituras do próprio áudio. No caso, o *input* positivo recebe a entrada *tip* do Jack P10 que vem diretamente da guitarra. Já o positivo do *output* irá para o *tip* do Jack P10, recebendo o áudio já processado pela placa e chegando de vez ao amplificador. Analogamente os pinos de sinal negativo vão estar conectados ao aterramento.

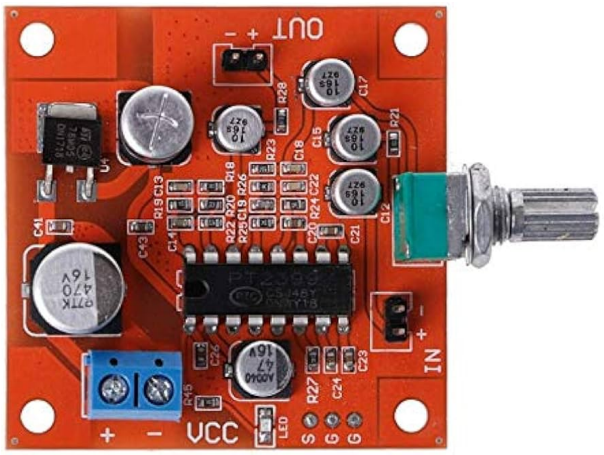

Figura 12 – Vista superior da placa PT2399. Fonte: walmart.ca

Em seguida, os terminais positivo e negativo do VCC, que estão revestidos por uma proteção de cor azul, devem estar conectados a uma fonte de 12 V de acordo com o Datasheet da placa *PT2399* [2] para que haja o funcionamento desejado. Além disso, o negativo do VCC deve estar aterrado também.

Além das alimentações e das entradas e saídas de áudio, é possível observar que há um potenciômetro verde soldado na placa. Esse potenciômetro é responsável pela regulação do volume do efeito de *delay*. Se ele está em seu valor máximo não haverá eco acontecendo.

Porém, conforme a resistência diminui há um aumento do volume dos ecos de forma que quanto maior a resistência, maior o eco.

Entretanto há um detalhe importante. Logo ao lado direito da escrita **VCC**, é possivel observar três entradas para pinos de contato na placa com as letras S, S e G. Esses pinos são entradas que podem receber o contato de um potenciômetro. Um potenciômetro neste local, conforme visto na Figura 7, é aquele que está em série com *R*. Ou seja, ele estará influenciando diretamente a entrada do *VCO* do circuito integrado PT2399. Ou seja, é uma entrada para um segundo potenciômetro, este responsável pelo tempo entre os ecos no efeito de *delay*.

Para que o trabalho possa ser executado, foi necessária a retirada do potenciômetro verde, para que ele pudesse ser substituído por três pinos de contato. Estes pinos serão a entrada para o controle por meio do potenciômetro digital *x9c104s*. Similarmente, foram soldados três pinos de contato também nas entradas S, S e G ao lado direito da alimentação da placa.

Esse processo foi realizado com o auxílio de técnicos de laboratório do SG-11 da Universidade de Brasília. Além disso, também foram soldadas as entradas dos dois potenciômetros digitais que atuam para o controle do Volume e do chamado *Feedback* do efeito de ecos.

3.2 Confecção do Circuito completo

Após a retirada do potenciômetro verde, das soldas dos pinos de contato nas entradas de leitura de potenciômetro na *PT2399* e das soldas de pinos de contato nos dois potenciômetros digitais, constrói-se, então, o circuito.

Primeiramente, é importante relembrar que de acordo com o que é dito no *datasheet* do potenciômetro digital [4], a entrada *VCC* deve receber uma alimentação de 5 V. Portanto, uma solução possivel é conectar essas entradas *VCC* dos potenciômetros *X9C104S* à entrada *Vin* do ESP32, que, de acordo com o *datasheet* do ESP32 [8], oferece essa alimentação. Porém, a alimentação da Placa de Ecos *PT2399* deve ser alimentada com 12 V. Nesse caso, a solução encontrada foi conecta-la diretamente a uma fonte de 12 V alimentada por energia elétrica.

Aqui deve estar clara a divisão entre o que é parte do circuito de controle e o que é o circuito de comando. A confusão entre esses dois pode causar danos irreversíveis aos potenciômetros digitais e ao microcontrolador. Em suma, as entradas **RH**, **RW** e **RL** podem receber o sinal que vem da placa PT2399 pois irão regulá-la, enquanto as entradas **INC**, **U/D** e **CS** recebem comandos da ESP32 para realizar o aumento e a diminuição da resistência fornecida pelos potenciômetros digitais.

Já os sinais de áudio são lidos pelos pinos positivos da placa de reverberação PT2399

e não estão em contato com os potenciômetros diretamente, já que eles não são feitos para suportar ondas de áudio e não precisam processar esses dados. O mais importante de tudo é isolar a tensão de 12 V da de 5 V para que não haja danos a qualquer dispositivo. Apesar disso, é recomendado que as entradas *GND* estejam compartilhadas entre ambos, para também evitar qualquer dano ou disparidade no funcionamento dos componentes.

Os sinais de áudio provêm de uma guitarra elétrica, plugada a um jack P10 que mandará, por meio do pino de contato *Tip*, várias frequências diferentes que serão lidas pela placa *PT2399*. Já o sinal de áudio processado irá sair do pino positivo *Out* e irá diretamente ao *Tip* de um outro jack p10, que estará plugado diretamente a um amplificador. Esse esquemático foi montado propriamente para a melhor visualização deste trabalho e pode ser visto na Figura 13.

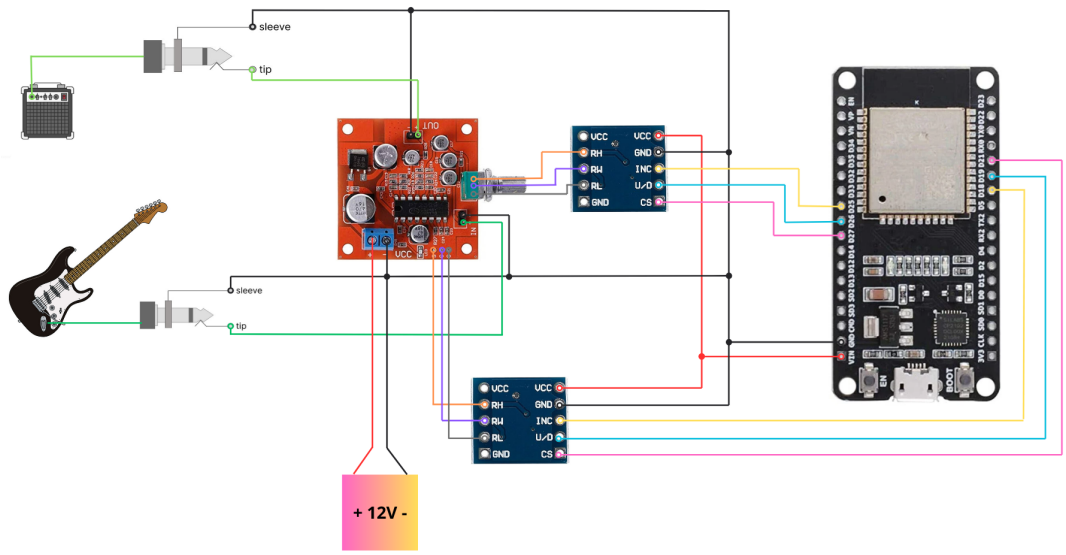

Figura 13 – Diagrama do projeto completo. Figura de autoria própria.

Na Figura 13, é possível perceber como os potenciômetros influenciam diretamente no efeito que é aplicado ao som da guitarra que sai pelo amplificador. Além disso, os pinos do ESP32 escolhidos para o controle desses potenciômetros podem ser observados na Tabela 3. Com esses dados, pode-se realizar a criação do código no Arduino IDE.

3.3 Código no Arduino IDE

Nesta seção haverá foco no código construído no próprio Arduino IDE, objetivando uma compreensão didática mais simplificada e será explicada a construção do código feito para o ESP32, com a finalidade de fazer o projeto funcionar da forma que foi descrita ao longo deste trabalho.

Pino do ESP32	Conexão	Unidade Controlada
18	INC	Feedback
19	U/D	Feedback
21	CS	Feedback
25	INC	Volume
26	U/D	Volume
27	CS	Volume

Tabela 3 – Organização dos pinos de controle da ESP32 para o projeto.

3.3.1 Passos iniciais

Primeiramente, deve-se importar as bibliotecas e declarar as variáveis globais. Esses parâmetros esão presentes na figura 14.

```
1   #include <WiFi.h>
2   #include <WebServer.h>
3
4   // Definições de pinos para controle do potenciômetro digital
5   const int pinoINC1 = 18;
6   const int pinoUD1 = 19;
7   const int pinoCS1 = 21;
8
9   const int pinoINC2 = 25;
10  const int pinoUD2 = 26;
11  const int pinoCS2 = 27;
12
13  // Definições para o ponto de acesso
14  const char* ssid = "IoMusT";
15  const char* password = "guitarra";
```

Figura 14 – Importando bibliotecas e declarando variáveis globais.

No Capítulo 2, ficou clara a necessidade do uso das bibliotecas **WiFi.h** e **Webserver.h**. Elas serão as únicas bibliotecas necessárias para esse código funcionar adequadamente. Em seguida, declaram-se seis variáveis globais para controlar o potenciômetro digital nas portas indicadas na Tabela 3. As variáveis com o término igual a 1 serão as controladoras do potenciômetro responsável pelo *feedback*, enquanto as variáveis cujos nomes estão com o último caractére igual a 2 são designadas ao controle do potenciômetro responsável pelo volume.

Um dos objetivos é transformar o ESP32 em um ponto de acesso com a biblioteca **WebServer.h**. Para isso, deve-se definir variáveis globais do tipo *string* ou simplesmente ponteiros de *const char* para guardar o nome da rede *(SSID)* e a senha para poder acessá-la. No caso, escolheu-se declarar duas variáveis globais: o SSID escolhido foi *IoMusT* e a senha

guitarra.

3.3.2 Página HTML

Em seguida, foi criada uma função chamada de **buidHTML()**, que retorna uma *string* contendo o código da página HTML que funcionará como interface para o controle do circuito.

```
20  // Função para construir a página HTML
21  String buildHTML() {
22    String html = "<html>\
23    <body>\
24      <h1>Controle de Delay</h1>\
25      <h2>Volume:</h2>\
26      <button onclick=\"location.href='/diminuir2'\">Aumenta</button>\
27      <button onclick=\"location.href='/aumentar2'\">Diminui</button>\
28      <h2>Feedback:</h2>\
29      <button onclick=\"location.href='/aumentar1'\">Aumenta</button>\
30      <button onclick=\"location.href='/diminuir1'\">Diminui</button>\
31      <h2>Liga/Desliga:</h2>\
32      <button onclick=\"location.href='/liga'\">Liga</button>\
33      <button onclick=\"location.href='/desliga'\">Desliga</button>\
34    </body>\
35  </html>";
36    return html;
37  }
```

Figura 15 – Função que cria a página HTML.

É importante notar que ao final de cada linha há uma barra invertida. Essa barra representa a quebra de linha para que o ESP32 possa interpretar a página e o protocolo possa encaminhar o código HTML de forma compreensível ao microcontrolador. A função pode ser vista na Figura 15.

3.3.3 Controle dos Potenciômetros

Na Figura 16 está representada a função criada que aumenta a resistência do potenciômetro. No caso, essa função tem um nome que termina com o número 1. Ou seja, ela foi feita para aumentar o *feedback*. A escolha de criar uma função de número 1 e outra de número 2 para realizar crescimento e o decrescimento das resistências ocorre para que possam ser separadas metodicamente as variáveis e as funções responsáveis por volume e por *feedback*. Isto se dá porque a função de aumento de resistência do volume possui uma lógica invertida. Além disso, a função **delay()** tem o papel de pausar o programa pela unidade de milissegundos que a função recebe como argumento, objetivando proporcionar o tempo necessário para que as alterações nos potenciômetros digitais ocorram. Esses dados estão explicados no *datasheet* do potenciômetro digital [4].

```
// Função para aumentar a resistência
void aumentarResistencia1() {
  for (int i = 0; i < 10; i++) {
    digitalWrite(pinoCS1, LOW);
    delay(5);
    digitalWrite(pinoUD1, HIGH); // Aumenta
    delay(5);
    digitalWrite(pinoINC1, HIGH);
    delay(5);
    digitalWrite(pinoINC1, LOW);
    delay(5);
    digitalWrite(pinoINC1, HIGH);
    digitalWrite(pinoCS1, HIGH);
    delay(10);
  }
  server.send(200, "text/html", buildHTML());
}
```

Figura 16 – Função que aumenta o feedback aumentando a resistência do potenciômetro.

De forma simplificada, ao se aumentar a resistência do potenciômetro responsável pelo volume, o volume é diminuido. Isso faz com que seja necessária a inversão da lógica entre aumento da resistência e diminuição do volume. Ou seja, a função **aumentaresistencia2()** é responsável por diminuir o volume, e a **diminuiresistencia2()** aumenta o volume. A função **diminuiresistencia2** pode ser vista na Figura 17.

```
// Função para diminuir a resistência
void diminuirResistencia2() {
  for (int i = 0; i < 10; i++) {
    digitalWrite(pinoCS2, LOW);
    delay(5);
    digitalWrite(pinoUD2, LOW); // Diminui
    delay(5);
    digitalWrite(pinoINC2, HIGH);
    delay(5);
    digitalWrite(pinoINC2, LOW);
    delay(5);
    digitalWrite(pinoINC2, HIGH);
    digitalWrite(pinoCS2, HIGH);
  }
  server.send(200, "text/html", buildHTML());
}
```

Figura 17 – Função que eleva o volume do efeito diminuindo a resistência do outro potenciômetro.

Importante também ressaltar dois pontos. Primeiramente a função **server.send()**, pertencente à biblioteca WiFi.h [5], é responsável por enviar a requisição HTTP com o valor

do potenciômetro atualizado após a execução da estrutura de repetição **for**. E também, a escolha de aumento em passos de valores de 10 a cada clique foi uma escolha para tornar mais visíveis as alterações no áudio que estará sendo reproduzido.

Além disso, foram criadas outras duas funções. Uma chamada **ligaefeito()** e outra chamada **tirarefeito()**. Na prática, elas são respectivamente responsáveis por aumentar e diminuir o volume totalmente. Isto é, são semelhantes as funções de aumentar e diminuir o volume, porém, o **for** dessas funções não varia em passos de 10, mas de 100. Fazendo com que todos os resistores de dentro do potenciômetro digital responsável pelo volume sejam ativados (no caso da função **tirarefeito()**) ou desativados (no caso da função **ligaefeito()**). Seria semelhante a ligar ou desligar um pedal de efeito de *delay*. Na Figura 18 pode ser observada a função **ligaefeito()**.

```
126  void ligaefeito() {
127    for (int i = 0; i < 100; i++) {
128      digitalWrite(pinoCS2, LOW);
129      delay(5);
130      digitalWrite(pinoUD2, LOW); // Diminui
131      delay(5);
132      digitalWrite(pinoINC2, HIGH);
133      delay(5);
134      digitalWrite(pinoINC2, LOW);
135      delay(5);
136      digitalWrite(pinoINC2, HIGH);
137      digitalWrite(pinoCS2, HIGH);
138    }
139    server.send(200, "text/html", buildHTML());
140  }
```

Figura 18 – Função que eleva ao máximo o volume do efeito de delay. Zerando a resistência do potenciômetro digital.

3.3.4 Funções setup() e loop()

Aqui, finalmente, teremos as principais funções que serão lidas pelo ESP32. As funções principais. A função **setup()**, que é executada uma única vez ao se iniciar o programa no ESP32 e a função **loop()**, que é executada repetidamente.

Antes de se aprofundar nessas funções, é importante salientar a existência de uma função adicionar que é responsável por fazer com que a página HTML seja interpretada finalmente pelo ESP32. Essa função se chama **handleRoot()** e pode ser averiguada na Figura 19.

```
142  // Função para configurar a página inicial do servidor web
143  void handleRoot() {
144    server.send(200, "text/html", buildHTML());
145  }
```

Figura 19 – Função handleRoot(), responsável pelo envio da página ao ESP32.

Então, primeiramente, a função **setup()**, presente na Figura 20. Realiza as seguintes funções:

```
147  void setup() {
148    // Configuração dos pinos de controle
149    pinMode(pinoINC1, OUTPUT);
150    pinMode(pinoUD1, OUTPUT);
151    pinMode(pinoCS1, OUTPUT);
152    digitalWrite(pinoCS1, HIGH); // Ativa Resistor
153
154    pinMode(pinoINC2, OUTPUT);
155    pinMode(pinoUD2, OUTPUT);
156    pinMode(pinoCS2, OUTPUT);
157    digitalWrite(pinoCS2, HIGH); // Ativa Resistor
158
159    // Configuração do ponto de acesso
160    WiFi.softAP(ssid, password);
161    Serial.begin(115200);
162    Serial.println();
163    Serial.print("IP do ponto de acesso: ");
164    Serial.println(WiFi.softAPIP());
165
166    // Configuração das rotas do servidor web
167    server.on("/", handleRoot);
168    server.on("/aumentar1", aumentarResistencia1);
169    server.on("/diminuir1", diminuirResistencia1);
170    server.on("/aumentar2", aumentarResistencia2);
171    server.on("/diminuir2", diminuirResistencia2);
172    server.on("/liga", ligaefeito);
173    server.on("/desliga", tirarefeito);
174    server.begin();
175  }
```

Figura 20 – Função setup().

1. Configura os pinos do ESP32 definidos na Tabela 3 como *OUTPUTs*, ou seja, pinos que irão **enviar** sinais e não recebê-los.

2. Pré salva o estado dos pinos responsáveis pelo controle da função de *Chip Select* dos potenciômetros digitais como ALTO, para que a lógica do diagrama estabelecido na Figura 11 possa ser executado da forma desejada desde o começo, havendo variação desse valor apenas quando houver variação das resistências nos potenciômetros.

3. Define, com a função **WiFi.softAP()**, o início da execução do servidor *web* dentro do ESP32 e o transforma em um ponto de acesso de Wi-Fi. Definindo também a velocidade com que o monitor serial do Arduino IDE deve ler os dados enviados do ESP32 para que estes possam ser interpretados pelo monitor. A taxa de transmissão do ESP32 DevKit V1 é, por padrão, 115200 *baud* (eventos por segundo). Isso torna possível a leitura das strings presentes nas funções **Serial.print()** por parte do *software*. Essas funções estão enviando a *string* "IP do ponto de acesso: "e o IP que foi atribuído ao ESP32. O IP atribuído pela função **WiFi.softAPIP()** é o que deve ser digitado para se acessar a página HTML gerada por esse programa na função **buildHTML()** presente na Figura 15. Por padrão, o *IP* atribuído pela função ao ESP32 é 192.168.4.1, que conforme explicado na Subsessão 2.1.2, é um endereço privado do tipo C que não se conecta a internet, apenas oferece acesso para a página ao ser digitado no navegador de internet.

4. Define 7 rotas que a página HTML possui por meio da função **server.on()**. Primeiramente, estabelece que o local inicial da página deve ser ela mesma, chamando a função **handleRoot()**. Em seguida, define as rotas dos botões presentes na página HTML de aumentar e diminuir as resistências dos potenciômetros, discutidas já nesta sessão.

5. A função **setup()** é finalizada com a ativação final do servidor, por meio da função **server.begin()**.

Por fim, função **loop()** exerce a única função de manter o servidor ativo para lidar com possíveis requisições HTTP do cliente. Isto é, do usuário. A função está na figura 21. A função **handleClient()** realiza esse trabalho e deve estar ativa para que as requisições possam ser atendidas a qualquer momento durante a execução do programa.

```
177  void loop() {
178    // Lida com as requisições HTTP
179    server.handleClient();
180  }
```

Figura 21 – Função loop().

4 Análises e Resultados

Neste capítulo o funcionamento do programa será descrito, utilizado, analisado e serão apresentados e discutidos os resultados obtidos, avaliando se satisfazem a proposta do projeto a partir do programa criado, bem como as expectativas de funcionamento dentro do circuito que foi construído.

4.1 A rede

Primeiramente, para interagir com os componentes do circuito via rede, deve-se acessar a página HTML criada. Para isso, uma sequência de passos deve ser seguida.

De acordo com a função **setup()**, presente na Figura 20, após o início código com a ativação dos pinos de controle e com a ativação dos *Chip Select's*, há a configuração do ESP32 como ponto de acesso.

Ao executar o programa e enviá-lo ao ESP32, foi criada a rede "IoMusT", o que pode ser verificado na Figura 22. Ao clicar nela e inserir a senha "guitarra", o acesso é obtido com sucesso.

Figura 22 – Rede "IoMusT"detectável pelo computador.

A rede alega estar sem acesso a internet, o que é esperado uma vez que o *IP* 192.168.4.1 não está conectado a qualquer interface que dê acesso a internet.

4.2 A página web

Uma vez conectado na rede IoMusT, deve-se digitar o *IP* 192.168.4.1 no navegador *web* do dispositivo, que está com acesso a essa rede para se obter a página formatada na função **buildHTML()** da Figura 15. A página foi acessada com sucesso, conforme mostra a Figura 23.

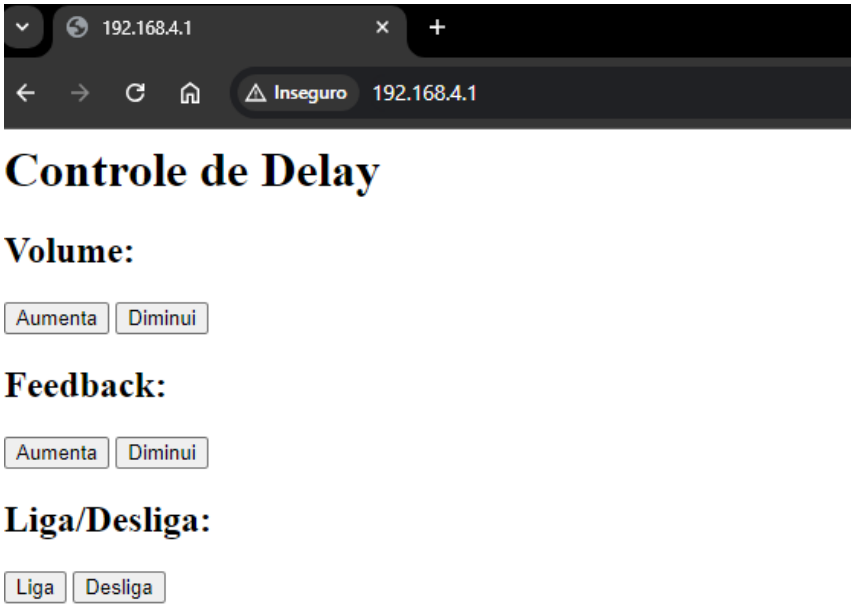

Figura 23 – Página gerada pela função buildHTML() acessível.

A página se encontra no estado esperado, com os botões formatados em HTML e os cabeçalhos da forma que foram programados na função.

4.3 Análise do áudio e a interação com o circuito via rede

Com a página HTML acessível e o circuito devidamente montado, propõe-se uma análise diretamente com um áudio para testar a variação de frequência a cada clique no botão de aumentar o *feedback* do efeito. Lembrando que cada clique no botão executa uma função que ativa um total de dez resistores dos cem que estão presentes dentro do circuito integrado *X9C104S*, cuja estrutura pode ser vista na Figura 10.

Após uma análise inícial, percebeu-se que a partir de 50 resistores ativados, o som torna-se muito difícil de ser distinguido. O que se deve a dois fatos: o de já haver um resistor em série com o potenciômetro digital, que está soldado na própria placa e, principalmente, porque há um aviso na mesma página em que se encontra a Figura 7 no *datasheet* da placa

PT2399 [2] que recomenda o uso de um resistor externo de até 50 kΩ, não de 100 kΩ. Portanto, idealmente a melhor ideia para este trabalho seria utilizar os potenciômetros do tipo *X9C503*, que não ultrapassam essa faixa de valor de resistência.

Uma solução possível para evitar esse problema seria a substituição desse resistor soldado por um curto circuito, pois isso causaria a diminuição da queda de tensão no resistor já soldado na placa aumentando a tensão que chega ao *VCO* do circuito integrado *PT2399*, o que diminuiria a sua frequência ainda mais para que valores de *delay* menores pudessem ser atingidos pelo usuário.

De todo modo, o experimento foi realizado com 0, 10, 20, 30 e 40 potenciômetros do potenciômetro digital ligados dos 100 presentes. Em seguida, um áudio de bipe do vídeo contido no link https://www.youtube.com/watch?v=WYG4Vke6WmQ foi reproduzido em um aparelho de celular. Aparelho conectado a um adaptador de P2 para P10, conectado a um cabo conectado ao jack p10 do circuito da Figura 13 no lugar da guitarra elétrica e, por fim, este som foi gravado para cada quantia de resistores ligada, variando por meio do clique no botão "Aumenta"da página HTML e analisado experimentalmente em um programa de edição de áudio e música em um *notebook*. Esse programa se chama REAPER e uma tela da análise pode ser observada na Figura 24. Os dados dessa análise foram coletados do programa e podem ser vistos na Tabela 4.

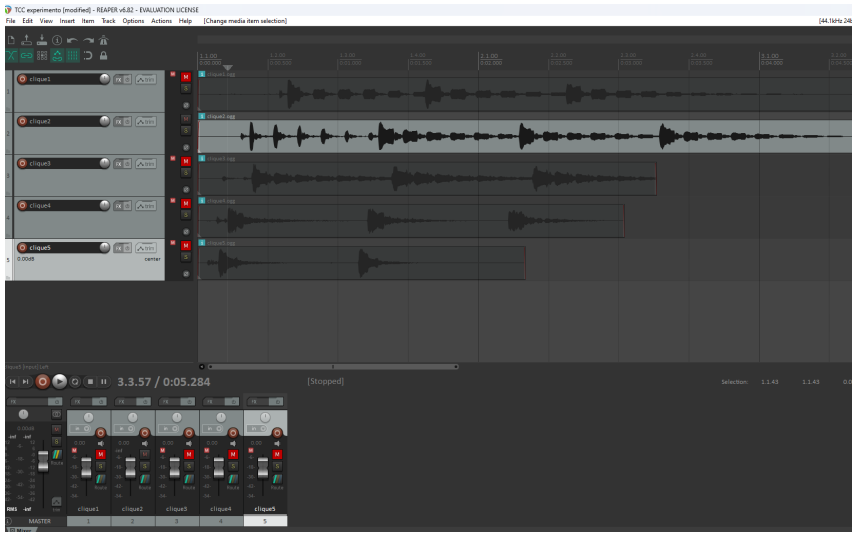

Figura 24 – Projeto do programa REAPER utilizado para análise de áudio.

Resistores Ligados no Potenciômetro	som 1 (ms)	som 2 (ms)	período (ms)	frequência (Hz)
0	643	823	0.18	5
10	624	793	0.169	5.917159763
20	544	694	0.15	6.66666667
30	415	472	0.057	17.54385965
40	364	389	0.025	40

Tabela 4 – Dados experimentais do áudio a partir da variação da resistência no potenciômetro digital.

Por fim, a partir da medição entre os intervalos entre os inícios dos picos dos bipes, foi montada a Tabela 4. Com os dados inseridos na tabela, foi montado um gráfico no *Google Spreadsheets* e pode-se perceber que a curva de frequência entre os ecos varia de forma que se aproxima de uma função exponencial. O gráfico pode ser visto na Figura 25.

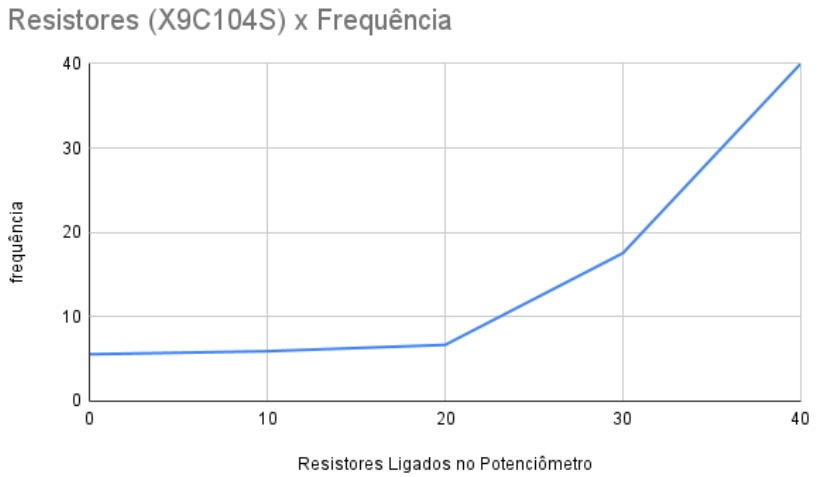

Figura 25 – Gráfico de número de resistores por frequência dos ecos.

4.4 Teste com a Guitarra

Com a guitarra e o amplificador conectados e ligados, a análise do circuito pode ser feita e o uso do efeito de *delay* realizado da forma esperada utilizando o próprio instrumento.

Para a realização dessa análise, foi realizado um vídeo próprio contendo comentários sobre o som. Este vídeo pode ser acessado pelo *link* https://abrir.link/EkMli. A análise obteve resultados satisfatórios e o som da guitarra permaneceu limpo, possuindo apenas pequenos elementos ruidosos naturais dos sons analógicos. O circuito funciona e o efeito está funcionando conforme o esperado.

Por fim, foi realizado o *upload* de um último vídeo. Vídeo realizado como teste para averiguação do efeito de *delay* na prática a partir da execução do *riff* de uma música chamada *Decode* da banda *Paramore*. Essa música foi escolhida pois ela possui em quase toda sua estrutura um *riff* de guitarra que faz uso intenso do *delay* com uma frequência que está presente na Tabela 4 com 20 resistores ligados. Além dessa música, outros *riffs* também foram criados de forma improvisada para testar o efeito da placa ao longo do vídeo. Esse vídeo pode ser acessado pelo *link* https://abrir.link/VhmdE.

Os *links* dessa sessão foram encurtados por meio do site https://abrir.link/.

5 Conclusões

A partir do término deste trabalho, concluiu-se que a Internet das Coisas Musicais é um campo com muito potencial, oferecendo inúmeras possibilidades de projetos que podem surgir ao longo dos anos para melhorar as apresentações dos músicos ao redor do mundo. Isso pode ser concluído pois o projeto aqui discutido e documentado foi feito com poucos recursos obtidos de maneira simples e que surtiu resultados surpreendentes, o que se dá justamente pelo fato de que a Internet das Coisas Musicais é uma área que começou a ganhar atenção recentemente e está ganhando força com o passar dos anos.

Conforme o que foi discutido na introdução deste trabalho, um dos objetivos foi justamente discutir a proposta de automatizar a troca de efeitos de forma a reduzir o trabalho feito pelo músico que está realizando sua apresentação. De fato, com o projeto aqui feito pode-se realizar a troca de efeitos sem um pedal de guitarra, apenas utilizando o circuito e com uma outra pessoa realizando as trocas no tempo certo a partir de um dispositivo conectado na rede do ESP32. O que já prova uma possível forma de facilitar os shows ao se possuir um técnico de som que conheça as músicas e saiba a hora de trocar os efeitos.

Além do desejo de se realizar um maior número de experimentos em um trabalho futuro, há a possibilidade de incrementar ainda mais o projeto a partir de algumas ideias futuras como, por exemplo, a automação da sincronia musical. Ou seja, seria possível evoluir esse trabalho a médio prazo de forma a criar um *software* capaz de contar os batimentos por segundo de uma música e, a partir de um certo número de compassos, a troca ser realizada sem a necessidade de um técnico de som ou uma outra pessoa responsável pela manipulação do código, o que já dispensaria a necessidade de um técnico de som para essa função específica. Essa ideia é possível e pode ser um próximo passo para uma larga geração de novos produtos musicais automatizados.

Uma outra ideia possível seria a criação de outros efeitos. Este trabalho utilizou apenas o caso do *delay*, porém é possível realizar diversos circuitos eletrônicos que geram diversos efeitos diferentes para guitarra e que podem ser automatizados similarmente a partir de um microcontrolador e controlado via rede. Podendo ser possível realizar uma música inteira e até shows completos com várias trocas de efeito automatizadas sem a necessidade de se pisar em um pedal.

Em resumo, é possível solucionar o problema da troca de efeitos e prover uma liberdade maior ao músico que está realizando seu show a partir dos conhecimentos de Engenharia Elétrica e da Internet das Coisas Musicais com grande custo benefício e de forma criativa. Havendo várias possibilidades para o músico engenheiro de criar suas soluções afim de facilitar seus números musicais e também o de pessoas do seu meio de convivência.

6 Referências bibliográficas

1. Turchet, Luca; Fischione, Carlo; Essl, Georg; Keller, Damián; Barthet, Mathieu; Internet of Musical Things: Vision and Challenges (2018); IEEE Access; Volume 6, páginas 61994 a 62017.

   Disponível https://ieeexplore.ieee.org/document/8476543

   Acessado em Maio de 2024.

2. Princeton Technology Corporation (2005). PT2399 Echo Processor IC Datasheet.

   Disponível em https://electricdruid.net/datasheets/PT2399.pdf

   Acessado em Maio de 2024.

3. Vieira, Rômulo; Sunflower: uma proposta de padronização para ambientes de Internet das Coisas Musicais; Universidade Federal de São João Del-Rei; Dissertação de Mestrado em Ciência da Computação (2021).

   Disponível em https://abrir.link/sHrGP

   Acessado em Maio de 2024.

4. Xicor INC (1994). X9C104S Potentiometer IC Datasheet.

   Disponível em https://abrir.link/sydeB

   Acessado em Maio de 2024.

5. Documentação do Sistema Embarcado Arduino. Bibliotecas WiFi.h e WebServer.h.

   Disponível em https://www.arduino.cc/reference/en/libraries/wifi/

   Acessado em Agosto de 2024.

6. Kurose, Jim; Ross, Keith; Redes de Computadores e a Internet: Uma abordagem top-down (8a edição); Editora Pearson; São Paulo; 2021.

7. Maxim Integrated. Audio Gain Control Using Digital Potentiometers.

   Disponível em https://pdfserv.maximintegrated.com/en/an/AN1828.pdf

   Acessado em Setembro de 2024.

8. Espress If. ESP32 Series Datasheet.

   Disponível em https://abrir.link/nlxpE

   Acessado em Maio de 2024.

9. Mission Engineering. Early History of the Amplifier

   Disponível em https://missionengineering.com/early-history-of-the-amplifier/

   Acessado em Setembro de 2024.

10. Ditto Music; Som mono vs som estéreo: qual você deve escolher e por quê?

    Disponível em https://dittomusic.com/pt/blog/mono-vs-stereo-sound-which-should-you-choose-and-why

    Acessado em Setembro de 2024.

11. André Batista; Whiplash; Notícias; Fuzz: a história do avô de todos os pedais de distorção

    Disponível https://abrir.link/PwnKs

    Acessado em Setembro de 2024.

12. Ghadiali, Zahid; 6G and the Internet-of-Musical Things (2023)

    Disponível https://abrir.link/muTtp

    Acessado em Setembro de 2024.

13. Viola, Olavo; Piovesan, Erick; Música: um estudo físico matemático sobre o som através da série de Fourier e do núcleo de Fejér com o uso de ferramentas espectrais (2022); Universidade Federal de Uberlândia; Projeto Acadêmico de Pesquisa do curso de Física;

    Disponível em https://www.scielo.br/j/rbef/a/rL4X9n6rVhY6ZJczHNRNjqH/

    Acessado em Setembro de 2024.