UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

FELIPE ARTHUR POVALUK DA SILVA

IMPLEMENTAÇÃO DE UM TEREMIM DIGITAL COM DISTORÇÃO

TOLEDO

2025

FELIPE ARTHUR POVALUK DA SILVA

IMPLEMENTAÇÃO DE UM TEREMIM DIGITAL COM DISTORÇÃO

Implementation of a digital theremin with distortion

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Tiago Piovesan Vendruscolo

TOLEDO

2025

FELIPE ARTHUR POVALUK DA SILVA

IMPLEMENTAÇÃO DE UM TEREMIM DIGITAL COM DISTORÇÃO

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 03/dezembro/2025

Tiago Piovesan Vendruscolo
Doutorado
Universidade Tecnológica Federal do Paraná

Alvaro Ricieri Castro e Souza
Doutorado
Universidade Tecnológica Federal do Paraná

Maurício Zardo Oliveira
Doutorado
Universidade Tecnológica Federal do Paraná

TOLEDO

2025

AGRADECIMENTOS

Primeiramente aos meus pais Ana e Jair que sempre estiveram comigo, me incentivando e felizes pela minha graduação.

Ao meu irmão Jean que me acompanhou no crescimento acadêmico enquanto eu acompanhei seu crescimento.

Aos meus amigos e colegas que me acompanharam nessa jornada e a tornaram mais animada e calorosa.

Às minhas duas filhas, Beatrice e Maria, as quais se juntaram a mim no meio dessa jornada, fornecendo um significado imensurável.

Ao meu professor orientador Tiago pelo acompanhamento e suporte com este trabalho.

E por fim e não menos importante, a Deus e ao universo como um todo, que me proporcionaram e permitiram concluir essa formação.

RESUMO

Este trabalho consiste na implementação de um teremim, um instrumento musical, normalmente implementado na forma analógica, composto por dois sensores que irão captar a distância da mão do usuário para alterar a frequência e amplitude do dispositivo, de acordo com a finalidade de cada. O teremim que este trabalho propõe será desenvolvido em uma versão digital, utilizando o microcontrolador ESP32 em conjunto com a linguagem C++. Os dados processados pelo microcontrolador serão enviados para um módulo DAC responsável pela conversão em um sinal audível e amplificado para ser reproduzido em um alto-falante. Esse teremim tem a opção de incluir distorções no sinal, análogas às de guitarras elétricas, como também um seletor de frequência máxima de operação. Como resultado, o teremim construído é de fácil manipulação, podendo ser operado por qualquer usuário, e também de fácil manutenção, pois seus componentes podem ser facilmente trocados em comparação à uma versão analógica.

Palavras-chave: teremim digital; microcontrolador; esp32.

ABSTRACT

This work consists of the implementation of a theremin, a musical instrument normally built in an analog form, composed of two sensors that detect the distance of the user's hand in order to alter the device's frequency and amplitude, according to each sensor's function. The theremin proposed in this work is developed as a digital version using the ESP32 microcontroller programmed in C++. The data processed by the microcontroller are sent to a DAC module, which converts them into an audible signal, then amplified and played through a speaker. This theremin also includes the option to apply signal distortions, similar to those used in electric guitars, as well as a selector for the maximum operating frequency. As a result, the constructed theremin is easy to operate by any user and offers better maintenance, since its components can be easily replaced compared to an analog version.

Keywords: digital theremin; microcontroller; esp32.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Siglas

DSB	Banda Lateral Dupla, do inglês *Double Side Band*
SSB	Banda Lateral Única, do inglês *Single Side Band*
BB	Banda-Base, do inglês *Baseband*
AM	Modulação de Amplitude, do inglês *Amplitude Modulation*
ADC	Conversor Analógico-Digital, do inglês *Analog-to-Digital Conversor*
DAC	Conversor Digital-Analógico, do inglês *Digital-to-Analog Conversor*
AMPOP	Amplificadores Operacionais
I2S	*Inter-Integrated Circuit Sound*
MIDI	Interface de Instrumentos Musicais Digitais, do inglês *Musical Instrument Digital Interface*
CI	Circuito Integrado
WS	Seletor de Letra, do inglês *Word Select*
SCK	*Clock* Serial
SD	Dados Seriais, do inglês *Serial Data*
FS	*Frame Sync*
ROM	Memória de Somente Leitura, do inglês *Read Only Memory*
SRAM	Memória Estática de Acesso Randômico, do inglês *Static Random-Access Memory*
PSRAM	Pseudo-Memória Estática de Acesso Randômico, do inglês *Pseudo Static Random-Access Memory*
RTC	Relógio de Tempo Real, do inglês *Real Time Clock*
GPIO	Entradas e Saídas de Propósito Geral, do inglês *General Purpose Input/Output*
COM	Comunicação
IDE	Ambiente de Desenvolvimento Integrado, do inglês *Integrated Development Environment*

PCI	Placa de Circuito Impresso	

Letras Latinas

F	Farad	$[\mathrm{C/V}]$
V	Volt	
p	Pico	$[10^{-12}]$
k	Quilo	$[10^{3}]$
M	Mega	$[10^{6}]$
Hz	Hertz	
B	Bytes	8 bits

Letras Gregas

μ	Micro	$[10^{-6}]$
π	Pi (constante circular)	$[\mathrm{rad}]$
Ω	Ômega (Resistência)	
ϵ_0	Épsilon zero (Permissividade elétrica no vácuo)	$[8{,}85\text{x}10^{-12}\mathrm{F/m}]$

SUMÁRIO

1 INTRODUÇÃO

Ao se tornar líder do novo laboratório experimental do Ioffe Physical Technical Institute em 1920, Lev Sergeevich Termen, começou seus estudos sobre sinais de alta frequência. Em 1927, ao se mudar para os Estados Unidos, mudou seu nome para Leon Theremin, levando consigo seu novo instrumento musical desenvolvido.

Composto por duas antenas acopladas a um circuito analógico, movimentar as mãos alterava a capacitância presente no circuito dessa antena, sendo que em uma antena controlava a frequência do sinal e na outra a amplitude. Esse instrumento foi denominado originalmente *aetherphone*, depois *thereminvox* e então *theremin*, ou então em português, teremim (NIKITIN, 2012). A Figura 1 exibe uma imagem de Leon Theremin operando sua invenção.

Figura 1 – Leon Theremin operando o theremin

Fonte: Adaptado de Nikitin (2012).

Com a criação de transistores em 1947, a criação de circuitos que até então era somente analógica passou a possibilitar a criação de circuitos lógicos, bem como microprocessadores e microcontroladores a partir da década de 70 (HEXSEL, 2006). Os microcontroladores permitem que seu funcionamento se dê a partir de um código escrito em uma linguagem específica para o mesmo, juntamente com a alimentação elétrica, de acordo com a especificação de cada microcontrolador.

Atualmente, a maioria dos dispositivos vigentes são digitais: computadores, celulares, relógios, entre outros, pois construir e prestar manutenção em um dispositivo digital é mais barato (do ponto de vista monetário) e rápido (do ponto de vista prático). Então, nesse trabalho será utilizado a fundamentação por trás de um teremim para sua construção, porém, sua implementação será realizada de forma digital, por meio de um microcontrolador, tendo ainda um adicional: permitir que o usuário possa aplicar distorção (semelhante ao efeito que pedais de guitarra fazem) enquanto manuseia o instrumento, sendo isto uma nova funcionalidade acoplada ao dispositivo.

1.1 Considerações iniciais

O teremim possui originalmente duas antenas, uma para controle de frequência e outra para controle de amplitude, e comercialmente existem em formatos tanto digital, quanto analógico, porém esses produtos exploram apenas essa versão do instrumento.

A proposta nesse trabalho é, além de construir um teremim com processamento e sintetização digital do sinal por meio de um microcontrolador, criar também uma funcionalidade que permita realizar distorção do sinal, fazendo com que o mesmo reproduza uma onda sonora diferente da anterior, de acordo com a distorção disponível selecionada pelo usuário.

Por se tratar de um instrumento com processamento digital do sinal, sua implementação é menos trabalhosa em comparação ao analógico, dado que um ajuste do código-fonte do microcontrolador é mais rápido do que alterar toda a estrutura do circuito analógico, o que inclusive acarretaria em novos cálculos dado o rearranjo do circuito. Como microcontrolador, utilizaremos o ESP32, capaz de operar um código da linguagem C++, o qual gerenciará a geração e processamento dos sinais envolvidos no teremim.

1.2 Objetivos

1.2.1 Objetivo geral

Implementar um teremim com 2 sensores, sendo um para controle de frequência e outro para amplitude e um botão para aplicar distorção do sinal de saída, utilizando um microcontrolador capaz de sintetizar o sinal e então reproduzi-lo em um alto-falante.

1.2.2 Objetivos específicos

São objetivos específicos:

a. Criar um código na linguagem C++ para operar no microcontrolador ESP32, capaz de gerenciar os sinais de cada sensor e sintetizar o sinal de saída para o alto-falante.

b. Definir as funções arbitrárias de distorção pré-definidas que possam ser selecionadas durante a operação do teremim.

c. Definir e confeccionar uma PCI do circuito final e uma caixa para comportar a mesma.

1.3 Justificativa

O teremim é um excelente exemplo da aplicabilidade de circuitos analógicos. Ele utiliza desde arranjos simples, como resistores e capacitores para controle de tensão, até componentes mais complexos.

O instrumento possui osciladores para gerar os sinais em cada antena e emprega amplificadores operacionais (AMPOPs) para fazer a diferenciação desses sinais. Em seguida, filtros passa-baixa são aplicados para melhorar a qualidade do áudio.

Cada um desses arranjos presentes no instrumento envolve uma carga de cálculo significativa, que se soma no projeto total (LIU, 2011). Apesar desse trabalho ser em grande parte digital, parte de conceitos analógicos ainda serão aplicados.

O uso do ESP32 exigirá a programação em linguagem C++, amplamente difundida e reconhecida, não apenas no meio acadêmico, mas também na indústria. Isso também proporcionará maior contato com microcontroladores, dispositivos amplamente utilizados em ambientes industriais. Será necessário utilizar conversores digitais-analógicos para a comunicação do microcontrolador com o conversor e os sensores.

Por fim, a implementação de um botão com a funcionalidade de aplicar distorção, além das duas antenas base que já funcionam como sensores, seria uma inovação ainda não realizada. Este terceiro sensor, manipulado pela mão do usuário ou acoplado de alguma forma a ela, permitiria aplicar uma distorção no sinal, agregando uma nova funcionalidade ao instrumento.

1.4 Estrutura do trabalho

O Capítulo 2 aborda a fundamentação teórica do teremim analógico, proposta geral do teremim digital e os princípios e protocolos a serem utilizados, suavização exponencial simples, a linguagem C++ e a plataforma Arduino, e os princípios de uma distorção do som.

No Capítulo 3 são explorados os componentes utilizados no projeto, tais como: ESP32, DAC UDA1334, Arduino IDE, sensor óptico VL53L0X, e a construção do mesmo, com o diagrama do teremim digital, estrutura geral do algoritmo, funções de distorção e confecção da PCI e caixa que irá comportá-la.

No Capítulo 4 é realizada a implementação do código em C++ que será utilizada no microcontrolador, feita a confecção da PCI, confecção da caixa em um modelo 3D para comportar a PCI, uma análise das frequências obtidas em relação a determinadas distâncias lidas, apresentação do modelo final e discussões sobre eventuais problemas identificados.

Finalmente o capítulo 5 discorre sobre o resultado obtido na implementação como um todo e seus principais problemas, bem como a estipulação de um trabalho futuro, implementando novas funcionalidades.

2 REFERENCIAL TEÓRICO

Um teremim precisa de um determinado circuito analógico para realizar a percepção de oscilação do movimento do usuário, para que isso provoque a alteração da frequência ou da amplitude, dependendo da antena, para que então seja reproduzido pelo alto-falante.

Porém, nesse trabalho foi implementada uma versão digital do mesmo, substituindo as antenas por sensores ópticos de distância como canal de entrada dos sinais, com o processamento sendo feito pelo microcontrolador ESP32.

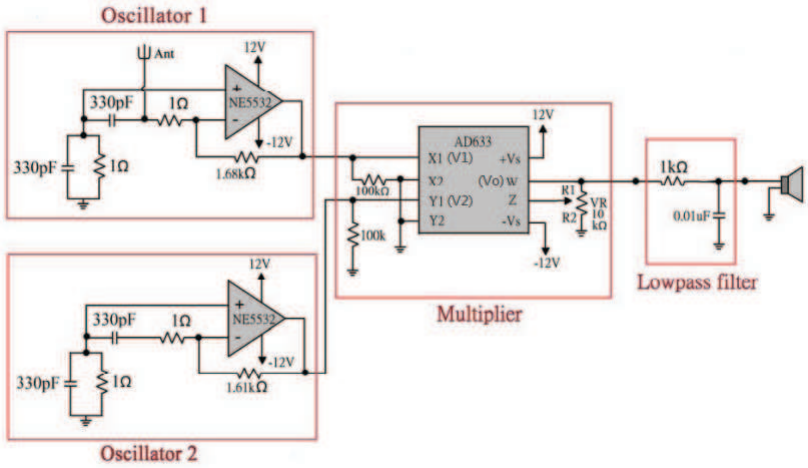
Além do teremim digital, foi feita a implementação de um botão no circuito do mesmo, para que uma distorção selecionada seja aplicada quando o mesmo for pressionado.

2.1 Teremim Analógico

Conforme Liu (2011), o design tradicional do teremim analógico utiliza dois osciladores, com um oscilador ressoando em uma frequência fixada, e o outro oscilador ressoando em frequências ajustadas pela capacitância induzida entre a antena e a mão, sendo ambos sinais senoidais. A necessidade de um oscilador em uma frequência estática se dá pelo fato que a interferência causada pela mão do usuário é baixa, sendo de uma capacitância de aproximadamente 2 a 6 pF (LIU, 2011).

O circuito de um teremim na Figura 2 exemplifica o mesmo considerando apenas uma antena, relacionada à frequência.

Figura 2 – Circuito do teremim para uma antena



Fonte: Adaptado de Liu (2011).

Para fins de relação de entrada/saída, temos que a relação é dada por:

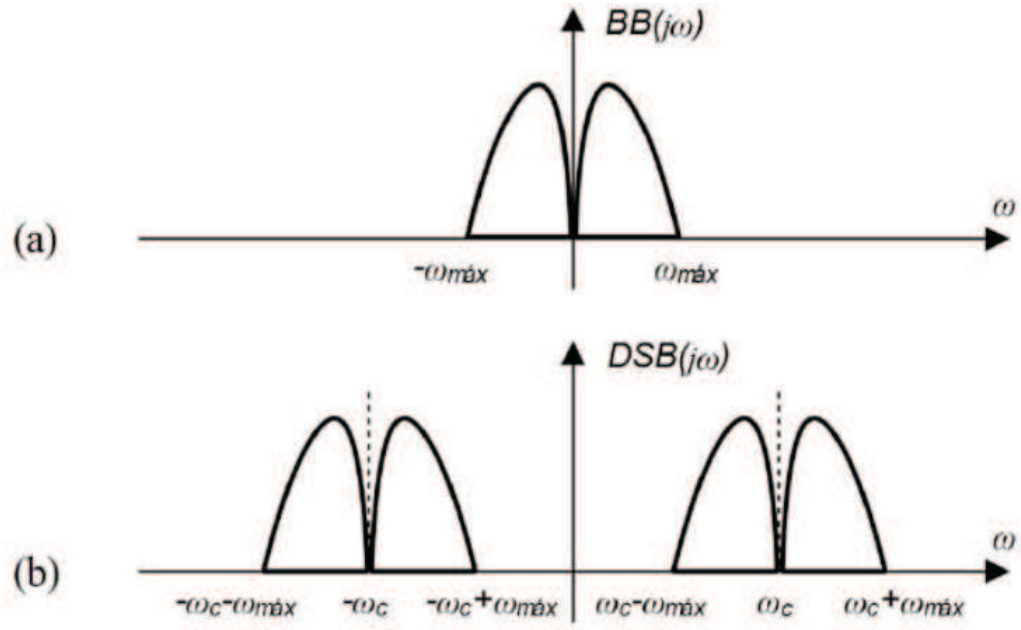
$$Vo = (V1 \times V2) \times \frac{1}{10(1 + R2/R1)}, \tag{1}$$

onde Vo é a tensão de saída no pino W do CI AD633 e V1 e V2 são os sinais de entrada aplicados nos pinos X1 e Y1. Para criar uma sensibilidade significativa, o oscilador 2 se mantém em 600kHz, o que permite fazer com que o oscilador 1 altere idealmente sua frequência de 20 Hz a 20kHz, abaixo de 600kHz o que corresponde a distância entre "longe" e "perto".

Essa faixa de 20Hz a 20kHz compreende o espectro de audição humana, que é de 16Hz a 16kHz, que compreende dos tons graves (frequências baixas) até os tons agudos (frequências altas), mas algumas literaturas tratam como sendo a faixa de 20Hz a 20kHz (BRAGA, 2015). Inclusive as notas musicais são compreendidas em determinadas frequências, sendo que em 1936 fixou-se a frequência do "lá" como sendo 440Hz (RIGHI, 2019), e então a sequência das notas "dó", "ré", "mi", "fá", "sol", "lá" e "si" com as respectivas frequências de 262, 269, 330, 349, 392, 440 e 494Hz, respectivamente (PEREIRA, 2012).

Ainda no circuito da Figura 2, o CI AD633 é utilizado como um modulador (ou multiplicador) DSB (*Double Side Band*, banda lateral dupla). Uma modulação *Double Side Band* corresponde ao espelhamento causado no domínio da frequência em relação ao eixo imaginário jw decorrente da transformada de Hilbert do sinal em questão, tendo que o sinal de banda-base (BB, *Baseband*) foi deslocado por conta da modulação de amplitude (AM). A Figura 3 exibe um exemplo de comparação do espectro de frequência de: (a) um sinal banda-base; e (b) a modulação DSB correspondente.

Figura 3 – Comparação entre SSB e DSB



Fonte: Adaptado de Lacerda (2017).

Portanto, o sinal proveniente do oscilador 2 (de frequência fixa) da Figura 1 está servindo de banda-base para o sinal do oscilador 1 (o qual sofre interferência da antena).

Por fim, mesmo o circuito com esse aparato, foi avaliado que a escala das notas é não-linear em relação à distância da mão com a antena, o que torna difícil de se ter um bom ritmo, além do timbre não ser tão rico, se assemelhando a um violino com baixa qualidade (LIU, 2011).

2.2 Teremim Digital

A utilização de muitos dispositivos analógicos, além de aumentar a complexidade do dispositivo, faz com que a manutenção do mesmo seja dificultada. Com isso, surge a oportunidade de utilizar dispositivos digitais para facilitar tal manuseio. Nesse ponto é utilizado um microcontrolador para realizar o processamento.

Um microcontrolador é um CI que possui uma unidade de processamento, uma memória própria para armazenar programas escritos, conversores analógico-digital e digital-analógico e portas de entrada e saída genéricas chamadas GPIO (ARAUJO, 2019). Comumente em microcontroladores, são utilizados códigos nas linguagens C/C++, os quais são criados em um computador comum e então compilados diretamente no microcontrolador. Na Seção 3.1 é descrito sobre esse processo de compilação e na Seção 2.4 é aprofundada a motivação de utilizar linguagem C++.

Para reproduzir um sinal analógico de saída pode se utilizar um conversor digital-analógico, DAC. Um conversor DAC consiste em converter o valor quantizado lido em algum dos níveis possíveis de tensão, sendo que seu alcance será até $2^N - 1$, onde N corresponde a quantidade de bits da quantização.

Para comparação, dada uma tensão que pode variar de 0V a 5V, temos que cada nível do DAC é dado por:

$$V_{passo} = \frac{V_{max}}{2^N - 1}, \tag{2}$$

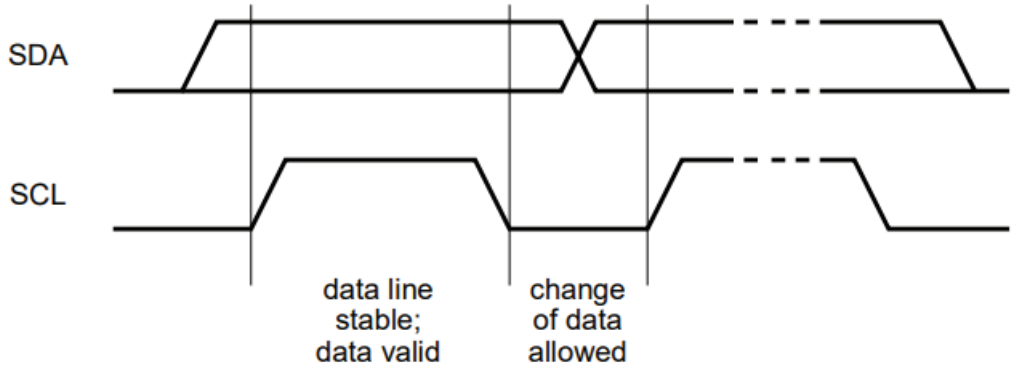
com isso, para resolução de 8 bits, temos que $V_{passo} \approx 19{,}60mV$, porém para 16 bits $V_{passo} \approx 0{,}07mV$, o que representa uma resolução muito melhor, pois há mais níveis para representar as possíveis tensões de saída.

O ESP32, microcontrolador a ser utilizado, possui um DAC de 8 bits de quantização, enquanto o DAC UDA1334 (SEMICONDUCTORS, 2000) possui suporte para 16 bits, mais adequado para comunicações de áudio.

Para realizar a comunicação do microcontrolador com os sensores ópticos de distância foi utilizado *Inter-Integrated Circuit*, I2C, um protocolo de comunicação entre circuitos que possui 2 canais de comunicação: *SDA* (*Serial Data*) como canal de dados e *SCL* (*Serial Clock*) para envio de *clock* entre os dispositivos.

Conforme a Figura 4, a transferência de cada *bit* é feita enquanto o canal *SCL* está em nível baixo, e quando estiver em nível alto o *SDA* deve ter seu nível estabilizado (em alto ou baixo).

Figura 4 – Diagrama do tempo para o protocolo I2C

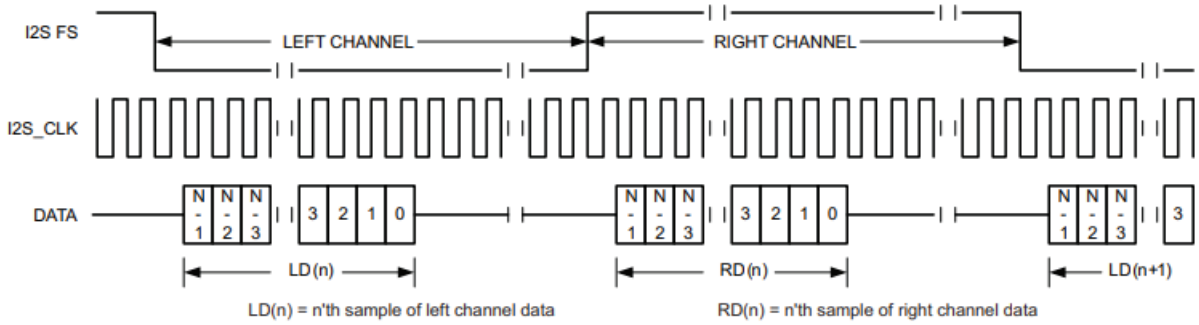


Fonte: Adaptado de Semiconductors (2021).

Em caso de utilização do mesmo barramento para vários dispositivos, há um processo para início e fim de transferência de informação, porém nesse trabalho cada sensor utiliza um barramento separado, portanto não necessitou desse nível de controle.

Para realizar a comunicação do microcontrolador com o DAC, foi utilizado o *Inter-IC Sound*, I2S, um protocolo de comunicação de som entre circuitos integrados (semelhante ao I2C, mas com um canal a mais), o qual se baseia em 3 canais de comunicação, sendo eles *WS* (*Word Select*) indica o canal (esquerdo ou direito), *SCK* (*Serial Clock*) é utilizado para envio de *clock* entre os dispositivos mestre-escravo e o *SD* (*Serial Data*) o canal de dados do áudio. Na Figura 5 há um exemplo da comunicação do protocolo.

Figura 5 – Diagrama do tempo para o protocolo I2S



Fonte: Adaptado de Texas Instruments (2010).

Como pode ser visto na Figura 5, o canal *WS* identificado como FS na figura (*Frame Sync*) quando está em nível baixo se trata da seleção do canal esquerdo, enquanto, em nível alto, do direito. O canal *SCK* identificado como I2S_CLK possui a característica de ativação por borda de descida pois o canal SD, identificado como DATA, está sincronizado com o mesmo.

2.3 Suavização Exponencial Simples

Existem vários métodos de suavização de um valor de entrada determinado por uma $f[n]$ qualquer, podendo ser com nenhuma tendência, tendência aditiva, aditiva amortecida, multiplicativa e multiplicativa amortecida (COELHO, 2008).

Uma suavização com nenhuma tendência pode ser intitulada como Suavização Exponencial Simples, *SES*, e ela pressupõe que os dados oscilam em uma média constante, sem uma tendência clara de crescimento ou decaimento que poderiam ser previsíveis por algum modelo (LUSTOSA, 2008).

Em tempo discreto uma SES pode ser definida conforme a equação a diferenças:

$$s[n+1] = s[n] + \alpha(f[n] - s[n]), \tag{3}$$

onde $s[n]$ é a saída atual do sistema, $s[n+1]$ é a próxima saída do sistema, α é o coeficiente de suavização que varia entre 0 e 1 e $f[n]$ o valor de entrada e também a referência.

Quanto maior o α mais rápido será a resposta do sistema para atingir o valor de referência $f[n]$, consequentemente quanto menor ele for mais suavizado e lento o sistema será (PAGANELLI, 2014).

Tomando $s[0] = 0$, $f[n] = 100$ e $\alpha = 0.8$, o Quadro 1 exemplifica as 4 primeiras saídas em um sistema discreto com SES aplicado:

Quadro 1 – Exemplos de saídas em um sistema discreto com SES aplicado

n	$s[n]$	$s[n+1] = s[n] + 0.8(f[n] + s[n])$
0	0	$80 = 0 + 0.8(100 - 0)$
1	80	$96 = 80 + 0.8(100 - 80)$
2	96	$99.2 = 96 + 0.8(100 - 96)$
3	99.2	$99.84 = 99.2 + 0.8(100 - 99.2)$

Fonte: Autoria própria (2025).

Veja que o sistema responde rapidamente a entrada, se aproximando em poucas iterações até a referência $f[n]$ desejada. Será utilizada uma SES em relação ao sensor de frequência que utiliza esse mesmo valor de α, mencionada na Seção 3.2.

2.4 A Linguagem C++ e a plataforma Arduíno

Retornando para a linguagem C++ citada anteriormente, ela é uma sucessora da linguagem C, que permite a criação de classes, as quais definem os objetos a serem criados com tais classes. Ou então como Morandini (1996) descreveu:

> C++ é uma linguagem de programação Orientada a Objetos, de propósito geral com o intuito de tornar a tarefa de programação mais agradável. C++ é um superconjunto da linguagem C e fornece facilidades para definição de novos tipo, que englobam objetos que contenham mesmo relacionamento de tipos.
>
> O conceito chave da linguagem C++ é a **classe**.

Em uma classe, temos os atributos e métodos que compõem a mesma, as quais poderão ser utilizadas pelos objetos os quais forem instanciados dessa classe.

O ESP32 é um microcontrolador que permite a utilização da plataforma Arduíno, a qual possui várias bibliotecas com classes já definidas para diversos contextos e aplicações diferentes. Por exemplo, a biblioteca "AudioTools.h" (SCHATZMANN, 2025) facilita a comunicação com dispositivos de áudio, permitindo fazer o envio de informações pelo protocolo I2S sem que seja criada uma estrutura totalmente nova, facilitando implementações de novos dispositivos.

Outro exemplo é a biblioteca "Adafruit_VL53L0X.h" que permite comunicação com o sensor de distância VL53L0X (HOME, 2025).

2.5 Distorção do Som

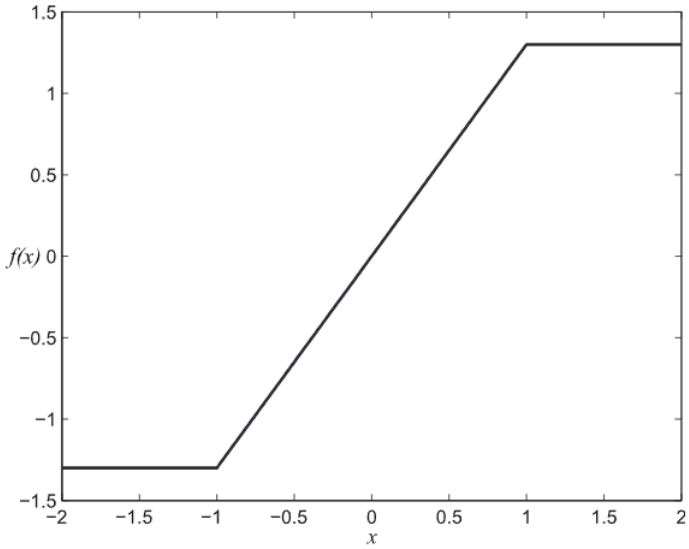
Um som pode ser definido por uma somatória de sinais senoidais. A distorção de um som se trata da alteração de sua onda sonora, e uma dessas formas é o *Waveshaping*, o qual altera o próprio formato da onda senoidal (DODGE, 1985), atuando não na frequência do mesmo, mas diretamente em sua amplitude.

Já Oliveira (2013) explora um *waveshaping* do tipo *hardclipping*, que realiza um corte de onda após determinado limiar, representado pela função 4 e patenteado por Doidic (1998), onde $K = 1{,}3$:

$$f(x) = \begin{cases} Kx & \text{se } |x| < 1 \\ Ksign(x) & \text{se } |x| \geq 1 \end{cases}, \tag{4}$$

e a função $sign(x)$ pode ser vista na Figura 6:

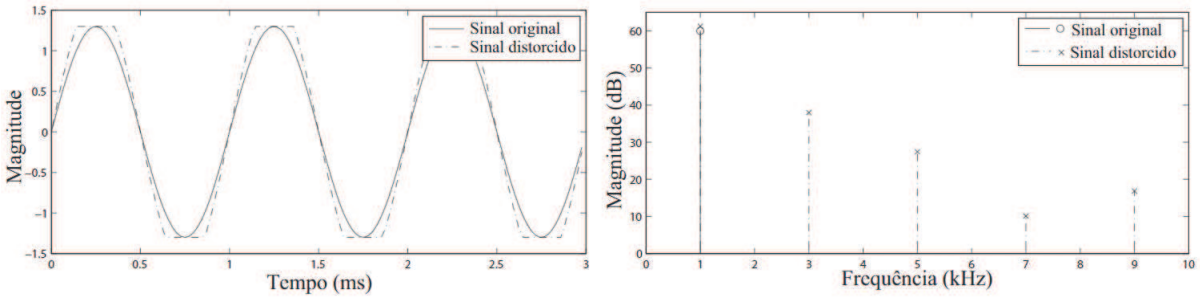
Figura 6 – Função sign(x)



Fonte: Adaptado de Oliveira (2013).

Aplicando um sinal senoidal de 1 kHz nessa distorção, chegamos nos gráficos da Figura 7, onde temos o sinal senoidal filtrado no domínio do tempo (esquerda) e da frequência (direita):

Figura 7 – Resultado da aplicação de 1 KHz



Fonte: Adaptado de Oliveira (2013).

Distorções como estas são as presentes nos pedais para guitarras elétricas, por exemplo, mas é comum que as funções de distorção dos pedais sejam complexas, e comumente patenteadas.

3 MATERIAIS E MÉTODOS

3.1 Materiais

ESP32-WROOM-32, o qual será referenciado apenas como ESP32, é o microcontrolador utilizado para o processamento dos sinais. Nele será compilado um código escrito na linguagem C++ que tem o objetivo de processar os sinais recebidos dos sensores de distância (que substituirão as antenas, os originais sensores de um teremim analógico), a fim de manipular a amplitude e a frequência, e então gerar um sinal de saída com base nos sinais de entrada.

Portanto, serão dois sinais digitais de entrada a serem recebidos pelo ESP32 e um sinal também digital de saída que o mesmo fornecerá. Esse sinal será uma senoide enviada a um DAC (Conversor Digital-Analógico, *Digital-to-Analog Converter*), que irá enviar o sinal agora digital para um alto-falante, então assim reproduzindo o som gerado.

O ESP32 possui uma baixa resolução de conversão, de apenas 8 bits (ESPRESSIF, 2024) o que resulta em apenas 256 níveis de informação. Essa faixa de 8 bits de quantização não é a mais adequada para lidar com áudio, que é comumente utilizado ao menos 16 bits (geralmente em CD-ROM) e 24 a 32 bits para gravações de áudio profissional (BODANESE, 2008). Posteriormente será descrito sobre o DAC UDA1334, o qual será utilizado para tal parte do processo.

Também serão acoplados ao dispositivo um botão para alterar a distorção sempre que o mesmo for pressionado, e um potenciômetro para permitir alterar a frequência máxima de operação do sensor de frequência, alterando o mapeamento da faixa de frequência reproduzida em relação a distância da mão do operador do teremim.

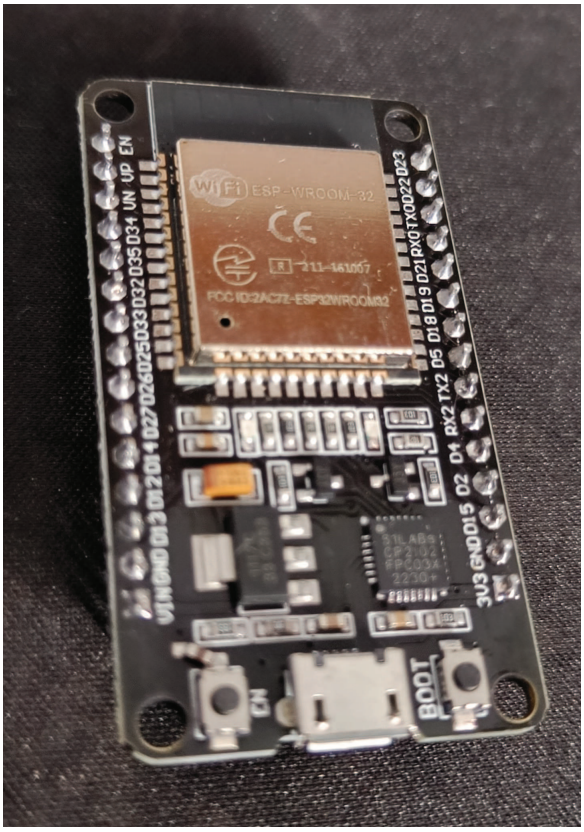
Fabricado pela Espressif Ltd., o modelo ESP32 foi o selecionado para a utilização no trabalho pela sua facilidade de encontro no mercado e por possuir diversas bibliotecas disponíveis para manipulação de dados com os GPIOs do mesmo. A Figura 8 exibe a imagem do ESP32 utilizado.

Conforme o *datasheet* do ESP32 (ESPRESSIF, 2024), as características mais relevantes do dispositivo para este trabalho seguem abaixo.

Referentes a CPU (Unidade Central de Processamento) e a memória:

- Xtensa® processador(es) LX6 de núcleo duplo;
- 2 núcleos com 240 MHz: 994.26 CoreMark; 4.14 CoreMark/MHz;
- 1310 kB ROM;
- 327 kB SRAM;
- 16 kB SRAM no RTC;

Figura 8 – Microcontrolador ESP32 utilizado



Fonte: Autoria própria (2025).

Clocks e *Timers*:

- Oscilador interno de 8MHz com calibração;
- Oscilador resistor-capacitor interno com calibração;
- Oscilador externo de 2 MHz a 60 MHz (40 MHz somente para o funcionamento do Wi-fi/Bluetooth);
- Oscilador de cristal de 32 kHz para o RTC com calibração;
- Grupos de dois *timers*, incluindo 2 de 64-bit e 1 para *watchdog* (cão-de-guarda) para cada grupo;
- Um *timer* RTC;
- RTC *watchdog* (cão-de-guarda).

Interfaces periféricas avançadas:

- 34 GPIOs programáveis:
  – 5 GPIOs para configuração do sistema;
  – 6 GPIOs de somente entrada;

– 6 GPIOs necessários para flash/PSRAM no encapsulamento (ESP32-D0WDR2-V3, ESP32-U4WDH).

- ADC SAR de 12 bits com até 18 canais;
- Dois 8-bit DAC;
- Duas interfaces I2S.

Na figura Figura 9 é exibido a pinagem o *ESP32 Dev Board* utilizado:

Figura 9 – Pinagem do *ESP32 Dev Board*

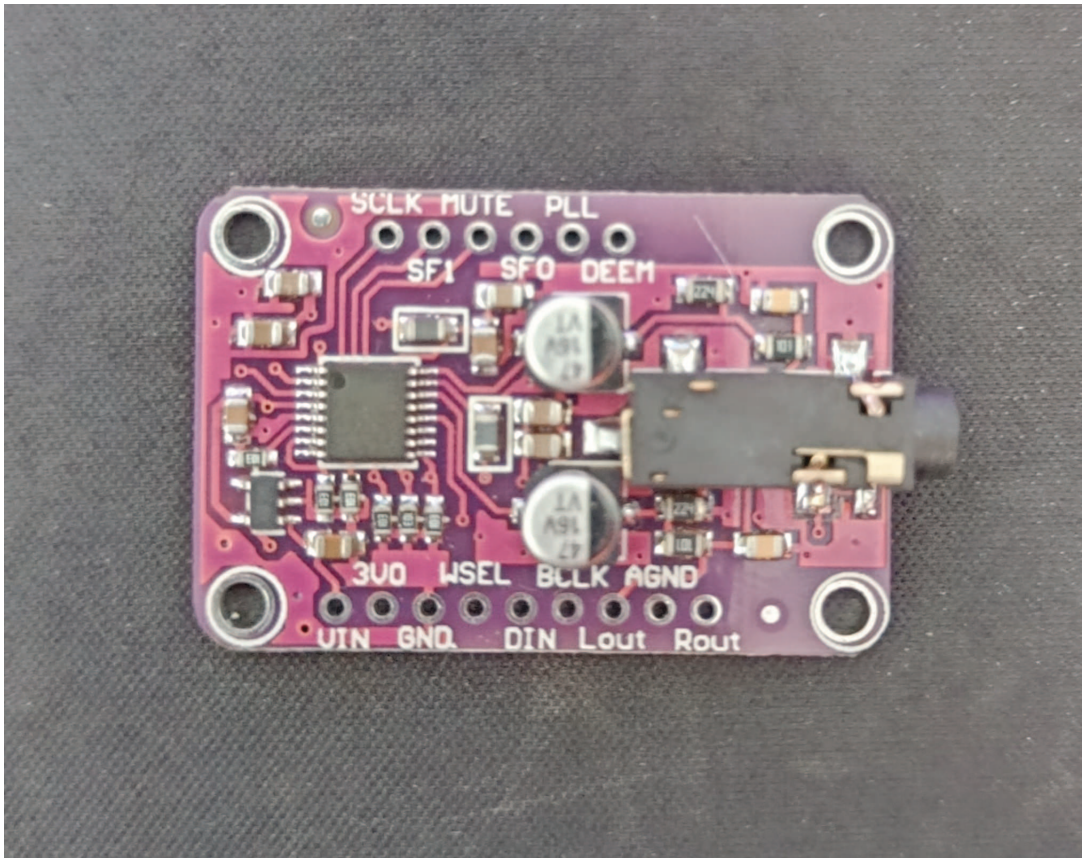
Fonte: Projects (2020).

Utilizando a IDE (*Integrated Development Environment*, Ambiente de Desenvolvimento Integrado) do Arduino (ARDUINO, 2025), é possível escrever códigos na linguagem C++ e com o auxílio de um cabo USB, é possível realizar o *upload* do código por meio da IDE do Arduino, após finalizar o código e definir corretamente as configurações (nesse caso a placa "ESP32 Dev Module").

Para realizar a saída de áudio analógico será utilizado o DAC UDA1334, que pode ser visualizado na Figura 10, e possui uma resolução de 16, 24 e 32 bits, *chip* já recomendado para reprodução de áudio, o qual utiliza o protocolo I2S para receber o sinal de entrada e realiza a saída de áudio em um soquete fêmea do tipo P2.

Para que o mesmo funcione corretamente, é necessário conectar 5 pinos com a ESP32:

Figura 10 – DAC UDA1334

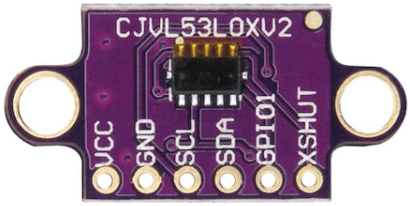


Fonte: Autoria própria (2025).

- VIN: "*Voltage Input*", se trata da alimentação do dispositivo, de 3.3V a 5V;
- GND: "*Ground*" (terra, ou aterramento) deve ser conectado ao GND do ESP32;
- WSEL: "*Word Select*", seleciona os canais direito ou esquerdo para saída de áudio;
- DIN: "*Data Input*", a entrada de dados são os dados do áudio em si, passados conforme o protocolo I2S;
- "BCLK": "*Bit Clock*", o canal de entrada de *clock*, no caso que o UDA 1334 é o dispositivo escravo na relação mestre-escravo do protocolo I2S.

O VL53L0X é um sensor de distância que utiliza o protocolo I2C para comunicação, com precisão milimétrica de alcance em até 2 metros, baseado no tempo de propagação da luz de um LED. A Figura 11 exibe uma imagem do mesmo:

Figura 11 – Sensor VL53L0X



Fonte: (HOME, 2025).

O pino Vcc exige uma alimentação de 3.3V a 5V para que o dispositivo opere de forma adequada e o GND (*Ground*) deve estar conectado ao aterramento comum ao circuito.

O pino SCL diz respeito ao *clock* de operação que será fornecido pelo dispositivo que estiver o operando (nesse caso, o ESP32), SDA são propriamente os dados lidos pelo dispositivo e o XSHUT é um pino que indica se o dispositivo está ligado ou não quando o mesmo está em estado alto (acima de 3.3V).

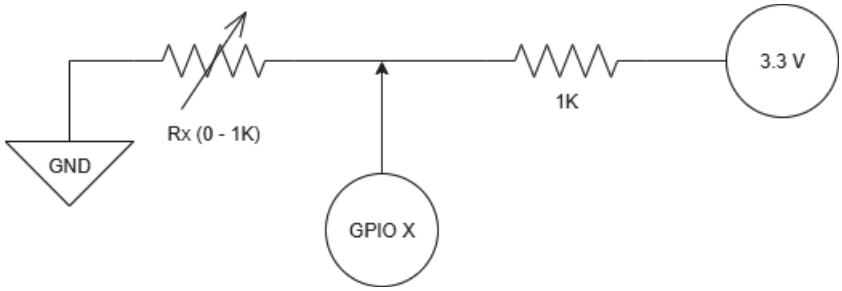
Também foram utilizados um potenciômetro de 0Ω a $1K\Omega$, um resistor de $1K\Omega$ para fazer divisão de tensão com o potenciômetro, outro resistor de $10K\Omega$ e um botão simples.

Para a confecção da PCI (Placa de Circuito Impresso) será utilizado uma placa de fenolite simples, folha com circuito impresso, percloreto de ferro para corrosão da placa, estanho para soldagem dos componentes e por fim material para realizar a impressão 3D da caixa que irá comportar a PCI.

3.2 Métodos

O ESP32 possui pinos de entrada e saída de propósito geral, GPIO (*General Purpose Input/Output*), e um desses pinos irá ler uma tensão nodal definida pelo seletor de frequência máxima, que está arranjado em um ramo com um divisor de tensão, exemplificado na Figura 12.

Figura 12 – Exemplo de divisor de tensão



Fonte: Autoria própria (2025).

A partir da análise nodal (SADIKU, 2013), tem se que o V_{ref} presente no ponto de leitura do pino GPIO X pode ser descrito como:

$$\frac{3{,}3 - V_{ref}}{1000} = \frac{V_{ref}}{R_x}, \tag{5}$$

o que resulta na equação:

$$\frac{3{,}3}{\frac{1000}{R_x}+1} = V_{ref}, \tag{6}$$

portanto se $R_x = 1000$, temos:

$$\frac{3{,}3}{\frac{1000}{1000}+1} = V_{ref} = 1{,}65V, \tag{7}$$

se $R_x \to 0$, temos:

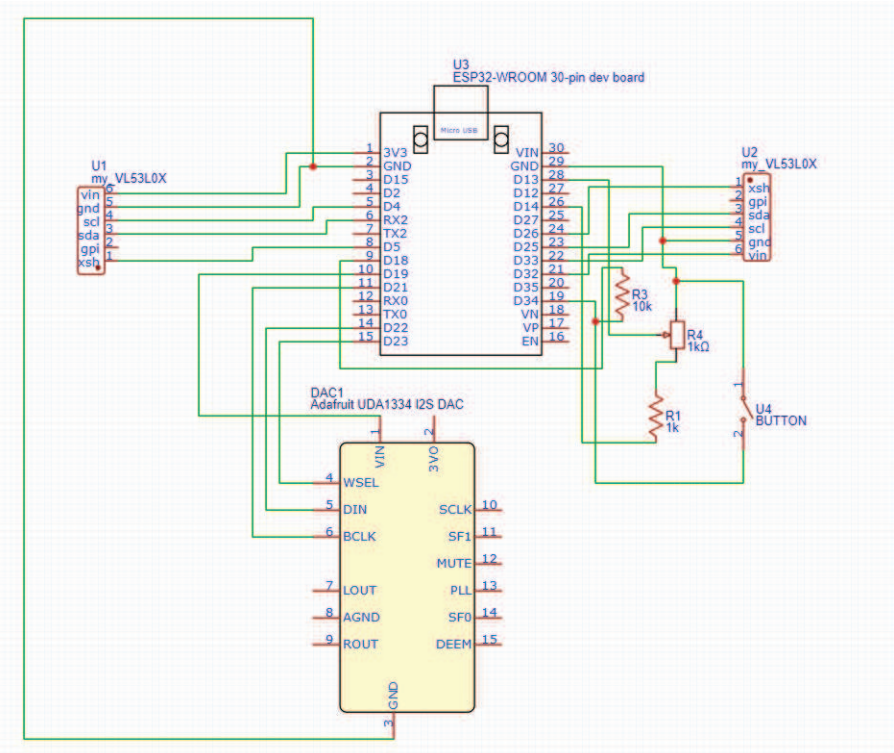
$$\frac{3{,}3}{\infty+1} = V_{ref} = 0V, \tag{8}$$

e por fim se $R_x = 0$ não se considera o R_x na equação (afinal causaria uma indefinição matemática, dada a divisão por zero), resultando simplesmente que o ponto $V_{ref} = 0$ (está aterrado).

Como sensores para manipulação do teremim serão utilizados os sensores VL53L0X. O valor atribuído será inversamente proporcional à distância lida em milímetros. O alcance limite foi fixado em 70 mm, portanto tendendo a 0 mm teremos a leitura máxima, resultando assim na amplitude ou frequência máxima (de acordo com a finalidade de cada sensor), enquanto em 70 mm teremos o menor valor lido e, além dessa distância, considera-se como sem operação.

O dispositivo possui o diagrama de circuito elétrico conforme a Figura 13, e a divisão de tensão realizará justamente esse mapeamento, sendo que quando o potenciômetro R4 estiver em 0Ω, sua tensão lida é de $0V$, portanto está com frequência máxima de 400Hz de operação do teremim (a onda senoidal reproduzida terá no máximo essa frequência).

Figura 13 – Diagrama do Circuito do Teremim Digital



Fonte: Autoria própria (2025).

Conforme gradualmente o potenciômetro R4 é alterado, sua tensão nodal dada pelo divisor de tensão (com o resistor R1 de $1k\Omega$) acima descrito irá aumentar, fazendo um aumento linear mapeado das frequências de $400Hz$ a $3000Hz$, respectivamente relacionados às tensões de $0V$ a $1{,}65V$.

Os dois sensores U1 e U2 de distância VL53L0X (mais à esquerda e à direita na Figura 13) estão ligados em cada lado do ESP32 para que cada mão opere segundo a ergonomia adequada (mão esquerda e direita).

Enquanto o U1 recebe alimentação direto do pino padrão de $3.3V$, o U2 recebe do pino D32, o qual foi setado como sendo saída digital, fornecendo uma tensão próxima de $3.3V$, podendo sofrer leves variações mas que não afetam o desempenho do sensor.

Há um botão U4 que é o responsável pela troca de distorção do som do dispositivo, o qual utiliza uma técnica de *debounce* simples, sendo este um processo de validação em que o botão precisa estar pressionado por alguns ciclos do microcontrolador para então considerar o mesmo como pressionado, e nesse trabalho representa um tempo de 0,15s de pressionamento para alterar a distorção. Esse processo é necessário para evitar pressionamentos duplicados e eventuais ruídos.

O botão está ligado ao GND e ao pino D34 que é um nó compartilhado com o resistor R3 de $10k\Omega$. Quando o botão não está pressionado, há um circuito aberto no mesmo e então toda tensão que parte do pino D18 (saída digital de $3.3V$) está também presente no pino D34 (entrada digital). Já quando o mesmo está pressionado, é ligado esse trecho do circuito ao GND, fazendo com que a tensão presente em D34 seja $0V$ (aterrada). O resistor R3 foi necessário porque o pino D34 não possui resistor de *pull-up* interno.

Após o processamento ocorrer pelo código desenvolvido para o ESP32, ele fornecerá um sinal de saída pelo protocolo I2S, pelos pinos D23, D22 e D21, os quais fornecerão as informações de seleção de canal (WSEL), dados de entrada (DIN) e sinal de *clock* (BCLK), respectivamente, que serão recebidos pelo DAC UDA1334. O UDA1334 terá a mesma referência do terra (GND) do ESP32, tais como demais dispositivos, e receberá a tensão de alimentação pelo pino D19. Demais pinos do UDA1334 não serão utilizados nessa implementação. Então, o UDA1334 irá converter o sinal digital em analógico transmitido pelo ESP32 e irá reproduzir o sinal de saída em um alto-falante.

A alimentação de todo o sistema se inicia no ESP32, pelo micro USB (presente no diagrama da Figura 13), e a saída de áudio por um soquete fêmea do UDA1334, o qual não aparece no diagrama mas pode ser visualizado na Figura 10.

O código em C++ que foi utilizado no ESP32 (SILVA, 2025) possui a seguinte estrutura algorítmica:

1. Inclui as bibliotecas "Wire.h", "Adafruit_VL53L0X.h"e "AudioToolsMin.h" necessárias;

2. Define os pinos de entrada e saída, de acordo com suas funções;

3. Declara as variáveis para controle de I2C, I2S, gerador de áudio, e demais variáveis de apoio;

4. Chama o método de "setup" que define 4 *tasks* relacionadas com base nos métodos:

    (a) ativa entrada/saída de tensão nos pinos;

    (b) liga os sensores VL53L0X e comunica pelo protocolo I2C;

    (c) ativa o UDA1334 e a saída de sinal senoidal pelo protocolo I2S;

    (d) define os métodos que envia áudio, lê a frequência e volume (sensores), e outro que define a frequência máxima com base no potenciômetro e altera a distorção pelo pressionamento do botão;

5. O método "vEnviaAudio" que envia áudio apenas reproduz o sinal vigente, sem *delay* e utiliza exclusivamente o núcleo 0, demais métodos compartilham o núcleo 1;

6. O método "vLeFrequencia" lê a frequência obtém a distância lida em milímetros do sensor, faz um mapeamento inverso (0mm é a frequência máxima), entre 0 até o limite máximo de distância e então aplica a suavização exponencial simples (SES);

7. O método "vLeVolume" lê o volume funciona de forma semelhante ao da frequência;

8. O método "vSetFreqMaxAndDist" também faz um mapeamento não inverso da frequência máxima e também define a distorção atual com base nos pressionamentos do botão;

A biblioteca *AudioTools.h* (SCHATZMANN, 2025) foi utilizada nesse trabalho para compor a parte de comunicação I2S, pois a mesma possui vários métodos já prontos para realizar a comunicação com o UDA1334, bem como também para geração de funções senoidais, a qual compõe o dado a ser transmitido. Porém a mesma é muito extensa ocupando muita memória do ESP32, com isso foi necessário criar uma versão reduzida da mesma, "AudioToolsMin.h" contendo apenas os métodos necessários. Também o arquivo "SoundGenerator.h" foi alterado para conter as funções de distorção utilizadas no teremim digital.

Foram feitas tentativas com comunicação via Bluetooth utilizando o aplicativo de celular *Arduino Bluetooth Control* (GIRISTUDIO, 2025) com o ESP32 para aplicar a distorção, mas a comunicação causava interferência no funcionamento do mesmo, impactando consideravelmente no áudio, resultando em falhas e travamento de áudio.

Devido aos problemas com Bluetooth, foi utilizado um botão para alterar a distorção do dispositivo. Sempre que pressionado, o mesmo alternava da sua onda senoidal pura padrão para uma das 10 funções de distorção criadas arbitrariamente, sem ter tido como base alguma referência direta, resultando em um total de 11 sinais possíveis para serem reproduzidos.

Essas funções são descritas com base no tempo (t), amplitude lida (A) e frequência lida (f), possuem os formatos de onda conforme o Quadro 2, sendo $\sin$ a função seno e $x = f(t) = \sin(2\pi t)$.

Quadro 2 – Formatos de onda dos sinais de distorção definidos arbitrariamente

Número da função	**função** $f(t,A,f)$
1	$f(t,A,f) = A\sin(2\pi ft)$
2	$f(t,A,f) = \begin{cases} 0.7A, & \text{se } x > 0.7, \\ -0.7A, & \text{se } x < -0.7. \\ A\sin(2\pi ft), & \text{c.c.} \end{cases}$
3	$f(t,A,f) = A\sin(2\pi ft)\sin(5 \cdot 2\pi ft)$
4	$f(t,A,f) = A\frac{\sin(2\pi ft)\sin(5\cdot 2\pi ft))}{2}$
5	$f(t,A,f) = A\sin(2\pi ft + \sin(2\pi ft))$
6	$f(t,A,f) = \begin{cases} A\sin(3 \cdot 2\pi ft), & \text{se } x > 0.2, \\ A\frac{\sin(4\cdot 2\pi ft)}{2}, & \text{c. c.} \end{cases}$
7	$f(t,A,f) = \begin{cases} A\sin(3 \cdot 2\pi ft), & \text{se } x > 0.2, \\ A\sin(\frac{\cdot 2\pi ft}{4}), & \text{c. c.} \end{cases}$
8	$f(t,A,f) = \begin{cases} A\sin(\frac{\cdot 2\pi ft)}{3}, & \text{se } x > 0.2, \\ A\sin(\frac{\cdot 2\pi ft}{4}), & \text{c. c.} \end{cases}$
9	$f(t,A,f) = A\sin(2\pi ft)\sin(\frac{20}{3}2\pi ft)\sin(42 \cdot 2\pi ft)$
10	$f(t,A,f) = A\frac{\sin(2\pi ft)+\sin(\frac{20}{3}2\pi ft)+\sin(42\cdot 2\pi ft)}{3}$
11	$f(t,A,f) = A\sin(\frac{20}{3}2\pi ft)\sin(42 \cdot 2\pi ft)$

Fonte: Autoria própria (2025).

4 PROTOTIPAGEM E RESULTADOS

Neste capítulo é apresentado os resultados obtidos durante e após a finalização do teremim digital.

4.1 Código Definitivo

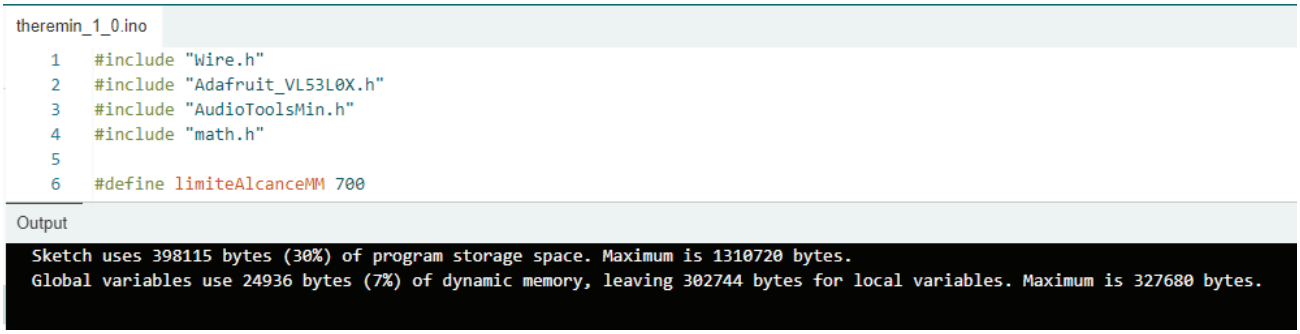
Devido a extensão dos códigos definitivos que foram criados e compilados no ESP32 a partir da IDE do Arduino, sendo eles o "theremin_1_0.ino", "AudioToolsMin.h" e "SoundGenerator.h", todos podem ser encontrados no repositório público do GitHub (SILVA, 2025), e não serão incluídos no corpo desse trabalho.

O código principal é o "theremin_1_0.ino", o qual possui toda a estrutura detalhada na seção 3.2.

Pela remoção de boa parte de outros *headers* (arquivos *.h*) que eram por padrão incluídos na biblioteca "AudioTools.h", foi criada a versão "AudioToolsMin.h". Isso foi necessário pois a versão padrão da biblioteca, juntamente com as demais presentes no código fonte, excediam a memória do ESP32, causando falha no *upload* do código para o microcontrolador.

Isso permitiu passar de um projeto que sequer era compilado para apenas 30% de memória de programa utilizada, conforme exibido na Figura 14. O espaço de memória para variáveis também foi pouco utilizado, sendo apenas 7% de sua capacidade.

Figura 14 – Saída no Console da Compilação do Código do Teremim Digital



Fonte: Autoria própria (2025).

Também a aplicação da suavização por exponencial simples (SES) deixou o dispositivo mais refinado, pois o áudio passou de ter saltos digitais de sua frequência (lida no sensor de frequência) para um som mais suave, sendo mais semelhante a um teremim analógico.

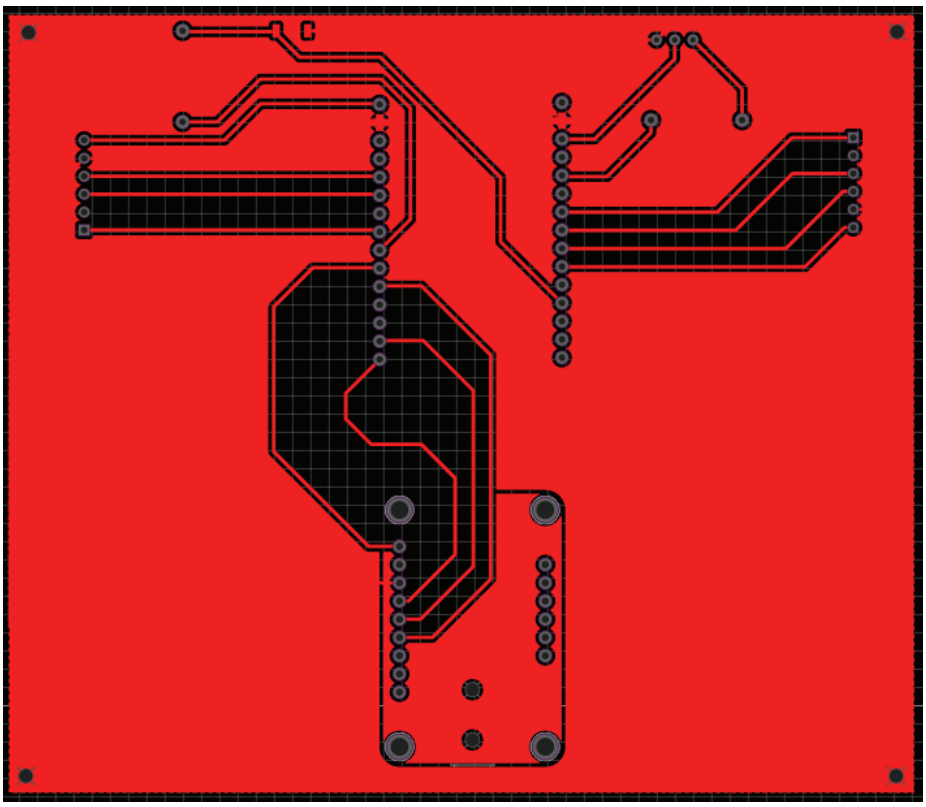
Como o processamento do áudio é feito dentro dos arquivos da biblioteca *AudioTools*, as funções de distorção foram incluídas no arquivo "SoundGenerator.h", que agora além do processamento do áudio propriamente dito, também define e seleciona qual a distorção será reproduzida.

4.2 Confecção da PCI

Inicialmente o teremim foi criado em uma *protoboard*, ligando seus respectivos componentes de forma a validar e realizar eventuais ajustes no modelo.

A partir do momento que o mesmo estava funcional: os dois sensores, o botão que altera frequência e o seletor de frequência máxima estavam todos funcionando conforme o esperado, foi feita a projeção do circuito no *software* EasyEDA (COMPANY, 2025), conforme ilustrado na Figura 13, na seção 3.2. Após isso, foi modelado no mesmo *software* a PCI (Placa de Circuito Impresso) dela, também chamada de *PCB* (*Printed Circuit Board*), resultando na Figura 15.

Figura 15 – PCI do teremim digital



Fonte: Autoria própria (2025).

Importante ressaltar que as partes em vermelho representam onde há circuito fechado, são as trilhas que conectam cada componente do teremim. Boa parte da figura está em vermelho pois é a malha de terra, conectada diretamente aos pinos GND do ESP32.

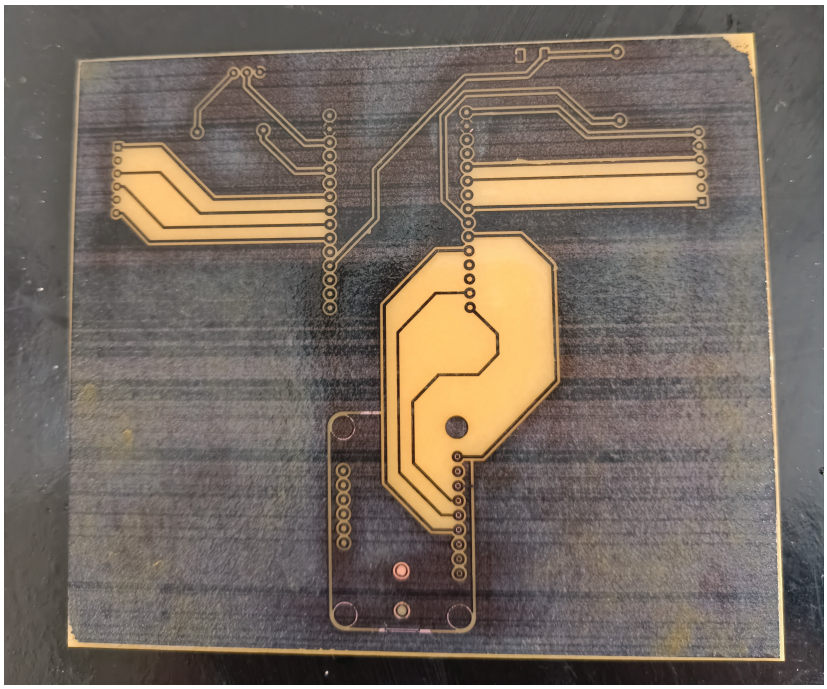
Foi optado por fazer uma PCI relativamente grande (13 cm x 11 cm) para facilitar o posicionamento dos componentes, em relação ao seu modo de operação, evitando a utilização de *jumpers* para a conexão entre os dispositivos.

Também durante a modelagem da PCI foi necessário alterar a pinagem de alguns dos componentes relacionados ao ESP32, pois algumas das trilhas se cruzavam causando um curto circuito. Por fim, a versão definitiva dos pinos é que pode se encontrar justamente no "theremin_-1_0.ino" no GitHub (SILVA, 2025).

Com a modelagem finalizada foi realizada a confecção da PCI, feita utilizando uma placa retangular de fenolite: material composto por uma camada de papel celulose e uma camada de cobre, sendo elas ligadas por uma resina fenólica.

Utilizando materiais disponibilizados pela UTFPR câmpus Toledo, foi feita a impressão do circuito da Figura 15 em um papel fotolito, transferido o mesmo para a placa fenolítica pelo contato com um ferro quente, após o esfriamento da mesma foi removida manualmente a celulose extra que ficou acoplada e então feito o processo de corrosão com percloreto férrico, resultando na Figura 16.

Figura 16 – PCI após o processo de impressão e corrosão



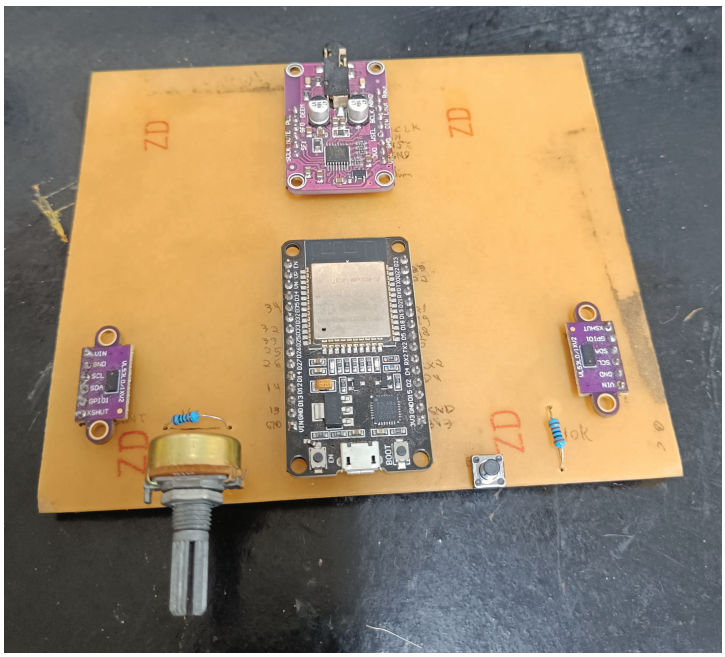
Fonte: Autoria própria (2025).

Depois disso, foi necessário remover a tinta (preta) visível, pois é onde estão as trilhas da PCI, realizada por raspagem com palha de aço. Após, foi feita a perfuração da PCI nos pontos de conexão dos componentes.

Com a PCI confeccionada, utilizando ferro de solda e estanho, foi feita a soldagem de todos os componentes em seus respectivos lugares, finalizando assim a confecção de toda a estrutura essencial para o funcionamento do dispositivo, o teremim digital com distorção.

A face superior do teremim pode ser vista na Figura 17.

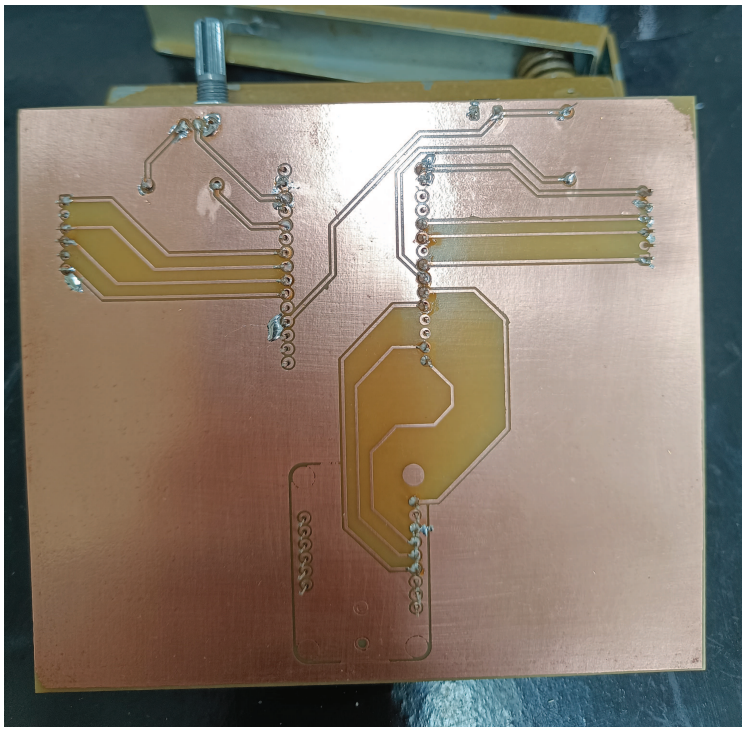
Figura 17 – Face superior do Teremim Digital



Fonte: Autoria própria (2025).

E a face inferior do mesmo, onde ocorreram as soldas, pode ser vista na Figura 18.

Figura 18 – Face inferior do Teremim Digital

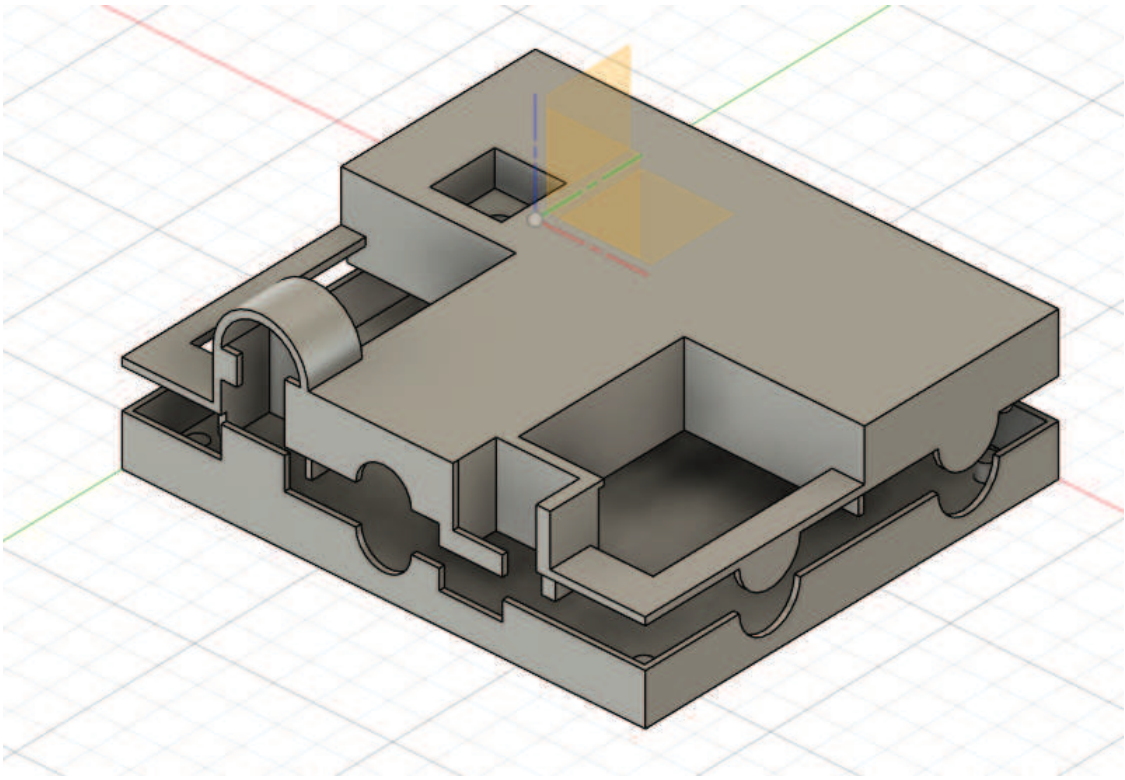


Fonte: Autoria própria (2025).

4.3 Confecção da caixa para comportar a PCI

Utilizando o software Fusion 360 foi feita a confecção 3D da caixa que irá comportar a PCI. Uma visão panorâmica da base e tampa da caixa podem ser vistas na Figura 19.

Figura 19 – Modelo da caixa para comportar a PCI feita no Fusion 360



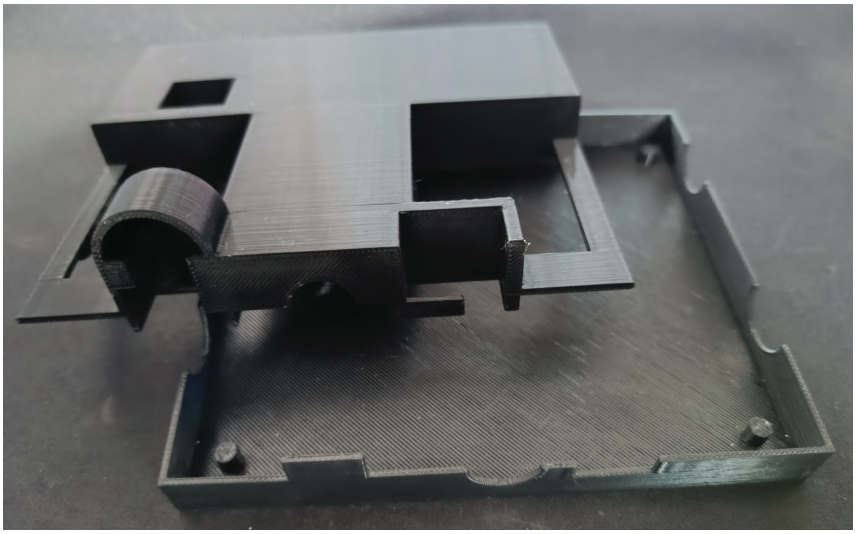
Fonte: Autoria própria (2025).

Ela possui pequenos pilares que suspendem em 6mm a PCI acima da base da caixa, a qual possui suas paredes com 1,5mm de espessura. A parte traseira possui um vão para conectar a entrada P2 com o alto-falante, enquanto a parte da frente possui um vão para a alimentação do ESP32 via micro USB, um vão para o botão que altera a distorção e outro para o potenciômetro seletor de frequência máxima.

Na parte superior há dois vãos para que os sensores de frequência e amplitude possam captar a distância da mão do usuário, e também há um deslocamento com uma base para acoplar um parafuso que será pressionado por uma porca, fixando a tampa com a base.

A Figura 20 exibe a caixa impressa fisicamente em uma impressora 3D no campus Toledo da UTFPR.

Figura 20 – Caixa impressa para comportar a PCI



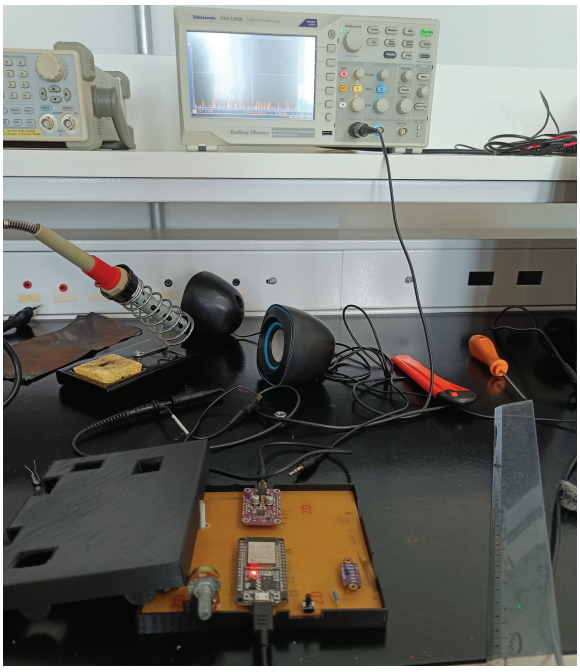
Fonte: Autoria própria (2025).

4.4 Análise de Frequências atingidas

Utilizando um osciloscópio digital TBS 1102B no laboratório da UTFPR, foi catalogada a relação entre a distância em relação ao sensor de frequência e a respectiva frequência reproduzida pelo teremim, utilizando como saída a função trivial dele, a senoidal pura (função 1 do Quadro 2).

A Figura 21 ilustra como foi feita a medição, utilizando um cabo P2 cortado e ligado aos conectores do osciloscópio.

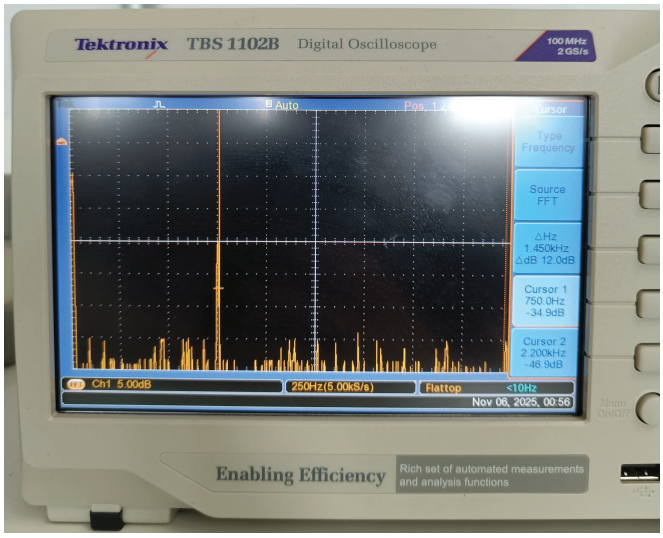
Figura 21 – Aparato de medição



Fonte: Autoria própria (2025).

E a Figura 22 mostra a medição feita no osciloscópio por meio da FFT (*Fast Fourier Transform*), junto com o apoio de cursores, linhas verticais marcadas manualmente nos picos em dB que indicam a frequência atual da onda senoidal.

Figura 22 – Osciloscópio captando a onda em 750Hz no cursor



Fonte: Autoria própria (2025).

Colocando a primeira versão da tampa diretamente acima do sensor de amplitude (conforme Figura 21), para que o sinal de saída seja captado de forma mais nítida, e fixando o seletor de frequência máxima no mínimo possível, foram obtidos os dados conforme a Tabela 1.

Tabela 1 – Tabela da relação frequência x distância com o seletor de frequência máxima no ponto mínimo

Distância (cm)	Frequência obtida (Hz)
0	750
5	700
10	650
15	600
20	550
25	500
30	450
35	400
40	390
45	340
50	300
55	260
60	220
65	210
70^{-}	200
75	0

Fonte: Autoria própria (2025).

Também foram coletados dados fixando o seletor de frequência máxima no máximo possível, resultando nos dados da Tabela 2.

Tabela 2 – Tabela da relação frequência x distância com o seletor de frequência máxima no ponto máximo

Distância (cm)	Frequência obtida (Hz)
0	5600
5	5200
10	4800
15	4400
20	4000
25	3600
30	3200
35	2850
40	2450
45	2200
50	1850
55	1450
60	1050
65	750
70^{-}	350
75	0

Fonte: Autoria própria (2025).

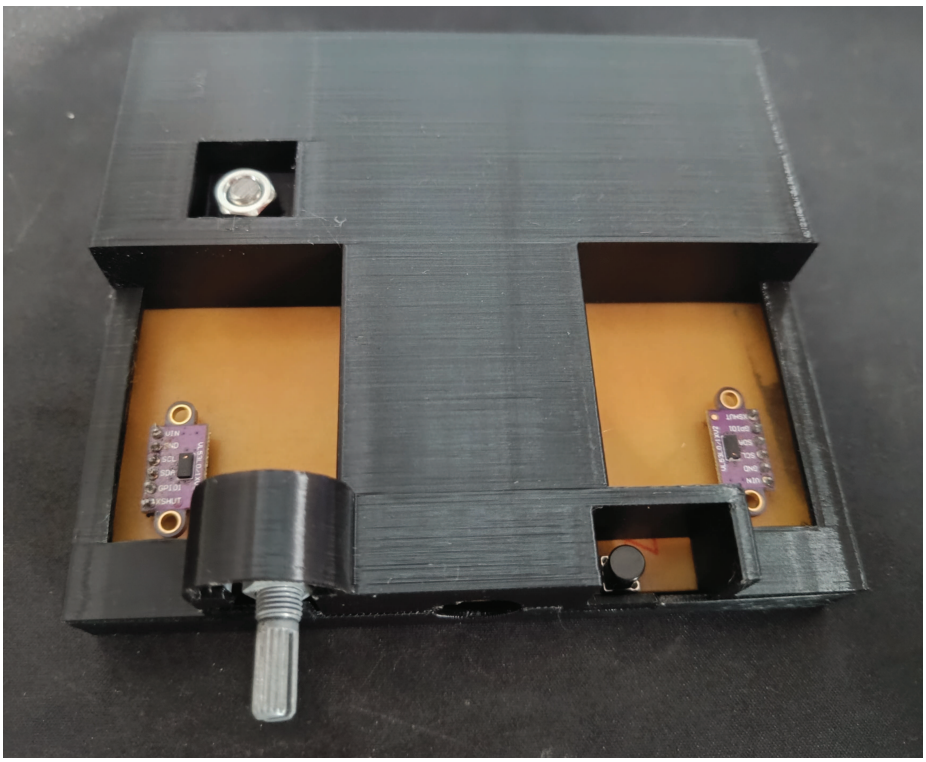
A partir de 70 cm é considerado como sem leitura, portanto o resultado é zero, mesmo que os sensores capturem até 2 m.

4.5 Modelo Final

Após a impressão da caixa pela impressora 3D, foi feito um furo na placa de pouco menos de 3,5 mm (na região de malha de terra, sem afetar as trilhas do circuito) para que um parafuso, fixado com cola na base, faça a fixação entre a tampa e a base, com o auxílio de uma porca metálica para prensar.

O dispositivo final, fisicamente implementado, pode ser visualizado na Figura 23.

Figura 23 – Teremim Digital com Distorção



Fonte: Autoria própria (2025).

4.6 Discussões

Durante a implementação alguns problemas foram encontrados, como por exemplo uma primeira versão da tampa causava interferência nos sensores, sendo necessário receber ajustes e então a confecção de uma nova, que é a atual presente no trabalho.

A disposição das trilhas para o potenciômetro não estava adequada, o que exigiu um reparo manual nas mesmas, sendo soldado um *jumper*.

As frequências catalogadas estão diferentes das esperadas, pois com o potenciômetro com a menor resistência captou uma frequência máxima de 750 Hz em vez de 400 Hz, ao mesmo tempo que com o mesmo na maior resistência, fazendo com que o teremim opere na maior faixa de frequência possível, o mesmo atingiu até 5600 Hz em vez de 3000 Hz.

As distorções arbitrariamente definidas e implementadas reproduziram sons interessantes no teremim, algumas mais semelhantes a outras devido as suas próprias funções, como por exemplo a função 4 sendo a função 1 mas com outra senoidal somada, já outras possuindo um som bem único, como a 5 sendo bastante agressiva, inclusive lembrando um ruído, enquanto a 6, 7 e 8 se assemelhando ao som de uma abelha.

Outro ponto que pode ser notado é que não houve uma certa linearidade de frequência reproduzida conforme era aumentada a distância, isso deve ter ocorrido devido a maiores ruídos percebidos pelos sensores a medida que a distância aumenta.

5 CONCLUSÃO

O projeto apresentado consiste na implementação de um teremim digital utilizando o ESP32 como microcontrolador. Este dispositivo é responsável por processar os sinais de entrada provenientes dos sensores de distância, que alteram a amplitude e a frequência do sinal senoidal de saída.

Esse sinal também pode receber distorções programadas, conforme pressionamento do botão. Foi utilizado o conversor digital-analógico (DAC) UDA1334 para receber o sinal digital e convertê-lo em analógico, permitindo que o sinal de saída seja reproduzido em um alto-falante com uma precisão de 16 bits, adequado para o som.

Tomando como referência a onda senoidal pura sem distorção, pode-se assumir qualitativamente como satisfatório o som emitido pelo mesmo, tendo uma relativa semelhança ao som de um teremim analógico. Porém, uma limitação do teremim digital em relação ao analógico é que não há efeito no som reproduzido ao mexer as mãos, que no analógico causa alterações no sinal captado pela antena, enquanto no digital é apenas considerado a distância lida em relação ao sensor óptico.

A relação das frequências catalogadas não foram conforme o esperado porém isso não é necessariamente um problema, pois não afeta a finalidade do dispositivo, somente sua faixa fixa de operação.

O teremim possui uma operação bem simplificada. Seu funcionamento que exige apenas uma caixa de som com entrada P2 e um micro USB conectado ao microcontrolador em uma fonte de alimentação de 5V e 2A, comuns de uso no dia-a-dia, permite seu uso imediato.

Caso seja notado falha em algum de seus dispositivos, ele pode ser trocado de forma ágil em relação a um teremim analógico, que exigiria uma análise do circuito como um todo a fim de se identificar onde estaria o problema.

Para um futuro trabalho seria interessante a implementação de uma interface MIDI para a saída de áudio ou envio para outro dispositivo via Bluetooth, permitindo a interação do dispositivo com computadores e outros *hardwares*, e também uma possível inserção de novas funções de distorção, por meio de algum novo controle ou um ambiente que inserisse dentro do próprio código uma nova função.

REFERÊNCIAS

ARAUJO, e. a. W. M. Visão geral sobre microcontroladores e prototipagem com arduino. **Revista Tecnologias em Projeção**, UniPROJEÇÃO, ago 2019. ISSN 2178-6267. Disponível em: https://projecaociencia.com.br/index.php/Projecao4/article/view/1357/1062. Acesso em: 02 fev. 2025.

ARDUINO. **Arduino IDE**. 2025. Disponível em: https://www.arduino.cc/en/software. Acesso em: 12 jan. 2025.

BODANESE, J. P. **IMPLEMENTAÇÃO DE UM EQUALIZADOR DE ÁUDIO EM DSP**. 2008. Site UOL Educação. Disponível em: https://www.professorpetry.com.br/Ensino/Defesas_Pos_Graduacao/Defesa%2001_Joao%20Paulo%20Bodanese_Implementacao%20de%20Equalizador%20de%20Audio%20em%20DSP.pdf. Acesso em: 07 jan. 2025.

BRAGA, N. C. **Fundamentos de Som e Acústica**. São Paulo, BRA, 2015.

COELHO, L. C. **Utilização de modelos de suavização exponencial para previsão de demanda com gráficos de controle combinados Shewhart-CUSUM**. Universidade Federal de Santa Catarina, 2008. Disponível em: https://qualimetria.paginas.ufsc.br/files/2013/02/6768_leandrodissertacao1.pdf. Acesso em: 14 out. 2025.

COMPANY, E. **EasyEDA**. 2025. Disponível em: https://easyeda.com. Acesso em: 16 out. 2025.

DODGE, T. A. J. C. **Computer Music**: Synthesis, composition and performanc. Thomson Learning, 1985. ISBN 0·02-864682-7. Disponível em: https://webarchiv.servus.at/campus/2011/music_class/class1/Dodge__Jerse_-_Computer_Music__Synthesis__Composition__and_Performance_2._ed.pdf. Acesso em: 02 fev. 2025.

DOIDIC, e. a. M. **Tube Modeling Programmable Digital Guitar Amplification System**. 1998. United States Patent nº 20040258250. Disponível em: https://patentimages.storage.googleapis.com/d5/3d/2a/8e72b2bd8bde16/US5789689.pdf. Acesso em: 02 fev. 2003.

ESPRESSIF. **ESP32 Series**: Datasheet Version 4.7. [*S.l.*], 2024. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Acesso em: 07 jan. 2025.

GIRISTUDIO. **Arduino Bluetooth Controller**. 2025. Disponível em: https://play.google.com/store/apps/details?id=com.giristudio.hc05.bluetooth.arduino.control. Acesso em: 15 out. 2025.

HEXSEL, R. A. **Sistemas Digitais e Microprocessadores**. Paraná, BRA, 2006. 175 p.

HOME, E. **VL53L0X Time Of Flight Distance Sensor**. 2025. Disponível em: https://esphome.io/components/sensor/vl53l0x. Acesso em: 23 set. 2025.

INSTRUMENTS, T. . **TMS320C5515/14/05/04 DSP Inter-IC Sound (I2S) Bus**. [*S.l.*], 2010. Disponível em: https://www.ti.com/lit/ug/sprufx4b/sprufx4b.pdf?ts=1736467780198&ref_url=https%253A%252F%252Fwww.google.com%252F. Acesso em: 09 jan. 2025.

LACERDA, F. de. **CONVERSOR DSB-SSB A CAPACITORES CHAVEADOS POR TRANSFORMADOR DE HILBERT EM TECNOLOGIA CMOS DE 180 nm**. mar. 2017. 129 p. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, mar. 2017. Disponível em: https://pantheon.ufrj.br/bitstream/11422/6234/1/865988.pdf.

LIU, T.-C. A modified quad-theremin for interactive computer music control. **IEEE**, p. 1–2, 2011.

LUSTOSA, L. e. a. **Planejamento e Controle da Produção**. Elsevier Editora Ltda., 2008. ISBN 978-85-352-2026-1. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=Gp97f09X7YEC&oi=fnd&pg=PA1&dq=LUSTOSA,+L.+et+al.+Planejamento+e+Controle+da+Produç~ao.+4.+ed.:+Rio+de+Janeiro:+Elsevier,+2008.&ots=wEze4RRedI&sig=28s7VREm6GOlPEwLpTGMbeyFyJ4&redir_esc=y#v=onepage&q&f=false. Acesso em: 14 out. 2025.

MORANDINI, M. **Subsídios para o Teste de Software Orientado a Objetos: Definição e Mapeamento de Programas C++ para a LI++**. dez. 1996. 121 p. Dissertação (Mestrado) — Universidade de São Paulo, São Paulo, dez. 1996.

NIKITIN, P. Leon theremin (lev termen). **IEEE Antennas and Propagation Magazine**, v. 54, n. 5, p. 1–2, 2012.

OLIVEIRA, e. a. T. C. A. Modelagem computacional de efeitos de distorções não lineares para guitarra elétrica. **Revista Brasileira de Computação Aplicada**, Universidade de Passo Fundo, out 2013. ISSN 2176-664. Disponível em: https://seer.upf.br/index.php/rbca/article/view/2877/2371. Acesso em: 02 fev. 2025.

PAGANELLI, S. da S. **ANÁLISES DE MODELOS QUANTITATIVOS DE PREVISÃO DA DEMANDA: AJUSTE E OTIMIZAÇÃO DE MODELOS À DEMANDA DO ADESIVO COMUM EM UMA GRÁFICA NA CIDADE DE BELÉM-PA**. Universidade Federal do Pará, 2014. Disponível em: https://bdm.ufpa.br/server/api/core/bitstreams/18b18197-d397-4231-8a77-ac0264f28859/content. Acesso em: 14 out. 2025.

PEREIRA, E. de S. **Discriminação de Diferença de Frequência de Sons e Apendizagem de Leitura Musical**. 2012. Disponível em: http://www.realp.unb.br/jspui/bitstream/10482/10457/1/2012_EmersondeSousaPereira.pdf.

PROJECTS, T. E. **ESP32 Pinout, Datasheet, Features Applications**. 2020. Disponível em: https://www.ansys.com/products/electronics/ansys-hfss. Acesso em: 23 set. 2025.

RIGHI, A. K. V. **Transdução sonora por banda de frequência para sinalização luminosa**. 2019. Disponível em: https://repositorio.ufsm.br/bitstream/handle/1/27747/Righi_Audren_Karine_Veduim_2019_TCC.pdf?sequence=1&isAllowed=y.

SADIKU, C. K. A. M. N. O. **Fundamentos de Circuitos Elétricos**. Bookman Editora, 2013. ISBN 9788582600542. Disponível em: https://books.google.com.br/books?id=6WU3AgAAQBAJ. Acesso em: 07 jan. 2025.

SCHATZMANN, P. **Arduino Audio Tools**. 2025. Disponível em: https://github.com/pschatzmann/arduino-audio-tools/blob/main/src/AudioTools.h. Acesso em: 25 jan. 2025.

SEMICONDUCTORS, N. **UDA1334ATS**: Low power audio dac with pll. [*S.l.*], 2000. Disponível em: https://www.nxp.com/docs/en/data-sheet/UDA1334ATS.pdf. Acesso em: 21 out. 2025.

SEMICONDUCTORS, N. **I2C-bus specification anduser manual**. 2021. Disponível em: https://www.nxp.com/docs/en/user-guide/UM10204.pdf. Acesso em: 08 out. 2025.

SILVA, F. A. P. **Diretório de arquivos do Teremim Digital**. 2025. Disponível em: https://github.com/theFLZ/teremim_digital/tree/main. Acesso em: 15 out. 2025.