UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”
FACULDADE DE ENGENHARIA
CÂMPUS DE ILHA SOLTEIRA

HENRIQUE DE MORAES RICARDI

IMPLEMENTAÇÃO DE UM TEREMIM BASEADO EM ONDAS ACÚSTICAS E TÉCNICAS DIGITAIS

Ilha Solteira
2024

HENRIQUE DE MORAES RICARDI

IMPLEMENTAÇÃO DE UM TEREMIM BASEADO EM ONDAS ACÚSTICAS E TÉCNICAS DIGITAIS

Trabalho de conclusão de curso apresentado à Faculdade de Engenharia de Ilha Solteira – Unesp como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Prof. Dr. Ricardo Tokio Higuti.
Orientador

Ilha Solteira
2024

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

R488i

Ricardi, Henrique de Moraes.
Implementação de um teremim baseado em ondas acústicas e técnicas digitais / Henrique de Moraes Ricardi. -- Ilha Solteira: [s.n.], 2024
74 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) - Universidade Estadual Paulista (UNESP), Faculdade de Engenharia, Ilha Solteira, 2024

Orientador: Ricardo Tokio Higuti

Inclui bibliografia

1. Teremim. 2. Sensor de Ultrassom. 3. Microcontroladores. 4. Filtros digitais. 5. Processamento digital de sinais.

Elaborado por Raiane da Silva Santos - CRB: 8/9999

unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos seis dias do mês de junho do ano de dois mil e vinte e quatro, o discente ***Henrique de Moraes Ricardi,*** matriculado sob o nº 172054087, tendo como banca examinadora o seu orientador, *o Prof. Dr. Ricardo Tokio Higuti, o Prof. Dr. Marcelo Carvalho Minhoto Teixeira e o Mestre Luis Miguel Quispe Valencia,* apresentou o Trabalho de Graduação intitulado **" Implementação de um Teremim baseado em Ondas Acústicas e Técnicas Digitais",** obtendo a nota 10,0 (DEZ) e conceito APROVADO.

Prof. Dr. Ricardo Tokio Higuti
- Orientador-

Henrique de Moraes Ricardi
- Discente -

Prof. Dr. Marcelo Carvalho Minhoto Teixeira
- Membro da Banca -

Mestre Luis Miguel Quispe Valencia
- Membro da Banca -

AGRADECIMENTOS

A realização deste Trabalho de Conclusão de Curso representa a culminação de uma jornada de aprendizado e crescimento, e não teria sido possível sem o apoio e a colaboração de muitas pessoas às quais sou profundamente grato.

Primeiramente, agradeço aos meus pais, Hélcio e Ofélia, por todo o amor, apoio incondicional, incentivo e confiança em minha capacidade em todos os momentos. Vocês são minha maior inspiração e motivação.

Aos meus irmãos, Amanda e Alexandre, pela grande parceria e amizade que vamos levar pelo resto da vida. Em especial a minha irmã, pelo enorme apoio junto aos meus pais durante a faculdade, sem o qual eu não estaria aqui hoje.

Ao meu orientador, Ricardo Tokio, pela paciência, atenção, orientação precisa e valiosas contribuições ao longo de todo o processo. Sua orientação e conhecimento foram fundamentais para a realização deste trabalho, desde a concepção da ideia deste projeto até sua conclusão.

Aos meus amigos e familiares, por todo o suporte e sempre acreditarem em mim. Em especial, agradeço aos colegas Hugo e Fernando, por terem contribuído significativamente para o desenvolvimento do projeto.

A todos os professores do curso, por compartilharem seu valioso conhecimento e por todas as lições que levarei para a vida profissional e pessoal.

Por fim, agradeço a todos que, de alguma forma, contribuíram para a realização deste trabalho. A cada um de vocês, meu sincero muito obrigado.

RESUMO

Um teremim é um instrumento musical eletrônico cujo funcionamento é tradicionalmente baseado em duas antenas, em geral, uma posicionada na vertical e outra na horizontal. De acordo com o posicionamento das mãos do músico em relação às antenas, a vertical controla o *pitch* ou tom, enquanto a horizontal controla o volume. O instrumento ainda pode contar com outros botões de controle fino suplementares. Neste projeto de Trabalho de Graduação (TG) foi desenvolvido um teremim baseado em sensores de distância ultrassônicos e a sintetização digital do som foi realizada com microcontrolador. O dispositivo possui o mesmo princípio básico de operação, com os parâmetros das ondas senoidais geradas sendo controlados com a distância das mãos do operador. O instrumento também possui um sistema para ajustar a área de detecção ao operador e sua envergadura, tornando o uso do instrumento mais ergonômico e consistente.

Palavras-chave: Teremim, Sensor de Ultrassom, Microcontroladores, Filtros Digitais, Processamento Digital de Sinais

ABSTRACT

A theremin is an electronic musical instrument traditionally based on two antennas, typically one positioned vertically and the other horizontally. Depending on the positioning of the musician's hands relative to the antennas, the vertical antenna controls the pitch or tone, while the horizontal antenna controls the volume. The instrument may also include additional fine-tuning control buttons. In this Undergraduate Research Project, a theremin was developed using ultrasonic distance sensors, with digital sound synthesis performed by a microcontroller. The instrument also includes a system to adjust the detection area to the operator and their arm span, making the use of the instrument more ergonomic and consistent.

LISTA DE FIGURAS

Figura 1 – Esquema de um teremim 13
Figura 2 - Diagrama de blocos dos componentes principais do teremim 16
Figura 3 – Sistema físico do teremim em protoboard 17
Figura 4 – Exemplo do funcionamento do sensor 21
Figura 5 – Sensor HC-SR04 21
Figura 6 – Operação regular do sensor de ultrassom 21
Figura 7 – Configuração de pinos ESP32 24
Figura 8 – Fluxograma geral do programa 26
Figura 9 – Fluxograma da geração do som iterativamente com filtro IIR 28
Figura 10 – Fluxograma do cálculo da frequência – Núcleo PRO 30
Figura 11 – Simulação de um filtro Phaser com dois estágios 36
Figura 12 – Pontos calculados do oscilador IIR com Simulação GNU Octave 38
Figura 13 - Resultado do oscilador IIR e saída com Simulação GNU Octave 39
Figura 14 - Paralelismo dos núcleos do ESP32 durante operação do algoritmo 40
Figura 15 – Sinal no domínio do tempo d_max = 30 cm; f_zero = 2600 Hz 44
Figura 16 – Sinal no domínio da frequência d_max = 30 cm; f_zero = 2600 Hz 44
Figura 17 - Sinal no domínio do tempo d_max = 39,79 cm; f_zero = 2600 Hz 45
Figura 18 - Sinal no domínio da frequência d_max = 39,79 cm; f_zero = 2600 Hz ...45
Figura 19 - Sinal no domínio do tempo d_max = 50 cm; f_zero = 2600 Hz 46
Figura 20 - Sinal no domínio da frequência d_max = 50 cm; f_zero = 2600 Hz 46
Figura 21 – Sinal no domínio do tempo d_max = 30 cm; f_zero = 1000 Hz 47
Figura 22 – Sinal no domínio da frequência d_max = 30 cm; f_zero = 1000 Hz 47
Figura 23 – Sinal no domínio do tempo d_max = 39,79 cm; f_zero = 1000 Hz 48
Figura 24 – Sinal no domínio da frequência d_max = 39,79 cm; f_zero = 1000 Hz ..48
Figura 25 – Sinal no domínio do tempo d_max = 50 cm; f_zero = 1000 Hz 49
Figura 26 – Sinal no domínio da frequência d_max = 50 cm; f_zero = 1000 Hz 49
Figura 27 – Demonstração do impacto do ruído com fs = 100 Hz – Domínio do tempo 50
Figura 28 - Demonstração do impacto do ruído com fs = 100 Hz – Domínio da frequência 50
Figura 29 – Sinal no domínio do tempo d_max = 30 cm; f_zero = 100 Hz 51
Figura 30 – Sinal no domínio da frequência d_max = 30 cm; f_zero = 100 Hz 51

Figura 31 – Sinal no domínio do tempo d_max = 39,79 cm; f_zero = 100 Hz 52
Figura 32 – Sinal no domínio da frequência d_max = 39,79 cm; f_zero = 100 Hz 52
Figura 33 – Sinal no domínio do tempo d_max = 50 cm; f_zero = 100 Hz 53
Figura 34 – Sinal no domínio da frequência d_max = 50 cm; f_zero = 100 Hz 53
Figura 35 - Sinal (Phaser) no domínio do tempo d_max = 30 cm; f_zero = 2600 Hz 55
Figura 36 - Sinal (Phaser) no domínio da frequência d_max = 30 cm; f_zero = 2600 Hz 55
Figura 37 - Sinal (Phaser) no domínio do tempo d_max = 50 cm; f_zero = 1000 Hz 56
Figura 38 - Sinal (Phaser) no domínio da frequência d_max = 50 cm; f_zero = 1000 Hz 56
Figura 39 - Sinal (Phaser) no domínio do tempo d_max = 38,98 cm; f_zero = 100 Hz 57
Figura 40 - Sinal (Phaser) no domínio da frequência d_max = 39,98 cm; f_zero = 100 Hz 57

LISTA DE TABELAS

Tabela 1 – Comparação de frequências no Sistema Temperado 33
Tabela 2 – Aquisição da amostragem do ESP32 72
Tabela 3 – Desvio da frequência (Hz) em relação a amostragem utilizada.....73

LISTA DE ABREVIATURAS E SIGLAS

ADC	Analog to Digital Converter (Conversor Analógico para Digital)
BLE	Bluetooth Low Energy (Bluetooth de Baixa Energia)
BR/EDR	Basic Rate/Enhanced Data Rate (Taxa Básica/Taxa de Dados Aprimorada)
CPU	Central Processing Unit (Unidade Central de Processamento)
DAC	Digital to Analog Converter (Conversor Digital para Analógico)
GPIO	General-Purpose Input/Output (Entrada/Saída de Uso Geral)
I2C	Inter-Integrated Circuit (Circuito Inter-Integrado)
IEEE	Institute of Electrical and Electronics Engineers (Instituto de Engenheiros Eletricistas e Eletrônicos)
IIR	Infinite Impulse Response (Resposta ao Impulso Infinito)
IoT	Internet of Things (Internet das Coisas)
IPC	Inter-Processor Call (Chamada entre Processadores)
LFO	Low-Frequency Oscillator (Oscilador de Baixa Frequência)
PWM	Pulse Width Modulation (Modulação por Largura de Pulso)
RAM	Random Access Memory (Memória de Acesso Aleatório)
ROM	Read-Only Memory (Memória Somente de Leitura)
SPI	Serial Peripheral Interface (Interface Periférica Serial)
SRAM	Static Random Access Memory (Memória de Acesso Aleatório Estática)
UART	Universal Asynchronous Receiver-Transmitter (Receptor-Transmissor Assíncrono Universal)

SUMÁRIO

1 INTRODUÇÃO 13
1.1 Motivação do Projeto 14
1.2 Objetivos 15
2 ESTRUTURAÇÃO E MATERIAIS 16
2.1 Características Gerais do Sistema 16
2.2 Esquema físico 17
2.3 Oscilador Digital 18
2.4 Pontos principais do desenvolvimento 19
2.5 Materiais 20
2.5.1 Sensor Ultrassônico HC-SR04 20
2.5.2 Microcontrolador ESP32 22
2.5.2.1 Características principais do ESP32 23
2.5.2.2 Programação do microcontrolador 25
2.5.2.3 Conclusões sobre o dispositivo 25
2.5.3 Programação 25
2.5.3.1 Diagrama de fluxo do programa 26
2.5.3.2 Definição de bibliotecas e variáveis globais 26
2.5.3.3 Setup 27
2.5.3.4 Loop principal 27
2.5.3.5 Loop núcleo PRO 27
2.5.3.6 Função Som() 28
2.5.3.7 Cálculo da frequência 30
2.5.3.8 Raw_cap() - Leitura crua 31
2.5.3.9 Função Position_calc – Cálculo das posições 31
2.5.3.10 Fuse_data - Fusão dos dados 32
2.5.3.11 Converter_exp - Conversão da distância por relação exponencial 33
2.5.3.12 aplicarPhaser - Filtro Phaser 35

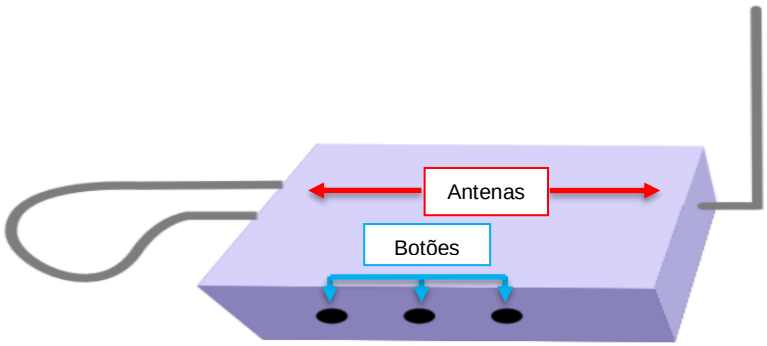
3 RESULTADOS 38
3.1 Testes do oscilador digital - Filtro IIR 38
3.2 Testes do paralelismo dos núcleos no ESP32 39
3.3 Cálculo da função de conversão da distância para frequência 40
3.4 Obtenção da frequência de amostragem do sistema ($\boldsymbol{fs}$) 42
3.5 Análise do sinal gerado pelo sistema 43
3.5.1 2600 Hz 44
3.5.1.1 Distância máxima: 30 cm 44
3.5.1.2 Distância máxima: 39,79 cm 45
3.5.1.3 Distância máxima: 50 cm 46
3.5.2 1000 Hz 47
3.5.2.1 Distância máxima: 30 cm 47
3.5.2.2 Distância máxima: 39,79 cm 48
3.5.2.3 Distância máxima: 50 cm 49
3.5.3 100 Hz 49
3.5.3.1 Ruído 50
3.5.3.2 30 cm 51
3.5.3.3 39,79 cm 52
3.5.3.4 50 cm 53
3.6 Análise do sinal com adição do filtro Phaser 54
3.6.1 2600 Hz – Ajuste: 30 cm 55
3.6.2 1000 Hz – Ajuste: 50 cm 56
3.6.3 100 Hz – Ajuste: 38,98 cm 57
4 CONSIDERAÇÕES FINAIS 58
REFERÊNCIAS 60
APÊNDICE A – Programação completa do ESP32 no ambiente Arduino IDE 62
APÊNDICE B – Algoritmo em Octave GNU para geração do sinal teórico 67
APÊNDICE C – ADAPTAÇÃO PARA AQUISIÇÃO DA AMOSTRAGEM DO DAC 69
APÊNDICE D – RESULTADOS DA AQUISIÇÃO DA AMOSTRAGEM 72

1 INTRODUÇÃO

O teremim é um instrumento musical eletrônico desenvolvido em 1920 por Leon Theremin (Nikitin, 2012) tendo em vista um instrumento onde o músico seja capaz de controlar o som com ajustes mais precisos e delicados do que em instrumentos convencionais, os quais podem ser limitados devido a interação com o instrumento ser feita por meio cordas, arcos ou teclas, impedindo o alcance de certas frequências.

A característica que difere o teremim nesse quesito é a interação entre músico e instrumento apenas por movimentos de aproximação, sem contato físico, caracterizando-o como um instrumento háptico. Um teremim possui duas antenas, sendo uma posicionada na vertical e a outra na horizontal, como ilustra a Figura 1.

Figura 1 – Esquema de um teremim



Fonte: Elaborado pelo autor

No instrumento regular, a posição relativa entre as mãos do operador e as antenas do teremim modifica a capacitância do circuito interno, o que altera a frequência de um circuito oscilador. A antena horizontal controla o volume do som gerado e a vertical controla a frequência, o que dará origem às diferentes notas musicais ou o *pitch*, em suma produzindo diferentes efeitos sonoros de acordo com a movimentação das mãos do operador. Controles adicionais presentes no corpo do instrumento em alguns modelos (ilustrados com os botões na Figura 1), permitem o ajuste fino de outros parâmetros como níveis de volume, afinação, timbre, formato de onda, entre outros.

Na prática, esse instrumento é sobretudo utilizado individualmente, tomando o papel da melodia. Seu som exótico foi utilizado para diversos temas e trilhas sonoras

nos mundos do filme e da televisão, muito associado a momentos misteriosos e a alienígenas. Também teve sua presença na música clássica e na popular, com novos compositores, intérpretes influentes e também com adaptações para peças e músicas já existentes.

Curiosamente, o teremim eletromagnético nasceu de estudos sobre sensores de proximidade, em um período de guerras. Foram encontradas algumas referências que utilizam o teremim convencional para outras aplicações (Tsung-Ching et al. 2011, Garcia-Guzman et al. 2018), e também utilizando sensores de ultrassom (Gomes et al. 2009, Hanindhito et al. 2015). No lugar da capacitância, sensores de ultrassom medem a distância para as mãos do operador e essa informação é utilizada para controlar o volume e o *pitch*. Ao passo que, em um teremim eletromagnético, as mãos podem estar em diferentes posições relativas às antenas, no caso do teremim ultrassônico as mãos deverão estar posicionadas na região do feixe dos sensores, bem à sua frente. Isso pode ser uma vantagem, em se tratando da existência de outras pessoas ou outros teremins operando próximos, mas exigirá um estudo da diretividade e sensibilidade dos sensores. Também existirá limitação de detecção de distâncias muito pequenas e muito grandes em relação aos sensores. Fez-se então necessário o estudo dos transdutores, seu tempo de resposta aos movimentos e a abertura do feixe, ou seja, seu cone de detecção, relacionado ao diagrama de radiação acústico.

Para a geração dos sinais senoidais, foi utilizado um oscilador digital operando em um microcontrolador, em contraste com o oscilador analógico utilizado em teremins “convencionais". Além disso, outros circuitos analógicos poderão ser utilizados para amplificar e adequar os níveis de tensão para o processamento digital e para gerar os sinais analógicos.

1.1 Motivação do Projeto

A possibilidade de interferência no teremim causada pela característica eletromagnética do ambiente é uma limitação marcante do instrumento. A presença de múltiplos teremins ou músicos próximos ao instrumento pode causar interferência, limitando o uso de vários instrumentos ao mesmo tempo e no mesmo local. Também é incapaz de tocar mais de uma nota por vez, ou seja, é um instrumento monofônico, apesar de existirem versões polifônicas, que por sua vez são bem mais difíceis de operar.

Utilizando sensores de ultrassom, a proposta é gerar um efeito semelhante ao teremim, mas com uma detecção menos intrusiva e mais consistente, possibilitando o uso de múltiplos teremins em conjunto. O uso de técnicas digitais poderá facilitar a introdução de outras técnicas de processamento do som, além de ser um instrumento com componentes de baixo custo, fácil acesso e altamente customizáveis. Com isto, pode ser criada uma opção mais versátil do instrumento.

1.2 Objetivos

O objetivo do trabalho é implementar um teremim baseado em sensores de distância ultrassônicos utilizando técnicas de processamento digital de sinais. O dispositivo deverá gerar sinais senoidais sintéticos com diferentes frequências e amplitudes, de acordo com a movimentação das mãos de um operador em relação aos sensores e possibilitar a aplicação de efeitos digitais ao sinal sintético antes da sua reprodução em um amplificador de som.

2 ESTRUTURAÇÃO E MATERIAIS

Para a realização do plano, foi necessário o estudo dos transdutores para medição por ultrassom, seu tempo de resposta aos movimentos e a abertura do feixe, ou seja, seu cone de detecção, relacionado ao diagrama de radiação acústico.

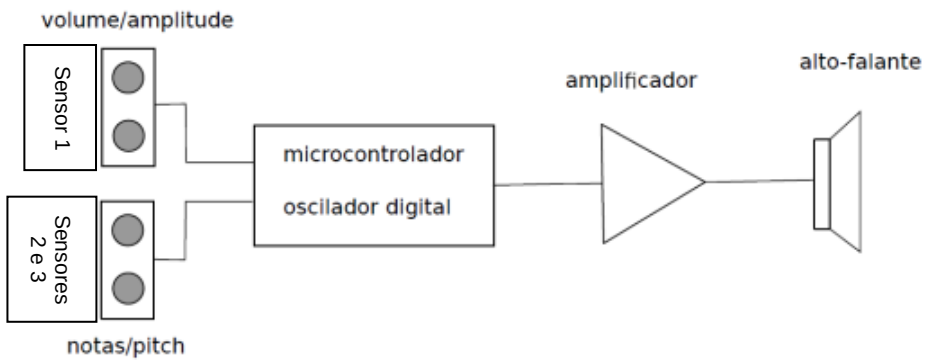
Quanto a geração dos sinais senoidais, foram avaliados diferentes osciladores digitais e a possibilidade de sua integração no sistema, levando em conta a qualidade do sinal que seria gerado e a facilidade computacional para tal, visto que a velocidade do processamento é um fator limitante para o sucesso do projeto.

Para integrar os estudos, foi necessário o desenvolvimento de algoritmos compatíveis ao *hardware* utilizado e a aplicação desejada. Para tal, foi sincronizada a geração dos sinais senoidais com as leituras dos sensores ultrassônicos, permitindo que a frequência e a amplitude dos sinais fossem controladas em tempo real e fluidamente pelo operador.

2.1 Características Gerais do Sistema

O sistema desenvolvido utiliza três sensores ultrassônicos no total para medir distâncias, o que permite uma coleta de dados precisa e em tempo real. Esses sensores são configurados para enviar sinais de "trigger" e receber os sinais de "echo", calculando a distância com base no tempo de retorno do sinal. O sistema incorpora um conversor digital-analógico (DAC) no próprio microcontrolador para a geração de som, que é ajustado conforme as medições de distância. A informação está resumida no diagrama de blocos presente na Figura 2.

Figura 2 - Diagrama de blocos dos componentes principais do teremim



Fonte: Elaborado pelo autor.

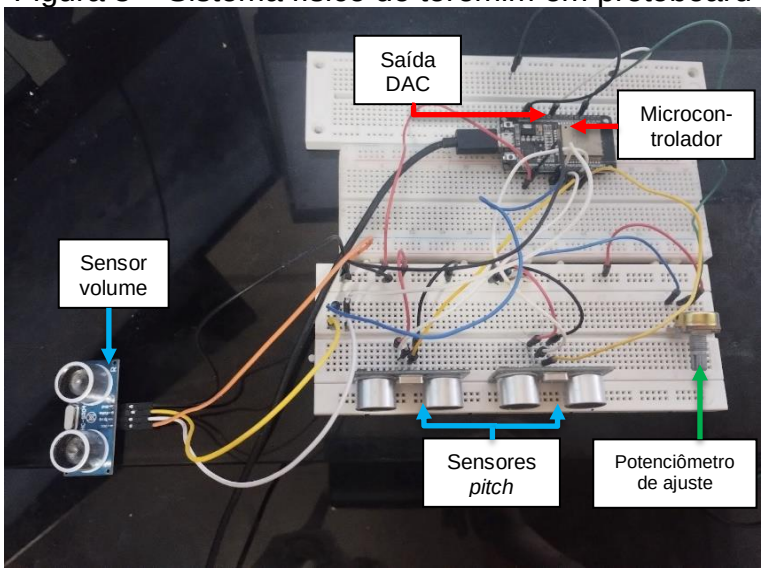
A geração de som é realizada através de um algoritmo que converte as distâncias medidas em frequências normalizadas para o sistema, que são em sequência definidas como variáveis da equação do oscilador digital responsável pela geração do sinal, permitindo a criação de sons com frequências variáveis com base na proximidade dos objetos. Um dos sensores também é responsável pelo controle do ganho ou amplitude do sinal. Ademais, foi implementado um potenciômetro ao sistema para ajuste da área de detecção dos sensores da frequência, podendo manipular a relação entre as distâncias e suas frequências correspondentes.

Outra característica importante do sistema é a utilização dos dois núcleos do ESP32, designados como PRO (PROtocol - CPU0) e APP (APPlication - CPU1) (Espressif Systems, 2024), para distribuir as tarefas de medição e geração de som. Essa abordagem de processamento em múltiplos núcleos melhora a eficiência e a capacidade de resposta do sistema, contribuindo para que as medições e a geração de som sejam feitas de forma coordenada e sem atrasos perceptíveis.

2.2 Esquema físico

O sistema foi desenvolvido em uma placa de ensaio ou *protoboard*, que oferece grande flexibilidade para prototipagem de sistemas pela facilidade na alteração dos componentes sem a necessidade de soldagem, como pode ser visto na Figura 3.

Figura 3 – Sistema físico do teremim em protoboard



Fonte: Elaborado pelo autor.

Foi conservada a disposição básica para operar o teremim, ou seja, manteve-se os sensores que controlam a frequência à direita do operador, sendo possível operá-lo com movimentos aproximando e afastando a mão entre a antena e o próprio corpo do músico. Da mesma forma, a antena da amplitude ficou estabelecida à esquerda, permitindo o ajuste do volume com movimentos ascendentes e descendentes da mão esquerda quando acima do sensor.

Adicionou-se um potenciômetro para poder ajustar a área de detecção dos sensores, visto que a melhor posição ergonômica varia entre operadores e a distância ideal para cada pode diferir em função de altura, dimensão dos braços, entre outras características pessoais. Podem também ser adicionados diversos outros potenciômetros para o controle de filtros, visto que apenas os pinos GPIO (General-Purpose Input/Output) 6 e 11 do microcontrolador não são capazes de receber inputs de dados analógicos.

Por fim, foram disponibilizados dois pinos de saída: de referência (GND) e o pino do output do som, ambos do lado oposto da placa em relação aos sensores. Desta forma, evita-se interferências de cabos do amplificador com os sensores.

2.3 Oscilador Digital

A geração do som é feita por um filtro digital de resposta ao impulso de duração infinita ou *Infinite Impulse Response* (IIR). O filtro em si possui função transferência descrita como traz a equação (1)

$$H(z) = \frac{1 - z^{-1}\cos(\omega_0)}{1 - 2z^{-1}\cos(\omega_0) + z^{-2}} \qquad (1)$$

Quanto aos polos da função, temos que o denominador será igual a zero quando z está no círculo unitário, conforme a equação (2),

$$z = \cos(\omega_0) \pm j\, sen(\omega_0) \qquad (2)$$

Desta forma, a equação possui estabilidade BIBO (*Bounded-Input Bounded-Output),* ou seja, com uma entrada limitada, é obtida uma saída limitada que não

diverge. Mais especificamente, a presença dos polos em $\pm\omega_0$ faz desse um filtro ressonante com frequência ω_0.

Verificada a função do filtro para este trabalho, foi obtida sua equação de diferenças para aplicação em algoritmo, que é definida conforme equação (3),

$$y[n] = x[n] - \cos(\omega_0)\, x[n-1] + 2\cos(\omega_0)\, y[n-1] - y[n-2] \qquad (3)$$

Como $x[n] = \delta[n]$, podemos simplificar conforme a equação (4) para,

$$y[n] = 2\cos(\omega_0)\, y[n-1] - y[n-2] \quad (4)$$

Onde: $y[0] = 1$;

A equação neste formato se torna mais fácil de aplicar em algoritmos, visto que possui parcelas simples de se calcular de forma iterativa.

2.4 Pontos principais do desenvolvimento

Para o desenvolvimento do sistema, foi necessário abordar os seguintes pontos principais:

A. **Precisão dos Sensores**: A precisão dos sensores é uma das características determinantes sobre a responsividade do sistema. Caso medidas sequenciais variem muito com o mesmo posicionamento das mãos, poderão ocorrer efeitos onde a frequência oscila entre notas involuntariamente, tornando o uso do instrumento mais difícil e menos consistente. A acurácia causaria um menor impacto, visto que o instrumento pode ser ajustado para prever um desvio nas medidas desde que sejam consistentes e com baixa variação.

B. **Geração de Som**: O sistema utilizado para geração do som definirá a qualidade do som, como seu timbre, clareza e suavidade. Foi necessário otimizar o código para garantir que o som gerado fosse claro e responsivo às mudanças nas medições de distância, sempre cumprindo os requisitos de velocidade para amostragem das diferentes frequências do espectro audível.

C. **Conversão de Distância para Frequência**: Métodos para realizar a conversão de distância para frequência de forma a melhorar a usabilidade do instrumento, sem prejudicar a geração.

D. **Sincronização de Tarefas**: Por fim, a utilização dos dois núcleos do ESP32 para distribuir tarefas de medição e geração de som deve ser executada de

forma coordenada para evitar conflitos e garantir a precisão das medições e da geração de som.

E. **Aplicação de efeitos digitais**: Será avaliada a possibilidade de aplicação de efeitos com filtros digitais etapa final da geração do sinal e suas possíveis alterações à qualidade do sinal.

Portanto, estes pontos e os esforços aplicados para tornar o sistema possível e otimizá-lo serão abordados nas próximas seções.

2.5 Materiais

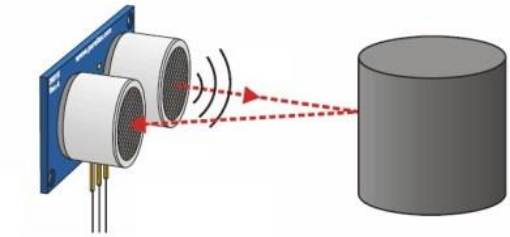
Para elaboração do projeto, foram utilizados diversos componentes comerciais que serão abordados nos próximos subitens, assim como o algoritmo desenvolvido para o funcionamento do teremim.

2.5.1 Sensor Ultrassônico HC-SR04

Sensores de ultrassom são dispositivos que utilizam ondas ultrassônicas para medir distâncias ou detectar objetos. Eles funcionam emitindo um pulso ultrassônico e medindo o tempo entre a emissão e o eco retornar ao sensor. Esses sensores são amplamente utilizados em aplicações de robótica, automação e sistemas de segurança devido à sua precisão, confiabilidade e baixo custo (Johnson, 2016).

O sensor de ultrassom HC-SR04 é um dos sensores de distância mais populares e amplamente utilizados em projetos de eletrônica e robótica. Ele é conhecido por sua simplicidade de uso, precisão e baixo custo. Este é constituído por um par de transdutores com funções de transmissor e receptor responsáveis, respectivamente, pela transmissão de um sinal de ultrassom com frequência de 40 kHz e pela detecção destas ondas após serem refletidas pelo ambiente ou objeto de interesse, conforme a Figura 4 exemplifica. Para isso, possui um circuito de controle integrado. O sensor é capaz de medir distâncias que variam de 2 cm a 400 cm, com uma precisão de aproximadamente 3 mm (Parikh, 2018).

Figura 4 – Exemplo do funcionamento do sensor



Fonte: (Mota, 2017)

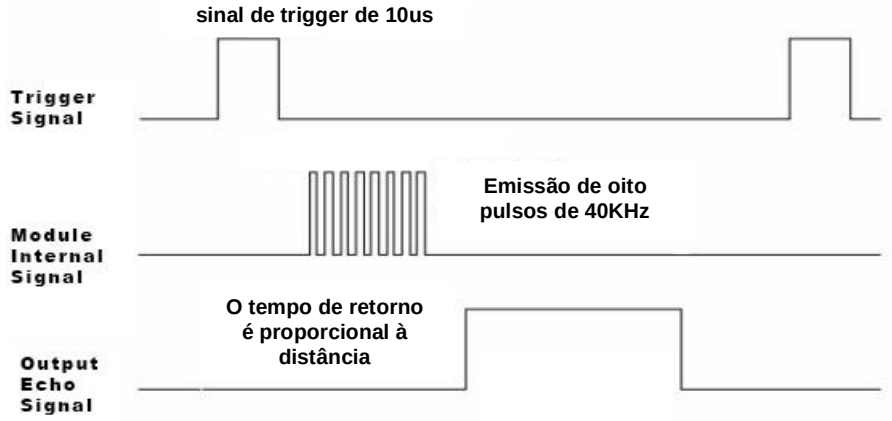
O modelo utilizado, visto na Figura 5, possui 4 pinos no total. Estes são: pinos de alimentação (GND e VCC), pino para ativação do transmissor (“Trigger”) e pino da saída do receptor (“Echo”), todos representados na imagem.

Figura 5 – Sensor HC-SR04

(Eletrogate, 2024)

Ao ativar o pino “Trigger” por 10 microssegundos, o transdutor correspondente gera 8 pulsos de ultrassom, que se propagam pelo ambiente. Ao mesmo tempo, o pino “echo” atrelado ao receptor se mantém no nível lógico alto, até ser detectado um sinal de ultrassom refletido. Na detecção, o pino “echo” alterna para o nível lógico baixo, como pode ser visto na Figura 6.

Figura 6 – Operação regular do sensor de ultrassom



Fonte: Adaptado de (Chao, 2012).

A partir do sinal no pino “echo” e conhecendo a velocidade do som no meio (ar) de aproximadamente $v_{som} = 343\,(\mathrm{m/s})$, pode-se fazer uma relação simples para encontrar a distância percorrida pelo ultrassom. Considerando que a onda percorre a distância duas vezes, o tempo T/2 até o objeto e T/2 para retornar ao sensor receptor, podemos obter a essa medida conforme equação (5).

$$d = v_{som} \times \frac{T_{echo}}{2} \qquad (5)$$

substituindo o valor de V = 343 m/s, temos a equação (6).

$$d = 343 \times \frac{T_{echo}}{2} \qquad (6)$$

Sendo:
d = distância entre o sensor e o objeto refletor da onda (m);
T_{echo} = Tempo do pulso no pino “echo” do sensor (s).

Em suma, o intervalo de tempo em que o pino permanece em nível lógico alto é utilizado para obter a distância percorrida pelas ondas até retornarem aos sensores, desde que seja conhecida a velocidade da onda no meio em que foi propagada.

O sensor de ultrassom HC-SR04 será utilizado para medir a distância das mãos do operador em relação ao teremim ultrassônico para controlar a frequência (*pitch*) e a amplitude (volume) dos sinais senoidais gerados pelo ESP32. Será avaliada a responsividade do dispositivo e o sucesso da sua aplicação.

2.5.2 Microcontrolador ESP32

Microcontroladores são dispositivos eletrônicos compactos que combinam um processador, memória e periféricos de entrada e saída em um único *chip*. Eles são amplamente utilizados em sistemas embarcados para controlar processos e dispositivos em tempo real. A popularidade dos microcontroladores se deve à sua capacidade de executar tarefas específicas de forma eficiente e com baixo consumo de energia, tornando-os ideais para uma vasta gama de aplicações, desde

eletrodomésticos até sistemas automotivos (Oki; Mantovani, 2013) e dispositivos de Internet das Coisas (IoT) (Chéour et al., 2020).

Os microcontroladores são programáveis, o que permite aos desenvolvedores criar softwares personalizados para controlar o *hardware* de acordo com as necessidades específicas do projeto. Eles geralmente incluem uma Unidade Central de Processamento (CPU), memória de acesso aleatório ou RAM (do inglês *Random Access Memory*), memória somente de leitura ou ROM (do inglês *Read-Only Memory*) ou memória flash, e uma variedade de interfaces de comunicação, como UART, SPI e I2C (Maier, 2017).

2.5.2.1 Características principais do ESP32

O ESP32 é um microcontrolador altamente integrado com conectividade Wi-Fi (*Wireless Fidelity*) (e Bluetooth, desenvolvido pela Espressif Systems. Este componente é amplamente utilizado em projetos de IoT devido à sua versatilidade, baixo custo e alto desempenho. Ele é baseado em um microprocessador dual-core Tensilica Xtensa LX6, que pode operar a uma frequência de até 240 MHz, proporcionando um desempenho robusto para uma ampla gama de aplicações (Espressif Systems, 2021a). Comparativamente, o Arduino Mega 2560, microprocessador popular de nível intermediário, com custo consideravelmente maior que o ESP32, possui velocidade de *clock* de 16 MHz.

Este dispositivo suporta os padrões IEEE 802.11 b/g/n, habilitando a conexão a redes Wi-Fi de 2.4 GHz e permitindo o seu uso até mesmo para aplicações de IoT. O componente também inclui suporte para Bluetooth v4.2 BR/EDR e BLE (Bluetooth Low Energy), permitindo a comunicação com dispositivos Bluetooth de baixa energia, como sensores e atuadores. Estas duas características tornam o dispositivo muito mais versátil, aumentando as opções de interface com o usuário, desenvolvedores e componentes periféricos utilizados na plataforma. Para isso, o ESP32 suporta várias interfaces de comunicação, incluindo UART, SPI, I2C e I2S, o que facilita a integração com uma variedade de periféricos (Espressif Systems, 2021b).

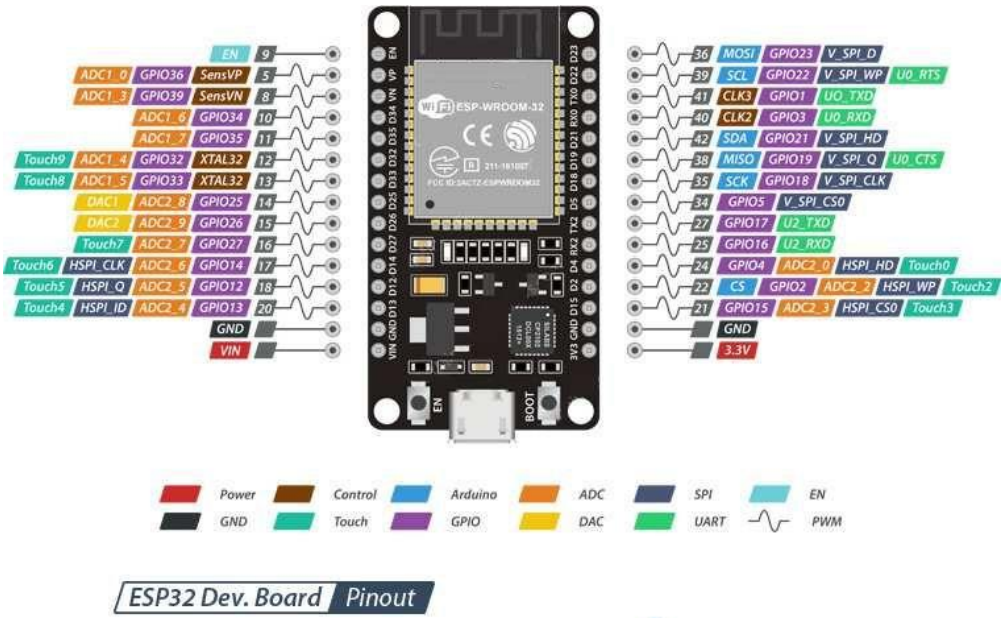
A sua arquitetura *dual-core* permite a execução de tarefas em paralelo, o que aumenta a sua eficiência e capacidade de processamento (Espressif Systems, 2021a). É uma característica imprescindível para este projeto, dado que qualquer

interrupção da geração do sinal reflete na sua frequência e, consequentemente, no som gerado.

Para garantir a agilidade do processamento e um armazenamento robusto, visando também que este projeto seja um sistema personalizável após sua conclusão, o ESP32 utilizado possui 520 KB de SRAM e pode ser expandido com memória flash externa. Comparativamente, o Arduino Mega 2560 possui SRAM de apenas 8 KB. A SRAM é usada para armazenar variáveis, buffers e outros dados temporários que o microcontrolador precisa acessar rapidamente durante a execução de tarefa, sendo crucial para o desempenho geral do sistema, especialmente em aplicações que exijam processamento rápido e eficiente (Espressif Systems, 2021a).

A ampla gama de pinos de entrada e saída de propósito geral (GPIOs), Modulação por largura de pulso (PWMs), DACs, como pode ser visto na Figura 7, também permite a elaboração de sistemas de maior complexidade. Estes pinos podem ser configurados para diversas funções, desde inputs ou outputs binários a funções mais complexas, como leitura de sensores, controle de atuadores e comunicação com outros dispositivos (Espressif Systems, 2021b).

Figura 7 – Configuração de pinos ESP32



Fonte: (Espressif, 2018)

Os conversores analógico-digital (ADC) e digital-analógico (DAC), permitem a interface com esses tipos de sinais (Espressif Systems, 2021a). Para essa aplicação, por exemplo, a função principal será a geração de senoides na faixa audível (20 Hz a

20 kHz) pelo microcontrolador, e ainda há o controle fino do instrumento com *input* analógico.

2.5.2.2 Programação do microcontrolador

O ESP32 é compatível com várias plataformas de desenvolvimento, incluindo o Arduino IDE, PlatformIO e o ESP-IDF (*Espressif IoT Development Framework*), facilitando a programação e a prototipagem. Possui uma vasta comunidade de desenvolvedores e uma ampla gama de bibliotecas disponíveis, que simplificam a implementação de funcionalidades complexas (Espressif Systems, 2021b).

Para este projeto, foram cogitadas as plataformas Arduino IDE e ESP-IDF. O Arduino IDE foi escolhido pela familiaridade ao ambiente e facilidade de uso, porém o ESP-IDF pode oferecer maior controle sobre o dispositivo, seus núcleos, frequência de amostragem, alteração de variáveis do sistema operacional (FreeRTOS) que são automatizadas no Arduino IDE, entre outros, caso sejam aplicadas melhorias futuras.

2.5.2.3 Conclusões sobre o dispositivo

Em resumo, o ESP32 é um componente poderoso e versátil, que oferece todas as funcionalidades necessárias para o desenvolvimento de sistemas complexos e precisos, de projetos em IoT a sistemas embarcados.

No contexto deste trabalho de conclusão de curso, o modelo ESP32-WROOM-32, dual core com 30 pinos, foi utilizado como o microcontrolador principal para o desenvolvimento de um teremim baseado em sensores de ultrassom. Suas capacidades de processamento e conectividade permitem a leitura precisa dos sensores, o processamento digital dos sinais e a comunicação com outros dispositivos, se necessário, sem limitar a frequência da geração do sinal sonoro. Além disso, a facilidade de programação e a disponibilidade de recursos levaram a escolha final do ESP32 para a implementação e prototipagem eficiente do projeto.

2.5.3 Programação

A programação disponibilizada no **APÊNDICE A – Programação completa do ESP32 no ambiente Arduino IDE** contém todos os algoritmos para o funcionamento

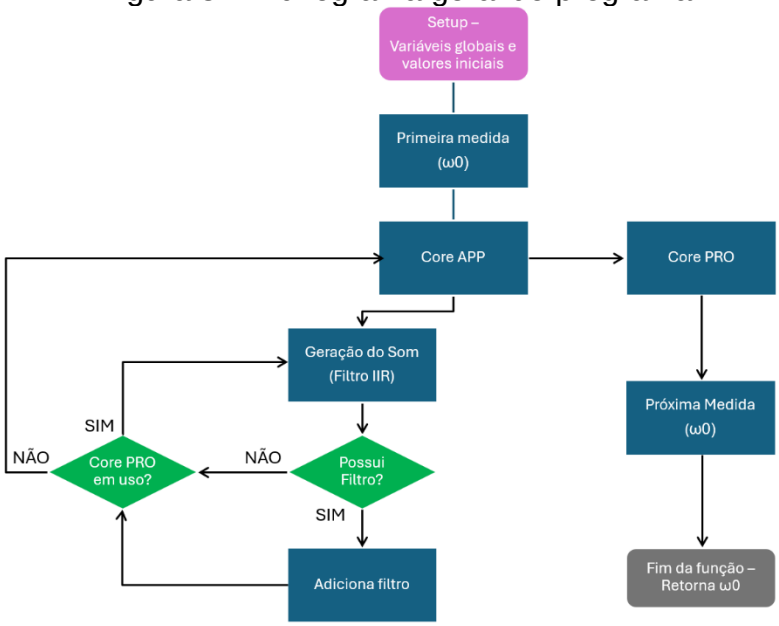
do projeto. Estes estão divididos em blocos, conforme traz o **Diagrama de fluxo do programa**. Cada bloco será explicado individualmente nas próximas seções.

2.5.3.1 Diagrama de fluxo do programa

O fluxo geral do programa idealizado segue o processo da

Figura 8. Não há uma função específica no programa para ativar e desativar filtros, mas foi adicionado ao fluxograma com o intuito de demonstrar a diferença entre o processamento com e sem o filtro.

Figura 8 – Fluxograma geral do programa



Fonte: Elaborado pelo autor.

2.5.3.2 Definição de bibliotecas e variáveis globais

Ponto inicial do programa em que são definidas as bibliotecas que serão utilizadas e variáveis globais, visto que na programação do ESP32 as variáveis possuem escopos que podem limitar o acesso a certas áreas do código. no caso das globais, são as que serão aplicadas em múltiplas funções e por isso podem ser acessadas por todo o código.

2.5.3.3 Setup

Nesta etapa são definidos parâmetros iniciais para o funcionamento do dispositivo. Estes são,

- **Pinos de entrada e saída:** Sensores, potenciômetro, gerador de sinal digital-analógico;
- **Valores da configuração do teremim:** Filtros aplicados, faixa de distâncias selecionada;
- **Características do sistema:** Frequência de amostragem do DAC, intervalo de distâncias de medição e de frequências geradas.
- **Condições iniciais:** Primeiro cálculo de distância para conversão do som, evitando erro por valor nulo.

2.5.3.4 Loop principal

Loop que é executado constantemente no núcleo APP do ESP32, sem necessidade de chamá-lo novamente.

Está responsável pela função de geração do som e faz a chamada do núcleo PRO para atualização da variável da frequência em paralelo a função “Som()”. A geração do som ocorre por meio da função “Som()”, utilizando o valor medido da distância no momento em que ela é iniciada.

2.5.3.5 Loop núcleo PRO

Tem a função de calcular os parâmetros principais e auxiliares para a geração do som pelo oscilador digital, incluindo:

- Medição da distância dos sensores e condicionamento do sinal (“Final_Y”);
- Conversão das medições para frequência em Hertz (f_zero);
- Frequência normalizada (“w_zero”);
- Amplitude do sinal (“amp”);

O paralelismo dos núcleos é feito através da função interna esp_ipc_call(). Ela ativa a tarefa IPC (Inter-Processor Call) presente no núcleo acionado, cessando a função atual e executando a tarefa prioritária enviada para o IPC.

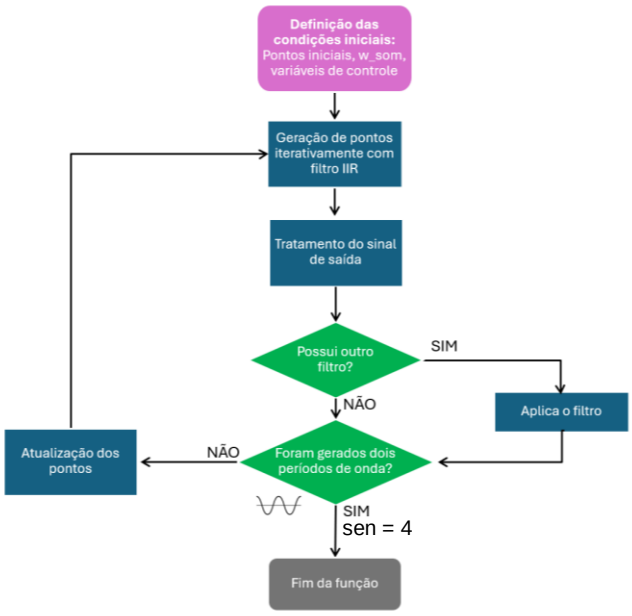
2.5.3.6 Função Som()

A função Som() é iniciada criando três variáveis do oscilador digital IIR no formato de equação de diferenças, chamados “Output”, “Output2” e “Output1”, correspondentes a y[n], y[n-1] e y[n-2], respectivamente.

Também é definida a frequência “w_som”, que receberá a última frequência w_zero calculada pelo núcleo PRO. O programa está organizado dessa maneira para não ocorrer a substituição da frequência da onda durante um ciclo de geração, algo que pode ocorrer devido ao paralelismo das atividades. Além disso, é definida a função “saída”, que traz o valor tratado para ser enviado ao pino DAC e variáveis de controle para a geração do sinal.

Por ser a função mais extensa do programa, o fluxograma específico na Figura 9 ilustra o seu funcionamento.

Figura 9 – Fluxograma da geração do som iterativamente com filtro IIR



Fonte: Elaborado pelo autor.

As condições iniciais de y[n], y[n-1] e y[n-2] para w_som são definidas e entramos no *loop* principal da função. Dentro da função *while* com a variável de

controle “sen”, temos a geração do restante da onda, que consiste na redefinição de y[n], y[n-1] e y[n-2] iterativamente. A condição de parada sucede essas definições, com o seguinte comportamento:

No início, temos s = sen = 0, sendo que Output começa em um pico da senoide, ou em 90º de um ciclo pela visão da fase da onda, com amplitude 1, conforme definido nas condições iniciais. Consequentemente, estamos em um ponto da senoide com derivada negativa. No momento em que o valor y[n-1] (Output2) se torna maior que y[n-2] (Output1), constata-se que a derivada se tornou positiva, o que significa que chegamos a um vale da senoide. Neste ponto, às variáveis de controle são atualizadas, com “sen” sendo incrementado em 1, contabilizando uma alteração no sinal da derivada, e “s” definido para 1, sinalizando que estamos em uma região de derivada positiva da função.

O restante do controle é feito seguindo esse raciocínio, buscando pela alteração da relação entre os pontos y[n-2] e y[n-1] para verificar a mudança do sinal da derivada da função. Por escolha do autor após testes práticos.

Para a saída no DAC, é feito um último estágio de condicionamento para enviar informações válidas à saída. O conversor digital-analógico do ESP32 possui 8 bits, de forma que recebe números de 0 a $2^8 - 1$, ou seja, 0 a 255. A função gera uma onda senoidal pura, com amplitude 2 centralizada em 0, com valores entre o intervalo [-1,1]. Para ajustá-lo a saída, é feita a rápida conversão com a equação (7)

$$saída = Output * amp + amp \qquad (7)$$

Sendo:

saída = valor final aplicado a saída DAC;

amp = valor da amplificação do sinal de saída;

Output = Saída do filtro IIR.

De forma a multiplicar a amplitude pelo volume obtido do sensor correspondente, somando o valor da amplitude para transladar todo o sinal para valores positivos. O programa está com a versão simplificada da equação (8),

$$saída = amp * (Output + 1) \qquad (8)$$

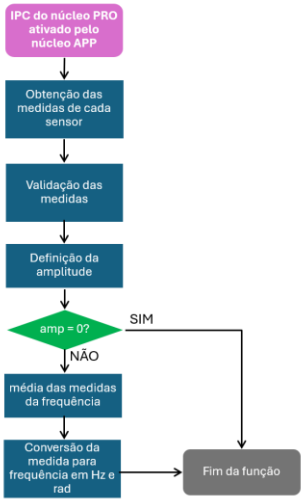
Tecnicamente, está sendo utilizado o fator (Output + 1.1), como uma rede de segurança para garantir que a saída receberá apenas valores positivos, porém esse ponto pode ser ajustado conforme visto necessário. Como último estágio, ainda são verificados com sequências de condições *if* se o valor está no intervalo de 0 a 255, ajustando a saída dependendo do caso simulando um efeito de *clipping*. Matematicamente, é impossível que a saída configure um valor fora desse estágio com os valores utilizados, porém, visando a adição de outros filtros que podem alterar o valor da saída e outras futuras modificações, este é um passo saudável no programa para garantir a integridade da saída e evitar inversões do sinal pelo DAC.

No momento em que são gerados dois ciclos de onda, ou seja, sen = 4, é terminada a geração da onda, são resetadas as variáveis de controle e a função verifica se o núcleo PRO ainda está medindo o próximo valor. Nesse caso, a função faz uma chamada recursiva, gerando mais dois ciclos de onda. Quando é detectado que o outro núcleo finalizou a próxima medição, o programa é retornado ao loop principal do núcleo APP, solicitando uma nova medida de w_zero e recebendo o último valor calculado.

2.5.3.7 Cálculo da frequência

Para o cálculo da frequência w_zero, são utilizadas as 4 funções a seguir. O fluxograma da Figura 10 a seguir simplifica o processo.

Figura 10 – Fluxograma do cálculo da frequência – Núcleo PRO



Fonte: Elaborado pelo autor.

2.5.3.8 Raw_cap() - Leitura crua

Esta função é responsável pelo funcionamento dos sensores ultrassônicos. Ela utiliza um loop “for”, limitado pela variável “n” que traz a quantidade de sensores utilizados no dispositivo. Ela então realiza o algoritmo de medição necessário para cada ultrassom já abordado e salva o valor no vetor “Raw” com o índice respectivo a cada sensor.

Especificamente, é aplicado um delay no final do cálculo de 15 milissegundos, que pode ser ajustado dependendo da função. Com este valor para essa função em específico foi encontrada uma responsividade admissível do instrumento. É aconselhado pelo *datasheet* o tempo de 60 ms para um ciclo total, evitando a interferência do sinal do “trigger” no sinal do “echo”.

Por fim, o valor do tempo do impulso é convertido conforme equação, ajustando os valores conforme grandezas que são trabalhadas no ESP32. É importante verificar que tempo do impulso é recebido em microssegundos, sendo necessária a conversão para segundos no processo. De forma similar, estamos recebendo o valor em centímetros, sendo necessária a conversão para metros. Levando esses dois pontos em conta, chegamos à relação (9).

$$d = \frac{\text{impulso}}{58{,}3} \qquad (9)$$

2.5.3.9 Função Position_calc – Cálculo das posições

A função valida as medições de cada sensor. Neste caso são 3 no total sensores, porém a função é generalizada para receber n - 1 sensores, considerando sempre que o primeiro (índice 0) é utilizado para o cálculo da amplitude, sendo esse valor ignorado.

A partir do sensor de índice 1, é verificado se ocorreu alguma medição, ou seja, se o valor da medida é maior que zero. Se sim, um vetor “test” é definido como 1 no índice correspondente para contabilizar essa medição no valor final. De outra forma, se o valor é zero, o vetor “test” é marcado como zero no índice do sensor.

2.5.3.10 Fuse_data - Fusão dos dados

Por fim, a função Fuse_data combina todos os valores dos sensores e designa um valor final correspondente às “hastes” da frequência e da amplitude.

Para tal, primeiramente é definido o valor da amplitude, com a simples relação linear $f(x) = \frac{x}{10}$. O valor é então limitado a 4, para impedir a saturação da saída. Com isso, a área de atuação do sensor de amplitude é efetivamente de 2 a 40 cm. O valor limitante (4) foi escolhido conforme testes práticos com o amplificador utilizado, podendo ser alterado ou até adicionado como uma variável de controle de volume do instrumento utilizando um potenciômetro, por exemplo.

Para a frequência, são somados todos os valores válidos e é feita a média aritmética desses valores, utilizando o vetor “test” para diferenciar valores inválidos.

No caso da medição do amplificador ser inválida, normalmente atingida no caso em que a mão é posicionada a menos de 2 cm do sensor, a amplitude é definida automaticamente em 0. Isto foi feito propositalmente para servir como um meio de parar o som durante a operação, em momentos em que um intervalo de silêncio entre as notas seja desejado, e é similar a maneira que o teremim clássico funciona.

Caso a amplitude seja válida, o programa é direcionado ao último passo da função. Inicialmente é verificado se existe um valor de Final_Y válido. Caso positivo, ocorre o condicionamento da variável “Final_Y” que contém a distância utilizada para o cálculo da frequência. Esta fase mapeia o dado em valores normalizados para o cálculo da frequência. Neste caso, está sendo utilizada a faixa de 2 a 40 cm como valores normalizados. Para o mapeamento, é utilizado o intervalo de distâncias selecionado no *setup* através do potenciômetro e os valores normalizados.

Como exemplo, se o potenciômetro for posicionado no ponto em que a distância máxima seja 35 e o valor medido for 29, este número será convertido proporcionalmente para o intervalo de 2 a 40, ou seja, teremos um $Final_Y \approx 33{,}14$.

Após validação da medida, o valor é convertido para a frequência w_zero através da função “converter_exp”.

2.5.3.11 Converter_exp - Conversão da distância por relação exponencial

Foram cogitados variados métodos para conversão da medida para frequência, sendo no início aplicado um conversor linear. Após a definição das bases do projeto, esta conversão foi alterada para exponencial, o que corresponde mais fielmente ao comportamento do som em relação às frequências.

A maneira padrão de organizar notas musicais no Ocidente é conforme o Sistema Temperado, que consiste no sistema de afinação responsável pela divisão do intervalo de uma oitava em doze semitons similares. A escala temperada, também conhecida como escala cromática, é composta pelas notas Dó, Dó#, Ré, Ré#, Mi, Fá, Fá#, Sol, Sol#, Lá, Lá#, Si. Uma relação importante das frequências e às notas é que, a cada ciclo, às notas dobram de frequência, dando emergência a uma razão exponencial. Como um exemplo prático, tomando a nota Dó como exemplo e suas oitavas, faz-se a comparação presente na Tabela 1.

Tabela 1 – Comparação de frequências no Sistema Temperado

Notas	Oitava anterior	Oitava posterior	ΔHz
C2 e C3	66	132	66
C6 e C7	1056	2112	1056

Fonte: Elaborado pelo autor

Conforme pode ser observado, a diferença entre a frequência das notas não é um intervalo fixo, aumentando exponencialmente de acordo com o aumento da frequência. Desta maneira, utilizar uma relação exponencial para converter a distância em frequência distribui uniformemente as notas ao longo da distância.

Explicada a escolha, a função utiliza uma equação exponencial negativa para converter a entrada em uma saída em Hz: Final_Y → w_zero. A função negativa foi utilizada para posicionar às frequências mais altas nas distâncias menores, simulando o mesmo funcionamento de um teremim com mecanismo capacitivo.

A equação utilizada foi no formato $f(x) = K^{x+b}$. O passo-a-passo para encontrá-la está disponível na seção **3.3**. Considerando um intervalo de frequências de 100 Hz a 2600 Hz, valores típicos do teremim, foi obtida a função $f_{zero}(Final_Y) = 0{,}918^{Final_Y-93{,}9}$, foi obtida uma curva que corresponde geralmente ao comportamento esperado dentro do intervalo de interesse.

Esta equação é definida na função e aplicada ao último valor calculado, resultando em uma frequência em Hz. A frequência ainda precisa ser normalizada para radianos (rad) como ω_{zero}, de forma a realizar o processamento correto da nota na saída do instrumento. Tal frequência é obtida utilizando teoria de processamento de sinais, onde a frequência normalizada ω_0 é obtida pela equação (10):

$$\omega_0 = \frac{2\pi f_0}{f_s} \qquad (10)$$

Onde:

ω_0 = Frequência normalizada (rad)

f_0 = Frequência analógico do sinal (Hz)

f_s = Frequência de amostragem do sistema (Hz)

Para tal, foi necessário aplicar métodos para calcular a amostragem do sistema e obter um valor preciso. Neste caso, temos uma amostragem de $f_s \approx 165\ \mathrm{kHz}$. Com isso, já conseguimos verificar que o intervalo de 0 permanecerá entre ω_{zero} = [0.0038,0.099].

Fazendo uma análise adicional, conforme o teorema de Nyquist, uma frequência pode ser recuperada corretamente de um sistema digital desde que esta seja menor que a metade da frequência de amostragem. Em sistemas de som, é comum desenvolver sistemas com a frequência de amostragem de 44,1 kHz, que já cobre todo o espectro audível de 20 a 20 kHz e traz ainda uma margem de erro para todo o sistema de 10,25%, prevendo oscilações.

Com uma amostragem muito superior de $f_s \triangleq$ 165 kHz, o sistema está posicionado em um bom ponto para adição de outras funções para processamentos mais complexos do sinal. Isto porque, com a adição de processamentos extras, como outros filtros para modificar o sinal, o processamento levaria mais tempo o que reduziria a frequência de amostragem, como será visto na adição do filtro Phaser. Quanto mais rápido o processamento da senoide pura, mais espaço temos para complementar o sinal com outros efeitos sem que a velocidade do processamento fique abaixo do limite previsto pelo teorema de Nyquist.

2.5.3.12 aplicarPhaser - Filtro Phaser

Como último passo do projeto, foi projetado um filtro Phaser de dois estágios para aplicação ao sistema.

O efeito Phaser é a combinação do sinal original com o sinal filtrado por um filtro Passa-tudo, o qual apenas desloca a fase do sinal (Carson et al., 2023). A função de transferência de um filtro passa-tudo de 1ª ordem H(z), com |a| < 1, pode ser descrita como na equação (11),

$$H(z) = \frac{z^{-1} - a}{1 + az^{-1}} \qquad (11)$$

Observando a equação, temos que o polo, valor em que H(z) → ∞, é obtido quando o denominador $1 + az^{-1} = 0$. Isto ocorre para z = -a. Logo, levando em conta que, para a estabilidade de um sistema no domínio z, todos os polos devem estar localizados dentro do círculo unitário (|z|<1), neste caso é obtido que |a| < 1.

Sobre o zero da função, ponto onde H(z) = 0, temos que este ocorre em $z = \frac{1}{a}$, e é a característica responsável pela variação da fase quando o filtro é aplicado a entrada X(z).

Por ser um filtro passa-tudo, temos que a amplitude ao longo do espectro é de |H(z)| = 1. A única alteração que teremos ao sinal é a mudança da fase, controlada pelo coeficiente a.

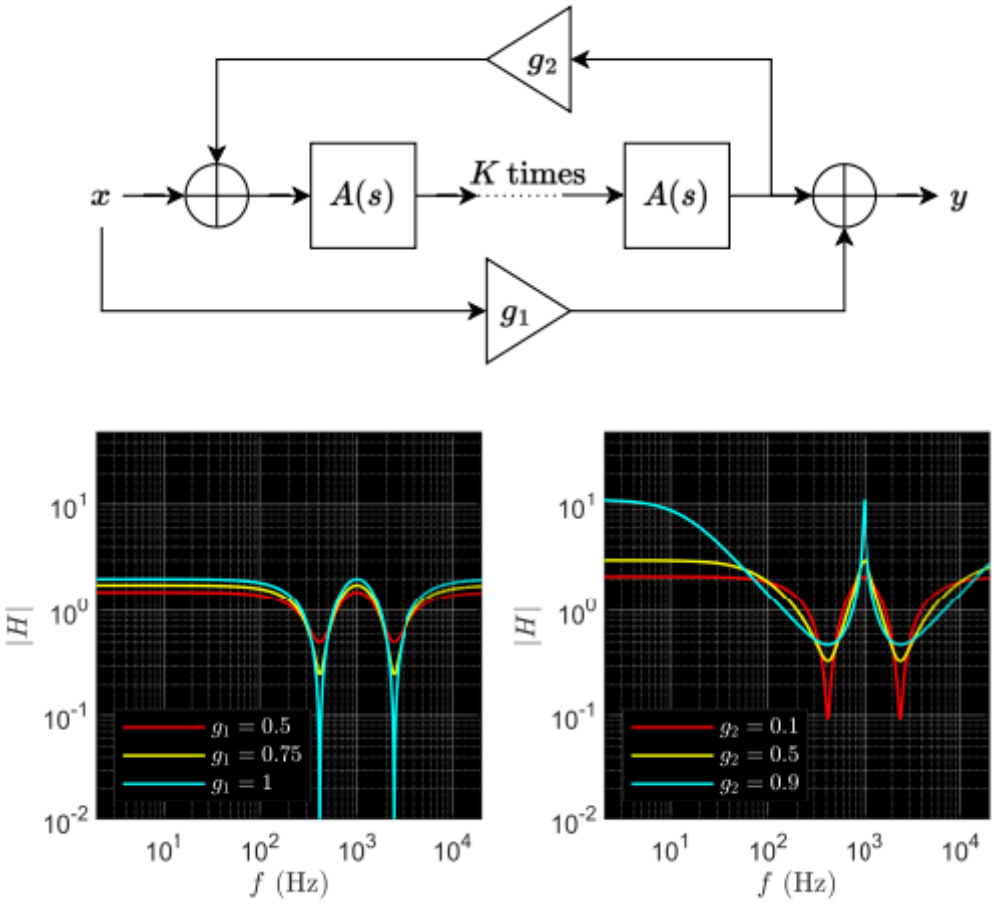
Esta equação pode ser descrita com seu formato de equação de diferenças, que foi obtido como traz a equação (12),

$$y[n] = -ay[n-1] + x[n-1] - ax[n] \qquad (12)$$

Sua complexidade é gerada a partir da combinação do sinal filtrado com o sinal original, causando “nós” no sinal resultante por conta da interferência destrutiva do sinal original com o sinal defasado. Esta complexidade pode ser aumentada com múltiplos estágios de filtro Passa-tudo, algo feito comumente em filtros para guitarras. Aplicando o mesmo filtro sequencialmente, obtemos o mesmo resultado do que utilizando a equação de segunda ordem, portanto essa não será calculada.

Em seguida, observa-se o estudo feito por Carson et. al (2023), onde foi aplicado um filtro de segunda ordem sobre condições de loop aberto e loop fechado, e pode-se observar os nós sendo criados no espectro de frequência para dois pontos específicos, nesse caso que está sendo utilizado um filtro de segunda ordem.

Figura 11 – Simulação de um filtro Phaser com dois estágios



Fonte: Adaptado de (Carson et. al, 2023)
Legenda: Diagrama de um sistema com K filtros passa-todas em cima, resposta da magnitude para H(s) sob condições de loop aberto na esquerda, para diferentes valores de g1, e sob condições de loop fechado na esquerda, para g1 = 1, e diferentes valores de g2, como indicado. ωb = 2π · 1000 rad/s.

Para adicionar um sentido de “movimento” ao som, é utilizado um Low Frequency Oscilator (LFO), que consiste em uma senoide de frequência baixa responsável por variar o coeficiente a, alternando a defasagem ao longo do tempo. Desta forma, chegamos ao formato final visível no programa. Para esse caso, será utilizado um LFO de 0,5 Hz, que irá oscilar entre φ = [1.3694,1.7722] radianos, valores encontrados experimentalmente onde o filtro se comporta bem.

A escolha dos limites φ também se deram por dois fatores adicionais. Será visto no algoritmo que, para tais valores, temos $|a| = |\cos(\varphi)| < 1$, mantendo a estabilidade do sistema. E a variação de $|\cos(\varphi)|$ foi levada em conta, sendo que os extremos possuem aproximadamente o mesmo módulo, evitando uma descontinuidade no sistema.

3 RESULTADOS

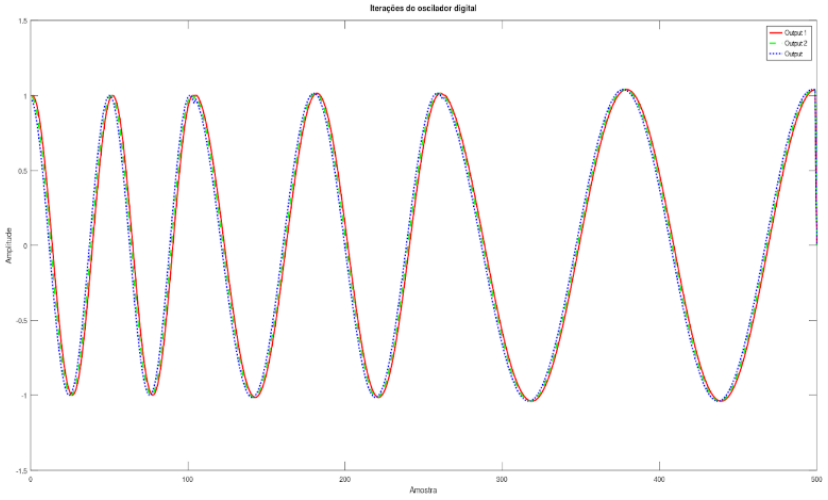
Esta seção trará os resultados dos testes realizados para concepção do projeto e os resultados finais do sistema completo.

3.1 Testes do oscilador digital - Filtro IIR

Foi aplicado o algoritmo do oscilador no GNU Octave, conforme APÊNDICE B – Algoritmo em Octave GNU para geração do sinal teórico, a fim de comprovar seu funcionamento. A equação de diferenças do filtro IIR foi utilizada conforme a equação (4), que já foi abordada.

O algoritmo foi adaptado ao programa para demonstrar diferentes frequências sendo geradas, conforme pode ser visto na Figura 12.

Figura 12 – Pontos calculados do oscilador IIR com Simulação GNU Octave



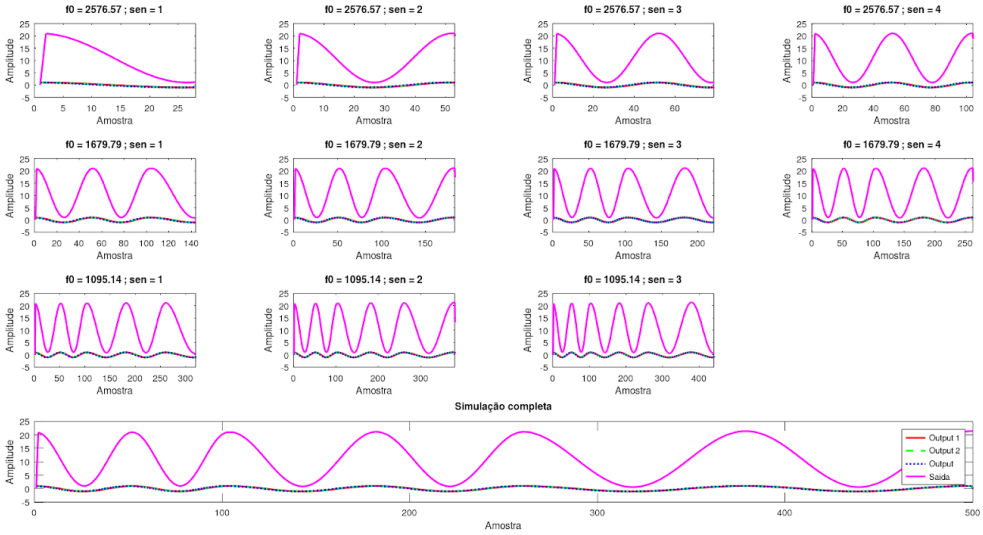
Fonte: Elaborado pelo autor.

A partir desse resultado, é possível concluir que o algoritmo está produzindo a onda conforme o projetado, gerando dois períodos completos com a frequência w_som. É importante salientar que a curva em questão segue o comportamento da função cosseno, sendo f(0) = 1. Após isso, a onda transiciona de forma suave para a próxima frequência, o que demonstra um bom comportamento para ser utilizado no sistema do teremim digital.

Foram também obtidas as ondas auxiliares, a fim de visualizar todos os coeficientes da equação.

Com os valores do oscilador obtidos, foi adicionado ao algoritmo o estágio de condicionamento da saída. A Figura 13 traz os valores de *output* gerados (Output1, Output2 e Output), a saída corrigida para o DAC (Saída) e as etapas da geração conforme descrito anteriormente, gerando dois períodos de onda e atualizando a frequência.

Figura 13 - Resultado do oscilador IIR e saída com Simulação GNU Octave



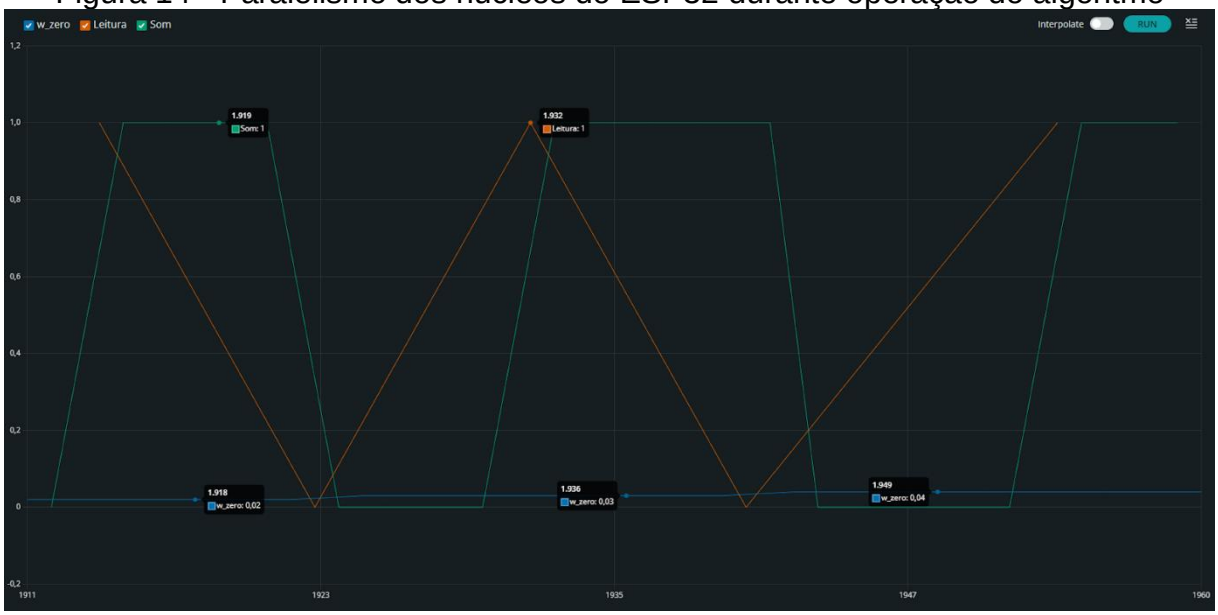
Fonte: Elaborado pelo autor.

3.2 Testes do paralelismo dos núcleos no ESP32

Foi concebido um algoritmo para avaliar o comportamento do IPC, para garantir que ele se comporte da maneira necessária para aplicação desse trabalho.

Foi trazido na Figura 14 o retorno do ESP32 durante a execução do algoritmo. Neste algoritmo, foi referenciada uma variável a cada núcleo, de forma que este está executando uma tarefa enquanto o valor da variável correspondente é 1. Um núcleo está responsável pela geração do Som enquanto o outro pela leitura dos sensores, assim como já foi abordado. Da imagem, é possível verificar que o núcleo responsável pela geração do Som apenas suspende para obter o valor novo da leitura, assim que este é calculado.

Figura 14 - Paralelismo dos núcleos do ESP32 durante operação do algoritmo



Fonte: Elaborado pelo autor.

É importante destacar que o eixo X não representa tempo, apenas a sequência de sinalizações de cada núcleo. Da maneira que foi definida essa demonstração, o núcleo APP envia mensagens repetidas, igual a quantidade de iterações que foram feitas aguardando a próxima leitura, ao completar uma onda. Com isso, pode parecer que ele permanece ocioso aproximadamente pelo mesmo tempo que gera, porém não é o que estamos verificando neste escopo.

Na verdade, temos apenas o tempo ocioso para realizar a chamada da próxima leitura do núcleo PRO, que será feita em paralelo a geração do oscilador utilizando a medição mais recente, como pode ser visualizado na figura pela linha w_zero.

A responsividade do instrumento depende principalmente neste período entre a medição pelo núcleo PRO e a efetiva utilização dessa medição na geração do sinal sonoro. O maior ponto de atenção é com as frequências baixas, as quais fisicamente exigem um tempo maior para serem geradas (quanto menor a frequência, consequentemente maior o período de uma onda) e podem causar atraso entre as leituras.

3.3 Cálculo da função de conversão da distância para frequência

Para definição da equação, foi tomado inicialmente $f(x) = a^{x-b}$, sendo que a equação deve passar pelos pontos (2,2600) e (40,100).

Para tal, foi abordado com um cálculo direto para obter os coeficientes a e b que aproximassem a curva desse comportamento.

Primeiramente, foram substituídos os pontos (2,2600) e (40,100), onde desejamos que a curva passe, na equação $f(x) = a^{x-b}$:

Para (2,2600): $2600 = a^{2-b}$

Para (40,100): $100 = a^{40-b}$

Em seguida, foi resolvido o sistema de equações, dividindo as duas equações para eliminar b:

$$\frac{2600}{100} = \frac{a^{(2-b)}}{a^{(40-b)}}$$

Simplificando:

$$26 = a^{(2-b)-(40-b)} \rightarrow 26 = a^{2-b-40+b} \therefore$$
$$26 = a^{-38}$$

Com isto, é possível determinar a, tomando o logaritmo da equação de ambos os lados:

$$-38\ log(a) = log(26)$$
$$log\ (a) = -\frac{log(26)}{38}$$

Tendo que $log(26) \approx 1{,}415$:

$$log(a) = -\frac{1{,}415}{38} \approx -0{,}0372$$
$$a = 10^{-0{,}0372}$$
$$a \approx 0{,}918$$

Em seguida, é determinado b, utilizando a = 0,918 na equação. 2600 = a^{2-b}:

$$2600 = 0.916^{2-b}$$

Tomamos o logaritmo da equação acima:

$$log(2600) = (2 - b) \cdot log(0.918)$$

Sendo que $log(2600) \approx 3{,}415$; $log(0.918) \approx -0{,}0372$:

$$3.415 = (2 - b) \cdot (-0{,}0372)$$
$$2 - b = \frac{3{,}415}{-0{,}0372}$$
$$b = \frac{3{,}415}{0{,}0372} + 2$$
$$b \approx 93{,}8$$

Ou seja, a equação (13) final é dada por:

$$f(x) = 0{,}918^{x-93{,}8} \qquad (13)$$

3.4 Obtenção da frequência de amostragem do sistema (f_s)

Um ponto essencial para aplicação do sistema é a obtenção de sua frequência de amostragem para reproduzir o sinal digital com sua frequência planejada (Oppenheim, 1989). Para tal, o algoritmo foi adaptado, conforme **APÊNDICE C – ADAPTAÇÃO PARA AQUISIÇÃO DA AMOSTRAGEM DO DAC**, com variáveis para obtenção do tempo na entrada da função som e no final da mesma. Também foi adicionado um contador de pontos para contabilizar quantas amostras foram geradas nesse período.

Com isto, foi possível obter uma média do desvio da frequência de amostragem teórica e prática de ΔFs = -683,56 Hz e erro percentual %ΔFs = -0,43%. Quanto ao sinal reproduzido, observou-se um desvio médio da frequência Δf_{zero}= -9,80 Hz e erro percentual % Δf_{zero} = -0,43%. Os dados estão dispostos em sua integra no **APÊNDICE D – RESULTADOS DA AQUISIÇÃO DA AMOSTRAGEM**, na Tabela 2 e Tabela 3.

3.5 Análise do sinal gerado pelo sistema

Por fim, foram obtidos em osciloscópio os sinais de onda gerados pelo sistema, para ser possível realizar a conclusão do nível de sucesso do sistema.

Para tal estudo, serão obtidos sinais com a frequência máxima (2600 Hz), mínima (100 Hz) e uma intermediária (1000 Hz). Além disso, o teste será feito com ajustes diferentes do potenciômetro, para verificar sua consistência.

O potenciômetro pode ser ajustado para variar a distância máxima do teremim correspondente à frequência de 2600 Hz. Esta variação, consequentemente, altera as demais relações do instrumento, similar a afinação de um instrumento de cordas com o aumento ou diminuição da tensão da corda. Foram obtidos valores para 3 ajustes ou “afinações” diferentes do potenciômetro, que foram escolhidos conforme os seguintes critérios:

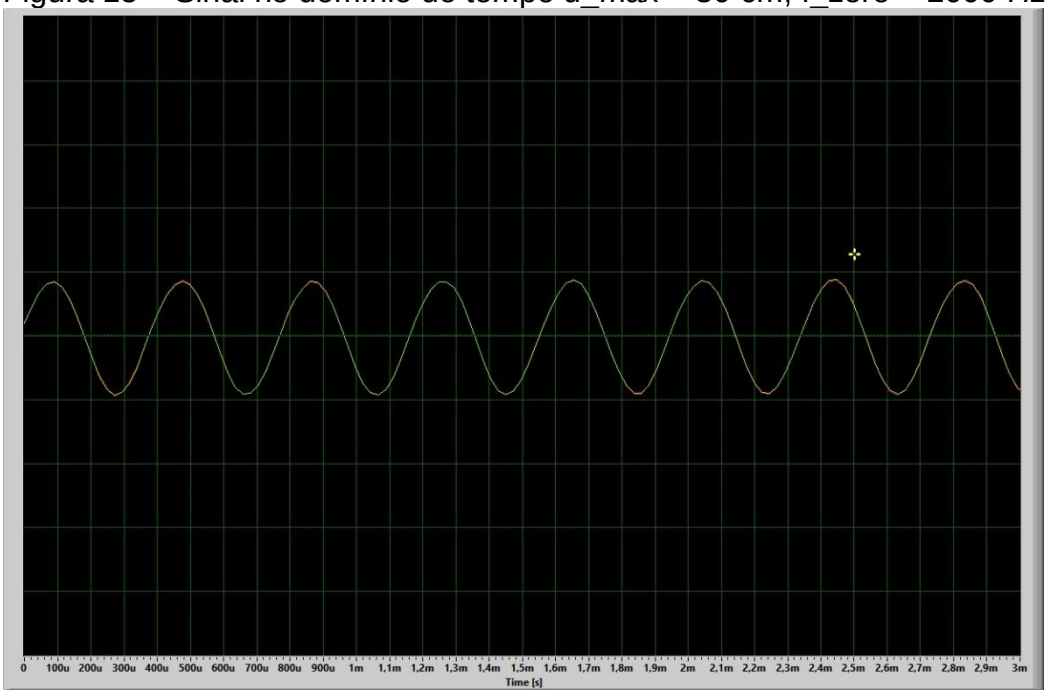
- 30 (trinta) centímetros, a distância mínima que pode ser ajustada no aparelho através do potenciômetro para a frequência máxima (o valor mínimo pode ser alterado no programa).
- Similarmente, 50 (cinquenta) centímetros é o ajuste máximo (pode ser alterado no programa).
- Por último foi movimentado o potenciômetro e feitas as medidas com um ajuste aleatório, neste caso com o valor máximo de 39,79 centímetros.

Por limitações da instrumentação para obter os sinais, está sendo filtrado um ruído inerente do equipamento em 60 Hz, exceto no exemplo que demonstra seu impacto, para apresentação dos dados a seguir.

3.5.1 2600 Hz

3.5.1.1 Distância máxima: 30 cm

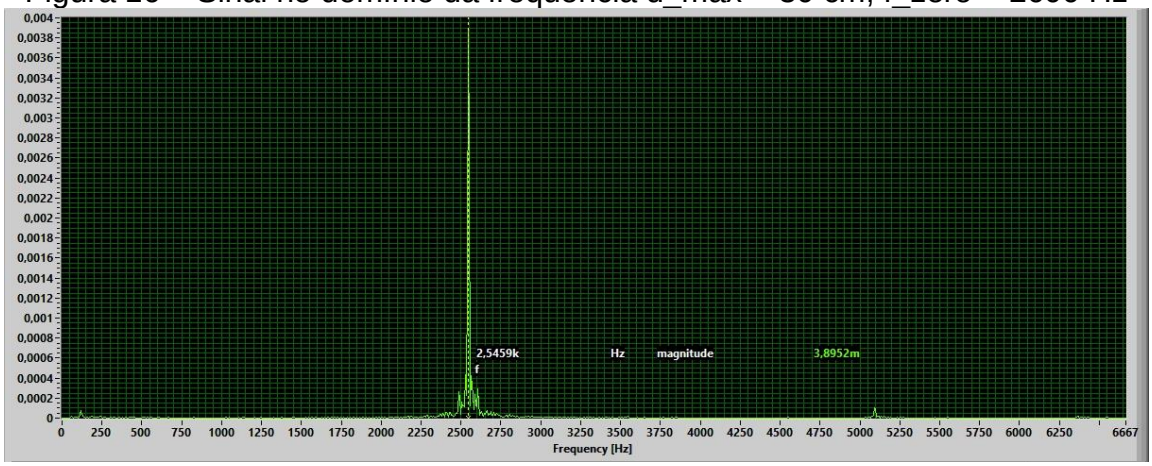
Figura 15 – Sinal no domínio do tempo d_max = 30 cm; f_zero = 2600 Hz



Fonte: Elaborado pelo autor.

Pode-se observar o sinal gerado sendo bem próximo a uma senoide pura.

Figura 16 – Sinal no domínio da frequência d_max = 30 cm; f_zero = 2600 Hz

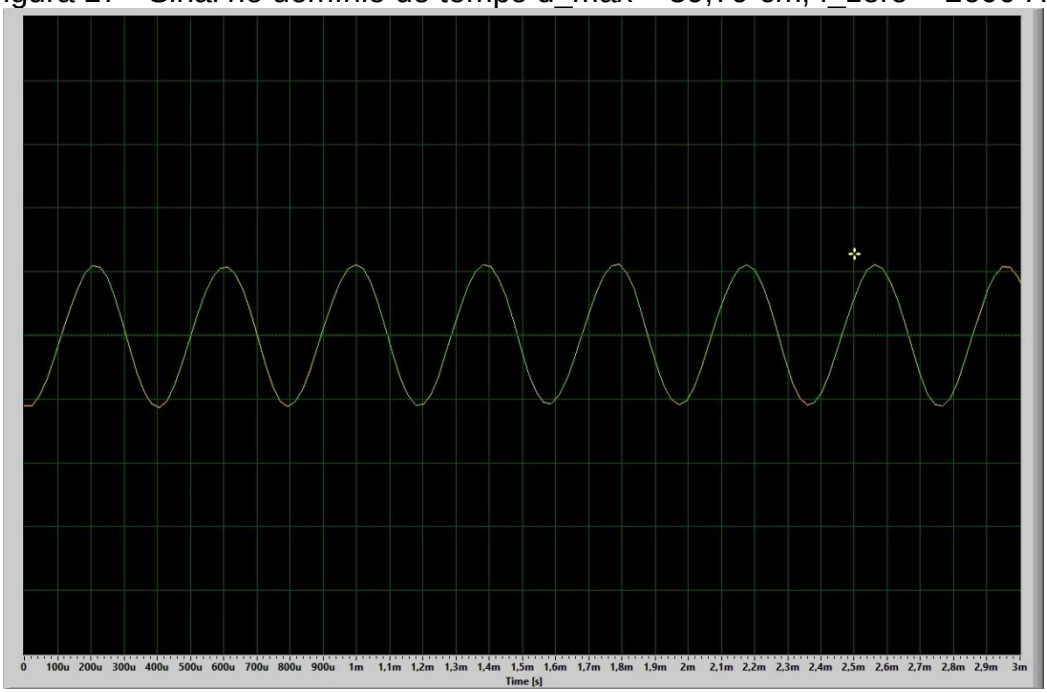


Fonte: Elaborado pelo autor.

Pelo espectro de frequências, conclui-se que o sinal realmente obteve a frequência projetada para ele.

3.5.1.2 Distância máxima: 39,79 cm

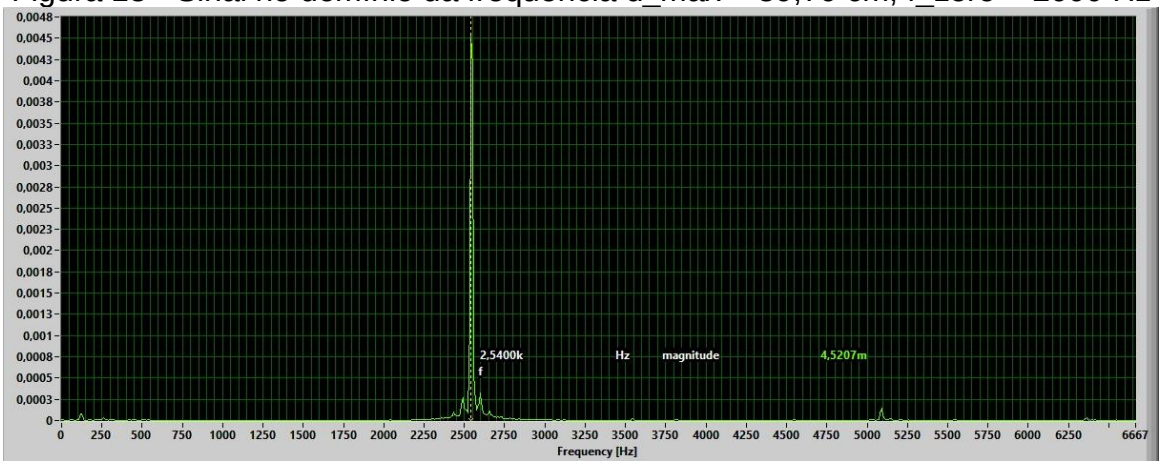
Figura 17 - Sinal no domínio do tempo d_max = 39,79 cm; f_zero = 2600 Hz



Fonte: Elaborado pelo autor.

O sinal mantém o mesmo comportamento mesmo com o ajuste do potenciômetro.

Figura 18 - Sinal no domínio da frequência d_max = 39,79 cm; f_zero = 2600 Hz

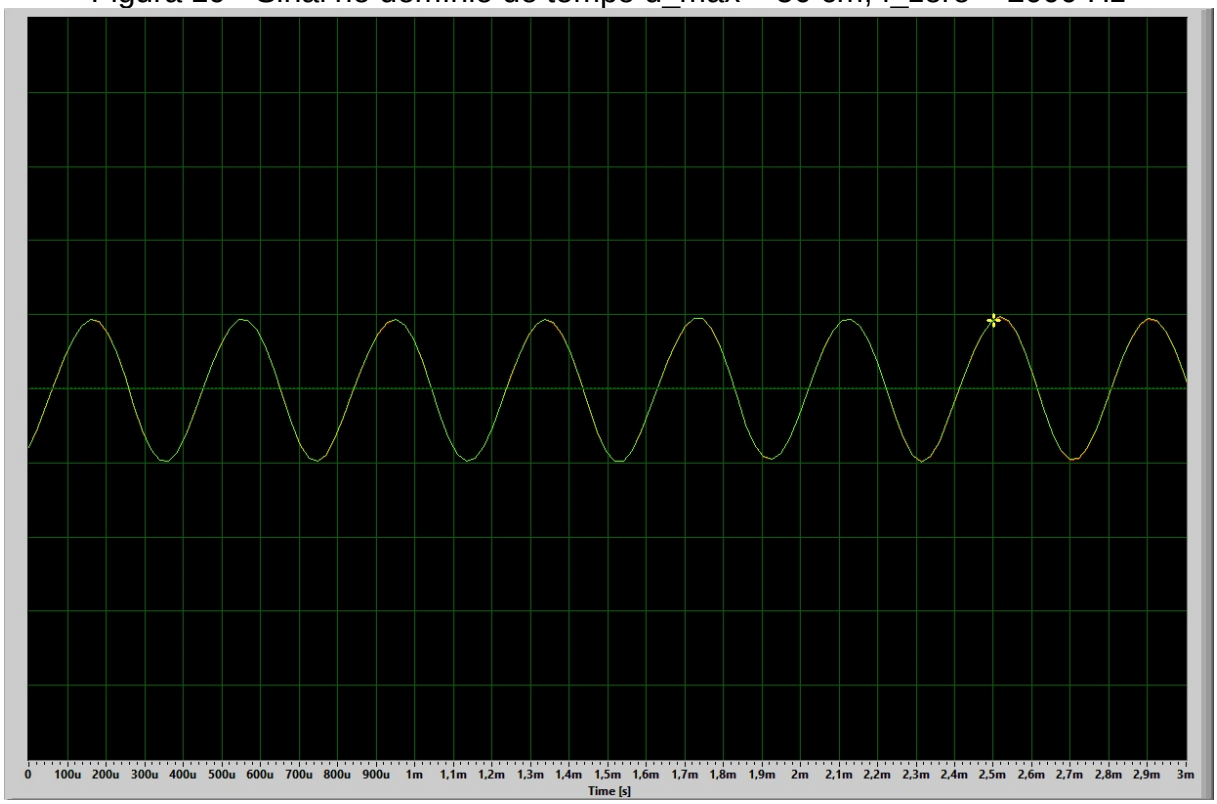


Fonte: Elaborado pelo autor.

A frequência também é preservada.

3.5.1.3 Distância máxima: 50 cm

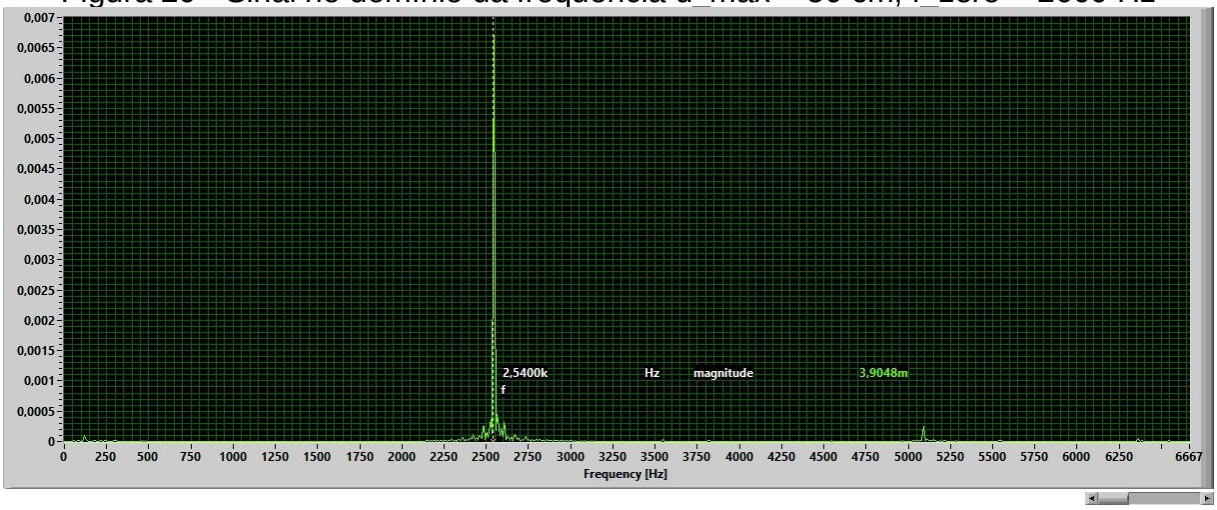
Figura 19 - Sinal no domínio do tempo d_max = 50 cm; f_zero = 2600 Hz



Fonte: Elaborado pelo autor.

Foi verificada a consistência nos três casos para a frequência de 2,6 kHz.

Figura 20 - Sinal no domínio da frequência d_max = 50 cm; f_zero = 2600 Hz



Fonte: Elaborado pelo autor.

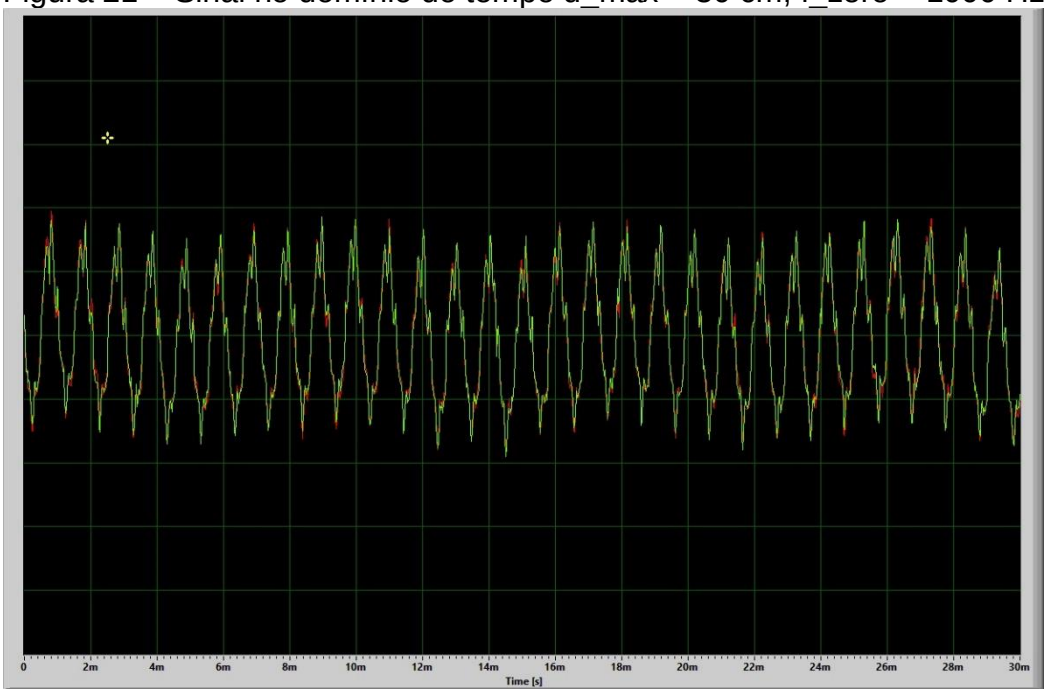
Da mesma forma, a frequência se manteve.

3.5.2 1000 Hz

A seguir os dados obtidos seguindo mesma metodologia para o caso de 2600Hz,

3.5.2.1 Distância máxima: 30 cm

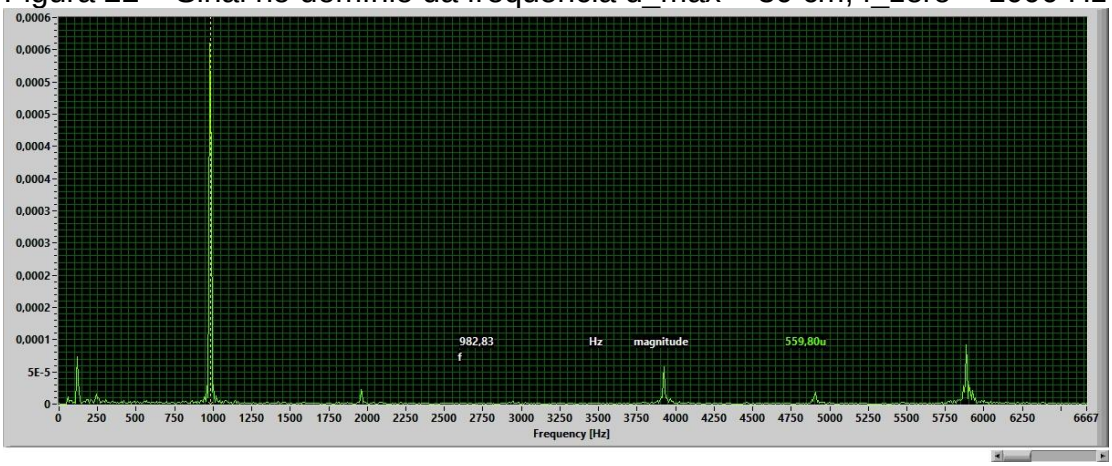
Figura 21 – Sinal no domínio do tempo d_max = 30 cm; f_zero = 1000 Hz



Fonte: Elaborado pelo autor.

Neste caso, observa-se um sinal menos “limpo”, com maiores harmônicas.

Figura 22 – Sinal no domínio da frequência d_max = 30 cm; f_zero = 1000 Hz

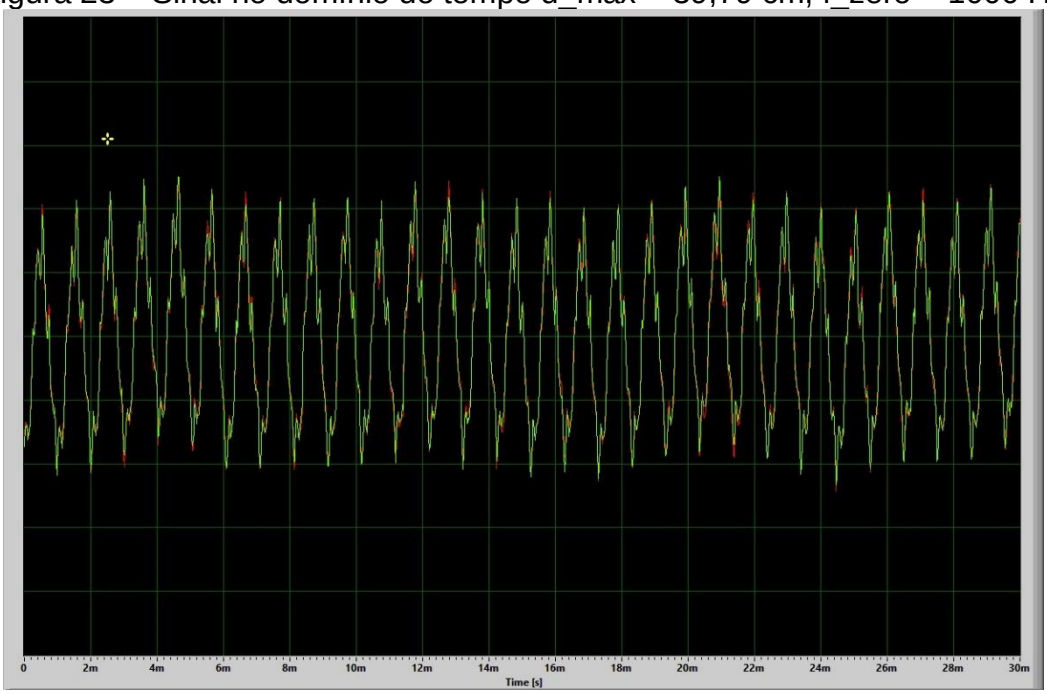


Fonte: Elaborado pelo autor.

A frequência principal do sinal se mantem bem próxima ao projetado, porém com presença de harmônicas de graus superiores. O comportamento se repete similar ao caso de 1000 Hz para outros ajustes de potenciômetro.

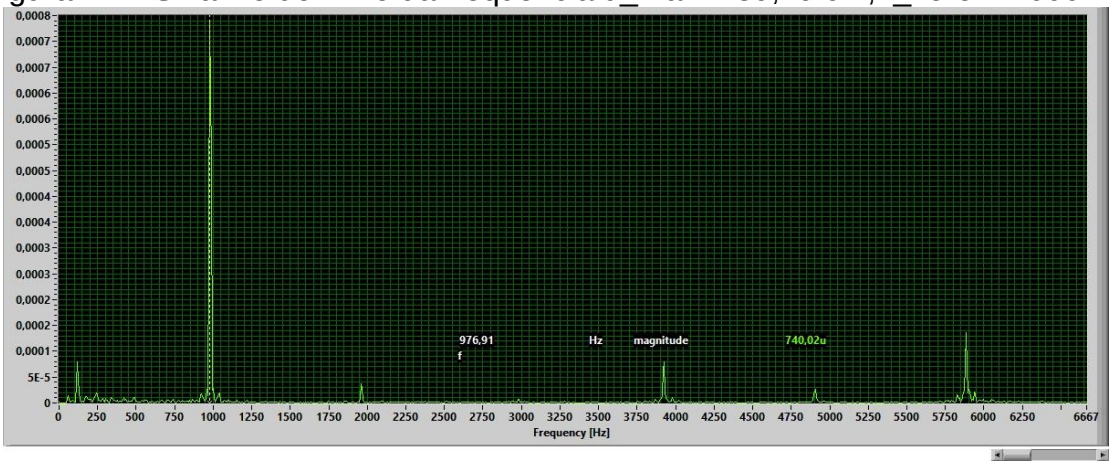
3.5.2.2 Distância máxima: 39,79 cm

Figura 23 – Sinal no domínio do tempo d_max = 39,79 cm; f_zero = 1000 Hz



Fonte: Elaborado pelo autor.

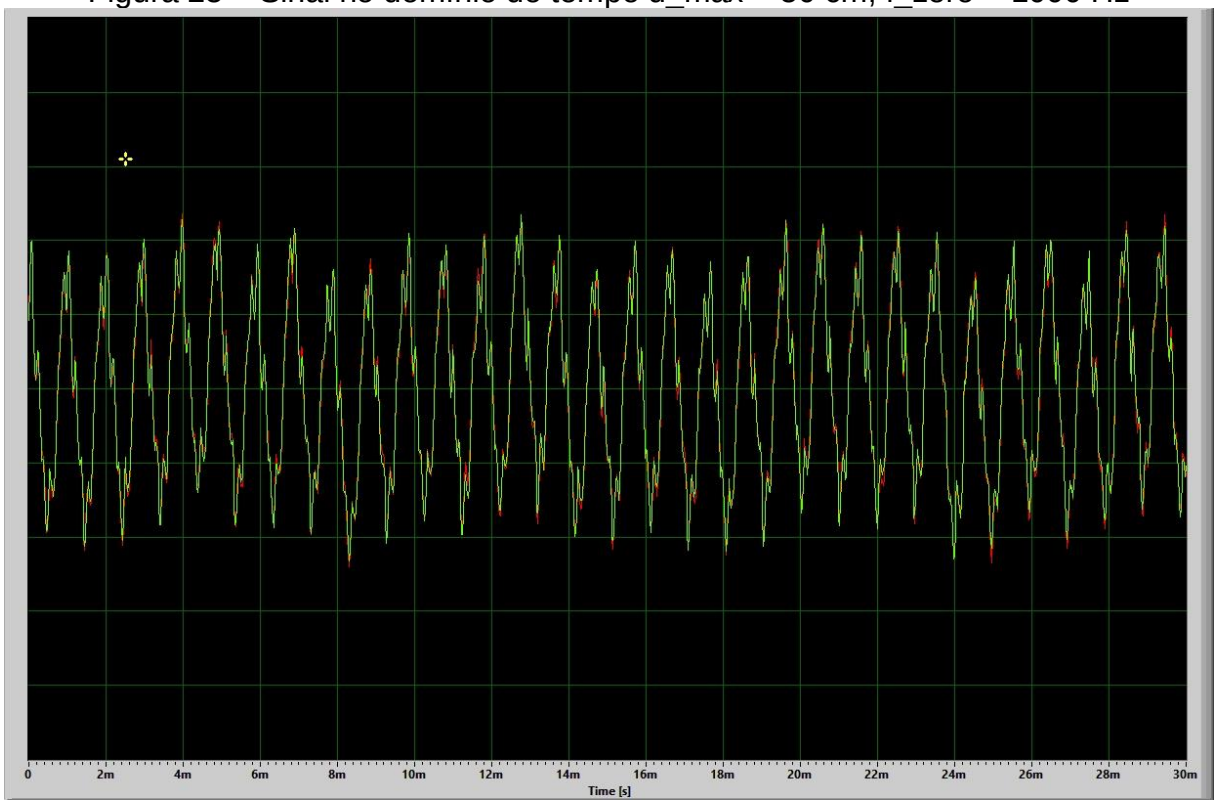
Figura 24 – Sinal no domínio da frequência d_max = 39,79 cm; f_zero = 1000 Hz



Fonte: Elaborado pelo autor.

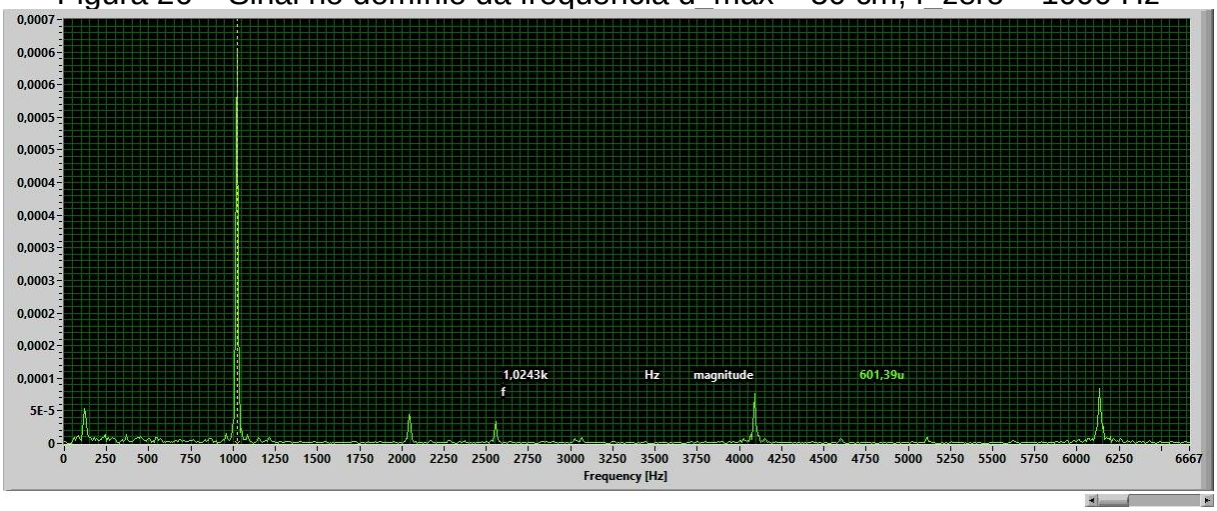
3.5.2.3 Distância máxima: 50 cm

Figura 25 – Sinal no domínio do tempo d_max = 50 cm; f_zero = 1000 Hz



Fonte: Elaborado pelo autor.

Figura 26 – Sinal no domínio da frequência d_max = 50 cm; f_zero = 1000 Hz



Fonte: Elaborado pelo autor.

3.5.3 100 Hz

A seguir os dados obtidos seguindo mesma metodologia, além da demonstração do impacto do ruído.

3.5.3.1 Ruído

Por limitação de instrumentos para obter o sinal, há um ruído de 60 Hz constante que é capturado junto ao sinal de interesse. Isto fica evidente pois, sensorialmente, é perceptível a frequência dominante de 100 Hz, porém ao capturá-la pelo osciloscópio, há uma dominância dos 60 Hz. Foi especulado que isto ocorre pela junção de dois fatores: Menor ganho do aparelho utilizado em frequências baixas e o ruído de 60 Hz sistemático.

Figura 27 – Demonstração do impacto do ruído com fs = 100 Hz – Domínio do tempo

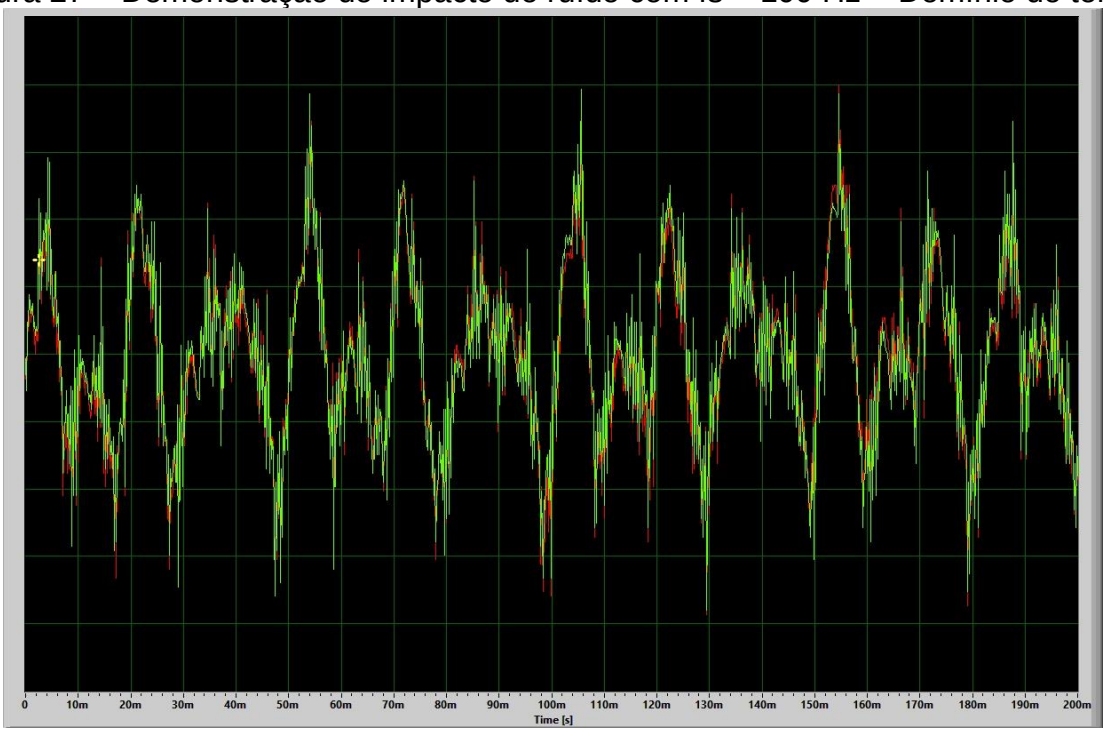
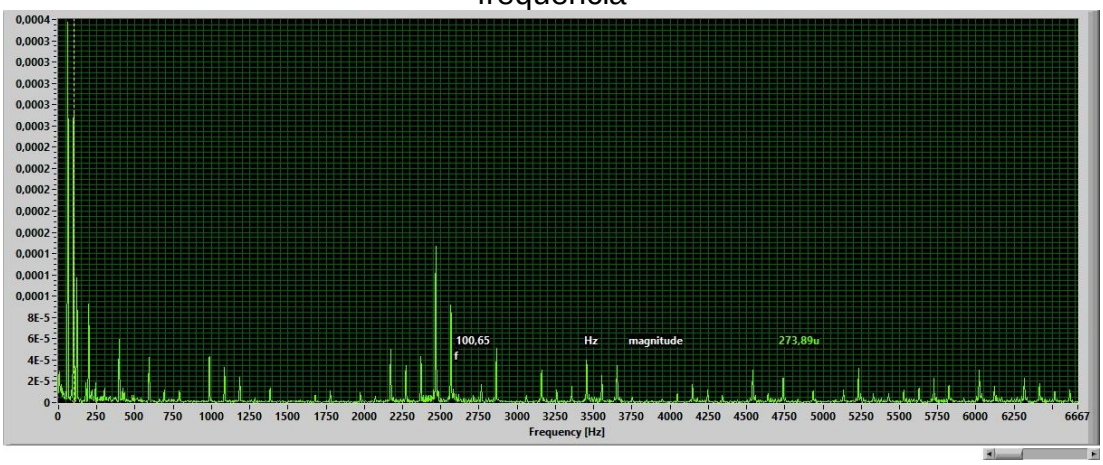


Figura 28 - Demonstração do impacto do ruído com fs = 100 Hz – Domínio da frequência

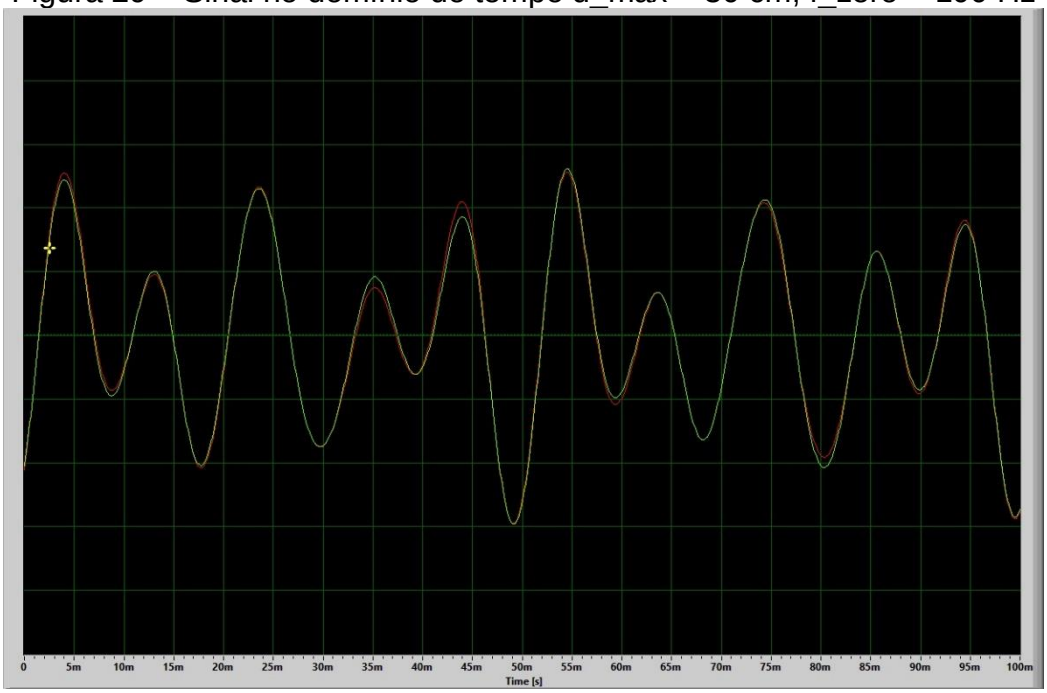


Fonte: Elaborado pelo autor.

3.5.3.2 30 cm

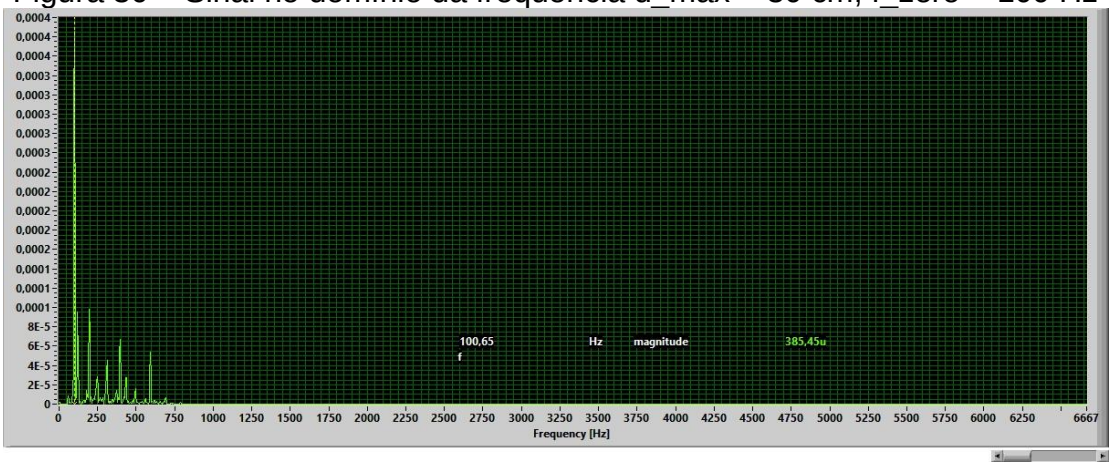
Mesmo com a filtragem do sinal, o sinal permanece alterado de forma bem mais significativa do que os casos com frequências maiores. Como a percepção do som é bem mais clara do que o visto no osciloscópio, é possível que tenha sido originada de uma dificuldade com o equipamento de medição.

Figura 29 – Sinal no domínio do tempo d_max = 30 cm; f_zero = 100 Hz



Fonte: Elaborado pelo autor.

Figura 30 – Sinal no domínio da frequência d_max = 30 cm; f_zero = 100 Hz



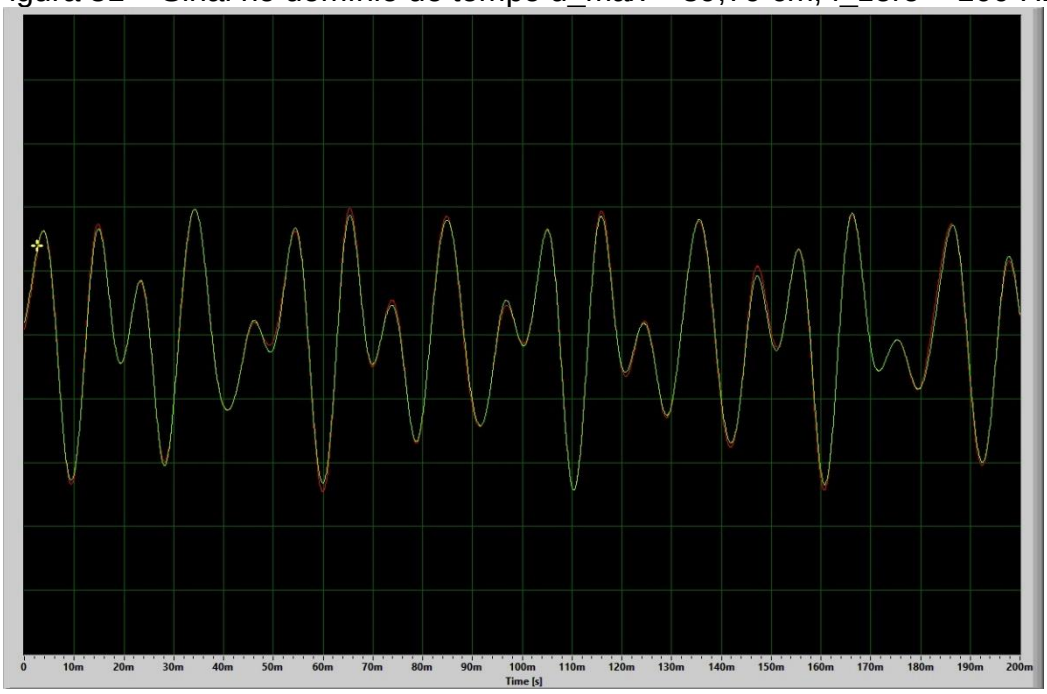
Fonte: Elaborado pelo autor.

Apesar dos ruídos, ainda é possível observar a dominância da frequência desejada, nesse caso de 100 Hz.

3.5.3.3 39,79 cm

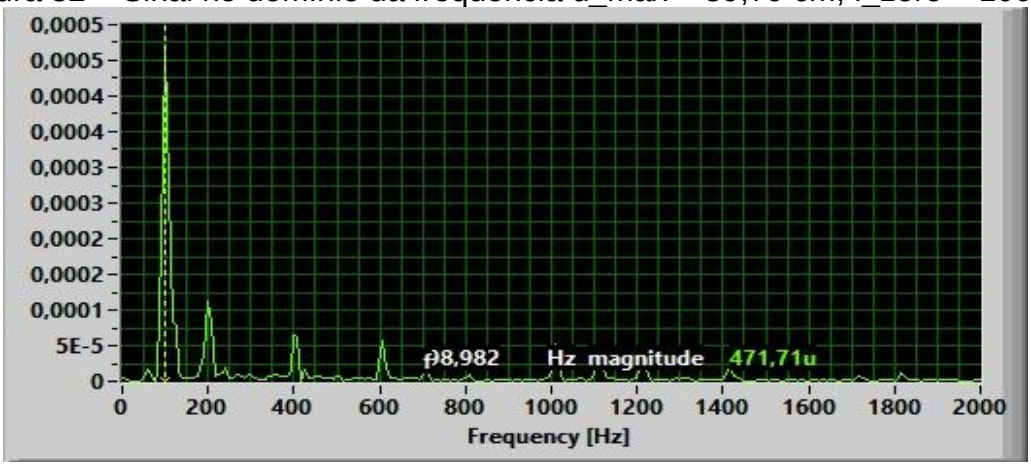
Este comportamento se repete tanto para o teste com ajuste de 39,79 cm, quanto para o teste com ajuste de 50 cm.

Figura 31 – Sinal no domínio do tempo d_max = 39,79 cm; f_zero = 100 Hz



Fonte: Elaborado pelo autor.

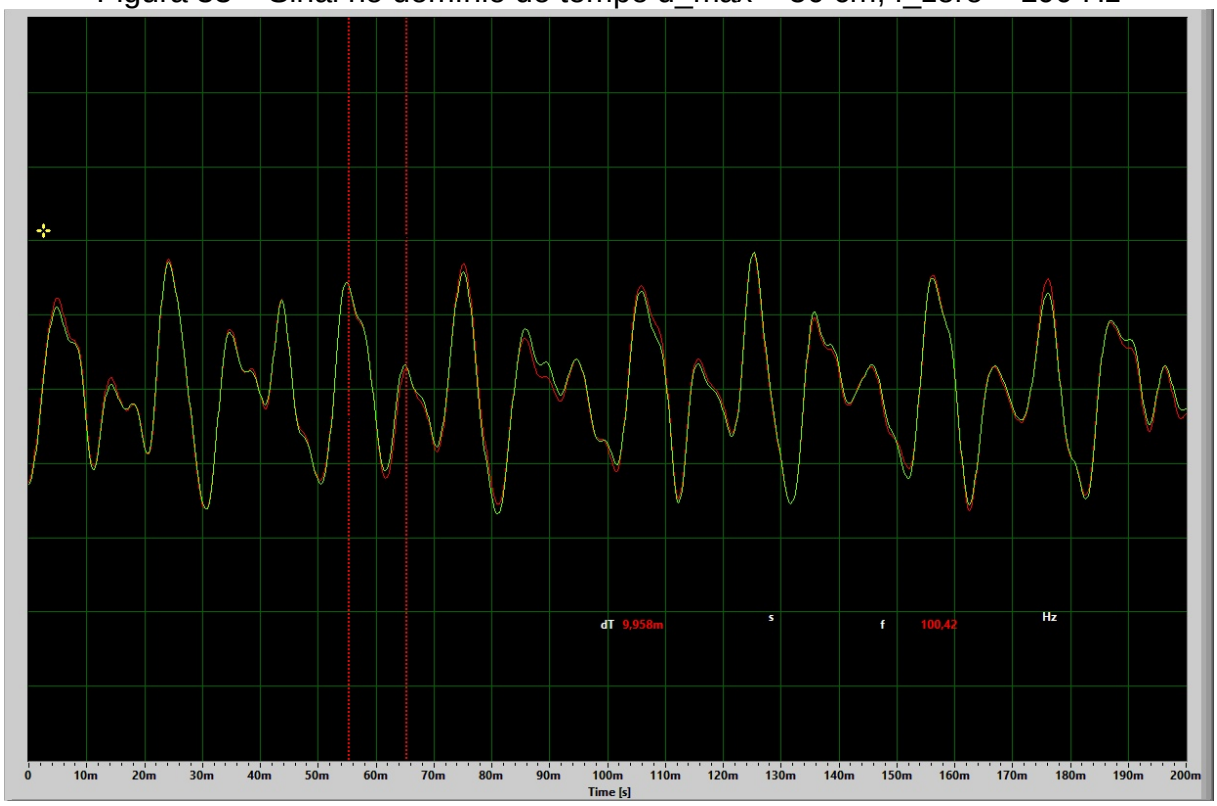
Figura 32 – Sinal no domínio da frequência d_max = 39,79 cm; f_zero = 100 Hz



Fonte: Elaborado pelo autor.

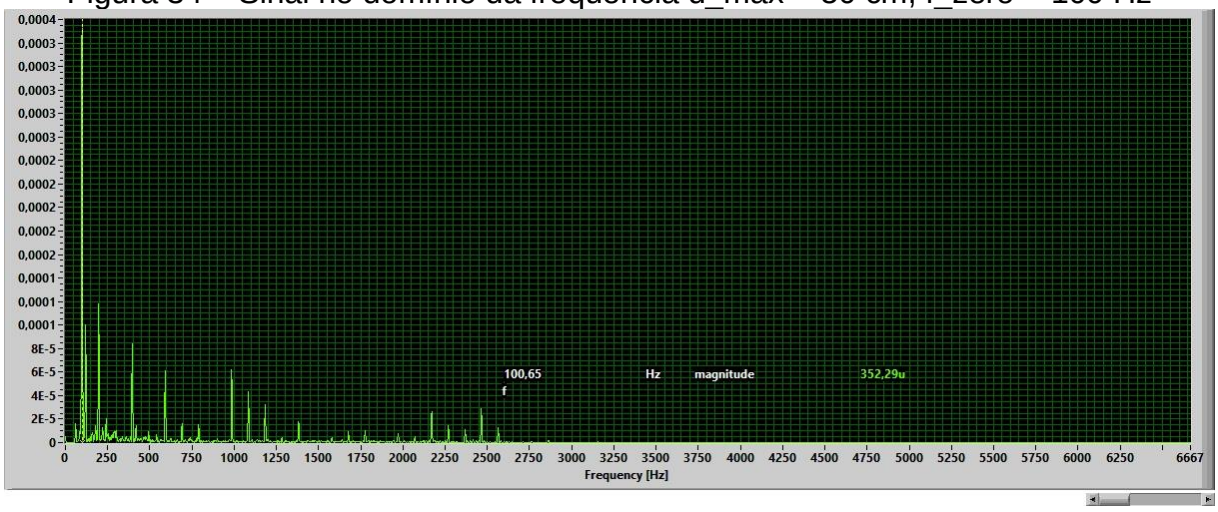
3.5.3.4 50 cm

Figura 33 – Sinal no domínio do tempo d_max = 50 cm; f_zero = 100 Hz



Fonte: Elaborado pelo autor.

Figura 34 – Sinal no domínio da frequência d_max = 50 cm; f_zero = 100 Hz



Fonte: Elaborado pelo autor.

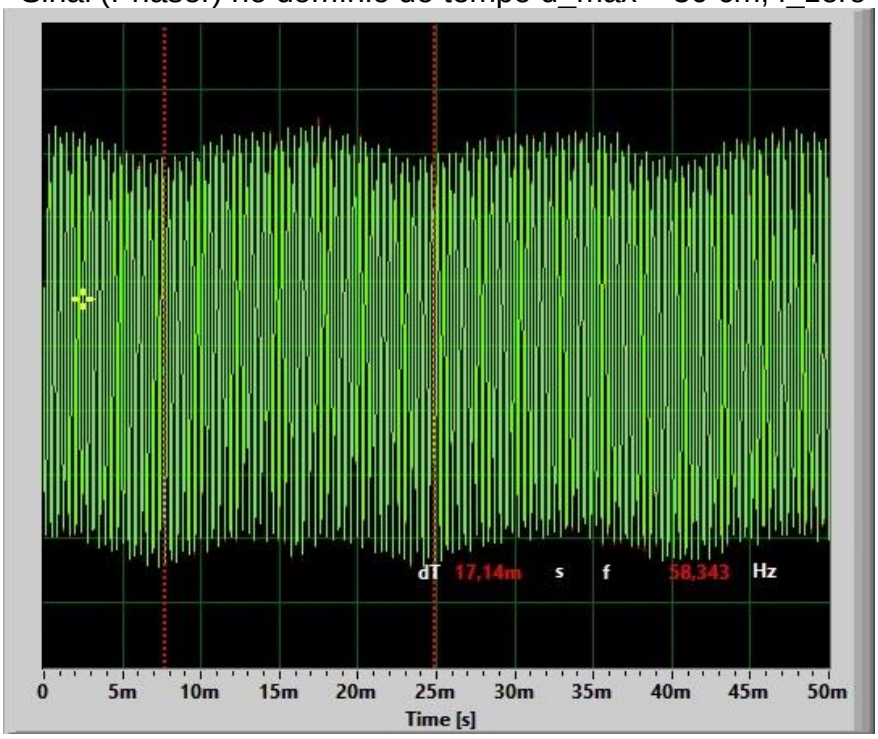
3.6 Análise do sinal com adição do filtro Phaser

A adição do filtro, além de modificar a saída, causou alterações nas variáveis do sistema, em especial a amostragem. Por conta disso, foi necessário refazer o seu cálculo com a presença do filtro. O problema foi abordado de forma mais simples, onde foi reproduzido um sinal fixando uma frequência w_zero. Desta forma, foi possível obter a amostragem diretamente pelo osciloscópio. Conhecendo o w_zero e f_zero da equação, é possível calcular a amostragem a partir da equação (10). Desta forma, tem-se um $f_{s-phaser} \approx 103\ kHz$.

Da mesma forma que abordado no item anterior, seguem os resultados do sinal com a aplicação do Phaser de segunda ordem com LFO de 0,5 Hz. Foram obtidos valores apenas para um ajuste do potenciômetro. Para esse caso, será apresentado um sinal no tempo com período maior, para demonstrar o seu efeito.

3.6.1 2600 Hz – Ajuste: 30 cm

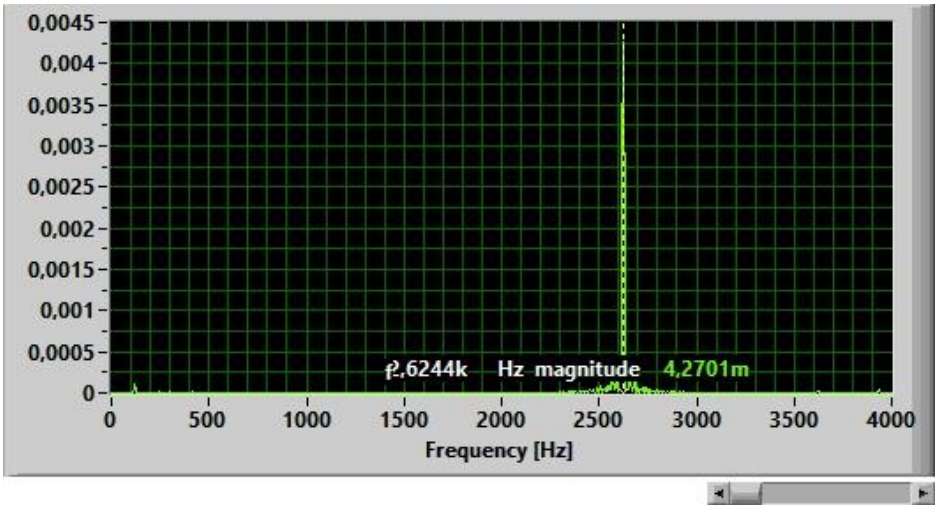
Figura 35 - Sinal (Phaser) no domínio do tempo d_max = 30 cm; f_zero = 2600 Hz



Fonte: Elaborado pelo autor.

O efeito do Phaser se manifesta no sinal com a oscilação de 58 Hz.

Figura 36 - Sinal (Phaser) no domínio da frequência d_max = 30 cm; f_zero = 2600 Hz

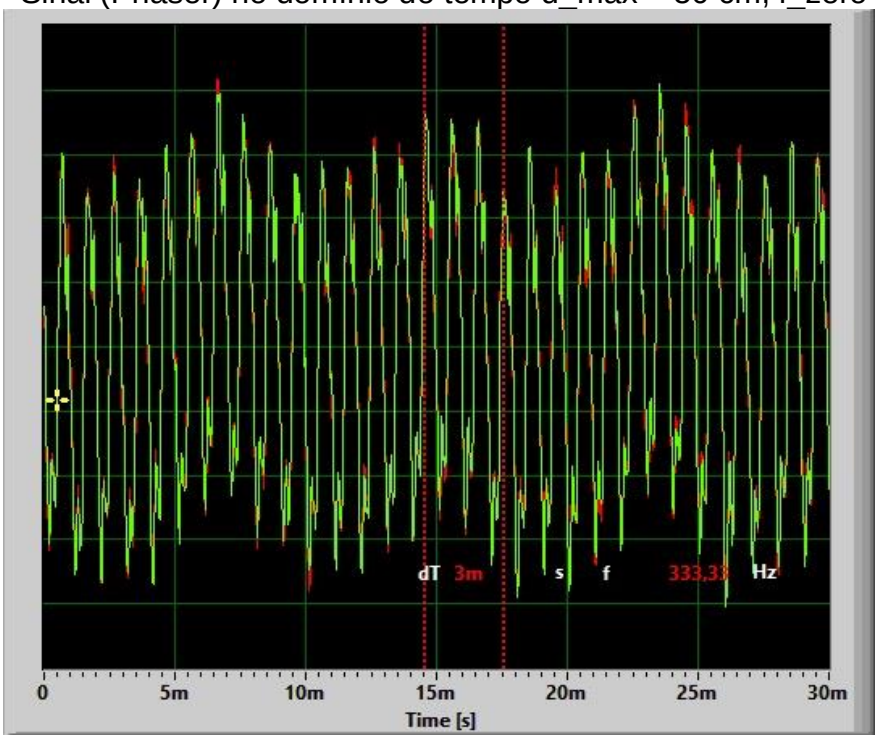


Fonte: Elaborado pelo autor.

No espectro da frequência, observa-se um pequeno pico de baixa frequência, originado do filtro Phaser.

3.6.2 1000 Hz – Ajuste: 50 cm

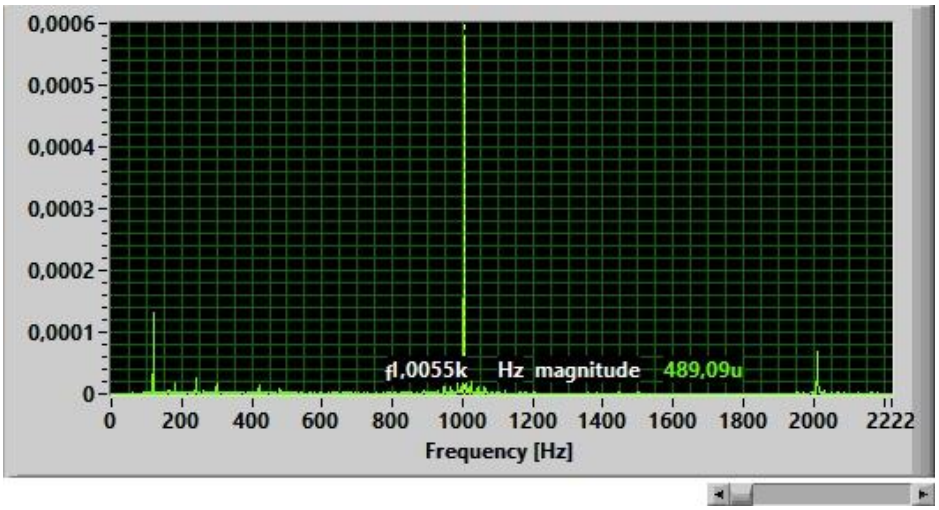
Figura 37 - Sinal (Phaser) no domínio do tempo d_max = 50 cm; f_zero = 1000 Hz



Fonte: Elaborado pelo autor.

Da mesma forma que o teste anterior, observa-se uma componente de baixa tensão alterando o sinal de 1000 Hz, causado pelo efeito do Phaser.

Figura 38 - Sinal (Phaser) no domínio da frequência d_max = 50 cm; f_zero = 1000 Hz

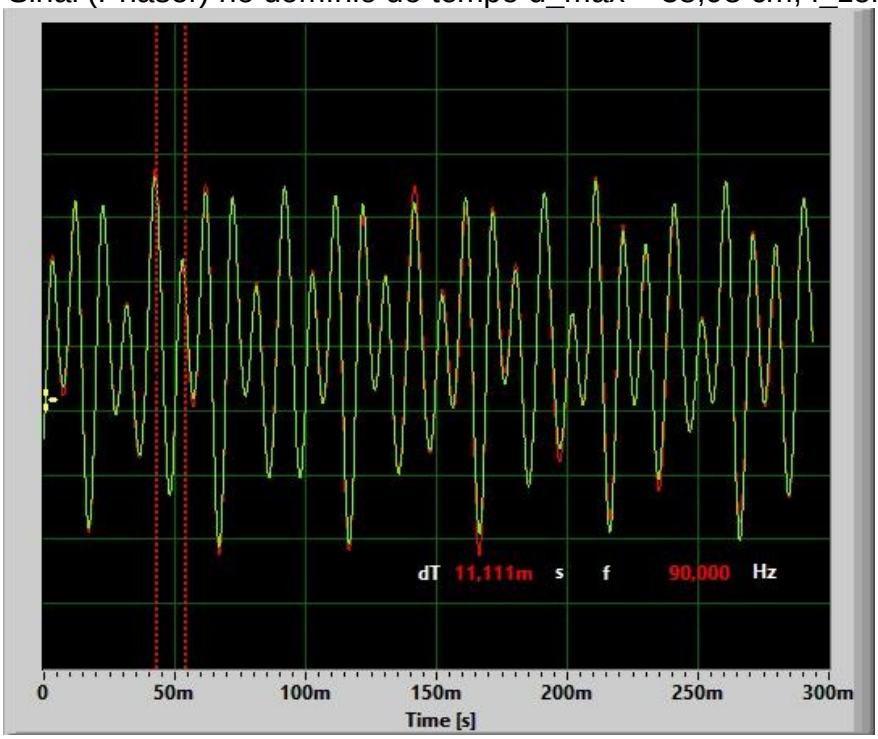


Fonte: Elaborado pelo autor.

Novamente, observa-se um pico de frequência baixa no espectro de frequência.

3.6.3 100 Hz – Ajuste: 38,98 cm

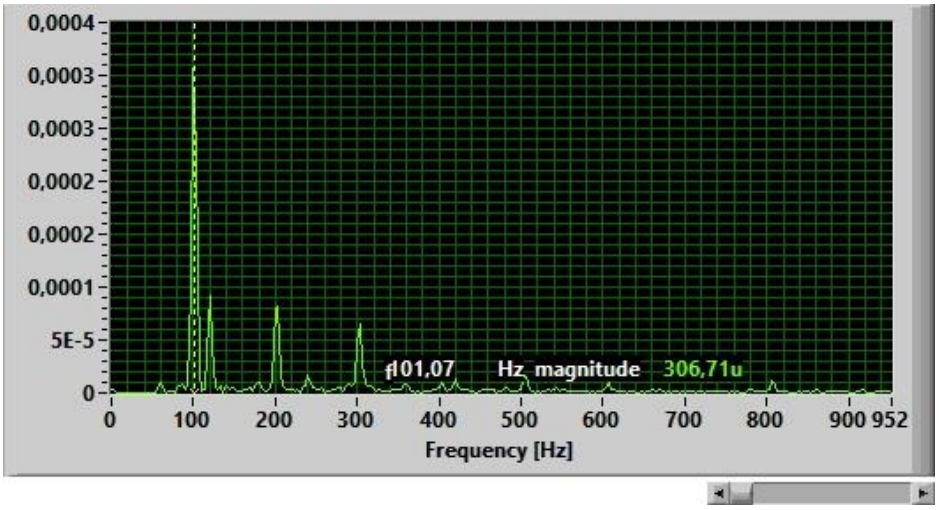
Figura 39 - Sinal (Phaser) no domínio do tempo d_max = 38,98 cm; f_zero = 100 Hz



Fonte: Elaborado pelo autor.

Por fim, temos o mesmo comportamento para baixa frequência, porém mais acentuado pois a componente adicionada pelo filtro é próxima à frequência principal.

Figura 40 - Sinal (Phaser) no domínio da frequência d_max = 39,98 cm; f_zero = 100 Hz



Fonte: Elaborado pelo autor.

4 CONSIDERAÇÕES FINAIS

Os objetivos propostos ao trabalho para desenvolvimento de um teremim baseado em ondas acústicas, assim como novos pontos levantados durante a elaboração, foram todos abordados e produziram resultados conforme foi apresentado. Desta forma, foi possível criar o sistema proposto com as funcionalidades visionadas.

Este projeto é uma prova de conceito de que o sistema proposto pode ser aplicado e o intuito é que sirva como uma plataforma para futuros estudos e melhoria do projeto, com novas análises de filtros, otimizações do programa e do *hardware*.

Quanto a aplicação de sensores ultrassônicos para esta aplicação, foram obtidos resultados inconsistentes causados pela baixa precisão do sensor ao alterar a distância rapidamente. Também a limitação da medição começar após 2 cm do sensor fere a sua aplicabilidade. Como uma proposta futura, uma boa análise seria realizar a medição com outros dispositivos, como sensores de infravermelho.

O ESP32 se provou mais que adequado para esta aplicação, podendo suportar uma frequência de amostragem até quatro vezes o mínimo requerido pelo teorema de Nyquist, sendo a amostragem mínima obtida de f_s = 155 kHz. Esta alta frequência também não seria obtida caso a programação não estivesse otimizada, o que demonstra que a metodologia aplicada foi eficaz. Um ponto de melhoria necessário para melhorar o projeto é reduzir os ruídos causados pelo *loop* como um todo. Com adição de tratamento de sinal analógico a saída do sistema para filtrar sinais causados pelo algoritmo que não sejam advindos do gerador de som, espera-se que o sinal se torne bem mais limpo.

Quanto a adição de filtros adicionais, pode haver melhorias. Por um lado, mesmo com a adição de dois estágios adicionais para o filtro Phaser, houve uma redução relativamente pequena da amostragem para $f_{s-phaser}$ = 103 kHz, ainda havendo uma tolerância para adição de estágios extras e filtros mais complexos de 2,5 vezes a amostragem de 40kHz, o limite para reproduzir todas as frequências audíveis. Levando em conta que estão sendo utilizadas frequências até 2,6 kHz, a tolerância passa a ser de 19,8 vezes acima da amostragem mínima.

Por outro lado, ficou evidente que a adição de filtros novos impacta no processamento do sistema, necessitando do cálculo da amostragem nova do sistema para cada caso específico de filtro. Um ponto importante que pode ser analisado no

futuro é o impacto de diferentes filtros no sistema. Também houve especificamente nesse caso uma necessidade de limitar o filtro em uma região específica de frequência e ângulos de defasagem para evitar que o sistema ficasse instável.

O estudo da possibilidade de adição de um algoritmo para fixar a amostragem automaticamente em um valor específico, por exemplo em 44,1 kHz, se provado possível, seria um grande passo para melhorar o sistema como um todo. Dessa maneira, mesmo com a adição de filtros, não seria necessário o cálculo da amostragem nova, e o sistema se tornaria mais resistente a variações de processamento no geral.

Não foi abordada a funcionalidade de adicionar interface com o usuário para selecionar os filtros ativos, o que demandaria a adição de outros componentes ao protótipo e novas funcionalidades no algoritmo que fugiram do escopo neste caso.

REFERÊNCIAS

CHAO. Frequency. 2012. Disponível em: https://w.electrodragon.com/w/File:Frequency..jpg . Acesso em: 24 de novembro 2024.

CHÉOUR, Rym et al. Microcontrollers for IoT: optimizations, computing paradigms, and future directions. In: **2020 IEEE 6th World Forum on Internet of Things (WF-IoT)**. IEEE, 2020. p. 1-7.

CARSON, Alistair et al. Differentiable grey-box modelling of phaser effects using frame-based spectral processing. **arXiv preprint arXiv:2306.01332**, 2023.

ELETROGATE. *Módulo Sensor de Distância Ultrassônico HC-SR04*. Disponível em: https://www.eletrogate.com/modulo-sensor-de-distancia-ultrassonico-hc-sr04. Acesso em: 24 de novembro 2024.

ESPRESSIF SYSTEMS. *ESP32 Series Datasheet*. 2021a. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Acesso em: 24 de novembro 2024.

ESPRESSIF SYSTEMS. *ESP-IDF Programming Guide*. 2021b. Disponível em: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/. Acesso em: 24 de novembro 2024.

ESPRESSIF SYSTEMS. Miscellaneous System API. 2024. Disponível em: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/misc_system_api.html. Acesso em: 24 de novembro 2024.

Garcia-Guzman, J., Salazar-Perez, D., Celis-Ronzon, G. & Velez-Enriquez, J. A. (2018). **Design and experimental prototype testing of a hybrid digital theremin system**, 2018 IEEE 8th International Conference on Consumer Electronics, IEEE, Berlin, pp. 1–4

Gomes, A., Albuquerque, D., Campos, G. & Vieira, J. (2009). Theremus: The ultrasonic theremin, *Audio Engineering Society Convention 127*. Disponível em: http://www.aes.org/e-lib/browse.cfm?elib=15055. Acesso em: 24 de novembro 2024.

Hanindhito, B., Hogantara, H., Rhamah, A. I., Ahmadi, N. & Adiono, T. (2015). **Ultrasonic sensor based contactless theremin using pipeline CORDIC as tone generator**, *2015 IEEE International Symposium on Intelligent Signal Processing and Communication Systems* (ISPACS), IEEE, Nusa Dua Bali, Indonesia, pp. 1–4.

JOHNSON, R. *Ultrasonic Sensors: Theory and Applications*. New York: Tech Press, 2016.

MAIER, Alexander; SHARP, Andrew; VAGAPOV, Yuriy. Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. In: **2017 Internet Technologies and Applications (ITA)**. IEEE, 2017. p. 143-148.

OKI, Nobuo; MANTOVANI, Suely Cunha Amaro. Microcontroladores - PIC. 2013. Disponível em: https://www.feis.unesp.br/Home/departamentos/engenhariaeletrica/microcontroladores-_pic-1.pdf. Acesso em: 24 de novembro 2024.

OPPENHEIM, A. V.; SCHAFER, R. W. Discrete-Time Signal Processing. New Jersey: Prentice Hall, 1989/1990.

PARIKH, A. *HC-SR04 Ultrasonic Sensor Datasheet*. 2018. Disponível em: https://www.electronicwings.com/sensors-modules/hc-sr04-ultrasonic-sensor. Acesso em: 24 de novembro 2024.

APÊNDICE A – Programação completa do ESP32 no ambiente Arduino IDE

O algoritmo a seguir traz a programação completa do teremim para o ESP32 no ambiente Arduino IDE incluindo a aplicação do efeito Phaser.

```
#include <esp_ipc.h>

//Definições Pinos de entrada/saída:
int n = 3;  //n° de sensores
int Tx[3] = { 16, 5, 19 };    //Definição dos pinos "Trigger"
int Rx[3] = { 17, 18, 21 };   //Definição dos pinos "Echo"
float Raw[3];

int pot = 34; //Definição entrada potenciometro
int PIN_OUT = 26; //Definição da saída de áudio (DAC 2)


//Parametros de verificação da medição:
int test[3];
int PRO_running = 0;

// valores finais calculados
float Final_Y, f_zero = 0, w_zero = 0.009, amp = 1;

//Parâmetros da conversão distância-frequência:
float f_min = 100;
float f_max = 2600;
float amostragem = 103000;
float w_min = 2*PI*100/amostragem;
float w_max = 2*PI*2600/amostragem;
float d_min = 2;
float d_max = 40;
float k = 0.918;
float b = -93.9;
float razao = 2 * PI/amostragem;
double Output1, saida;
// Variáveis globais para o LFO
const float LFO_FREQ = 0.5; // Frequência do LFO em Hz
float lfo_phase = 1.3694;
float lfo_increment = PI * LFO_FREQ / amostragem;

// Variáveis para os filtros passa-tudo
float x_prev = 0.0, y_prev = 0.0;
float x_prev2 = 0.0, y_prev2 = 0.0;

void setup() {
  for (int i = 0; i < n; i++) {
    pinMode(Tx[i], OUTPUT);
    pinMode(Rx[i], INPUT);
  }

  // Setup "afinação":
  Serial.begin(115200);
  setup_teremim();
  Serial.print("k:");
```

```
  Serial.println(k);
  Serial.print("b:");
  Serial.println(b);

  Final_Y = d_max;
  Final_Y  = map(Final_Y*1000,2000,d_max*1000,2000,40000);
  Final_Y  = (float)Final_Y/1000;

  converter_exp();
  Serial.print("d_max:");
  Serial.println(d_max);
  Serial.print("Final_Y:");
  Serial.println(Final_Y);
  Serial.print("f_zero:");
  Serial.println(f_zero);
  Serial.print("w_min:");
  Serial.println(w_min, 4);


  Final_Y = d_min;
  Final_Y  = map(Final_Y*1000,2000,d_max*1000,2000,40000);
  Final_Y  = (float)Final_Y/1000;

  converter_exp();
  Serial.print("d_min:");
  Serial.println(d_min);
  Serial.print("Final_Y:");
  Serial.println(Final_Y);
  Serial.print("f_zero:");
  Serial.println(f_zero);
  Serial.print("w_max:");
  Serial.println(w_max, 4);

  esp_ipc_call(PRO_CPU_NUM, runOnCPUCore, NULL);
}

// CORE APP:
void loop() {
  Som();
  esp_ipc_call(PRO_CPU_NUM, runOnCPUCore, NULL);
}

// CORE PRO:
void runOnCPUCore(void *arg) {
  PRO_running = 1;
  Raw_cap();
  Position_calc();
  Fuse_data();
  PRO_running = 0;
}

// Geração do som:
void Som() {
  double Output2, Output, teste;
  float w_som;
  int sen = 0;
  int s = 0;
```

```
    w_som = w_zero;

    double output_aux = cos(w_som) * 2;
    Output1 = 1;
    Output2 = output_aux / 2;
    Output = output_aux * Output2 - Output1;

    while (sen < 4) {

      Output1 = Output2;
      Output2 = Output;
      Output = output_aux * Output2 - Output1;

      if (Output1 < Output2 && s == 0) {
        sen++;
        s = 1;
      }
      if (Output1 > Output2 && s == 1) {
        sen++;
        s = 0;
      }

    //aplicação do filtro:
    aplicarPhaser();

    //peparação para saida do DAC:
    saida = (saida + 1.1) * amp;

      if (saida > 255) {
        dacWrite(PIN_OUT, 255);

      }
      if (saida < 0) {
        dacWrite(PIN_OUT, 0);

      }
      else {
        dacWrite(PIN_OUT, saida);

      }
    }

    s = 0;
    sen = 0;

    if (PRO_running == 1) {
      Som();
    }

    return;
  }

  // Medição da distância de cada sensor:
  void Raw_cap() {
    for (int i = 0; i < n; i++) {
```

```
      Raw[i] = 0;
      digitalWrite(Tx[i], LOW);
      delayMicroseconds(2);
      digitalWrite(Tx[i], HIGH);
      delayMicroseconds(10);
      digitalWrite(Tx[i], LOW);
      Raw[i] = pulseIn(Rx[i], HIGH, 5000);
      delay(15);
      Raw[i] = Raw[i] / 58.3;
    }
  }

  // Validação da medição:
  void Position_calc() {
    float d1, d2;

    for (int i = 1; i < n; i++) {
      if (Raw[i] > 0) {
        test[i] = 1;
      } else {
        test[i] = 0;
      }
    }
  }

  // Definição dos parâmetros da onda:
  void Fuse_data() {
    int count = 0;
    float aux_Y = 0;

    // Volume no Sensor 0:
    amp = Raw[0] / 10;
    amp = constrain(amp, 0, 4);

    // Cálculo final para frequência - média Sensores 1-2:
    for (int i = 1; i < n; i++) {
      if (test[i] == 1) {
        aux_Y = aux_Y + Raw[i];
        count++;
      }
    }

    aux_Y = aux_Y / count;

    // Teste de falhas de leitura:
    if (isnan(amp)) {
      amp = 0;
      return;
    }
    if (!isnan(aux_Y)) {
      aux_Y = map(aux_Y*1000,2000,d_max*1000,2000,40000);
      aux_Y = aux_Y/1000;
      Final_Y = aux_Y;
      converter_exp();
      return;
    }
  }
```

```
// Conversão da distância para frequência - exponencial negativa:
void converter_exp() {
  float razao = 2 * PI / amostragem;
  float exponencial = Final_Y + b;
  f_zero = pow(k, exponencial);
  w_zero = razao * f_zero;
  w_zero = constrain(w_zero, w_min, w_max);
  return;
}

// Definição parâmetros iniciais:
void setup_teremim() {
  // Distância máxima
  d_max = analogRead(pot);
  d_max = map(d_max, 0, 4095, 30000, 50000);
  d_max = d_max / 1000;
  d_max = constrain(d_max, 30, 50);
  Serial.print(d_max);

  return;
}

// Função para aplicar o efeito phaser
void aplicarPhaser() {
    // Atualiza a fase do LFO
    lfo_phase += lfo_increment;
     if (lfo_phase >= 1.7722) {
        lfo_phase = 1.3694;
    }

  float a = abs(cos(lfo_phase));
    // Primeiro estágio do filtro passa-tudo

    float y1 = (-a) * (y_prev) + x_prev - a*Output1;
    y_prev = y1;
    x_prev = Output1;

    // Segundo estágio do filtro passa-tudo
    float y2 = (-a) * (y_prev) + x_prev - a*y1;
    y_prev2 = y2;
    x_prev2 = y1;

    // Retorna o sinal processado
    Saída = Output1 * 0.3 + y2;
}
```

APÊNDICE B – Algoritmo em Octave GNU para geração do sinal teórico

```
PONTOS = 500;
amp = 10;

Final_Y = 2;

s=0;
sen=0;
index_plot=0;
f_zero = 0;
w_zero = 0;
k = 0.918;
amostragem = 131800;
razao = 2 * pi/amostragem;
exponencial = Final_Y-93.8;
cont = 1;
Output1 = zeros(1,PONTOS);
Output2 = zeros(1,PONTOS);
Output = zeros(1,PONTOS);
saida = zeros(1,PONTOS);

while (cont < PONTOS)

  f_zero = k^exponencial;
  w_zero = razao * f_zero;

  Output1(cont) = 1;
  Output2(cont) = cos(w_zero);
  Output(cont) = 2*cos(w_zero)*Output2(1) - Output1(1);
  cont++;

  while (sen < 4 && cont < PONTOS)
    Output1(cont) = Output2(cont-1);
    Output2(cont) = Output(cont-1);
    Output(cont) = 2*cos(w_zero)*Output2(cont) - Output1(cont);
    saida(cont) = (Output1(cont) * amp) + amp*1.1

    teste = Output1(cont-1);

    if (teste < Output1(cont) && s == 0)
      sen++;
      s = 1;
    end

    if (teste > Output1(cont) && s == 0)
      sen++;
      s = 0;
    end
```

```
  end

  saida(cont) = saida(cont-1);
  sen = 0;
  s = 0;
  Final_Y = Final_Y + 5;
  exponencial = Final_Y-93.8;
end
```

APÊNDICE C – ADAPTAÇÃO PARA AQUISIÇÃO DA AMOSTRAGEM DO DAC

Para essa verificação, a parte do algoritmo para aquisição da leitura foi simplificada, dado que o valor de interesse é quanto a amostragem do algoritmo de geração de som, que não é afetado pelo processamento do outro núcleo.

O valor da leitura nesse caso pode ser alterado apenas pelo potenciômetro, sem uso dos sensores ultrassônicos.

```
#include <esp_ipc.h>
int PRO_running = 0;
float amostragem = 160000;

int pot = 34;
int PIN_OUT = 26;
int cont = 0;

float f_min = 100;
float f_max = 2600;
float d_min = 2;
float d_max = 40;
float k = 0.918;
float razao = 2 * PI/amostragem;

float Final_Y, f_zero = 0, w_zero = 0.11;
float amp = 2;

unsigned long tempo_zero = 0;
unsigned long tempo_fim = 0;
unsigned long tempo = 0;

void setup() {

  Serial.begin(115200);

  while(cont<=40) {
    esp_ipc_call(PRO_CPU_NUM, runOnCPUCore, NULL);
    Som();
    cont++;
  }
}

void loop() {

  }

void runOnCPUCore(void *arg) {
  PRO_running = 1;
  leitura();
  converter_exp();
  PRO_running = 0;
}
```

```
void Som(){

  float amostragem_loop = 0;
  float pontos = 0;
  tempo_zero = micros();

  double Output1, Output2, Output, teste;
  double saida;
  float w_som;
  int sen = 0;
  int s = 0;

  w_som = w_zero;
  double output_aux = cos(w_som)*2;
  Output1 = 1;
  Output2 = output_aux/2;
  Output = output_aux * Output2 - Output1;


  while(sen<4){
    pontos++;
    Output1 = Output2;
    Output2 = Output;
    Output = output_aux*Output2 - Output1;

    saida = (Output1 + 1.1) * amp;

    if(Output1 < Output2 && s == 0){
      sen++;
      s = 1;
    }

    if(Output1 > Output2 && s == 1){
    sen++;
    s = 0;
    }

     if(saida > 255){

      dacWrite(PIN_OUT,255);

    }

    if(saida < 0){

      dacWrite(PIN_OUT,0);

    }

    else{

      dacWrite(PIN_OUT, saida);

    }
  }
```

```
  s = 0;
  sen = 0;

  //Medição amostragem:
  tempo_fim = micros();
  tempo = tempo_fim - tempo_zero;
  amostragem_loop = pontos * 1000000/tempo;
  Serial.print("Pontos:");
  Serial.println(pontos);
  Serial.print("Tempo:");
  Serial.println(tempo);
  Serial.print("Fs:");
  Serial.println(amostragem_loop);
  Serial.print("Fs_t:");
  Serial.println(amostragem);
  Serial.print("Frequencia(Nota):");
  Serial.println(f_zero);
  Serial.print("w_zero:");
  Serial.println(w_som);
  Serial.print("Final_Y:");
  Serial.println(Final_Y);

  if(PRO_running == 1){

    Som();

  }
  return;
}

void leitura(){

  Final_Y = analogRead(pot);
  Final_Y = Final_Y * 9.9768/1000;
  Final_Y = cont;
  Final_Y = constrain(Final_Y,2,40);

  razao = 2 * PI/amostragem;
  return;
}

void converter_exp() {
  float exponencial = Final_Y-93.8;
  f_zero = pow(k,exponencial);

  w_zero = razao * f_zero;
  w_zero = constrain(w_zero,0.003,0.99);
  return;
}
```

APÊNDICE D – RESULTADOS DA AQUISIÇÃO DA AMOSTRAGEM

Tabela 2 – Aquisição da amostragem do ESP32

Final_Y (cm)	Pontos	Tempo (μs)	Fs_real (Hz)	ΔFs (Hz)	%ΔFs
2,00	128,00	820,00	156097,56	-3902,44	**-2,44%**
3,00	128,00	820,00	155000,00	-5000,00	**-3,13%**
4,00	140,00	894,00	156599,55	-3400,45	**-2,13%**
5,00	152,00	971,00	156539,66	-3460,34	**-2,16%**
6,00	166,00	1054,00	157495,25	-2504,75	**-1,57%**
7,00	180,00	1141,00	157756,36	-2243,64	**-1,40%**
8,00	196,00	1245,00	157429,72	-2570,28	**-1,61%**
9,00	214,00	1356,00	157817,11	-2182,89	**-1,36%**
10,00	233,00	1469,00	158611,30	-1388,70	**-0,87%**
11,00	254,00	1603,00	158452,91	-1547,09	**-0,97%**
12,00	277,00	1746,00	158648,34	-1351,66	**-0,84%**
13,00	301,00	1891,00	159175,05	-824,95	**-0,52%**
14,00	328,00	2061,00	159146,05	-853,95	**-0,53%**
15,00	358,00	2249,00	159181,86	-818,14	**-0,51%**
16,00	390,00	2443,00	159639,78	-360,22	**-0,23%**
17,00	424,00	2660,00	159398,50	-601,50	**-0,38%**
18,00	462,00	2897,00	159475,31	-524,69	**-0,33%**
19,00	504,00	3162,00	159392,80	-607,20	**-0,38%**
20,00	548,00	3418,00	160327,67	327,67	**0,20%**
21,00	597,00	3727,00	160182,45	182,45	**0,11%**
22,00	651,00	4068,00	160029,50	29,50	**0,02%**
23,00	709,00	4428,00	160117,44	117,44	**0,07%**
24,00	772,00	4815,00	160332,30	332,30	**0,21%**
25,00	841,00	5256,00	160007,61	7,61	**0,00%**
26,00	917,00	5719,00	160342,72	342,72	**0,21%**
27,00	998,00	6220,00	160450,16	450,16	**0,28%**
28,00	1088,00	6793,00	160164,88	164,88	**0,10%**
29,00	1185,00	7398,00	160178,42	178,42	**0,11%**
30,00	1290,00	8037,00	160507,66	507,66	**0,32%**
31,00	1397,00	8705,00	160482,47	482,47	**0,30%**
32,00	1397,00	8705,00	160482,47	482,47	**0,30%**
33,00	1397,00	8700,00	160574,70	574,70	**0,36%**
34,00	1397,00	8705,00	160482,47	482,47	**0,30%**
35,00	1397,00	8711,00	160371,94	371,94	**0,23%**
36,00	1397,00	8705,00	160482,47	482,47	**0,30%**
37,00	1397,00	8704,00	160500,91	500,91	**0,31%**
38,00	1397,00	8707,00	160445,61	445,61	**0,28%**
39,00	1397,00	8703,00	160519,36	519,36	**0,32%**
40,00	1397,00	8704,00	160500,91	500,91	**0,31%**

Fonte: Elaborado pelo autor.

Tabela 3 – Desvio da frequência (Hz) em relação a amostragem utilizada

Final_Y (cm)	w_zero (rad)	F_zero_real (Hz)	f_zero_teorico (Hz)	Δf_zero (Hz)	%Δf_zero
2,00	0,10118	2513,74	2576,58	-62,84	**-2,44%**
3,00	0,09812	2420,43	2498,50	-78,08	**-3,13%**
4,00	0,09007	2244,87	2293,61	-48,75	**-2,13%**
5,00	0,08268	2059,99	2105,53	-45,54	**-2,16%**
6,00	0,07590	1902,62	1932,88	-30,26	**-1,57%**
7,00	0,06968	1749,50	1774,39	-24,88	**-1,40%**
8,00	0,06397	1602,71	1628,88	-26,17	**-1,61%**
9,00	0,05872	1474,92	1495,32	-20,40	**-1,36%**
10,00	0,05391	1360,79	1372,71	-11,91	**-0,87%**
11,00	0,04949	1247,97	1260,15	-12,18	**-0,97%**
12,00	0,04543	1147,04	1156,81	-9,77	**-0,84%**
13,00	0,04301	1089,49	1095,14	-5,65	**-0,52%**
14,00	0,04301	1089,29	1095,14	-5,84	**-0,53%**
15,00	0,03948	1000,21	1005,35	-5,14	**-0,51%**
16,00	0,03624	920,82	922,89	-2,08	**-0,23%**
17,00	0,03327	844,03	847,21	-3,18	**-0,38%**
18,00	0,03054	775,20	777,75	-2,55	**-0,33%**
19,00	0,02804	711,27	713,98	-2,71	**-0,38%**
20,00	0,02574	656,78	655,44	1,34	**0,20%**
21,00	0,02363	602,37	601,68	0,69	**0,11%**
22,00	0,02169	552,46	552,36	0,10	**0,02%**
23,00	0,01991	507,43	507,05	0,37	**0,07%**
24,00	0,01828	466,44	465,47	0,97	**0,21%**
25,00	0,01678	427,32	427,30	0,02	**0,00%**
26,00	0,01540	393,10	392,26	0,84	**0,21%**
27,00	0,01414	361,11	360,10	1,01	**0,28%**
28,00	0,01192	303,78	303,46	0,31	**0,10%**
29,00	0,01094	278,90	278,58	0,31	**0,11%**
30,00	0,01004	256,55	255,74	0,81	**0,32%**
31,00	0,00922	235,47	234,76	0,71	**0,30%**
32,00	0,00846	216,16	215,51	0,65	**0,30%**
33,00	0,00769	196,53	195,82	0,70	**0,36%**
34,00	0,00713	182,16	181,61	0,55	**0,30%**
35,00	0,00655	167,11	166,72	0,39	**0,23%**
36,00	0,00601	153,53	153,07	0,46	**0,30%**
37,00	0,05518	1409,55	1405,15	4,40	**0,31%**
38,00	0,00507	129,34	128,98	0,36	**0,28%**
39,00	0,00465	118,80	118,41	0,38	**0,32%**
40,00	0,00427	109,05	108,71	0,34	**0,31%**

Fonte: Elaborado pelo autor.