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

DIEGO EDUARDO TOMASI

DESENVOLVIMENTO DE UM POTENCIOSTATO COM INTEGRAÇÃO A SERVIDOR REMOTO E EXPORTAÇÃO DE DADOS PARA PLANILHAS EXCEL

Porto Alegre
2025

DIEGO EDUARDO TOMASI

DESENVOLVIMENTO DE UM POTENCIOSTATO COM INTEGRAÇÃO A SERVIDOR REMOTO E EXPORTAÇÃO DE DADOS PARA PLANILHAS EXCEL

Projeto de diplomação apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica na Universidade Federal do Rio Grande do Sul.

Orientador (a): Prof. Dr. Altair Sória Pereira

Porto Alegre

2025

CIP - Catalogação na Publicação

Tomasi, Diego Eduardo
Desenvolvimento de um potenciostato com integração a servidor remoto e exportação de dados para planilhas excel / Diego Eduardo Tomasi. -- 2025.
124 f.
Orientador: Altair Sória Pereira.

Trabalho de conclusão de curso (Graduação) -- Universidade Federal do Rio Grande do Sul, Escola de Engenharia, Curso de Engenharia Elétrica, Porto Alegre, BR-RS, 2025.

1. Potenciostato. 2. ESP32. 3. Servidor. 4. Python. I. Sória Pereira, Altair, orient. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da UFRGS com os dados fornecidos pelo(a) autor(a).

DIEGO EDUARDO TOMASI

DESENVOLVIMENTO DE UM POTENCIOSTATO COM INTEGRAÇÃO A SERVIDOR REMOTO E EXPORTAÇÃO DE DADOS PARA PLANILHAS EXCEL

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de “Bacharel” e aprovado em sua forma final pelo Curso de Engenharia Elétrica, obtendo conceito **C.**

Porto Alegre, 14 de janeiro de 2025.

_____
Prof. Ivan Müller, Dr.
Coordenador do Curso

Banca Examinadora:

_____
Prof. Altair Sória Pereira, Dr.
Orientador
Universidade Federal do Rio Grande do Sul

_____
Prof. Antônio Manuel Santos Spencer Andrade, Dr.
Universidade Federal do Rio Grande do Sul

_____
Prof. Alexandre Ambrozi Junqueira, MSc.
Universidade Federal do Rio Grande do Sul

RESUMO

Foi desenvolvido, a partir da adaptação de um circuito padrão presente em outros trabalhos, um potenciostato de baixo custo e portátil usando um microcontrolador ESP32, que consegue utilizar mais de uma técnica de medição e que envia os dados a um servidor distante, que não necessita estar na mesma rede wi-fi que o ESP32, através de um broker (servidor) utilizando o protocolo MQTT de comunicação. Além do teste com componentes elétricos resistivos comerciais, foi realizado um teste com um líquido, comparando-o com um potenciostato comercial. Foram feitas simulações no software Proteus 8, testando a capacidade do instrumento de realizar a espectroscopia de impedância. Apesar do número limitado de testes não permitir que o protótipo desenvolvido seja validado como aplicação final, foi demonstrado que o mesmo apresenta as funcionalidades desejadas de conseguir se comunicar com um servidor não conectado na mesma rede wi-fi, bem como de possuir um sistema de organização dos dados e de cálculos fora do ESP32, utilizando a linguagem de programação Python. Dessa forma, pode servir de uma base sólida para o desenvolvimento de um potenciostato, que seja uma solução instrumental robusta para a necessidade de um equipamento portátil, de baixo custo e de uso flexível.

Palavras-chave: Potenciostato; ESP32; Servidor; Python; Wi-Fi.

ABSTRACT

From the adaptation of a standard circuit present in other works, a low-cost and portable potentiometer was developed using an ESP32 microcontroller, which can use more than one measurement technique and which sends the data to a distant server, which does not require be on the same Wi-Fi network as the ESP32, through a broker (server) using the MQTT communication protocol. In addition to testing with commercial resistive electrical components, a test was carried out with a liquid, comparing it with a commercial potentiostat. Simulations were carried out in the Proteus 8 software, testing the instrument's ability to perform impedance spectroscopy. Although the limited number of tests does not allow the developed prototype to be validated as a final application, it was demonstrated that it presents the desired functionality of being able to communicate with a server not connected to the same Wi-Fi network, as well as to have a systematic organization system

of data and calculations outside the ESP32, using the Python programming language. In this way, it can serve as a solid basis for the development of a robust instrumental solution to the need for portable, low-cost and flexible-use equipment potentiostat.

Keywords: Potentiostat; ESP32; Server; Python; Wi-Fi.

GLOSSÁRIO

Broker: Segundo Yuan (2017, apud MARTINS, 2019) um broker é um servidor que permite que dispositivos se comuniquem entre si, enviando ou recebendo dados, de forma que o broker seja o intermediário.

Mosquitto: Segundo Light *et al*. (2017, apud MARTINS, 2019) Mosquitto é um broker *Message Queuing Telemetry Transport* (MQTT), que atua como intermediário na comunicação entre dispositivos em sistemas de publicação/assinatura. É um *software* de código aberto, amplamente utilizado em aplicações de Internet das Coisas (IoT).

Clientes: Segundo Yuan (2017, apud MARTINS, 2019) clientes são dispositivos capazes de se comunicar com um broker. Os que recebem dados dele são os assinantes (subscribers). Os que enviam informações a ele são os publicadores (publishers).

MQTT: Sigla do protocolo de comunicação *Message Queuing Telemetry Transport*.

IOT: Sigla para Internet das coisas (*Internet of Things*), que, segundo Lemos (2013, apud ALMEIDA *et al*., 2019), é um conjunto de redes, sensores, atuadores e outros objetos ligados por sistemas informatizados, que ampliam a comunicação entre pessoas e objetos e entre objetos de forma autônoma, automática e sensível ao contexto.

ESP32: Circuito integrado, produzido pela *Espressif*, que trabalha como microcontrolador próprio para IOT, uma vez que contém módulo *wifi* e *Bluetooth* integrados.

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO DO TRABALHO: IMPORTÂNCIA DA ELETROQUÍMICA E MEDIÇÕES POTENCIOSTÁTICAS EM DIVERSAS ÁREAS (MEDICINA, ELETROQUÍMICA)

O potenciostato é resumidamente uma fonte de tensão elétrica controlada que mede a corrente elétrica, utilizada em técnicas de caracterização de propriedades elétricas de diferentes tipos de sistema, usando tensão elétrica contínua ou alternada de formas de onda diversas. Essas formas de onda dependem do tipo de técnica de caracterização que é empregada.

Um exemplo dessas técnicas é a Espectroscopia de Impedância, com a qual, a partir das medições de tensão elétrica RMS, corrente elétrica RMS, potências ativa e aparente, pode ser determinada a impedância de líquidos em diferentes frequências e assim ser determinado o circuito elétrico equivalente do mesmo, considerando modelos teóricos propostos para descrever as propriedades eletroquímicas desses líquidos.

Uma aplicação interessante de um potenciostato é a medição da concentração de um líquido em outro, segundo Santos (2013). Eletrodos usados em potenciostatos são segundo Morawski (2024) também usados para desenvolver dispositivos point-of-care, para o diagnóstico de fibrose cística, uma das doenças verificadas na triagem neonatal, que tem o objetivo de identificar doenças em crianças em seus primeiros 5 dias de vida.

Em diversas situações de aplicações de técnicas que utilizam potenciostatos, como medidas feitas em campo (por exemplo, em hospitais, plantas industriais, redes de esgoto, etc.), seria extremamente interessante a disponibilidade de um dispositivo portátil, preferencialmente de baixo custo, e que envie dados através da Internet de forma autônoma, sem a necessidade de se levar junto um computador para armazenar as informações.

Dentro desse contexto, a proposta do presente trabalho é desenvolver uma primeira versão de um protótipo de um potenciostato de baixo custo e portátil usando

um microcontrolador ESP32, que tenha a capacidade de se comunicar com um servidor não conectado na mesma rede wi-fi. A intenção é que o mesmo possa se tornar, com melhorias e desenvolvimentos futuros, uma solução instrumental robusta para essa necessidade identificada na aplicação de técnicas que necessitam de um potenciostato.

1.2 MOTIVAÇÃO: JUSTIFICATIVA DO USO DO MICROCONTROLADOR ESP32, PYTHON E INTEGRAÇÃO COM SERVIDORES

Quando as medições com potenciostatos são feitas em laboratório é mais fácil utilizar computadores locais para armazenamento de dados. Porém, quando as medições são feitas em campo, é interessante que os dados sejam enviados por um dispositivo portátil através da Internet para um servidor. Isso faz com que as medições sejam coletadas de forma organizada e mostradas em tempo real. Essa possibilidade seria interessante principalmente no contexto industrial como, por exemplo em indústrias químicas ou de tratamento de esgoto, que têm interesse em coletar medições em tempo real de inúmeras plantas ou estações ou regiões e organizar essas informações em um servidor central, facilitando a tomada de decisões.

Hoje em dia a IOT está se tornando cada vez mais popular, pois ela permite que dados sejam coletados por sensores pequenos e enviados pela Internet. Após eles chegarem em um servidor, eles são enviados a um cliente, que pode ser um código em Python. Python é uma linguagem de programação prática e de fácil aprendizado que tem muitas bibliotecas prontas para muitas aplicações.

Para realizar a coleta de dados é necessário um dispositivo que consiga tanto fazer a leitura de valores analógicos de tensão elétrica e convertê-los para o domínio digital a fim de enviar informações, quanto injetar valores analógicos de tensão elétrica. Esse dispositivo além de precisar ter acesso a Internet para possibilitar o envio de informações de forma sem fio, também precisa ter uma frequência de clock mínima para conseguir realizar as leituras de forma frequente. O ESP32 é um microcontrolador que possui uma frequência de clock na faixa de centenas de megahertz e possui tanto ADC quanto DAC (conversores analógico-digital e digital-

analógico, usados respectivamente para ler potencial elétrico e para aplicar potencial elétrico). O microcontrolador ESP32 também conta com módulos Wi-Fi e Bluetooth. Por causa dessas características, o ESP32 foi o microcontrolador escolhido como dispositivo para realizar a coleta e envio dos dados das medições.

2. OBJETIVOS

2.1 OBJETIVO GERAL

Desenvolver um sistema de potenciostato de baixo custo e alta portabilidade baseado em um microcontrolador ESP32, integrando-o com um código em Python e servidores MQTT (Message Queuing Telemetry Transport) para coleta, análise e organização de dados de medições eletroquímicas.

2.2 OBJETIVOS ESPECÍFICOS

1 Projeto e Implementação do Potenciostato

- Projetar e construir um potenciostato utilizando amplificadores operacionais e resistores, adequado para técnicas eletroquímicas como medição de corrente elétrica no decorrer do tempo após voltametria cíclica e espectroscopia de impedância eletroquímica.
- Verificar o funcionamento do circuito com componentes comerciais e com líquidos.

2 Integração do ESP32

- Configurar o microcontrolador ESP32 para controlar o potenciostato, adquirir dados de medições e transmitir os resultados via Internet para um servidor MQTT.
- Garantir a confiabilidade na transmissão dos dados, mesmo em condições de conexão instável.

3 Desenvolvimento de Código em Python

- Criar um código em Python para receber os dados do servidor MQTT, processá-los e organizá-los em uma planilha Excel para análise posterior.
- Implementar visualização gráfica em tempo real dos dados recebidos.

4 Verificação do funcionamento do Sistema

- Comparar algum resultado obtido com um potenciostato comercial.
- Fazer testes com componentes comerciais a fim de verificar o funcionamento do sistema.

5 Exploração de Técnicas Eletroquímica

- Implementar 2 diferentes técnicas de medição.

6 Portabilidade e Aplicabilidade Industrial

- Fazer com que o sistema seja portátil.
- Facilitar o armazenamento e o acesso centralizado a dados de diversas medições, permitindo uma análise integrada e tomada de decisão mais eficiente.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 POTENCIOSTATOS

3.1.1 Conceitos Básicos: Princípios de funcionamento e aplicações

Um potenciostato aplica uma diferença de potencial elétrico entre 2 eletrodos mergulhados em um líquido e mede a corrente resultante. Um deles é o eletrodo de referência (RE), com o potencial de referência e o outro é o eletrodo de trabalho (WE), que tem potencial igual ao terra (apesar de nem sempre estar conectado diretamente ao terra). Neste trabalho, o WE não está conectado diretamente ao terra apesar de ter um potencial elétrico igual ao terra. Um terceiro eletrodo, chamado de eletrodo de controle (CE) serve para controlar o potencial elétrico do eletrodo RE.

Essa é uma configuração padrão, mas existem diversas técnicas de aplicação da tensão elétrica entre RE e WE, cada uma empregando uma forma de onda da tensão elétrica diferente. Em aplicações de eletroquímica, a corrente medida é influenciada pela corrente faradaica (a ideal) e também por uma capacitância parasita indesejada que gera um ruído de corrente elétrica.

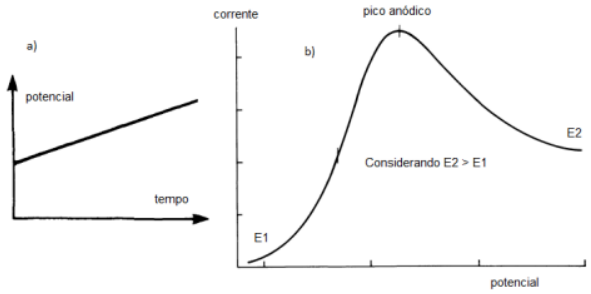
3.1.2 Principais tipos de Voltametria

3.1.2.1 Voltametria de varredura linear

Uma das técnicas usadas é a Voltametria de varredura linear. Nela, o potencial elétrico do RE varia de forma linear enquanto a corrente elétrica é medida. A ideia é encontrar o pico de corrente (pico anódico) em um determinado líquido, a fim de caracterizá-lo.

É importante notar que, como são líquidos com propriedades específicas, seu comportamento não segue o que seria esperado de um material com impedância constante, independente da tensão elétrica. Nesse caso, inicialmente aumentar a tensão elétrica pode fazer a corrente elétrica aumentar, mas em seguida ela pode começar a diminuir. A representação dessa técnica é mostrada na figura 1.

Figura 1 - Representação da voltametria com varredura linear: a) Variação do potencial elétrico em função do tempo, b) Variação da corrente elétrica em função do tempo

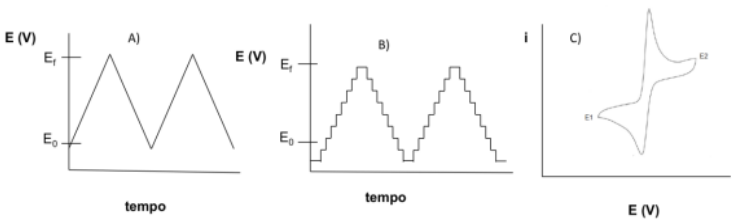


Fonte: Pacheco *et al*. (2013).

3.1.2.2 Voltametria cíclica

A voltametria cíclica consiste na aplicação de uma forma de onda de tensão elétrica triangular, coletando o valor da corrente elétrica. A representação dessa técnica é mostrada na figura 2.

Figura 2 - Representação da voltametria cíclica: a) Tensão elétrica com varredura linear b) Tensão elétrica do tipo escada. c) Voltamograma obtido para um sistema reversível



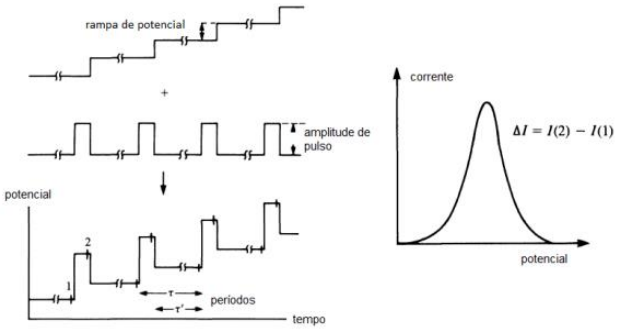
Fonte: Pacheco *et al*. (2013).

Ela pode ser em forma de varredura linear, quando a tensão aumenta de forma linear sendo uma função do tempo como uma função de 1 grau. Ou pode ser do tipo escada, sendo formada por degraus.

3.1.2.3 Voltametria de pulso diferencial

Nessa técnica se aplica uma tensão igual a uma onda quadrada mais uma espécie de rampa discreta, como na figura 3. Notar que a “rampa” só aumenta seu valor quando a onda quadrada reduz. A representação dessa técnica é mostrada na figura 3.

Figura 3 - Sinais de excitação para voltametria de pulso diferencial



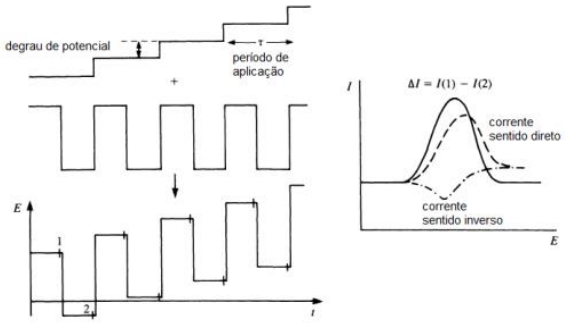
Fonte: Pacheco *et al*. (2013).

Mede-se a corrente, assim como nos outros métodos, mas há algumas diferenças: A corrente é medida apenas no final da aplicação de cada pulso da onda quadrada. Em seguida, se subtrai o valor de cada dupla de correntes (em 1 e 2 conforme mostrado na figura 3) e o gráfico plotado é o da diferença das correntes em função do potencial. A derivada em cada ponto é I2(t)-I1(t).

3.1.2.4 Voltametria de onda quadrada

Essa técnica é parecida com a voltametria de pulso diferencial e ambas reduzem significativamente o ruído capacitivo. A diferença é que ao invés de a “rampa” aumentar seu valor quando a onda quadrada reduz, ela faz isso quando a onda quadrada aumenta seu valor. Além disso, a corrente também é amostrada ao final de todos os pulsos da onda quadrada. Ambas oferecem uma velocidade maior de aquisição de dados em comparação com a voltametria cíclica. A representação dessa técnica é mostrada na figura 4.

Figura 4 - Voltametria de onda quadrada



Fonte: Pacheco *et al*. (2013).

3.1.2.5 Espectroscopia de Impedância Eletroquímica (EIS)

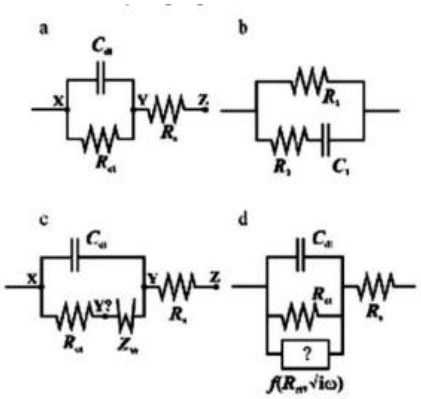
É um método que, conforme Ribeiro (2020), consiste em aplicar um sinal de tensão elétrica sinusoidal entre RE e WE e ler a impedância elétrica, tanto a parte real quanto a imaginária. Essa forma de onda sinusoidal pode ser quantizada, pois nem sempre são usados circuitos analógicos para gerar ela. Os circuitos podem ser em parte digitais e por isso o número de amostras por ciclo pode ser baixo.

3.2 MODELO DE RANDLES

Os líquidos estudados pela eletroquímica em geral são descritos por um modelo elétrico chamado Modelo de Randles. A resposta em frequência desses líquidos, ao ser injetada uma tensão elétrica (entre os eletrodos RE e WE, conectados a pontos do líquido) cuja frequência é variada, se assemelha aos circuitos abaixo, cujos valores dos componentes dependem de características dos líquidos e dos eletrodos.

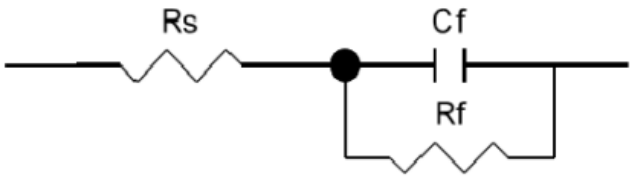
Existem modelos diferentes para situações diferentes, propostos por pesquisadores diferentes. Alguns modelos aparecem nas figuras 5, 6 e 7.

Figura 5 - Modelos equivalentes: a) para uma reação redox simples, b) forma equivalente para (a), c) circuito de Randles por reação redox com difusão, e d) forma alternativa para (c)



Fonte: Melo *et al*. (2015).

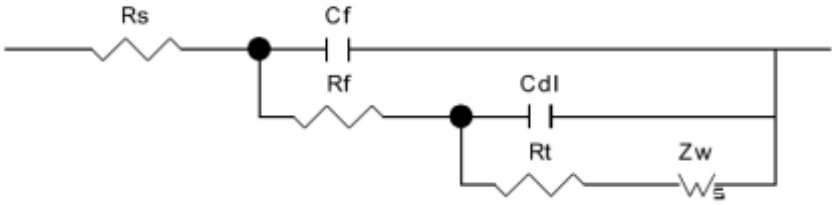
Figura 6 - Circuito Equivalente para metais com revestimentos orgânicos intactos



Fonte: Melo *et al*. (2015).

O resistor RS é a resistência da solução. Já RF depende da tinta do eletrodo ou do metal que de fato entra em contato com o líquido. Nesse caso, também há um capacitor CF. Existe também, em algumas representações de circuitos equivalentes, um elemento de fase constante, ou elemento de Warburg (ZW). Esse elemento presente em circuitos equivalentes de Randles possui a fase da impedância elétrica constante independentemente da frequência, o que foge um pouco de circuitos simples como o RC em série ou paralelo, onde a fase da impedância muda conforme a frequência. Adicionalmente, podem ser incluídos componentes para representar as contribuições de interface (Cdl) e da reação (Rct).

Figura 7 - Circuito equivalente para metais com revestimentos orgânicos com impedância de Warburg, proposto por Mansfeld



Fonte: Melo *et al*. (2015).

A partir dos diagramas de Bode e Nyquist é possível obter informações sobre esse modelo equivalente. Vale dizer que esse modelo é uma aproximação simples, não é um modelo 100% correto. Outra coisa é que se observa que o circuito tem a parte imaginária da impedância sempre igual ou maior que 0. Logo as medidas não precisam considerar se o ângulo da impedância é positivo ou negativo, ele é sempre negativo.

3.3 MICROCONTROLADOR ESP32: RECURSOS RELEVANTES, COMO COMUNICAÇÃO WI-FI, DACS E ADCS.

O microcontrolador ESP32 possui dois DACs de 8 bits, o que permite gerar até 2^8 = 256 valores diferentes de tensão elétrica entre os pinos DAC e o pino GND. Esses DACs foram utilizados para produzir o potencial elétrico de referência (RE).

Na opção de espectroscopia de impedância, a tensão gerada é alternada e aplicada em diferentes frequências. Já na análise da corrente de pico por cronoamperometria, a tensão elétrica gerada permanece constante ao longo do tempo.

O microcontrolador ESP32 também possui ADCs, que convertem os valores analógicos lidos em digitais, além de possuir a possibilidade de se conectar em uma rede wi-fi. Dessa forma, permite o uso da Internet para se comunicar com algum servidor através do protocolo MQTT.

3.4 PROCESSAMENTO DE DADOS E INTEGRAÇÃO COM PYTHON: BIBLIOTECAS UTILIZADAS PARA CONVERSÃO DE DADOS E COMUNICAÇÃO COM O SERVIDOR

A biblioteca paho.mqtt.client foi usada para se comunicar com o broker, que é um servidor usado para receber as informações do microcontrolador ESP32. Através do comando client.connect(broker, port, 60) ele se conecta ao broker (igual à variável chamada broker, com o endereço dele), na porta informada (pela variável port).

Para converter os dados recebidos pelo broker, que são em forma de texto, em uma tabela excel, o código em python usa a biblioteca from openpyxl, Workbook. Essa biblioteca permite criar ou abrir uma tabela em excel com o nome especificado e, além disso, incluir nela palavras nas células indicadas.

Pelos comandos:
wb = Workbook()
ws = wb.active
ws.title = "Dados do ESP32"
o código nomeia a planilha Excel como "Dados do ESP32".

Já pelo comando:

```
ws.append(["Mensagem MQTT", "frequencia", "POTENCIA", "corrente_rms", "tensao_rms", "potencia_aparente", "Z", "cos(phi)", "R", "X", "C", "corrente_dc" ])
```

o código inclui na planilha strings separadas por vírgula. Opcionalmente, ele pode executar esse comando inúmeras vezes, incluindo palavras uma linha após a outra da planilha.

3.5 COMUNICAÇÃO CLIENTE-SERVIDOR, PROTOCOLO MQTT

O protocolo de comunicação MQTT é usado em aplicações de IOT, onde os dados lidos por sensores são enviados para um servidor chamado de broker. Após chegar no broker, eles são coletados por um cliente.

No caso deste trabalho, os dados são enviados para o broker mosquitto, um servidor gratuito para aplicações de IOT. Isso foi feito com o uso de bibliotecas para o microcontrolador ESP32 e para Python, que permitem o envio de informações ao broker (o microcontrolador ESP32 as envia) e o recebimento das mesmas (o código em Python é configurado para recebê las).

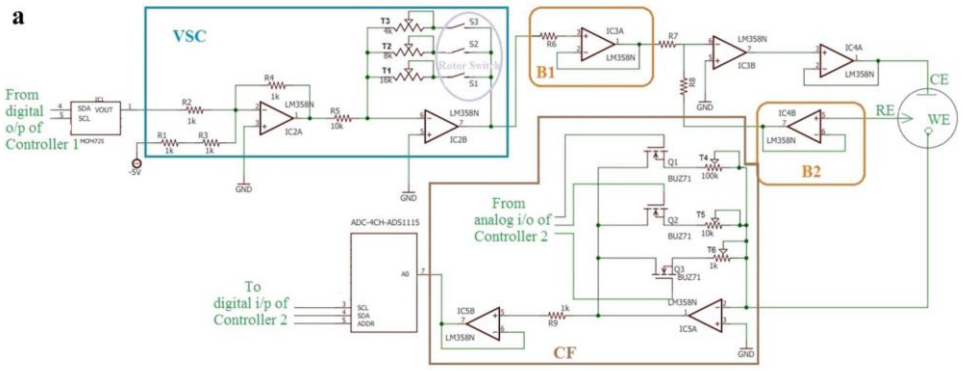
Os códigos devem especificar o endereço do broker, nesse caso “test.mosquitto.org”, na porta 1883 e no tópico “testtopic/massa”. Os tópicos servem para separar as informações de cada cliente que envia algo ao broker. No presente trabalho, não foi usada senha para acessar ao broker nem outros tipos de segurança, que poderiam ser usados para proteger os dados (criptografia por exemplo), mas são opções que podem ser usadas.

4 METODOLOGIA

4.1 DESENVOLVIMENTO DO POTENCIOSTATO

Foi usado um modelo como referência para o projeto e construção do potenciostato, que foi adaptado ao longo do trabalho. O modelo usado como referência é mostrado na figura 8.

Figura 8 - Circuito de um potenciostato de um projeto de Das *et al*. (2021)

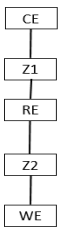


Fonte: Das *et al*. (2021).

4.1.1 Especificações do circuito

Um potenciostato normalmente possui 3 eletrodos que entram em contato com o líquido: CE, WE e RE. Simplificadamente, é como se o líquido e os eletrodos formassem o circuito equivalente da figura 9:

Figura 9 - Eletrodos CE, RE e WE e as impedâncias elétricas entre eles



Fonte: Elaborado pelo autor (2025).

CE, WE e RE são os eletrodos. Z1 e Z2 são impedâncias elétricas (entre CE e RE e entre RE e WE). A tensão elétrica entre CE e WE vai ser a necessária para que entre RE e WE tenha a tensão elétrica de referência. O potencial elétrico em WE é igual ao potencial elétrico do terra. Isso ocorre porque é conectado no pino v- de um amplificador operacional, que, por sua vez, está com o pino v+ conectado no terra, havendo uma realimentação negativa forçando que WE seja igual ao potencial elétrico do terra.

Figura 10 - Medição indireta da corrente elétrica que passa entre RE e WE, que tem que ser a mesma que passa por um outro resistor em série com a solução. Medindo a tensão elétrica entre os terminais do resistor se descobre a corrente elétrica

Fonte: Elaborado pelo autor (2025).

A corrente que passa entre RE e WE só pode circular através de um resistor que é escolhido de acordo com o máximo potencial elétrico que pode ser lido pelo microcontrolador e de acordo com a corrente entre RE e WE. Esse resistor é mostrado pela figura 10. Portanto a corrente que passa entre RE e WE passa em seguida por esse resistor. A medição da corrente entre RE e WE é feita através da tensão elétrica entre os terminais desse resistor (sendo que um dos terminais é conectado em WE e, portanto, têm potencial elétrico igual ao do terra), sabendo o valor da resistência elétrica do mesmo. Esse resistor poderia ser “acionado” através do próprio microcontrolador, que iria escolher qual resistor se conecta com o circuito, através de chaves ou optoacopladores. Porém, neste trabalho, não se usou essa

técnica, pois ao se conhecer as características do circuito pode-se escolher um valor fixo de resistência elétrica para ele.

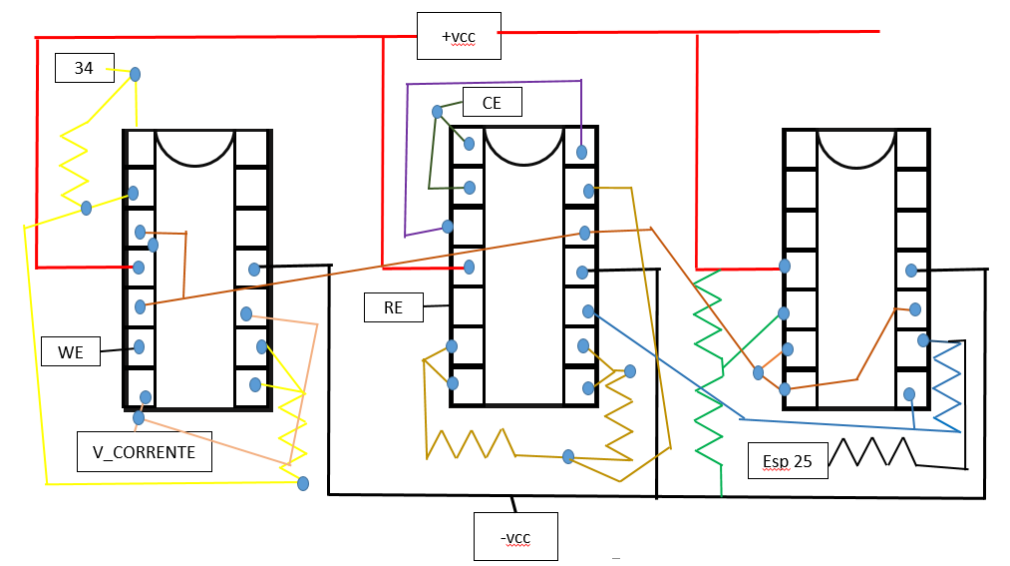
$$I_{RE-WE}(A) = \frac{-V_{resistor}(V)}{R_{resistor}(\Omega)}$$

Equação 1

$I_{RE-WE}\ (A)$ é a corrente elétrica entre RE e WE, $V_{resistor}\ (V)$ é a tensão elétrica entre os terminais do resistor (o potencial elétrico no terminal $V_{corrente}$ menos o potencial elétrico no terminal WE) usado para medir corrente elétrica e $R_{resistor}\ (\Omega)$ é a resistência elétrica do mesmo.

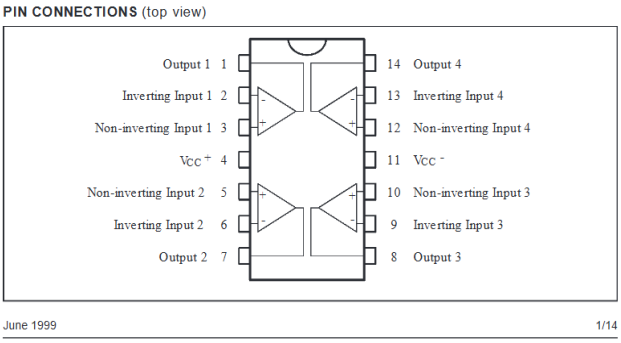
Como a tensão $V_{resistor}\ (V)$ é negativa, o sinal dela deve ser invertido através de um amplificador inversor antes de ser lida pelo microcontrolador ESP32. Através dela o ESP32 calcula a corrente entre RE e WE, seguindo a equação 1. O esquemático do potenciostato criado é mostrado pela figura 11. A figura 12 mostra a pinagem do LM324, que foi utilizado.

Figura 11 - Esquemático do circuito usado no potenciostato criado



Fonte: Elaborado pelo autor (2025).

Figura 12 - Esquemático dos pinos do LM 324, amplificadores operacionais usados



Fonte: Stmicroelectronics (1999).

Um potenciostato contém os eletrodos CE, WE e RE. A tensão elétrica entre RE e WE é controlada pelo potencial elétrico em CE, ela deve ser igual a uma referência imposta. O potencial elétrico no pino CE é o potencial necessário para que entre RE e WE haja a tensão elétrica de referência.

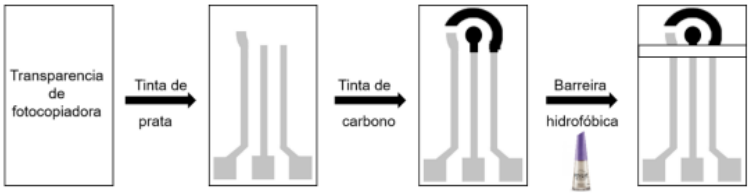
Essa tensão elétrica de referência no caso deste trabalho é oriunda da tensão elétrica entre um dos pinos do ESP32 e o pino GND. O sinal da mesma é invertido por um amplificador inversor e depois passa por um circuito comparador, comparando-a com a tensão elétrica entre RE e WE.

4.1.2 Eletrodos

Uma parte importante na realização de medidas com potenciostatos são os eletrodos, que devem ser feitos de material não suscetível à oxidação e reações químicas indesejadas. Além disso, as dimensões dele devem ser próprias para que a distância entre RE e CE seja menor do que a entre CE e WE.

As dimensões dos eletrodos afetam a medição uma vez que a corrente elétrica que passa no líquido entre RE e WE é influenciada pela impedância entre esses eletrodos, que é influenciada pela distância entre ambos.

Figura 13 - Eletrodos de carbono próprios para aplicações onde se quer evitar corrosão dos mesmos em líquidos. Uma barreira hidrofóbica evita que o líquido atinja regiões indesejadas



Fonte: Morawski (2024).

Além de serem impressos, há uma barreira aplicada, com resina usada como base de unha, para que o líquido não saia do local onde de fato estão os eletrodos (para as trilhas). O Laboratório de Catálise Molecular da Universidade Federal do Rio Grande do Sul (LAMOCA UFRGS) utiliza esses eletrodos, ilustrados na figura 13, para medir características de líquidos com um potenciostato.

Obviamente, quando a medição é em resistores e capacitores comerciais e não em líquidos, os eletrodos passaram a não ser importantes.

4.2 CONFIGURAÇÃO DO ESP32

Código para aquisição de dados:

O código para ESP32 possui 2 partes: uma delas é para o caso de se querer usar a técnica de espectroscopia de impedância eletroquímica e a outra para o caso da técnica ser a de cronoamperometria.

Porém, no caso da cronoamperometria, ele não realiza a varredura de tensões elétricas para achar a corrente de pico (como normalmente ocorreria). Como a tensão elétrica onde a corrente de pico ocorre já é conhecida, aplica-se essa tensão elétrica de forma constante e é medida a corrente elétrica ao longo do tempo, a fim de comparar com o potenciostato comercial na mesma função.

Para a espectroscopia de impedância não houve comparação com o potenciostato comercial, mas foram feitos testes com resistores, plotando a impedância real e imaginária ao longo da frequência.

Comunicação com o servidor:

O servidor usado como broker é o MQTTX, que utiliza o protocolo MQTT para receber os dados. O ESP32 é configurado para primeiro acessar o wi-fi (sabendo o nome e senha dele). Ele não se conecta em wi-fis que exigem cadastros ou tarefas mais complexas a não ser que isso seja descrito no código e também não se conecta com wi-fi de 5 GHz, apenas com 2,4 GHz. Após conectar-se ao wi-fi, ele se conecta com o broker, coleta os dados e os envia.

Foi observado que nem todos os dados chegam de fato ao broker, mas não pode ser identificado se isso é consequência de algum problema de conexão. Por isso, o código prevê que pode haver uma redundância no envio de informações, caso o usuário queira. Ou seja, nessa condição haverá o envio de uma quantidade maior que a necessária de informações, sabendo-se que algumas poderão ser perdidas.

Cálculos realizados pelo ESP32:

$V_{corrente}$ é o nome de um ponto do circuito, no qual a tensão elétrica entre ele e WE é usada para medir indiretamente a corrente elétrica entre RE e WE, por isso possui esse nome. WE por sua vez possui o mesmo potencial elétrico que o terra e que o pino GND do ESP32 (o qual é conectado ao terra).

A potência aparente é calculada pela equação 3, a potência ativa pela equação 2, o VRMS pela equação 4, e o IRMS pela equação 5. Essas integrais e multiplicações são resolvidas numericamente pelo ESP32. Como o valor da corrente elétrica no “líquido” é lido indiretamente, a informação que o ESP32 possui, na verdade, é o valor da tensão elétrica entre o pino $V_{corrente}$ e o GND. Então, o código em Python deve dividir o resultado pelo valor da resistência por onde essa corrente passa logo antes de ser medida pelo ESP32. $V(t)$ é a tensão elétrica ao longo do tempo entre RE e WE (V), enquanto V_{RMS} é o seu valor RMS (V). $I(t)$ é a corrente entre RE e WE (A), enquanto I_{RMS} é seu valor RMS (A) T é o período (s), e t é o tempo em segundos.

$$\int_0^T \frac{V(t)\cdot I(t)}{T}\cdot dt = potência\ ativa\ (W)$$

Equação 2

$$V_{RMS} \cdot I_{RMS} = potência\ aparente\ (VA)$$

Equação 3

$$\sqrt{\int_0^T \frac{V^2(t)}{T} \cdot dt} = VRMS\ (V)$$

Equação 4

$$\sqrt{\int_0^T \frac{I^2(t)}{T} \cdot dt} = IRMS\ (A)$$

Equação 5

A frequência é varrida de um valor baixo até valores altos. A cada iteração o valor da tensão elétrica aplicada entre o pino 25 do ESP32 e o GND é dada pela equação 6 onde $offset$ é o seu valor médio (V), $amplitude$ é a sua amplitude e $sin(angle)$ é o seno de um ângulo, que é proporcional ao tempo e que é dado pela equação 7, gerando uma senoide. O cálculo é feito ao longo de iterações numeradas ($iteração$) e $samplesPerCycle$ é o número de amostras por ciclo geradas.

$$offset + (amplitude * sin(angle)) = V_{25}\ (V)$$

Equação 6

$$angle = \frac{2 \cdot \pi \cdot iteração}{samplesPerCycle} \left(\frac{Rad}{s}\right)$$

Equação 7

O $samplesPerCycle$ para frequências baixas acaba sendo maior, porque o ESP32 tem um delay mínimo para ser usado. O delay entre cada iteração depende da frequência e do número de $samplesPerCycle$.

Algo importante a se considerar é que o ESP32 faz, para cada frequência, 2 varreduras: a primeira para calcular o valor médio da corrente elétrica, a segunda para fazer os outros cálculos. Isso ocorre porque o ESP32 apenas lê valores positivos de tensão elétrica e portanto a corrente elétrica lida tem valor médio diferente de 0 A. Na varredura seguinte, esse valor médio é descontado do valor lido, de forma que ele possa assumir valores negativos no cálculo.

4.3 DESENVOLVIMENTO DO CÓDIGO PYTHON

O código do ESP32 envia informações ao broker, que são acessadas pelo código em Python. O código em Python inicialmente abre uma planilha em Excel de um nome específico (definido no código), que deverá estar salva na mesma pasta que o código em Python. Em seguida o código em Python preenche a primeira linha com os nomes das variáveis que virão abaixo (corrente, potência…).

O broker, um servidor, contém as informações oriundas do ESP32 em forma de texto. O código em Python acessa esse servidor (sabendo seu endereço, nesse caso como o broker é público não necessita senha). Para evitar que informações sejam perdidas, o código presente no ESP32 pode enviar informações repetidas, redundantes ao broker.

O código em Python sabendo que informações redundantes poderão chegar, ignora uma mensagem caso ela seja igual à anterior. Sendo a mensagem válida, ele primeiro coleta a mensagem e separa as variáveis oriundas da mesma e, depois, salva essas variáveis em uma planilha Excel.

A mensagem enviada pelo microcontrolador contém o nome das variáveis seguido do valor delas (separados por “\n”). Logo após o código em Python registrar na matriz o valor da variável “potencia_aparente" ele registra o valor de todas variáveis na próxima linha da planilha em excel.

O código em Python separa as informações por cada caractere de quebra de linha (“\n”). A cada ocorrência de “\n”, o código em Python considera que há uma nova informação e acrescenta esse valor na matriz. Ele cria essa matriz onde seus valores são as informações entre os “\n” e separa as informações entre nome da variável e valor. Por exemplo, se a mensagem for: “frequencia\n15\n” ele sabe que o valor da variável frequência é 15 Hz. Há uma lógica que sabe todas as variáveis possíveis, varre a matriz e, se encontrar, nesse exemplo, a palavra “frequência”, identifica que a próxima informação é o valor da variável frequência, conforme a figura 14.

Figura 14 - Trecho de código em Python onde, após separar as mensagens oriundas do broker por cada "\n", ele separa as informações por variável (frequência, corrente RMS, …)

```
        wb.save("dados_esp32.xlsx")  # Salva a planilha
        if (dados[indice_dos_dados]=="frequencia"):
            frequencia_grafico.append(float(dados[indice_dos_dados+1]))
            frequencia_excel=float(dados[indice_dos_dados+1])
        if (dados[indice_dos_dados]=="POTENCIA"):
            POTENCIA_grafico.append(float(dados[indice_dos_dados+1])/resistencia_para_medir_corrente)
            POTENCIA_excel=float(dados[indice_dos_dados+1])/resistencia_para_medir_corrente
        if (dados[indice_dos_dados]=="corrente_rms"):
            corrente_rms_grafico.append(float(dados[indice_dos_dados+1])/resistencia_para_medir_corrente)
            corrente_rms_excel=float(dados[indice_dos_dados+1])/resistencia_para_medir_corrente
        if (dados[indice_dos_dados]=="tensao_rms"):
            tensao_rms_grafico.append(float(dados[indice_dos_dados+1]))
            tensao_rms_excel=float(dados[indice_dos_dados+1])
        if (dados[indice_dos_dados]=="potencia_aparente"):

            potencia_aparente_grafico.append(float(dados[indice_dos_dados+1])/resistencia_para_medir_corrente)
            potencia_aparente_excel=float(dados[indice_dos_dados+1])/resistencia_para_medir_corrente
            x.append(frequencia_grafico[len(potencia_aparente_grafico)-1])
```

Fonte: Elaborado pelo autor (2025).

Espectroscopia de impedância em circuitos RC:

Quando o ESP32 está na configuração de espectroscopia de impedância, ele recebe mais informações do que na configuração de injetar apenas uma tensão elétrica constante e medir apenas a corrente elétrica ao longo do tempo.

O valor de algumas variáveis é o mesmo disponibilizado pelo microcontrolador ESP32. O valor de outras variáveis é o valor disponibilizado pelo microcontrolador ESP32 dividido pelo valor de uma resistência elétrica (a que está entre WE e $V_{corrente}$).

Como nesse caso um gráfico é plotado, então as variáveis são armazenadas tanto em uma matriz, para que seja possível plotar um gráfico enquanto o código roda, quanto em uma variável não matriz para salvar numa tabela em Excel.

O quadro 1 informa como o código em Python calcula o valor de algumas variáveis:

Quadro 1 - Origem de cada variável.

Variável	Como é chamada no código	Como é calculado/fornecido
Frequência	Frequência	Direto pela mensagem
Potência ativa	POTENCIA	Valor disponibilizado pela mensagem/resistência usada para medir corrente
Corrente elétrica RMS	corrente_rms	Valor disponibilizado pela mensagem/resistência usada para medir corrente
Tensão elétrica RMS	tensao_rms	Direto pela mensagem
Potência aparente	Potência aparente	Valor disponibilizado pela mensagem/resistência usada para medir corrente

Fonte: Elaborado pelo autor (2025).

Potência ativa:

A potência ativa é dada pela equação 2, código no ESP32, porém, o ESP32 não mede diretamente a corrente, pois ele na verdade lê uma tensão elétrica proporcional a ela. O valor da corrente elétrica é o valor dessa tensão dividido por uma resistência elétrica. Porém o código do ESP32 não contém o valor dessa resistência uma vez que ao fazer a medição em diferentes líquidos ela pode ser trocada para se adequar ao range de tensões elétricas que o ESP32 pode ler. O resultado da “corrente” fornecido pelo ESP32 na verdade é da tensão elétrica entre os terminais do resistor entre $V_{corrente}$ e o WE. O cálculo da corrente elétrica propriamente dita é feito pelo código em Python. O sinal dessa tensão elétrica entre $V_{corrente}$ e WE é invertido por um amplificador inversor de ganho unitário. Portanto, ao tentar calcular conforme a equação 2, o sistema precisa fazer o cálculo conforme a

equação 8, onde $V_{corrente}$ (t) é a tensão elétrica (V) entre o ponto chamado $V_{corrente}$ e WE, e R é a resistência elétrica (Ω) usada para a medição indireta da corrente.

$$\int_0^T \frac{V(t)\cdot V_{corrente}(t)}{T\cdot R}\cdot dt = P\ (W)$$

Equação 8

Por praticidade, apenas o código em Python sabe o valor de R da equação 8. Esse valor é digitado nele, pois é mais fácil modificar o código em Python que o do ESP32 que demora a carregar o código na placa. Logo, o valor de potência ativa fornecido pela mensagem deve ser dividido por essa resistência elétrica.

Potência aparente:

É calculada pelo ESP32 pela equação 3, de forma análoga ao que ocorre com a potência ativa. O valor fornecido acaba sendo dividido por R, pois a "corrente RMS" fornecida é na verdade o valor de uma tensão elétrica e tem que ser dividido.

O código em Python calcula os valores de algumas variáveis conforme as equações 9, 10, 11, 12 e 13, com base nos valores fornecidos pela mensagem. $|Z|$ é o módulo da impedância elétrica entre RE e WE, $cos(\phi)$ é o fator de potência, ϕ é o ângulo de fase da impedância elétrica, X é a reatância entre RE e WE.

$$|Z| = \frac{V_{RMS}}{I_{RMS}}(\Omega)$$

Equação 9

$$cos(\phi) = \frac{P}{S}$$

Equação 10

$$\phi = acos(\frac{P}{S})\ (\frac{Rad}{s})$$

Equação 11

$$R = Z\cdot cos(\phi)\ (\Omega)$$

Equação 12

$$X = Z \cdot sin(\phi)\ (\Omega)$$

Equação 13

4.4 MONTAGEM E TESTES DO SISTEMA

Para verificar o funcionamento do sistema houve a comparação com um potenciostato comercial, usando as mesmas amostras e eletrodos que o potenciostato produzido neste trabalho, e também houve a comparação da resposta do potenciostato usando resistores ao invés do líquido com cálculos de circuitos com a mesma tensão elétrica e impedância elétrica.

4.5 SIMULAÇÃO NO SOFTWARE PROTEUS 8

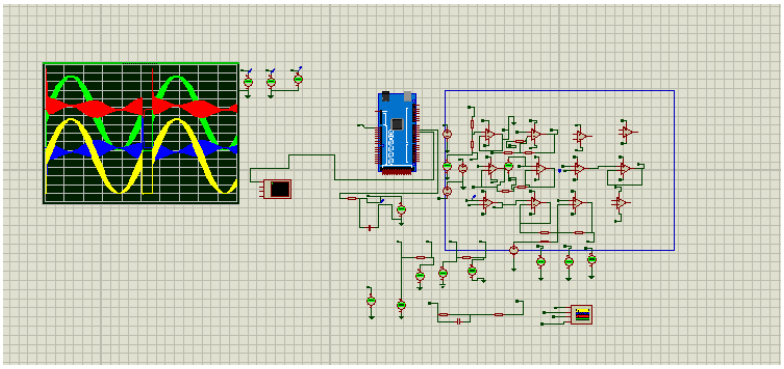
Além dos testes feitos, foram realizadas simulações no software Proteus 8, usando um circuito quase similar ao circuito montado (com alguns ajustes) e com um código ajustado. Esse teste no Proteus teve o objetivo de validar a técnica de espectroscopia de impedância. O software permite 2 tipos de simulação: a simulação dinâmica, que permite a interação com o usuário, e a simulação com o Analogue Analysis, que plota gráficos.

4.5.1 Montagem da simulação

O circuito simulado é mostrado pela figura 15. Para plotar os resultados foi utilizado o Analogue Analysis, um recurso do Proteus 8 que serviu para plotar a forma de onda de tensões elétricas importantes para o entendimento do circuito, como ilustrado pela figura 16. Para visualizar as mensagens que seriam enviadas ao broker foi usado o Virtual Terminal. Inicialmente, para simular o microcontrolador, foi instalada uma biblioteca para proteus do ESP32, mas devido a problemas com a mesma, ela foi substituída por uma biblioteca de simulação de placa Arduino Mega no Proteus, salva no local ilustrado pela figura 17. A figura 18 mostra a mesma sendo simulada. Uma vez que a placa Arduino Mega não possui DAC’s, foi usado um circuito que converte o sinal PWM que ela pode gerar em alguns pinos em um sinal analógico. Esse circuito é mostrado na figura 19. A tela do Virtual Terminal é

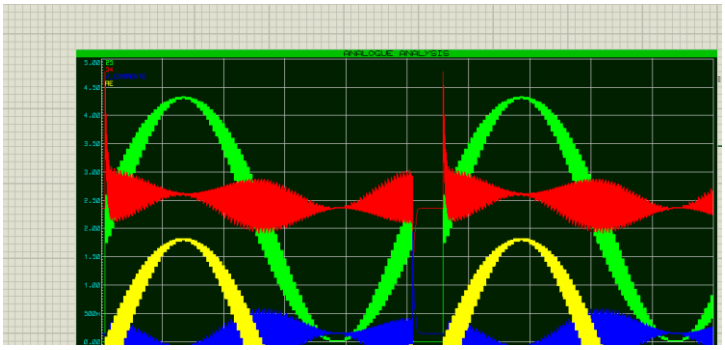
mostrada pela figura 20. Para verificar os resultados mostrados pelo Virtual Terminal, foram feitos cálculos teóricos a fim de validar o funcionamento do sistema simulado. Mais detalhes sobre a montagem da simulação se encontram no apêndice A.

Figura 15 - Simulação do circuito no software Proteus



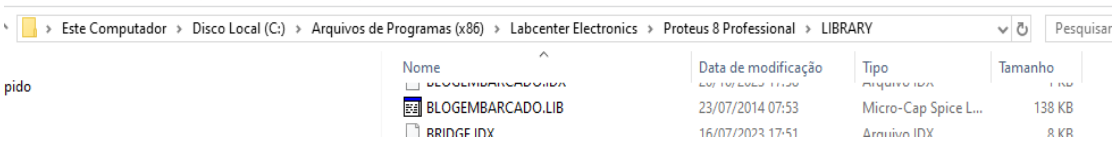
Fonte: Elaborado pelo autor (2025).

Figura 16 - Analogue Analysis, um recurso do Proteus 8 que serviu para plotar a forma de onda de tensões elétricas importantes para o entendimento do circuito. Na imagem: 25 (verde), uma tensão analógica de referência parecida com a aplicada entre RE e WE, a tensão 34 que é lida pelo Arduino (vermelho), $V_{corrente}$, que é uma tensão elétrica entre o ponto $V_{corrente}$ e WE, proporcional a corrente elétrica no "líquido" (azul) e a tensão entre RE e WE (amarelo)



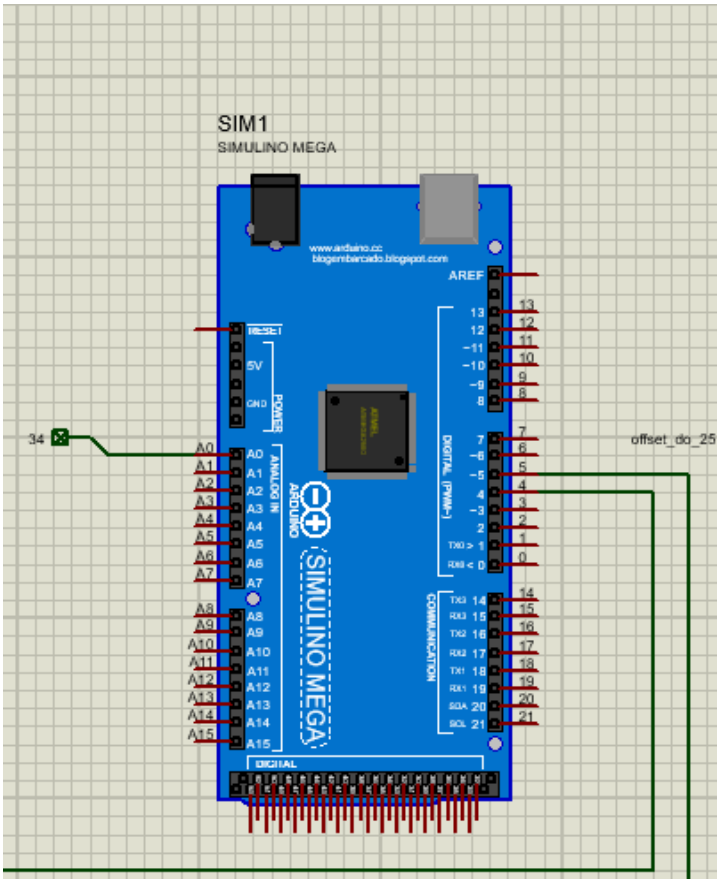
Fonte: Elaborado pelo autor (2025).

Figura 17 - Biblioteca para simular Arduino Mega no Proteus 8. Existem diversos modelos de placas para simulação no Proteus disponíveis na Internet para download (Arduino Uno, Mega,...)



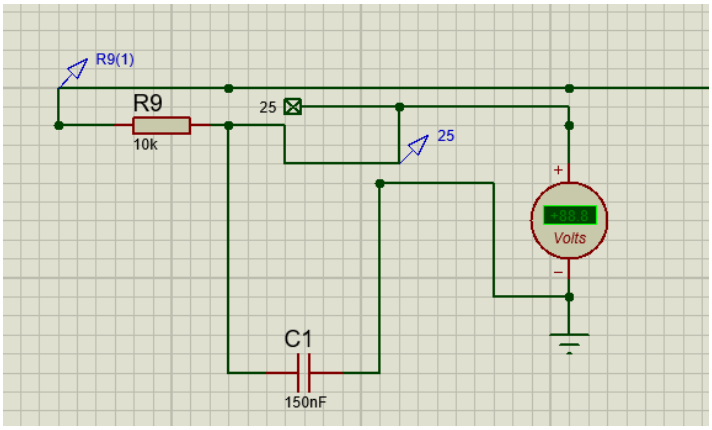
Fonte: Elaborado pelo autor (2025).

Figura 18 - Placa Arduíno mega Simulino mega simulada no Proteus 8



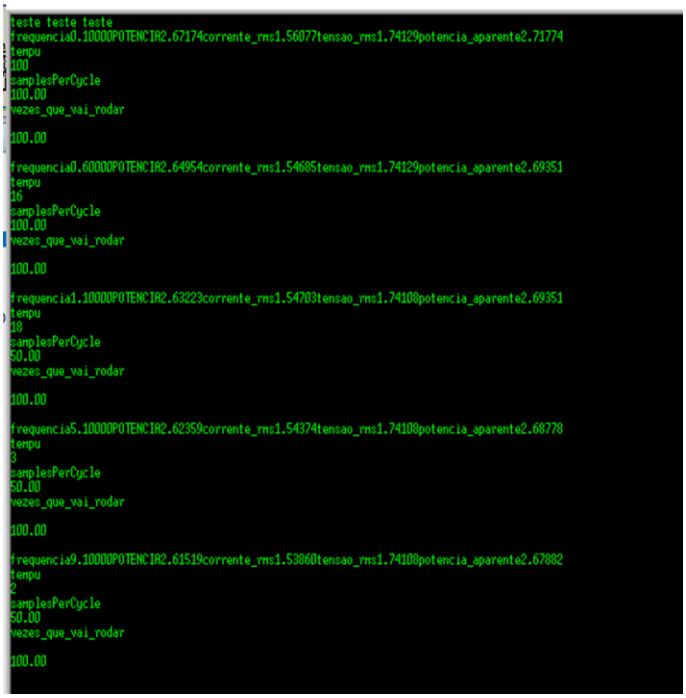
Fonte: Elaborado pelo autor (2025).

Figura 19 - Filtro após o PWM do Arduino



Fonte: Elaborado pelo autor (2025).

Figura 20 - O que aparece no Virtual Terminal quando há apenas um resistor entre RE e WE



Fonte: Elaborado pelo autor (2025).

4.5.2 Método de verificação dos resultados

Os resultados oriundos do Proteus, que aparecem no virtual terminal são: frequência, corrente RMS, tensão RMS, potência aparente, potência ativa. Alguns deles devem ser recalculados.

$$VRMS_{MEDIDA} \approx \frac{5}{2} \cdot \frac{1}{\sqrt{2}} \approx 1{,}77\ V$$

Equação 17

A tensão medida será aproximadamente o valor de pico da tensão elétrica entre o pino “ref” e o terra, que possui uma amplitude de $\frac{5}{2}$, pois é uma senoide que segue um valor que é o valor do pino 5 do arduíno somado com -2,5 V, o resultado disso com o sinal invertido (o arduino gera valores entre 0 e 5 V). Como a tensão elétrica entre o pino “ref” e o terra é uma senóide centralizada em algo próximo a 0 V, tem valor RMS igual a sua amplitude dividida por raiz de 2, conforme a equação 17. Como o arduino faz uma integral numérica não exata (ele tem uma limitação de samples por ciclo, que é o número de amostras por ciclo para calcular), pode ser que o valor medido e gerado por ele tenha algum erro numérico e, portanto, não seja uma senóide perfeita. Nas equações 18, 19 e 20, $IRMS$, $IRMS_{MEDIDA}$, P, P_{MEDIDA}, S, S_{MEDIDA} são respectivamente a corrente RMS, corrente RMS medida, potência ativa, potência ativa medida, potência aparente e potência aparente medida. A equação 21 mostra uma aproximação para VRMS. Já o valor VRMS usado na análise teórica é diferente do medido, ele é de 1,55 V.

$$IRMS\ (A) = \frac{IRMS_{MEDIDA}}{R12}$$

Equação 18

$$P\ (W) = \frac{P_{MEDIDA}}{R12} \cdot \frac{1{,}55}{1{,}77}$$

Equação 19

$$S\ (VA) = \frac{S_{MEDIDA}}{R12} \cdot \frac{1{,}55}{1{,}77}$$

Equação 20

$$VRMS\ (V) \approx 1{,}55\ V$$

Equação 21

Para calcular os valores teóricos se foram empregados os valores dos componentes usados entre RE e WE e se calculou para cada frequência a impedância elétrica equivalente entre RE e WE, comparando isso com os resultados oriundos do Virtual terminal recalculados. Mais detalhes sobre esses cálculos se encontram no apêndice A.

5 RESULTADOS E DISCUSSÃO

5.1 FUNCIONAMENTO DO POTENCIOSTATO

O potenciostato segue essencialmente o circuito padrão apresentado na figura 8, com alguns pequenos ajustes. Os componentes elétricos usados foram:

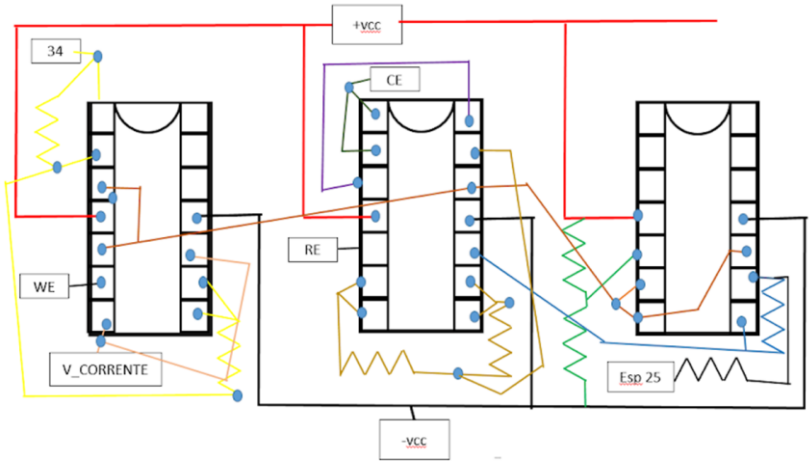
- Amplificadores operacionais, 3 chips modelo LM324N.
- 8 Resistores de 68 kΩ com tolerância de 5%.
- 3 soquetes para os chips LM324N.
- Diversos jumpers.
- Placa universal perfurada.
- Fonte de alimentação de 19 V.
- Caixa metálica com furos.

A figura 21 mostra o esquemático usado para o potenciostato montado. Ele, diferentemente da simulação, injeta tensões elétricas apenas positivas entre RE e WE. Isso se justifica pelo fato de que a montagem foi usada para verificar mais a parte da comunicação, para validar o funcionamento do circuito e código de forma geral. A simulação foi usada para demonstrar o funcionamento do sistema de forma mais reprodutível e simples, uma vez que os arquivos são mais facilmente verificados do que um modelo construído. Para fazer com que na montagem fossem injetadas tensões elétricas negativas, seria necessário seguir o mesmo esquemático da simulação, com uma única diferença: como fontes de tensão de 2,5 V e -2,5 V não são comuns comercialmente, deveria-se usar divisores de tensão e buffers de tensão elétrica, associados a alimentação do circuito (9,5 V em VCC e -9,5 V em -VCC).

O sistema montado possui os nós CE (eletrodo de controle), RE (eletrodo de referência), WE (eletrodo de trabalho), 34 (conectado ao pino 34 do microcontrolador ESP32 para a leitura de tensão elétrica, com o objetivo de determinar a corrente elétrica entre RE e WE), 25 que é um pino do microcontrolador ESP32 com DAC (conversor digital analógico) o qual injeta um sinal que é a tensão elétrica desejada entre RE e WE. $V_{corrente}$ é um nó que possui uma tensão elétrica (entre ele e o terra)

proporcional à corrente elétrica entre RE e WE. +VCC e -VCC são nós referentes à alimentação (9,5 V e -9,5 V em relação ao terra).

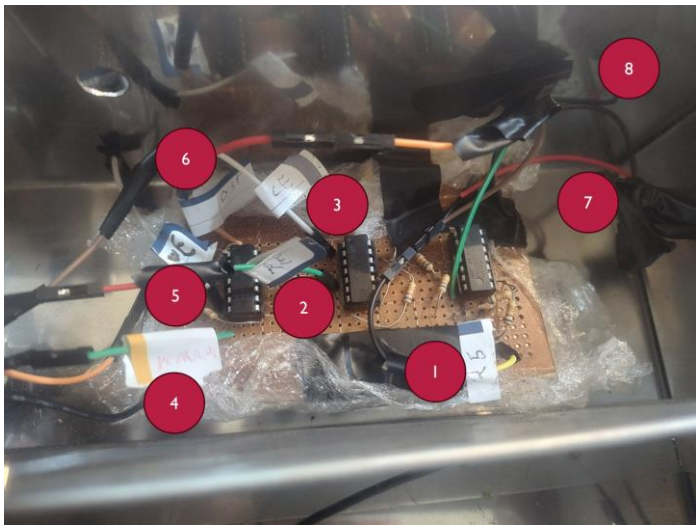
Figura 21 - Esquemático da montagem



Fonte: Elaborado pelo autor (2025).

-

Figura 22 - Placa montada, dentro de uma caixa metálica. A numeração indica as conexões e pontos de medida, que são descritos no texto.



Fonte: Elaborado pelo autor (2025).

A figura 22 mostra a placa montada, com o jumper para se conectar com o pino 25 do microcontrolador ESP32 (1), o jumper para se conectar com o eletrodo RE (2), o jumper para se conectar com CE (3), o nó $V_{corrente}$ (4), para medição de uma tensão elétrica, proporcional a corrente elétrica entre RE e WE e o jumper para

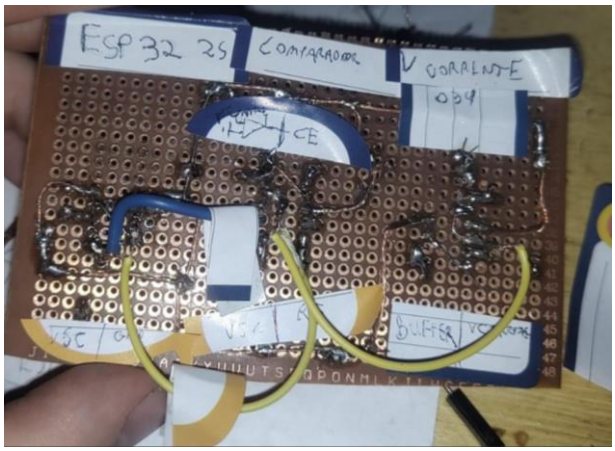
se conectar com WE (5). A leitura do microcontrolador ESP32 ocorre no nó 34 (6), conectado em seu pino 34. A alimentação está nos nós +VCC (7) e -VCC (8). A figura 23 mostra a fonte de alimentação de 19 V utilizada.

Figura 23 - Fonte de alimentação de 19 V

Fonte: Elaborado pelo autor (2025).

A figura 24 mostra detalhes da parte traseira da placa, que foi colocada em uma caixa de inox (figura 22), com um material polimérico atrás (papel filme), evitando o contato elétrico com a carcaça da caixa. O terra do circuito é a média entre o fio positivo e o negativo da fonte de alimentação e é conectado à carcaça caixa de inox. A caixa de inox possui portabilidade e apesar de durante os testes o ESP32 ter sido alimentado por um computador, ele também pode ser facilmente alimentado por uma bateria, fazendo com que o dispositivo seja portátil.

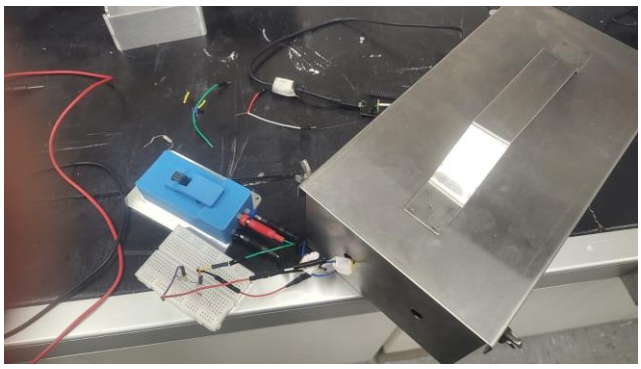
Figura 24 - Parte traseira da placa montada, com os componentes



Fonte: Elaborado pelo autor (2025).

O potenciostato construído é mostrado na figura 25. Seu desempenho foi comparado com o potenciostato comercial da figura 26 , como descrito na seção 5.3, utilizando para isso os eletrodos de carbono ilustrados na figura 11, conforme a montagem apresentada na figura 28. As medidas que o potenciostato comercial faz são exemplificadas pela figura 27.

Figura 25 - Potenciostato sem os eletrodos de carbono, conectado a uma protoboard para ter um resistor entre RE e $V_{corrente}$ e um conector

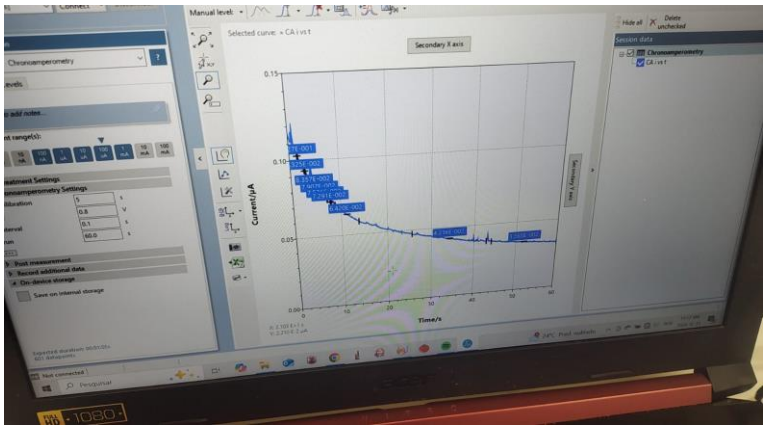


Fonte: Elaborado pelo autor (2025).

Figura 26 - Potenciostato comercial

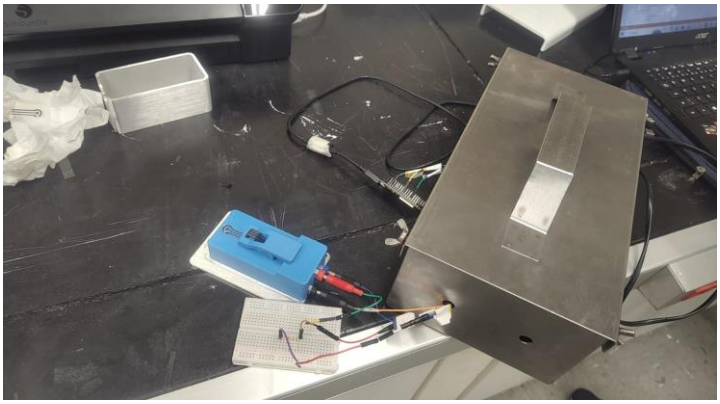
Fonte: Elaborado pelo autor (2025).

Figura 27 - Exemplo de medição feita pelo potenciostato comercial



Fonte: Elaborado pelo autor (2025).

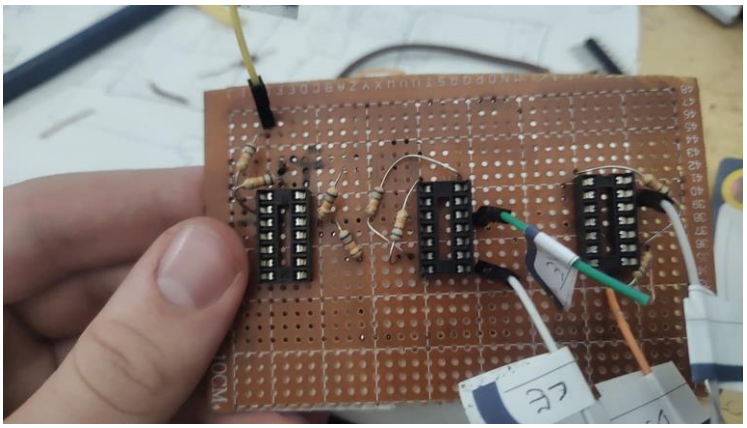
Figura 28 - Potenciostato com os eletrodos de carbono e um conector compatível (em tamanho) conectado a eles



Fonte: Elaborado pelo autor (2025).

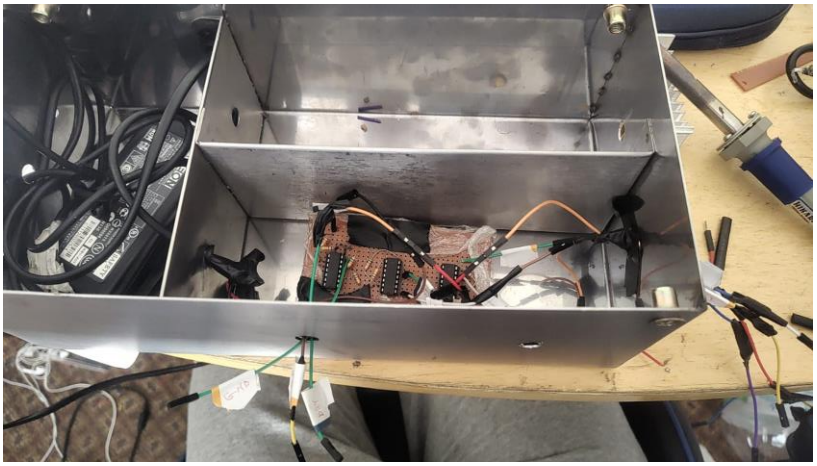
É importante chamar a atenção sobre o resistor que pode ser visto fora da caixa de aço inox, que deve ser escolhido após ser conhecido o valor da corrente. O valor da sua resistência não deve ser pequeno demais, para a sensibilidade do sensor não ser baixa, nem grande demais, para que não seja excedida a tensão máxima aceita pelo microcontrolador (entre o pino usado para leitura e o GND). Nas figuras 29 e 30 são mostrados detalhes adicionais da parte frontal da placa usada, bem como da montagem final do protótipo na caixa de aço inox.

Figura 29 - Parte frontal da placa usada no potenciostato, sem os amplificadores operacionais LM324



Fonte: Elaborado pelo autor (2025).

Figura 30 - Caixa de inox contendo a fonte de alimentação de 19 V e a placa usada no potenciostato



Fonte: Elaborado pelo autor (2025)

5.2 DADOS NO SERVIDOR E CONVERSÃO DE DADOS COM PYTHON

Os dados do ESP32 chegaram devidamente ao servidor, isso foi verificado através do código em Python que recebeu os mesmos. Os dados foram recebidos pelo código em Python e foi gerada a tabela em excel, conforme a figura 31.

Figura 31 – Tabela em Excel gerada.

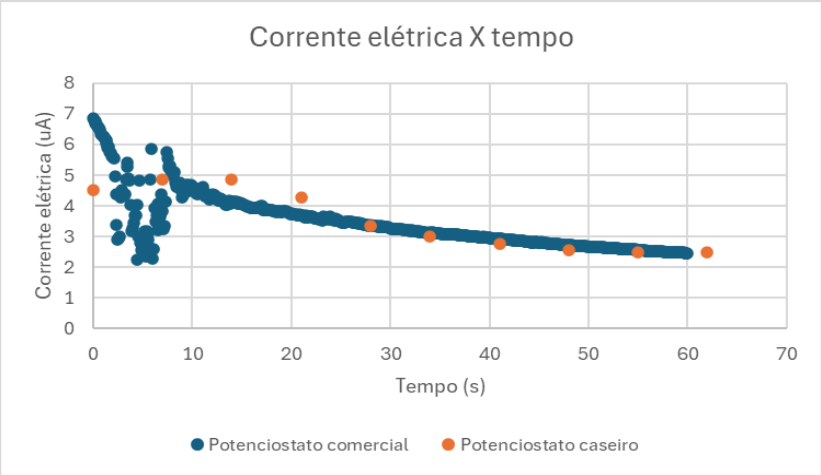
Mensager	frequenci	POTENCIA	corrente_	tensao_rn	potencia_	Z	cos(phi)	R	X	C	corrente_dc
48.9351	0.6	1.55E-05	1.35E-05	1.17042	1.58E-05	86864.77	0.9819	79400	15315.55	1.73E-05	0
89.66966	1.1	1.63E-05	1.41E-05	1.17042	1.65E-05	83035.79	0.988501	79400	12146.12	1.19E-05	0
126.395	2.1	1.61E-05	1.39E-05	1.17042	1.63E-05	84031.29	0.989139	79400	11798.68	6.42E-06	0
163.6175	3.1	1.62E-05	1.41E-05	1.17042	1.65E-05	83202.67	0.986182	79400	13337.99	3.85E-06	0
205.5491	5.1	1.61E-05	1.4E-05	1.17042	1.63E-05	83889.02	0.986663	79400	13099.29	2.38E-06	0
238.9504	6.1	1.63E-05	1.41E-05	1.17042	1.65E-05	83178.15	0.98799	79400	12417.8	2.1E-06	0
271.8627	7.1	1.61E-05	1.39E-05	1.17042	1.63E-05	84298.97	0.989358	79400	11677.29	1.92E-06	0
347.6497	12.12	1.61E-05	1.39E-05	1.17042	1.63E-05	84202.95	0.986658	79400	13101.65	1E-06	0
381.8972	14.544	1.61E-05	1.39E-05	1.17042	1.63E-05	83973.72	0.989282	79400	11719.66	9.34E-07	0
415.4606	17.4528	1.62E-05	1.39E-05	1.17042	1.63E-05	84092.73	0.992399	79400	9846.037	9.26E-07	0
455.5179	30.15844	1.59E-05	1.38E-05	1.17042	1.62E-05	84795.45	0.985381	79400	13727.81	3.84E-07	0
496.6637	43.42815	1.59E-05	1.38E-05	1.17042	1.61E-05	84851.06	0.985872	79400	13490.06	2.72E-07	0

Fonte: Elaborado pelo autor (2025).

5.3 COMPARAÇÃO COM SISTEMAS COMERCIAIS

O modelo de potenciostato comercial usado é o PalmSens4, presente no LAMOCA, laboratório da UFRGS. Esse potenciostato aplicou o método de cronoamperometria para encontrar a tensão elétrica onde ocorre a corrente de pico em uma solução de água da torneira com álcool 60%. Vale ressaltar que a condutividade provavelmente foi muito mais afetada pela quantidade de sal na água do que pela quantidade de álcool, uma vez que o sal forma íons em solução aquosa. A comparação é mostrada na figura 32.

Figura 32 - Comparação da medição de corrente elétrica no tempo entre o potenciostato comercial (azul) e o potenciostato deste trabalho (laranja). Com a tensão elétrica CC de 0,8 V entre RE e WE



Fonte: Elaborado pelo autor (2025).

O resultado final dessa comparação é dado pela figura A-9, onde pode-se concluir que o potenciostato tem um desempenho de acordo com o esperado para a situação à qual foi submetido, gerando um resultado compatível com o do potenciostato comercial. Porém o número de comparações dele com o potenciostato comercial não é suficiente para validar o dispositivo como aplicação final. As diferenças entre o valor medido pelo protótipo e pelo potenciostato comercial foram de 2,32 µA, 1,04 µA, 2,75 µA, 0,56 µA, 0,04 µA, 0,13 µA, 0,18 µA, 0,15 µA, 0,07 µA em ordem de tempo (foi reduzindo com o tempo e o que mais importa nesse tipo de medição é o valor final). O último valor lido pelo protótipo foi aos 62 s desde o início da medição, sendo que o potenciostato comercial apenas leu até os 60 s. As medições feitas pelo protótipo foram: 4,53 µA (0 s), 4,85 µA (7 s), 4,85 µA (14 s), 4,26 µA (21 s), 3,34 µA (28 s), 3,00 µA (34 s), 2,76 µA (41 s), 2,57 µA (48 s), 2,50 µA (55 s), 2,49 µA (62 s). O intervalo entre uma medição e outra foi um pouco menos de 7 s.

5.4 COMPARAÇÃO COM CARGA RESISTIVA

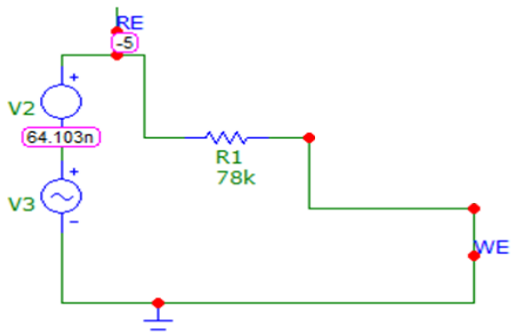
Foi feito um teste prático com espectroscopia de impedância, com componentes resistivos entre RE e WE. Adicionalmente, foram feitas simulações no Proteus 8.

5.4.1 Fonte prevista de erro

A tensão elétrica injetada pelo sistema entre RE e WE foi uma senóide não centralizada em 0 V. Para corrigir isso poderiam ter sido acrescentados circuitos amplificadores somadores de ganho unitário a ela, somando com a tensão necessária para centralizar essa tensão em 0 V. Algo parecido poderia ter sido feito antes de ler o resultado. A simulação feita no Proteus 8 foi feita já com essa correção.

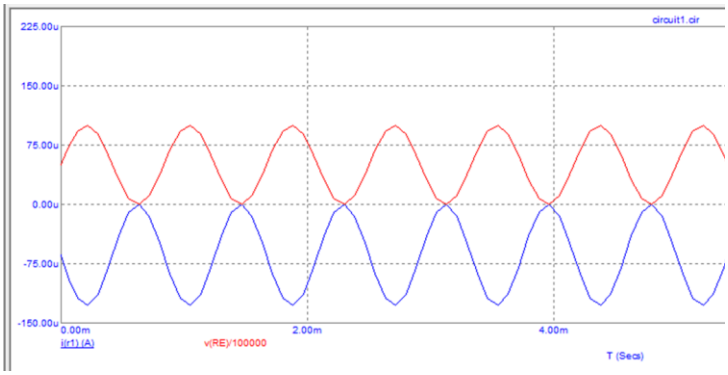
É importante chamar a atenção, se fosse um teste com um capacitor entre os componentes elétricos entre RE e WE, a corrente elétrica não se comportaria da mesma forma que a tensão elétrica entre ambos (com o pico superior em 3,3 V e o inferior em 0 V), pois um de seus picos não seria em 0 A, seria deslocado. Isso pode ser mostrado por simulações no Microcap 12. Na figura 33 há um circuito cuja impedância equivalente entre RE e WE é apenas resistiva, na figura 34 são mostradas formas de onda resultantes desse circuito. Na figura 35 há um circuito que possui um capacitor entre os componentes que estão entre RE e WE e na figura 36 são mostradas formas de onda resultantes desse circuito.

Figura 33 - Circuito do teste prático, no Microcap12



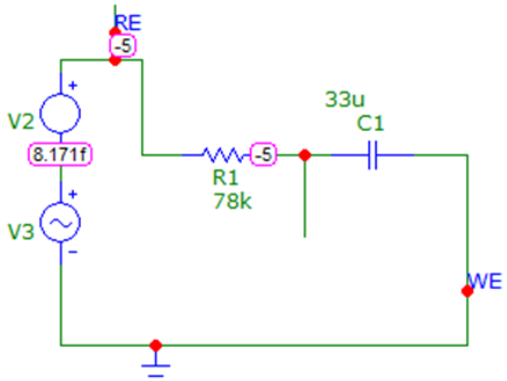
Fonte: Elaborado pelo autor (2025).

Figura 34 - Forma de onda esperada de tensão e corrente no circuito de teste da figura 33, no Microcap12. Em azul a corrente elétrica e em vermelho a tensão elétrica dividida por valor arbitrário, para melhor visualização



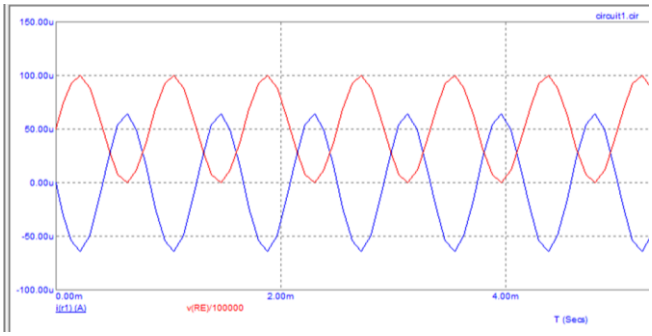
Fonte: Elaborado pelo autor (2025).

Figura 35 - Circuito do teste hipotético usando um capacitor, no Microcap12



Fonte: Elaborado pelo autor (2025).

Figura 36 - Forma de onda esperada de tensão e corrente no circuito de teste da figura 35, no Microcap12. Em azul a corrente elétrica e em vermelho a tensão elétrica dividida por algum valor, para melhor visualização



Fonte: Elaborado pelo autor (2025).

Como é possível ver na figura 36, a corrente em circuitos com resistores e capacitores em série é centralizada em 0 A. Apesar da tensão elétrica entre RE e WE não ser centralizada em 0 V. Como não houve ajuste na leitura do ESP32, se esperaria que o valor RMS da corrente elétrica medida fosse diferente do esperado. Como há leitura de meio ciclo de onda, seu valor RMS reduz para $\frac{1}{\sqrt{2}}$ vezes seu valor original. O ESP32 não lê valores negativos.

5.4.2 Resultado do teste

Um dos testes do potenciostato foi com carga resistiva, no intuito de testar a técnica de espectroscopia de impedância. No lugar dos eletrodos com o líquido foram colocados resistores. A figura 37 mostra o circuito e a figura 38 mostra a representação dele no Microcap 12. Durante o teste, o potenciostato aplicou diferentes frequências com a mesma amplitude e foram medidas grandezas elétricas, registradas em um arquivo Excel. A tabela 1 mostra os dados oriundos da planilha em Excel que foi gerada pelo código em Python, apenas as primeiras colunas, que foram comparadas com os resultados esperados. A coluna mais à esquerda da tabela 1 mostra a frequência (Hz), a segunda coluna mais à esquerda mostra a potência ativa (chamada de “POTENCIA” pelo código) em W, a terceira coluna mostra a corrente elétrica RMS em A, a quarta coluna mostra a tensão elétrica RMS em V e a quinta coluna mostra a potência aparente em VA.

Tabela 1. Informações que o código em Python salvou em Excel (com nomenclaturas do código em Python).

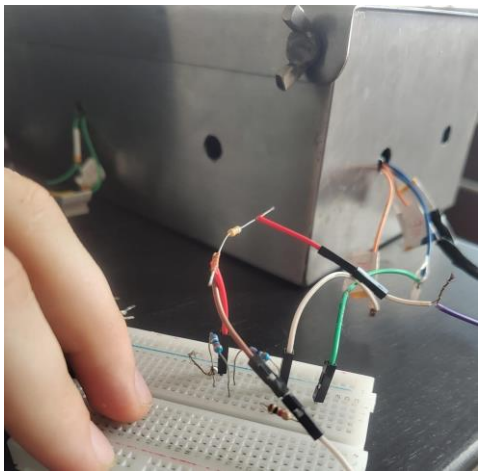
Frequência	POTENCIA	corrente_rms	tensao_rms	potencia_aparente
1.1	1.60616E-05	1.42265E-05	1.14912	1.63479E-05
2.1	1.61507E-05	1.43474E-05	1.14912	1.64868E-05
3.1	1.60093E-05	1.41944E-05	1.14912	1.6311E-05
4.1	1.59931E-05	1.42669E-05	1.14912	1.63944E-05

660.9042	1.58226E-05	1.41709E-05	1.1481	1.62696E-05
726.9947	1.59853E-05	1.42871E-05	1.1481	1.64028E-05
799.6942	1.59796E-05	1.42826E-05	1.1481	1.63979E-05
879.6636	1.60028E-05	1.41247E-05	1.15112	1.62593E-05
1699.052	1.60379E-05	1.4356E-05	1.14851	1.64881E-05
1716.042	1.59571E-05	1.41669E-05	1.14851	1.62709E-05
1733.203	1.58447E-05	1.41916E-05	1.14851	1.62993E-05
1750.535	1.5694E-05	1.40454E-05	1.14851	1.61313E-05
30.15844	1.62735E-05	1.43118E-05	1.14851	1.64371E-05
36.19013	1.58343E-05	1.38856E-05	1.14851	1.59476E-05
43.42815	1.57031E-05	1.45631E-05	1.14851	1.67259E-05
52.11378	1.57468E-05	1.38293E-05	1.14851	1.58831E-05
30.15844	1.60622E-05	1.42635E-05	1.14912	1.63904E-05
36.19013	1.60746E-05	1.43315E-05	1.14912	1.64687E-05
43.42815	1.61184E-05	1.43004E-05	1.14912	1.64329E-05
52.11378	1.60475E-05	1.42034E-05	1.14912	1.63213E-05
1311.75	1.59303E-05	1.41154E-05	1.14851	1.62118E-05
1324.867	1.58775E-05	1.40584E-05	1.14851	1.61462E-05
1338.116	1.59525E-05	1.41922E-05	1.14851	1.62999E-05
1351.497	1.59884E-05	1.41556E-05	1.14851	1.62579E-05
3.1	1.64334E-05	1.55959E-05	1.14851	1.79121E-05
4.1	1.61138E-05	1.42365E-05	1.14851	1.63507E-05
5.1	1.51344E-05	1.3949E-05	1.14851	1.60204E-05
6.1	1.55331E-05	1.36272E-05	1.14851	1.5651E-05
5.1	1.59082E-05	1.41091E-05	1.14912	1.62129E-05
6.1	1.58918E-05	1.41372E-05	1.14912	1.62453E-05

7.1	1.59785E-05	1.41843E-05	1.14912	1.62993E-05
8.1	1.60716E-05	1.42121E-05	1.14912	1.63313E-05
9.1	1.59688E-05	1.4239E-05	1.14912	1.63622E-05
10.1	1.59787E-05	1.43275E-05	1.14912	1.6464E-05
1141.173	1.57975E-05	1.41631E-05	1.14851	1.62665E-05
1152.585	1.5764E-05	1.39879E-05	1.14851	1.60654E-05
1164.111	1.61529E-05	1.46949E-05	1.14851	1.68772E-05
1175.752	1.58859E-05	1.41679E-05	1.14851	1.62721E-05
1187.51	1.59163E-05	1.42201E-05	1.14851	1.63321E-05
1.1	1.64609E-05	1.47599E-05	1.14851	1.69519E-05
2.1	1.56466E-05	1.44999E-05	1.14851	1.66532E-05
3.1	1.59037E-05	1.39638E-05	1.14851	1.60376E-05
4.1	1.54656E-05	1.38769E-05	1.14851	1.59378E-05
0.1	1.56841E-05	0.00001415	1.14912	0.00001626
0.6	0.00001586	1.42043E-05	1.14912	1.63224E-05
1.1	1.59269E-05	1.41825E-05	1.14912	1.62972E-05

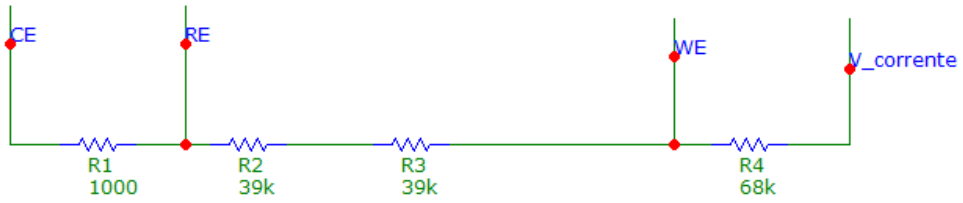
Fonte: Elaborado pelo autor (2025).

Figura 37 - Exemplo de combinação de elementos resistivos em série entre RE e WE



Fonte: Elaborado pelo autor (2025).

Figura 38 - Combinação de elementos usados no teste



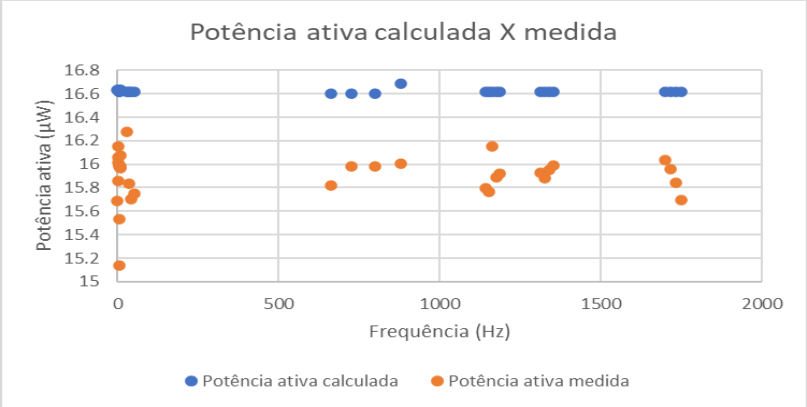
Fonte: Elaborado pelo autor (2025).

.

Entre os fios CE e RE há um resistor de valor pequeno, de 1 kΩ. Entre o RE e o WE há uma resistência elétrica equivalente de 78 kΩ. A resistência elétrica usada para medir indiretamente a corrente (pela tensão elétrica entre seus terminais) é de 68 kΩ.

A potência ativa esperada em carga puramente resistiva é a tensão RMS medida ao quadrado dividida pela resistência elétrica. Na figura 39, é apresentada a comparação entre os valores da potência ativa calculada dessa forma e aqueles medidos, considerando os dados da tabela 1. Como na tabela 1 há valores repetidos de frequência, se considerou apenas os dados correspondentes aos primeiros valores medidos para uma dada frequência.

Figura 39 - Potência ativa calculada X medida no teste



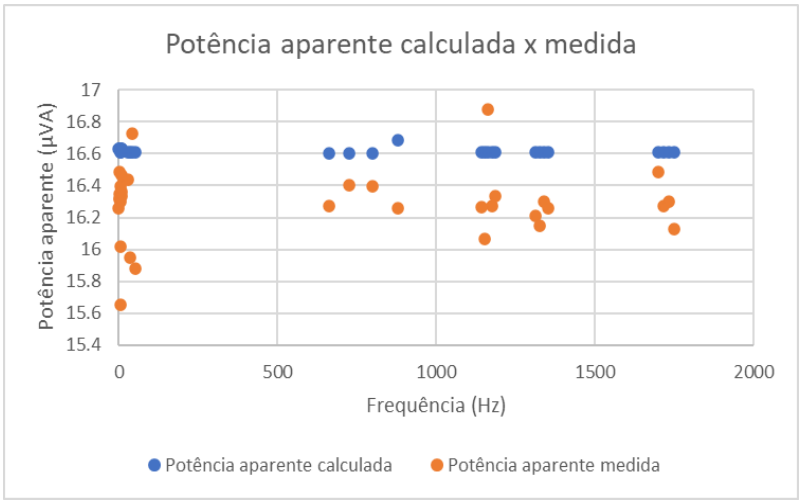
Fonte: Elaborado pelo autor (2025).

Na figura 39 o desvio máximo observado entre os valores calculados e medidos foi de 1,48 µW. Esse desvio ocorre devido a vários motivos, como, por exemplo, o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza e a faixa de tolerância alta (5%) dos componentes utilizados.

A potência ativa calculada foi calculada com base na tensão RMS medida e não foi sempre constante (apesar de se esperar que ela não deveria mudar em função da frequência), embora a variação tenha sido baixa.

A potência aparente calculada é igual à ativa, uma vez que a carga é puramente resistiva. A comparação da mesma com as medições da tabela 1 é mostrada pela figura 40. Da mesma forma que na figura 39, como na tabela 1 há valores repetidos de frequência, foram considerados apenas os primeiros valores medidos para uma dada frequência.

Figura 40 - Potência aparente calculada X medida no teste



Fonte: Elaborado pelo autor (2025).

Na figura 40 o desvio máximo observado entre os valores calculados e medidos foi de 0,962 µVA. Assim como no caso da potência ativa, esse desvio ocorre devido a vários motivos, como o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza e a alta faixa de tolerância (5%) dos componentes.

5.6 SIMULAÇÕES NO SOFTWARE PROTEUS 8

Os resultados das simulações bem como os circuitos simulados se encontram no apêndice B. Eles demonstram alguma similaridade com os resultados esperados (calculados), embora não sejam 100% iguais. Isso se deve provavelmente às aproximações feitas nos cálculos feitos pelo microcontrolador (simulado), que podem resultar em erros numéricos. Essas aproximações são feitas porque é muito difícil, senão impossível, realizar uma integral numérica ao longo do tempo em forma de somatório com dt tendendo a 0, o intervalo de tempo é sempre finito e discreto nesses microcontroladores. Além disso, as leituras possuem uma resolução limitada.

Nas 4 simulações feitas usando capacitor, no geral, as diferenças entre os valores calculados e os simulados são maiores para baixas frequências e essas diferenças diminuem para frequências mais elevadas. Em maiores frequências o valor lido se estabiliza. Esse comportamento pode ser verificado nas figuras B-16 e B-23. Logo há alguma espécie de atraso na simulação, apesar disso, verifica-se grande similaridade entre os valores calculados e os simulados. O comportamento teórico da potência aparente (µVA), corrente RMS (µA) e da potência ativa (µW) é de aumentar com a frequência nos circuitos com capacitor, algo semelhante também ocorre nos resultados das simulações. O gráfico da simulação mais condizente com os valores calculados é ilustrado pela figura B-28, pois é do circuito que contêm apenas resistores, nesse caso a potência aparente (µVA), corrente RMS (µA) e a potência ativa (µW) independem da frequência.

6 CONCLUSÃO

6.1 RESUMO DOS RESULTADOS: CONEXÃO ENTRE OS OBJETIVOS E O QUE FOI ALCANÇADO

Apesar do sensor construído precisar de mais testes para ser melhor caracterizado e confirmar sua repetibilidade em diferentes condições de temperatura e concentrações das soluções medidas, foi desenvolvido um protótipo de potenciostato de baixo custo com integração com servidor MQTT para coleta de dados e posterior análise em um código em Python e uma planilha em Excel, o que é uma interessante forma de medição com uso de potenciostatos. O que é mais comum nesse tipo de aplicação é a conexão do ESP32 no wi-fi, enviando informações para dispositivos conectados na mesma rede, como no trabalho de Sousa (2021). O sistema desenvolvido no presente trabalho é portátil e, portanto, próprio para aplicações em campo.

A técnica de Espectroscopia de impedância eletroquímica foi verificada por simulação e por um teste prático e a técnica de cronoamperometria foi verificada por um teste prático. O potenciostato foi validado com diferentes componentes entre RE e WE através de simulação e também de forma prática, embora o número de testes ainda seja pequeno para garantir que possa ser usado como aplicação final. Em caso de conexões instáveis no código do microcontrolador ESP32 há a possibilidade de envio de mensagens de forma repetida. Nesses casos, o código em Python acaba ignorando os valores redundantes. O código em Python permite a visualização gráfica em tempo real, embora a mesma possa ainda ser melhorada. O protótipo de potenciostato foi comparado com um comercial, e o resultado dessa comparação demonstra similaridade entre as medidas feitas pelo potenciostato comercial e o potenciostato construído. Portanto, de forma geral, os objetivos foram alcançados embora o número reduzido de testes e em poucas condições diferentes seja ainda insuficiente para o validar o instrumento com a precisão mínima necessária para suas aplicações. Além disso, vale ressaltar que embora haja um alto potencial para a aplicação industrial de um instrumento com as características do protótipo que foi construído, para viabilizar essa aplicação, vários aprimoramentos seriam necessários como, por exemplo, o desenvolvimento da placa em algum

software de PCB (placa de circuito impresso). Embora haja potencial para haver maior aplicação industrial com posterior desenvolvimento.

Outro ponto importante para evoluir na direção de uma aplicação industrial seria a realização de testes em condições de tensões e correntes elétricas mais baixas do que os valores apresentados. Nesses casos podem haver projetos de circuitos mais precisos para tal. Além disso, deve ser eliminada a limitação da montagem atual, que não aplica tensões elétricas negativas entre RE e WE, o que seria facilmente resolvido com a adição de amplificadores somadores de ganho unitário.

Algumas sugestões para trabalhos futuros:

- Uso de mais funcionalidades do microcontrolador ESP32, evitando que a frequência seja prejudicada com os atrasos desnecessários.
- Implementação de dashboards, para visualizar de forma gráfica com melhor qualidade, e de interfaces mais amigáveis para o usuário.
- Correção de offset nos amplificadores operacionais e também dos diferentes ruídos presentes neles, a fim de tornar o circuito mais confiável.
- Fazer a montagem conforme a simulação, com a tensão elétrica entre RE e WE centralizada em 0 V, ao invés de um valor positivo elevado.

7 REFERÊNCIAS

ALMEIDA, Adrianne Veras de; SOARES, Maria Valquíria Maia. **Internet das coisas aplicada na educação: um mapeamento sistemático da literatura**. 2019. Monografia (Graduação) - Licenciatura em Computação, UFRA (Universidade Federal Rural da Amazônia), Amazônia (AM), 2019.

DAS, Abhranila *et al*. HOME-Stat: a handheld potentiostat with open-access mobile-interface and extended measurement ranges. **Proceedings of the Indian National Science Academy**, v. 87, n. 1, p. 84-93, 2021.

MARTINS, Victor Ferreira. **Automação residencial usando protocolo MQTT, Node-RED e Mosquitto Broker com ESP32 e ESP8266**. 2019. Monografia (Graduação) - Engenharia de Controle e Automação, UFU (Universidade Federal de Uberlândia), Minas Gerais (MG), 2019.

MELO, Rodrigo de S.; MAIA, Fernanda T.M. Avaliação do Uso de Circuitos Equivalentes na Análise por Impedância Eletroquímica de Revestimentos Anticorrosivos. **Revista de Engenharias da Faculdade Salesiana** , n. 02, pág. 2-9, 2015.

MORAWSKI, Rodrigo. **Estudo de nanotubos de polipirrol e nanocompósitos com Ag/AgCl para o dignóstico de fibrose cística utilizando eletrodos impressos**. 2014. Dissertação (Mestrado) – Programa de Pós-Graduação em Ciências dos Materiais, UFRGS (Universidade Federal do Rio Grande do Sul), Porto Alegre (RS), 202
4.

PACHECO, Wagner Felipe *et al*. Voltametrias: Uma breve revisão sobre os conceitos. **Revista Virtual de Química**, v. 5, n. 4, p. 516-537, 2013.

RIBEIRO, Josimar. Espectroscopia de Impedância Eletroquímica: Uma ferramenta nas investigações eletroquímicas. **Revista Virtual de Química**, v. 12, n. 6, p. 1626-1641, 2020.

SANTOS, Vagner Bezerra dos. **Desenvolvimento de um potenciostato/galvanostato portátil e eletrodos impressos para determinações in situ em análises em fluxo com transmissão de dados em tempo real**. 2013. Tese (Doutorado) – Programa de Pós-Graduação em Química, UFSCAR (Universidade Federal de São Carlos), São Carlos (SP), 2013.

SOUSA, Lucas Reis**. Potenciostato de baixo custo com transmissão de dados sem fio**. 2021. Monografia (Graduação) - Engenharia de Controle e Automação, Instituto de Ciência e Tecnologia de Sorocaba, (UNESP), Universidade Estadual Paulista, Sorocaba (SP), 2021.

STMICROELECTRONICS. LM324. **Datasheet**. [s.l]: Stmicroelectronics, 1999. Disponível em: https://www.alldatasheet.com/datasheet pdf/view/22762/STMICROELECTRONICS/LM324P.html. Acesso em: 05 jan. 2025.

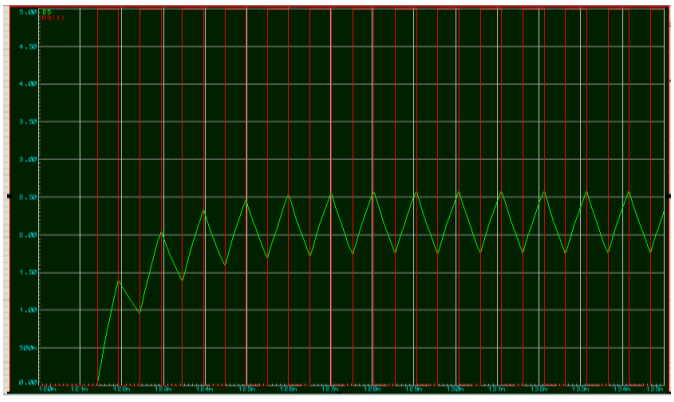
8 APÊNDICES

8.1 APÊNDICE A - METODOLOGIA DAS SIMULAÇÕES

A parte de comunicação com o broker não foi simulada, uma vez que há limitações no software e não foram encontradas bibliotecas para simular a comunicação da placa com o servidor. Há uma biblioteca para proteus que possibilita simular o ESP32, no entanto, ela não possui o pino tx disponível, impossibilitando de enxergar o que haveria escrito no “Serial.println”, a não ser que outros pinos fossem configurados para tal (o que acabou sendo feito). Porém, nessa biblioteca para ESP32 que foi encontrada, não foi encontrado ADC. Portanto, foi simulada uma placa Arduíno mega ao invés de uma placa ESP32 e o código foi ajustado para tal.

Como o PWM do Arduino gera valores digitais (0 V ou 5 V) que representam valores analógicos, após o sinal do PWM foi colocado uma espécie de integrador, que consiste em um resistor em série com um capacitor, onde o sinal PWM incide sobre um dos terminais do resistor; O outro terminal do resistor é conectado a um terminal do capacitor, que tem seu outro terminal conectado no terra. A saída é coletada no ponto onde está conectado o capacitor e o resistor simultaneamente (chamado de 25 na figura 19). A transformação do sinal PWM em um valor analógico é mostrada pela figura A-1.

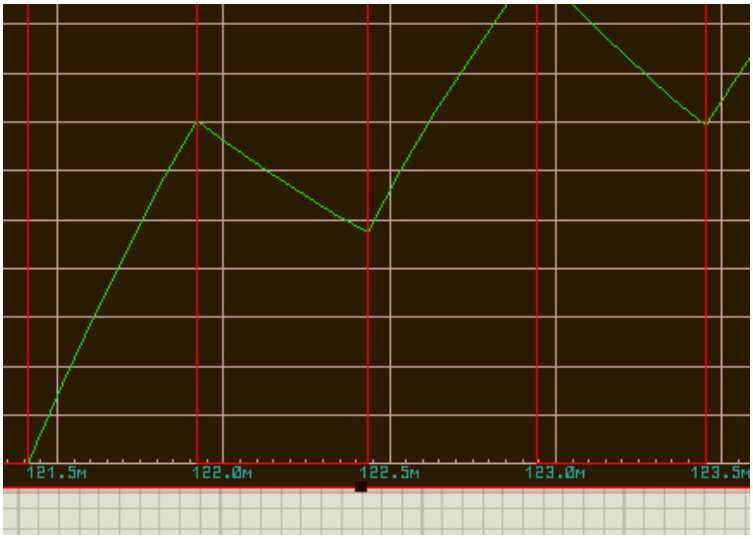
Figura A-1 - Conversão sinal PWM do Arduino em um sinal analógico. No eixo x o tempo em ms, no eixo Y as tensões elétricas do pino 5 da placa (vermelho) e do valor analógico, 25, em verde



Fonte: Elaborado pelo autor (2025).

Analogue Analysis do Proteus demonstrando a conversão de um sinal PWM (em vermelho) em um sinal analógico (verde). Uma melhoria que poderia ser feita seria colocar um buffer de potencial elétrico entre o pino 5 do arduino, que gera o sinal PWM e o resistor do circuito que converte esse sinal em analógico.

Figura A-2 - Conversão sinal PWM do Arduino em um sinal analógico. No eixo x o tempo em ms, no eixo Y as tensões elétricas do pino 5 da placa (vermelho) e do valor analógico, 25, em verde



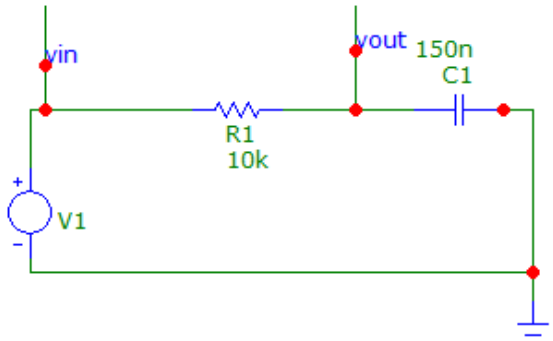
Fonte: Elaborado pelo autor (2025).

Como visto na figura A-2, o PWM desse microcontrolador possui um período de em torno de 1 ms. Portanto sua frequência é de em torno de 1000 Hz. A frequência de corte do circuito com o resistor e o capacitor deve ser menor que ela. O cálculo dessa frequência fc, com base na resistência R e da capacitância C, de C1 e R9 é feito da seguinte forma: R é 10 k Ω, C é 150 nF, a frequência de corte é 1 dividido por 2πRC, o que resulta em aproximadamente 106 Hz.

As frequências analisadas pelo potenciostato devem ser menores que a frequência de corte desse circuito RC. A frequência do PWM f_{pwm} deve ser maior que a frequência de corte fc, que, por sua vez, deve ser maior que a frequência de operação do potenciostato $f_{potenciostato}$.

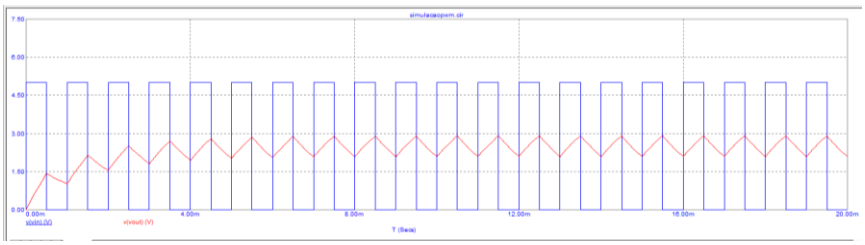
Para validar isso, foi feita uma simulação no Microcap12, com uma fonte de tensão elétrica configurada para gerar um sinal PWM entre um pino chamado de vin e o terra. E com um ponto chamado vout que está em um dos terminais de um capacitor, conforme a figura A-3. A figura A-4 mostra o caso onde a frequência do PWM maior que a frequência de corte do circuito RC e a figura A-5 mostra o caso onde a frequência do PWM é menor que a frequência de corte do circuito RC.

Figura A-3 - Circuito que converte um sinal PWM em um valor analógico quase fixo, no Microcap12



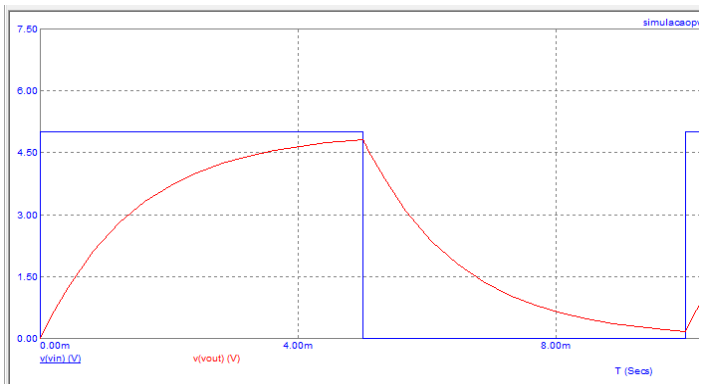
Fonte: Elaborado pelo autor (2025).

Figura A-4 - Simulação do circuito que converte um sinal PWM em um valor analógico quase fixo com frequência do PWM maior que a frequência de corte RC, no Microcap12



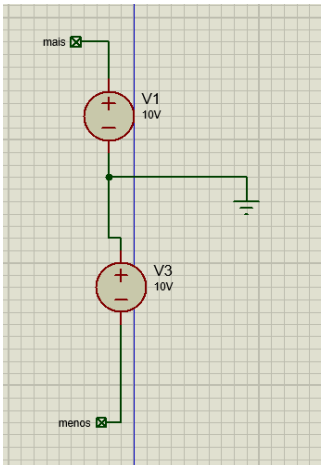
Fonte: Elaborado pelo autor (2025).

Figura A-5 - Simulação do circuito que converte um sinal PWM em um valor analógico quase fixo com frequência do PWM parecida com a frequência de corte RC, no Microcap12



Fonte: Elaborado pelo autor (2025).

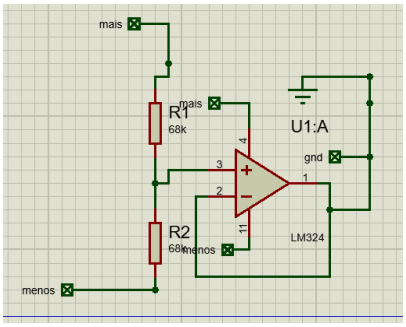
Figura A-6 - Fontes de alimentação



Fonte: Elaborado pelo autor (2025).

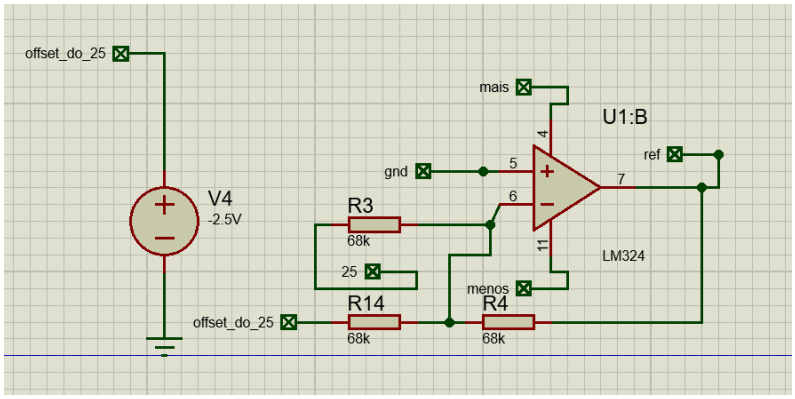
A alimentação dos amplificadores operacionais é oriunda de fontes de tensão elétrica, conforme a figura A-6. O amplificador operacional usado foi o LM324, o primeiro chip foi usado para “gerar” o terra, conforme a figura A-7 e para inverter o sinal 25 (sinal analógico oriundo do sinal PWM gerado pelo Arduino) somando-o com um valor de tensão, conforme a figura A-8.

Figura A-7 - Amplificador operacional que coleta a média entre as tensões de alimentação para “gerar” o terra



Fonte: Elaborado pelo autor (2025).

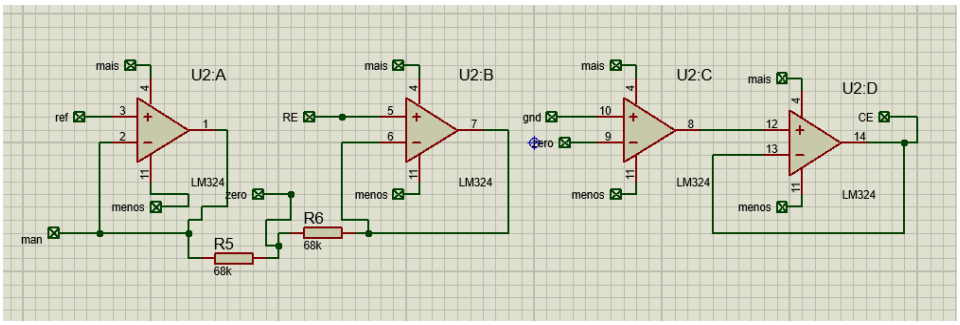
Figura A-8 - Circuito que coleta o sinal analógico e inverte seu sinal e soma com um offset



Fonte: Elaborado pelo autor (2025).

Uma das diferenças entre a simulação e o circuito montado é que a senoide entre o RE e o WE está deslocada de forma que pode assumir valores tanto positivos quanto negativos. Por isso, na figura A-8 há um somador inversor, com uma espécie de offset de 2,5 V, uma vez que a tensão entre os pinos do arduino e o terra vão de 0 a 5 V e o valor médio entre seus limites é 2,5 V. Assim espera-se que em ref haja uma senoide centrada em 0 V. A saída desse somador é o pino ref.

Figura A-9 - Trecho do circuito que faz com que a tensão elétrica entre o pino de referência “ref” e o terra seja igual a tensão entre RE e WE, com o sinal invertido. Ele faz isso através do potencial elétrico de controle em CE

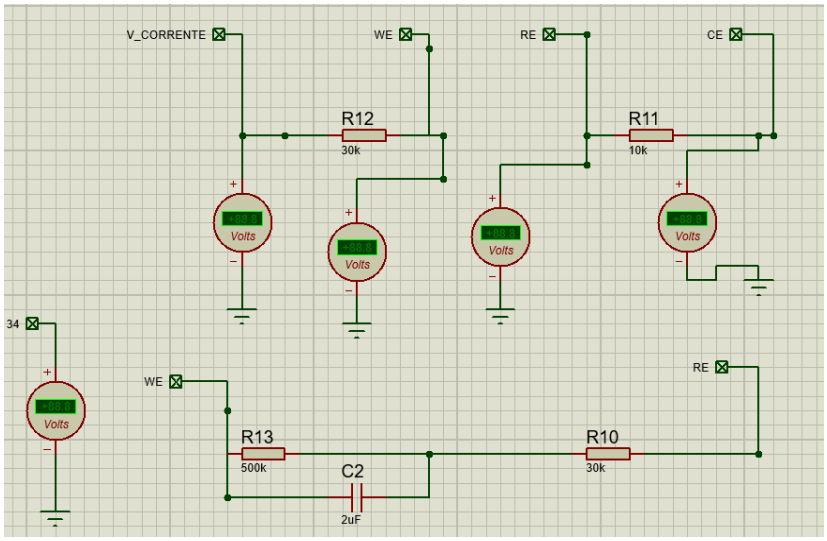


Fonte: Elaborado pelo autor (2025).

O pino ref é conectado a um buffer, conforme a figura A-9, assim como o pino do eletrodo RE. Ambos buffers têm sua saída conectada a um resistor de 68 kΩ, são 2 resistores assim. O ponto “zero” é conectado em ambos resistores e no pino - de um comparador, para comparar o potencial do pino “zero” com o terra. A tensão elétrica entre o pino “zero” e o terra deve ser igual a 0 V (ou muito próximo disso), pois o comparador tem sua saída ligada a um buffer que é ligado ao eletrodo CE que é conectado a um resistor conectado ao eletrodo RE. Logo há uma realimentação, que faz com que o comparador na verdade seja uma garantia que ambos os potenciais elétricos (“gnd” e “zero”) sejam iguais, para que o amplificador operacional (que possui um ganho diferencial relativamente grande) não sature.

Se o amplificador operacional estivesse saturado, o potencial elétrico em CE seria igual ao de VCC ou -VCC, logo o potencial elétrico no eletrodo RE seria algo parecido. No caso de RE ser próximo de VCC, o potencial elétrico no “zero” seria a média entre algo próximo de VCC e “ref”, ou seja, seria positivo em relação ao terra (a tensão entre “ref” e o terra é negativa mas tem seu módulo menor que VCC, logo a média entre o potencial em RE e em “ref” é maior que 0 V), fazendo o amplificador saturar para o lado contrário. E vice-versa, logo o amplificador operacional não tende a saturar.

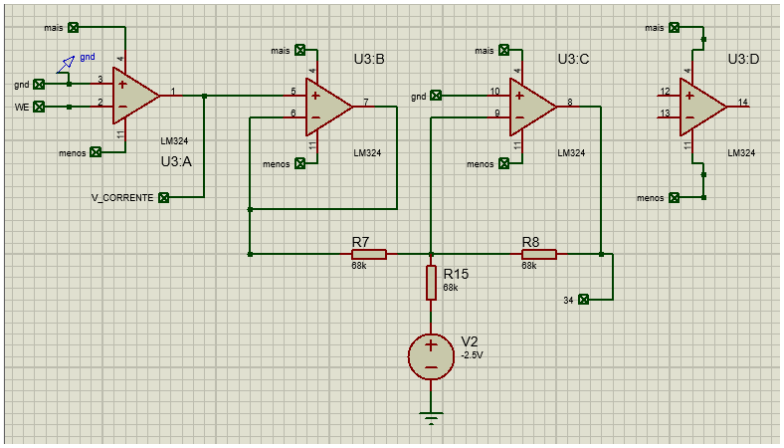
Figura A-10 - Trecho do circuito com um “líquido” (impedância) entre RE e WE, ao qual se aplica uma tensão elétrica e se mede a corrente elétrica entre RE e WE através do resistor entre WE e $V_{corrente}$



Fonte: Elaborado pelo autor (2025).

A figura A-10 mostra os eletrodos WE, RE, CE, o pino $V_{corrente}$ e os componentes entre RE e WE. O potencial elétrico em CE controla o potencial elétrico em RE de forma que a tensão elétrica entre RE e WE ela seja igual ao inverso da tensão elétrica entre “ref” e o terra, que é uma senoide. O potencial elétrico do WE é igual ao terra, porém não há uma conexão entre WE e o terra. Há uma combinação de elementos resistivos e capacitivos entre o RE e o WE que simula a combinação de elementos resistivos e capacitivos no circuito equivalente do líquido entre os eletrodos RE e WE. Entre WE e $V_{corrente}$ há um resistor que “transforma” a corrente que passa entre RE e WE em uma tensão elétrica, entre os seus terminais. Como mostrado na figura A-11, o potencial elétrico em um dos terminais desse resistor (o $V_{corrente}$) incide sobre um buffer de tensão, para depois, a tensão entre a saída do buffer e o terra ser injetada em um somador inversor para que haja uma leitura adequada pelo Arduino.

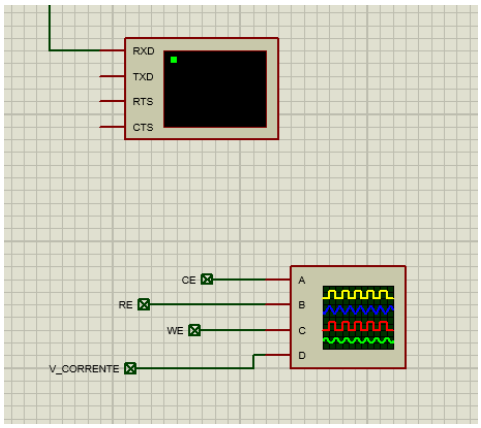
Figura A-11 - Trecho do circuito onde o potencial elétrico em $V_{corrente}$ incide em um buffer de tensão, cuja saída em tensão elétrica é injetada em um circuito somador inversor, cuja saída (34) é lida pelo Arduino



Fonte: Elaborado pelo autor (2025).

Da mesma forma que o potencial em “zero” é igual ao potencial no terra por causa de um comparador e uma realimentação negativa, o potencial em WE é igual ao do terra pelo mesmo motivo. $V_{corrente}$ tem um potencial elétrico que é um incide em um buffer de tensão elétrica cuja saída é injetada em um circuito inversor somador, somando essa saída com uma tensão elétrica de -2,5 V e depois invertendo o sinal do resultado dessa operação. A saída desse circuito amplificador inversor é o ponto 34, cuja tensão elétrica entre ele e o GND é lida pelo microcontrolador. O microcontrolador por sua vez faz os cálculos necessários e printa o resultado no virtual terminal, conforme a figura A-12.

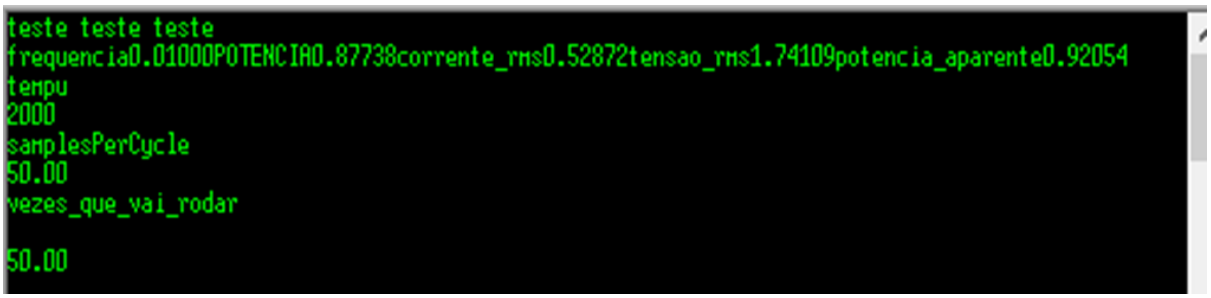
Figura A-12 - Osciloscópio e Virtual Terminal, usados na simulação dinâmica do Proteus 8



Fonte: Elaborado pelo autor (2025).

Virtual terminal (conectado ao pino 4 do Arduino, que foi configurado para fazer o papel do pino tx onde as informações são mostradas em forma de texto) e osciloscópio, usados na simulação dinâmica do Proteus. Um exemplo do que aparece no Virtual Terminal aparece na figura A-13.

Figura A-13 - Exemplo do que aparece no Virtual Terminal



Fonte: Elaborado pelo autor (2025).

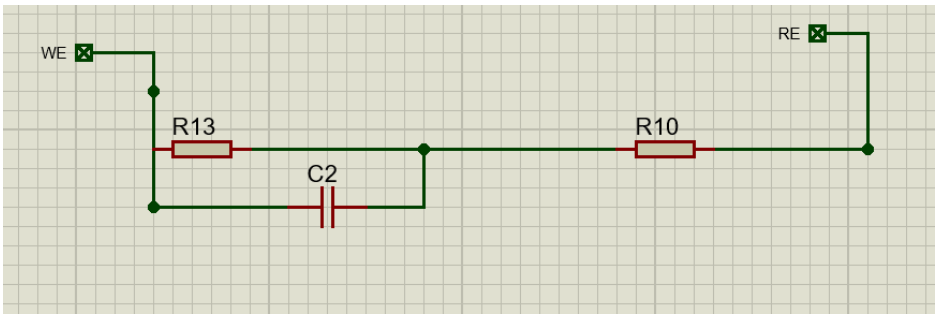
Exemplo de texto no Virtual terminal, mostrando a frequência (Hz), algo proporcional a potência ativa, algo proporcional a corrente RMS, tensão RMS e algo proporcional a potência aparente. Além disso ele mostra “tempu” que é o tempo entre uma amostra e outra da senoide aplicada, samplesPerCycle que é a quantidade de pontos da senóide calculados/aplicados para cada período dela, vezes_que_vai_rodar é igual ao samplesPerCycle a não ser quando a senoide for

aplicada por mais de um período, quando esse valor expressa a quantidade total de pontos de uma senóide aplicada.

No caso da leitura da corrente e potências, como o valor lido pelo pino A0 é uma tensão elétrica (entre A0 e o GND) proporcional ao valor do resistor usado entre WE e $V_{corrente}$ e proporcional também a corrente entre RE e WE, o valor lido não corresponde a corrente elétrica no líquido e sim a corrente elétrica no líquido vezes o valor do resistor entre WE e $V_{corrente}$. E esse resistor poderia ser modificado, logo por convenção do autor, foi decidido que esse valor seria calculado fora do microcontrolador, com base em suas mensagens, no código em Python, mas como não há simulação deste código em Python, deve-se calcular de outra forma no caso da simulação no Proteus 8.

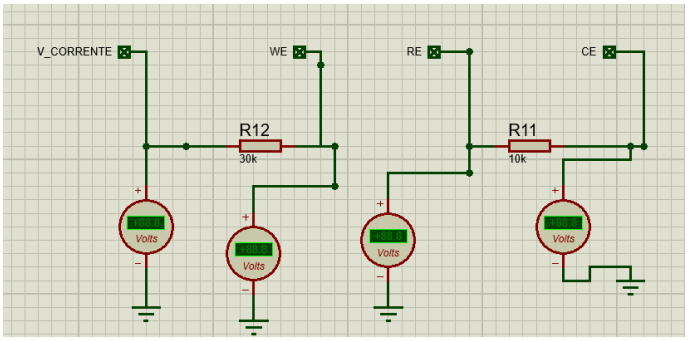
Por fins de validação, os resultados no Proteus 8 foram comparados com cálculos teóricos. Os circuitos usados entre os pinos RE e WE se assemelham com circuitos padrões do modelo de Randles, como na figura A-14, que consistem em uma resistência elétrica (R10) em série com um outro circuito de um capacitor (C2) em paralelo com uma outra resistência elétrica (R13), que é o modelo de metais com revestimentos orgânicos intactos. A figura A-15 mostra os eletrodos CE, RE, WE e $V_{corrente}$, da simulação.

Figura A-14 - Componentes que foram usados nas simulações entre RE e WE, em cada simulação houve pelo menos alguma alteração nos seus valores de resistência elétrica e/ou capacitância



Fonte: Elaborado pelo autor (2025).

Figura A-15 - Eletrodos CE, RE, WE e $V_{corrente}$, na simulação



Fonte: Elaborado pelo autor (2025).

Para calcular os valores teóricos se usou o valor de resistência elétrica dos resistores R10, R12 e R13 e do capacitor C2 além da tensão RMS de saída do circuito que transforma o sinal PWM oriundo do Arduino em um valor analógico, de em torno de 1,55 V (nos cálculos se usou o valor de 1,55 V) e não o valor printado. Esse valor de 1,55 V foi escolhido por ser aproximadamente o valor da tensão medida no caso de haver apenas um resistor entre RE e WE, de 30 kΩ, com R12 de resistência elétrica também de 30 kΩ.

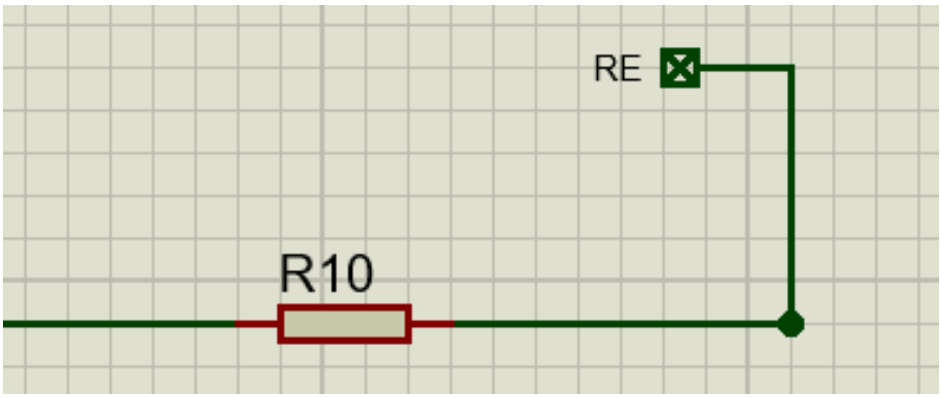
Quando a impedância entre RE e WE é formada apenas por um resistor de 30 kΩ e o resistor entre WE e $V_{corrente}$ é de 30 kΩ, o valor RMS lido pela placa Arduino na simulação é de em torno de 1,55 V.

Primeiro deve-se calcular o módulo e o ângulo da impedância elétrica entre o eletrodo RE e o WE. Para isso, o circuito entre RE e WE foi subdividido em 2 partes, uma com o resistor R10 (parte 1) e outra com o paralelo entre o resistor R13 e o C2 (parte 2).

Parte 1:

A figura A-16 mostra o circuito considerado na parte 1 do cálculo. Por fins de cálculo, o circuito foi dividido em 2 partes.

Figura A-16 - Parte 1 do cálculo da impedância equivalente entre RE e WE



Fonte: Elaborado pelo autor (2025).

Parte 1.

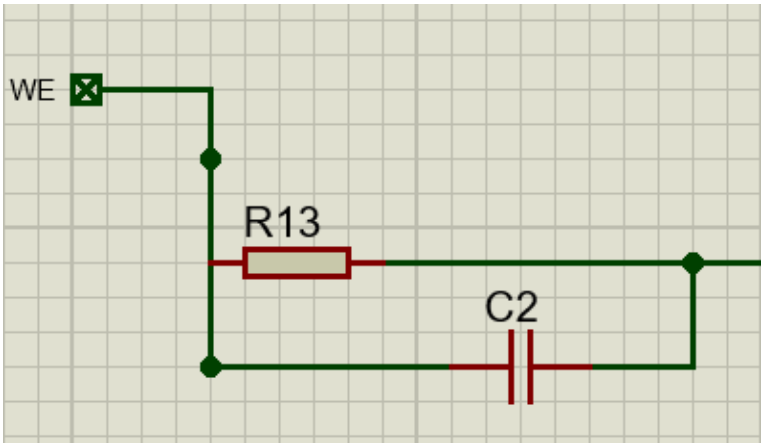
Parte real: R10 Ω.

Parte imaginária: 0 Ω.

Parte 2:

A figura A-17 mostra o circuito considerado na parte 2 do cálculo.

Figura A-17 - Parte 2 do cálculo da impedância equivalente entre RE e WE



Fonte: Elaborado pelo autor (2025).

Parte 2: As equações A-1, A-2 e A-3 mostram respectivamente os valores de XC, R, $Z_{EQUIVALENTE}$, a equação A-4 mostra também uma outra forma de representar $Z_{EQUIVALENTE}$, A-5 mostra seu módulo e A-6 mostra seu ângulo, onde $\omega\ (\frac{Rad}{s})$ é $\pi \cdot 2 \cdot f$, onde f é a frequência usada (Hz). O restante das variáveis das equações são os valores dos componentes elétricos do circuito.

$$XC\ (\Omega) = \frac{-j}{\omega \cdot C_2}$$

Equação A-1

$$R\ (\Omega) = R_{13}\ \Omega$$

Equação A-2

$$Z_{EQUIVALENTE}\,(\Omega) = \frac{1}{\frac{1}{XC_2} + \frac{1}{R_{13}}} = \frac{1}{\omega \cdot C_2 \cdot j + \frac{1}{R_{13}}}$$

Equação A-3

$$Z_{EQUIVALENTE}\,(\Omega) = \frac{R_{13}}{\omega \cdot C_2 \cdot j \cdot R_{13} + 1}$$

Equação A-4

$$|Z_{EQUIVALENTE}|\ (\Omega) = \frac{R_{13}}{\sqrt{(\omega \cdot C_2 \cdot R_{13})^2 + 1^2}}$$

Equação A-5

$$\angle Z_{EQUIVALENTE}(\frac{Rad}{s}) = -atan(\frac{\omega \cdot C_2 \cdot R_{13}}{1})$$

Equação A-6

Parte real: $Z_{EQUIVALENTE} \cdot cos(-atan(\frac{\omega \cdot C_2 \cdot R_{13}}{1}))\ \Omega$.

Parte imaginária: $Z_{EQUIVALENTE} \cdot sen(-atan(\frac{\omega \cdot C_2 \cdot R_{13}}{1}))\ \Omega$.

Parte 1 e 2 em série:

Parte real: A equação A-7 mostra a componente real da soma entre a parte 1 e 2, enquanto a parte imaginária é dada pela equação A-8

A componente real da soma entre a parte 1 e 2 é dada pela equação 28.

$$|Z_{EQUIVALENTE}| \cdot cos(-atan(\frac{\omega \cdot C_2 \cdot R_{18}}{1})) + R_{10}\ \Omega\text{=A}$$

Equação A-7

Parte imaginária:

$$|Z_{EQUIVALENTE}| \cdot sen(-atan(\frac{\omega \cdot C_2 \cdot R_{18}}{1}))\ \Omega\text{=B.}$$

Equação A-8

Corrente RMS teórica equação A-9:

$$IRMS_{TEÓRICA}\ (A) = \frac{1,55}{|Z_{EQUIVALENTE}|}$$

Equação A-9

1,55 V é o valor RMS aproximado da tensão elétrica entre os eletrodos RE e WE (é um pouco menor que o valor RMS da tensão elétrica entre o pino 5 e o GND).

Potência aparente teórica ($S_{TEÓRICA}$), equação A-10:

$$S_{TEÓRICA}(VA) = \frac{1{,}55^2}{|Z_{EQUIVALENTE}|}$$

Equação A-10

Ou pela equação A-11

$$S_{TEÓRICA}(VA) = IRMS_{TEÓRICA} \cdot 1{,}55$$

Equação A-11

Potência ativa teórica, equação A-12:

$$P\ (W) = \frac{1,55^2}{|Z_{EQUIVALENTE}|} \cdot cos(atan(\frac{B}{A}))$$

Equação A-12

ou pela equação A-13:

$$P\ (W) = IRMS_{TEÓRICA} \cdot 1{,}55 \cdot cos(atan(\frac{B}{A}))$$

Equação A-13

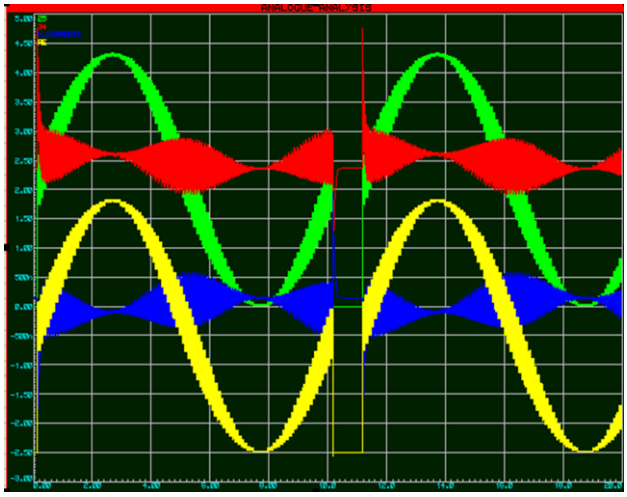
8.2 APÊNDICE B - SIMULAÇÕES REALIZADAS

8.2.1 Teste 1

As figuras B-1, B-2, B-3, B-4, B-5 e B-6 são referentes ao teste 1. A figura B-1 mostra a tela do Analogue Analysis, a figura B-2 mostra a tela do Virtual Terminal, a figura B-3 mostra os componentes elétricos entre RE e WE, as figuras B-4, B-5 e B-6 são comparações entre valores simulados e calculados. A figura B-4 é referente a corrente elétrica, a figura B-5 é referente a potência aparente, a figura B-6 é referente a potência ativa.

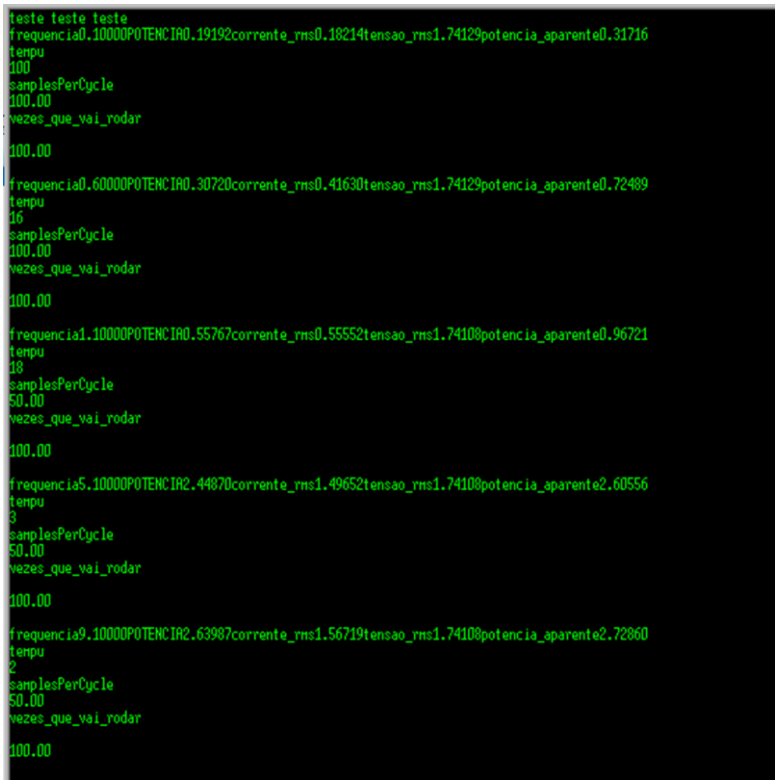
A figura B-4 mostra os valores de corrente elétrica (µA) simulados de cada frequência, que são os valores da figura B-2 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ), conforme a equação 18. A figura B-5 mostra os valores de potência aparente (µVA) simulados de cada frequência, que são os valores da figura B-2 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ) vezes 1,55 sobre 1,77, como na equação 19.A figura B-6 mostra os valores de potência ativa (µW) simulados de cada frequência, que são os valores da figura B-2 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ) vezes 1,55 sobre 1,77, como na equação 20. Os valores calculados seguem o que foi descrito no apêndice A

Figura B-1 - Tela do Analogue Analysis no teste 1 de simulação no Proteus 8, um recurso do Proteus 8 que serviu para plotar a forma de onda de tensões elétricas importantes para o entendimento do circuito. Na imagem: 25 (verde), uma tensão analógica de referência (entre o ponto 25 e o terra) parecida com a aplicada entre RE e WE, a tensão 34 que é lida pelo Arduino (vermelho) (entre o ponto 25 e o terra), $V_{corrente}$, que é uma tensão elétrica (entre o ponto $V_{corrente}$ e o terra) proporcional a corrente elétrica no "líquido" (azul) e a tensão entre RE e WE (amarelo)



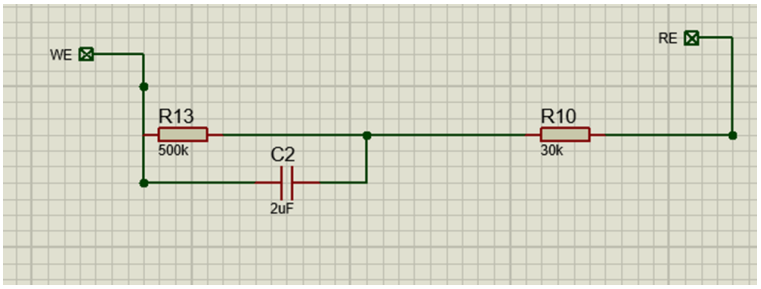
Fonte: Elaborado pelo autor (2025).

Figura B-2 - Tela do Virtual Terminal no teste 1 de simulação no Proteus 8



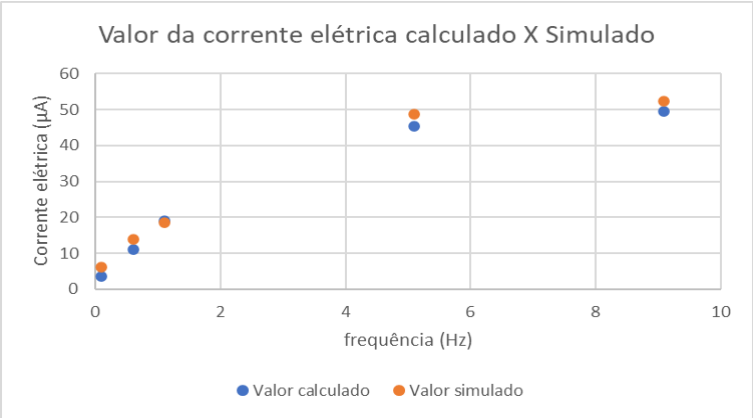
Fonte: Elaborado pelo autor (2025).

Figura B-3 - Combinação de componentes elétricos entre RE e WE no teste 1 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

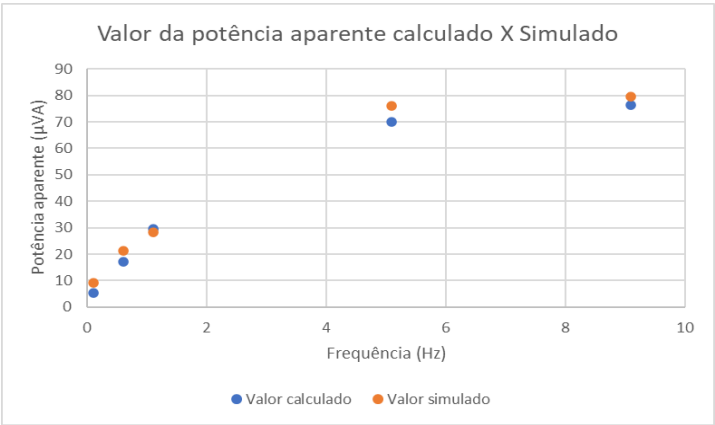
Figura B-4 - Corrente elétrica calculada (azul) X simulada (laranja) no teste 1 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

Na figura B-4 o desvio máximo observado entre os valores calculados e medidos foi de 3,57 µA. Alguns motivos para que desvio ocorra são o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico.

Figura B-5 - Potência aparente calculada (azul) X simulada (laranja) no teste 1 de simulação no Proteus 8

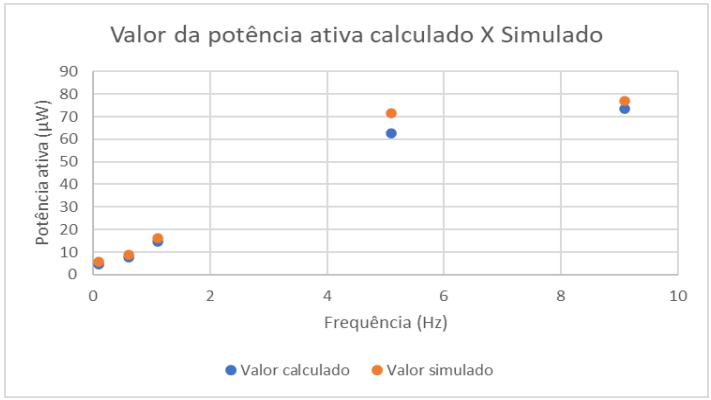


Fonte: Elaborado pelo autor (2025).

Na figura B-5 o desvio máximo observado entre os valores calculados e medidos foi de 5,89 µVA. Alguns motivos para que desvio ocorra são o erro

numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico.

Figura B-6 - Potência ativa calculada (azul) X simulada (laranja) no teste 1 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

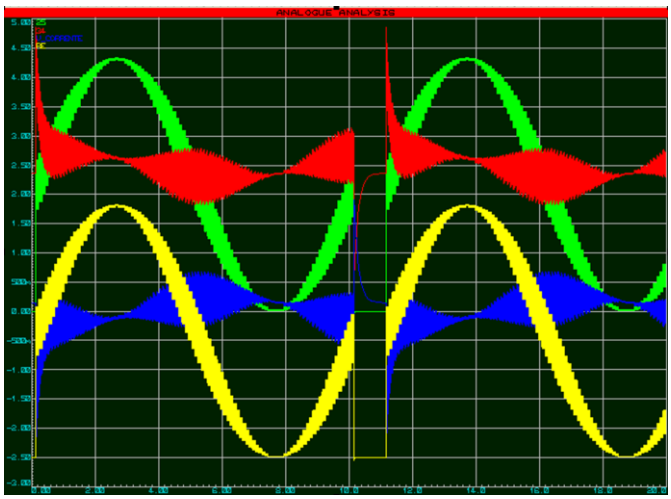
Na figura B-6 o desvio máximo observado entre os valores calculados e medidos foi de 9 µW. Alguns motivos para que desvio ocorra são o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico. Porém nota-se que nesse caso o maior desvio é muito diferente do menor (eles são em ordem de valor de frequência: 0,97 µW, 1,14 µW, 1,71 µW, 9,00 µW e 3,56 µW) e para outras frequências ele foi em geral relativamente bem menor.

8.2.2 Teste 2

As figuras B-7, B-8, B-9, B-10, B-11 e B-12 são referentes ao teste 2. A figura B-7 mostra a tela do Analogue Analysis, a figura B-8 mostra a tela do Virtual Terminal, a figura B-9 mostra os componentes elétricos entre RE e WE, as figuras B-10, B-11 e B-12 são comparações entre valores simulados e calculados. A figura B-10 é referente a corrente elétrica, a figura B-11 é referente a potência aparente, a figura B-12 é referente a potência ativa.

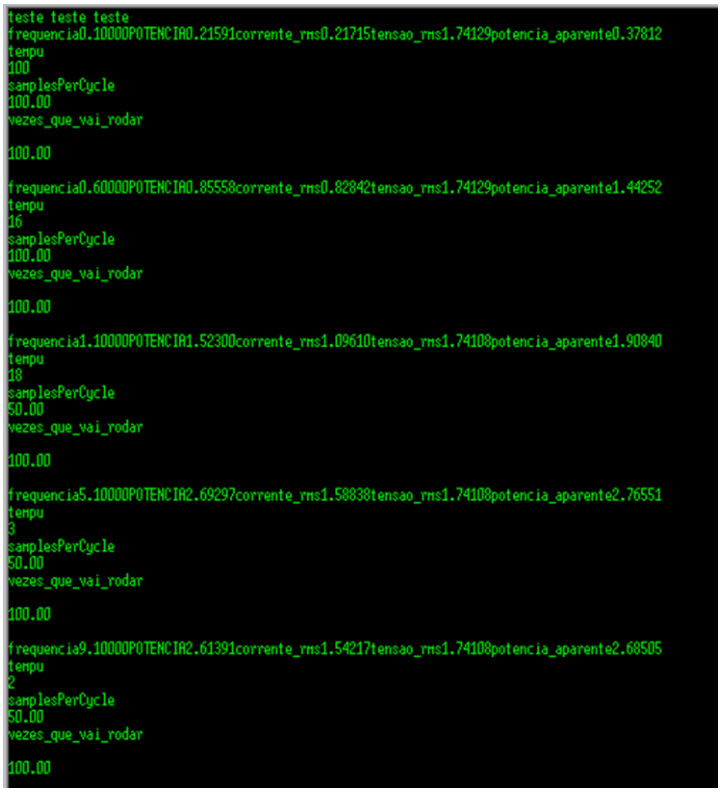
A figura B-10 mostra os valores de corrente elétrica (µA) simulados de cada frequência, que são os valores da figura B-8 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ), conforme a equação 18. A figura B-11 mostra os valores de potência aparente (µVA) simulados de cada frequência, que são os valores da figura B-8 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ) vezes 1,55 sobre 1,77, como na equação 19.A figura B-12 mostra os valores de potência ativa (µW) simulados de cada frequência, que são os valores da figura B-8 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ) vezes 1,55 sobre 1,77, como na equação 20. Os valores calculados seguem o que foi descrito no apêndice A.

Figura B-7 - Tela do Analogue Analysis no teste 2 de simulação no Proteus 8, um recurso do Proteus 8 que serviu para plotar a forma de onda de tensões elétricas importantes para o entendimento do circuito. Na imagem: 25 (verde), uma tensão analógica de referência (entre o ponto 25 e o terra) parecida com a aplicada entre RE e WE, a tensão 34 que é lida pelo Arduino (vermelho) (entre o ponto 25 e o terra), $V_{corrente}$, que é uma tensão elétrica (entre o ponto $V_{corrente}$e o terra) proporcional a corrente elétrica no "líquido" (azul) e a tensão entre RE e WE (amarelo)



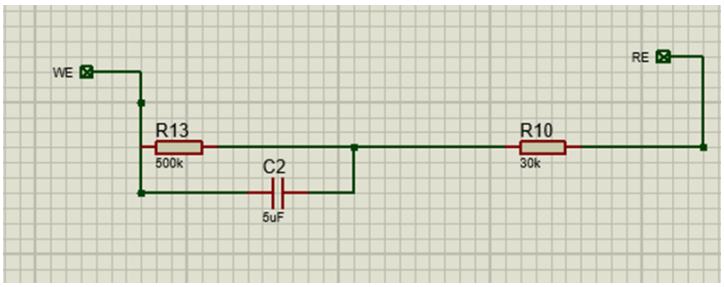
Fonte: Elaborado pelo autor (2025).

Figura B-8 - Tela do Virtual Terminal no teste 2 de simulação no Proteus 8



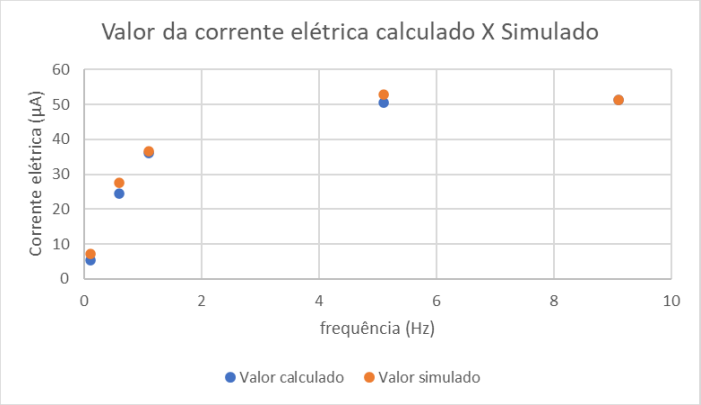
Fonte: Elaborado pelo autor (2025).

Figura B-9 - Combinação de componentes elétricos entre RE e WE no teste 2 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

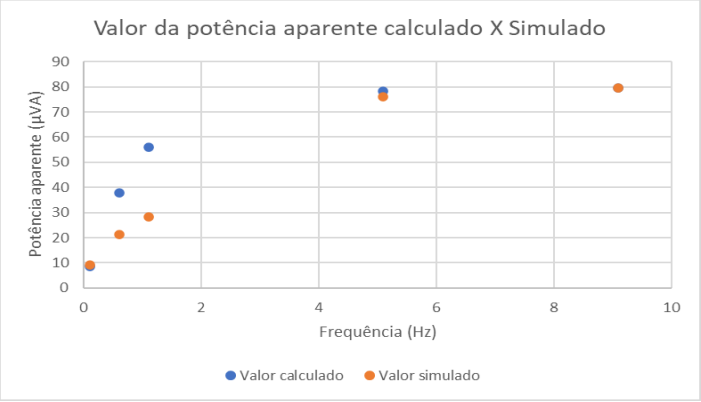
Figura B-10 - Corrente elétrica calculada (azul) X simulada (laranja) no teste 2 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

Na figura B-10 o desvio máximo observado entre os valores calculados e medidos foi de 3,16 µA. Alguns motivos para que desvio ocorra são o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico.

Figura B-11 - Potência aparente calculada (azul) X simulada (laranja) no teste 2 de simulação no Proteus 8

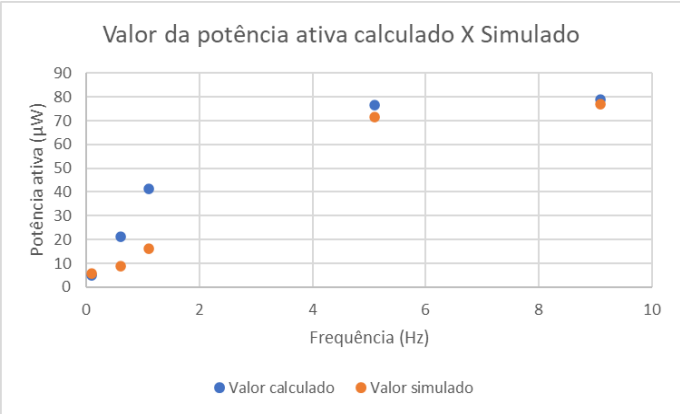


Fonte: Elaborado pelo autor (2025).

Na figura B-11 o desvio máximo observado entre os valores calculados e medidos foi de 27,86 µVA. Alguns motivos para que desvio ocorra são o erro

numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico. Porém nota-se que nesse caso o maior desvio é muito diferente do menor (eles são em ordem de valor de frequência: 0,85 µVA, 16,75 µVA, 27,86 µVA, 2,15 µVA e 0,17 µVA) e para outras frequências ele foi em geral relativamente bem menor.

Figura B-12 - Potência ativa calculada (azul) X simulada (laranja) no teste 2 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

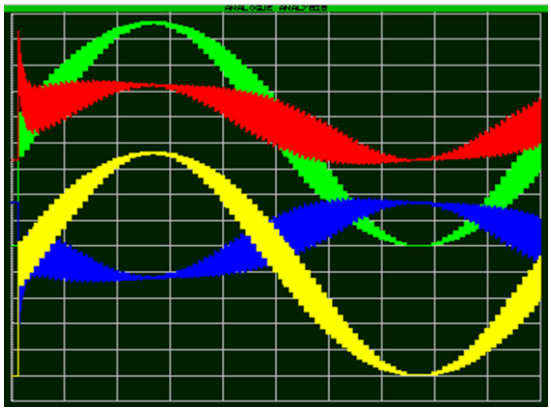
Na figura B-12 o desvio máximo observado entre os valores calculados e medidos foi de 25,19 µW. Alguns motivos para que desvio ocorra são o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico. Mas nesse caso houve maior desvio relativo, isso pode ser justificado por algum problema na contagem de tempo e execução dos comandos do microcontrolador ESP32 em frequências menores, da mesma forma que na figura B-11. Porém nota-se que nesse caso o maior desvio é muito diferente do menor (eles são em ordem de valor de frequência: 0,48 µW, 12,3 µW, 25,19 µW, 5,10 µW e 1,89 µW) e para outras frequências ele foi em geral relativamente bem menor.

8.2.3 Teste 3

As figuras B-13, B-14, B-15, B-16, B-17 e B-18 são referentes ao teste 3. A figura B-13 mostra a tela do Analogue Analysis, a figura B-14 mostra a tela do Virtual Terminal, a figura B-15 mostra os componentes elétricos entre RE e WE, as figuras B-16, B-17 e B-18 são comparações entre valores simulados e calculados. A figura B-16 é referente a corrente elétrica, a figura B-17 é referente a potência aparente, a figura B-18 é referente a potência ativa.

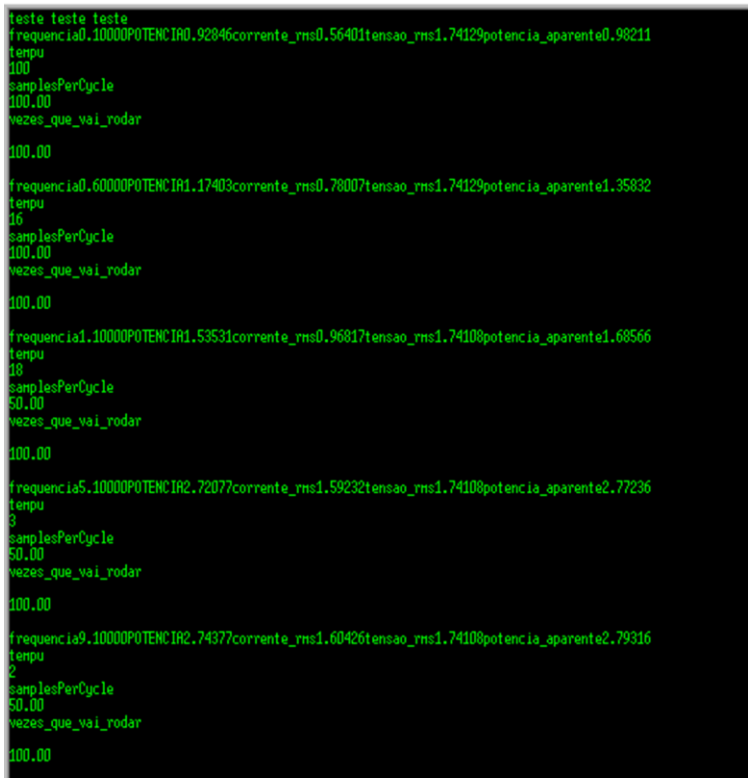
A figura B-16 mostra os valores de corrente elétrica (µA) simulados de cada frequência, que são os valores da figura B-14 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ), conforme a equação 18. A figura B-17 mostra os valores de potência aparente (µVA) simulados de cada frequência, que são os valores da figura B-14 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ) vezes 1,55 sobre 1,77, como na equação 19.A figura B-18 mostra os valores de potência ativa (µW) simulados de cada frequência, que são os valores da figura B-14 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ) vezes 1,55 sobre 1,77, como na equação 20. Os valores calculados seguem o que foi descrito no apêndice A.

Figura B-13 - Tela do Analogue Analysis no teste 3 de simulação no Proteus 8, um recurso do Proteus 8 que serviu para plotar a forma de onda de tensões elétricas importantes para o entendimento do circuito. Na imagem: 25 (verde), uma tensão analógica de referência (entre o ponto 25 e o terra) parecida com a aplicada entre RE e WE, a tensão 34 que é lida pelo Arduino (vermelho) (entre o ponto 25 e o terra), $V_{corrente}$, que é uma tensão elétrica (entre o ponto $V_{corrente}$ e o terra) proporcional a corrente elétrica no "líquido" (azul) e a tensão entre RE e WE (amarelo)



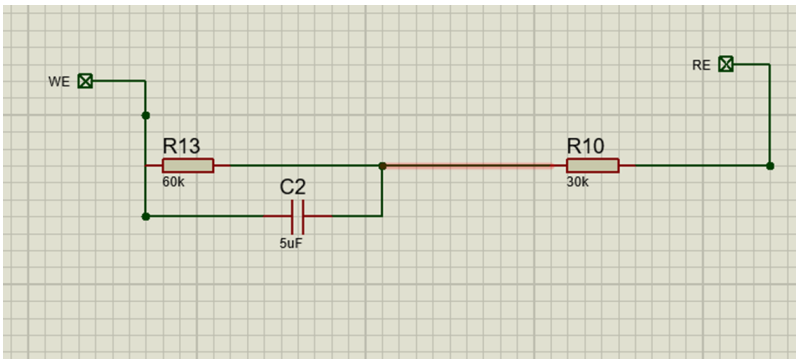
Fonte: Elaborado pelo autor (2025).

Figura B-14 - Tela do Virtual Terminal no teste 3 de simulação no Proteus 8



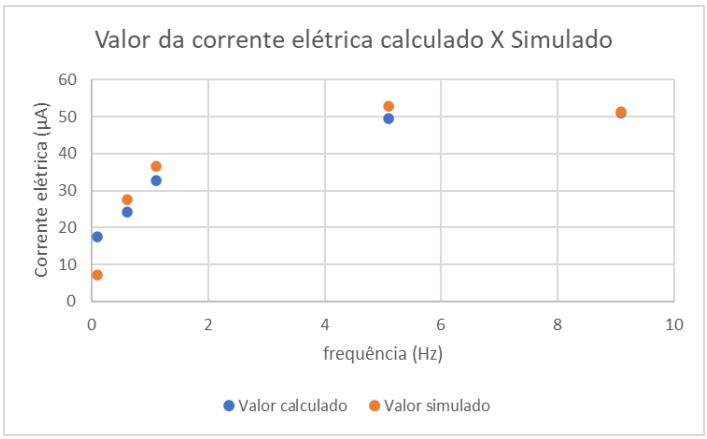
Fonte: Elaborado pelo autor (2025).

Figura B-15 - Combinação de componentes elétricos entre RE e WE no teste 3 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

Figura B-16 - Corrente elétrica calculada (azul) X simulada (laranja) no teste 3 de simulação no Proteus 8

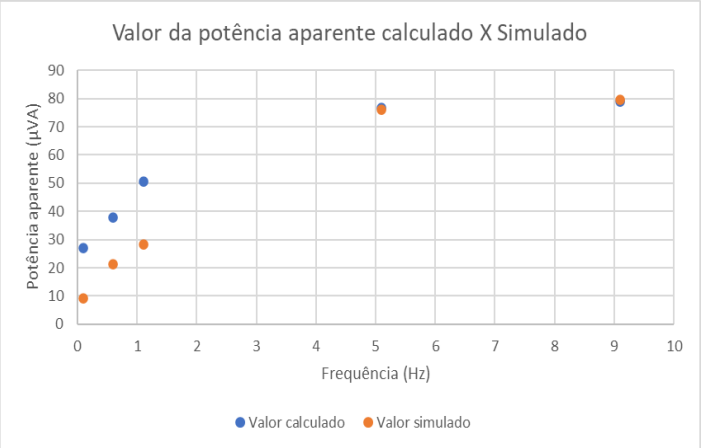


Fonte: Elaborado pelo autor (2025).

Na figura B-16 o desvio máximo observado entre os valores calculados e medidos foi de 10,25 µA. Alguns motivos para que desvio ocorra são o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico. Porém nota-se que nesse caso o maior desvio é muito diferente do menor (eles são em ordem de valor de frequência: 10,25

µA, 3,29 µA, 3,92 µA, 3,36 µA e 0,43 µA) e para outras frequências ele foi em geral relativamente bem menor.

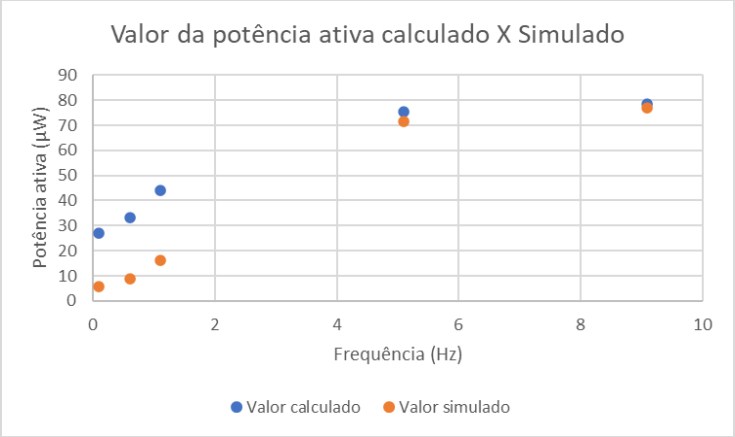
Figura B-17 - Potência aparente calculada (azul) X simulada (laranja) no teste 3 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

Na figura B-17 o desvio máximo observado entre os valores calculados e medidos foi de 22,32 VA. Alguns motivos para que desvio ocorra são o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico. Porém nota-se que nesse caso o maior desvio é muito diferente do menor (eles são em ordem de valor de frequência: 17,8 µVA, 16,54 µVA, 22,31 µVA, 0,80 µVA e 0,63 µVA). O comportamento teórico da potência aparente é de aumentar com a frequência, algo semelhante também ocorre nos resultados da simulação, além disso para frequências maiores os valores simulados são mais semelhantes aos calculados.

Figura B-18 - Potência ativa calculada (azul) X simulada (laranja) no teste 3 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

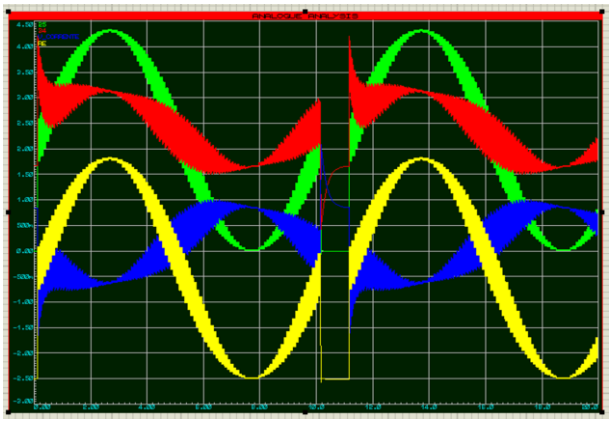
Na figura B-18 o desvio máximo observado entre os valores calculados e medidos foi de 27,67 µW. Alguns motivos para que desvio ocorra são o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico. Porém nota-se que nesse caso o maior desvio é muito diferente do menor (eles são em ordem de valor de frequência: 21,30 µW, 24,37 µW, 27,67 µW, 3,87 µW e 1,44 µW) e para outras frequências ele foi em geral relativamente bem menor. O comportamento teórico da potência ativa é de aumentar com a frequência, algo semelhante também ocorre nos resultados da simulação, além disso para frequências maiores os valores simulados são mais semelhantes aos calculados.

8.2.4 Teste 4

As figuras B-19, B-20, B-21, B-22, B-23 e B-24 são referentes ao teste 4. A figura B-19 mostra a tela do Analogue Analysis, a figura B-20 mostra a tela do Virtual Terminal, a figura B-21 mostra os componentes elétricos entre RE e WE, as figuras B-22, B-23 e B-24 são comparações entre valores simulados e calculados. A figura B-22 é referente a corrente elétrica, a figura B-23 é referente a potência aparente, a figura B-24 é referente a potência ativa.

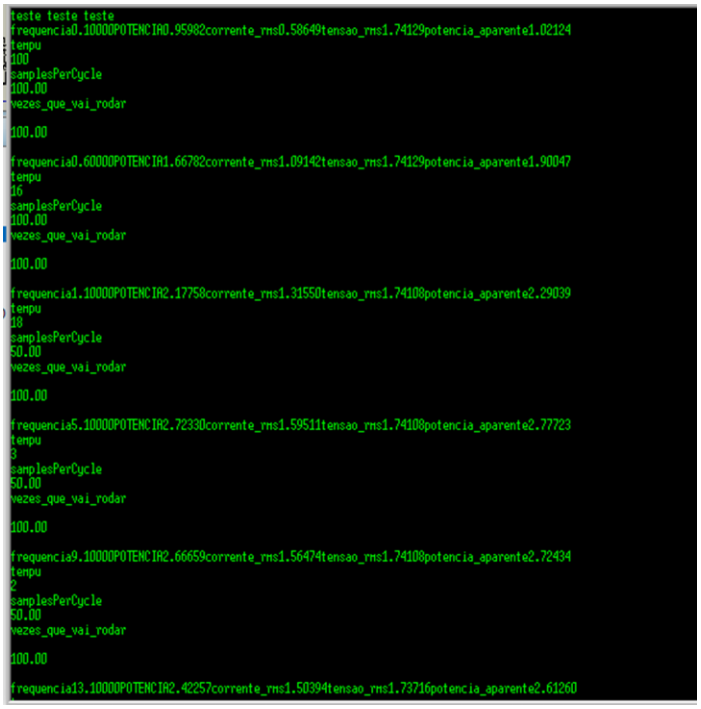
A figura B-22 mostra os valores de corrente elétrica (µA) simulados de cada frequência, que são os valores da figura B-20 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ), conforme a equação 18. A figura B-23 mostra os valores de potência aparente (µVA) simulados de cada frequência, que são os valores da figura B-20 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ) vezes 1,55 sobre 1,77, como na equação 19.A figura B-24 mostra os valores de potência ativa (µW) simulados de cada frequência, que são os valores da figura B-20 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ) vezes 1,55 sobre 1,77, como na equação 20. Os valores calculados seguem o que foi descrito no apêndice A.

Figura B-19 - Tela do Analogue Analysis no teste 4 de simulação no Proteus 8, um recurso do Proteus 8 que serviu para plotar a forma de onda de tensões elétricas importantes para o entendimento do circuito. Na imagem: 25 (verde), uma tensão analógica de referência (entre o ponto 25 e o terra) parecida com a aplicada entre RE e WE, a tensão 34 que é lida pelo Arduino (vermelho) (entre o ponto 25 e o terra), $V_{corrente}$, que é uma tensão elétrica (entre o ponto $V_{corrente}$ e o terra) proporcional a corrente elétrica no "líquido" (azul) e a tensão entre RE e WE (amarelo)



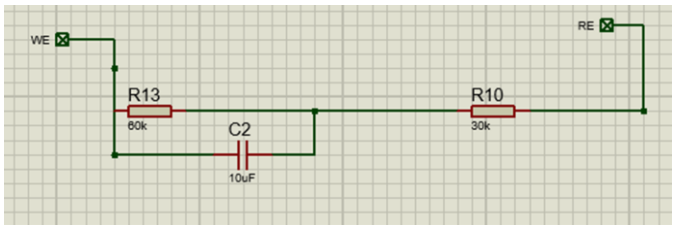
Fonte: Elaborado pelo autor (2025).

Figura B-20 - Tela do Virtual Terminal no teste 4 de simulação no Proteus 8



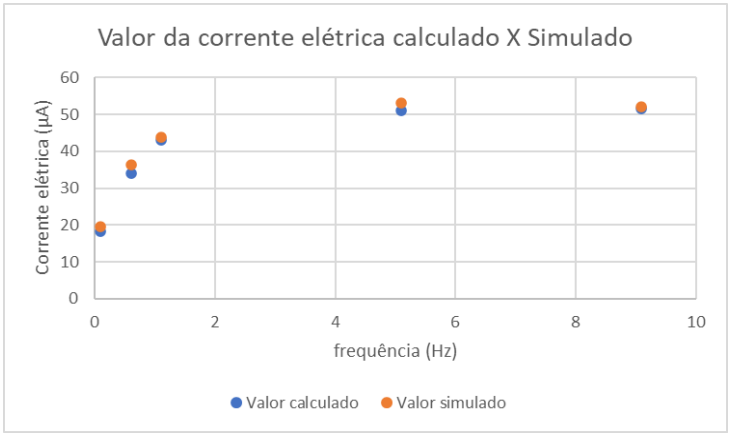
Fonte: Elaborado pelo autor (2025).

Figura B-21 - Combinação de componentes elétricos entre RE e WE no teste 4 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

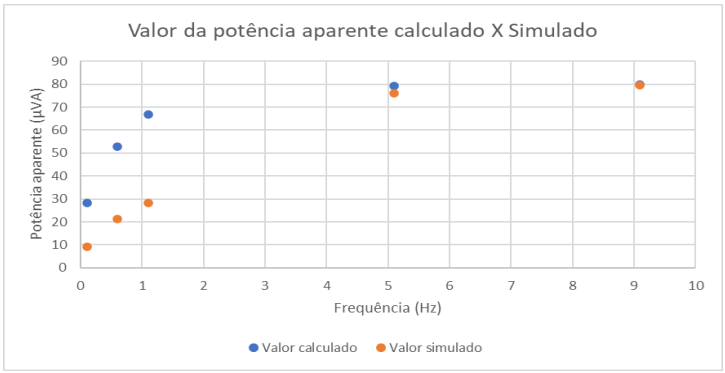
Figura B-22 - Corrente elétrica calculada (azul) X simulada (laranja) no teste 4 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

Na figura B-22 o desvio máximo observado entre os valores calculados e medidos foi de 2,37 µA. Alguns motivos para que desvio ocorra são o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico.

Figura B-23 - Potência aparente calculada (azul) X simulada (laranja) no teste 4 de simulação no Proteus 8

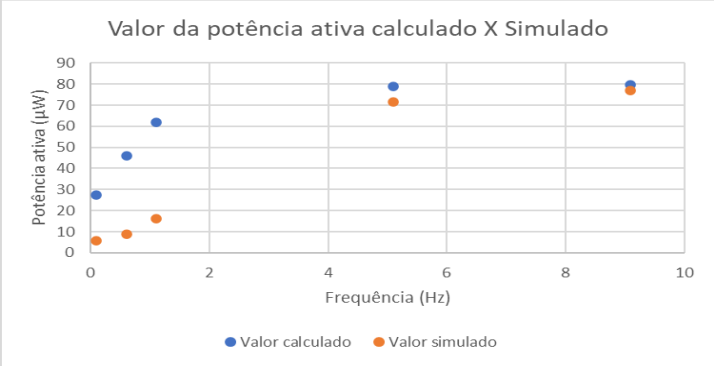


Fonte: Elaborado pelo autor (2025).

Na figura B-23 o desvio máximo observado entre os valores calculados e medidos foi de 38,51 µVA. Alguns motivos para que desvio ocorra são o erro

numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico. Porém nota-se que nesse caso o maior desvio é muito diferente do menor (eles são em ordem de valor de frequência: 19,05 µVA, 31,55 µVA, 38,51 µVA, 3,18 µVA e 0,16 µVA). O comportamento teórico da potência aparente é de aumentar com a frequência, algo semelhante também ocorre nos resultados da simulação, além disso para frequências maiores os valores simulados são mais semelhantes aos calculados.

Figura B-24 - Potência ativa calculada (azul) X simulada (laranja) no teste 4 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

Na figura B-24 o desvio máximo observado entre os valores calculados e medidos foi de 45,46 µW. Alguns motivos para que desvio ocorra são o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico. Porém nota-se que nesse caso o maior desvio é muito diferente do menor (eles são em ordem de valor de frequência: 21,92 µW, 37,08 µW, 45,46 µW, 7,34 µW e 2,62 µW). O comportamento teórico da potência aparente é de aumentar com a frequência, algo semelhante também ocorre nos resultados da simulação, além disso para frequências maiores os valores simulados são mais semelhantes aos calculados. O problema ocorre em baixas frequências e com uma capacitância maior em C2. Isso demonstra algum possível problema no tempo para executar ou simular os comandos do microcontrolador, uma

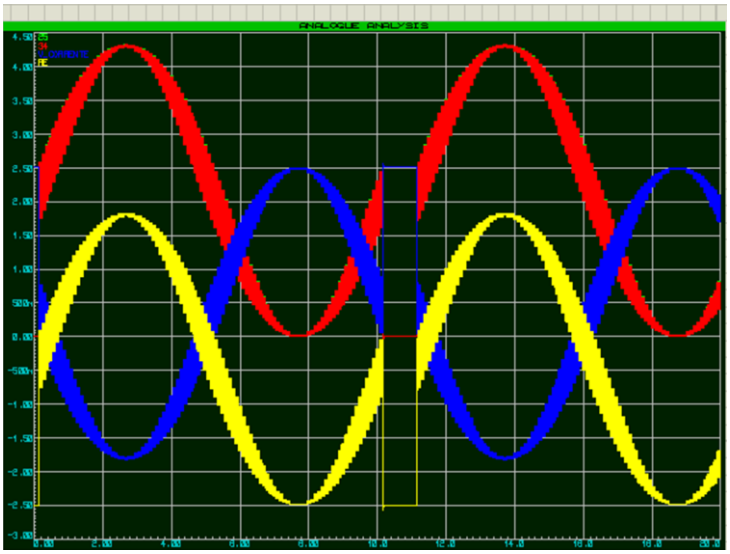
vez que a curva dos resultados calculados segue o mesmo comportamento e se estabiliza em um valor muito parecido, logo algum ajuste poderia ser feito no sentido de considerar a capacitância como se fosse menor do que a de fato usada em todas as simulações (mas isso acabou não sendo feito).

8.2.5 Teste 5

As figuras B-25, B-26, B-27, B-28, B-29 e B-30 são referentes ao teste 5. A figura B-25 mostra a tela do Analogue Analysis, a figura B-26 mostra a tela do Virtual Terminal, a figura B-27 mostra os componentes elétricos entre RE e WE, as figuras B-28, B-29 e B-30 são comparações entre valores simulados e calculados. A figura 81 é referente a corrente elétrica, a figura 82 é referente a potência aparente, a figura 83 é referente a potência ativa.

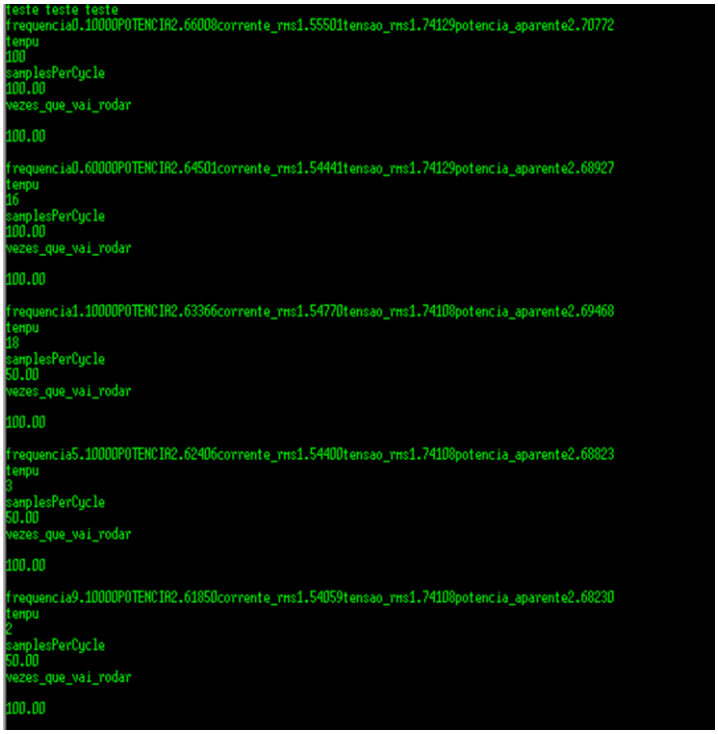
A figura B-28 mostra os valores de corrente elétrica (µA) simulados de cada frequência, que são os valores da figura B-26 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ), conforme a equação 18. A figura B-29 mostra os valores de potência aparente (µVA) simulados de cada frequência, que são os valores da figura B-26 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ) vezes 1,55 sobre 1,77, como na equação 19.A figura B-30 mostra os valores de potência ativa (µW) simulados de cada frequência, que são os valores da figura B-26 divididos por 30 mil (o resistor entre WE e $V_{corrente}$ é de 30 kΩ) vezes 1,55 sobre 1,77, como na equação 20. Os valores calculados seguem o que foi descrito no apêndice A.

Figura B-25 - Tela do Analogue Analysis no teste 5 de simulação no Proteus 8, um recurso do Proteus 8 que serviu para plotar a forma de onda de tensões elétricas importantes para o entendimento do circuito. Na imagem: 25 (verde), uma tensão analógica de referência (entre o ponto 25 e o terra) parecida com a aplicada entre RE e WE, a tensão 34 que é lida pelo Arduino (vermelho) (entre o ponto 25 e o terra), $V_{corrente}$, que é uma tensão elétrica (entre o ponto $V_{corrente}$ e o terra) proporcional a corrente elétrica no "líquido" (azul) e a tensão entre RE e WE (amarelo)



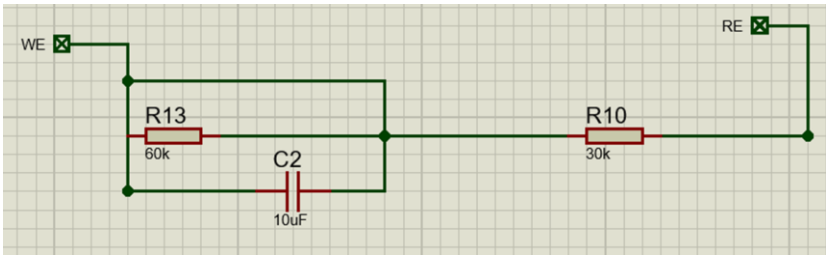
Fonte: Elaborado pelo autor (2025).

Figura B-26 - Tela do Virtual Terminal no teste 5 de simulação no Proteus 8



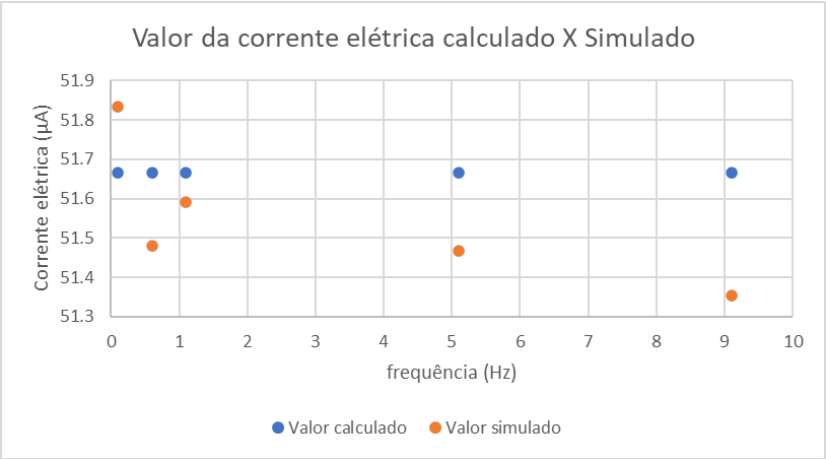
Fonte: Elaborado pelo autor (2025).

Figura B-27 - Combinação de componentes elétricos entre RE e WE no teste 5 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

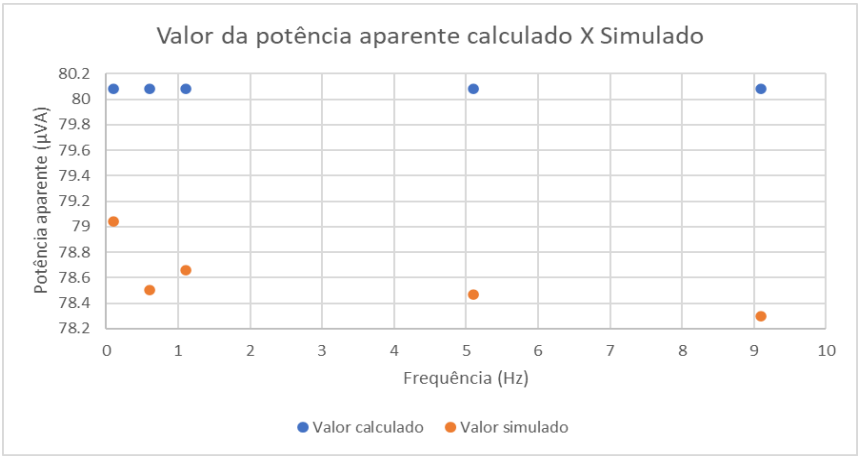
Figura B-28 - Corrente elétrica calculada (azul) X simulada (laranja) no teste 5 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

Na figura B-28 o desvio máximo observado entre os valores calculados e medidos foi de 0,31 µA. Alguns motivos para que desvio ocorra são o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico.

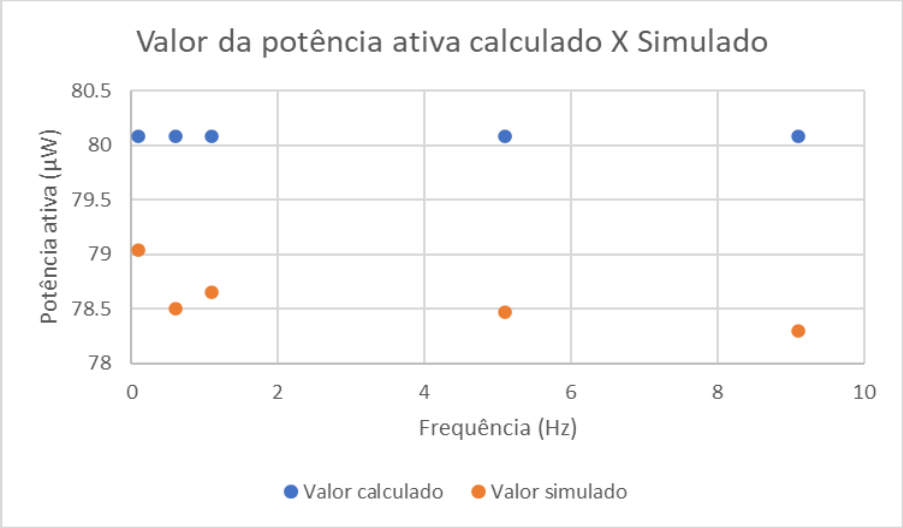
Figura B-29 - Potência aparente calculada (azul) X simulada (laranja) no teste 5 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

Na figura B-29 o desvio máximo observado entre os valores calculados e medidos foi de 1,79 µVA. Alguns motivos para que desvio ocorra são o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico.

Figura B-30 - Potência ativa calculada (azul) X simulada (laranja) no teste 5 de simulação no Proteus 8



Fonte: Elaborado pelo autor (2025).

Na figura B-30 o desvio máximo observado entre os valores calculados e medidos foi de 1,79 µW. Alguns motivos para que desvio ocorra são o erro numérico das integrais aproximadas que o microcontrolador ESP32 realiza, sabendo que o número de amostras é finito e também a aproximação realizada para o circuito que converte o PWM em um valor analógico.

8.3 SCRIPTS FEITOS PARA ESP32 E PYTHON

Observação: instalar todas as bibliotecas antes de rodar os códigos.

8.3.1 Código em python

```
import paho.mqtt.client as mqtt
import matplotlib.pyplot as plt
import math
import time
from openpyxl import Workbook
global resistencia_usada
resistencia_usada=15000
# Configurações do broker MQTT
broker = "broker.emqx.io"  # Insira o endereço do broker
#broker = "167.250.30.226"  # Insira o endereço do broker
port = 1883
topic = "testtopic/massa"  # Tópico que você está usando no ESP32

# Configura a planilha do Excel
wb = Workbook()
ws = wb.active
ws.title = "Dados do ESP32"
#ws.append(["Mensagem MQTT"])  # Cabeçalho
ws.append(["Mensagem MQTT", "frequencia", "POTENCIA", "corrente_rms",
"tensao_rms", "potencia_aparente", "Z", "cos(phi)", "R", "X", "C", "corrente_dc" ])

# Função chamada quando a conexão com o broker é estabelecida
def on_connect(client, userdata, flags, rc):
    print("Conectado ao broker com código de resultado: " + str(rc))
    client.subscribe(topic)

# Função chamada quando uma mensagem é recebida
def on_message(client, userdata, msg):
    global mensagem_anterior_por_que_se_for_igual_tem_que_ignorar
    global frequencia_grafico
    global POTENCIA_grafico
    global corrente_rms_grafico
    global tensao_rms_grafico
    global potencia_aparente_grafico
    global corrente_dc_excel_grafico
```

```
global inicio

global frequencia_excel
global POTENCIA_excel
global corrente_rms_excel
global tensao_rms_excel
global potencia_aparente_excel
global corrente_dc_excel

global Z_excel
global COSSENO_DO_PHI_excel
global R_excel
global X_excel
global C_excel
global corrente_dc

mensagem = msg.payload.decode()
if mensagem==mensagem_anterior_por_que_se_for_igual_tem_que_ignorar:
    mensagem=mensagem
    print("ignorando mensagem")
else:
    print("não ignorando mensagem \n")
    #print("mensagem")
    #print(mensagem)
    #print("mensagem_anterior_por_que_se_for_igual_tem_que_ignorar")
    #print(mensagem_anterior_por_que_se_for_igual_tem_que_ignorar)
    #print("mensagi")
    #print(f"Mensagem recebida: {mensagem}")
    mensagem_anterior_por_que_se_for_igual_tem_que_ignorar=mensagem

    if not mensagem:
        print("Mensagem recebida está vazia.")
        return
    else:
        if (len(mensagem)<8):
            #print("if (len(mensagem)<8):")
            try:
                mensagem=mensagem
                print("mesagem pequena")
                #print(mensagem)
                mensagem_sem_ser_em_binario=float(mensagem[-1])*1
                if (len(mensagem)>1):
```

```
                mensagem_sem_ser_em_binario+=float(mensagem[-2])*2
            if (len(mensagem)>2):
                mensagem_sem_ser_em_binario+=float(mensagem[-3])*4
            if (len(mensagem)>3):
                mensagem_sem_ser_em_binario+=float(mensagem[-4])*8
            if (len(mensagem)>4):
                mensagem_sem_ser_em_binario+=float(mensagem[-5])*16
            if (len(mensagem)>5):
                mensagem_sem_ser_em_binario+=float(mensagem[-6])*32
            if (len(mensagem)>6):
                mensagem_sem_ser_em_binario+=float(mensagem[-7])*64
            if (len(mensagem)>7):
                mensagem_sem_ser_em_binario+=float(mensagem[-8])*128
            mensagem_sem_ser_em_binario=mensagem_sem_ser_em_binario/100
        except:
            mensagem_sem_ser_em_binario = mensagem
        #print(f"Mensagem recebida: {str(mensagem_sem_ser_em_binario)}")
        #ws.append([mensagem_sem_ser_em_binario])  # Salva a mensagem na
planilha
        #wb.save("dados_esp32.xlsx")  # Salva a planilha
    else:
        print("else")
        #print(mensagem)
        try:
            # Tenta converter para float

mensagem_sem_ser_em_binario=float(mensagem[0])*128+float(mensagem[1])*64+f
loat(mensagem[2])*32+float(mensagem[3])*16+float(mensagem[4])*8+float(mensage
m[5])*4+float(mensagem[6])*2+float(mensagem[7])
            mensagem_sem_ser_em_binario=mensagem_sem_ser_em_binario/100
        except ValueError:
            # Se der erro, mantém como string
            mensagem_sem_ser_em_binario = mensagem

        #print(f"Mensagem recebida: {str(mensagem_sem_ser_em_binario)}")
        dados = mensagem_sem_ser_em_binario.split("\n")
        for indice_dos_dados in range (len(dados)):
            #print("dados[indice_dos_dados]")
            #print(dados[indice_dos_dados])
            #ws.append([dados[indice_dos_dados]])  # Salva a mensagem na
planilha
            wb.save("dados_esp32.xlsx")  # Salva a planilha
```

```
                if (dados[indice_dos_dados]=="frequencia"):
                    frequencia_grafico.append(float(dados[indice_dos_dados+1]))
                    frequencia_excel=float(dados[indice_dos_dados+1])
                if (dados[indice_dos_dados]=="POTENCIA"):
                    POTENCIA_grafico.append(float(dados[indice_dos_dados+1]))
                    POTENCIA_excel=float(dados[indice_dos_dados+1])
                if (dados[indice_dos_dados]=="corrente_rms"):
                    corrente_rms_grafico.append(float(dados[indice_dos_dados+1]))
                    corrente_rms_excel=float(dados[indice_dos_dados+1])
                if (dados[indice_dos_dados]=="tensao_rms"):
                    tensao_rms_grafico.append(float(dados[indice_dos_dados+1]))
                    tensao_rms_excel=float(dados[indice_dos_dados+1])
                if (dados[indice_dos_dados]=="potencia_aparente"):

potencia_aparente_grafico.append(float(dados[indice_dos_dados+1]))
                    potencia_aparente_excel=float(dados[indice_dos_dados+1])
                    x.append(frequencia_grafico[len(potencia_aparente_grafico)-1])

Z_excel=tensao_rms_excel*tensao_rms_excel/potencia_aparente_excel

COSSENO_DO_PHI_excel=POTENCIA_excel/potencia_aparente_excel
                    R_excel=Z_excel*COSSENO_DO_PHI_excel

X_excel=Z_excel*math.sin(math.acos(abs(COSSENO_DO_PHI_excel)))

C_excel=1/(2*math.pi*frequencia_grafico[len(potencia_aparente_grafico)-1]*X_excel)

                    X_excel=resistencia_usada/R_excel*X_excel
                    R_excel=resistencia_usada

C_excel=1/(2*math.pi*frequencia_grafico[len(potencia_aparente_grafico)-1]*X_excel)


                    ws.append([time.time()-inicio, frequencia_excel, POTENCIA_excel,
corrente_rms_excel, tensao_rms_excel,
potencia_aparente_excel,Z_excel,COSSENO_DO_PHI_excel,R_excel,X_excel,
C_excel,0])  # Salva a mensagem na planilha
                    wb.save("dados_esp32.xlsx")  # Salva a planilha
                    #print("potencia_aparente_grafico")
                    #print(potencia_aparente_grafico)
                    plt.plot(x, potencia_aparente_grafico, color="red")
                    plt.title("Gráfico de Dispersão")
```

```
                    plt.xlabel("Frequencia")
                    plt.ylabel("Potencia")
                    plt.show()
                if (dados[indice_dos_dados]=="corrente_dc"):
                    print("corrente dc")
                    print(dados[indice_dos_dados+1])
                    corrente_dc_grafico.append(float(dados[indice_dos_dados+1]))
                    corrente_dc_excel=float(dados[indice_dos_dados+1])

                    x.append(time.time()-inicio)
                    ws.append([len(corrente_dc_grafico), 0, 0, 0, 0, 0,0,0,0, 0,0,
corrente_dc_excel])  # Salva a mensagem na planilha
                    wb.save("dados_esp32.xlsx")  # Salva a planilha
                    plt.plot(x, corrente_dc_grafico, color="red")
                    plt.title("Gráfico de Dispersão")
                    plt.xlabel("Tempo")
                    plt.ylabel("Corrente")
                    plt.show()
global inicio
inicio=time.time()
global mensagem_anterior_por_que_se_for_igual_tem_que_ignorar
mensagem_anterior_por_que_se_for_igual_tem_que_ignorar="nunca vai ser isso
então essa é a inicial, depois vai ser outra"
global frequencia_grafico
global POTENCIA_grafico
global corrente_rms_grafico
global tensao_rms_grafico
global potencia_aparente_grafico

global corrente_dc_grafico
global x

global frequencia_excel
global POTENCIA_excel
global corrente_rms_excel
global tensao_rms_excel
global potencia_aparente_excel
global Z_excel
global COSSENO_DO_PHI_excel
global R_excel
global X_excel
global C_excel
```

```
global corrente_dc_excel

x=[]
frequencia_grafico=[]
POTENCIA_grafico=[]
corrente_rms_grafico=[]
tensao_rms_grafico=[]
potencia_aparente_grafico=[]
corrente_dc_grafico=[]


# Configura o cliente MQTT
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

# Conecta-se ao broker e inicia o loop
client.connect(broker, port, 60)
client.loop_start()

# Mantenha o script rodando
try:
    while True:
        pass
except KeyboardInterrupt:
    print("Desconectando...")
    client.loop_stop()
    client.disconnect()
```

8.3.2 Código para esp32

```
#include <Arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>

bool ackRecebido = false;              // Flag para saber se o "ok" foi recebido
unsigned long tempoEnvio = 0;          // Tempo do último envio
const unsigned long timeout = 5000;    // Tempo limite para reenviar a mensagem
String ultimaMensagem;                 // Armazena a última mensagem enviada
```

```
char id_da_mensagem='0'; // é para saber se a mensagem realmente foi enviada

int saida;
const int analogPin = 34;  // Define o pino analógico (GPIO 34 no ESP32)
float voltage = 0.0;       // Variável para armazenar a tensão lida

// Configurações Wi-Fi
const char* ssid = "DESK";         // Insira o nome da sua rede Wi-Fi
const char* password = "12345678";    // Insira a senha da sua rede Wi-Fi

// Configurações do Broker Mosquitto
const char* mqtt_server = "broker.emqx.io";
const int mqtt_port = 1883;
const char* mqtt_topic = "testtopic/massa";  // Escolha um tópico para enviar dados

WiFiClient espClient;
PubSubClient client(espClient);
const int pinDigital = 15; // Substitua pelo número do pino que você está usando

// Configurações
const int dacPin = 25; // Pino DAC do ESP32 (25 ou 26)
//const float samplesPerCycle = 50;
float samplesPerCycle = 50;
const int amplitude = 10; // Amplitude da onda (0-255 para DAC de 8 bits)
const int offset = 128; // Offset da senoide (centra a onda em 0-255)
//const int analogPin = 34;  // Define o pino analógico (GPIO 34 no ESP32)

// Quantidade de pontos da senoide

// Tabela para armazenar a senoide
int sineTable; // Ajuste o tamanho para suportar a taxa de amostragem



// Configuração do tópico de resposta do broker
const char* ack_topic = "testtopic/massa";
```

```
void callback(char* topic, byte* payload, unsigned int length) {
  String mensagemRecebida = "";
  for (unsigned int i = 0; i < length; i++) {
    mensagemRecebida += (char)payload[i];
  }
  Serial.println("mensagemRecebida");
  Serial.println(mensagemRecebida);
  Serial.println("ok");
  Serial.println("id_da_mensagem");
  Serial.println(id_da_mensagem);
  String soma_id_mensagem=String('o')+String('k')+String(id_da_mensagem);
  Serial.println("ok+id_da_mensagem");
  Serial.println(soma_id_mensagem);
  if (String(topic) == ack_topic && mensagemRecebida == soma_id_mensagem) {
    Serial.println("Confirmação recebida do broker: OK");
    ackRecebido = true; // Marca que o "ok" foi recebido
  }
  else {
      ackRecebido = true;
      Serial.println("funcionando");
  }
}

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Conectando-se a ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
```

```
  Serial.println("Conectado ao Wi-Fi!");
  Serial.println("\n");
  Serial.print("IP obtido: ");
Serial.println(WiFi.localIP());
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Conectando ao broker MQTT...");
    Serial.println("\n");
    if (client.connect("ESP32Client122")) {
      Serial.println("Conectado!");
    } else {
      Serial.print("Falha, rc=");
      Serial.print(client.state());
      Serial.println(" tentando novamente em 5 segundos");
      delay(1000);
    }
  }
}




void setup() {
    pinMode(pinDigital, INPUT);
    client.subscribe(ack_topic);
    analogReadResolution(12);  // Define a resolução do ADC para 12 bits
    Serial.begin(9600);
    setup_wifi();
    client.setServer(mqtt_server, mqtt_port);
    saida = 0;
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback); // Configura o callback para mensagens recebidas
```

```
    analogReadResolution(12);
    Serial.begin(9600);
  // Gera a tabela da senoide
    Serial.print("funcionando");
    Serial.print("\n");
}
char topic[] = "testtopic/massa";    // Simula o tópico recebido
//byte payload[] = {'o', 'k','0'};            // Simula o payload recebido
//unsigned int length = sizeof(payload);  // Tamanho do payload
void envia_mensagem(String mensagem)
{
        int intervalo=500;
        String message = mensagem;
        ackRecebido = false;
        id_da_mensagem='1';
        // Publica a mensagem XML no tópico MQTT
        for (int vezes_que_vai_mandar_a_mensagem = 0;
vezes_que_vai_mandar_a_mensagem < 1;
vezes_que_vai_mandar_a_mensagem++)
        {
                //callback(topic, payload, length);
                if (client.publish(mqtt_topic, message.c_str())) {
                  Serial.println("Mensagem XML enviada com sucesso: ");
                } else {
                  Serial.println("Falha ao enviar a mensagem XML.");
                }
                delay(intervalo);
                if (!client.connected()) {
                        reconnect();
                        }
        }
}
```

```
void loop() {
  int valor = digitalRead(pinDigital);
  if (valor == HIGH)
  {
    Serial.println("O pino está em nível ALTO (HIGH)");

      dacWrite(26, 128);
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  // Gera a onda seno no DAC
  float frequencia=0.1;
  float corrente;
  float VRMS=3.3/(1.414214);

  for (int indice_frequencia = 1; indice_frequencia < 100; indice_frequencia++)
  {




        if (frequencia>100)
        {
            samplesPerCycle=25;

        }
        if (frequencia>200)
        {
            samplesPerCycle=15;

        }
        if (frequencia>400)
        {
            samplesPerCycle=10;
```

```
}
if (frequencia>800)
{
    samplesPerCycle=8;

}
if (frequencia>1000)
{
    samplesPerCycle=4;

}
//frequencia=frequencia*2;
float valor_medio_corrente=0;
float vezes_que_vai_rodar=samplesPerCycle*10;

if (frequencia<1)
{
    vezes_que_vai_rodar=samplesPerCycle;
}
if (frequencia>100)
{
    vezes_que_vai_rodar=samplesPerCycle*20;
}
if (frequencia>500)
{
    vezes_que_vai_rodar=samplesPerCycle*50;
}
if (frequencia>1000)
{
    vezes_que_vai_rodar=samplesPerCycle*100;
}
for (int i = 0; i < vezes_que_vai_rodar; i++) {
  float angle = 2 * PI * i / samplesPerCycle;
  sineTable = offset + (amplitude * sin(angle));
  dacWrite(dacPin, sineTable);
  corrente=float(analogRead(analogPin)*3.3);
```

```
  corrente=corrente/4095;
  valor_medio_corrente=valor_medio_corrente+corrente;
  int tempu=int (1000000 / frequencia/samplesPerCycle);
  delayMicroseconds(tempu);

}
valor_medio_corrente=valor_medio_corrente/vezes_que_vai_rodar;
/*
Serial.print("valor_medio_corrente");
Serial.print("\n");
Serial.print(valor_medio_corrente);
Serial.print("\n");
delay(2000);
Serial.print("frequencia");
Serial.print("\n");
Serial.print(frequencia);
Serial.print("\n");

*/


double POTENCIA=0;
float iquad=0;
double tensao=0;
float iantiga;
float Q=0;
double tensa0_vezes_corrente=0;
float tensao_rms=0;
float corrente_rms=0;

for (int i = 0; i < vezes_que_vai_rodar; i++) {
  float angle = 2* PI * i / samplesPerCycle;
  sineTable = offset + (amplitude * sin(angle));
  dacWrite(dacPin, sineTable);
  iantiga=corrente;
  corrente=float(analogRead(analogPin)*3.3);
```

```
corrente=corrente/4095;
//Serial.print("corrente antes de subtrair");
//Serial.print("\n");
//Serial.print(corrente);
//Serial.print("\n");

corrente=corrente-float(valor_medio_corrente);

iquad=(corrente-iantiga)/(2*PI*1/ samplesPerCycle);
tensao=double((sineTable-offset)*3.3/255);
tensa0_vezes_corrente=double(corrente*tensao);
POTENCIA+=tensa0_vezes_corrente;
Q=float(Q)+float(tensao*iquad);

int tempu=int (1000000 / frequencia/samplesPerCycle);
delayMicroseconds(tempu);
//delay(20000);
corrente_rms+=corrente*corrente;
tensao_rms+=tensao*tensao;

/*
Serial.print("--------------------------");


Serial.print("valor_medio_corrente");
Serial.print("\n");
Serial.print(valor_medio_corrente);
Serial.print("\n");



Serial.print("tensao");
Serial.print("\n");
Serial.print(tensao);
Serial.print("\n");
```

```
  Serial.print("iquad");
  Serial.print("\n");
  Serial.print(iquad);
  Serial.print("\n");


  Serial.print("corrente");
  Serial.print("\n");
  Serial.print(corrente);
  Serial.print("\n")
  //Serial.print("sineTable");
  //Serial.print("\n");
  //Serial.print(sineTable);
  //Serial.print("\n");

  //Serial.print("POTENCIA");
  //Serial.print("\n");
  //Serial.print(POTENCIA);
  //Serial.print("\n");



  //Serial.print("Q");
  //Serial.print("\n");
  //Serial.print(Q);
  //Serial.print("\n");

  //Serial.print(tensa0_vezes_corrente);
  */
}


if (!client.connected()) {
 reconnect();
}
```

```
        Serial.print("Conexão MQTT: ");
        Serial.println(client.connected());



        POTENCIA=POTENCIA/vezes_que_vai_rodar;
        Q=Q/vezes_que_vai_rodar;
        if (!client.connected()) {
         reconnect();
        }
        String
mensagem_a_ser_enviada="\nfrequencia\n"+String(frequencia,5)+"\nPOTENCIA\n"+String
(POTENCIA,5);
        tensao_rms=tensao_rms/vezes_que_vai_rodar;
        tensao_rms=sqrt(tensao_rms);

        corrente_rms=corrente_rms/vezes_que_vai_rodar;
        corrente_rms=sqrt(corrente_rms);

        if (!client.connected()) {
         reconnect();
        }

mensagem_a_ser_enviada+="\ncorrente_rms\n"+String(corrente_rms,5)+"\ntensao_r
ms\n"+String(tensao_rms,5);

        float potencia_aparente=corrente_rms*tensao_rms;
        if (!client.connected()) {
         reconnect();
        }


mensagem_a_ser_enviada+="\npotencia_aparente\n"+String(potencia_aparente,5);
        envia_mensagem(mensagem_a_ser_enviada);
```

```
int intervalo=20000;

delay(intervalo);
if (frequencia<1)
{
   frequencia=frequencia+0.5;

}

//frequencia=frequencia*1.1;

else if (frequencia<10)
{
   frequencia=frequencia+1;

}

else if (frequencia<10)
{
   frequencia=frequencia+1;

}
else if (frequencia<100)
{
   frequencia=frequencia*1.2;

}

else if (frequencia<1000)
{
   frequencia=frequencia*1.1;

}
else if (frequencia>1000)
{
   frequencia=frequencia*1.01;
```

```
      }
  }




  } else {
    float corrente=0;

    for (unsigned int i = 0; i < 100000; i++)
    {
          corrente+=float(analogRead(analogPin)*3.3)/4095;

    }
    corrente=corrente/100000;
    Serial.println("O pino está em nível BAIXO (LOW)");
    Serial.println(corrente,5);
    dacWrite(25, 62);
    String mensagem_a_ser_enviada="\ncorrente_dc\n"+String(corrente,5);

    envia_mensagem(mensagem_a_ser_enviada);


    //delay (1000);



  }

}
```

8.3.3 Código para simulino no proteus

```
#include <SoftwareSerial.h>
```

```
#include <Arduino.h>
#include <PubSubClient.h>
SoftwareSerial mySerial(3, 4); // RX no pino 10, TX no pino 11

bool ackRecebido = false;              // Flag para saber se o "ok" foi recebido
unsigned long tempoEnvio = 0;          // Tempo do último envio
const unsigned long timeout = 5000;    // Tempo limite para reenviar a mensagem
String ultimaMensagem;                 // Armazena a última mensagem enviada

char id_da_mensagem='0'; // é para saber se a mensagem realmente foi enviada
long tempu;
int saida;
const int analogPin = A0;  // Define o pino analógico (GPIO 34 no ESP32)
float voltage = 0.0;       // Variável para armazenar a tensão lida

// Configurações Wi-Fi
const char* ssid = "DESK";         // Insira o nome da sua rede Wi-Fi
const char* password = "12345678";    // Insira a senha da sua rede Wi-Fi

// Configurações do Broker Mosquitto
//const char* mqtt_server = "broker.emqx.io";
const char* mqtt_server = "test.mosquitto.org";
const int mqtt_port = 1883;
const char* mqtt_topic = "testtopic/massa";  // Escolha um tópico para enviar dados



// Configurações
//const int dacPin = 9; // Pino DAC do ESP32 (9) change
const int dacPin = 5; // Pino DAC do ESP32 (9)
//const float samplesPerCycle = 50;
float samplesPerCycle = 50;
const int amplitude = 126; // Amplitude da onda (0-255 para DAC de 8 bits)
const int offset = 127; // Offset da senoide (centra a onda em 0-255)
//const int analogPin = 34;  // Define o pino analógico (GPIO 34 no ESP32)
```

```
// Quantidade de pontos da senoide

// Tabela para armazenar a senoide
int sineTable; // Ajuste o tamanho para suportar a taxa de amostragem



// Configuração do tópico de resposta do broker
const char* ack_topic = "testtopic/massa";
void envia_mensagem(String mensagem)
{
	int intervalo=1500;
	String message = mensagem;
	ackRecebido = false;
	id_da_mensagem='1';
	// Publica a mensagem XML no tópico MQTT
	for (int vezes_que_vai_mandar_a_mensagem = 0;
vezes_que_vai_mandar_a_mensagem < 1;
vezes_que_vai_mandar_a_mensagem++)
	{

		delay(intervalo);
	}
}

void callback(char* topic, byte* payload, unsigned int length) {
  String mensagemRecebida = "";
  for (unsigned int i = 0; i < length; i++) {
    mensagemRecebida += (char)payload[i];
  }
  Serial.println("mensagemRecebida");
  Serial.println(mensagemRecebida);
  Serial.println("ok");
  Serial.println("id_da_mensagem");
  Serial.println(id_da_mensagem);
  String soma_id_mensagem=String('o')+String('k')+String(id_da_mensagem);
  Serial.println("ok+id_da_mensagem");
```

```
  Serial.println(soma_id_mensagem);
  if (String(topic) == ack_topic && mensagemRecebida == soma_id_mensagem) {
    Serial.println("Confirmação recebida do broker: OK");
    ackRecebido = true; // Marca que o "ok" foi recebido
  }
  else {
      ackRecebido = true;
      Serial.println("funcionando");
  }
}


void setup() {

    //TCCR1B = TCCR1B & 0b11111000 | 0x01; // Prescaler = 1 (alta frequência) pino
9 aumenta a frequência do pwm

    mySerial.begin(9600);   // Comunicação serial alternativa

    pinMode(dacPin, OUTPUT);

    saida = 0;



  // put your setup code here, to run once:
  pinMode(5, OUTPUT);

}

void loop() {
   float frequencia=0.1;
  float corrente;
  float VRMS=5/(1.414214);
  //analogWrite(dacPin, 100);
  mySerial.println("teste teste teste");
```

```
   for (int indice_frequencia = 1; indice_frequencia < 200; indice_frequencia++)
{
  delay (100);



        if (frequencia<10)
        {
            samplesPerCycle=50;

        }

        if (frequencia<1)// para não ficar lento
        {
            samplesPerCycle=100;

        }

        if (frequencia<0.1)// para não ficar lento
        {
            samplesPerCycle=200;

        }

        if (frequencia>10)
        {
            samplesPerCycle=20;

        }
        if (frequencia>20)
        {
            samplesPerCycle=10;

        }
        if (frequencia>40)
        {
```

```
    samplesPerCycle=5;

}
if (frequencia>80)
{
    samplesPerCycle=3;

}
if (frequencia>1000)
{
    samplesPerCycle=4;

}
//frequencia=frequencia*2;
float valor_medio_corrente=0;
float vezes_que_vai_rodar=samplesPerCycle*2;

if (frequencia<1)
{
    vezes_que_vai_rodar=samplesPerCycle;
}

if (frequencia>100)
{
    //vezes_que_vai_rodar=samplesPerCycle*20;
    vezes_que_vai_rodar=samplesPerCycle;
}
if (frequencia>500)
{
    //vezes_que_vai_rodar=samplesPerCycle*50;
    vezes_que_vai_rodar=samplesPerCycle;
}
if (frequencia>1000)
{
    //vezes_que_vai_rodar=samplesPerCycle*100;
    vezes_que_vai_rodar=samplesPerCycle;
```

```
}
for (int i = 0; i < vezes_que_vai_rodar; i++) {
  float angle = 2 * PI * i / samplesPerCycle;
  sineTable = offset + (amplitude * sin(angle));
  analogWrite(dacPin, sineTable);
  corrente=float(analogRead(analogPin)*5);
  corrente=corrente/1024;
  valor_medio_corrente=valor_medio_corrente+corrente;
  long tempu=1000L / frequencia/samplesPerCycle;
  //delayMicroseconds(tempu);
  delay(tempu);
}
analogWrite(dacPin, 0);
delay(1000);
valor_medio_corrente=valor_medio_corrente/vezes_que_vai_rodar;

double POTENCIA=0;
float iquad=0;
double tensao=0;
float iantiga;
float Q=0;
double tensa0_vezes_corrente=0;
float tensao_rms=0;
float corrente_rms=0;

for (int i = 0; i < vezes_que_vai_rodar; i++)
{
  float angle = 2* PI * i / samplesPerCycle;
  sineTable = offset + (amplitude * sin(angle));
  analogWrite(dacPin, sineTable);
  iantiga=corrente;
  corrente=float(analogRead(analogPin)*5);
  corrente=corrente/1024;

  corrente=corrente-float(valor_medio_corrente);
```

```
        iquad=(corrente-iantiga)/(2*PI*1/ samplesPerCycle);
        tensao=double(sineTable-offset);
        tensao=tensao*5/256;
        tensa0_vezes_corrente=double(corrente*tensao);
        POTENCIA+=tensa0_vezes_corrente;
        Q=float(Q)+float(tensao*iquad);

        tempu=1000L/frequencia/samplesPerCycle;

        delay(tempu);

        corrente_rms+=corrente*corrente;
        tensao_rms+=tensao*tensao;




    }

    analogWrite(dacPin, 0);
    delay(1000);
    POTENCIA=POTENCIA/vezes_que_vai_rodar;
    Q=Q/vezes_que_vai_rodar;

    String
mensagem_a_ser_enviada="\nfrequencia\n"+String(frequencia,5)+"\nPOTENCIA\n"+
String(POTENCIA,5);
    tensao_rms=tensao_rms/vezes_que_vai_rodar;
    tensao_rms=sqrt(tensao_rms);

    corrente_rms=corrente_rms/vezes_que_vai_rodar;
    corrente_rms=sqrt(corrente_rms);
```

```
mensagem_a_ser_enviada+="\ncorrente_rms\n"+String(corrente_rms,5)+"\ntensao_r
ms\n"+String(tensao_rms,5);

      float potencia_aparente=corrente_rms*tensao_rms;


mensagem_a_ser_enviada+="\npotencia_aparente\n"+String(potencia_aparente,5);
      envia_mensagem(mensagem_a_ser_enviada);
      mySerial.println(mensagem_a_ser_enviada);
      mySerial.println("tempu");
      mySerial.println(tempu);
      mySerial.println("samplesPerCycle");
      mySerial.println(samplesPerCycle);
      int intervalo=100;

      mySerial.println("vezes_que_vai_rodar");
      mySerial.println("\n");
      mySerial.println(vezes_que_vai_rodar);
      mySerial.println("\n");

      delay(intervalo);
      if (frequencia<0.1)
      {
         frequencia=frequencia*10;

      }
      else if (frequencia<1)
      {
         frequencia=frequencia+0.5;

      }

      //frequencia=frequencia*1.1;

      else if (frequencia<10)
```

```
        {
            frequencia=frequencia+4;

        }

        else if (frequencia<10)
        {
            frequencia=frequencia+4;

        }
        else if (frequencia<100)
        {
            frequencia=frequencia*2;

        }

        else if (frequencia<1000)
        {
            frequencia=frequencia*1.1;

        }
        else if (frequencia>1000)
        {
            frequencia=frequencia*1.01;

        }
    }


}
```