UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA – EST

GABRIEL SILVA DA ROCHA

PROTÓTIPO DE SISTEMA INDUSTRIAL PARA MONITORAMENTO DE PARÂMETROS ELÉTRICOS E DA QUALIDADE DE FORNECIMENTO DE ENERGIA ELÉTRICA

Manaus
2022

GABRIEL SILVA DA ROCHA

PROTÓTIPO DE SISTEMA INDUSTRIAL PARA MONITORAMENTO DE PARÂMETROS ELÉTRICOS E DA QUALIDADE DE FORNECIMENTO DE ENERGIA ELÉTRICA

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentado à banca avaliadora do Curso de Engenharia Eletrônica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Engenheiro Eletrônico.

Orientador: Fábio de Sousa Cardoso, Dr.

Manaus

2022

Universidade do Estado do Amazonas – UEA
Escola Superior de Tecnologia - EST

Banca Avaliadora composta por: Data da defesa: <17/10/2022>.
Prof. Fábio de Sousa Cardoso, Dr *(Orientador)*
Prof. Israel Gondres Torné, Dr
Prof. Rubens de Andrade Fernandes, Me

CIP – Catalogação na Publicação

Rocha, Gabriel Silva da

Protótipo de sistema industrial para monitoramento de parâmetros elétricos e da qualidade de fornecimento de energia elétrica/ Gabriel Silva da Rocha; [orientado por] Fábio de Sousa Cardoso. – Manaus: 2022.
62 p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Eletrônica). Universidade do Estado do Amazonas, 2022.

1. Monitoramento de parâmetros elétricos. 2. ADE7758. 3. ESP32. 4. MQTT. 5. SQLite. 6. Armazenamento de dados. I. Cardoso, Fábio de Sousa.

GABRIEL SILVA DA ROCHA

PROTÓTIPO DE SISTEMA INDUSTRIAL PARA MONITORAMENTO DE PARÂMETROS ELÉTRICOS E DA QUALIDADE DE FORNECIMENTO DE ENERGIA ELÉTRICA

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Eletrônica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Engenheiro Eletrônico.

Nota obtida: 9,8 (Nove vírgula oito)

Aprovada em 17 / 10 / 2022

Área de concentração: Sistemas Embarcados

BANCA EXAMINADORA

Orientador: Fábio de Sousa Cardoso, Dr.

Avaliador: Israel Gondres Torné, Dr.

Avaliador: Rubens de Andrade Fernandes, Me.

Manaus 2022

AGRADECIMENTOS

A Deus, por me dar saúde. Aos meus pais e meus irmãos, por todo apoio dado durante esta trajetória. À Greicy pelo companheirismo e por todo incentivo dado durante os anos que estive na faculdade. Aos meus amigos da universidade que me fizeram chegar até aqui graças à colaboração no decorrer da graduação. Ao LSE, em especial aos meus amigos Rubens Fernandes, Beatriz Santos, Heitor Lifsitch, Samuel Bruno e Matheus Assunção. A todos os professores que contribuíram durante minha formação acadêmica.

Nada é fácil...
Ninguém disse que seria fácil...

RESUMO

O presente trabalho tem como objetivo o desenvolvimento de um protótipo de sistema industrial para monitoramento de parâmetros elétricos e da qualidade de fornecimento de energia elétrica. O sistema deve realizar a aquisição dos parâmetros elétricos, realizar o processamento desses dados e enviá-los para que sejam armazenados, onde será possível gerar gráficos com os parâmetros coletados. Inicialmente, será apresentada a fundamentação teórica responsável pela definição dos conceitos e aplicações dos assuntos e tecnologias utilizadas no decorrer deste trabalho, sendo: qualidade de fornecimento de energia elétrica, eficiência energética, *smart meters*, transdutores de parâmetros elétricos, ADE7758, microcontroladores, ESP32, Internet das Coisas, TCP/IP, MQTT e base de dados SQLite. Posteriormente, são apresentadas as etapas e os materiais necessários para a elaboração dos protótipos de *hardware* e *firmware*, e do *script* de armazenamento dos dados coletados, bem como as ferramentas utilizadas e a própria implementação das etapas apresentadas anteriormente. Como resultado dos testes de validação submetidos no decorrer da implementação deste trabalho, serão exibidos os gráficos gerados a partir de medições coletadas e que serão comparadas com as medições coletadas a partir de um medidor comercial. Por fim, os dados obtidos mostraram que o protótipo de sistema desenvolvido é capaz de monitorar os parâmetros elétricos e a qualidade de fornecimento de energia elétrica ao ser comparado com um analisador de energia comercial.

Palavras-chave: Monitoramento de parâmetros elétricos, ADE7758, ESP32, MQTT, SQLite, Armazenamento de dados.

ABSTRACT

The present work aimed to develop a prototype of an industrial system for monitoring electrical parameters and the quality of electricity supply. The system must perform the acquisition of electrical parameters, perform the processing of these data and send them to be stored, where it will be possible to generate graphs with the parameters collected. Initially, the theoretical foundation responsible for defining the concepts and applications of the subjects and technologies used in the course of this work will be presented, namely: quality of electricity supply, energy efficiency, smart meters, electrical parameter transducers, ADE7758, microcontrollers, ESP32, Internet of Things, TCP/IP, MQTT and SQLite database. Subsequently, the steps and materials necessary for the development of hardware and firmware prototypes, and the script for storing the collected data are presented, as well as the tools used and the implementation of the steps presented above. As a result of the validation tests submitted during the implementation of this work, the graphs generated from the measurements collected will be displayed and will be compared with the measurements collected from a commercial meter. Finally, the data obtained showed that the prototype system developed is capable of monitoring the electrical parameters and the quality of electrical energy supply when compared with a commercial energy analyzer.

Keywords: monitoring electrical parameters, ADE7758, ESP32, MQTT, SQLite, data storage.

SUMÁRIO

INTRODUÇÃO

Este trabalho tem como tema o desenvolvimento de um protótipo de sistema industrial para monitoramento de parâmetros elétricos e da qualidade de fornecimento de energia elétrica. O consumo de energia elétrica no mundo tem se tornado algo cada vez mais abrangido em discussões sobre a tomada de decisões para o melhor aproveitamento dos recursos utilizados para a obtenção da energia elétrica. Uma das maneiras de melhorar esse aproveitamento é a aplicação de regras para o uso eficiente da energia. O setor de energia e a indústria estão recorrendo ao uso do petróleo para a geração de energia elétrica devido à alta no preço do carvão e gás, além dos apagões contínuos (INTERNATIONAL ENERGY AGENCY, 2021).

Sendo um dos maiores consumidores de energia elétrica, a indústria tem um papel importante e um grande potencial no uso eficiente de energia, e este aproveitamento pode ser feito através de ações como o aperfeiçoamento na gestão da distribuição da energia (BRASIL, 2020). O problema observado nesta pesquisa foi a oportunidade de inserir recursos digitais para monitoramento da distribuição de energia elétrica em setores industriais, através de paradigmas digitais, como a Internet das Coisas, do inglês *Internet of Things* (IoT). Desta forma, as estruturas prediais industriais pré-existentes podem se interconectar através de módulos sensores inteligentes.

Como hipótese, propõe-se de que é possível desenvolver um protótipo de sistema para monitoramento de parâmetros elétricos em setores industriais através do desenvolvimento de módulos sensores com recursos de conexão em redes de dados sem fio. Como objetivo, têm-se o desenvolvimento de um protótipo de sistema de monitoramento, no qual será desenvolvido um módulo sensor composto por um microcontrolador que dispõe de conexão com redes de dados sem fio e recursos para realizar o processamento dos dados a serem coletados através de um circuito integrado que fará a digitalização de parâmetros elétricos, onde serão coletados em um quadro de distribuição de baixa tensão, que serão monitorados e armazenados em bancos de dados, tais parâmetros serão a entrada do sistema e consistem em parâmetros de tensão, corrente, fator de potência, potências ativa, reativa e aparente e o monitoramento de eventuais interrupções no fornecimento de energia elétrica. Através da disposição gráfica dos dados armazenados é possível validar as medições comparando-as com medições feitas por dispositivo de medição preciso.

Esta pesquisa justifica-se, pois, através da análise dos dados obtidos por medições realizadas, será possível identificar potenciais pontos de melhorias para um melhor aproveitamento energético no local onde o sistema será instalado, assim como rastrear e

disponibilizar os maiores pontos de consumo da instalação. Nesse contexto, as indústrias, uma das maiores unidades consumidoras situadas em área metropolitana, poderiam usufruir desse sistema para otimizar o gerenciamento de seus recursos energéticos.

Este trabalho está dividido em quatro seções primárias: referencial teórico, materiais e métodos, implementação do projeto e análise dos resultados obtidos.

A primeira seção primária apresenta a revisão literária responsável pela fundamentação deste trabalho, no qual são apresentados conceitos e tecnologias utilizadas no decorrer desta pesquisa. É apresentado o conceito de qualidade do serviço de fornecimento de energia elétrica, suas métricas e definições, assim como o conceito de eficiência energética, apresentando sua definição. Para a contextualização no assunto de medidores inteligentes são apresentados conceitos e aplicações da área. A unidade de processamento central se dá pelo uso de um microcontrolador, cujo definição é apresentada juntamente com o módulo utilizado no protótipo. Os conceitos de Internet das Coisas, dos protocolos TCP/IP, MQTT e da base de dados SQLite são apresentados para melhor entendimento da transmissão e armazenamento dos dados coletados. Os conceitos de transdutores de parâmetros elétricos são apresentados para o entendimento da aquisição dos parâmetros elétricos, bem como a apresentação do ADE7758, responsável pelas medições.

Na segunda seção primária é apresentada a metodologia utilizada para a implementação do projeto, sendo descrito as etapas de forma sequencial para o desenvolvimento do protótipo de sistema de monitoramento, a apresentação de um diagrama em blocos com a sequência seguida e os materiais e ferramentas utilizadas para a implementação.

Na terceira seção primária são apresentadas as etapas de implementação do projeto, utilizando os materiais e ferramentas e aplicando os métodos apresentados na seção primária anterior, além de apresentar os protótipos de *hardware*, *firmware* e o *script* de armazenamento de dados.

Na quarta seção primária é realizada a análise dos resultados obtidos através do teste de validação final no qual o protótipo de sistema foi submetido, sendo realizada a comparação dos dados coletados através do protótipo com os dados coletados através de um dispositivo de medição comercial.

Por fim, na conclusão é apresentada novamente a hipótese e o objetivo comparando ambos com os resultados obtidos e apresentando considerações finais sobre o trabalho desenvolvido, bem como sugestões para trabalhos futuros.

1 REFERENCIAL TEÓRICO

Serão apresentados neste capítulo conceitos que contextualizarão esta pesquisa, tais como a qualidade do serviço de fornecimento de energia elétrica, eficiência energética, *smart meter*, transdutores de parâmetros elétricos, transdutor de tensão e transdutor de corrente elétrica, ADE7758, microcontroladores, Módulo ESP32, *Internet of Things – IoT*, TCP/IP, protocolo MQTT e base de dados SQLite, sendo resultados de pesquisas bibliográficas e levantamento de tecnologias.

1.1 QUALIDADE DO SERVIÇO DE FORNECIMENTO DE ENERGIA ELÉTRICA

Qualidade do fornecimento de energia elétrica é definida basicamente como a continuidade do fornecimento, sendo tratado somente as interrupções ocorridas durante certo período, sendo elas ocasionadas pela ação de dispositivos de proteção, defeitos da rede ou por atividades de manutenção devido a serviços necessários a serem realizados (KAGAN; ROBBA; SCHMIDT, 2009). Segundo Faias e Esteves (2013) a qualidade do fornecimento de energia elétrica segue três aspectos principais: continuidade de fornecimento (confiabilidade e disponibilidade da rede), qualidade da tensão (características da tensão de alimentação) e qualidade comercial (atualidade no atendimento dos pedidos dos clientes), no qual proporcionam um equilíbrio entre a disposição dos clientes que pagam tarifas de rede e suas expectativas sobre níveis mínimos de qualidade de serviço.

A análise do fornecimento é feita através de monitoramento da rede elétrica, onde a continuidade do fornecimento é observada para a identificação de interrupções.

1.2 EFICIÊNCIA ENERGÉTICA

Eficiência energética é a relação entre a quantidade de energia final utilizada e de um bem produzido ou serviço realizado, em que a eficiência está associada à quantidade efetiva de energia utilizada e não à quantidade necessária para realizar um serviço (BRASIL, 2010). Uma das formas de pôr em prática melhorias na eficiência energética, é a utilização de produtos elétricos que consumam menos energia elétrica e, principalmente na indústria, a utilização de equipamentos mais eficientes, ou seja, que possam fazer o mesmo serviço consumindo menos.

Segundo Fortes *et al.* (2017) é possível adotar ações para melhorar a eficiência no uso da energia elétrica através do uso de *smart meters*, no qual suas funções combinadas de coleta

de dados e comunicação, permitem monitorar a qualidade e quantidade da energia entregue ao consumidor.

1.3 SMART METER

A definição de medidor inteligente, do inglês *Smart Meter*, pode ser dada como um dispositivo de medição capaz de lidar com diferentes sistemas, tais como eletrônicos, elétricos, mecânicos, entretanto, aprimorado com sistemas eletrônicos e recursos digitais de forma a torná-lo uma ferramenta flexível e interativa, sempre fazendo uso de uma interface digital (ANGLANI *et al.*, 2011). Para Arif *et al.* (2013), *smart meters* são dispositivos de medição com capacidade muito grande em relação à quantidade de dados coletados e disponibilizados aos usuários, permitindo aos consumidores manterem-se bem informados sobre o seu consumo de energia, para que possam tomar melhores decisões quando estiverem usando a energia.

Segundo Romancini *et al.* (2022) há uma possibilidade de desenvolver equipamentos de medições integrados à rede que forneçam aos consumidores dados relevantes, informativos e transparentes para um maior controle sobre a utilização da energia elétrica em seus estabelecimentos e, com iniciativas próprias, administrar o uso de maneira econômica, logo, eficiente.

1.4 TRANSDUTORES DE PARÂMETROS ELÉTRICOS

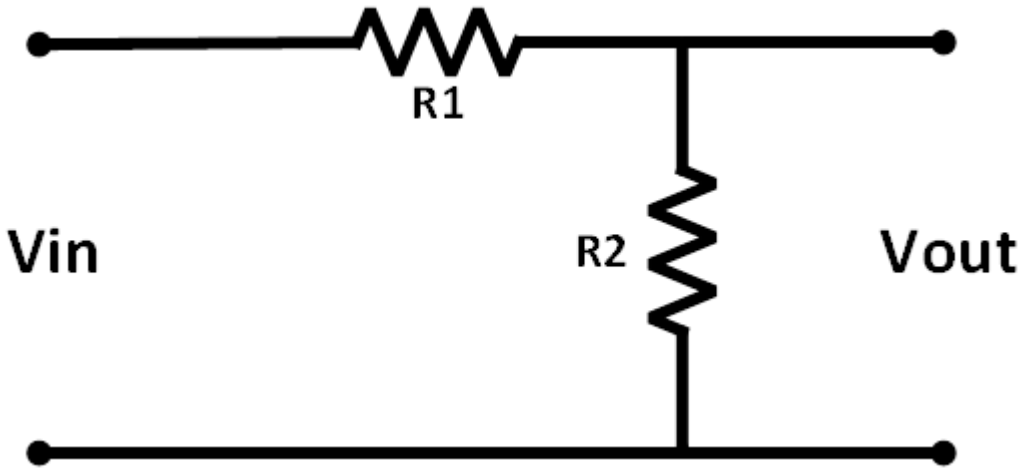
Para a aquisição de parâmetros elétricos tais como tensão e corrente, é necessário o condicionamento dos sinais elétricos para a possibilidade de processamento dos sinais por um dispositivo eletrônico. Para tanto, é necessário a utilização de transdutores, que segundo Fernandes (2022) são dispositivos que convertem parâmetros de qualquer natureza em sinais elétricos, de forma a amostrar e quantizar sinais para posterior análise e processamento. Dentre os transdutores, destacam-se os de tensão e de corrente, utilizados para análise de parâmetros elétricos.

1.4.1 Transdutor de Tensão

Para a aquisição de sinais de tensão, seu condicionamento deve ser feito respeitando as exigências impostas pelo dispositivo de medição empregado. O método mais simples e eficiente de se condicionar sinais de tensão, mantendo características do sinal é com o uso de resistores,

e com o uso do circuito de divisor de tensão resistivo, no qual se aplicam dois resistores em série, medindo a tensão aplicada sobre um dos resistores (LOCCI; MUSCAS; SULIS, 2009), como ilustra a Figura 1.

Figura 1 – Circuito divisor de tensão resistivo.



Fonte: Autoria Própria.

Para a obtenção da tensão em Vout, a Equação 1 é aplicada.

$$Vout = \frac{R2}{R1 + R2} * Vin \qquad \text{(Equação 1)}$$

Onde Vin é a tensão de entrada do circuito, R1 e R2 são resistores utilizados como divisores de tensão do circuito, e Vout é a queda de tensão no resistor R2, utilizada como tensão de saída do circuito.

Para a aplicação em circuitos de medição, a entrada (Vin) será a tensão da rede elétrica e a saída (Vout) irá para o dispositivo de medição.

1.4.2 Transdutor de Corrente Elétrica

Para a aquisição do sinal de corrente elétrica o condicionamento deve ser realizado, e para isso, métodos semelhantes ao do transdutor de tensão podem ser aplicados. Transdutores do tipo Transformadores de Corrente (TC) são os mais práticos de serem utilizados e seu funcionamento é simplificado. Para Oliveira (2001), o transformador de corrente ideal possui a

relação ampères-espira (excitação) do primário exatamente igual à magnitude da relação ampères-espira do secundário, que pode ser expresso como mostra a Equação 2.

$$N_P I_P = N_S I_S \quad \text{(Equação 2)}$$

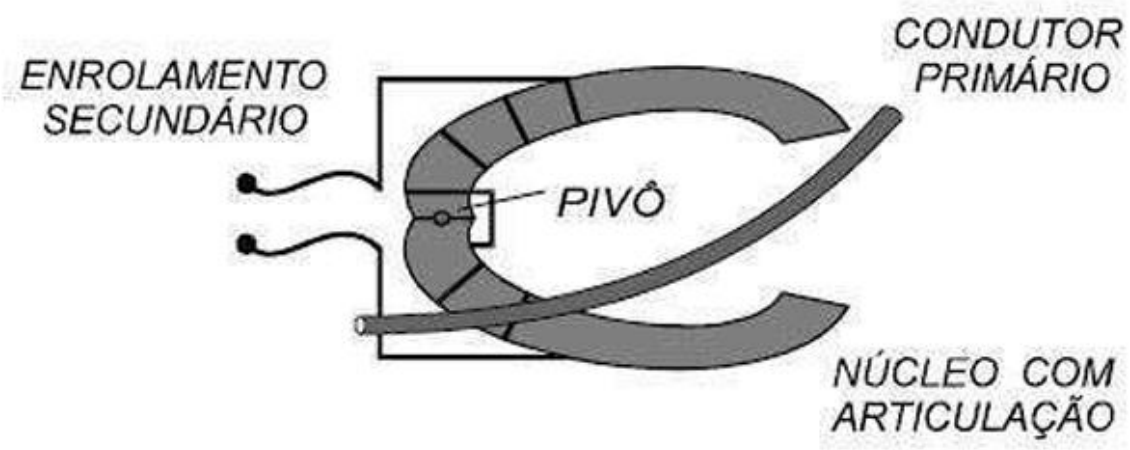
Consequentemente, a corrente no secundário será expressa como mostra a Equação 3.

$$I_S = I_P * \frac{N_P}{N_S} \quad \text{(Equação 3)}$$

Onde I_P e I_S são as correntes no primário e secundário, respectivamente, e N_P e N_S são o número de voltas nas espiras primária e secundária, respectivamente.

Para este trabalho, será utilizado o transformador de corrente do tipo Núcleo Dividido, pois segundo Oliveira (2001), o Transformador de corrente do tipo Núcleo Dividido é constituído por uma parte do núcleo que é separável ou basculante, para facilitar o enlaçamento do condutor primário. O amperímetro tipo "alicate" nada mais é do que um TC de núcleo dividido, o qual possibilita medir a corrente sem a necessidade de abrir o circuito para colocá-lo em série, como ilustra a Figura 2.

Figura 2 – Funcionamento de um TC de núcleo dividido.

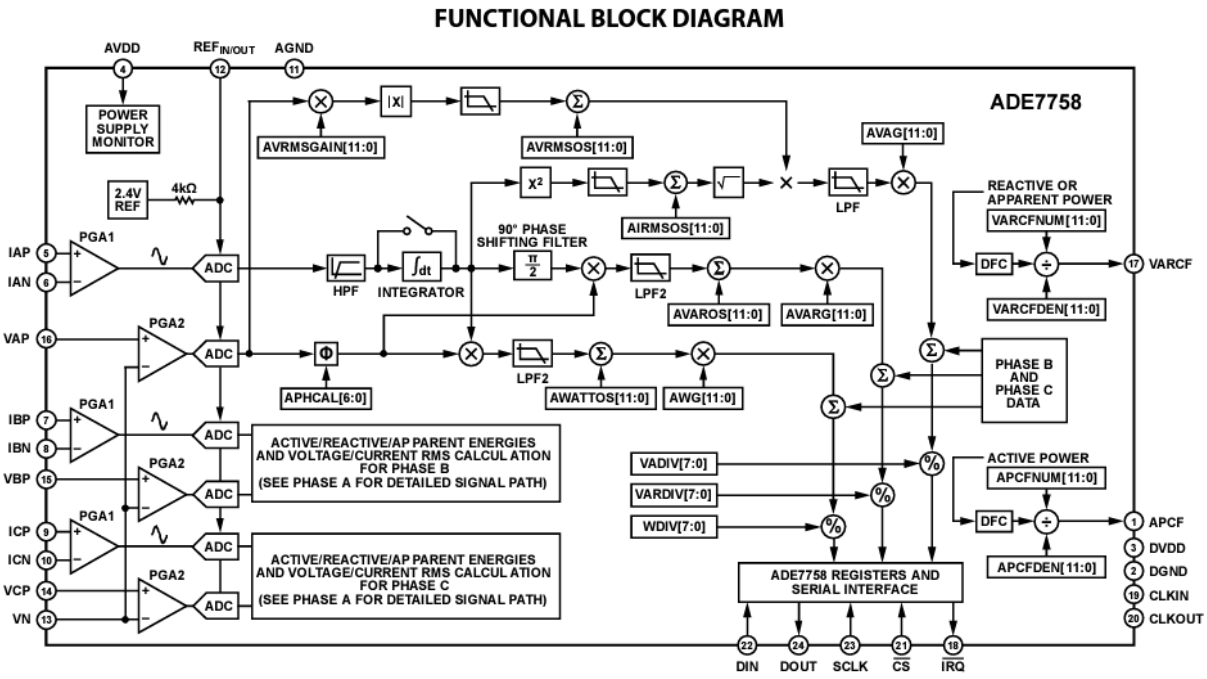


Fonte: (SOARES, 2015, p. 19).

1.5 ADE7758

O ADE7758 é um Circuito Integrado (CI) medidor de energia elétrica da Analog Devices que utiliza processadores digitais de sinais capazes de realizar medições trifásicas de tensão RMS, corrente RMS e energias ativa, reativa e aparente, assim como realiza compensação de fase, ganho e *offset*, através de métodos de calibração e comparação de parâmetros elétricos. Tais medições são feitas através de conversores analógicos-digitais, do inglês *Analog-to-Digital Converter* (ADC), e são processados através do ADE7758, no qual disponibiliza valores medidos e informações por cada fase medida através de registradores específicos, que são acessíveis através de comunicação via interface serial SPI (ANALOG DEVICES, 2011). O diagrama funcional em blocos do ADE7758 exemplifica o uso de suas funcionalidades, de acordo com a Figura 3.

Figura 3 – Diagrama em Blocos do ADE7758.



Fonte: (ANALOG DEVICES, 2011, p. 1).

1.6 MICROCONTROLADORES

Um microcontrolador é considerado um microcomputador que contém processador, memória e periféricos de entrada e saída em um único *chip*, de tal maneira que se resuma a uma forma genérica da integração entre uma Unidade de Processamento Central, do inglês *Central*

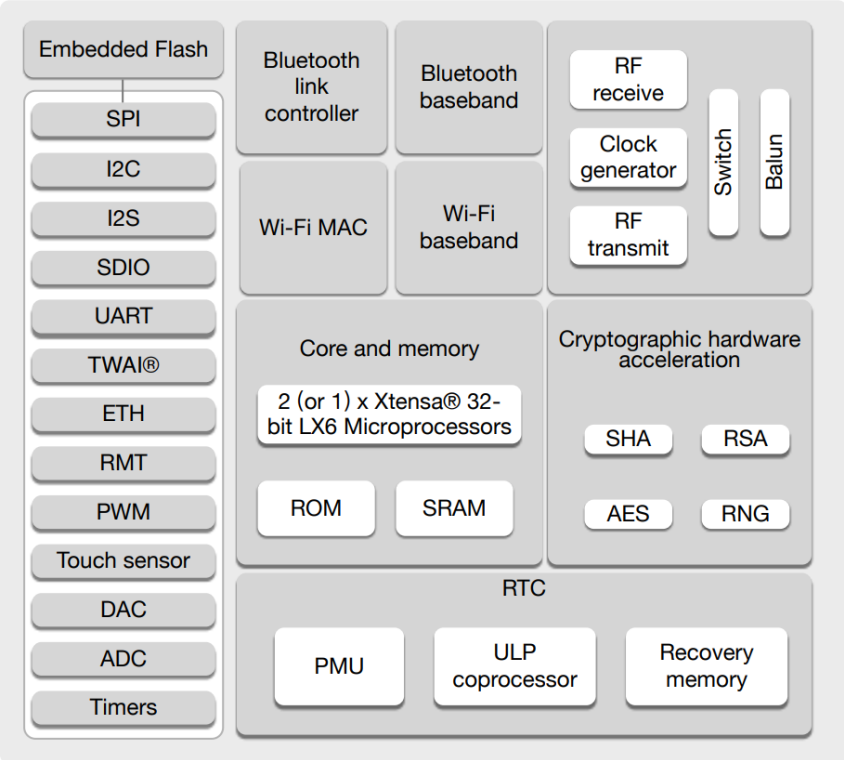
Processing Unit (CPU), e os periféricos que fazem a interface do microcontrolador com o mundo externo (OLIVEIRA JÚNIOR; DUARTE, 2010).

1.6.1 Módulo ESP32

O módulo ESP32 é um microcontrolador Sistema em um Chip, do inglês *System on a Chip* (SoC) capaz de realizar múltiplas tarefas e com poder de processamento bastante elevado devido ao uso de dois núcleos de processamento de 32-bit. Devido à utilização já embarcada de dois módulos de comunicação bastante utilizados, que são os módulos de Wi-Fi e o Bluetooth, a escolha deste microcontrolador para aplicações IoT se tornou extensa, pois facilita a integração com diferentes dispositivos. As variadas interfaces periféricas do ESP32, tais como, *Serial Peripheral Interface* (SPI), *Inter Integrated Circuit* (I2C), *Inter-IC Sound* (I2S) e *Universal Asynchronous Receiver Transmitter* (UART), possibilitam a integração com diferentes sensores e módulos (ESPRESSIF, 2021).

O diagrama em blocos do ESP32 possibilita a visualização de seus periféricos e suas tecnologias de forma a facilitar e ilustrar a utilização e disposição dos mesmos, conforme ilustrado na Figura 4.

Figura 4 – Diagrama em Blocos do ESP32.



Fonte: (ESPRESSIF, 2021, p. 12).

1.7 INTERNET OF THINGS – IOT

Internet das coisas, do inglês *Internet of Things*, é uma rede de interconexão de dispositivos com acesso a rede local ou global de Internet. Através dessa conexão, pode haver uma troca de informações utilizando a estrutura da conexão para que haja, por exemplo, uma coleta de dados de diferentes sensores de forma remota, possibilitando o monitoramento das "coisas" por qualquer pessoa em qualquer lugar (STEVAN JUNIOR, 2018).

1.8 TCP/IP

O TCP/IP é um modelo de conjunto de camadas de comunicação nos quais oferecem recursos necessários para a conectividade. Esse modelo é dividido em 4 pilhas de protocolos, conhecidos como camadas, sendo eles: Camada de aplicação, Camada de transporte, Camada de rede e Camada de Enlace-Física (OLIVEIRA, 2017). Na camada de aplicação é por onde programas de rede se situam, como por exemplo, o protocolo *Message Queuing Telemetry Transport* (MQTT), e na camada de enlace-física é por onde se faz a conexão física com a rede, como por exemplo o Wi-Fi, conforme ilustra Figura 5.

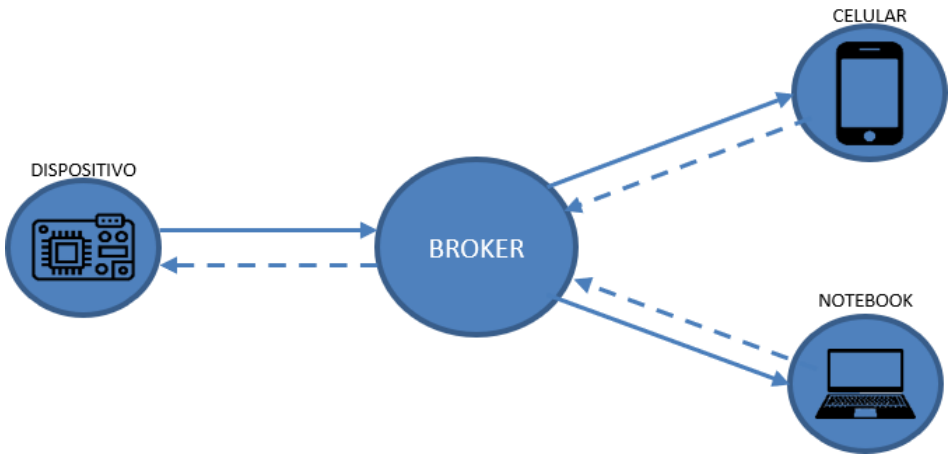
Figura 5 – Camadas do modelo TCP/IP.

Fonte: Autoria Própria.

1.9 PROTOCOLO MQTT

O MQTT é um protocolo simples e de baixa complexidade de transporte de mensagens que utiliza o modelo publicador/subscritor, do inglês *publisher/subscriber*, através de um gerenciador denominado corretor, ou *broker*, de forma que um produtor de dados publique uma mensagem para um destino chamado de tópico, e caso haja algum cliente inscrito nesse tópico, este irá receber a mensagem enviada (MESNIL, 2014). O protocolo MQTT utiliza o protocolo TCP para transportar dados. Por ser um protocolo de reduzida complexidade, implicando em menor utilização de recursos computacionais, o MQTT é bastante utilizado em aplicações embarcadas utilizando microcontroladores para fazer a interface entre sensores e a rede de Internet, ou seja, a aplicação IoT se torna simplificada ao fazer o uso deste protocolo de transporte de mensagens, sua aplicação é exemplificada na Figura 6.

Figura 6 – Arquitetura do protocolo MQTT.



Fonte: Autoria Própria.

1.10 BASE DE DADOS SQLITE

Segundo Allen e Owens (2010), o SQLite é uma base de dados relacional embarcada de código aberto (*open source*) lançada em 2000 para providenciar o gerenciamento de dados para aplicações de maneira conveniente, considerando sua alta portabilidade, facilidade de uso, eficiência, e sem a complexidade de sistemas que utilizam bases de dados relacionais dedicadas. O uso do banco de dados possibilita o armazenamento de dados para análises futuras. O SQLite é o banco de dados mais amplamente implantado do mundo, sendo encontrado principalmente em *smartphones* (SQLITE, 2021).

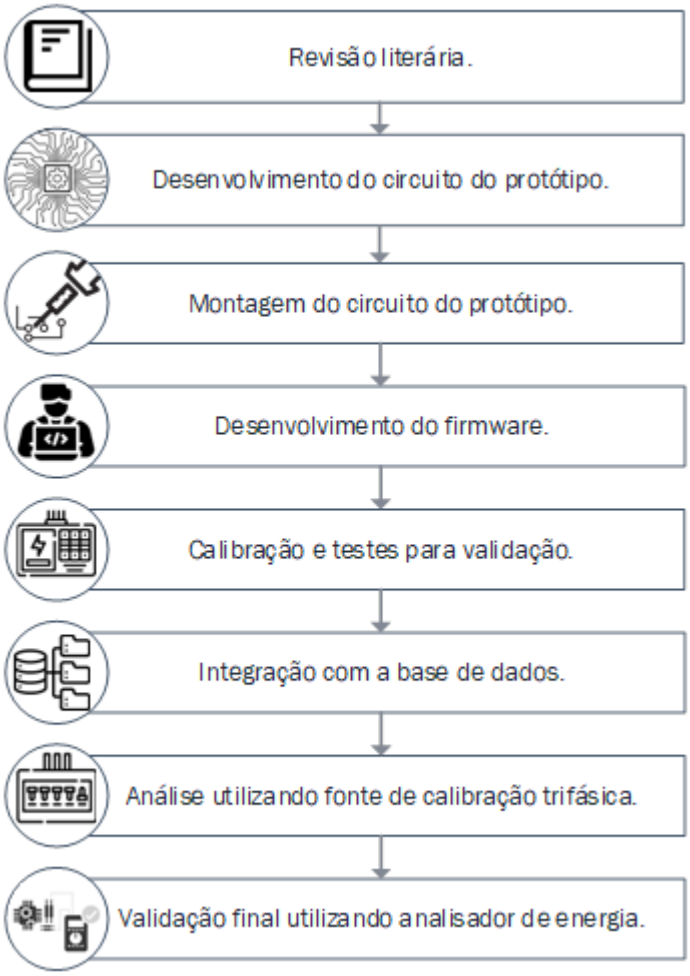
2 MATERIAIS E MÉTODOS

Neste capítulo será apresentado a sequência dos métodos utilizados para o desenvolvimento deste trabalho, contendo as etapas da implementação do protótipo de sistema. Os materiais necessários para a implementação do protótipo serão apresentados em seguida.

2.1 MÉTODO PROPOSTO

As etapas do método proposto são compostas por: revisão literária, desenvolvimento do circuito do protótipo, montagem do circuito do protótipo, desenvolvimento do *firmware*, calibração de medições e testes de validação, integração com a base de dados, análise utilizando fonte de calibração trifásica e, por fim, validação final utilizando analisador de energia. Estas etapas estão ilustradas na Figura 7 com o uso de um diagrama em blocos.

Figura 7 – Diagrama das etapas de implementação do projeto.



Fonte: Autoria Própria.

Como primeira etapa, foi realizada uma revisão de literatura nas áreas de microcontroladores e microprocessadores, análise de circuitos elétricos, conversores analógicos-digitais, eletricidade aplicada, as linguagens de programação C e Python, *smart meters*, Internet das Coisas, MQTT e Base de Dados. A revisão realizada na área de microcontroladores e microprocessadores possibilitou a escolha do microcontrolador a ser utilizado no protótipo. O estudo na área de conversores analógicos-digitais proporcionou a determinação do CI ADE7758 para a aquisição das medições de tensão e corrente.

Como segunda etapa, o desenvolvimento do circuito do protótipo consistiu primeiramente no desenvolvimento do esquemático do circuito utilizando o *software Altium Designer 21*. Os estudos realizados na área de eletricidade aplicada e análise de circuitos elétricos instruíram a elaboração de circuitos condicionadores de sinais para a realização da medição, bem como o circuito de alimentação dos principais módulos. Foram desenvolvidos circuitos de proteção contra surtos a fim de garantir a integridade do protótipo. Com a finalização do esquemático, o layout da placa de circuito impresso foi desenvolvido a fim de definir o posicionamento dos componentes na placa.

A terceira etapa consistiu na realização da montagem do circuito do protótipo no Laboratório de Sistemas Embarcados (LSE) localizado no Hub Tecnologia e Inovação, utilizando a placa de circuito impresso desenvolvida. Testes unitários foram realizados para a validação parcial dos componentes do circuito, verificando suas funcionalidades isoladas. Testes de integração entre circuitos foram realizadas para a verificação de funcionalidades entre circuitos.

Na quarta etapa, foi desenvolvido o *firmware* utilizando o *software* Visual Studio Code para programação em linguagem C integrado com a extensão da própria fabricante do microcontrolador para auxílio na programação a ser embarcada no módulo. O *framework* ESP-IDF foi utilizado para a compilação e o embarque do *firmware* no módulo ESP32 contido na placa desenvolvida.

Na quinta etapa, a realização da calibração do protótipo utilizando uma fonte trifásica garantiu a precisão nas medições, eliminando erros de leitura e correção de fatores essenciais para o correto funcionamento do protótipo. Para a praticidade na calibração do protótipo, foi desenvolvido um *script* na linguagem de programação Python contendo interface visual para melhor usabilidade na operação do processo de calibração. Testes foram feitos para validação em laboratório, de tal forma que a utilização em campo possa ser feita de maneira segura. Estes testes consistiram na utilização do protótipo para a realização de medições isoladas, garantindo a normalidade no funcionamento esperado do protótipo.

Na sexta etapa foi realizada a integração entre os dados transmitidos e a base de dados SQLite utilizando *script* desenvolvido utilizando a linguagem de programação Python. O desenvolvimento do *script* em Python para gerenciamento dos dados contidos na base de dados garante o gerenciamento das medições e eventos ocorridos no protótipo. A base de dados SQLite permite, através de um navegador de dados, a visualização e geração de gráficos dos dados contidos na mesma.

Na sétima etapa foi realizada a análise utilizando uma fonte de calibração trifásica para obtenção dos parâmetros elétricos, e simulações de interrupções no fornecimento de energia elétrica, no qual será instalado em conjunto com o analisador de qualidade de energia modelo FLUKE 434 série II.

Por fim, foi realizada a validação final das medições realizadas pelo protótipo, comparando os dados obtidos através do protótipo com os dados obtidos pelo analisador de qualidade de energia modelo FLUKE 434 série II.

2.2 MATERIAIS UTILIZADOS

Para a realização de testes com o protótipo e validação de seu funcionamento, foram utilizados os seguintes itens:

- Fonte trifásica PTS400.3 MTE;
- Analisador de qualidade de energia FLUKE 434 série II;
- Fonte de bancada HIKARI HF-3205S;
- Fonte 5V.

Para a implementação do projeto, foram utilizadas as seguintes ferramentas:

- Interface de desenvolvimento Visual Studio Code versão 1.71.2 com extensão Espressif IDF versão 1.5.0;
- Altium Designer 21;
- PyCharm 2022.1;
- DB Browser para SQLite versão 3.12.2;
- Power Log 430-II versão 5.9;
- Cabos para conexão.

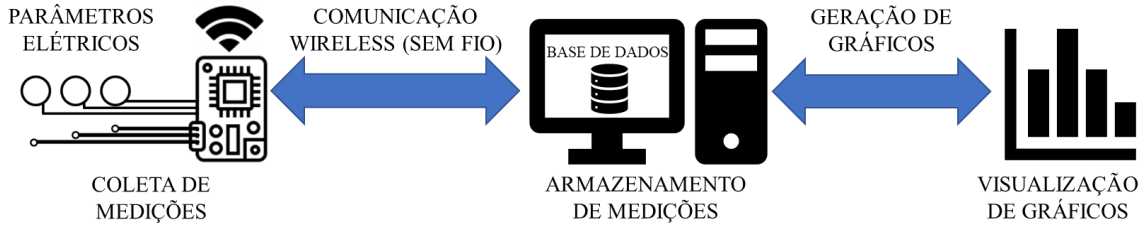
Para a construção do protótipo de *hardware*, foram utilizados os seguintes componentes:

- 1 Módulo ESP-WROOM-32D;
- 1 circuito integrado ADE7758;
- 1 circuito integrado SI8663;
- Circuito de proteção;
- Circuito de condicionamento de tensão;
- Circuito de condicionamento de corrente;
- Circuito de alimentação.

3 IMPLEMENTAÇÃO DO PROJETO

Este capítulo apresenta os procedimentos detalhados para o desenvolvimento do protótipo de sistema, composto por um protótipo de *hardware* capaz de fazer aquisição de parâmetros elétricos. Os dados coletados serão adquiridos pelo ADE7758, processados por um microcontrolador ESP32, e publicados em um servidor MQTT. O armazenamento dos dados será realizado por um *script* em Python integrado a uma base de dados SQLite. A visualização dos parâmetros de energia será feita em um navegador de base de dados DB Browser para SQLite. A arquitetura geral do protótipo é demonstrada na Figura 8.

Figura 8 – Arquitetura geral do protótipo de sistema.



Fonte: Autoria Própria.

Neste capítulo serão apresentados os seguintes tópicos:

a) preparação do ambiente de desenvolvimento de *firmware*;
b) desenvolvimento do protótipo de *hardware*;
c) desenvolvimento do protótipo de *firmware*;
d) desenvolvimento do *script* para armazenamento de dados;
e) calibração do protótipo de *hardware* e testes de validação;
f) integração com a base de dados; e
g) validação do protótipo de sistema.

3.1 PREPARAÇÃO DO AMBIENTE DE DESENVOLVIMENTO DE *FIRMWARE*

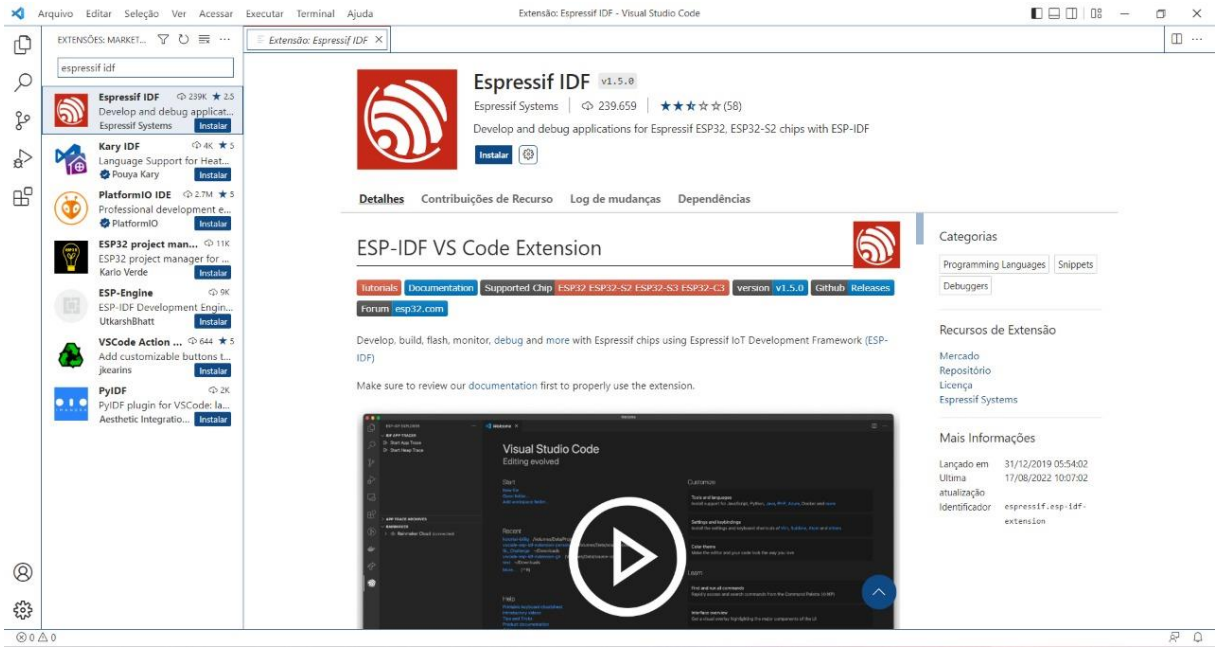
Para o desenvolvimento do *firmware* do sistema foi escolhido a utilização do Framework ESP-IDF como estrutura de desenvolvimento. O ESP-IDF é a estrutura de desenvolvimento oficial do microcontrolador ESP32, portanto, a escolha do mesmo se deu pela recomendação da fabricante em usá-lo. Como editor de código, foi escolhido o Visual Studio Code, que permite a inclusão da extensão Espressif IDF, utilizado para compilar o código e

embarcar no microcontrolador. Para a implementação do *firmware* foi necessário a configuração do ambiente de desenvolvimento, iniciando pela instalação da extensão Espressif IDF no Visual Studio Code. As etapas para instalação, foram:

a) Instalação da extensão Espressif IDF;
b) Configuração do Framework; e
c) Criação de um novo projeto.

Para a instalação da extensão é necessário clicar na aba “Extensões” do Visual Studio Code e pesquisar pela extensão “Espressif IDF”, conforme mostrado na Figura 9. Após isso, deve-se então instalar a extensão.

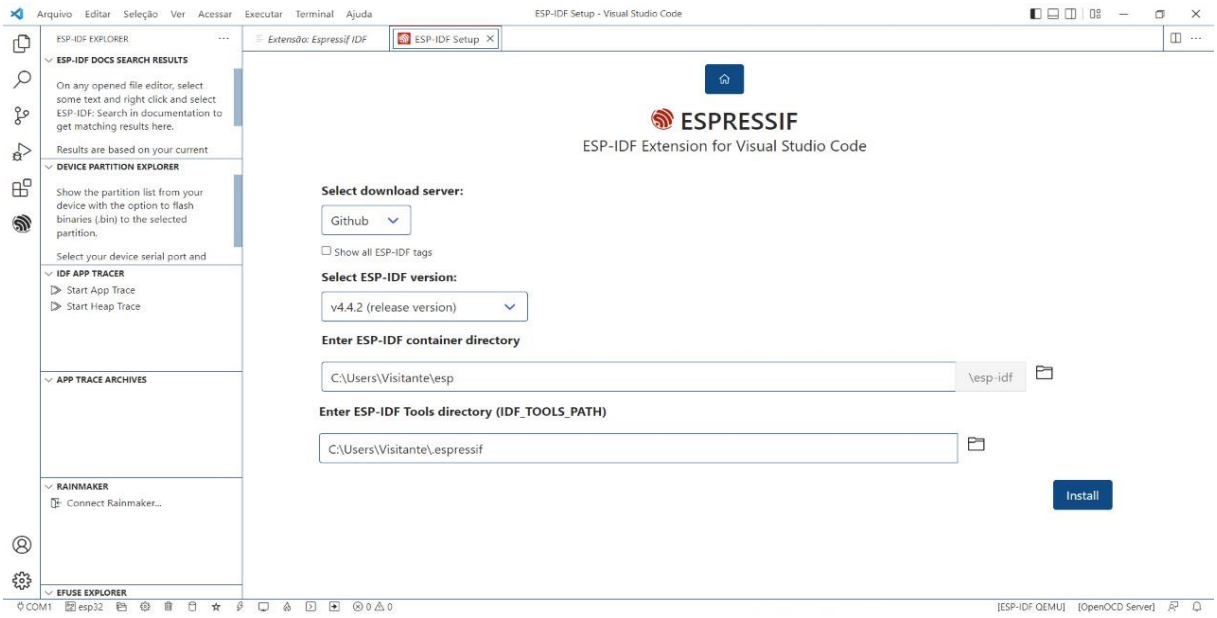
Figura 9 – Instalação da extensão Espressif IDF.



Fonte: Autoria Própria.

Após a instalação da extensão, deve-se adicionar as configurações necessárias para a codificação e compilação do ESP32. Para configurar, deve-se então inserir os dados do ESP32 e selecionar a versão do ESP-IDF que deseja utilizar, conforme mostra a Figura 10.

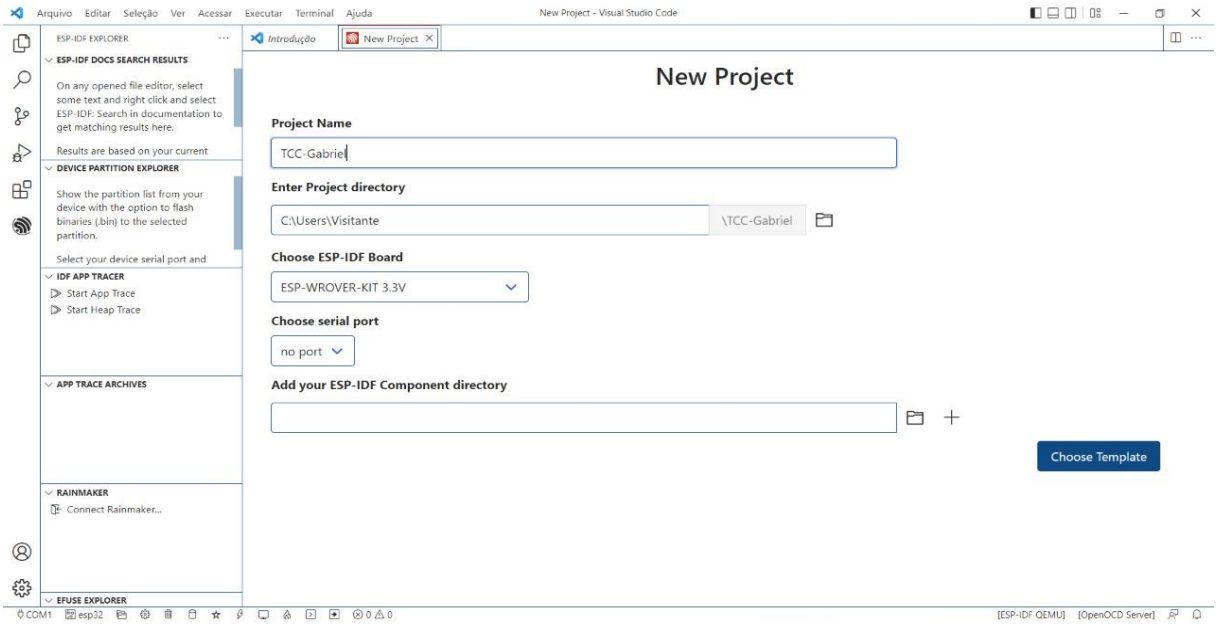
Figura 10 – Configuração da extensão Espressif IDF.



Fonte: Autoria Própria.

Com a extensão Espressif IDF já instalada e configurada, deve-se então criar um novo projeto para a finalização da preparação do ambiente de desenvolvimento do *firmware*. Com esse propósito, é necessário configurar o nome do projeto desejado, o local no qual ficará hospedado, a placa que será utilizada, a porta serial que está conectada, e se necessário, os componentes que serão incluídos, conforme Figura 11.

Figura 11 – Configuração na criação de um novo projeto.



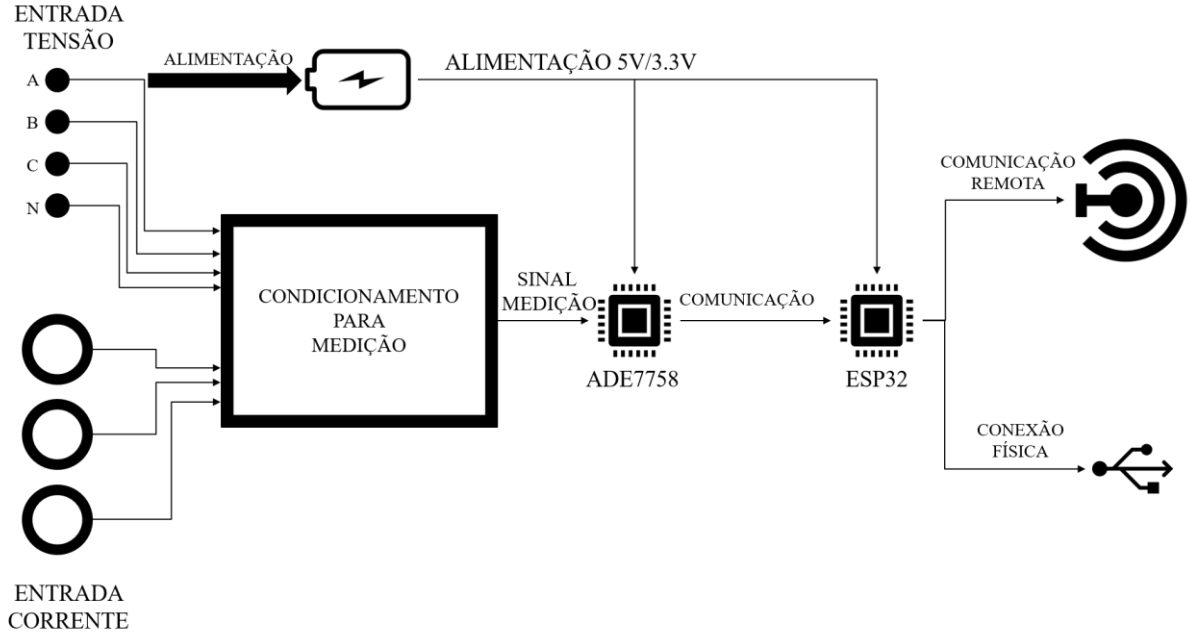
Fonte: Autoria Própria.

Com as configurações finalizadas, o ambiente de desenvolvimento encontra-se preparado para o desenvolvimento do *firmware*.

3.2 DESENVOLVIMENTO DO PROTÓTIPO DE *HARDWARE*

Para o desenvolvimento do protótipo de *hardware* do sistema, foi necessário a projeção da arquitetura desejada do protótipo para uma melhor visualização e para definição de estratégias de desenvolvimento. Para a elaboração da arquitetura, foi realizado o levantamento dos materiais a serem utilizados no protótipo, identificados por cada funcionalidade, sendo eles: ESP32 (utilizado como unidade de processamento), ADE7758 (circuito integrado medidor de energia elétrica), entradas de tensão e corrente (entrada dos parâmetros a serem medidos), circuito de proteção, circuitos de condicionamento e circuito de alimentação. Na Figura 12 é ilustrada a arquitetura do protótipo de *hardware*.

Figura 12 – Arquitetura do protótipo de *hardware.*



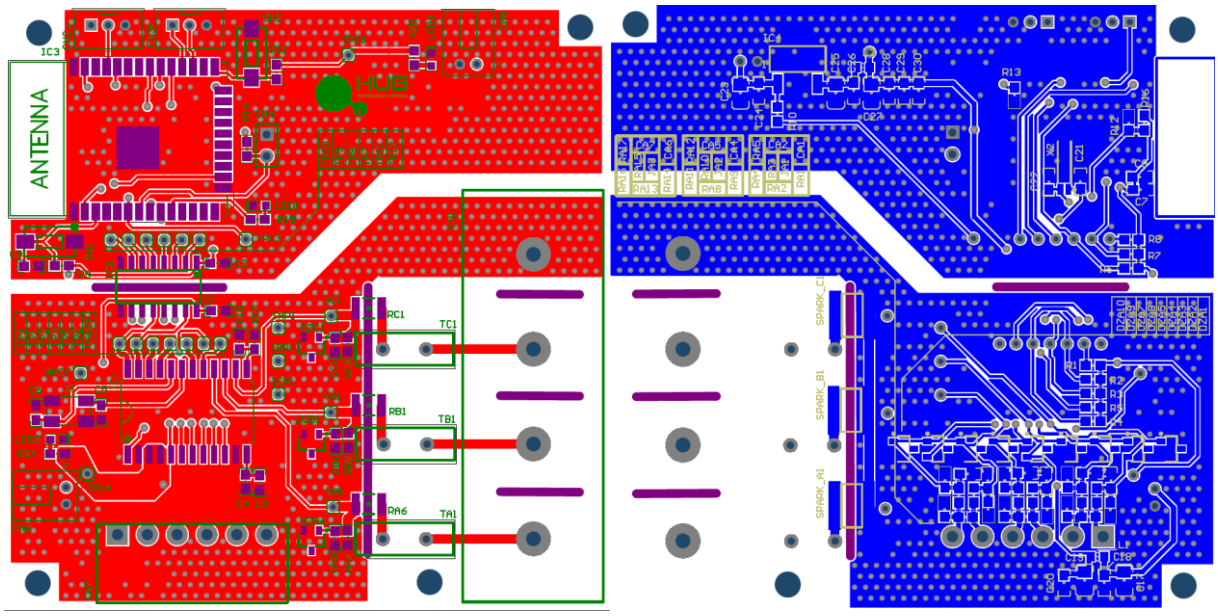
Fonte: Autoria Própria.

Com a arquitetura do protótipo de *hardware* finalizada, realizou-se o desenvolvimento do esquemático do protótipo e seu respectivo *layout*, responsáveis pela definição dos circuitos em si do protótipo de *hardware*, e para isso, foi utilizado o *software* Altium Designer 21 a fim de gerar o desenho da Placa de Circuito Impresso (PCI, do inglês *Printed Circuit Board – PCB)*. Os circuitos de condicionamento de tensão e de corrente elétrica foram dimensionados de

acordo com o exigido na folha de especificação do fabricante do circuito integrado medidor de energia elétrica ADE7758, no qual estabelece uma entrada diferencial com tensão de no máximo 1V pico a pico (1 Vpp) (ANALOG DEVICES, 2011). Para a segurança do protótipo, as entradas dos parâmetros elétricos de tensão e corrente elétrica contam com circuitos de proteção, no qual foram dimensionados para atuarem em caso de surtos, fechando o circuito de entrada de tensão, já que a entrada de tensão é realizada de maneira não isolada da rede elétrica. O circuito de alimentação do protótipo foi dimensionado para que o ADE7758 receba 5V e o ESP32 3.3V, de acordo com o especificado em suas respectivas folhas de especificação. Para que a comunicação entre o ESP32 e o ADE7758, feita através da interface serial SPI, possa ser realizada sem ruídos e mesmo para a isolação das interfaces analógicas e digitais dos circuitos, foi utilizado o CI isolador digital SI8663, que oferece vantagens substanciais de taxa de dados, baixo atraso de propagação, potência, tamanho, confiabilidade (SKYWORKS, 2022).

Com os circuitos devidamente dimensionados e o esquemático do protótipo desenvolvido, a etapa de *layout* do protótipo é iniciada, no qual serão realizadas distribuições dos componentes por todo o espaço da placa, que quando devidamente localizadas, deve ser realizado então o roteamento para os circuitos, que consiste na distribuição de trilhas, conectando os diversos pontos da placa. Como próxima etapa do roteamento, têm-se a definição da malha de terra de toda a placa, responsável pela criação de um ponto de referência para os circuitos da placa. O *layout* final da placa é ilustrado nas Figuras 13a e 13b.

Figura 13 – *Layout* do protótipo de *hardware*, visualização superior (a) e inferior (b).

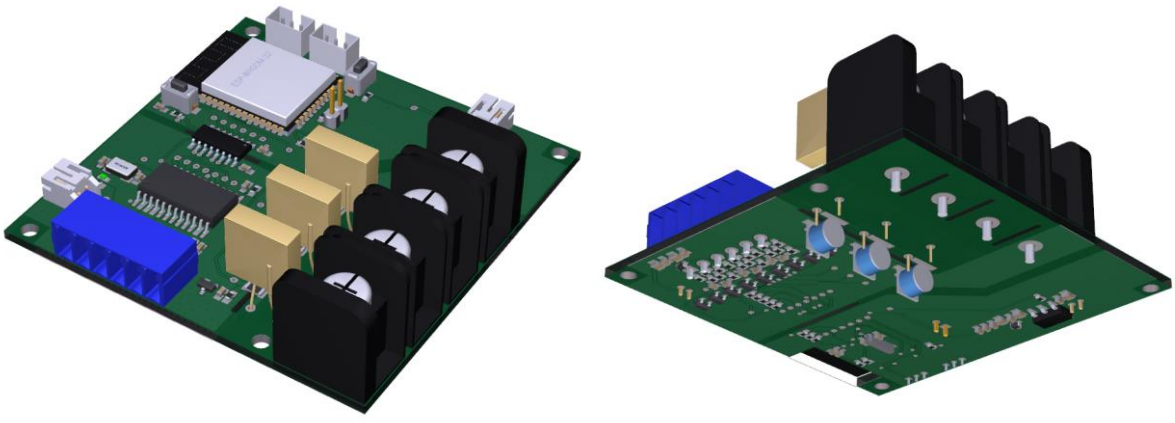


(a) (b)

Fonte: Autoria Própria.

Com o *layout* da placa definido, o *software* Altium Designer 21 proporciona uma visualização da prévia da placa, permitindo sua visualização em 3D, como mostram as Figuras 14a e 14b.

Figura 14 – Visualização em 3D do protótipo de *hardware*, superior (a) e inferior (b).



(a) (b)

Fonte: Autoria Própria.

Com o *layout* da placa finalizado, o próximo passo é a fabricação e montagem do protótipo de *hardware*, no qual foram realizados no Laboratório de Sistemas Embarcados (LSE) localizado no Hub Tecnologia e Inovação, na Escola Superior de Tecnologia (EST) da

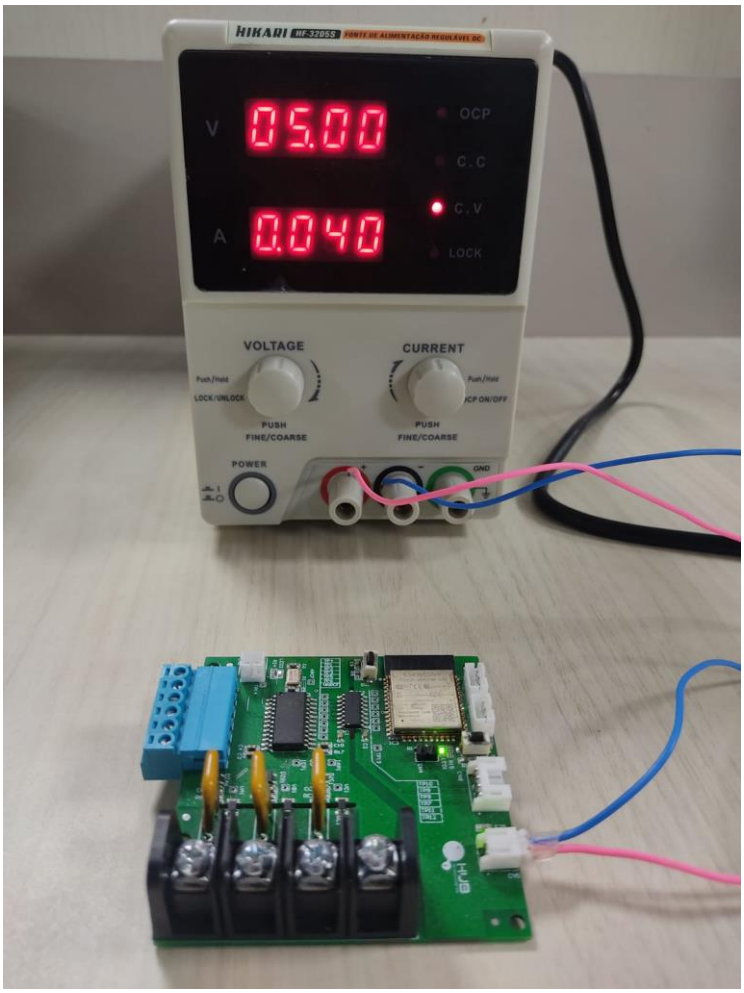
Universidade do Estado do Amazonas (UEA). Para a montagem, foi necessário o uso da Placa de Circuito Impresso desenvolvida, e dos componentes citados no decorrer deste trabalho. A montagem do protótipo foi realizada em bancada com ajuda de colaboradores do laboratório. A Figura 15 demonstra o protótipo de *hardware* finalizado e montado.

Figura 15 – Protótipo de *hardware* finalizado.

Fonte: Autoria Própria.

Para a validação da placa, testes unitários foram realizados para validações parciais da placa, e gradativamente testes de integração entre circuitos foram realizados. Na Figura 16 é demonstrado o teste final de validação dos componentes da placa, realizado em bancada, para revisão do funcionamento normal dos circuitos, utilizando uma fonte de bancada modelo HIKARI HF-3205S e monitorando a corrente para detectar possíveis anomalias no consumo.

Figura 16 – Realização de testes em bancada no protótipo de *hardware*.



Fonte: Autoria Própria.

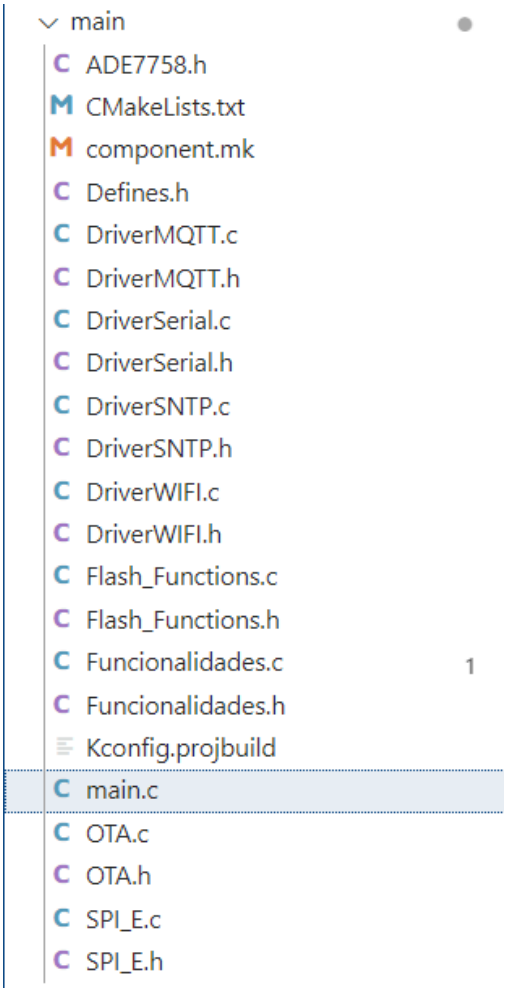
3.3 DESENVOLVIMENTO DO PROTÓTIPO DE *FIRMWARE*

Para o desenvolvimento do protótipo de *firmware* foi utilizado o *software* Visual Studio Code com a extensão Espressif IDF, utilizando linguagem de programação C. Para o processamento e envio dos parâmetros elétricos coletados pelo dispositivo de medição, é necessário implementar um sistema operacional em tempo real, para uma melhor coordenação entre as tarefas atribuídas à unidade de processamento (ESP32), para tanto, é utilizado o Sistema Operacional em Tempo Real "FreeRTOS", nativo do microcontrolador ESP32.

O início da implementação do *firmware* iniciou-se pelo desenvolvimento das bibliotecas de funcionalidades do dispositivo. Para cada funcionalidade atribuída ao dispositivo é gerada uma biblioteca responsável pelo gerenciamento e tratamento de cada funcionalidade especificada e para isso, arquivos fonte e arquivos de cabeçalho (do inglês *header file*) são gerados.

A estrutura dos arquivos gerados é demonstrada na Figura 17, onde cada *driver* representa sua respectiva função.

Figura 17 – Estrutura do *firmware* dividido em *drivers.*



Fonte: Autoria Própria.

Dentre os arquivos fonte e arquivos de cabeçalho há arquivos de configuração do compilador e do *framework* ESP-IDF, nos quais são os arquivos “CMakeLists.txt”, “component.mk” e “Kconfig.projbuild”, estes arquivos são gerados automaticamente na criação de um novo projeto utilizando a extensão Espressif IDF no Visual Studio Code.

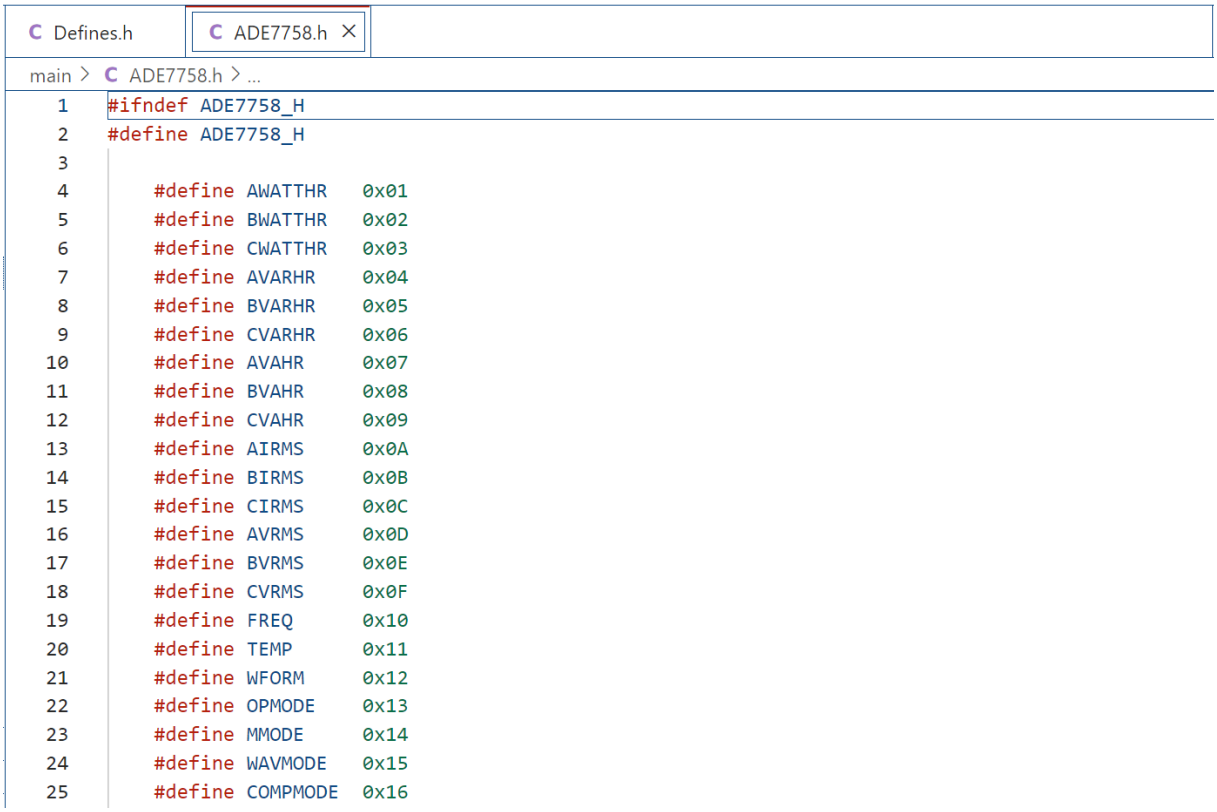
Os arquivos fonte e arquivos de cabeçalho (também denominados de bibliotecas) responsáveis pelas funcionalidades do sistema, são:

- Flash_Functions.c e Flash_Functions.h
- SPI_E.c e SPI_E.h
- DriverSerial.c e DriverSerial.h
- DriverMQTT.c e DriverMQTT.h

- DriverWIFI.c e DriverWIFI.h
- DriverSNTP.c e DriverSNTP.h
- Funcionalidades.c e Funcionalidades.h
- OTA.c e OTA.h

Os arquivos de cabeçalho “ADE7758.h” e “Defines.h” são arquivos responsáveis pela criação de definições e macros com identificadores do sistema, como mostra as Figuras 18 e 19.

Figura 18 – Arquivo de cabeçalho responsável pelo ADE7758.



Fonte: Autoria Própria.

O arquivo “ADE7758.h” é responsável por mapear todos registradores do CI ADE7758 para auxiliar e facilitar os acessos aos registradores, atribuindo um identificador a cada endereço de registrador do CI, seja para escrever ou para consultar o registrador. O mapeamento foi realizado conforme a folha de especificação do CI ADE7758 indica em sua lista de registradores.

Figura 19 – Arquivo de cabeçalho responsável por definições gerais do sistema.

Fonte: Autoria Própria.

O arquivo de cabeçalho “Defines.h” é responsável por mapear as portas de entrada e saída de uso geral, do inglês *General Purpose Input and Output* (GPIO) do ESP32, onde cada porta foi definida previamente no *hardware* e deve ser declarada no *firmware* para que sejam atribuídas suas devidas funções. Foram atribuídos identificadores às portas GPIO de acordo com sua função, facilitando o uso em chamadas de função. Para a criação de configuração inicial de conexão, tanto do Wi-Fi quanto do MQTT, foram criadas macros com definições de conexão padrão.

Para a inicialização do *firmware* é necessário a inicialização das funcionalidades de memória Flash, embarcada no microcontrolador, cujo processo foi desenvolvido na biblioteca de funções “Flash_Functions.c”, onde funções de cadastro e de carregamento da memória foram implementados, como demonstrado na Figura 20. Os protótipos de funções e a inclusão de outras bibliotecas foram implementadas no arquivo de cabeçalho “Flash_Functions.h”.

Figura 20 – Função de inicialização da memória Flash e carregamento dos dados salvos.

Flash_Functions.c

main > Flash_Functions.c > ...

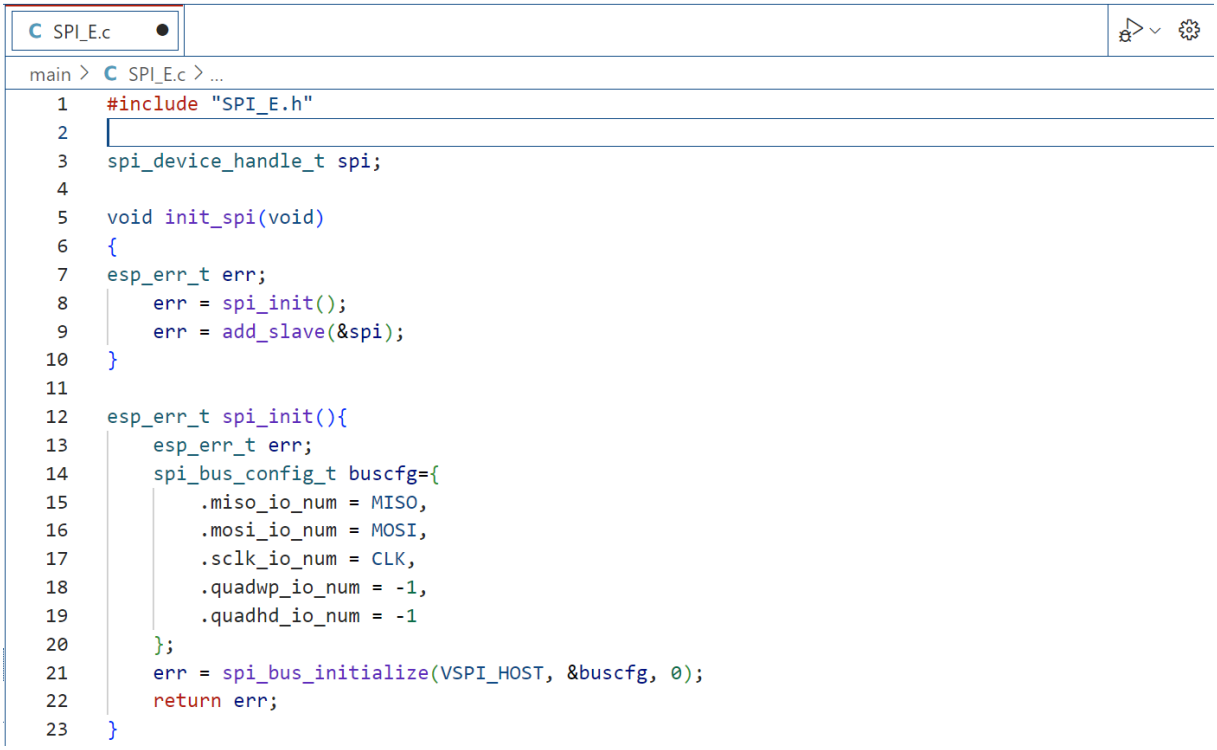
```c
#include "Flash_Functions.h"

cJSON *config_json;
char config_str[1024];

uint8_t init_cadastro(){
    esp_err_t ret = nvs_flash_init();
    if (ret == ESP_ERR_NVS_NO_FREE_PAGES) {
      ESP_ERROR_CHECK(nvs_flash_erase());
      ret = nvs_flash_init();
    }

    ESP_ERROR_CHECK(ret);

    if(ReadFlash("JSON", config_str)){
        config_json = cJSON_Parse(config_str);
        return 1;
    }
```

Fonte: Autoria Própria.

O padrão de formatação de dados implementado foi o formato “Notação de Objeto JavaScript”, do inglês *JavaScript Object Notation* (JSON), e sua implementação no *firmware* foi feita utilizando a biblioteca nativa cJSON, ao fazer a inclusão do arquivo de cabeçalho “cJSON.h” para utilização.

A comunicação com o CI ADE7758 é realizada utilizando a interface serial SPI, e para a utilização desta interface, algumas funcionalidades foram implementadas no *driver* “SPI_E.c”, tais como a inicialização da interface, a inclusão do ADE7758 como escravo, do inglês *slave*, da interface, e a declaração das configurações das portas a serem utilizadas como barramento SPI no ESP32, como mostra a Figura 21. A prototipação das funções e a inclusão de outras bibliotecas foram realizadas no arquivo de cabeçalho “SPI_E.h”.

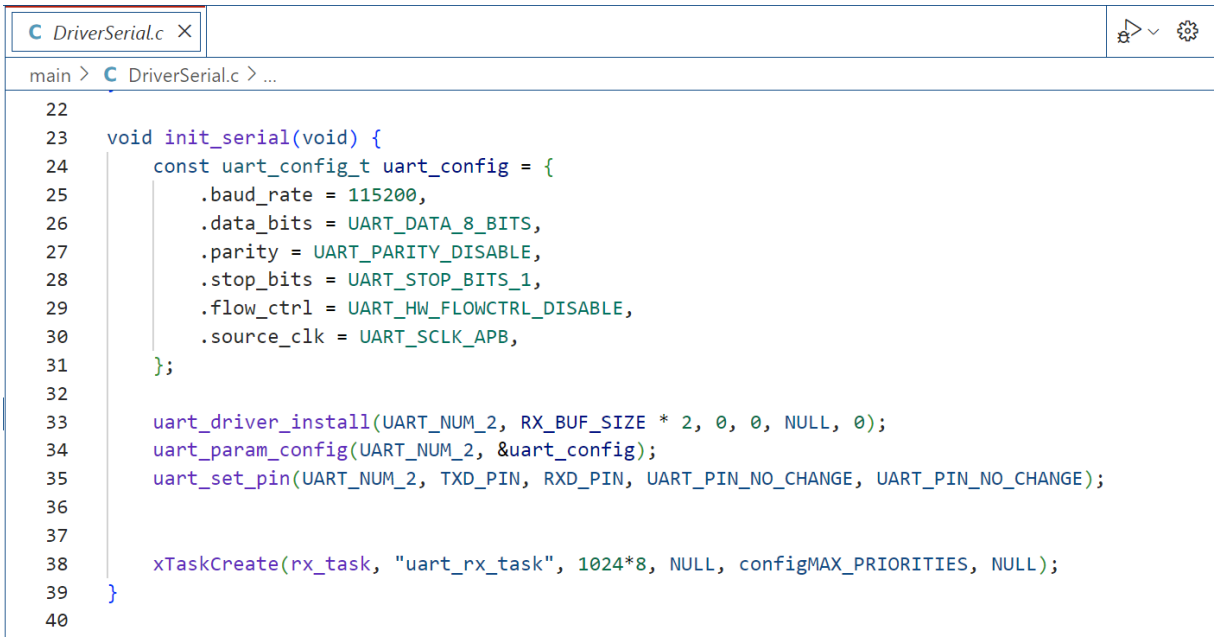
Figura 21 – Inicialização da interface SPI.



Fonte: Autoria Própria.

Como comunicação cabeada com o dispositivo, foi escolhida a comunicação serial UART, logo, suas implementações no *firmware* foram realizadas de acordo com a necessidade. A interface serial UART será utilizada no processo de calibração do protótipo. As implementações das funcionalidades da UART foram realizadas no *driver* “DriverSerial.c”, sendo a inicialização o ponto mais importante da implementação, conforme demonstra a Figura 22. Os protótipos das funções e a inclusão de outras bibliotecas foram realizadas no arquivo de cabeçalho “DriverSerial.h”.

Figura 22 – Inicialização do *driver* da UART.



Fonte: Autoria Própria.

O protocolo MQTT foi o protocolo escolhido como interface de comunicação remota com o *firmware*, funcionando como interface para com o sistema de armazenamento e de geração de gráficos. A implementação das funcionalidades de MQTT desenvolvidas no *driver* “DriverMQTT.c” gerenciam conexões, reconexões, recebimento e envio de dados, e suas funções foram prototipadas no arquivo de cabeçalho “DriverMQTT.h”, no qual é responsável pela inclusão de outras bibliotecas. As funções do MQTT principais são demonstradas na Figura 23.

Figura 23 – Funções principais de MQTT.

DriverMQTT.c

main > DriverMQTT.c > mqtt_event_handler(esp_mqtt_event_handle_t)

```
61  void enviaMQTT(char message[])
62  {
63      if(connected) esp_mqtt_client_publish(client, (char*) cJSON_GetObjectItem(config, "PUBLISH")
64  }
65
66  static void init_mqtt( void ){
67      mqtt_event_group = xEventGroupCreate();
68      config = get_json();
69      const esp_mqtt_client_config_t mqtt_cfg = {
70          .uri = {(char*) cJSON_GetObjectItem(config, "HOST")->valuestring},
71          .client_id = { (char*) cJSON_GetObjectItem(config, "ID")->valuestring},
72          .event_handle = mqtt_event_handler
73      };
74      client = esp_mqtt_client_init(&mqtt_cfg);
75      esp_mqtt_client_start(client);
76      xEventGroupWaitBits(mqtt_event_group, CONNECTED_BIT0, false, true, pdMS_TO_TICKS(20000));
77
78      xTaskCreatePinnedToCore( vTaskCode, "TASK_MQTT", 4096, NULL, 2, NULL, 1);
79  }
```

Fonte: Autoria Própria.

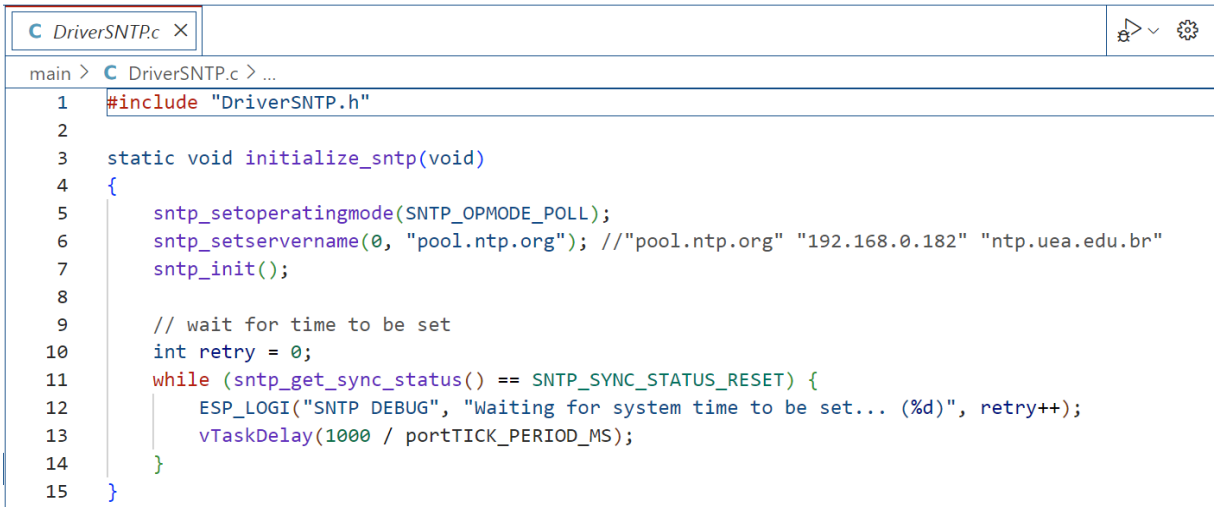
Através do MQTT são enviadas medições periódicas ou através de requisições, realizadas através dos tópicos e do *broker* configurados no *firmware*. Para a conexão do MQTT, o *firmware* precisa primeiramente da conexão com o Wi-Fi, e estas funcionalidades estão implementadas no *driver* “DriverWIFI.c”, bem como toda configuração necessária para realização de conexão e tratamento de reconexões. No arquivo de cabeçalho “DriverWIFI.h” constam as prototipações das funções implementadas e a inclusão de outras bibliotecas. A função principal implementada no *driver* do Wi-Fi é demonstrada na Figura 24.

Figura 24 – Função principal implementada no *driver* do Wi-Fi.

Fonte: Autoria Própria.

Para a sincronização temporal dos pacotes de medição enviados via remota, foi escolhido o Protocolo de Tempo de Rede Simples, do inglês *Simple Network Time Protocol* (SNTP), cujo implementação de funcionalidades foi realizada no *driver* “DriverSNTP.c”, onde sua funcionalidade principal é demonstrada na Figura 25. Os protótipos das funções e a inclusão de outras bibliotecas é realizada no arquivo de cabeçalho “DriverSNTP.h”.

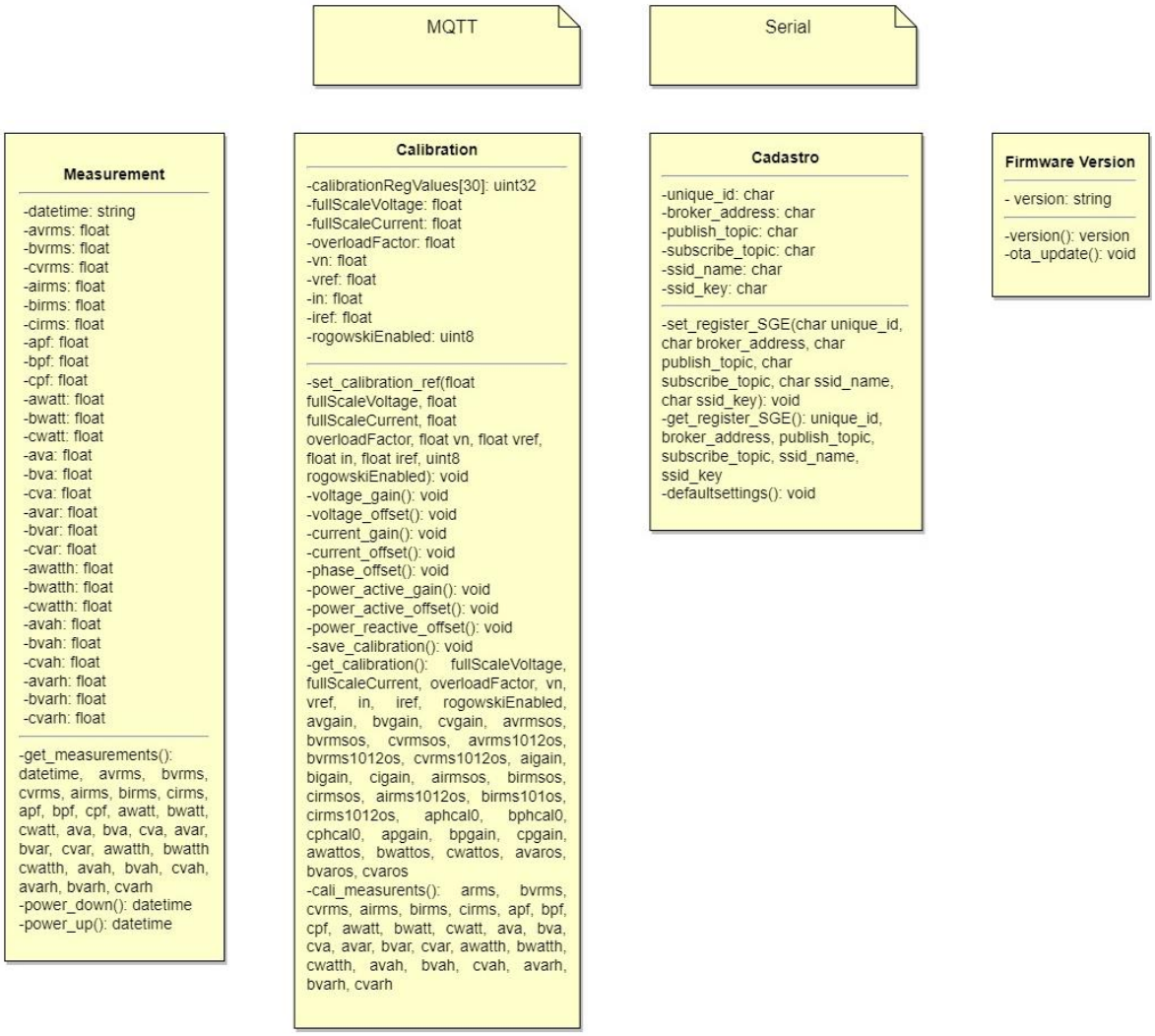
Figura 25 – Funcionalidade principal do *driver* do SNTP.



Fonte: Autoria Própria.

Para o tratamento dos comandos enviados tanto via Serial quanto via MQTT, a biblioteca “Funcionalidades.c” foi implementada. A tarefa de medição, o envio periódico das medições e a geração de eventos de interrupção no fornecimento de energia elétrica é realizado nesta biblioteca, bem como a implementação das funções de cadastro e de calibração, como demonstra a Figura 26. Os protótipos de funções e a inclusão de outras bibliotecas foram realizadas no arquivo de cabeçalho “Funcionalidades.h”.

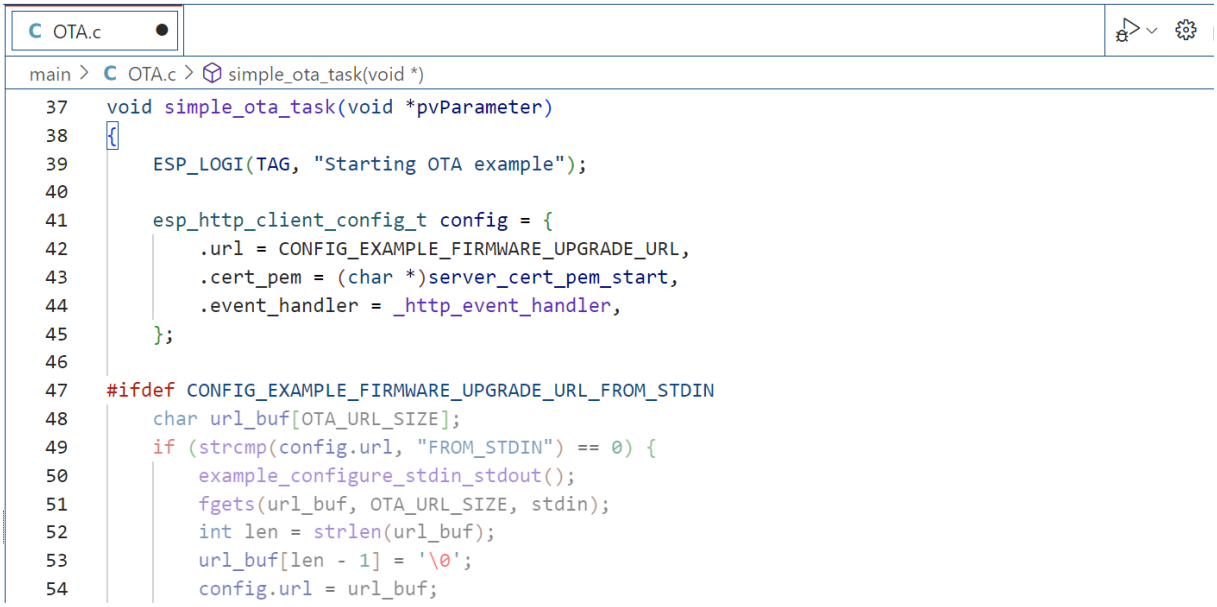
Figura 26 – Modelagem do tratamento das funcionalidades em “Funcionalidades.c”.



Fonte: Autoria Própria.

Para a praticidade em ocasionais atualizações de *firmware*, foi implementado a funcionalidade de atualização “Pelo Ar”, do inglês *Over-The-Air* (OTA), onde sua inicialização e suas funcionalidades foram implementadas na biblioteca “OTA.c”, demonstrado na Figura 27. Os protótipos das funções e a inclusão de outras bibliotecas foram realizadas no arquivo de cabeçalho “OTA.h”

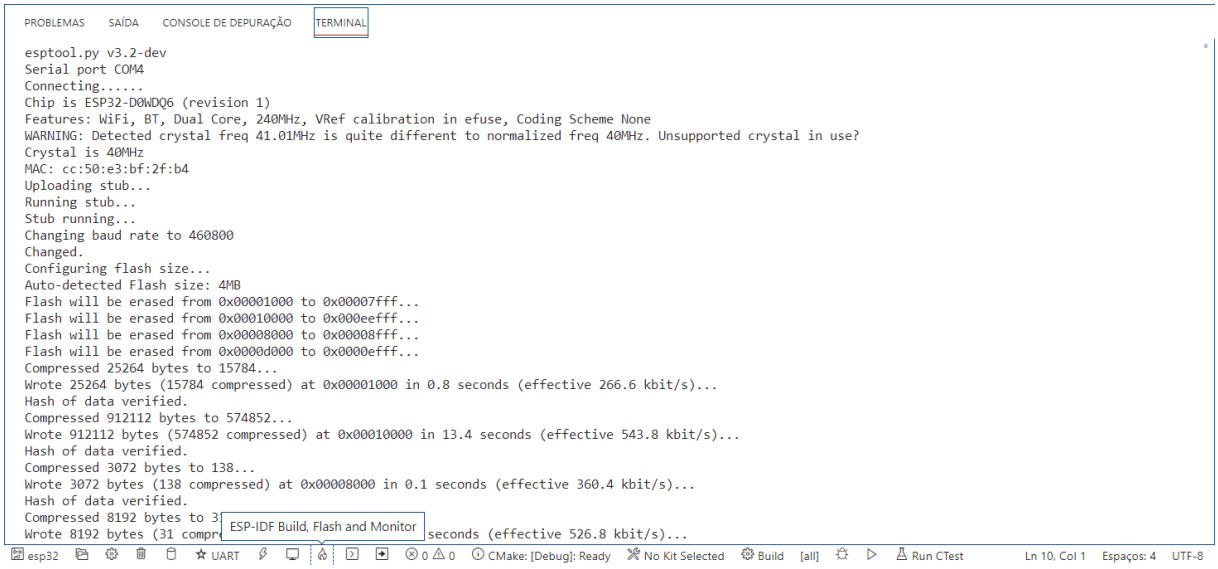
Figura 27 – Funções da biblioteca “OTA.c”



Fonte: Autoria Própria.

Com o *firmware* desenvolvido pode-se embarca-lo no protótipo do *hardware*. Utilizando o *software* Visual Studio Code com a extensão da Espressif IDF, a compilação e o embarque do *firmware* no protótipo são facilitados pelas ferramentas da extensão. Na Figura 28 é demonstrada a compilação e o embarque do *firmware*.

Figura 28 – Compilação e embarque do *firmware*.



Fonte: Autoria Própria.

3.4 DESENVOLVIMENTO DO *SCRIPT* PARA ARMAZENAMENTO DE DADOS

A implementação da base de dados iniciou-se pelo desenvolvimento de um *script* para armazenamento dos dados coletados, desenvolvido em linguagem de programação Python, utilizando o *software* PyCharm 2022.1 com o uso da biblioteca "*peewee"*. Para a criação de uma base de dados, primeiramente é nomeado e criado o arquivo, e por se tratar de uma base de dados relacional, deve-se modelar as tabelas e os dados que serão criados na base de dados. Para a implementação da base de dados deste trabalho, os dados enviados pelas medições foram modelados de acordo com os dados enviados pelo *firmware*, nos quais são os parâmetros elétricos de tensão RMS, corrente RMS, fator de potência, potências ativa, reativa e aparente, e energias ativa, reativa e aparente. Através do *script* "Database_create.py" foram modelados esses dados e gerada a base de dados, conforme demonstrado na Figura 29.

Figura 29 – Criação da base de dados.

Database_create.py | TCC_DATABASE.py

```
import peewee

db = peewee.SqliteDatabase('Data.db')


class BaseModel(peewee.Model):

    class Meta:
        database = db


class Data_Placa(BaseModel):

    time_msg = peewee.TimeField()
    power_status = peewee.CharField()
    avrms = peewee.FloatField()
    bvrms = peewee.FloatField()
    cvrms = peewee.FloatField()
    airms = peewee.FloatField()
    birms = peewee.FloatField()
    cirms = peewee.FloatField()
    apf = peewee.FloatField()
    bpf = peewee.FloatField()
    cpf = peewee.FloatField()
    awatt = peewee.FloatField()
```

Fonte: Autoria Própria.

Para o recebimento, tratamento e armazenamento dos dados na base de dados foi implementado um *script* em Python responsável por adquirir os dados enviados pelo *firmware* através do protocolo MQTT, filtrar os dados recebidos e armazena-los na base de dados. Na implementação do *script* "TCC_DATABASE.py" foram utilizadas quatro bibliotecas, sendo elas: a biblioteca "*paho-MQTT*" responsável pela conexão e gerenciamento do MQTT; a

biblioteca “datetime” responsável pelas funcionalidades de data e hora do sistema de aquisição de dados; a biblioteca “time” responsável pelo cálculo de tempo entre as notificações de interrupção no fornecimento de energia elétrica; e a biblioteca “json” responsável por converter os dados transmitidos em formato JSON em formato de estrutura de dados. A classe da base de dados criada anteriormente é incluída neste *script* para o uso das funcionalidades de base de dados, conforme mostra Figura 30.

Figura 30 – *Script* de tratamento de dados e armazenamento na base de dados.

Database_create.py | TCC_DATABASE.py

```python
import paho.mqtt.client as mqtt
from Database_create import *
from datetime import datetime
from time import time
import json


class Backend():

    def __init__(self):
        self.time_interrupt = 0
        self.id = "TCC_Gabriel_ID"
        self.broker = "broker.hivemq.com"
        self.sub = "TCC_Gabriel/001P"
        self.pub = "TCC_Gabriel/001S"

        self.client = mqtt.Client(self.id)
        self.client.on_message = self.on_message
        self.client.on_connect = self.on_connect

        self.client.connect(self.broker)

    def on_connect(self, client, userdata, flags, rc):

        print("[STATUS] Conectado ao Broker. Resultado da conexao: " + str(rc) + " " + datetime.now().strftime('%d/%m/%Y
        self.client.subscribe(self.sub)

    def on_message(self, client, userdata, msg):
        message = json.loads(msg.payload.decode('utf-8'))  # Converte a messagem recebida
        if msg.topic == "TCC_Gabriel/001P":
            for k in message:
                if k == "cmd":
                    if message["cmd"] == "get_measurements":
                        Data_Placa.create(time_msg=message["datetime"], power_status="ok", avrms=message["avrms"], bvrms
                    elif message["cmd"] == "power_down":
                        self.time_interrupt = time()
                        Interrupt.create(time_msg=message["datetime"], power_status="power_down", interrupt_time=0)
                    elif message["cmd"] == "power_up":
                        if self.time_interrupt > 0:
                            self.time_interrupt = time() - self.time_interrupt
```

Fonte: Autoria Própria.

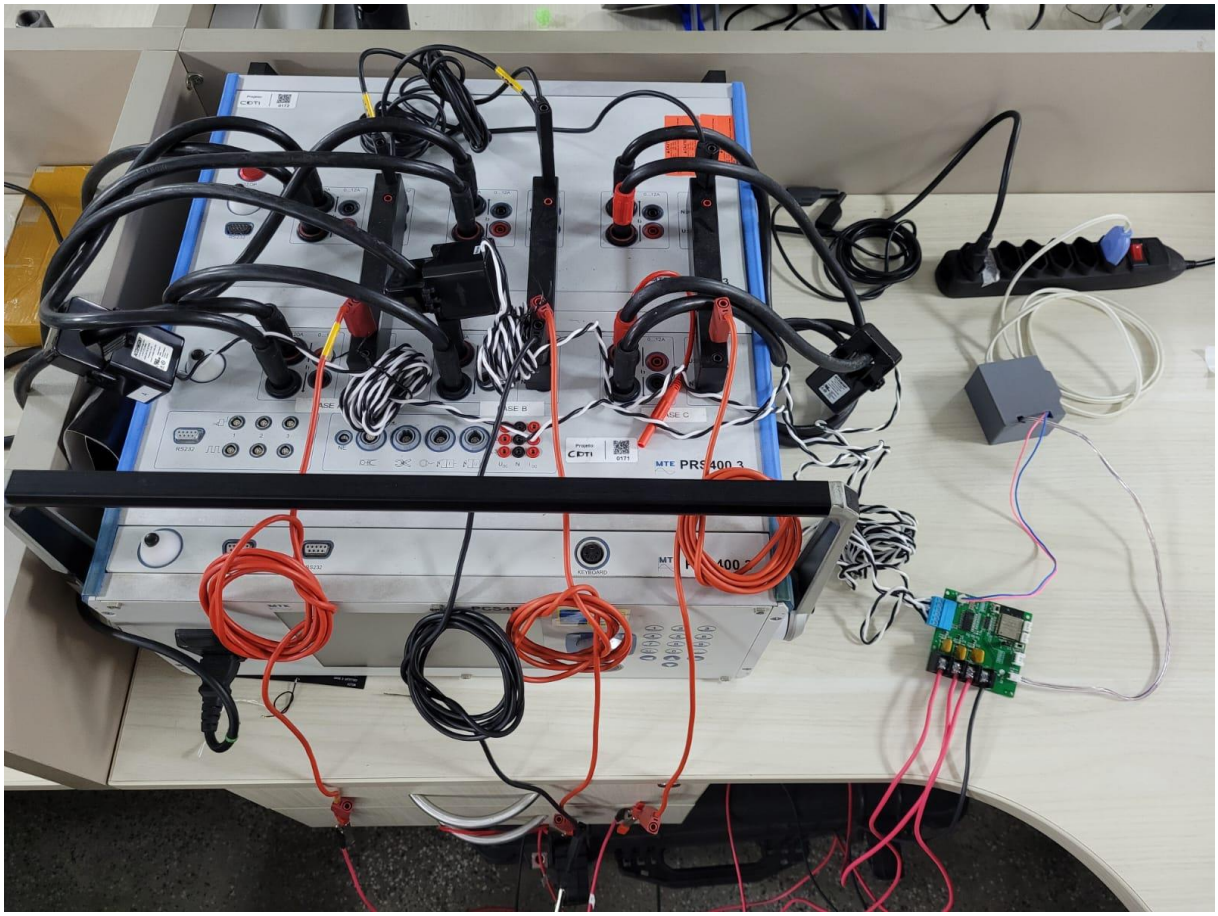
3.5 CALIBRAÇÃO DO PROTÓTIPO DE *HARDWARE* E TESTES DE VALIDAÇÃO

A calibração do protótipo é realizada de acordo com a folha de especificação do ADE7758, no qual foram implementados devidos cálculos de calibração no *firmware* e

desenvolvidas funções para facilitar o processo de calibração, nas quais são as funções de calibração de ganho de tensão e ganho de corrente, calibração de *offset* de tensão, calibração de ganho das potências ativa e aparente e calibração de *offset* de fase. Devem ser enviados comandos através da interface serial UART externada no protótipo para o acesso às funcionalidades de calibração.

O processo de calibração é realizado utilizando uma fonte de calibração trifásica modelo PTS400.3, no qual gera sinais de tensão e corrente precisos para a calibração de dispositivos. Os parâmetros elétricos são ajustáveis através de interface com teclado, onde os parâmetros de tensão, corrente e defasagem são utilizados para a calibração do protótipo. A conexão do protótipo com a fonte é realizada na parte superior da ferramenta, conforme demonstra Figura 31.

Figura 31 – Protótipo conectado à fonte de calibração.

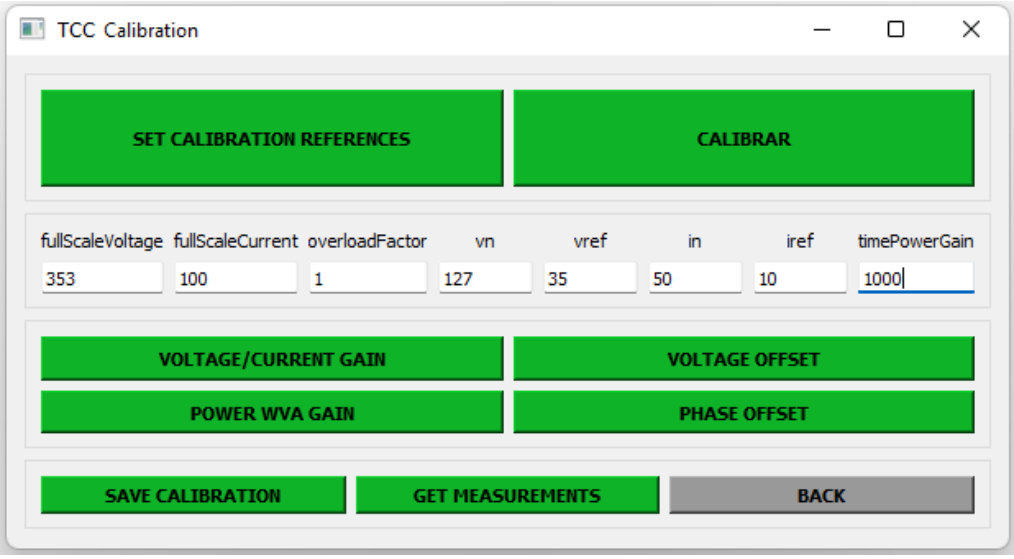


Fonte: Autoria Própria.

Para a facilitação no envio dos comandos via interface serial e praticidade no processo de calibração, foi desenvolvido uma interface visual em Python utilizando a biblioteca PyQt5,

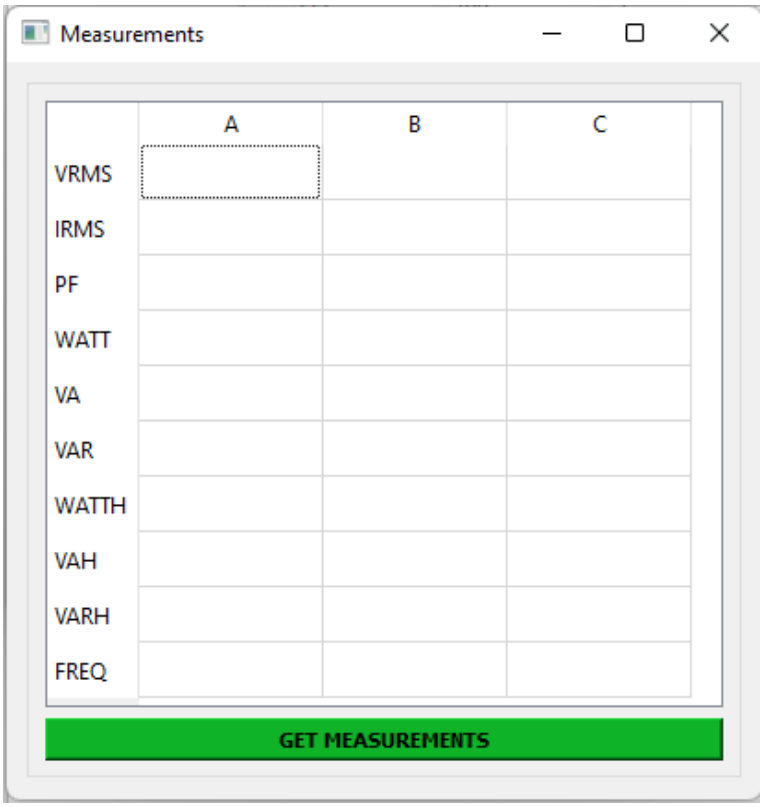
que permite a programação de telas interativas. Para a calibração do protótipo foram desenvolvidas duas telas interativas, a primeira referente às funcionalidades de calibração e de salvamento de calibração (Figura 32), e a segunda tela sendo a tela de medições (Figura 33).

Figura 32 – Tela de calibração do protótipo.



Fonte: Autoria Própria.

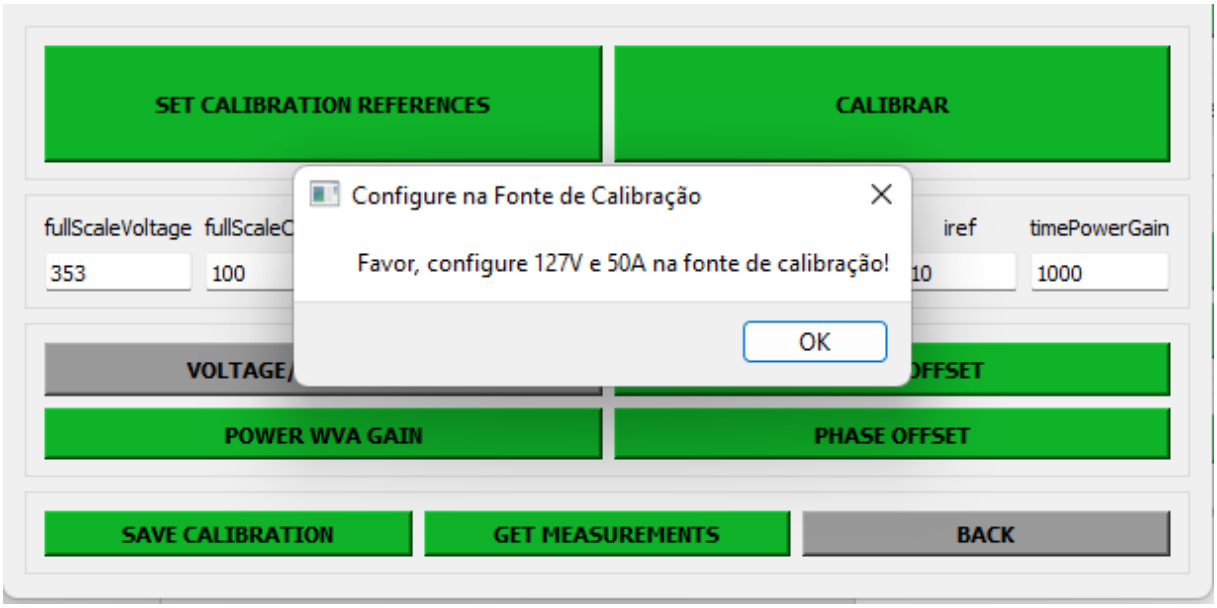
Figura 33 – Tela de medições para calibração do protótipo.



Fonte: Autoria Própria.

A tela de calibração (Figura 32) conta com caixa de textos para inserção de parâmetros de referência para a calibração, tais como o fundo de escala de tensão do protótipo, fundo de escala de corrente do TC utilizado, fator de sobrecarga do TC, tensão nominal a ser calibrada, tensão de referência a ser utilizada na calibração (geralmente 10% do valor do fundo de escala), corrente nominal a ser calibrada, corrente de referência (geralmente 10% do valor do fundo de escala) e o tempo em milissegundos do cálculo de potência. Após o preenchimento de todos os parâmetros de calibração, o processo de calibração pode ser realizado passo a passo ao clicar em cada botão de função, ou pode-se clicar no botão "Calibrar" para iniciar o processo de calibração automatizado, onde os comandos são enviados automaticamente e instruções aparecem na tela para a continuidade da calibração, como demonstra a Figura 34.

Figura 34 – Telas de instruções para a calibração.

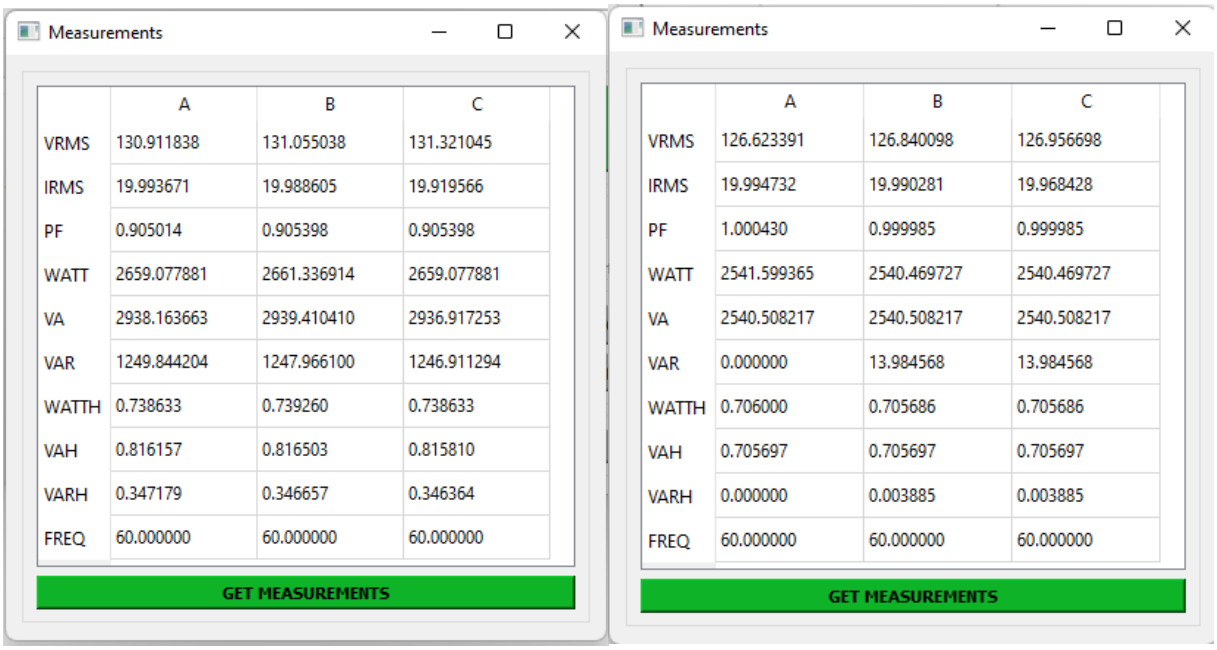


Fonte: Autoria Própria.

Ao seguir as instruções de calibração, configurando a fonte de calibração conforme instruído nas telas, é possível calibrar o protótipo de maneira facilitada, garantindo uma rapidez na calibração e a segurança de que todos os passos estão sendo seguidos corretamente. Ao final da calibração, os parâmetros de calibração devem ser salvos para que o protótipo consiga realizar medições com menor erro de leitura. Para salvar a calibração deve-se clicar no botão "*save calibration*".

Para comparação de calibração, as figuras 35a e 35b demonstram os valores medidos antes e depois do processo de calibração com a fonte configurada para 127V, 20A e 0º de defasagem entre tensão e corrente.

Figura 35 – Medições realizadas antes (a) e depois (b) da calibração.

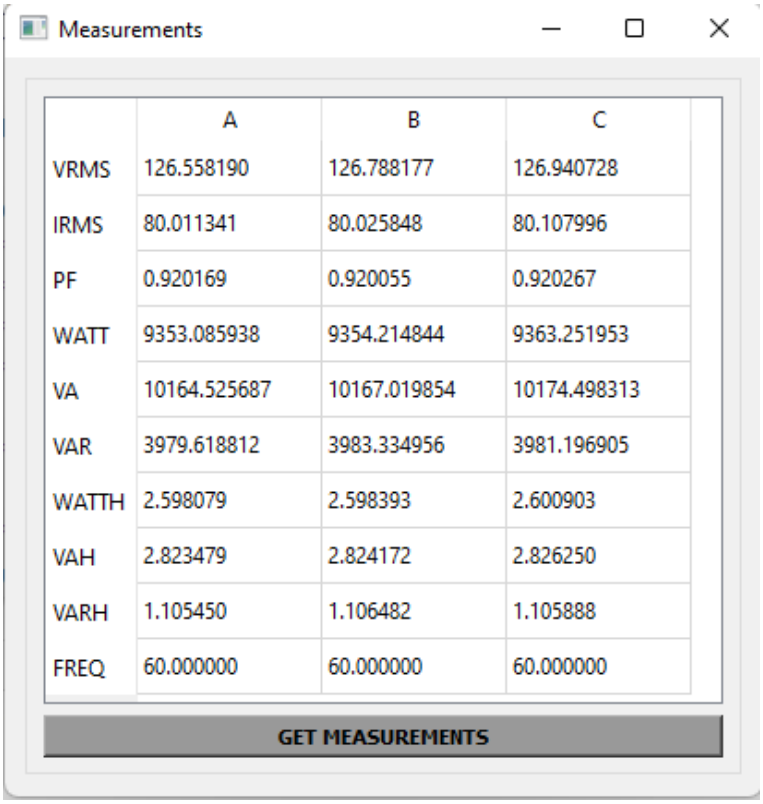


(a) (b)

Fonte: Autoria Própria.

Com o protótipo calibrado, pode-se realizar testes de validação com valores mais elevados na fonte de calibração para a validação da segurança no uso do dispositivo em campo. Para isso, a fonte foi configurada em 127V, 80A e 23,074º de defasagem entre tensão e corrente, garantindo o teste de validação e a própria validação da calibração do dispositivo, conforme demonstra a tela de medições na Figura 36.

Figura 36 – Medições realizadas na fonte de calibração trifásica.



	A	B	C
VRMS	126.558190	126.788177	126.940728
IRMS	80.011341	80.025848	80.107996
PF	0.920169	0.920055	0.920267
WATT	9353.085938	9354.214844	9363.251953
VA	10164.525687	10167.019854	10174.498313
VAR	3979.618812	3983.334956	3981.196905
WATTH	2.598079	2.598393	2.600903
VAH	2.823479	2.824172	2.826250
VARH	1.105450	1.106482	1.105888
FREQ	60.000000	60.000000	60.000000

Fonte: Autoria Própria.

Foram realizados testes de comunicação remota para a validação da comunicação com o protótipo. Para a realização do teste o protótipo foi ligado à fonte 5V, sem conectar os cabos de tensão na entrada de tensão e os Transformadores de Corrente na entrada de corrente. Após isso, foi realizado o teste com o *script* "TCC_DATABASE.py" sem a base de dados para a validação somente da comunicação, conforme mostra a Figura 37, para que o protótipo envie apenas medições nulas.

Figura 37 – Dados sendo recebidos pelo *script* em Python.

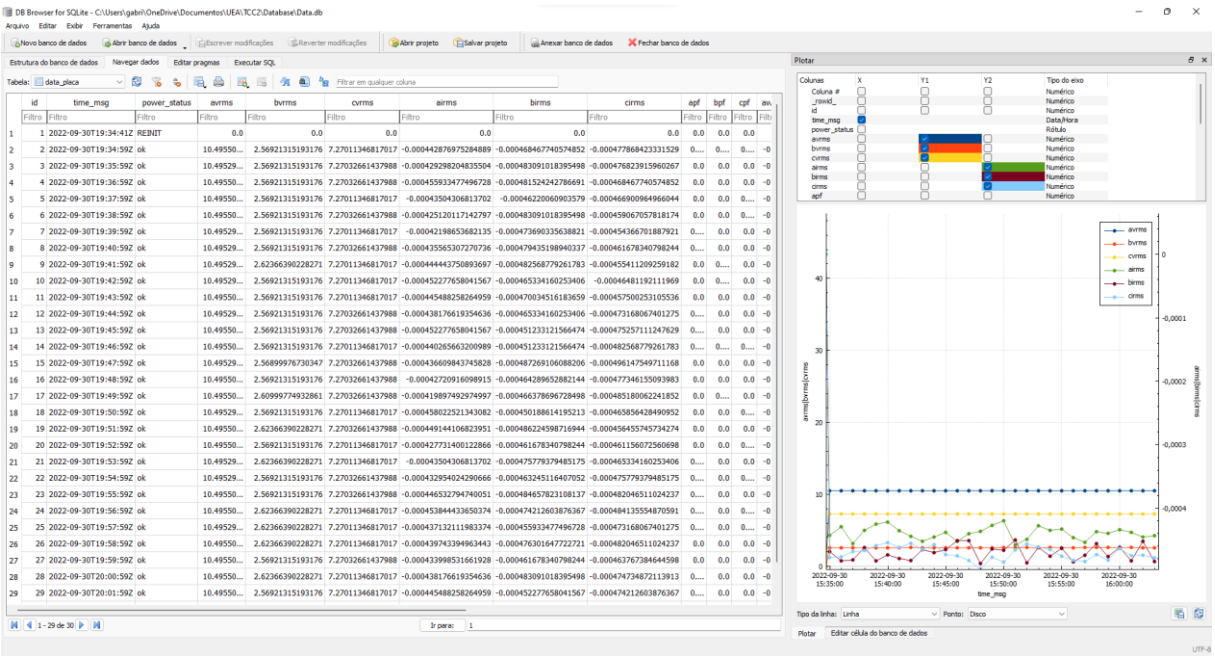
Fonte: Autoria Própria.

O recebimento de dados de acordo com o esperado valida a comunicação e a sincronização do protótipo, bem como o recebimento e tratamento dos dados por parte do *script*.

3.6 INTEGRAÇÃO COM A BASE DE DADOS

Com a validação da comunicação do protótipo e do *script* de recebimento realizada, o próximo passo é a integração da comunicação do protótipo com o recebimento e armazenamento na base de dados. Para a validação da base de dados, o teste foi realizado com o protótipo ligado à fonte 5V, sem os cabos de tensão conectados à entrada de tensão e sem os Transformadores de Corrente conectados à entrada de corrente. Para o teste de integração, foi utilizado o *script* “TCC_DATABASE.py” com a base de dados, validando seu funcionamento e sua conexão. Utilizando o *software* “DB Browser para SQLite” para a visualização dos dados é possível validar a comunicação do protótipo com a base de dados. Através do *software* de visualização dos dados é possível gerar gráficos interativos para uma análise mais precisa dos dados, sendo possível realizar o cruzamento dos dados com dois eixos verticais. Na Figura 38 é possível verificar a tabela com dados de medição, no qual exibe valores insignificantes, pois não há nada conectado nas entradas de tensão e corrente, logo o valor lido é residual da calibração, ainda na Figura 38 é possível visualizar a geração de gráficos no lado direito da tela.

Figura 38 – Utilização do *software* DB Browser para visualização dos dados armazenados.



Fonte: Autoria Própria.

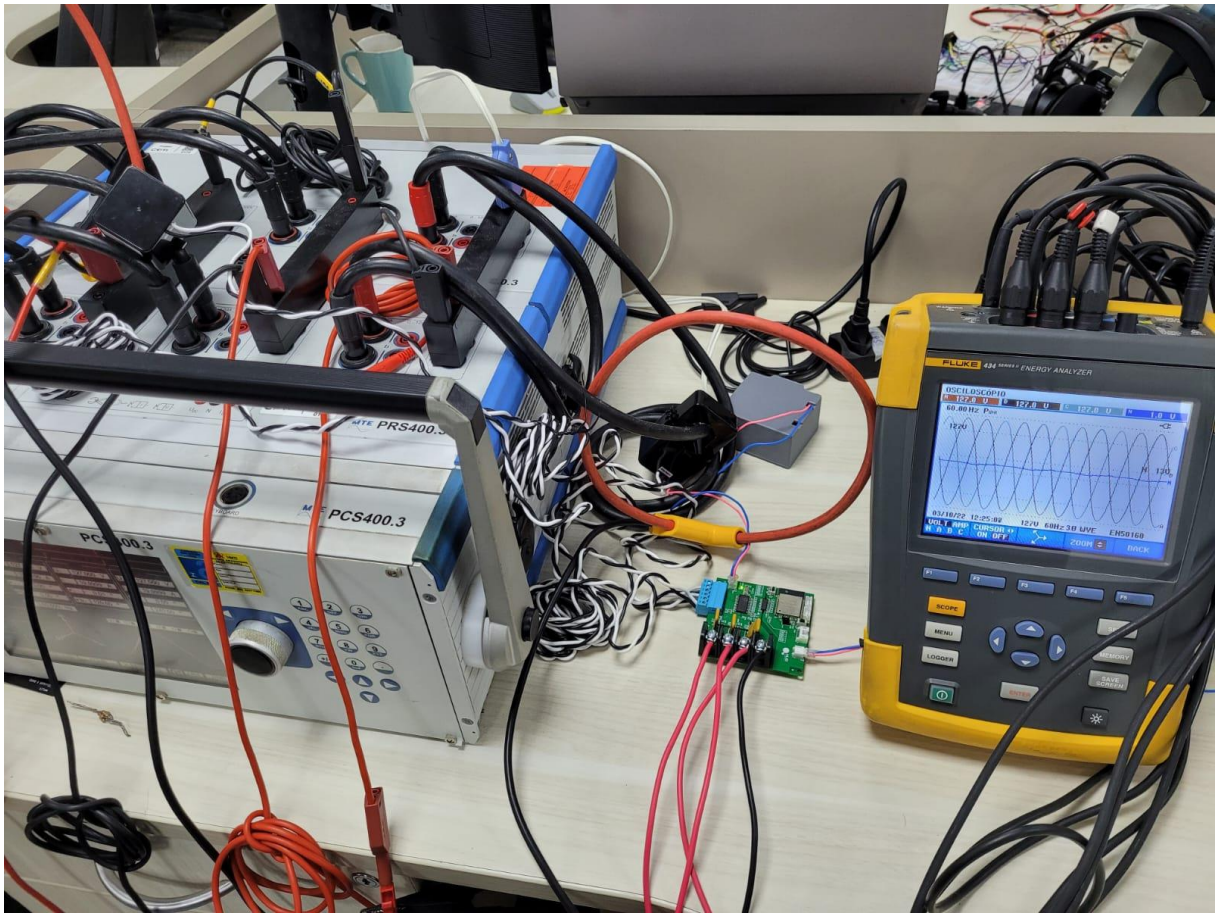
3.7 VALIDAÇÃO DO PROTÓTIPO DE SISTEMA

Para a validação final do protótipo de sistema, o protótipo de *hardware* foi instalado na fonte de calibração PTS400.3 juntamente com o analisador de energia FLUKE 434 Série II. O FLUKE 434 Série II foi utilizado na validação final do protótipo pois providencia medições precisas e o armazenamento dessas medições para uma análise futura.

“O Analyzer oferece um conjunto abrangente de medições para verificar os sistemas de distribuição de força. Algumas proporcionam uma impressão geral do desempenho do sistema de força. Outros são usados para investigar detalhes específicos.” (FLUKE, 2012, p. 3-1).

A validação final foi realizada durante 19 horas, onde ambos dispositivos estavam conectados aos mesmos pontos, conforme mostra Figura 39.

Figura 39 – Teste de validação final do protótipo de sistema.

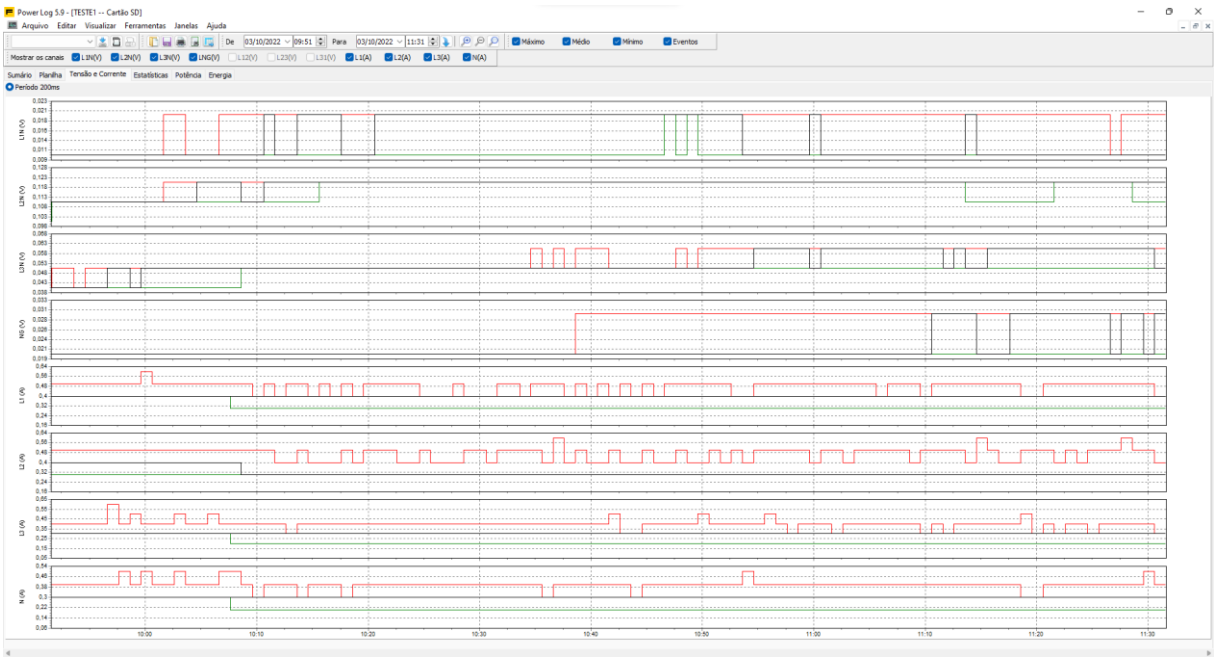


Fonte: Autoria Própria.

Durante o procedimento de teste foram simuladas situações de interrupção no fornecimento de energia elétrica para a análise de dados posterior. Ao final do teste de validação

final, o cartão de memória (SD Card) presente no FLUKE 434 foi retirado do dispositivo e inserido no notebook para aquisição dos dados armazenados, onde é possível realizar a visualização dos dados através do *software* “Power Log 430-II versão 5.9”, conforme demonstra Figura 40.

Figura 40 – *Software* de visualização de dados do FLUKE 434 Série II.



Fonte: Autoria Própria.

Ao final do teste de validação, os dados armazenados no arquivo “Database_TCC.db” do protótipo de sistema foram visualizados no *software* DB Browser para SQLite, enquanto os dados do FLUKE 434 Série II foram visualizados no *software* Power Log 430-II. A visualização dos dados de ambos dispositivos permitiu a comparação entre as medições, podendo desta forma validar o protótipo de sistema.

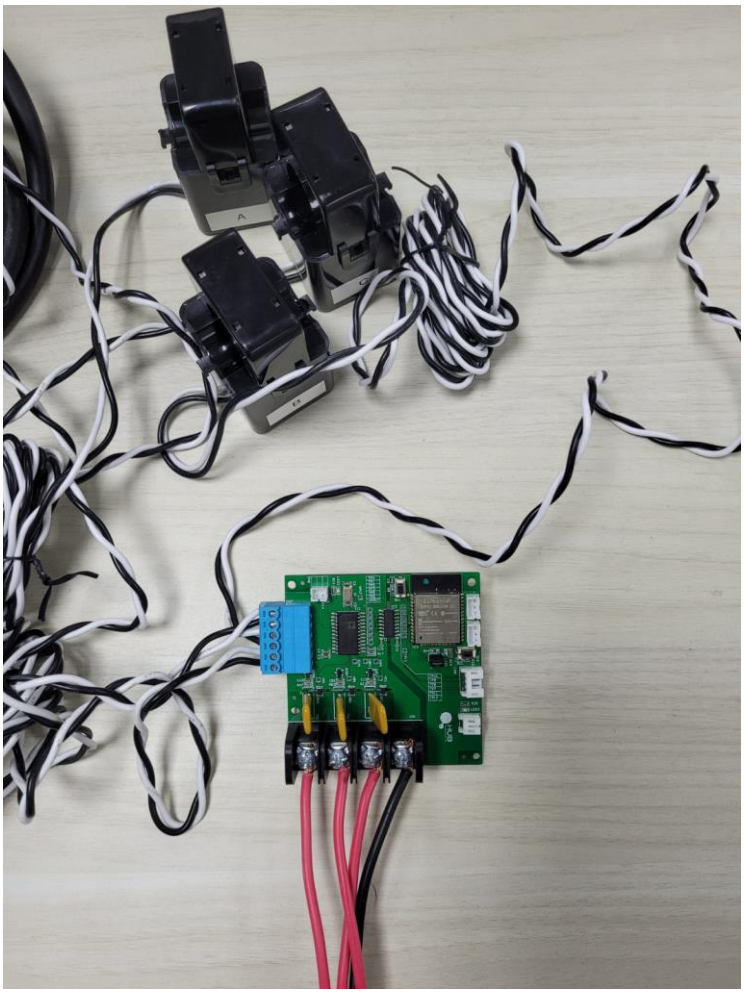
4 RESULTADOS OBTIDOS

Este capítulo apresenta os resultados obtidos com a realização da implementação de todas as etapas e está dividido em 3 seções, onde serão apresentados o protótipo de sistema desenvolvido, os dados coletados ao fim do teste de validação e a comparação dos dados obtidos pelo protótipo de sistema com os dados obtidos pelo analisador de energia FLUKE 434 Série II.

4.1 PROTÓTIPO DE SISTEMA DE MONITORAMENTO

Nesta seção será apresentado o protótipo de sistema industrial para monitoramento de parâmetros elétricos e da qualidade de fornecimento de energia elétrica. O protótipo de sistema de monitoramento conta com três partes do sistema, o protótipo de *hardware*, o protótipo de *firmware* e o *script* de tratamento e armazenamento de dados. O *hardware* é responsável pela entrada dos parâmetros elétricos, tais como tensão e corrente elétrica e contém um microcontrolador para a realização do processamento dos dados, no qual é realizado pelo protótipo de *firmware*, responsável pelo processamento dos dados e pelo envio para o *script* de armazenamento, que é responsável pelo armazenamento e disposição gráfica dos parâmetros transmitidos, sendo considerado a saída do sistema. Na Figura 41 pode-se visualizar o protótipo de *hardware* desenvolvido neste trabalho. Na Figura 42 pode-se visualizar trecho do protótipo de *firmware* desenvolvido no decorrer deste trabalho. Na Figura 43 pode-se visualizar trecho do *script* responsável pelo tratamento e armazenamento dos dados, desenvolvido no decorrer deste trabalho.

Figura 41 – Protótipo de *hardware* desenvolvido.



Fonte: Autoria Própria.

Figura 42 – Trecho do protótipo de *firmware* responsável pelas medições.

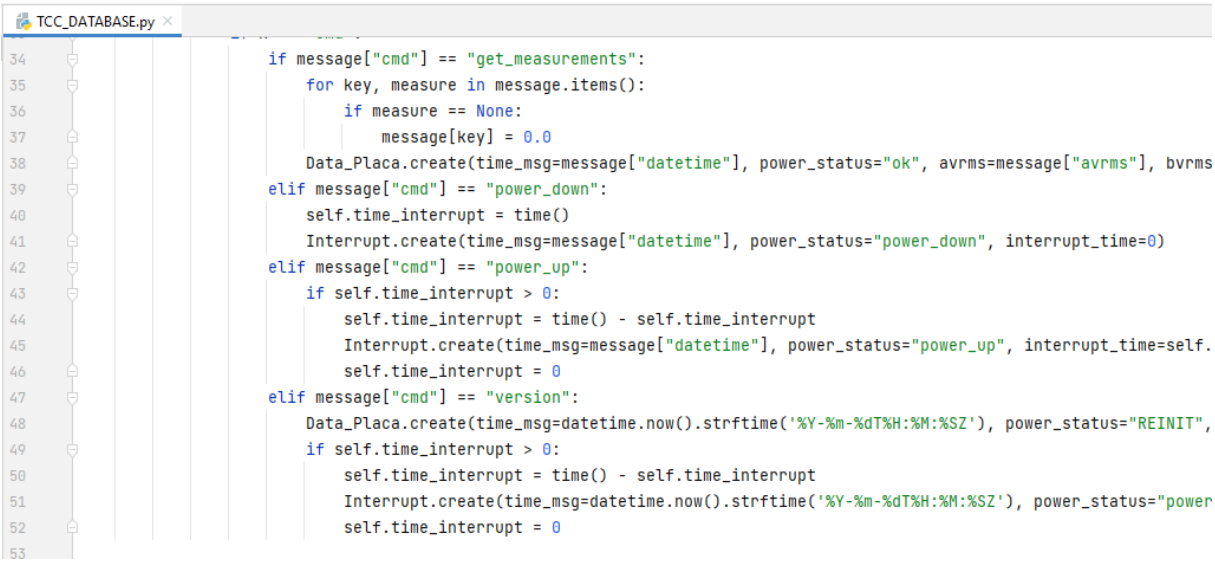
Funcionalidades.c

main > Funcionalidades.c > init_functions(void)

```
138         else if (!strcmp((char *) cJSON_GetObjectItem(request_cmd, "cmd")->valuestring, "get_measurements"))
139         {
140             cJSON_AddStringToObject(response_cmd, "cmd", "get_measurements");
141             cJSON_AddStringToObject(response_cmd, "datetime", data_formatada);
142             cJSON_AddNumberToObject(response_cmd, "avrms",   (ADE7758_average.avrms));
143             cJSON_AddNumberToObject(response_cmd, "bvrms",   (ADE7758_average.bvrms));
144             cJSON_AddNumberToObject(response_cmd, "cvrms",   (ADE7758_average.cvrms));
145             cJSON_AddNumberToObject(response_cmd, "airms",   (ADE7758_average.airms));
146             cJSON_AddNumberToObject(response_cmd, "birms",   (ADE7758_average.birms));
147             cJSON_AddNumberToObject(response_cmd, "cirms",   (ADE7758_average.cirms));
148             cJSON_AddNumberToObject(response_cmd, "apf",     (ADE7758_average.apf));
149             cJSON_AddNumberToObject(response_cmd, "bpf",     (ADE7758_average.bpf));
150             cJSON_AddNumberToObject(response_cmd, "cpf",     (ADE7758_average.cpf));
151             cJSON_AddNumberToObject(response_cmd, "awatt",   (ADE7758_average.awatt));
152             cJSON_AddNumberToObject(response_cmd, "bwatt",   (ADE7758_average.bwatt));
153             cJSON_AddNumberToObject(response_cmd, "cwatt",   (ADE7758_average.cwatt));
154             cJSON_AddNumberToObject(response_cmd, "ava",     (ADE7758_average.ava));
155             cJSON_AddNumberToObject(response_cmd, "bva",     (ADE7758_average.bva));
156             cJSON_AddNumberToObject(response_cmd, "cva",     (ADE7758_average.cva));
157             cJSON_AddNumberToObject(response_cmd, "avar",    (ADE7758_average.avar));
158             cJSON_AddNumberToObject(response_cmd, "bvar",    (ADE7758_average.bvar));
159             cJSON_AddNumberToObject(response_cmd, "cvar",    (ADE7758_average.cvar));
160             cJSON_AddNumberToObject(response_cmd, "awatth", (ADE7758_average.awatth));
```

Fonte: Autoria Própria.

Figura 43 – Trecho do *script* de tratamento e armazenamento de dados.

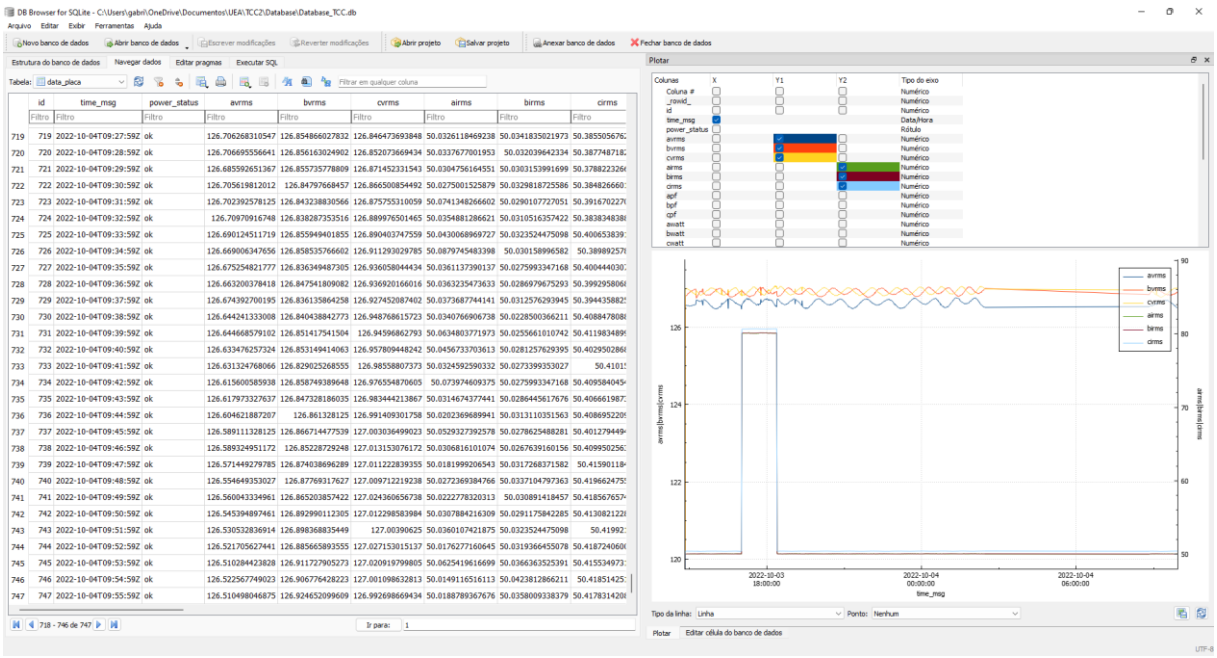


Fonte: Autoria Própria.

4.2 DADOS COLETADOS PELO PROTÓTIPO DE MONITORAMENTO

Como resultado do teste de validação final do protótipo de sistema têm-se os dados coletados no qual foram armazenados a base de dados nomeada como “Data.db”. O *software* DB Browser para SQLite foi utilizado para a visualização dos dados armazenados na base de dados e para a geração de gráficos com os parâmetros medidos, conforme mostrado na Figura 44.

Figura 44 – Visualização dos dados coletados pelo protótipo de sistema.

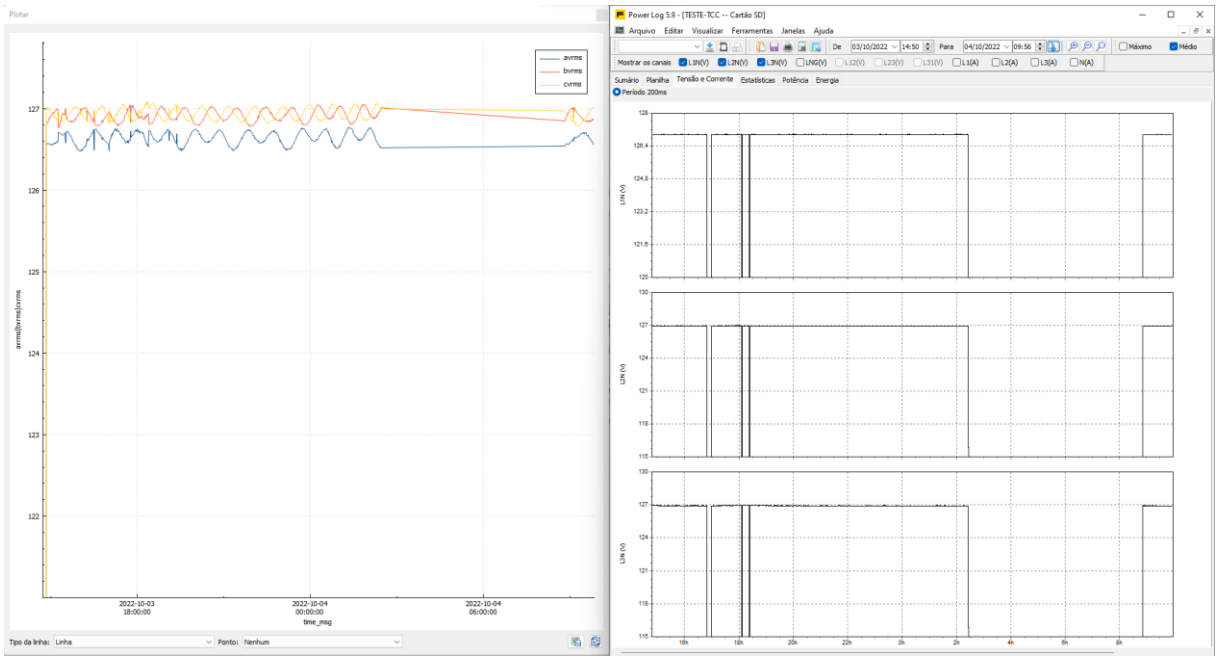


Fonte: Autoria Própria.

4.3 COMPARAÇÃO DOS DADOS COLETADOS

Para a comparação dos dados coletados foram utilizados o *software* DB Browser para SQLite para a visualização dos dados coletados pelo protótipo de sistema e o *software* Power Log 430-II para a visualização dos dados coletados pelo FLUKE 434. A análise comparativa deu-se primeiramente pela comparação visual dos gráficos gerados pelos *softwares*, conforme demonstrado nas Figuras 45 e 46, e posteriormente por comparação de medição escolhida arbitrariamente, sincronizadas pelo horário e comparados cada parâmetro simultaneamente, conforme demonstrado no Quadro 1. Na Figura 46 é possível visualizar o monitoramento das interrupções no fornecimento de energia elétrica, identificadas pela data e horário do evento ocorrido (campo "*time_msg*"), tipo de evento (campo "*power_status*", onde "*power_down*" representa a interrupção no fornecimento, e o "*power_up*" representa o retorno do fornecimento) e o tempo da interrupção (campo "*interrupt_time*", onde o zero representa o início da interrupção, caracterizado pelo campo "*power_status*" como "*power_down*", e o tempo total da interrupção como o tempo transcorrido em segundos, caracterizado pelo campo "*power_status*" como "*power_up*").

Figura 45 – Comparação visual dos gráficos das medições gerados pelos *softwares*.



Fonte: Autoria Própria.

Figura 46 – Visualização dos dados de interrupções geradas durante o teste de validação.

Tabela: interrupt — Filtrar em qualquer coluna

	id	time_msg	power_status	interrupt_time
	Filtro	Filtro	Filtro	Filtro
1	1	2022-10-03T16:49:12Z	power_down	0
2	2	2022-10-03T16:59:13Z	power_up	601
3	3	2022-10-03T18:06:57Z	power_down	0
4	4	2022-10-03T18:07:13Z	power_up	16
5	5	2022-10-03T18:23:15Z	power_down	0
6	6	2022-10-03T18:23:33Z	power_up	18
7	7	2022-10-04T02:27:02Z	power_down	0
8	8	2022-10-04T08:49:49Z	power_up	22967

Fonte: Autoria Própria.

Quadro 1 – Comparação de medição pontual, escolhida arbitrariamente.

Parâmetro medido	Protótipo	FLUKE	Erro relativo
Data e horário	03/10/2022 – 17:30	03/10/2022 – 17:30	-
Tensão RMS (A)	126,63V	126,97V	0,2678%
Tensão RMS (B)	126,86V	126,96V	0,0788%
Tensão RMS (C)	127,03V	126,93V	-0,0788%
Corrente RMS (A)	80,05A	79,7A	-0,4391%
Corrente RMS (B)	80,08A	80,8A	0,8911%
Corrente RMS (C)	80,67A	79,7A	-1,2171%
Fator de Potência(A)	0,92	0,92	0,0000%
Fator de Potência(B)	0,92	0,92	0,0000%
Fator de Potência(C)	0,92	0,92	0,0000%
Potência Ativa(A)	9371W	9330W	-0,4394%
Potência Ativa(B)	9374W	9470W	1,0137%
Potência Ativa(C)	9436W	9330W	-1,1361%
Potência Aparente(A)	10177VA	10120VA	-0,5632%
Potência Aparente(B)	10189VA	10260VA	0,6920%
Potência Aparente(C)	10247VA	10110VA	-1,3551%
Potência Reativa(A)	3970VAr	3990VAr	0,5013%
Potência Reativa(B)	3992VAr	4050VAr	1,4321%
Potência Reativa(C)	3995VAr	3990VAr	-0,1253%

Fonte: Autoria Própria.

CONCLUSÃO

Neste trabalho foi desenvolvido um protótipo de sistema de monitoramento de parâmetros elétricos e da qualidade de fornecimento de energia elétrica. Para isso, foram apresentadas pesquisas referentes aos assuntos e tecnologias empregadas no decorrer deste trabalho, tais como definição de qualidade de fornecimento de energia elétrica, microcontroladores, com ênfase no microcontrolador ESP32, utilizado na implementação do protótipo, ADE778, responsável pelas medições, protocolo MQTT e base de dados SQLite. Também foram apresentados conceitos que definem a pesquisa, tais como eficiência energética, *smart meter* e técnicas de transdução de tensão e corrente elétrica.

As etapas necessárias para a elaboração do protótipo foram descritas detalhadamente no capítulo de materiais e métodos, no qual, seguindo tais etapas, a implementação do projeto ocorreu de maneira satisfatória. Também foram apresentados ferramentas e *softwares* utilizados na implementação do protótipo.

Referente a hipótese e no objetivo citados na introdução, e com base nos resultados obtidos, conclui-se que o protótipo desenvolvido é capaz de realizar o monitoramento de parâmetros elétricos e da qualidade de fornecimento de energia elétrica de forma satisfatória, conforme comprovam os testes e análises comparativas realizadas. Os protótipos de *hardware* e *firmware* e o *script* de armazenamento dos dados coletados desenvolvidos operaram conforme o esperado, onde no *hardware* tem-se um módulo sensor composto por um microcontrolador que se conecta remotamente através de conexão com redes de dados sem fio e possui recursos para realização de processamento dos dados coletados através do ADE7758, no qual digitaliza os parâmetros elétricos. Todos os dados são processados e enviados pelo *firmware*, responsável por toda a lógica do sistema operacional, coletando os dados digitalizados pelo ADE7758 e realizando o monitoramento de eventuais interrupções no fornecimento de energia elétrica. Já o *script* de armazenamento de dados é responsável por todo o tratamento e armazenamento das medições enviadas pelo módulo sensor. Nas análises realizadas foi possível obter uma média de erro de aproximadamente 1% quando comparado à um dispositivo comercial de análise de energia, comprovando o funcionamento correto do sistema.

Para a realização de trabalhos futuros, recomenda-se a implementação de um sistema de interface gráfica e de navegação de dados mais intuitiva e mais acessível para o operador. Outra sugestão é a aplicação de algoritmos responsáveis por calcular indicadores coletivos de continuidade, tais como o de Duração Equivalente de Interrupção por Consumidor (DEC) e de

Frequência Equivalente de Interrupção por Consumidor (FEC), aplicados sobre o conceito de qualidade de fornecimento de energia elétrica.

REFERÊNCIAS

ALLEN, G.; OWENS, M. **The Definitive Guide to SQLite.** 2nd. ed. New York: apress, 2010.

ANALOG DEVICES. **ADE7758 (REV. E)**. [S. l.], 2011. Disponível em: <https://www.analog.com/media/en/technical-documentation/data-sheets/ADE7758.pdf>. Acesso em: 12 out. 2021.

ANGLANI, N. *et al.* **Energy smart meters integration in favor of the end user.** *In: 2011 IEEE International Conference on Smart Measurements of Future Grids (SMFG)*, 2011, Bologna. *Proceedings…* Bologna: IEEE, 2011, p. 16-21. Disponível em: <https://ieeexplore.ieee.org/xpl/conhome/6112336/proceeding>. Acesso em: 13 out. 2021.

ARIF, A. et al. **Experimental study and design of smart energy meter for the smart grid**. In: *Renewable and Sustainable Energy Conference (IRSEC)*, 2013, Ouarzazate. *Proceedings…* Ouarzazate: IRSEC, 2013, p. 515-520. Disponível em: <https://ieeexplore.ieee.org/document/6529714>. Acesso em: 10 ago. 2022.

BRASIL. MINISTÉRIO DE MINAS E ENERGIA – MME; EMPRESA DE PESQUISA ENERGÉTICA – EPE. **Plano Nacional de Energia 2050**. Brasília, 2020. Disponível em: <https://www.epe.gov.br/sites-pt/publicacoes-dados-abertos/publicacoes/PublicacoesArquivos/publicacao-227/topico-563/Relatorio%20Final%20do%20PNE%202050.pdf>. Acesso em: 12 out. 2021.

BRASIL. MINISTÉRIO DE MINAS E ENERGIA – MME; EMPRESA DE PESQUISA ENERGÉTICA – EPE. **Série de estudos de demanda**: Eficiência energética na indústria e nas residências no horizonte decenal (2010 – 2019). Rio de Janeiro, 2010. Disponível em: <https://www.epe.gov.br/sites-pt/publicacoes-dados-abertos/publicacoes/PublicacoesArquivos/publicacao-245/topico-270/20100809_4[1].pdf>. Acesso em: 14 out. 2021.

ESPRESSIF SYSTEMS. **ESP32 Series Datasheet**. [S. l.], 2021. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Acesso em: 11 out. 2021.

FAIAS, S.; ESTEVES, J. **Continuity of supply in the Portuguese distribution network and comparison with other European countries**. In: *International Conference On The European Energy Market (EEM*), 2013, Stockholm. *Proceedings…* Stockholm: EEM, 2013, p. 1-8. Disponível em: <https://ieeexplore.ieee.org/document/6607355>. Acesso em: 8 ago. 2022.

FERNANDES, R. A. **Uma estratégia de retrofit para monitoramento de parâmetros elétricos em tempo real com base no metamodelo SmartLVGrid**. Orientador: Ozenir Farah da Rocha Dias. 2022. 110 f. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Amazonas – UFAM, Manaus, 2022. Disponível em: <https://tede.ufam.edu.br/handle/tede/8926>. Acesso em: 14 set. 2022.

FLUKE. **Fluke 434-II/435-II/437-II**: Analisadores de Qualidade de Potência e Energia Trifásicos. [S.l.], 2012. Disponível em: <https://gpcabling.com.br/produto_documento/96710/96710.pdf>. Acesso em: 25 set. 2022.

FORTES, M. Z. *et al*. **Análise da adoção de medidores inteligentes como instrumento da política pública de eficiência energética**. Engevista, v. 19, n. 2, p. 316-327, 2017. Disponível em: <https://periodicos.uff.br/engevista/article/view/9104>. Acesso em: 10 set. 2022.

INTERNATIONAL ENERGY AGENCY – IEA. **Oil Market Report – October 2021**. Paris, 2021. Disponível em: <https://www.iea.org/reports/oil-market-report-october-2021>. Acesso em: 14 out. 2021.

KAGAN, N.; ROBBA, E. J.; SCHMIDT, H. P. **Estimação de indicadores de qualidade da energia elétrica**. São Paulo: Blucher, 2009. Disponível em: <https://books.google.com.br/books?id=ALjLDwAAQBAJ&lpg=PP1&hl=pt-BR&pg=PA2#v=onepage&q&f=false>. Acesso em: 14 out. 2021.

LOCCI, N.; MUSCAS, C.; SULIS, S. **Experimental comparison of MV voltage transducers for power quality applications**. In: *2009 IEEE Instrumentation and Measurement Technology Conference (IMTC)*, 2009, Singapore. *Proceedings*… Singapore: IMTC, 2009, p. 92–97. Disponível em: <https://ieeexplore.ieee.org/document/5168422>. Acesso em: 25 ago. 2022.

MESNIL, J. **Mobile and Web Messaging:** Messaging Protocols for Web and Mobile Devices. Sebastopol: O' Reilly Media, 2014.

OLIVEIRA JÚNIOR, M.; DUARTE, R. O. **Introdução ao Projeto com Microcontroladores e Programação de Periféricos**. Minas Gerais, 2010. Disponível em: <https://docplayer.com.br/656429-Apostila-sobre-introducao-ao-projeto-com-microcontroladores-e-programacao-de-perifericos.html>. Acesso em: 11 out. 2021.

OLIVEIRA, P. C. **Análise de transformadores de corrente para medição**. Dissertação (Mestrado) – Programa de Pós-Graduação em Metrologia para a Qualidade Industrial, Pontífica Universidade Católica do Rio de Janeiro – PUC-RJ, Rio de Janeiro, 2001. Disponível em: <http://repositorio.bom.org.br:8080/xmlui/handle/123456789/1768>. Acesso em: 02 set. 2022.

OLIVEIRA, S. **Internet das Coisas com ESP8266, Arduino e Raspberry Pi**. São Paulo: Novatec, 2017.

ROMANCINI, E. M. R. *et al***. Monitoramento inteligente do consumo de energia elétrica em residências utilizando recursos de IoT**. Anais do Computer on the Beach, v. 13, p. 134-141, 2022.

SKYWORKS. **Si8660/61/62/63 Data Sheet**. [S.l.], 2022. Disponível em: <https://www.skyworksinc.com/-/media/SkyWorks/SL/documents/public/data-sheets/si866x.pdf>. Acesso em: 05 set. 2022.

SOARES, F. S. **Estudo do comportamento de transformadores de corrente em regime transitório**. 2015. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Campina Grande, 2015. Disponível em: <https://repositorio.ufc.br/bitstream/riufc/35009/3/2017_tcc_lmbelmino.pdf>. Acesso em: 20. ago. 2022.

SQLITE. **About SQLite**. [S. l.], 2021. Disponível: <https://www.sqlite.org/about.html>. Acesso em: 11 out. 2021.

STEVAN JUNIOR, S. L. **Internet das Coisas: Fundamentos e aplicações em Arduino e NodeMCU**. São Paulo: Saraiva, 2018.