UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Murilo Roberto Bacini

Desenvolvimento de sistema IoT para monitoramento de energia em casas inteligentes

São José do Rio Preto
2025

Murilo Roberto Bacini

Desenvolvimento de sistema IoT para monitoramento de energia em casas inteligentes

Trabalho de Conclusão de Curso (TCC) apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Conselho de Curso de Bacharelado em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Orientador: Prof. Dr. Rodrigo Capobianco Guido

São José do Rio Preto
2025

B125d

Bacini, Murilo Roberto
Desenvolvimento de sistema IoT para monitoramento de energia em casas inteligentes / Murilo Roberto Bacini. -- São José do Rio Preto, 2025
68 p. : il., tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (UNESP), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto
Orientador: Rodrigo Capobianco Guido

1. Internet das coisas. 2. Energia elétrica Consumo. 3. Sistemas embarcados (Computadores). I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

Murilo Roberto Bacini

Desenvolvimento de sistema IoT para monitoramento de energia em casas inteligentes

Trabalho de Conclusão de Curso (TCC) apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Conselho de Curso de Bacharelado em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Data da defesa: 06/11/2025

Comissão Examinadora:

Prof. Dr. Rodrigo Capobianco Guido
UNESP – Câmpus de São José do Rio Preto
Orientador

Prof. Dr. Leandro Alves Neves
UNESP – Câmpus de São José do Rio Preto

Prof. Dr. Adriano Mauro Cansian
UNESP – Câmpus de São José do Rio Preto

São José do Rio Preto
2025

À minha família e aos meus amigos.

I learned a lot, by the end of everything. The past is past, but that's okay! It's never really gone completely. The future is always built on the past, even if we won't get to see it. Still, it's time for something new, now.

— Alex Beachum, *Outer Wilds* (2019)

Agradecimentos

Primeiramente, agradeço a Deus, por me dar força e guiar meus passos ao longo desta jornada.

À minha família, em especial aos meus pais, Roberto e Sueli, por todo o apoio incondicional e por me proporcionarem uma vida saudável, e ao meu irmão, Bruno, por sua amizade e apoio.

Aos meus amigos, que me acompanham desde muito antes desta jornada acadêmica começar. A amizade de vocês foi um refúgio essencial, um lembrete constante de que há vida para além dos desafios do curso.

Ao corpo docente do curso de Ciência da Computação do IBILCE e à Universidade Estadual Paulista "Júlio de Mesquita Filho", pela formação de excelência e pelo ambiente de aprendizado que me foi proporcionado.

Por fim, ao meu orientador, Prof. Dr. Rodrigo Capobianco Guido, minha mais profunda e sincera gratidão, por toda a ajuda e disponibilidade. Seu apoio foi decisivo para que eu pudesse superar os obstáculos e concluir minha formação.

Resumo

O consumo energético residencial tem crescido significativamente, exigindo soluções inovadoras para o uso eficiente dos recursos. Com o avanço da Internet das Coisas (IoT), tornou-se possível desenvolver sistemas inteligentes capazes de monitorar, em tempo real, o consumo de energia em ambientes domésticos. Este trabalho propõe o desenvolvimento de um sistema completo para este fim, desde o hardware de medição até a visualização dos dados. A solução implementada consiste em um protótipo baseado no microcontrolador ESP32, que coleta dados de tensão e corrente e os transmite de forma segura via protocolo MQTT. A infraestrutura de nuvem foi arquitetada como um pipeline de dados robusto, utilizando um agente Telegraf, executado em um contêiner Docker, para coletar as mensagens e persisti-las em um banco de dados de séries temporais (InfluxDB). Finalmente, uma plataforma de visualização (Grafana) apresenta os dados ao usuário em um dashboard interativo. O projeto foca na acessibilidade, baixo custo e escalabilidade, demonstrando a viabilidade de uma arquitetura profissional com ferramentas de código aberto.

Palavras-chave: Internet das Coisas, monitoramento de energia, casa inteligente, eficiência energética, séries temporais

Abstract

Residential energy consumption has grown significantly, demanding innovative solutions for efficient resource use. With the advancement of the Internet of Things (IoT), it has become possible to develop intelligent systems capable of monitoring electricity consumption in real-time. This work proposes the development of a complete system for this purpose, from the measurement hardware to data visualization. The implemented solution consists of a prototype based on the ESP32 microcontroller, which collects voltage and current data and transmits them securely via the MQTT protocol. The cloud infrastructure was architected as a robust data pipeline, using a Telegraf agent, running in a Docker container, to collect the messages and persist them in a time-series database (InfluxDB). Finally, a visualization platform (Grafana) presents the data to the user in an interactive dashboard. The project focuses on accessibility, low cost, and scalability, demonstrating the feasibility of a professional architecture with open-source tools.

Keywords: Internet of Things, energy monitoring, smart home, energy efficiency, time series.

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

AC	Corrente Alternada (do inglês, Alternating Current)
ADC	Conversor Analógico-Digital (do inglês, Analog-to-Digital Converter)
CLI	Interface de Linha de Comando (do inglês, Command-Line Interface)
DC	Corrente Contínua (do inglês, Direct Current)
EPM	Erro Percentual Médio
ESP	Refere-se a uma família de microcontroladores da Espressif Systems
GPIO	Entrada/Saída de Propósito Geral (do inglês, General-Purpose Input/Output)
IDE	Ambiente de Desenvolvimento Integrado (do inglês, Integrated Development Environment)
IoT	Internet das Coisas (do inglês, Internet of Things)
MQTT	Message Queuing Telemetry Transport
PCB	Placa de Circuito Impresso (do inglês, Printed Circuit Board)
RMS	Valor Eficaz (do inglês, Root Mean Square)
TLS	Segurança da Camada de Transporte (do inglês, Transport Layer Security)
VA	Volt-Ampère (unidade de potência aparente)
W	Watt (unidade de potência ativa)
Wi-Fi	Wireless Fidelity

Sumário

Capítulo 1

Introdução

1.1 Considerações Iniciais

A gestão do consumo de energia elétrica é um dos desafios centrais da sociedade contemporânea. Globalmente, o setor de edificações (residenciais e comerciais) é responsável por cerca de um terço do consumo final de energia e aproximadamente um quarto das emissões de CO_2, destacando a urgência por soluções que promovam a eficiência energética (International Energy Agency, 2024). No Brasil, o cenário é igualmente relevante, com dados do Balanço Energético Nacional indicando um crescimento contínuo do consumo no setor residencial, impulsionado pela digitalização dos lares (Empresa de Pesquisa Energética, 2025). Nesse contexto, os avanços recentes na área da Internet das Coisas (IoT) e das casas inteligentes (*smart homes*) surgem como uma solução promissora. Essas tecnologias permitem a criação de sistemas de baixo custo capazes de monitorar o consumo em tempo real, fornecendo aos moradores dados claros para uma gestão de energia mais eficiente e sustentável. É nesta interseção entre a demanda por eficiência energética e a oportunidade tecnológica que este trabalho se insere.

1.2 Objetivos

Este trabalho tem como objetivo desenvolver um sistema de monitoramento de energia para residências inteligentes, baseado em tecnologias de IoT. A proposta é criar uma solução

de baixo custo, acessível e escalável, que permita aos usuários acompanhar seu consumo energético, identificar padrões de uso e, assim, promover uma maior consciência energética.

Além disso, busca-se contribuir para a disseminação de tecnologias sustentáveis no ambiente doméstico e fomentar a adoção de práticas mais responsáveis no consumo de energia elétrica.

Ressalta-se que o escopo deste projeto concentra-se na camada de sensoriamento e visualização (telemetria) da arquitetura de uma casa inteligente. O sistema proposto atua de forma passiva, fornecendo dados para a tomada de decisão humana, não contemplando, nesta etapa, a atuação automática ou controle direto de cargas.

1.3 Justificativa e Motivação

A relevância deste trabalho se justifica pela crescente necessidade de ferramentas que promovam o uso consciente de energia elétrica em residências, onde muitas famílias desconhecem seus próprios padrões de consumo. A abordagem baseada em Internet das Coisas (IoT) é particularmente pertinente devido ao avanço tecnológico e à popularização de dispositivos de baixo custo, como sensores e microcontroladores. Tais tecnologias tornam as soluções de monitoramento, antes restritas a ambientes industriais, tecnicamente viáveis e economicamente acessíveis para o consumidor doméstico.

A principal motivação por trás deste projeto é a preocupação com a sustentabilidade e com os custos associados ao desperdício energético. A oportunidade de utilizar os conhecimentos da Ciência da Computação para desenvolver uma ferramenta prática, que traduza dados brutos em informações úteis para o usuário, serviu como um impulso fundamental. O objetivo é contribuir com uma solução que alinhe a inovação tecnológica às necessidades de uma sociedade cada vez mais digital e ambientalmente responsável.

1.4 Metodologia

O desenvolvimento do presente trabalho seguirá uma metodologia de pesquisa aplicada com a construção de um protótipo funcional, dividida em três fases principais: fundamenta-

ção, desenvolvimento e validação.

A fase de fundamentação consistiu em uma revisão bibliográfica para a seleção das tecnologias que compõem a arquitetura do sistema. Esta análise resultou na escolha do microcontrolador ESP32 como unidade de processamento e dos sensores ACS712 e ZMPT101B para a aquisição de dados. Para a infraestrutura de nuvem, foi projetado um pipeline de dados completo, utilizando o protocolo MQTT para a comunicação, o broker HiveMQ como intermediário de mensagens, um agente coletor Telegraf (executado em um contêiner Docker na plataforma Railway) para a integração, e as plataformas InfluxDB e Grafana para o armazenamento e visualização dos dados, respectivamente.

A fase de desenvolvimento abrangerá a montagem do circuito eletrônico em placa de circuito perfurada e o desenvolvimento do firmware em `C++` no ambiente Arduino IDE, que será responsável pela leitura dos sensores, cálculo de potência e transmissão dos dados via Wi-Fi. Finalmente, a fase de validação compreenderá a execução de um plano de testes controlado, no qual as medições do protótipo serão comparadas com as de um medidor de energia comercial de referência para aferir a sua precisão através de métricas estatísticas, como o Erro Percentual Médio.

1.5 Organização do Trabalho

Este trabalho está estruturado em cinco capítulos, de modo a apresentar a pesquisa de forma clara e sequencial. O Capítulo 2 apresenta a fundamentação teórica que sustenta o projeto, abordando desde os conceitos de Internet das Coisas e casas inteligentes até as especificidades das tecnologias de hardware, software e comunicação selecionadas, concluindo com uma análise do estado da arte através de trabalhos correlatos. O Capítulo 3 detalha a metodologia de desenvolvimento, descrevendo a arquitetura do sistema proposto, os procedimentos para a montagem do circuito e desenvolvimento do firmware, e o plano para a validação experimental. Posteriormente, o Capítulo 4 expõe os resultados obtidos, incluindo a apresentação do protótipo funcional, a análise dos dados de consumo coletados e a validação da precisão do sistema. Por fim, o Capítulo 5 encerra o trabalho com a conclusão, onde são sintetizadas as contribuições da pesquisa, discutidas as limitações do protótipo e propostas

direções para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta a revisão da literatura que fundamenta este trabalho. Inicialmente, são abordados os conceitos teóricos essenciais para a compreensão do sistema proposto, incluindo as tecnologias de base e suas arquiteturas. Em seguida, será realizada uma análise dos trabalhos correlatos mais relevantes, situando o presente projeto no contexto do estado da arte em monitoramento de energia e casas inteligentes.

2.1 Internet das Coisas

A Internet das Coisas, ou IoT (do inglês, *Internet of Things*), refere-se à rede de objetos físicos, as chamadas "coisas", equipados com sensores, software e outras tecnologias com o propósito de se conectar e trocar dados com outros dispositivos e sistemas pela internet. A visão por trás da IoT é a de um mundo onde o ambiente físico é digitalizado por meio de sensores onipresentes, permitindo a coleta e a troca de dados para a criação de serviços inteligentes e contextuais (SANTOS et al., 2016).

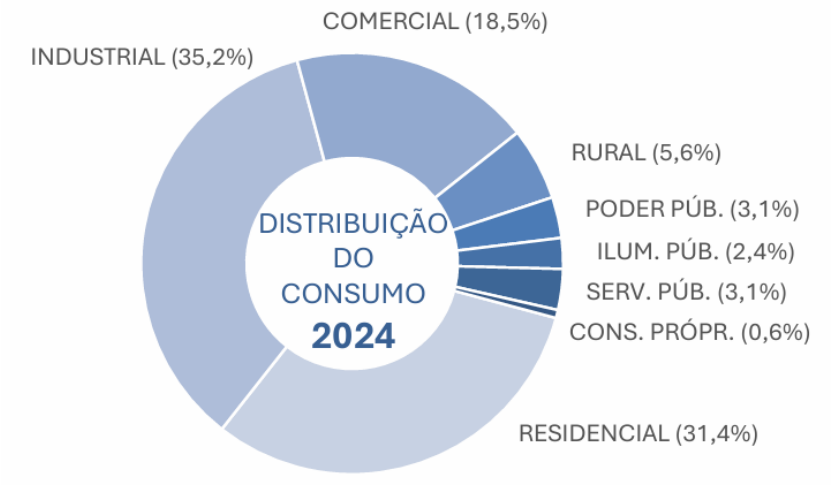
A arquitetura de um sistema IoT é frequentemente descrita em camadas para gerenciar a complexidade da coleta, transmissão e processamento de dados. Na base, a camada de percepção é composta por sensores, responsáveis por coletar dados do ambiente físico, e atuadores, que realizam ações no mundo real. Esses dispositivos, como microcontroladores, são então conectados por meio de uma camada de rede, utilizando tecnologias de comunicação sem fio para enviar as informações a plataformas na nuvem.

Essa integração entre hardware e software permite que objetos cotidianos se tornem "inteligentes", possibilitando que usuários monitorem ambientes e controlem sistemas remotamente. O impacto da IoT transcende a simples conectividade, habilitando aplicações transformadoras em setores como saúde, indústria, agricultura e, de forma central para este trabalho, a automação residencial. O verdadeiro valor da tecnologia reside na capacidade de converter o grande volume de dados coletados em informações e *insights* acionáveis, permitindo a otimização de processos e a tomada de decisões mais inteligentes (MASCHIETTO et al., 2021).

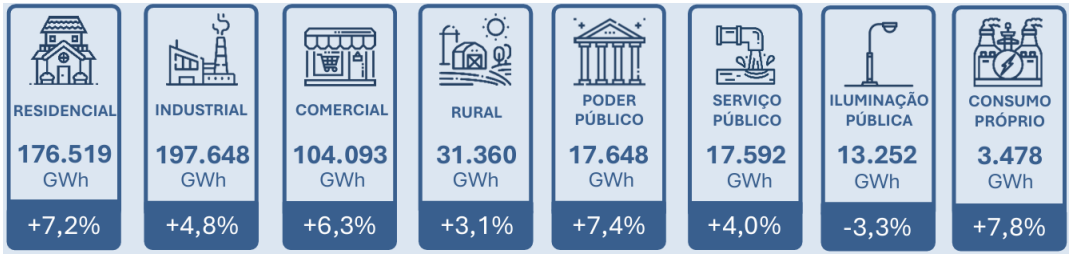
2.2 Casas Inteligentes e Eficiência Energética

A consolidação da Internet das Coisas, conforme discutido na seção anterior, serve como a infraestrutura tecnológica fundamental que viabiliza o conceito de casas inteligentes. Uma casa inteligente, ou *smart home*, transcende a simples automação de tarefas, representando um ecossistema de dispositivos interconectados que coletam dados do ambiente e atuam sobre ele, visando otimizar o conforto, a segurança e, de forma central para este trabalho, a sustentabilidade dos recursos energéticos. A transição de uma residência convencional para uma casa inteligente é, em sua essência, a aplicação direta dos princípios da IoT no ambiente doméstico, onde objetos do cotidiano adquirem capacidade computacional e de comunicação (Associação Brasileira da Indústria Elétrica e Eletrônica, 2019).

A relevância desta aplicação torna-se evidente ao analisar o panorama do consumo energético nacional. O setor residencial brasileiro representa uma parcela significativa da matriz de consumo elétrico, com uma tendência de crescimento contínuo. Segundo a Empresa de Pesquisa Energética (EPE), o consumo de energia elétrica nas residências brasileiras atingiu 176.519 GWh em 2024, um aumento de 7,2% em relação ao ano anterior. Este valor representa 31,4% de todo o consumo elétrico do país, conforme detalhado na Figura 2.1 (Empresa de Pesquisa Energética, 2025).



(a) Distribuição percentual do consumo de energia elétrica por classe em 2024.



(b) Consumo total (GWh) e variação anual por classe em 2024.

Figura 2.1: Consumo de energia elétrica no Brasil por classe (2024).

Fonte: Adaptado de EPE (2025).

É neste contexto que as casas inteligentes surgem não como um luxo, mas como uma ferramenta estratégica para a eficiência energética. A dissertação de Silva (2021), por exemplo, destaca que a falta de informação clara e em tempo real é um dos maiores obstáculos para a mudança de comportamento dos consumidores, que muitas vezes só percebem o impacto de seus hábitos na fatura mensal, quando a oportunidade de agir já passou (SILVA, 2021). A principal contribuição das casas inteligentes para a eficiência energética reside na sua capacidade de transformar dados brutos de consumo em *insights* acionáveis para o morador. Esta transformação ocorre através de três mecanismos principais, habilitados pela tecnologia IoT: monitoramento, controle e automação (MAHBUB et al., 2020).

1. **Monitoramento em Tempo Real:** Diferentemente da medição agregada e mensal fornecida pelas concessionárias, os sistemas de uma casa inteligente permitem a desagre-

gação do consumo em nível de aparelho ou circuito. Ao fornecer dados granulados e instantâneos, o morador adquire consciência sobre seus padrões de uso, identificando facilmente os chamados "vilões energéticos", equipamentos com alto consumo ou que operam de forma ineficiente, como dispositivos em modo *standby* (SILVA, 2021).

2. **Controle Remoto e Ativo:** A conectividade intrínseca à IoT permite que os moradores controlem seus aparelhos remotamente. Essa funcionalidade vai além da conveniência, permitindo ações de economia como desligar luzes ou aparelhos de ar condicionado que foram esquecidos ligados, otimizando o consumo de forma ativa e mitigando o desperdício (Associação Brasileira da Indústria Elétrica e Eletrônica, 2019).

3. **Automação Inteligente:** Este é o nível mais avançado de gestão energética. A casa inteligente pode tomar decisões autônomas com base em regras pré-definidas ou em aprendizado de padrões. Exemplos incluem o ajuste automático da intensidade da iluminação artificial com base na luz natural disponível, a otimização do ciclo de funcionamento do ar condicionado de acordo com a presença de pessoas no ambiente ou até mesmo a gestão da recarga de veículos elétricos para horários de tarifa reduzida (MAHBUB et al., 2020).

Portanto, a aplicação da IoT para a gestão de energia em casas inteligentes estabelece um ciclo virtuoso: os dados coletados pelo monitoramento informam as ações de controle e alimentam os sistemas de automação, que por sua vez otimizam o consumo e geram novos dados. A efetivação deste ciclo, contudo, depende de um conjunto específico de componentes de hardware e software capazes de realizar a medição, a comunicação e o processamento destas informações. A análise detalhada destas ferramentas será o foco da próxima seção.

2.3 Tecnologias para Monitoramento de Energia em IoT

Uma vez estabelecido o potencial das casas inteligentes para a promoção da eficiência energética, conforme discutido na seção anterior, torna-se fundamental detalhar os componentes que formam a espinha dorsal de um sistema de monitoramento. Esta seção, portanto,

apresentará o "arsenal tecnológico" necessário para a construção do protótipo proposto neste trabalho. A análise será dividida em subsecções que abordam cada camada da arquitetura, iniciando pela unidade central de processamento (as plataformas de hardware), seguindo para os dispositivos responsáveis pela coleta de dados (os sensores) e, por fim, detalhando os protocolos e plataformas que viabilizam a comunicação e a visualização das informações.

2.3.1 Plataformas de Hardware e a Escolha do Microcontrolador

A materialização de um sistema IoT, conforme o "arsenal tecnológico" introduzido anteriormente, inicia-se com a seleção de sua unidade central de processamento. No contexto de sistemas embarcados de baixo custo, esta unidade é tipicamente um microcontrolador ou um microcomputador de placa única. A escolha da plataforma de hardware é uma das decisões mais críticas do projeto, pois ela dita as capacidades de processamento, as opções de conectividade e o consumo energético de todo o sistema. O mercado atual oferece uma vasta gama de opções, contudo, três plataformas destacam-se pela sua popularidade, vasta documentação e robusta comunidade de desenvolvedores: Arduino, a família ESP e o Raspberry Pi. A seguir, será apresentada uma análise comparativa destas plataformas, a fim de fundamentar a seleção da mais adequada para os requisitos deste trabalho.

O Arduino, introduzido em 2005, revolucionou a prototipagem eletrônica ao oferecer uma plataforma de hardware e software de código aberto e de fácil utilização. Modelos como o Arduino Uno, conforme ilustrado na Figura 2.2, são construídos em torno de microcontroladores da família ATMega da Atmel e são célebres pela sua simplicidade e pela vasta quantidade de bibliotecas e *shields* (placas de expansão) disponíveis, o que acelera o desenvolvimento de projetos para iniciantes (SILVA et al., 2019). No entanto, a sua principal limitação para aplicações de IoT reside na ausência de conectividade de rede nativa. Para conectar um Arduino à internet, é necessário o uso de *shields* adicionais de Wi-Fi ou Ethernet, o que aumenta o custo, o tamanho e a complexidade do projeto (JUNIOR; FARINELLI, 2018).

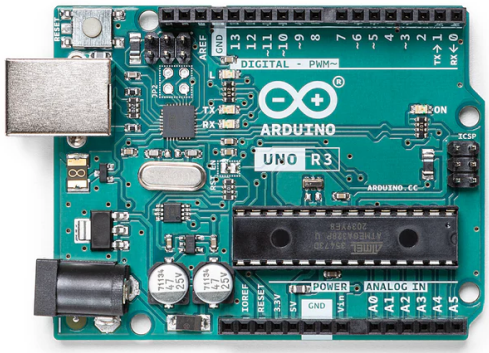


Figura 2.2: Plataforma Arduino Uno R3

Fonte: Arduino.cc (2025).

Em contrapartida, a família ESP, desenvolvida pela fabricante chinesa Espressif Systems, surgiu como uma solução projetada especificamente para a era da Internet das Coisas. O microcontrolador ESP8266, lançado em 2014, e o seu sucessor mais poderoso, o ESP32, integram conectividade Wi-Fi e, no caso do ESP32, também Bluetooth, no próprio chip. Esta integração nativa, visível na placa de desenvolvimento da Figura 2.3, representa uma vantagem monumental, pois simplifica drasticamente o design do hardware e reduz o custo final do dispositivo (MONK, 2019). Além disso, o ESP32 possui um poder de processamento superior ao de um Arduino Uno padrão, com um processador dual-core, e pode ser programado utilizando a mesma Arduino IDE, aproveitando o ecossistema e a familiaridade já estabelecidos pela plataforma Arduino (JUNIOR; FARINELLI, 2018).

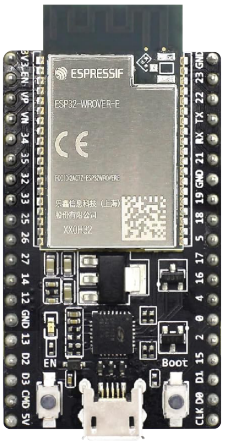


Figura 2.3: Placa de desenvolvimento ESP32 DevKitC

Fonte: Espressif (2025).

Num patamar de desempenho superior, encontra-se o Raspberry Pi. Diferentemente do Arduino e do ESP32, o Raspberry Pi não é um microcontrolador, mas sim um microcomputador de placa única (SBC - Single-Board Computer) plenamente funcional. Ele executa um sistema operacional completo, como o Linux, e possui um poder de processamento e uma capacidade de memória RAM muito superiores, comparáveis aos de um computador de entrada (SILVA et al., 2019). A complexidade e a variedade de conectores de um modelo como o Raspberry Pi 4, visto na Figura 2.4, o tornam uma ferramenta extremamente poderosa, capaz de atuar como um servidor completo, mas seu uso como um simples nó de sensor para a coleta de dados de consumo energético seria um subaproveitamento de seus recursos. Adicionalmente, o seu custo é significativamente mais elevado e o seu consumo de energia é muito maior que o de um microcontrolador, tornando-o menos ideal para uma aplicação embarcada que visa o baixo custo e a eficiência (MONK, 2019).

Figura 2.4: Microcomputador Raspberry Pi 4

Fonte: Raspberry Pi Foundation (2025).

Diante desta análise comparativa, a escolha do ESP32 como plataforma de hardware para este projeto é justificada de forma conclusiva. O Arduino, apesar da sua simplicidade, impõe a penalidade de custo e complexidade pela necessidade de módulos de rede externos. O Raspberry Pi, por sua vez, é superdimensionado e de custo elevado para os requisitos de coleta de dados deste trabalho. O ESP32, portanto, emerge como a solução de melhor custo-benefício: ele oferece a conectividade Wi-Fi nativa essencial para um dispositivo IoT, um poder de processamento mais do que suficiente para os cálculos de energia e a conveniência de ser programável através do ambiente familiar da Arduino IDE. Conforme apontam Junior e Farinelli (2018), a família ESP representa a evolução natural das plataformas de prototipagem para projetos que demandam conectividade, tornando-a a escolha técnica mais acertada para o desenvolvimento deste medidor inteligente (JUNIOR; FARINELLI, 2018). Com o "cérebro" do sistema definido, o próximo passo é selecionar os "sentidos" do nosso medidor: os sensores responsáveis por coletar os dados brutos de tensão e corrente da rede elétrica, o que será detalhado na seção a seguir.

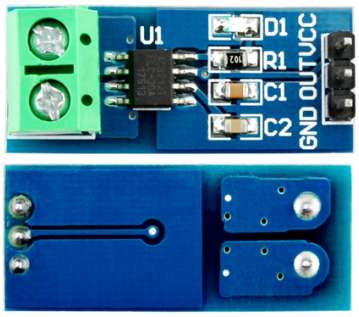
2.3.2 Sensores para Medição de Energia

Uma vez definido o "cérebro" do sistema na seção anterior, a materialização do medidor inteligente exige a seleção de seus "sentidos": os componentes responsáveis por traduzir as grandezas físicas da rede elétrica em sinais que o microcontrolador possa processar. A aferição do consumo de energia (medido em kWh) baseia-se na medição contínua de duas

grandezas fundamentais: a tensão (em Volts) e a corrente (em Ampères), uma vez que a potência instantânea (em Watts) é o produto direto entre elas (SOUZA, 2020). A seleção de sensores adequados para estas duas medições é, portanto, um passo crítico que impacta diretamente a precisão, a segurança e o custo do protótipo.

Para a medição da corrente, optou-se por uma abordagem invasiva, que consiste na inserção de um sensor diretamente no circuito da carga a ser monitorada. O componente selecionado para esta finalidade foi o sensor de corrente ACS712, ilustrado na Figura 2.5. Este circuito integrado utiliza o Efeito Hall como princípio de funcionamento: um campo magnético, gerado pela corrente que atravessa um condutor de cobre interno ao chip, induz uma diferença de potencial em um transdutor Hall, resultando em uma tensão de saída analógica (Allegro MicroSystems, 2024). A principal vantagem desta tecnologia é a geração de uma tensão de saída linearmente proporcional à corrente de entrada, tanto para correntes alternadas (AC) quanto contínuas (DC), o que simplifica significativamente a sua calibração e leitura por parte do microcontrolador (Allegro MicroSystems, 2024). Conforme destacam estudos na área de sistemas embarcados para monitoramento, a utilização de sensores baseados em Efeito Hall como o ACS712 é uma estratégia consolidada e de baixo custo, oferecendo uma relação custo-benefício adequada para o desenvolvimento de medidores inteligentes em aplicações de monitoramento residencial (NASCIMENTO; SOUZA; GOMES, 2021).

Figura 2.5: Módulo sensor de corrente ACS712



Fonte: UsinaInfo (2025).

Para a medição da tensão, o desafio consiste em reduzir os altos valores da rede (127V/220V) para a faixa de operação do ESP32 (0-3.3V), garantindo o isolamento galvânico entre os circuitos. O módulo sensor de tensão ZMPT101B, apresentado na Figura 2.6, foi o componente

selecionado. Este módulo utiliza um transformador de potencial de alta precisão para isolar e reduzir a tensão da rede de forma segura (InnovatorsGuru, 2020; ABUBAKAR et al., 2017). A sua principal vantagem é a inclusão de um circuito de condicionamento de sinal na própria placa, o que simplifica o projeto eletrônico. Estudos de calibração para este sensor, como o realizado por Abubakar et al. (2017), demonstram que ele possui boa consistência e alcança um erro de medição inferior a 2.5% para tensões de rede, uma precisão considerada excelente para o escopo de aplicações de monitoramento residencial (ABUBAKAR et al., 2017).

Figura 2.6: Módulo sensor de tensão ZMPT101B

Fonte: Eletrogate (2022)

2.3.3 Protocolos de Comunicação para IoT

A seleção de uma plataforma de hardware robusta e de sensores precisos, conforme detalhado nas seções anteriores, resolve a etapa de aquisição de dados. Contudo, para que um dispositivo embarcado se torne verdadeiramente parte da Internet das Coisas, é essencial definir o método pelo qual ele irá transmitir essas informações para a nuvem. A escolha do protocolo de comunicação é, portanto, um pilar fundamental da arquitetura, impactando diretamente a eficiência, a confiabilidade e a escalabilidade do sistema. No universo de IoT, onde os dispositivos frequentemente operam com recursos computacionais limitados e em redes que podem ser instáveis, protocolos leves e eficientes são imperativos (SANTOS et al., 2016). Dentre as opções disponíveis, o protocolo MQTT (*Message Queuing Telemetry Transport*) consolidou-se como o padrão de facto para esta finalidade.

O MQTT é um protocolo de mensagens leve, baseado no modelo *publish/subscribe* (publicar/subscrever), que opera sobre a pilha TCP/IP. Diferentemente do modelo requisição/resposta do HTTP, o MQTT desacopla os produtores de informação (publicadores) dos consumidores (subscritores) através de um elemento central chamado Broker (MASCHIETTO et al., 2021). O funcionamento é análogo a um sistema de rádio: o publicador (neste caso, o ESP32) envia a sua mensagem para o Broker, endereçada a um "canal" específico, chamado de tópico. Os subscritores (como a plataforma de visualização) sintonizam-se nos tópicos de seu interesse para receber as mensagens, sem que as partes precisem de se conhecer diretamente.

A adequação do MQTT para o presente trabalho é justificada por três características técnicas principais:

1. **Leveza (*Low Overhead*):** O cabeçalho das mensagens MQTT é extremamente pequeno (apenas 2 bytes no mínimo), o que reduz drasticamente o consumo de banda de rede. Esta eficiência é crucial para dispositivos embarcados que precisam de enviar atualizações frequentes sem sobrecarregar a rede Wi-Fi residencial (SANTOS et al., 2016).

2. **Qualidade de Serviço (QoS):** O protocolo define três níveis de Qualidade de Serviço, que são acordos de garantia sobre a entrega das mensagens. O nível QoS 1, por exemplo, garante que a mensagem chegue ao destino pelo menos uma vez, oferecendo um excelente balanço entre confiabilidade e desempenho para dados de monitoramento, sendo este o nível adotado neste projeto (MASCHIETTO et al., 2021).

3. **Suporte à Segurança:** A segurança é um pilar não negociável em sistemas IoT. O protocolo MQTT foi projetado para operar sobre camadas de transporte seguras. A sua integração com o TLS (*Transport Layer Security*) permite a criação de um canal de comunicação totalmente criptografado entre o dispositivo e o Broker, garantindo a confidencialidade e a integridade dos dados. Adicionalmente, o próprio protocolo suporta a autenticação de clientes através de utilizador e senha, assegurando que apenas dispositivos autorizados possam conectar-se ao Broker (MORAES; HAYASHI, 2021).

Desta forma, a adoção do MQTT não é apenas uma escolha de conveniência, mas uma decisão de engenharia fundamentada na eficiência, confiabilidade e segurança do protocolo.

Com a comunicação devidamente estabelecida, a próxima etapa da arquitetura é definir como as informações transmitidas serão armazenadas de forma persistente para permitir a análise histórica, o que nos leva à discussão sobre os bancos de dados.

2.3.4 Infraestrutura de Nuvem para Coleta e Persistência de Dados

Uma vez que os dados são publicados pelo dispositivo de campo via MQTT, a arquitetura de nuvem necessita de um mecanismo para receber essas mensagens e persisti-las de forma confiável em um banco de dados. Para este fim, uma abordagem moderna e escalável consiste no uso de um *pipeline* composto por um agente coletor, tecnologia de conteinerização e uma plataforma de hospedagem.

O Telegraf é um agente de coleta de métricas de código aberto, mantido pela InfluxData, projetado para ser leve e modular. Sua função em uma arquitetura de IoT é atuar como uma ponte robusta, subscrevendo a fontes de dados, como um broker MQTT, e escrevendo as informações em diversos destinos (*outputs*), como bancos de dados de séries temporais (PEREIRA, 2021). A sua escolha se justifica pela integração nativa com o InfluxDB e pela sua alta performance.

Para garantir que um serviço como o Telegraf opere de forma contínua e isolada na nuvem, utiliza-se a tecnologia de Docker. O Docker permite "empacotar" uma aplicação e todas as suas dependências em uma unidade padronizada para desenvolvimento de software, chamada de contêiner (SANTANA, 2020). Esta abordagem resolve o problema de "na minha máquina funciona", garantindo que o software se comporte de maneira idêntica em qualquer ambiente, desde o desenvolvimento local até a produção na nuvem.

Finalmente, para hospedar o contêiner Docker, foram selecionadas as Plataformas como Serviço (PaaS), como o Railway. Estas plataformas abstraem a complexidade do gerenciamento de servidores, permitindo que o desenvolvedor se concentre apenas no código. O Railway, em particular, integra-se diretamente com o Docker, automatizando o processo de *build* e *deploy* de um contêiner a partir de um `Dockerfile`, o que o torna uma escolha estratégica para a implantação rápida e contínua de serviços de back-end (Railway, 2025).

2.3.5 Bancos de Dados de Séries Temporais

Com a comunicação dos dados de consumo devidamente estabelecida através do protocolo MQTT, a próxima etapa da arquitetura de um sistema de monitoramento robusto é o seu armazenamento persistente. A simples visualização de dados em tempo real é útil, mas o verdadeiro valor de um sistema de monitoramento reside na sua capacidade de criar um histórico, permitindo a análise de tendências, a identificação de padrões de consumo e a geração de relatórios ao longo do tempo. A escolha de um sistema de gerenciamento de banco de dados (SGBD) adequado é, portanto, crucial. No entanto, o volume, a velocidade e a natureza dos dados gerados por sensores de IoT impõem desafios que bancos de dados relacionais tradicionais, como o MySQL ou PostgreSQL, não foram originalmente projetados para resolver de forma otimizada (SANTANA, 2020).

Bancos de dados relacionais são excelentes para armazenar dados transacionais, mas para dados de IoT, que consistem em um fluxo incessante de medições indexadas pelo tempo (ex: “às 14:30:01, a potência era de 150.2W; às 14:30:02, era de 151.1W...”), o principal desafio é a alta taxa de escrita (*ingestão*) e a eficiência das consultas baseadas em intervalos de tempo. Conforme aponta Santana (2020), o uso de um SGBD relacional para esta finalidade pode levar a problemas de desempenho à medida que a tabela de medições cresce para milhões ou milhares de milhões de registos (SANTANA, 2020).

Para solucionar este problema, surgiu uma categoria especializada de bancos de dados NoSQL conhecida como Bancos de Dados de Séries Temporais (*Time-Series Databases* - TSDB). Estas plataformas são projetadas desde a sua fundação para uma única tarefa: gerir de forma extremamente eficiente dados que são medidos ao longo do tempo (InfluxData, 2023). Elas utilizam mecanismos internos de armazenamento e indexação otimizados para o tempo, o que resulta num desempenho de escrita e de consulta muito superior para este tipo de carga de trabalho. Para este projeto, foi selecionado o InfluxDB, um dos TSDBs de código aberto mais populares e amplamente adotados no mercado. A sua escolha é justificada por um conjunto de características técnicas distintivas.

Primeiramente, destaca-se a sua alta performance de escrita e compactação de dados. O InfluxDB foi construído para lidar com centenas de milhares de escritas por segundo e uti-

liza algoritmos de compactação de dados altamente eficientes, reduzindo significativamente o espaço de armazenamento necessário para os dados históricos (InfluxData, 2023).

Em segundo lugar, a plataforma possui uma linguagem de consulta especializada (Flux ou InfluxQL) que inclui funções nativas para manipulação de dados temporais. Funções como agregações (médias, somas, máximos) em janelas de tempo, transformações e outras operações seriam muito complexas de realizar em SQL padrão, mas são simplificadas no InfluxDB (InfluxData, 2023).

Finalmente, o ecossistema e a integração do InfluxDB são um diferencial decisivo. A plataforma integra-se nativamente com as principais ferramentas do ecossistema de monitoramento, incluindo um "subscriber" MQTT (o Telegraf) e a plataforma de visualização Grafana. Conforme Santana (2020), esta sinergia simplifica enormemente a construção da arquitetura de software e consolida a sua posição como uma escolha de excelência para sistemas de monitoramento (SANTANA, 2020). A adoção do InfluxDB, portanto, representa uma decisão técnica que alinha o projeto com as melhores práticas de mercado para sistemas de IoT, garantindo que o sistema seja não apenas funcional, mas também escalável e performático. Com os dados agora devidamente armazenados, a etapa final da fundamentação teórica é apresentar a ferramenta que irá consumi-los e transformá-los em informação visualmente compreensível.

2.3.6 Plataformas de Visualização de Dados

A etapa final na arquitetura de um sistema de monitoramento de IoT é a transformação dos dados brutos armazenados em informação visualmente compreensível e útil para o utilizador final. Um sistema que apenas coleta e armazena dados sem uma interface de visualização eficaz falha no seu objetivo principal de fornecer *insights* acionáveis. Enquanto o desenvolvimento de uma aplicação web ou móvel customizada é uma abordagem possível, ela representa um esforço de engenharia de software considerável e que foge ao escopo central deste trabalho. Uma estratégia mais eficiente e alinhada com as práticas da indústria de monitoramento de sistemas é a utilização de plataformas de *dashboarding* especializadas (PEREIRA, 2021).

Estas plataformas são ferramentas de *software* projetadas para se conectar a diversas fon-

tes de dados e apresentar as informações em painéis interativos, ou *dashboards*, compostos por gráficos, medidores, tabelas e alertas. Para este projeto, foi selecionada a plataforma de código aberto **Grafana**, que se consolidou como a ferramenta líder de mercado para a visualização de dados de séries temporais, especialmente em aplicações de monitoramento de infraestrutura, sistemas e IoT (Grafana Labs, 2023).

A escolha do Grafana é justificada pela sua sinergia com os outros componentes selecionados para este projeto. A sua principal vantagem é a integração nativa e otimizada com o banco de dados InfluxDB, permitindo a criação de consultas complexas diretamente na sua interface para a manipulação e apresentação dos dados históricos de consumo energético (PEREIRA, 2021). Além disso, o Grafana oferece uma vasta biblioteca de painéis de visualização (gráficos de linha, medidores, histogramas, etc.) que podem ser configurados de forma intuitiva através de uma interface gráfica, abstraindo a complexidade do desenvolvimento *front-end*. Conforme a sua documentação oficial, a filosofia do Grafana é "democratizar as métricas", permitindo que os utilizadores criem *dashboards* ricos e informativos sem a necessidade de conhecimento avançado em programação (Grafana Labs, 2023).

A utilização do Grafana, portanto, eleva a qualidade da apresentação dos resultados a um padrão profissional, permitindo a criação de uma interface de monitoramento robusta e alinhada com as melhores práticas da indústria.

2.4 Segurança e Privacidade em Sistemas IoT

A interconexão de dispositivos no ambiente doméstico, embora traga inúmeros benefícios de conforto e eficiência, introduz inevitavelmente novas vulnerabilidades e desafios relacionados à segurança da informação e à privacidade dos moradores. Um sistema de IoT, por mais simples que seja, constitui uma "superfície de ataque" que pode ser explorada por agentes mal-intencionados se não for devidamente protegido (MORAES; HAYASHI, 2021). Conforme alertam Moraes e Hayashi (2021), a segurança não pode ser tratada como um recurso adicional, mas sim como um requisito fundamental e intrínseco ao projeto de qualquer solução de IoT.

A arquitetura de um sistema como o proposto neste trabalho apresenta três pontos princi-

pais de vulnerabilidade. O primeiro é o próprio dispositivo embarcado (o ESP32), que pode ser suscetível a ataques físicos ou a ataques lógicos caso utilize senhas fracas. O segundo ponto é a camada de comunicação, onde os dados trafegam pela rede. Sem a devida criptografia, um atacante na mesma rede poderia "escutar" a comunicação e interceptar os dados de consumo. O terceiro ponto é a plataforma de nuvem (o *Broker* MQTT e o banco de dados), que, se mal configurada, poderia permitir o acesso não autorizado aos dados históricos (PINHEIRO; MARTINS; GUEDES, 2019).

Além dos riscos de segurança, as implicações de privacidade são igualmente críticas. Os dados de consumo energético, embora pareçam inócuos, são uma fonte riquíssima de informação sobre os hábitos dos moradores. Conforme destaca a pesquisa de Pinheiro et al. (2019), a análise destes dados pode revelar os horários em que a casa está vazia, os padrões de sono da família e até mesmo inferir o número de pessoas que vivem na residência. A proteção destes dados não é apenas uma questão técnica, mas também ética, para garantir que informações tão íntimas não sejam acedidas ou utilizadas indevidamente (PINHEIRO; MARTINS; GUEDES, 2019).

Diante destes desafios, o presente projeto adota estratégias de mitigação alinhadas com as boas práticas de segurança para IoT. A comunicação foi estabelecida utilizando o protocolo MQTT sobre a camada de transporte TLS (*Transport Layer Security*), conforme detalhado na seção 2.3.3. No entanto, por limitações do ambiente *cloud* gratuito e do microcontrolador ESP32, optou-se pela implementação do TLS sem validação de Autoridade Certificadora (CA). Complementarmente, o acesso ao *broker* é protegido por autenticação com utilizador e credenciais fixas, assegurando que apenas o dispositivo autorizado publique dados. Estas medidas constituem a base de segurança possível dentro do escopo do protótipo (MORAES; HAYASHI, 2021).

2.5 Trabalhos Correlatos (Estado da Arte)

Para contextualizar o presente projeto e identificar suas contribuições originais, foi realizada uma análise de trabalhos acadêmicos recentes que abordam o desenvolvimento de medidores de energia inteligentes baseados em IoT. A investigação da literatura revela uma diver-

sidade de abordagens arquitetônicas, que variam em complexidade, custo e funcionalidade. Nesta seção, serão analisados três trabalhos representativos dessas diferentes abordagens, a fim de estabelecer o estado da arte e posicionar a solução aqui proposta.

O trabalho de Nolêto (2023) apresenta o desenvolvimento de um medidor inteligente com uma arquitetura focada no acesso local. O autor utiliza um microcontrolador ESP32 e desenvolve um firmware customizado para realizar a leitura dos sensores de tensão e corrente, demonstrando um rigoroso processo de calibração. Contudo, a visualização dos dados é implementada através de um servidor web embarcado no próprio ESP32, o que restringe o acesso às informações à rede Wi-Fi local (NOLêTO, 2023). Embora seja uma solução de baixo custo e eficaz como prova de conceito, o próprio autor aponta, em suas considerações finais, a necessidade de evoluir o sistema para uma arquitetura com um servidor em nuvem que permita o armazenamento de dados e o acesso remoto.

Em uma abordagem distinta, Belo (2023) propõe um sistema que já incorpora a comunicação em nuvem. O autor utiliza um hardware similar, com um microcontrolador da família ESP e um sensor de corrente SCT-013, implementando a comunicação via protocolo MQTT com um broker na nuvem (HiveMQ). A principal diferença metodológica reside no software embarcado, pois, em vez de desenvolver um firmware próprio, o autor opta por instalar e configurar o firmware de código aberto Tasmota. A visualização dos dados é realizada através de um aplicativo de celular genérico para painéis MQTT (BELO, 2023). Esta estratégia acelera o desenvolvimento, mas abstrai a complexidade da programação de baixo nível do dispositivo e resulta em uma interface de usuário menos robusta que as plataformas de visualização especializadas.

Uma terceira linha de pesquisa, representada pelo trabalho de Santos et al. (2023), foca no uso de plataformas de IoT “tudo-em-um”. Neste artigo, os autores desenvolvem um dispositivo com ESP-32 e o integram à plataforma Blynk. Esta plataforma simplifica a arquitetura ao prover, em um único serviço, o broker, o armazenamento de dados e a interface de visualização, que é configurada em um aplicativo de celular. A solução é funcional e de rápida implementação, demonstrando o controle e o monitoramento de uma carga. Esta abordagem, embora excelente para prototipagem rápida, pode apresentar limitações em termos de

flexibilidade e da granularidade no controle do armazenamento e análise dos dados, quando comparada a uma arquitetura que utiliza serviços especializados (SANTOS et al., 2023).

A análise destes trabalhos define o estado da arte como um campo que explora soluções de monitoramento energético, mas que frequentemente se concentra ou em sistemas de acesso local ou em arquiteturas de nuvem que priorizam a rapidez de implementação. O presente trabalho posiciona-se neste cenário ao propor uma arquitetura mais robusta, pois combina o rigor do desenvolvimento de um firmware customizado, como visto em Nolêto (2023), com uma arquitetura de nuvem completa e profissional. Ao adotar componentes de mercado especialistas para cada tarefa, como HiveMQ para a comunicação de mensagens, InfluxDB como banco de dados de séries temporais e Grafana para a visualização, permite que este projeto não apenas implemente a evolução sugerida por Nolêto (2023), mas também proponha uma solução com maior controle e escalabilidade.

Para sintetizar as diferenças entre as abordagens, a Tabela 2.1 apresenta um comparativo de recursos. Ressalta-se que uma comparação quantitativa de custos e margem de erro não foi possível devido à ausência desses dados específicos nas publicações de referência. Portanto, a análise foca nos recursos arquiteturais, como flexibilidade da infraestrutura, propriedade dos dados e capacidade de customização.

Tabela 2.1: Comparativo de recursos (*features*) entre trabalhos correlatos e a proposta.

Trabalho	**Infraestrutura**	**Armazenamento**	**Customização (UI)**
Nolêto (2023)	Servidor Local (Sem nuvem)	Local (Cartão SD - Limitado)	Baixa (HTML fixo no chip)
Belo (2023)	Nuvem de Terceiros (Tasmota)	Proprietário da Plataforma	Baixa (App Genérico)
Santos et al. (2023)	Plataforma Tudo-em-um (Blynk)	Nuvem Blynk (Caixa Preta)	Média (Widgets prontos)
Proposta	**Modular (Microserviços em Docker)**	**Banco Dedicado (InfluxDB)**	**Alta (Grafana Open Source)**

Fonte: Elaborado pelo autor (2025).

A análise da Tabela 2.1 evidencia que a proposta se diferencia por oferecer uma arquitetura modular baseada em microserviços. Diferente das plataformas fechadas, o uso de um banco de dados dedicado (InfluxDB) garante a propriedade e portabilidade dos dados históricos, enquanto o Grafana oferece capacidade superior de visualização.

Capítulo 3

Metodologia e Desenvolvimento do Sistema

Este capítulo apresenta em detalhes a metodologia empregada para a concepção e construção do protótipo de monitoramento de energia. A abordagem adotada visa não apenas a implementação de um dispositivo funcional, mas também a criação de uma arquitetura de sistema robusta, escalável e alinhada com as melhores práticas para aplicações de Internet das Coisas. As seções subsequentes descreverão a arquitetura do sistema proposto, os materiais e ferramentas selecionados, os procedimentos de montagem e desenvolvimento de software, e, por fim, a metodologia que será utilizada para a validação experimental do protótipo.

3.1 Arquitetura do Sistema Proposto

A arquitetura do sistema foi concebida de forma modular, com cada componente desempenhando uma função específica no fluxo de dados, desde a aquisição no ambiente físico até a sua visualização em uma interface remota. Esta abordagem garante não apenas a clareza na implementação, mas também a flexibilidade para futuras manutenções ou expansões. O fluxo de informações, ilustrado no diagrama de blocos da Figura 3.1, segue uma sequência lógica de seis etapas principais: sensoriamento, processamento local, comunicação, coleta de dados, armazenamento e visualização.

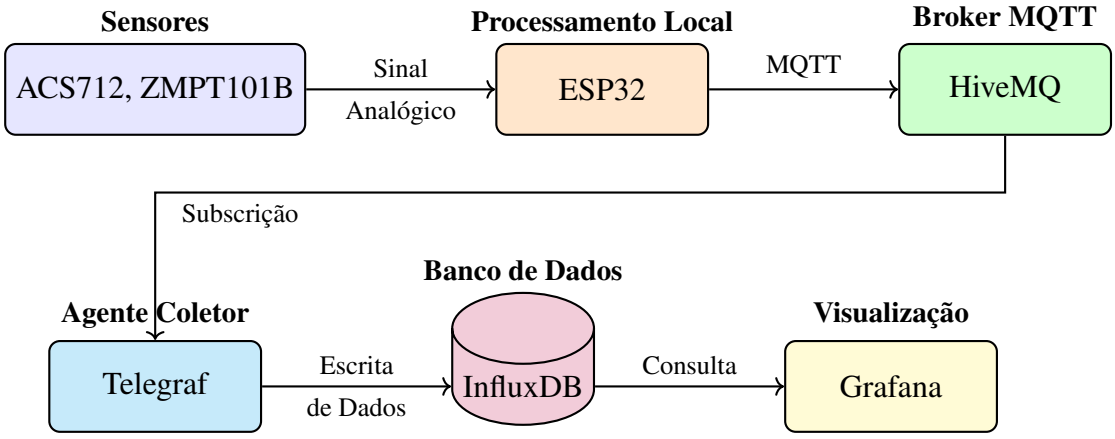
O ponto de partida é a camada de sensoriamento, onde os sensores de corrente (ACS712)

e tensão (ZMPT101B) realizam a aquisição dos sinais analógicos da rede elétrica. Estes sinais são então encaminhados para a unidade de processamento local, o microcontrolador ESP32, onde o firmware desenvolvido é responsável por digitalizar os sinais, realizar os cálculos necessários para obter os valores de tensão, corrente, potência ativa e potência aparente, e preparar os dados para a transmissão.

Uma vez processadas, as informações são publicadas de forma segura, utilizando o protocolo MQTT, para um broker na nuvem (HiveMQ), que atua como um intermediário central de mensagens. A partir do broker, os dados são coletados por um agente Telegraf, executado em um container Docker e hospedado de forma contínua na plataforma Railway. Este agente subscreve ao tópico MQTT e tem a função de persistir os dados, encaminhando-os para o InfluxDB, um banco de dados de séries temporais otimizado para o armazenamento eficiente deste tipo de informação.

Finalmente, a plataforma de visualização Grafana conecta-se ao InfluxDB para consultar as informações históricas e em tempo real, apresentando-as ao usuário final por meio de dashboards interativos e compreensíveis, compostos por gráficos e medidores.

Figura 3.1: Diagrama de blocos da arquitetura do sistema proposto.



Fonte: Elaborado pelo autor (2025).

3.2 Metodologia de Desenvolvimento

Esta seção detalha os procedimentos técnicos adotados para a construção do protótipo, desde a seleção dos componentes físicos até a configuração das plataformas de software.

O processo é apresentado de forma sequencial, descrevendo os materiais e as ferramentas utilizadas, a montagem do circuito eletrônico, a lógica do firmware embarcado e, finalmente, a configuração da infraestrutura de nuvem responsável por receber, armazenar e apresentar os dados.

3.2.1 Materiais e Ferramentas

Para a materialização do protótipo, foram selecionados componentes de hardware de baixo custo e amplo suporte da comunidade, bem como plataformas de software de código aberto ou com planos de serviço gratuitos. A Tabela 3.1 apresenta a lista de componentes de hardware utilizados na montagem do circuito, enquanto a Tabela 3.2 detalha as ferramentas de software e as plataformas em nuvem que compõem a arquitetura do sistema.

Ressalta-se que a escolha específica do modelo ACS712 (faixa de $\pm$5A) representa um compromisso de engenharia (*trade-off*). A seleção do sensor foi norteada pela disponibilidade imediata do componente e pela estratégia de validação da arquitetura. O objetivo primário era comprovar a viabilidade do fluxo de dados (do sensor à nuvem) em um ambiente controlado de bancada. O modelo de 5A, por oferecer maior sensibilidade para as cargas de baixa potência utilizadas nos testes, mostrou-se adequado para essa prova de conceito, assumindo-se que a arquitetura validada é escalável para sensores de maior capacidade em uma aplicação final. No entanto, essa característica implica que o sistema, em sua configuração atual, destina-se ao monitoramento de circuitos dedicados ou tomadas específicas, e não ao quadro geral de distribuição energética.

Tabela 3.1: Componentes de Hardware Utilizados no Protótipo.

Componente	**Descrição/Modelo Específico**
Microcontrolador	Placa de desenvolvimento ESP32 WROOM - DEVKIT
Sensor de Tensão	Módulo ZMPT101B (0 a 250V)
Sensor de Corrente	Módulo ACS712 (faixa de ±5A)
Fonte de Alimentação	Módulo Hi-Link HLK-PM01 (5VDC)
Varistor	10D471K (250V)
Filtro de Linha AC	Filtro para supressão de ruídos da rede elétrica
Fusível	10A 250V
Capacitor	Eletrolítico 1000μF 25V
Resistores	10kΩ e 470Ω
LED Indicador	LED difuso 5mm (Vermelho)
Placa de Montagem	Placa universal dupla face fibra vidro 8 x 12 cm

Tabela 3.2: Ferramentas de Software e Plataformas Utilizadas.

Ferramenta/Plataforma	**Descrição/Versão**
Desenvolvimento do Firmware	
Ambiente de Desenvolvimento	Arduino IDE 2.3.6
Linguagem do Firmware	C++
Infraestrutura de Nuvem	
Broker MQTT	HiveMQ Cloud (Plano Gratuito)
Agente Coletor de Dados	Telegraf (v1.28 ou superior)
Tecnologia de Containerização	Docker
Plataforma de Hospedagem	Railway (Plano Hobby)
Banco de Dados	InfluxDB Cloud (Plano Gratuito)
Plataforma de Visualização	Grafana Cloud (Plano Gratuito)
Ferramentas de Suporte	
Interface de Linha de Comando	Railway CLI (v4.10.0 ou superior)

3.2.2 Montagem do Circuito Eletrônico

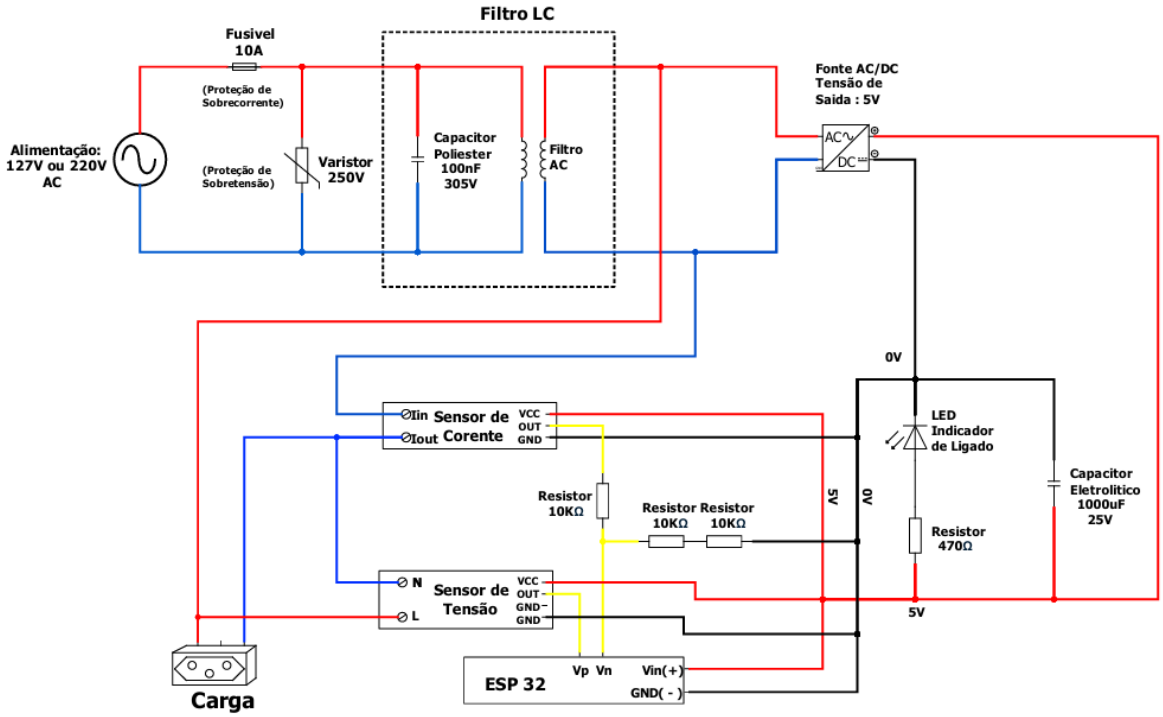
A montagem do circuito eletrônico foi realizada em uma placa de circuito perfurada universal, visando criar um protótipo robusto e com conexões elétricas estáveis. A interligação dos componentes, conforme ilustrado no esquema elétrico da Figura 3.2, foi projetada para garantir a correta aquisição dos sinais de tensão e corrente, bem como a alimentação segura de todo o sistema.

O coração do circuito é a placa de desenvolvimento ESP32. A alimentação do sistema é provida pelo módulo Hi-Link HLK-PM01, que converte a tensão da rede elétrica (AC) em uma tensão contínua e regulada de 5V (DC) para alimentar o microcontrolador. O circuito de entrada da fonte foi equipado com componentes de proteção, como um fusível e um varistor, para proteger o sistema contra sobrecorrente e surtos de tensão da rede.

O módulo sensor de tensão ZMPT101B, responsável por aferir a tensão da rede, foi co-

nectado aos pinos de alimentação e o seu pino de sinal foi ligado à porta de entrada analógica GPIO 36 do ESP32. Para a medição de corrente, foi utilizado o módulo ACS712. Este componente é inserido em série com a carga, e seu pino de sinal foi conectado à porta de entrada analógica GPIO 39. Ambos os módulos de sensores já possuem circuitos de condicionamento de sinal integrados, o que simplificou significativamente o design do hardware.

Figura 3.2: Esquema elétrico do circuito de medição.



Fonte: Elaborado pelo autor (2025).

3.2.3 Desenvolvimento do Firmware

O firmware do sistema, embarcado no microcontrolador ESP32, foi desenvolvido em linguagem C++ utilizando o ambiente de desenvolvimento integrado (IDE) do Arduino. A lógica do programa é o componente central do protótipo, responsável por orquestrar a leitura dos sensores, realizar os cálculos das grandezas elétricas e gerenciar a comunicação com a infraestrutura em nuvem. Para tal, foram utilizadas bibliotecas de código aberto que abstraem funcionalidades complexas, com destaque para a `EmonLib` e a `ZMPT101B`, para a medição de energia, e a `PubSubClient`, para a implementação do cliente MQTT.

O código foi estruturado em blocos funcionais, iniciando pela definição das configurações

de rede e pela inicialização das bibliotecas, conforme ilustrado na Figura 3.3. Nesta etapa, são definidas as credenciais de acesso à rede Wi-Fi e ao broker MQTT, bem como o mapeamento dos pinos do microcontrolador conectados aos sensores.

Figura 3.3: Trecho de código exibindo a inclusão das bibliotecas e as configurações iniciais do firmware.

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include "EmonLib.h"
#include <ZMPT101B.h>

const char* ssid = "NOME_DA_REDE_WIFI";
const char* password = "SENHA_WIFI";

const char* mqtt_server = "ENDERECO_DO_BROKER_HIVEMQ";
const int mqtt_port = 8883;
const char* mqtt_user = "USUARIO_MQTT";
const char* mqtt_password = "SENHA_MQTT";

const char* topic_publish = "tcc/murilo/telemetria";

const int PINO_TENSAO = 36;
const int PINO_CORRENTE = 39;

#define SENSITIVITY 216.5f
#define ACS_MPY 2.0
```

Fonte: Captura de tela elaborada pelo autor (2025).

A execução do firmware inicia-se na função `setup()`, responsável pela inicialização dos periféricos e pela configuração da conectividade. As tarefas executadas nesta etapa incluem a inicialização da comunicação serial para depuração e, crucialmente, a calibração dos sensores e o estabelecimento da conexão com a rede Wi-Fi local e com o broker MQTT.

Um passo fundamental para garantir a acurácia das medições do protótipo foi a calibração dos sensores. Este processo, realizado experimentalmente, consiste em ajustar os coeficientes no firmware para corrigir desvios e garantir que os valores lidos pelo sistema correspondam aos de referência. Para este projeto, os coeficientes foram previamente determinados e inseridos no código-fonte, conforme ilustrado na Figura 3.4.

Para o sensor de tensão, foi ajustado o coeficiente de sensibilidade da biblioteca `ZMPT101B`

através do comando `voltageSensor.setSensitivity(SENSITIVITY)`, utilizando o valor de `216.5f`, que foi empiricamente determinado. Para o sensor de corrente, a calibração foi aplicada na inicialização da biblioteca `EmonLib` com o comando `emon1.current(PINO_CORRENTE, ACS_MPY)`, onde o valor de `2.0` representa o multiplicador de ajuste para o sensor ACS712.

Vale notar que a calibração do sensor de tensão foi realizada através de um ajuste linear focado no ponto de operação nominal da rede elétrica local (aproximadamente 127V). Devido à sensibilidade deste componente a ruídos e distorções harmônicas, a precisão reportada assume condições de estabilidade da rede, não tendo sido realizada a caracterização completa da curva de erro para toda a faixa de operação (0-250V) do sensor.

Figura 3.4: Trecho de código da função `setup()` destacando a aplicação dos coeficientes de calibração.

```
41  void setup() {
42    Serial.begin(115200);
43    Serial.println("Iniciando o Medidor Inteligente...");
44
45    voltageSensor.setSensitivity(SENSITIVITY);
46    emon1.current(PINO_CORRENTE, ACS_MPY);
47
48    emon1.voltage(PINO_TENSAO, SENSITIVITY, 1.7);
49
50    setup_wifi();
51
52    client.setServer(mqtt_server, mqtt_port);
53    espClientSecure.setInsecure();
54  }
```

Fonte: Captura de tela elaborada pelo autor (2025).

O ciclo de execução principal e contínuo do sistema ocorre na função `loop()`. Dentro desta função, um temporizador não bloqueante, baseado na função `millis()`, garante que a leitura e o envio dos dados ocorram em intervalos de tempo regulares de cinco segundos, sem impedir a execução de outras tarefas. A cada intervalo, o firmware executa a rotina de medição, onde as bibliotecas são invocadas para calcular os valores RMS de tensão e corrente. Com base nestes valores, o programa calcula as grandezas de potência ativa e aparente, formata os dados em uma estrutura de texto JSON e os publica no tópico MQTT, conforme detalhado na Figura 3.5. Funções auxiliares, como `setup_wifi()` e `reconnect_mqtt()`,

garantem a robustez da conexão, restabelecendo-a automaticamente em caso de falha.

Figura 3.5: Trecho de código do ciclo principal `loop()`, exibindo a rotina de medição, cálculo, formatação e publicação dos dados.

```
56  void loop() {
57
58    if (!client.connected()) {
59      reconnect_mqtt();
60    }
61    client.loop();
62
63    unsigned long now = millis();
64    if (now - lastMsg >= interval) {
65      lastMsg = now;
66
67      emon1.calcVI(20, 2000);
68
69      vrms = voltageSensor.getRmsVoltage();
70      irms = emon1.Irms;
71      pot_aparente = vrms * irms;
72      pot_ativa = pot_aparente * emon1.powerFactor;
73
74      if (vrms < 10.0) vrms = 0;
75      if (irms < 0.06) irms = 0;
76      if (pot_ativa < 5.0) pot_ativa = 0;
77
78      String payload = "{";
79      payload += "\"tensao\":" + String(vrms) + ",";
80      payload += "\"corrente\":" + String(irms) + ",";
81      payload += "\"potencia_ativa\":" + String(pot_ativa) + ",";
82      payload += "\"potencia_aparente\":" + String(pot_aparente);
83      payload += "}";
84
85      char msg[100];
86      payload.toCharArray(msg, 100);
87
88      client.publish(topic_publish, msg);
89
90      Serial.println("Dados publicados: " + payload);
91    }
92  }
```

Fonte: Captura de tela elaborada pelo autor (2025).

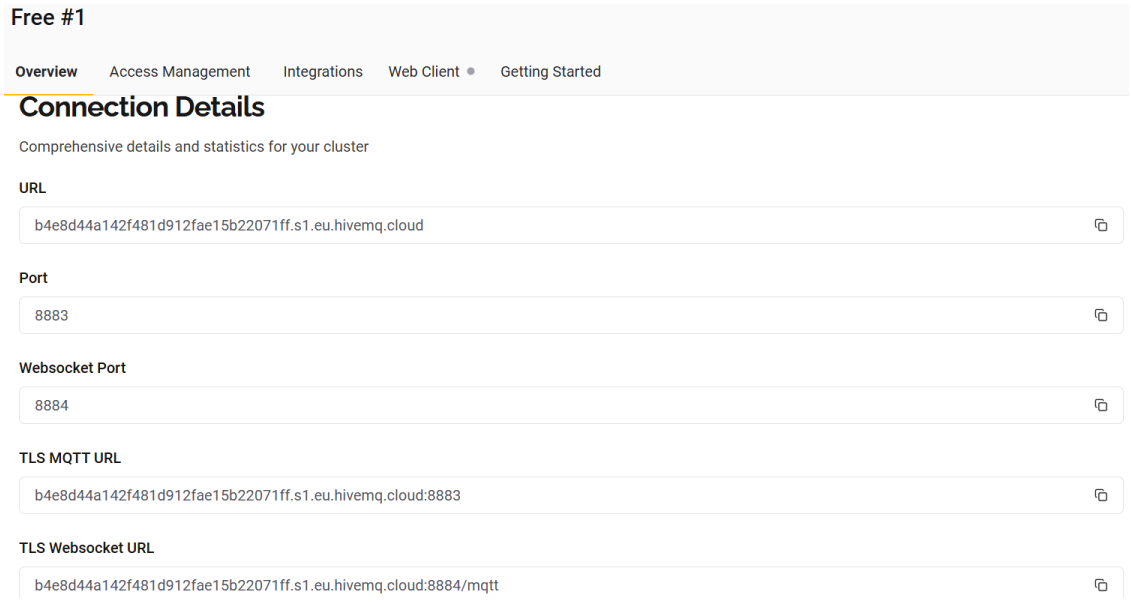
3.2.4 Configuração da Infraestrutura de Nuvem

A funcionalidade central de um sistema de Internet das Coisas reside na sua capacidade de transmitir e tornar os dados acessíveis remotamente. Para este fim, foi configurada uma infraestrutura de nuvem robusta, utilizando um conjunto de plataformas especializadas que

operam de forma integrada. O processo de configuração seguiu a ordem cronológica do desenvolvimento, iniciando pelo broker de mensagens, passando pela implementação da ponte de integração de dados, e finalizando com a configuração do banco de dados e da plataforma de visualização.

O primeiro componente configurado foi o broker MQTT, utilizando a plataforma HiveMQ Cloud. O broker atua como o intermediário central de mensagens, desacoplando o dispositivo de campo (o ESP32) das aplicações de back-end. O processo consistiu na criação de uma instância gratuita, a partir da qual foram obtidas as informações primárias de conexão, como o endereço do servidor e a porta para conexões seguras (TLS), conforme ilustrado na Figura 3.6. Adicionalmente, na aba "Access Management" da mesma plataforma, foram geradas as credenciais de acesso (nome de usuário e senha) para garantir a autenticação do cliente. O conjunto completo destas informações foi então inserido no firmware do ESP32.

Figura 3.6: Painel de gerenciamento do HiveMQ Cloud.

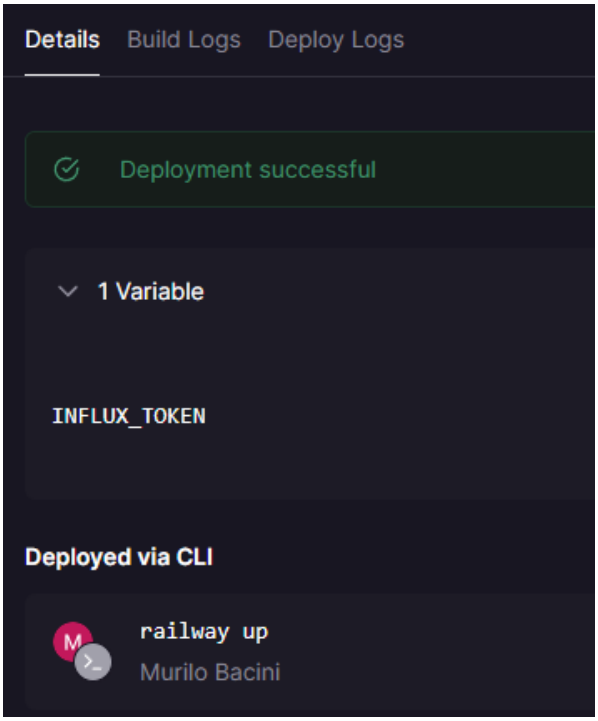


Fonte: Captura de tela elaborada pelo autor (2025).

A etapa subsequente consistiu na implementação da ponte de integração entre o broker e o banco de dados. Diante da complexidade de estabelecer uma conexão direta, a solução adotada foi utilizar o agente coletor Telegraf. Para garantir a operação contínua e independente deste agente na nuvem, utilizou-se a tecnologia de conteinerização Docker. Foi desenvolvido um `Dockerfile`, um arquivo de instruções que serviu como "planta" para construir um con-

têiner padronizado contendo o Telegraf e sua configuração. O processo para materializar esta solução na nuvem foi orquestrado pela plataforma Railway, por meio de sua interface de linha de comando (CLI). A execução do comando `railway up` enviou o projeto para a plataforma, que automaticamente construiu e implantou o contêiner. Este processo culminou em uma implantação bem sucedida, conforme detalhado na Figura 3.7, que evidencia não apenas o sucesso do deploy, mas também a aplicação de boas práticas de segurança, como o uso de uma variável de ambiente para o token de acesso ao banco de dados.

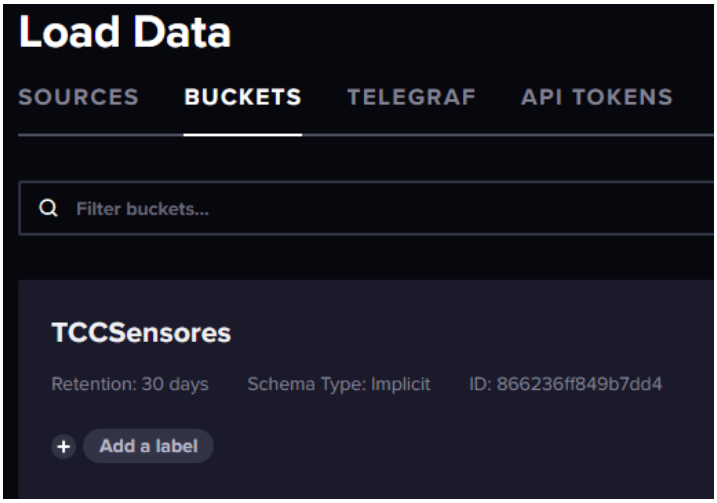
Figura 3.7: Painel de detalhes do serviço na plataforma Railway após uma implantação bem-sucedida.



Fonte: Captura de tela elaborada pelo autor (2025).

Com a ponte de dados estabelecida, o terceiro passo foi a configuração do sistema de persistência no InfluxDB Cloud. Na plataforma, foi criado um *bucket*, um contêiner de dados nomeado "TCCSensores" para armazenar as medições de consumo energético, conforme ilustrado na Figura 3.8. Em uma etapa subsequente na mesma plataforma, foi gerado um token de autenticação com permissões de escrita para este *bucket*, o qual foi utilizado na variável de ambiente do Railway para autorizar a escrita dos dados.

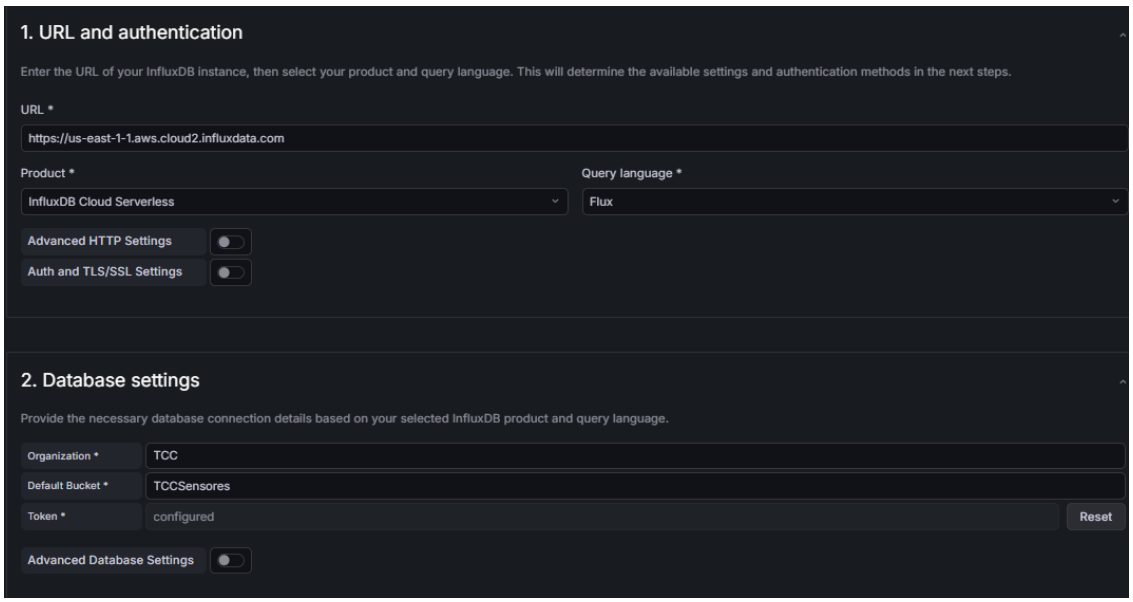
Figura 3.8: Interface do InfluxDB Cloud, exibindo o bucket de dados “TCCSensores”.



Fonte: Captura de tela elaborada pelo autor (2025).

Finalmente, a camada de visualização foi implementada na plataforma Grafana Cloud. O Grafana foi conectado ao InfluxDB, estabelecendo-o como uma fonte de dados (*data source*), conforme apresentado na Figura 3.9. Esta conexão foi autenticada utilizando as credenciais da organização, o nome do *bucket* e o token gerado no InfluxDB, completando o fluxo de dados desde a medição física até a apresentação visual.

Figura 3.9: Tela de configuração da fonte de dados no Grafana.



Fonte: Captura de tela elaborada pelo autor (2025).

3.3 Metodologia de Validação

Após a descrição detalhada da arquitetura e do desenvolvimento do sistema, torna-se imperativo estabelecer um procedimento experimental rigoroso para a validação de sua funcionalidade e precisão. Esta seção descreve a metodologia que será empregada para aferir o desempenho do protótipo. O objetivo é verificar objetivamente a acurácia das medições de consumo energético, comparando os dados coletados pelo sistema proposto com os de um instrumento de referência, sob diferentes condições de carga.

3.3.1 Plano de Testes

O experimento de validação foi concebido para avaliar a precisão do protótipo em cenários de consumo com diferentes características de carga, simulando o uso de aparelhos distintos em um ambiente residencial. Para tal, foi utilizada uma bancada de testes composta por uma régua de tomadas conectada à rede elétrica. O protótipo e um wattímetro comercial de referência foram instalados em série, de modo que ambos medissem a mesma carga total simultaneamente.

O procedimento consistiu em um teste de cargas incrementais e decrementais para simular o uso progressivo de aparelhos em um ambiente residencial. A sequência de acionamento foi a seguinte: iniciou-se com o sistema em vazio; em seguida, foi adicionada a primeira carga (uma lâmpada); posteriormente, uma segunda carga com componente indutivo (um ventilador) foi adicionada ao circuito; após um período de medição com ambas as cargas, a lâmpada foi removida; e, finalmente, o ventilador foi desligado, retornando o sistema ao estado em vazio. Durante todo o processo, o comportamento da potência foi registrado para a análise qualitativa da responsividade do sistema. Para a análise quantitativa, foram extraídas as medições de estado estacionário de cada um dos três cenários de carga (lâmpada, ventilador e ambos) para o posterior cálculo do erro.

3.3.2 Métricas de Avaliação

Para quantificar a precisão do protótipo e avaliar o seu desempenho de forma objetiva, a métrica de avaliação selecionada foi o Erro Percentual Médio (EPM). Esta métrica é amplamente utilizada em instrumentação para determinar a acurácia de um dispositivo de medição em relação a um padrão de referência (SOUZA, 2020). A escolha do EPM se justifica por sua simplicidade de cálculo e, principalmente, por sua interpretabilidade intuitiva, uma vez que expressa o desvio médio das medições em termos percentuais.

O cálculo do EPM será realizado a partir de um conjunto de n amostras de potência instantânea coletadas simultaneamente do protótipo ($P_{\text{protótipo}}$) e do wattímetro de referência ($P_{\text{referência}}$), conforme a Equação 3.1.

$$EPM = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{P_{\text{protótipo},i} - P_{\text{referência},i}}{P_{\text{referência},i}}\right| \times 100\% \tag{3.1}$$

Como critério de sucesso para a validação, foi estabelecido que o Erro Percentual Médio do protótipo deve ser inferior a 5%. Este limiar é consistente com os níveis de acurácia reportados na literatura para protótipos de monitoramento de energia de baixo custo. Tais trabalhos frequentemente consideram erros nesta faixa como um indicador de desempenho satisfatório para aplicações não comerciais, cujo objetivo principal é a conscientização do consumidor, não necessitando da precisão de um medidor para fins de faturamento (ABUBAKAR et al., 2017).

Capítulo 4

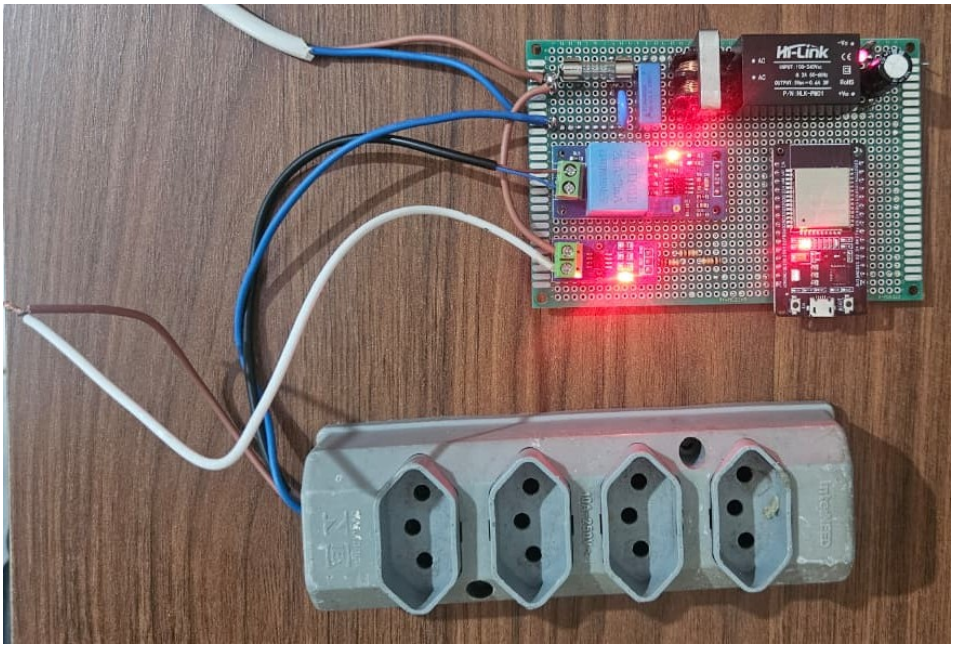
Resultados e Discussão

Este capítulo apresenta os resultados práticos obtidos a partir da implementação da metodologia descrita no capítulo anterior. O foco desta seção é demonstrar a funcionalidade do sistema desenvolvido, analisar os dados coletados em um cenário de uso e validar a acurácia do protótipo em relação a um instrumento de referência. Inicialmente, será apresentado o protótipo finalizado, tanto em seu aspecto físico quanto em sua interface de software. Em seguida, serão analisados os padrões de consumo registrados pelo sistema, e, por fim, serão discutidos os resultados dos testes de validação, contextualizando o desempenho do protótipo e suas implicações.

4.1 Apresentação do Protótipo Funcional

A execução da metodologia descrita resultou em um protótipo funcional composto por duas partes integrantes: o hardware de medição e a interface de software para visualização dos dados. A Figura 4.1 exibe o dispositivo de hardware finalizado e em operação, com o microcontrolador ESP32, os sensores de tensão e corrente, e o circuito de alimentação e proteção montados em uma placa de circuito perfurada. A montagem buscou otimizar a organização dos componentes e garantir a segurança e a estabilidade do dispositivo, cujo estado operacional é indicado por LEDs ativos.

Figura 4.1: Protótipo de hardware finalizado e em operação.



Fonte: Elaborado pelo autor (2025).

A interface de software, que constitui o principal meio de interação do usuário com os dados, foi implementada na plataforma Grafana. O painel de controle, ou *dashboard*, ilustrado na Figura 4.2, foi projetado para apresentar as informações de consumo de forma clara e intuitiva. O *dashboard* é composto por um conjunto de painéis que exibem as métricas em tempo real, como medidores para Tensão (V), Corrente (A) e Potência Ativa (W) e dados agregados, como o Consumo Total (kWh) e o Custo Estimado (R$) para o período selecionado. O painel principal consiste em um gráfico de série temporal que plota o histórico da potência ativa (W), permitindo a identificação de tendências e padrões.

Figura 4.2: Dashboard de monitoramento desenvolvido na plataforma Grafana.

Fonte: Captura de tela elaborada pelo autor (2025).

4.2 Análise dos Dados de Consumo Coletados

Com o protótipo devidamente implementado, a primeira etapa de sua avaliação consiste em demonstrar sua capacidade de monitorar e responder a variações de carga em tempo real. Para este fim, foi conduzido um experimento de cargas incrementais e decrementais, seguindo o plano de testes delineado na Seção 3.3.1. O objetivo desta análise é verificar qualitativamente a resposta do sistema à adição e remoção sequencial de diferentes aparelhos.

A série temporal da potência ativa (W), registrada pelo sistema durante a execução do experimento, pode ser observada no painel “Histórico de Potência” do dashboard, apresentado anteriormente na Figura 4.2. A análise do gráfico demonstra a capacidade do protótipo de reagir em tempo real, exibindo um perfil característico de “degraus”, onde cada variação abrupta no patamar de potência corresponde a uma alteração no estado das cargas conectadas.

O experimento iniciou-se com o sistema em vazio, registrando um consumo próximo de zero. Em seguida, ao conectar a primeira carga (uma lâmpada), observa-se um degrau nítido no consumo, que se estabiliza em um novo patamar. Posteriormente, com a adição da segunda carga (um ventilador), o sistema registra um segundo degrau, e a potência total medida

corresponde à soma das duas cargas. O gráfico também captura com precisão as etapas de remoção: primeiramente, ao desligar a lâmpada, a potência retorna ao patamar de consumo apenas do ventilador e, finalmente, ao desligar o ventilador, o consumo retorna ao nível zero. Esta análise qualitativa, portanto, valida a funcionalidade essencial do protótipo como uma ferramenta de monitoramento responsiva, capaz de rastrear dinamicamente as variações de consumo.

4.3 Validação da Precisão do Sistema

Após a análise qualitativa da resposta do sistema a diferentes cargas, a presente seção se dedica à validação quantitativa da acurácia do protótipo. O objetivo deste procedimento é determinar, por meio de comparação direta com instrumentos de referência, o desvio das medições do sistema proposto, assegurando sua confiabilidade como ferramenta de monitoramento.

Para a aferição, foram utilizados como referência um multímetro digital Minipa (modelo ET-2517A/ET-2615A) para a medição da tensão e um alicate amperímetro de precisão Aneng para a medição da corrente. A grandeza de comparação selecionada foi a potência aparente (VA), calculada a partir do produto dos valores de tensão (V) e corrente (A) medidos simultaneamente tanto pelos instrumentos de referência quanto pelo protótipo. A Tabela 4.1 consolida os dados coletados durante o experimento. Para cada um dos três cenários de carga, a tabela apresenta a potência aparente calculada para a referência e para o protótipo, bem como o erro percentual individual para cada cenário.

Tabela 4.1: Resultados comparativos da medição de potência aparente

Cenário de Teste	**Potência de Referência (VA)**	**Potência do Protótipo (VA)**	**Erro Percentual (%)**
Carga 1: Lâmpada	143,36	139,70	2,55
Carga 2: Ventilador	136,96	135,68	0,94
Carga 3: Lâmpada + Ventilador	273,42	265,86	2,77

A partir dos erros percentuais individuais, foi computado o Erro Percentual Médio (EPM) geral do sistema, conforme a métrica definida pela Equação 3.1. Este valor representa a média da imprecisão do protótipo ao longo dos cenários testados. O resultado final obtido para o protótipo foi de aproximadamente:

$$\textbf{EPM} \approx \textbf{2,1\%}$$

O valor de 2,1% para o Erro Percentual Médio se encontra confortavelmente dentro do limiar de 5% estabelecido como critério de sucesso na metodologia. Este resultado, portanto, valida quantitativamente a precisão do protótipo, confirmando que o hardware e o firmware desenvolvidos são capazes de aferir o consumo de energia com um grau de acurácia satisfatório para aplicações de monitoramento residencial.

4.4 Discussão dos Resultados

A análise conjunta dos resultados apresentados nas seções anteriores permite uma avaliação abrangente do sistema desenvolvido, tanto em seu comportamento funcional quanto em sua precisão quantitativa. A discussão a seguir visa interpretar esses resultados, refletir sobre o processo de desenvolvimento e contextualizar o desempenho do protótipo em relação aos objetivos estabelecidos para este trabalho.

A primeira constatação, derivada da análise qualitativa dos dados (Seção 4.2), é que o sistema se comportou conforme o esperado. O perfil de consumo em "degraus" demonstrou a capacidade do protótipo de reagir em tempo real às variações de carga, validando a eficácia da arquitetura de coleta e transmissão de dados. No que tange à acurácia, a validação quantitativa (Seção 4.3) confirmou que a precisão do protótipo é satisfatória, com um Erro Percentual Médio de aproximadamente 2,1%, resultado que atesta a qualidade do circuito e da calibração realizada.

Entretanto, é importante ressaltar as fronteiras de validade destes resultados. A validação experimental concentrou-se em cargas lineares e em regime estacionário (após a estabilização da corrente). O experimento não contemplou a análise de resposta transiente (como o

overshoot de partida de motores), nem testes de longa duração (24 horas) para verificação de deriva térmica ou flutuações da tensão da rede elétrica.

Adicionalmente, o cenário de testes utilizou cargas com fator de potência unitário ou levemente indutivo. Em um ambiente residencial real, a presença de cargas não lineares (como fontes chaveadas de computadores e eletrônicos modernos) introduz distorções harmônicas que podem afetar a precisão da leitura. Portanto, o erro médio de 2,1% deve ser considerado válido estritamente para o perfil de cargas resistivas e indutivas simples testado em bancada.

Apesar do sucesso final, o processo de desenvolvimento não foi isento de desafios. As principais dificuldades encontradas residiram na calibração inicial dos sensores e na integração da infraestrutura de nuvem. A obtenção de uma medição precisa exigiu um ajuste fino e iterativo dos coeficientes de calibração no firmware, um processo que se mostrou fundamental para a confiabilidade dos dados. Adicionalmente, a implementação da ponte de dados entre o broker HiveMQ e o banco de dados InfluxDB revelou-se uma tarefa complexa. A solução de utilizar um agente coletor Telegraf, hospedado em um contêiner Docker na plataforma Railway, embora robusta, demandou um extenso trabalho de depuração para superar uma série de obstáculos, desde a configuração inicial da ferramenta de implantação até a resolução de incompatibilidades no ambiente de execução do contêiner.

Em suma, os resultados obtidos demonstram que o sistema se provou funcional, preciso e robusto. Apesar das dificuldades inerentes a um projeto que integra hardware, firmware e múltiplos serviços de nuvem, a combinação da metodologia proposta com um processo de depuração iterativo permitiu cumprir com sucesso o objetivo central deste trabalho: o desenvolvimento de um sistema de monitoramento de energia de baixo custo, eficaz e alinhado com os princípios da Internet das Coisas.

Capítulo 5

Conclusão

Este capítulo final consolida os resultados e as contribuições do presente trabalho. A primeira seção retoma os objetivos propostos na introdução e sintetiza como o protótipo desenvolvido e a arquitetura implementada os alcançaram. Em seguida, são discutidas as limitações inerentes a este projeto, reconhecendo os seus limites e o escopo definido. Finalmente, são delineadas possíveis direções para trabalhos futuros, explorando as evoluções e os aprimoramentos que podem ser construídos a partir da base estabelecida por esta pesquisa.

5.1 Síntese das Contribuições

O presente trabalho atingiu com sucesso o seu objetivo principal de desenvolver e validar um protótipo funcional de um sistema de monitoramento de energia para residências inteligentes, baseado em tecnologias de Internet das Coisas. Conforme os objetivos propostos na introdução, foi realizada uma fundamentação teórica que guiou a seleção de uma arquitetura de hardware e software robusta, seguida pelo desenvolvimento de um firmware customizado e pela configuração de uma infraestrutura de nuvem completa.

A principal contribuição deste projeto reside na implementação de uma solução de baixo custo que não apenas monitora o consumo em tempo real, mas o faz através de uma arquitetura de nuvem que supera as limitações de sistemas de acesso puramente local. Foi demonstrado que a arquitetura proposta, composta pelo microcontrolador ESP32, o broker MQTT HiveMQ, o agente coletor Telegraf (em um container Docker hospedado na plata-

forma Railway), o banco de dados de séries temporais InfluxDB e a plataforma de visualização Grafana, é capaz de coletar, transmitir de forma segura, armazenar e apresentar os dados de consumo de maneira eficaz e escalável.

Os testes de validação, realizados em ambiente controlado, comprovaram a eficácia do sistema, que alcançou uma precisão satisfatória, com um erro percentual médio de aproximadamente 2,1% quando comparado a um medidor comercial de referência. Dessa forma, o trabalho contribui para a área ao apresentar uma solução completa e alinhada com as melhores práticas da indústria de monitoramento, servindo como uma base sólida para futuras investigações no campo da eficiência energética residencial.

5.2 Limitações do Trabalho

É importante reconhecer que o presente trabalho, por sua natureza de protótipo acadêmico, possui limitações inerentes que definem o seu escopo e abrem caminho para futuras melhorias. Estas limitações não invalidam os resultados obtidos, mas contextualizam a contribuição desta pesquisa.

Do ponto de vista do hardware, o protótipo foi montado em uma placa de circuito perfurada universal. Embora esta abordagem ofereça maior robustez em comparação com uma montagem em protoboard, ela ainda não possui a otimização de espaço e a integração de uma Placa de Circuito Impresso (PCB) projetada especificamente para a aplicação.

Além da questão construtiva da placa, a segurança física do dispositivo é um ponto crítico que restringe o uso atual. O protótipo apresenta uma montagem aberta (*open-frame*), adequada apenas para ambientes controlados de teste. Para que a solução possa ser instalada em um ambiente residencial real, é mandatório o encapsulamento de todo o circuito em um gabinete de material dielétrico. Isso garantiria o isolamento elétrico necessário (Classe II), impedindo o contato acidental do usuário com as partes energizadas da rede AC e protegendo os componentes contra fatores ambientais.

Adicionalmente, a escolha do sensor de corrente ACS712 na versão de ±5A impõe um limite funcional ao protótipo. Esta faixa de operação restringe a medição a cargas de baixa e média potência, tipicamente até 635W em uma rede de 127V, sendo adequado para o monito-

ramento de equipamentos eletrônicos individuais, mas inviabilizando a medição de aparelhos de alto consumo, como chuveiros elétricos ou fornos. A capacidade máxima do sistema é, ademais, limitada por outros componentes de proteção, como o fusível de 10A.

No âmbito do software e da segurança lógica, o projeto adotou simplificações visando a agilidade na validação funcional. As credenciais de rede Wi-Fi e do broker foram mantidas estáticas no código-fonte (*hardcoded*), e a validação da cadeia de certificação SSL/TLS foi inibida através do método `setInsecure()`. Esta configuração foi uma opção de projeto para contornar limitações de gerenciamento de certificados no ESP32, mas reconhece-se que reduz o rigor da autenticação. Para uma aplicação comercial robusta, é imprescindível a implementação de validação estrita de Autoridade Certificadora (CA) e um mecanismo de provisionamento dinâmico de credenciais.

Adicionalmente, a interface de visualização atual, implementada na plataforma Grafana, apresenta um perfil técnico (orientado a operadores/NOC). Embora o sistema forneça dados essenciais como custo estimado e consumo acumulado, a navegação e a apresentação visual atendem primariamente à persona de um técnico ou instalador, focada na validação dos dados brutos (tensão, corrente e potência instantânea). Reconhece-se a ausência de um dashboard simplificado e exclusivo para a persona do morador, cuja implementação através de um aplicativo móvel nativo, com foco em usabilidade e resumos gerenciais, é sugerida como a evolução natural do produto.

Por fim, o escopo funcional do sistema foi deliberadamente focado no monitoramento do consumo energético agregado. Funcionalidades como o controle de cargas (ligar ou desligar aparelhos remotamente) ou a desagregação do consumo por aparelho individual, conhecida como Non-Intrusive Load Monitoring (NILM), não foram objeto de implementação neste trabalho.

5.2.1 Dependência de Serviços e Reprodutibilidade

A arquitetura atual baseia-se nos planos gratuitos (*Free Tier*) dos serviços HiveMQ, Railway e InfluxDB. Embora funcional para prototipagem, essa dependência impõe restrições operacionais significativas, como limites baixos de conexões simultâneas MQTT (dificultando a

escalabilidade para múltiplos sensores na residência) e janelas curtas de retenção de dados históricos. Além disso, métricas de latência e confiabilidade de entrega de mensagens não foram formalmente aferidas neste escopo.

No que tange à implantação, a configuração do ambiente foi realizada manualmente através de consoles web, o que fragmenta a gestão da infraestrutura. Para garantir a reprodutibilidade e a portabilidade do sistema em um cenário de produção, é necessária a adoção de práticas de Infraestrutura como Código (IaC), utilizando manifestos (como Docker Compose, Terraform ou Ansible) para automatizar o provisionamento dos serviços e eliminar a configuração manual distribuída.

5.3 Trabalhos Futuros

As limitações identificadas na seção anterior, inerentes à natureza de um protótipo acadêmico, delineiam um claro caminho para a evolução e o aprimoramento do sistema, visando a sua transformação em um produto robusto e comercializável. As sugestões para trabalhos futuros são apresentadas a seguir, abrangendo melhorias de hardware, software e a expansão de funcionalidades.

Do ponto de vista do hardware, a evolução natural do protótipo seria o projeto de uma Placa de Circuito Impresso (PCB) dedicada. Esta abordagem permitiria uma maior compactação do circuito, aumentaria a sua confiabilidade ao substituir as ligações manuais por trilhas impressas e viabilizaria a produção em escala. Adicionalmente, o desenvolvimento de um invólucro (*case*) customizado, utilizando tecnologias como a impressão 3D, garantiria a proteção dos componentes eletrônicos e conferiria ao dispositivo um acabamento de produto final, seguro para o utilizador doméstico.

No âmbito do firmware, uma melhoria crucial seria a implementação de um modo de configuração para o utilizador final. Em vez de ter as credenciais da rede Wi-Fi inseridas diretamente no código-fonte, poderia ser desenvolvido um sistema de provisionamento via *Bluetooth Low Energy* (BLE), permitindo que o utilizador configure a rede de forma simples e segura através de um aplicativo de celular, tornando o dispositivo verdadeiramente *plug-and-play*.

Para a interface com o utilizador, a plataforma Grafana poderia ser substituída pelo desenvolvimento de um aplicativo móvel nativo para os sistemas Android e iOS. Tal aplicativo ofereceria uma experiência de uso mais simples e intuitiva para o consumidor final, focada na visualização clara do consumo em tempo real e na geração de relatórios de gastos, além de servir como interface para a configuração do dispositivo.

Finalmente, o escopo funcional do projeto pode ser significativamente expandido. A adição de um módulo relé permitiria não apenas o monitoramento, mas também o controle remoto de cargas, possibilitando ligar ou desligar aparelhos. Esta funcionalidade, por sua vez, abriria portas para a integração com assistentes de voz, como a Amazon Alexa ou o Google Assistente, adicionando uma camada de automação e conveniência. Uma linha de pesquisa mais avançada seria a exploração de técnicas de Desagregação de Cargas Não Intrusiva, ou NILM, utilizando algoritmos de aprendizado de máquina para tentar identificar o consumo de aparelhos individuais a partir do sinal agregado medido, agregando um valor imenso à solução.

Apêndice A

Código-Fonte do Firmware

Este apêndice apresenta o código-fonte completo do firmware desenvolvido para o microcontrolador ESP32, que constitui a lógica embarcada do protótipo. O programa foi escrito em C++ e compilado utilizando o ambiente de desenvolvimento Arduino IDE, versão 2.3.6.

A listagem a seguir está documentada com comentários que detalham a função de cada bloco, desde a inicialização dos periféricos e a conexão com a rede até o ciclo principal de leitura dos sensores, cálculo das grandezas elétricas e publicação dos dados para o broker MQTT.

Listing A.1: Código-fonte completo do firmware embarcado no ESP32.

```
// Autor: Murilo Roberto Bacini (2025)

// Inclusão das bibliotecas
#include <WiFi.h>              // Conexão Wi-Fi
#include <WiFiClientSecure.h>  // Redes seguras
#include <PubSubClient.h>      // Protocolo MQTT
#include "EmonLib.h"           // Biblioteca para sensor de corrente
#include <ZMPT101B.h>          // Biblioteca para o sensor de tensão

// Configurações de Rede (as informações sensíveis foram substituídas)
const char* ssid = "NOME_DA_REDE_WIFI";
const char* password = "SENHA_WIFI";
// Configurações de Conexão
const char* mqtt_server = "ENDERECO_DO_BROKER_HIVEMQ";
```

```
15 const int mqtt_port = 8883;
16 const char* mqtt_user = "USUARIO_MQTT";
17 const char* mqtt_password = "SENHA_MQTT";
18
19 // Tópico MQTT para publicação dos dados de telemetria
20 const char* topic_publish = "tcc/murilo/telemetria";
21
22 // Configuração dos Sensores e Pinos
23 const int PINO_TENSAO = 36;    // GPIO 36 (ADC1_CH0)
24 const int PINO_CORRENTE = 39; // GPIO 39 (ADC1_CH3)
25 // Calibração dos Sensores
26 #define SENSITIVITY 216.5f   // Calibração para o sensor de tensão
27 #define ACS_MPY 2.0  // Calibração para o sensor de corrente ACS712
28
29 // Criação dos "Objetos"
30 EnergyMonitor emon1;   // Objeto para o monitor de energia (corrente)
31 ZMPT101B voltageSensor(PINO_TENSAO, 60.0); // Objeto para o sensor de
      tensão, já informando o pino e a frequência da rede (60Hz).
32
33 WiFiClientSecure espClientSecure; // Cliente Wi-Fi
34 PubSubClient client(espClientSecure); // Cliente MQTT, conexão segura
35
36 // Variáveis Globais
37 unsigned long lastMsg = 0;   // tempo (em ms) da última publicação
38 const long interval = 5000;  // intervalo entre publicações (5 s)
39
40 // Variáveis para armazenar as leituras dos sensores
41 float vrms = 0.0;
42 double irms = 0.0;
43 double pot_aparente = 0.0;
44 double pot_ativa = 0.0;
45
46 // FUNÇÃO DE SETUP
47 void setup() {
48   // Inicializa a comunicação serial com a taxa de 115200 bps.
```

```
49    Serial.begin(115200);
50    Serial.println("Iniciando o Medidor Inteligente...");
51    // Configura os sensores com os valores de calibração
52    voltageSensor.setSensitivity(SENSITIVITY);
53    emon1.current(PINO_CORRENTE, ACS_MPY);
54    // A EmonLib precisa saber da tensão para calcular a Potência Ativa
          corretamente. Mesmo lendo a tensão com outra biblioteca, passamos
           o pino para a EmonLib.
55    emon1.voltage(PINO_TENSAO, SENSITIVITY, 1.7);
56
57    setup_wifi();  // Conecta ao Wi-Fi
58    // Configura o cliente MQTT
59    client.setServer(mqtt_server, mqtt_port);
60    espClientSecure.setInsecure();
61  }
62
63  // FUNÇÃO DE LOOP
64  void loop() {
65    // Garante que a conexão com o servidor MQTT esteja sempre ativa
66    if (!client.connected()) {
67      reconnect_mqtt();
68    }
69    client.loop(); // Permite que o cliente MQTT processe mensagens e/s
70
71    //  Lógica de temporizador
72    unsigned long now = millis();
73    if (now - lastMsg >= interval) {
74      lastMsg = now; // Atualiza o tempo da última mensagem
75
76  //EmonLib para fazer uma leitura completa. O calcVI lê várias amostras
        de tensão e corrente para dar um resultado preciso.
77      emon1.calcVI(20, 2000); // 20 ciclos de onda, com um timeout de
            2000ms.
78
79      vrms = voltageSensor.getRmsVoltage();     // Leitura mais precisa
```

```
80      irms = emon1.Irms;          // Corrente RMS calculada pela EmonLib
81      pot_aparente = vrms * irms;             // Potência Aparente (VA)
82      pot_ativa = pot_aparente * emon1.powerFactor;//Potência Ativa (W)
83
84      // Filtro de ruído
85      if (vrms < 10.0) vrms = 0;
86      if (irms < 0.06) irms = 0;
87      if (pot_ativa < 5.0) pot_ativa = 0;
88
89      // Publicação dos Dados - Monta uma string no formato JSON
90      String payload = "{";
91      payload += "\"tensao\":" + String(vrms) + ",";
92      payload += "\"corrente\":" + String(irms) + ",";
93      payload += "\"potencia_ativa\":" + String(pot_ativa) + ",";
94      payload += "\"potencia_aparente\":" + String(pot_aparente);
95      payload += "}";
96
97      // Converte a String do payload para um array de char
98      char msg[100];
99      payload.toCharArray(msg, 100);
100
101     // Publica a mensagem formatada no tópico MQTT especificado
102     client.publish(topic_publish, msg);
103
104     // Imprime os dados no monitor serial para fins de depuração
105     Serial.println("Dados publicados: " + payload);
106   }
107 }
108
109 // FUNÇÕES AUXILIARES
110 void setup_wifi() { // Função para conectar ao Wi-Fi
111   delay(10);
112   Serial.println();
113   Serial.print("Conectando-se a ");
114   Serial.println(ssid);
```

```
115
116   WiFi.begin(ssid, password);
117
118   while (WiFi.status() != WL_CONNECTED) {
119     delay(500);
120     Serial.print(".");
121   }
122
123   Serial.println("");
124   Serial.println("WiFi conectado!");
125   Serial.print("Endereço de IP: ");
126   Serial.println(WiFi.localIP());
127 }
128
129 // Função para se reconectar ao servidor MQTT caso a conexão caia
130 void reconnect_mqtt() {
131   while (!client.connected()) {
132     Serial.print("Tentando conectar ao MQTT...");
133 // Cria um Client ID aleatório -> evitar conflitos de sessão no broker
134     String clientId = "ESP32Client-";
135     clientId += String(random(0xffff), HEX);
136
137     // Tenta se conectar ao broker com as credenciais definidas
138     if (client.connect(clientId.c_str(), mqtt_user, mqtt_password)) {
139       Serial.println("conectado!");
140     } else {
141       Serial.print("falhou, rc=");
142       Serial.print(client.state());
143       Serial.println(" tentando novamente em 5 segundos");
144       delay(5000);
145     }
146   }
147 }
```

Referências Bibliográficas

ABUBAKAR, I. et al. Calibration of ZMPT101B voltage sensor module using polynomial regression for accurate load monitoring. *ARPN Journal of Engineering and Applied Sciences*, v. 12, n. 4, p. 1076–1084, 2017.

Allegro MicroSystems. *ACS712 - Fully Integrated, Hall-Effect-Based Linear Current Sensor IC*. [S.l.], 2024. Datasheet, Rev. 22. Disponível em: www.allegromicro.com.

Associação Brasileira da Indústria Elétrica e Eletrônica. *Internet das Coisas (IoT) e a Transformação Digital no Setor Elétrico Brasileiro*. São Paulo, SP, 2019.

BELO, G. S. *Sistema IoT para monitoramento e controle de smart home*. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2023.

Empresa de Pesquisa Energética. *Anuário Estatístico de Energia Elétrica 2025 (ano-base 2024) - FactSheet*. Brasília, DF, 2025. Disponível em: https://www.epe.gov.br/sites-pt/publicacoes-dados-abertos/publicacoes/PublicacoesArquivos/publicacao-160/topico-168/anuario-factsheet.pdf. Acesso em: 03 out. 2025.

Grafana Labs. *Grafana Documentation*. 2023. <https://grafana.com/docs/grafana/latest/>. Acesso em: 29 set. 2025.

InfluxData. *Technical White Paper: The InfluxDB Time Series Platform*. [S.l.], 2023. Disponível em: www.influxdata.com.

InnovatorsGuru. *ZMPT101B Micro Precision Voltage Transformers*. 2020. <https://innovatorsguru.com/zmpt101b/>. Acesso em: 22 set. 2025.

International Energy Agency. *World Energy Outlook 2024*. Paris, 2024. Disponível em: https://www.iea.org/reports/world-energy-outlook-2024. Acesso em: 04 out. 2025.

JUNIOR, S. L. S.; FARINELLI, F. A. *DOMÓTICA-Automação Residencial e Casas Inteligentes com Arduíno e ESP8266*. [S.l.]: Saraiva Educação, 2018.

MAHBUB, M. et al. A review of smart home energy management systems. *IEEE Access*, v. 8, p. 118021–118044, 2020.

MASCHIETTO, L. G. et al. *Arquitetura e Infraestrutura de IoT*. Porto Alegre, RS: SAGAH, 2021.

MONK, S. *Raspberry Pi Cookbook: Software and Hardware Problems and Solutions*. 3rd. ed. [S.l.]: O'Reilly Media, 2019.

MORAES, A. d.; HAYASHI, V. T. *Segurança em IoT: Entenda os riscos e ameaças em Internet das Coisas*. Rio de Janeiro, RJ: Alta Books, 2021.

NASCIMENTO, L. M.; SOUZA, G. L.; GOMES, H. V. Desenvolvimento de um medidor de energia elétrica residencial de baixo custo com análise de qualidade de energia. In: *Anais do Simpósio Brasileiro de Sistemas Elétricos (SBSE)*. Online: Sociedade Brasileira de Automática (SBA), 2021. p. 1–6.

NOLêTO, I. A. *Desenvolvimento de medidor inteligente para energia elétrica e transferência de dados usando internet das coisas*. Monografia (Trabalho de Conclusão de Curso (Engenharia Elétrica)) — Universidade Federal Rural do Semi-Árido, Caraúbas, RN, 2023.

PEREIRA, F. *Monitoramento de Sistemas com Zabbix, Grafana e InfluxDB*. São Paulo, SP: Casa do Código, 2021.

PINHEIRO, P. R.; MARTINS, A. S.; GUEDES, L. A. Segurança e privacidade na internet das coisas: Uma revisão sistemática da literatura. In: *Anais do XIX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg)*. São Paulo, SP: Sociedade Brasileira de Computação (SBC), 2019. p. 1–14.

Railway. *Railway Documentation*. 2025. Disponível em: https://docs.railway.app/. Acesso em: 12 out. 2025.

SANTANA, A. R. *Big Data e Internet das Coisas: Uma Abordagem Prática*. São Paulo, SP: Érica, 2020.

SANTOS, B. P. et al. *Internet das Coisas: da Teoria à Prática*. Porto Alegre, RS: Sociedade Brasileira de Computação (SBC), 2016. (Minicursos do Congresso da Sociedade Brasileira de Computação).

SANTOS, N. S. d. A. et al. Desenvolvimento de um dispositivo iot para controle e monitoramento de consumo de energia. *Revista FT*, v. 27, n. 128, p. 1–26, 2023.

SILVA, A. C. d. *Desenvolvimento de um sistema de baixo custo para monitoramento de energia baseado em IoT*. Dissertação (Dissertação de Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, SC, 2021.

SILVA, L. A. M. et al. *Sistemas Embarcados: Hardware e Software na Prática*. 1. ed. Rio de Janeiro, RJ: LTC, 2019.

SOUZA, D. J. *Instrumentação Eletrônica: Princípios e Aplicações*. São Paulo, SP: Editora Érica, 2020.