Universidade de Brasília

Faculdade de Ciências e Tecnologias em Engenharia

Engenharia Eletrônica

Desenvolvimento de um Software Embarcado para Monitoramento de Módulos Fotovoltaicos

Matheus Alves do Nascimento

TRABALHO DE CONCLUSÃO DE CURSO

ENGENHARIA ELETRÔNICA

Brasília

2026

Universidade de Brasília
Faculdade de Ciências e Tecnologias em Engenharia
Engenharia Eletrônica

Desenvolvimento de um Software Embarcado para Monitoramento de Módulos Fotovoltaicos

Matheus Alves do Nascimento

Trabalho de Conclusão de Curso submetido como requisito parcial para obtenção do grau de Engenheiro Eletrônico.

Orientador: Prof. Dr. Alex Reis

Brasília
2026

FICHA CATALOGRÁFICA

Alves do Nascimento, Matheus.

Desenvolvimento de um Software Embarcado para Monitoramento de Módulos Fotovoltaicos / Matheus Alves do Nascimento; orientador Alex Reis. -- Brasília, 2026.

99 p.

Trabalho de Conclusão de Curso (Engenharia Eletrônica) -- Universidade de Brasília, 2026.

1. Módulos Fotovoltaícos. 2. Sistemas Embarcados. 3. Sensoriamento. I. Reis, Alex, orient. II. Título.

Universidade de Brasília
Faculdade de Ciências e Tecnologias em Engenharia
Engenharia Eletrônica

Desenvolvimento de um Software Embarcado para Monitoramento de Módulos Fotovoltaicos

Matheus Alves do Nascimento

Trabalho de Conclusão de Curso submetido como requisito parcial para obtenção do grau de Engenheiro Eletrônico.

Trabalho aprovado. Brasília, 21 de Julho de 2026:

Prof. Dr. Alex Reis,
UnB/FCTE
Orientador

Profª. Drª. Loana Nunes Velasco
UnB/FCTE
Avaliador 1

Prof. Dr. Luiz Carlos Gomes de Freitas
UFU/FEELT
Avaliador 2

Dedico a Deus, primeiramente, pela fé que sustenta meus sonhos e pela força que guia meu caminho. E também, aos meus pais, Kárita e Frederico, pilares da minha vida, que sempre apostaram em mim e confiaram no meu potencial. A vocês, minha eterna gratidão e amor.

Agradecimentos

Agradeço primeiramente a minha família por sempre estar ao meu lado e me dar forças nesta jornada.

Um agradecimento especial às pessoas que mais convivem comigo e que foram fundamentais para o meu desenvolvimento e suporte: meus pais, Kárita e Frederico; meus irmãos, Lucas e Henrique; meus avós, José Hélio, Cledalva e Telma; meu tio, Gustavo; vocês são a minha base e meu porto seguro. Agradeço especialmente à minha namorada, Flávia, que participou desse trabalho juntamente a mim, me apoiou e me motivou durante essa jornada. Agradeço aos meus amigos que estiveram comigo nessa jornada da graduação, Arthur Rodrigues, João Pedro e Giovanna.

Aos meus orientadores, Prof. Dr. Alex Reis, Prof. Dr. Loana Nunes Velasco e Prof. Dr. Renato Coral Sampaio, por sua paciência, sabedoria e incentivo, que foram essenciais para o meu desenvolvimento e sucesso deste trabalho.

À toda minha família, pelo amor e apoio incondicional; aos meus colegas e amigos que fiz durante a faculdade, pelo apoio e companheirismo dessa jornada.

À Universidade de Brasília e ao seu corpo docente, que promoveu o meu desenvolvimento intelectual e profissional nesses anos de graduação.

A cada um que, de alguma maneira, ajudou a tornar este trabalho possível, meu sincero agradecimento.

“Não creio que haja uma emoção mais intensa para um inventor do que ver suas criações funcionando. Essas emoções fazem você esquecer de comer, de dormir, de tudo.”

(Nikola Tesla)

Resumo

A crescente adoção da energia solar fotovoltaica exige sistemas de monitoramento robustos para garantir a eficiência, a segurança e a longevidade das instalações. Este trabalho apresenta o desenvolvimento de um software embarcado (*firmware*) dedicado ao monitoramento em tempo real de módulos fotovoltaicos, operando ao nível do módulo. O sistema foi desenvolvido com base no microcontrolador (SoC) ESP32-C6, aproveitando a sua conectividade Wi-Fi nativa e elevada eficiência energética. A arquitetura de *software* implementada é responsável pela aquisição de dados críticos do painel, nomeadamente tensão, corrente e temperatura, por meio de sensores dedicados e processamento de conversão analógico-digital (ADC). Para além da recolha contínua de telemetria, o *firmware* integra um algoritmo de detecção de falhas baseado em lógica de histerese. Esta abordagem permite identificar eventos críticos, como sobretensão e sobrecorrente, com elevada confiabilidade, mitigando o risco de falsos alarmes causados por ruídos ou oscilações transitórias. Os dados processados e os alertas de anomalias são transmitidos de forma assíncrona para uma plataforma externa utilizando o protocolo MQTT. O desenvolvimento incluiu a caracterização dos sensores e a calibração do sistema, demonstrando que a solução proposta atinge uma elevada precisão na medição das grandezas e uma resposta assertiva na sinalização de falhas. Este trabalho contribui para o avanço da manutenção preditiva e otimização do desempenho de sistemas de geração fotovoltaica.

Palavras-chave: Módulos Fotovoltaícos; Sistemas Embarcados; Sensoriamento.

Abstract

The increasing adoption of solar photovoltaic energy requires robust monitoring systems to ensure the efficiency, safety, and longevity of the installations. This work presents the development of embedded software (firmware) dedicated to the real-time monitoring of photovoltaic modules, operating at the module level. The system was developed based on the ESP32-C6 System-on-Chip (SoC), leveraging its native Wi-Fi connectivity and high energy efficiency. The implemented software architecture is responsible for acquiring critical panel data, namely voltage, current, and temperature, through dedicated sensors and analog-to-digital conversion (ADC) processing. In addition to continuous telemetry collection, the firmware integrates a fault detection algorithm based on hysteresis logic. This approach allows the identification of critical events, such as overvoltage and overcurrent, with high reliability, mitigating the risk of false alarms caused by noise or transient oscillations. The processed data and anomaly alerts are asynchronously transmitted to an external platform using the MQTT protocol. The development included sensor characterization and system calibration, demonstrating that the proposed solution achieves high accuracy in measuring the physical quantities and an assertive response in fault signaling. This work contributes to the advancement of predictive maintenance and performance optimization of photovoltaic generation systems.

Keywords: Photovoltaic Modules; Embedded Systems; Sensing .

Lista de figuras

Figura 2.1 Representação de uma curva I-V (Villalva, 2013) 26
Figura 2.2 Representação de uma curva P-V (Villalva, 2013) 26
Figura 2.3 Representação do modelo SDM (Selvaraj; Veerasamy; Krithiga, 2023) . . 27
Figura 2.4 Exemplo de uma tarefa para uma determinada função (Almeida, 2021) . 32
Figura 2.5 Exemplo de um fluxo de um software embarcado (Fonte: Autoria Própria) 36
Figura 3.1 Exemplo de um fluxo da metodologia utilizada (Fonte: Autoria Própria) 38
Figura 3.2 Devkit da ESP32C6 (Fonte: Espressif) 41
Figura 3.3 Circuito interno LV25-P (Fonte: (LEM, 2022b)) 42
Figura 3.4 Esquemático do LV25-P (Fonte: (Feijó, 2021)) 43
Figura 3.5 Diagrama interno do LA55-P (Fonte: (LEM, 2022a)) 43
Figura 3.6 Esquemático do LA55-P (Fonte: (Feijó, 2021)) 44
Figura 3.7 Esquemático do NXRT15XV103FA1B040 (Fonte: Autoria própria) . . . 45
Figura 3.8 Esquemático do Dispositivo contra surtos (Fonte: Autoria própria) . . . 45
Figura 3.9 Fluxograma de Firmware (Fonte: Autoria própria) 46
Figura 3.10 Método de agregação em multicamadas, baseado na norma IEC 61400-4-30 (Fonte: autoria própria). 48
Figura 3.11 Dinâmica de proteção: histerese com janelamento de validação e acionamento do MOSFET. Fonte: Autoria Própria 50
Figura 4.1 Placa de circuito impresso produzida para o projeto. Fonte: autoria própria 54
Figura 4.2 Disposição experimental da bancada de testes para caracterização do circuito de tensão. 60
Figura 4.3 Curva de caracterização do transdutor de tensão LV 25-P gerada via MATLAB.Fonte: autoria própria 61
Figura 4.4 Validação do envio de tensão em regime permanente.Fonte: autoria própria 61
Figura 4.5 Disposição do banco de cargas para a caracterização do sensor de corrente.Fonte: autoria própria 63
Figura 4.6 Curva de caracterização do transdutor de corrente gerada via MATLAB. 64
Figura 4.7 Validação do envio de corrente em regime permanente.Fonte: autoria própria 64
Figura 4.8 Montagem experimental para caracterização do sistema de temperatura. 65
Figura 4.9 Curva de resposta do sensor de temperatura gerada via MATLAB.Fonte: autoria própria 66
Figura 4.10 Comparação entre a temperatura medida e temperatura lida. Fonte: autoria própria 67

Figura 4.11 Gráfico gerado após uma ocorrência de evento de sobrecorrente. Fonte: autoria própria 69
Figura 4.12 Gráfico gerado após uma ocorrência de evento de sobretensão Fonte: autoria própria 70
Figura 4.13 Gráfico gerado após uma ocorrência de evento de sobretemperatura. Fonte: autoria própria 71
Figura 4.14 Forma de onda lida pela mudança de estado do pino a cada ISR do Timer. Fonte: autoria própria 74

Lista de tabelas

Tabela 3.1 Características Técnicas do Lv25-P (Fonte: (LEM, 2022b)) 42
Tabela 3.2 Características Técnicas do LA55-P (Fonte: (LEM, 2022a)) 44
Tabela 3.3 Características elétricas do Termistor NTC NXRT15XV103FA1B040(Fonte: (Murata, 2022)) 44
Tabela 4.1 Relação entre a tensão lida pelo ADC e a tensão de referência. 60
Tabela 4.2 Relação empírica entre a tensão lida pelo ADC e as correntes de referência. 63
Tabela 4.3 Relação entre a tensão lida pelo ADC e a temperatura de referência. . . . 66

Sumário

1 Introdução 17
1.1 Motivação 17
1.2 Estado da Arte 20
1.3 Objetivos Geral e Específicos 21
1.4 Contribuições 22
1.5 Estrutura do Trabalho 22

2 Referencial Teórico 24
2.1 Considerações Iniciais 24
2.2 Fundamentos de Módulos Fotovoltaicos 24
2.2.1 O Efeito fotovoltaico 24
2.2.2 Características Elétricas de uma Célula e Módulo Fotovoltaico . . . 25
2.2.3 Circuito equivalente 27
2.3 Fatores que acarretam a eficiência energética 28
2.3.1 Irradiância Solar 28
2.3.2 Temperatura do Módulo 29
2.3.3 Sombreamento 29
2.3.4 Sujeira e poeira 29
2.3.5 Degradação do Módulo ao Longo do Tempo 30
2.4 MLPE - Eletrônica de Potência em nível de módulo 30
2.5 Fundamentos de Sistemas Embarcados e IoT 31
2.5.1 Arquitetura de Sistemas Embarcados 31
2.5.2 Conceitos de Software Embarcado 35

3 Metodologia 38
3.1 Análise de Requisitos 39
3.1.1 Requisitos Funcionais 39
3.1.2 Requisitos Não Funcionais 40
3.2 Arquitetura de Hardware 41
3.2.1 Unidade de Processamento 41
3.2.2 Sensoriamento 42
3.2.3 Desconexão do módulo fotovoltaico 44
3.3 Software Embarcado 45
3.3.1 Fluxograma de firmware 45
3.3.2 Aquisição e Processamento 47
3.3.3 Detecção de Eventos 48

3.3.4 Comunicação e Envio de Dados . . . 51
3.3.5 Atuação sobre o módulo . . . 52

4 Resultados . . . 54
4.1 Considerações Iniciais . . . 54
4.2 Inicialização do firmware. . . . 54
4.2.1 Aquisição e processamento . . . 55
4.2.2 Comunicação . . . 56
4.3 Configuração do ADC . . . 57
4.3.1 One shot - Mode . . . 57
4.4 Regime Permanente . . . 59
4.4.1 Medição de tensão . . . 59
4.4.2 Medição de corrente . . . 62
4.4.3 Medição de temperatura . . . 65
4.5 Detecção de Eventos . . . 67
4.5.1 Evento de sobrecorrente. . . . 68
4.5.2 Evento de sobretensão . . . 69
4.5.3 Evento de sobretemperatura. . . . 70
4.6 Envio de dados . . . 71
4.6.1 Monitoramento em Regime Permanente . . . 71
4.6.2 Comunicação Bidirecional e Acionamento Remoto . . . 72
4.6.3 Registro de Eventos Críticos . . . 72
4.7 Validação da aquisição de 960 Hz. . . . 73

5 Conclusões . . . 76
5.1 Sugestões para Trabalhos Futuros . . . 76

Referências . . . 78

Apêndices 81

Apêndice A Código-fonte principal do sistema embarcado . . . 82

Apêndice B Código em MATLAB para caracterização dos circuitos dos sensores 97

Lista de abreviaturas e siglas

ADC	*Analog-to-Digital Converter* – Conversor Analógico-Digital (p. 28)
API	*Application Programming Interface* – Interface de Programação de Aplicação (p. 31)
CA	Corrente Alternada (p. 42)
CC	Corrente Contínua (p. 20)
CPU	*Central Processing Unit* – Unidade Central de Processamento (p. 28)
DDM	*Double Diode Model* – Modelo de Diodo Duplo (p. 22)
DRAM	*Dynamic Random-Access Memory* – Memória de Acesso Aleatório Dinâmica (p. 29)
EEPROM	*Electrically Erasable Programmable Read-Only Memory* – Memória Somente de Leitura Programável e Apagável Eletricamente (p. 28)
EPROM	*Erasable Programmable Read-Only Memory* – Memória Somente de Leitura Programável e Apagável (p. 28)
ESP-IDF	*Espressif IoT Development Framework* – Estrutura de Desenvolvimento IoT da Espressif (p. 49)
GPIO	*General Purpose Input/Output* – Entrada/Saída de Propósito Geral (p. 48)
GPS	*Global Positioning System* – Sistema de Posicionamento Global (p. 29)
I2C	*Inter-Integrated Circuit* – Circuito Inter-Integrado (p. 29)
IDE	*Integrated Development Environment* – Ambiente de Desenvolvimento Integrado (p. 49)
IoT	*Internet of Things* – Internet das Coisas (p. 14)
ISC	*Short-Circuit Current* – Corrente de Curto-Circuito (p. 20)
ISR	*Interrupt Service Routine* – Rotina de Serviço de Interrupção (p. 69)
MLPE	*Module-Level Power Electronics* – Eletrônica de Potência em Nível de Módulo (p. 25)

MOSFET *Metal-Oxide-Semiconductor Field-Effect Transistor* – Transistor de Efeito de Campo Metal-Óxido-Semicondutor (p. 34)

MPP *Maximum Power Point* – Ponto de Máxima Potência (p. 20)

MPPT *Maximum Power Point Tracking* – Rastreamento do Ponto de Máxima Potência (p. 20)

MQTT *Message Queuing Telemetry Transport* – Transporte de Telemetria de Enfileiramento de Mensagens (p. 35)

NMOS *N-channel MOSFET* – MOSFET de canal N (p. 39)

NTC *Negative Temperature Coefficient* – Coeficiente de Temperatura Negativo (p. 38)

NVS *Non-Volatile Storage* – Armazenamento Não Volátil (p. 48)

PC *Personal Computer* – Computador Pessoal (p. 26)

PCB *Printed Circuit Board* – Placa de Circuito Impresso (p. 49)

PROM *Programmable Read-Only Memory* – Memória Somente de Leitura Programável (p. 28)

PWM *Pulse Width Modulation* – Modulação por Largura de Pulso (p. 28)

RAM *Random-Access Memory* – Memória de Acesso Aleatório (p. 29)

ROM *Read-Only Memory* – Memória Somente de Leitura (p. 28)

RPC *Remote Procedure Call* – Chamada de Procedimento Remoto (p. 47)

RTOS *Real-Time Operating System* – Sistema Operacional de Tempo Real (p. 31)

SAR *Successive Approximation Register* – Registrador de Aproximação Sucessiva (p. 52)

SDM *Single Diode Model* – Modelo de Diodo Simples (p. 22)

SoC *System on Chip* – Sistema em um Chip (p. 36)

SPI *Serial Peripheral Interface* – Interface Periférica Serial (p. 29)

SRAM *Static Random-Access Memory* – Memória de Acesso Aleatório Estática (p. 29)

TCP/IP *Transmission Control Protocol/Internet Protocol* – Protocolo de Controle de Transmissão/Protocolo de Internet (p. 35)

TTL *Transistor-Transistor Logic* – Lógica Transistor-Transistor (p. 40)

UART *Universal Asynchronous Receiver-Transmitter* – Receptor/Transmissor Assíncrono Universal (p. 29)

VOC *Open-Circuit Voltage* – Tensão de Circuito Aberto (p. 20)

1 Introdução

1.1 Motivação

O setor elétrico brasileiro está passando por transformações significativas, impulsionadas pelo crescimento da demanda por energia, pela diversificação das fontes geradoras e pelo avanço contínuo das tecnologias. Esse cenário torna o ambiente nacional mais dinâmico e promissor, abrindo oportunidades para investidores, consumidores e agentes reguladores. A tendência é que o setor caminhe cada vez mais rumo à descentralização e à digitalização, ampliando o espaço para fontes renováveis e consolidando o Mercado Livre de Energia como uma alternativa viável e atrativa ((EPE), 2023).

Dentre as fontes renováveis, a energia solar tem se destacado de forma expressiva nos últimos anos. O Brasil vem ocupando posição de destaque no cenário global como um dos principais geradores de energia solar. Segundo dados da Associação Brasileira de Energia Solar Fotovoltaica (ABSOLAR), o setor tem potencial para atrair aproximadamente R$ 39,4 bilhões em investimentos e apresentar um crescimento de 25% até o final de 2025, ultrapassando a marca de 64 GW de capacidade instalada (ABSOLAR, 2024).

Com esse aumento significativo na geração solar, cresce também a demanda por novas instalações fotovoltaicas. Consequentemente, torna-se essencial o desenvolvimento de sistemas de monitoramento e automação que garantam uma operação eficiente, segura e sustentável dessas usinas. A automação e o monitoramento desempenham um papel fundamental para assegurar a confiabilidade do fornecimento de energia, reduzir perdas e facilitar a manutenção preditiva dos sistemas. Essas soluções contribuem diretamente para a ampliação da matriz energética brasileira de forma mais limpa, reduzindo a dependência de fontes hídricas e de combustíveis fósseis ((ANEEL), 2022).

O monitoramento de painéis fotovoltaicos permite a análise do desempenho do sistema por meio de variáveis elétricas como tensão e corrente. A partir desses parâmetros, é possível avaliar se a autonomia do painel está operando em sua capacidade máxima. Caso contrário, é viável identificar os fatores que estão comprometendo a eficiência do sistema, como sombreamentos, sujeira, falhas em módulos ou inversores, entre outros (Alves; Silva; Pereira, 2020).

Além disso, sistemas automatizados permitem o controle em tempo real de diferentes variáveis ambientais e elétricas, como temperatura dos módulos, irradiância solar e presença de falhas. O uso de tecnologias como IoT (Internet das Coisas) e aprendizado de máquina têm sido incorporadas em usinas modernas para melhorar a eficiência operacional e possibilitar respostas rápidas a anomalias (Pereira; Costa, 2021).

Portanto, estudos voltados ao desenvolvimento de métodos de monitoramento e automação de usinas fotovoltaicas são de grande relevância para o setor elétrico brasileiro. Eles representam um passo importante na consolidação de uma matriz energética mais diversificada, inteligente e sustentável, alinhada aos objetivos de desenvolvimento econômico e preservação ambiental.

O monitoramento da geração de energia solar possibilita uma gestão mais eficiente do sistema, fornecendo informações em tempo real sobre o desempenho dos principais componentes. Isso permite identificar com precisão o momento adequado para realizar intervenções, como a limpeza dos módulos fotovoltaicos, ajustes no ângulo de inclinação dos painéis ou manutenções corretivas no inversor. Ao adotar uma gestão proativa com base nesses dados, é possível manter o sistema operando em condições ideais e garantir a máxima eficiência energética.

Além disso, o monitoramento contínuo permite a realização de manutenções preditivas, reduzindo a necessidade de intervenções emergenciais. A análise dos dados operacionais viabiliza a identificação precoce de possíveis falhas ou desgastes nos equipamentos, o que possibilita ações corretivas antes que os problemas se agravem. Essa abordagem reduz significativamente os custos com reparos e amplia a vida útil dos componentes, contribuindo para a sustentabilidade e a confiabilidade do sistema.

Outro benefício importante é a agilidade no suporte técnico. Em situações de falhas ou mau funcionamento, o sistema de monitoramento permite localizar rapidamente a origem do problema, facilitando o diagnóstico remoto e reduzindo o tempo de resposta da equipe de manutenção. Isso evita longos períodos de inatividade, previne prejuízos na geração de energia e assegura que a usina solar continue operando de forma estável.

O monitoramento fornece dados precisos sobre a produção de energia, o que facilita o acompanhamento dos índices de economia e do retorno financeiro do investimento. Com acesso a relatórios detalhados, os proprietários conseguem avaliar o desempenho do sistema fotovoltaico, identificar oportunidades de otimização e tomar decisões mais assertivas sobre a expansão ou aprimoramento da usina solar.

Além dos benefícios operacionais e econômicos, o avanço tecnológico no setor fotovoltaico tem ampliado o acesso a ferramentas de monitoramento e automação cada vez mais eficientes e acessíveis. Com a popularização de microcontroladores, sensores inteligentes e plataformas baseadas em Internet das Coisas (IoT), tornou-se viável implementar soluções personalizadas e de baixo custo capazes de acompanhar o desempenho energético em tempo real. Esse cenário impulsiona iniciativas voltadas à inovação tecnológica, especialmente em contextos onde há limitações de infraestrutura ou necessidade de otimização contínua.

Nesse contexto, o desenvolvimento de sistemas inteligentes voltados à supervisão e controle de usinas solares apresenta-se como uma estratégia promissora para fortalecer a matriz energética nacional. A integração entre automação, monitoramento e energia reno-

vável representa uma contribuição significativa para a construção de um modelo energético mais sustentável, resiliente e descentralizado. Ao facilitar a gestão e o acompanhamento da geração solar, tais soluções incentivam o uso responsável dos recursos e fomentam a expansão das energias limpas em todo o país.

Atualmente, o mercado já dispõe de soluções comerciais avançadas de monitoramento e proteção, impulsionadas pela adoção da Eletrônica de Potência em Nível de Módulo (MLPE) e por normativas rigorosas de segurança operacional. Fabricantes de destaque, como a Tigo Energy e a Huawei, têm liderado a transição tecnológica rumo a sistemas fotovoltaicos mais seguros e inteligentes. A Tigo Energy, por exemplo, é pioneira no desenvolvimento de otimizadores e módulos de desligamento rápido (*Rapid Shutdown*), a exemplo de sua família de produtos TS4. Essa tecnologia permite a supervisão granular, a mitigação de perdas por sombreamento e, crucialmente, o seccionamento individualizado da tensão do painel em casos de emergência (TIGO ENERGY, 2023). De maneira complementar, a Huawei consolidou-se com ecossistemas que integram inversores *string* tradicionais a otimizadores inteligentes, a exemplo do modelo SUN2000-450W-P2. Além da otimização de potência (MPPT individual), esse dispositivo garante a desenergização instantânea do circuito de corrente contínua para níveis seguros (*Rapid Shutdown*), protegendo instalações e equipes de manutenção (HUAWEI, 2023).

Atualmente, o mercado já dispõe de soluções comerciais avançadas de monitoramento e proteção, impulsionadas pela adoção da Eletrônica de Potência em Nível de Módulo (MLPE) e por normativas rigorosas de segurança operacional. Fabricantes de destaque, como a Tigo Energy e a Huawei, têm liderado a transição tecnológica rumo a sistemas fotovoltaicos mais seguros e inteligentes. A Tigo Energy, por exemplo, é pioneira no desenvolvimento de otimizadores e módulos de desligamento rápido (*Rapid Shutdown*), a exemplo de sua família de produtos TS4. Essa tecnologia permite a supervisão granular, a mitigação de perdas por sombreamento e, crucialmente, o seccionamento individualizado da tensão do painel em casos de emergência (**??**). De maneira complementar, a Huawei consolidou-se com ecossistemas que integram inversores *string* tradicionais a otimizadores inteligentes, a exemplo do modelo SUN2000-450W-P2. Além da otimização de potência (MPPT individual), esse dispositivo garante a desenergização instantânea do circuito de corrente contínua para níveis seguros (*Rapid Shutdown*), protegendo instalações e equipes de manutenção (HUAWEI, 2023).

A implementação desses dispositivos atende aos padrões internacionais de segurança contra incêndios e eleva a confiabilidade da geração distribuída. Contudo, apesar da robustez oferecida pelas plataformas da Tigo e da Huawei, essas soluções operam predominantemente em ecossistemas fechados. Elas dependem de protocolos de comunicação proprietários, frequentemente utilizando *Power Line Communication* (PLC), e restringem o acesso aos dados brutos de telemetria às suas próprias plataformas em nuvem. Essa característica

não apenas impõe custos elevados de aquisição, mas também limita a flexibilidade para customização e integração com infraestruturas locais de automação.

Atualmente, o mercado já dispõe de soluções comerciais avançadas de monitoramento e proteção, impulsionadas pela adoção da Eletrônica de Potência em Nível de Módulo (MLPE) e por normativas rigorosas de segurança operacional. Fabricantes de destaque, como a Tigo Energy e a Huawei, têm liderado a transição tecnológica rumo a sistemas fotovoltaicos mais seguros e inteligentes. A Tigo Energy, por exemplo, é pioneira no desenvolvimento de otimizadores e módulos de desligamento rápido (*Rapid Shutdown*), a exemplo de sua família de produtos TS4. Essa tecnologia permite a supervisão granular, a mitigação de perdas por sombreamento e, crucialmente, o seccionamento individualizado da tensão do painel em casos de emergência (TIGO ENERGY, 2023). De maneira complementar, a Huawei consolidou-se com ecossistemas que integram inversores *string* tradicionais a otimizadores inteligentes, a exemplo do modelo SUN2000-450W-P2. Além da otimização de potência (MPPT individual), esse dispositivo garante a desenergização instantânea do circuito de corrente contínua para níveis seguros (*Rapid Shutdown*), protegendo instalações e equipes de manutenção (HUAWEI, 2023).

A implementação desses dispositivos atende aos padrões internacionais de segurança contra incêndios e eleva a confiabilidade da geração distribuída. Contudo, apesar da robustez oferecida pelas plataformas da Tigo e da Huawei, essas soluções operam predominantemente em ecossistemas fechados. Elas dependem de protocolos de comunicação proprietários, frequentemente utilizando *Power Line Communication* (PLC), e restringem o acesso aos dados brutos de telemetria às suas próprias plataformas em nuvem. Essa característica não apenas impõe custos elevados de aquisição, mas também limita a flexibilidade para customização e integração com infraestruturas locais de automação.

1.2 Estado da Arte

O estado da arte é fundamental para mapear as pesquisas em andamento, tanto no meio acadêmico quanto no mercado de trabalho, visto que muitas delas estão diretamente interligadas. Ao examinar trabalhos acadêmicos, patentes e produtos comerciais, busca-se identificar as abordagens existentes, as tecnologias empregadas, os resultados alcançados e, crucialmente, as lacunas ainda presentes na literatura e no mercado.

Devido à alta demanda por monitoramento de módulos fotovoltaicos, a referência (Felipe T.A. ; Melo, 2021) propôs um sistema capaz de detectar o mau funcionamento decorrente de causas acidentais e/ou técnicas, por meio da análise de curvas I-V (corrente-tensão). Nesse contexto, é notável que causas acidentais podem levar à redução da produtividade energética devido ao acúmulo excessivo de sujeira nos módulos fotovoltaicos, sombreamento parcial e erros eventuais durante a instalação.

Tendo em vista que os módulos fotovoltaicos são susceptíveis às mudanças climáticas e à exposição ao meio ambiente, é necessário monitorá-los para saber o seu real estado. Em (Silva* Fernando C. Melo, 2018), foram analisados 48 painéis fotovoltaicos de 5 modelos diferentes. A partir das curvas I-V, foi possível analisar a condição real do painel, verificando se estava sujo, em mau funcionamento, entre outros aspectos.

A fim de monitorar módulos fotovoltaicos, surge a necessidade de definir como transmitir esses dados e o que irá processá-los. Na pesquisa realizada pela Universidade Federal do Ceará (Pereira Ivonne M. Duponta, 2017), uma *Raspberry Pi* foi utilizada como computador para todo o processamento dos dados, e um módulo *IoT* para a aquisição dos dados relevantes. Um sistema interligado com Wi-Fi permite que qualquer usuário possa saber o que está acontecendo com o sistema, monitorando, assim, se a eficiência dos módulos está satisfatória.

Para abordar a necessidade de monitoramento eficiente em sistemas fotovoltaicos, (Pereira Sandro C.S. Jucá b, 2019) propuseram uma rede de monitoramento baseada em Internet das Coisas (*IoT*), visando reduzir os custos de *dataloggers* comerciais e módulos de sensoriamento. O sistema desenvolvido utiliza *software* livre e se conecta a um servidor na nuvem via Wi-Fi, permitindo a consulta online de dados e o monitoramento em tempo real da temperatura dos módulos fotovoltaicos, irradiação solar, temperatura ambiente, umidade relativa e velocidade do vento. A inovação do projeto reside na utilização de módulos ESP 32 e ESP 8266 de baixo custo para sensoriamento, processamento e transmissão de dados, o que possibilita a implementação de uma rede de monitoramento modular e expansível em diferentes locais.

Com isso, é notória a necessidade de monitorar módulos fotovoltaicos, a fim de garantir maior eficiência e promover a sustentabilidade. Pesquisas como as citadas anteriormente servem para reforçar que o monitoramento de qualquer dispositivo é essencial para todos.

1.3 Objetivos Geral e Específicos

Esse trabalho tem por objetivo o desenvolvimento e testes de um *software* embarcado, com enfoque na coleta e processamento de grandezas elétricas (tensão e corrente) e físicas (temperatura e vibração) para monitoramento de funcionamento de módulos fotovoltaicos.

Para isso, destacam-se os objetivos específicos:

- Monitorar níveis de tensão, corrente, temperatura e vibração durante o funcionamento do módulo;
- Monitorar comportamentos anormais do módulo, acarretando sua vida útil;
- Implementar lógicas para detecção de condições anormais de funcionamento e desconexão do módulo;

- Criar algoritmos de análise para calcular a eficiência dos módulos fotovoltaicos com base nos dados coletados, identificando desvios e anomalias;

1.4 Contribuições

O sistema de monitoramento e automação para módulos fotovoltaicos proposto neste projeto oferece contribuições significativas em termos de eficiência e segurança. Ao extrair e processar dados físicos e elétricos em tempo real, o sistema permite identificar e corrigir rapidamente desvios de desempenho, como sombreamento ou sujeira, o que otimiza a geração de energia e viabiliza uma manutenção preditiva. Isso não só maximiza a produção energética, mas também reduz custos operacionais, minimizando perdas e o tempo de inatividade dos sistemas fotovoltaicos.

Adicionalmente, o projeto eleva a segurança operacional e impulsiona a inovação. A capacidade de interromper automaticamente a operação em caso de mau funcionamento é crucial, protegendo equipamentos de danos e, mais importante, garantindo a segurança de pessoas. Essa funcionalidade previne acidentes e prolonga a vida útil dos módulos. Além disso, o desenvolvimento de um sistema inteligente contribui para o avanço tecnológico no setor de energias renováveis, servindo como uma base valiosa para futuras pesquisas em otimização, diagnóstico de falhas e aplicação de inteligência artificial em sistemas fotovoltaicos, promovendo a sustentabilidade e a eficiência energética.

1.5 Estrutura do Trabalho

Este documento é dividido em cinco capítulos, conforme detalhado a seguir:

- **Capítulo 1 - Introdução:** Neste capítulo, são abordadas as motivações que foram levantadas para a elaboração deste trabalho, as quais foram importantes para o seguimento dele. Assim, detalhando os objetivos principais e específicos, pode-se começar a traçar um caminho de pesquisa e desenvolvimento. Além disso, no estado da arte são apresentados trabalhos e teses na literatura que já tiveram o mesmo objetivo ou ideia deste trabalho, usados como inspiração ou até mesmo como suporte em determinados segmentos. E, por fim, as contribuições que este trabalho pode trazer para o meio acadêmico e social.
- **Capítulo 2 - Referencial Teórico:** Este capítulo aborda conceitos determinantes para o desenvolvimento do trabalho. Inicialmente, apresenta-se a teoria relacionada ao funcionamento específico de módulos fotovoltaicos, como funcionam, como é gerada energia, o que pode ser medido. Com isso, passa-se para a fase do monitoramento, na qual se apresenta a teoria de sistemas embarcados no geral e como os elementos dos módulos se conectam com a parte embarcada, podendo assim monitorar e automatizar.

- **Capítulo 3 - Metodologia:** Neste capítulo, apresenta-se a abordagem desenvolvida para o monitoramento e automação dos módulos. No qual é feito um estudo de caso, se faz um levantamento do que precisa ser feito e assim começa a parte de integração de sistemas.
- **Capítulo 4 - Resultados:** Este capítulo apresenta e analisa os resultados obtidos com a aplicação da metodologia proposta. Primeiramente, mostra-se o que foi desenvolvido durante o processo do trabalho. Mostram-se os resultados que foram obtidos e assim pode-se seguir para uma nova etapa.
- **Capítulo 5 - Conclusões:** No último capítulo, são destacadas as principais conclusões do estudo, que inclui as contribuições científicas e práticas do modelo proposto para o monitoramento e automatização. Além disso, são discutidas as limitações do trabalho, como a dependência de grandezas elétricas, e sugestões para estudos futuros.

2 Referencial Teórico

2.1 Considerações Iniciais

Neste capítulo, os fundamentos teóricos empregados neste trabalho são apresentados para fornecer uma base para a compreensão a respeito do monitoramento de módulos fotovoltaicos e sistemas embarcados.

Os conteúdos de cada seção deste capítulo são descritos a seguir:

- **Seção 2.2:** apresenta os fundamentos sobre o funcionamento dos módulos fotovoltaicos.
- **Seção 2.3:** explora a análise de geração de energia dos módulos fotovoltaicos a respeito da eficiência;
- **Seção 2.4:** aborda os fundamentos de sistemas embarcados e IoT;
- **Seção 2.5:** detalha o processo de monitoramento e automação juntamente com os módulos;

2.2 Fundamentos de Módulos Fotovoltaicos

2.2.1 O Efeito fotovoltaico

O efeito fotovoltaico é o fenômeno físico pelo qual a luz é convertida diretamente em eletricidade em materiais semicondutores. É o princípio fundamental por trás do funcionamento das células e módulos solares fotovoltaicos. Esse processo ocorre sem a necessidade de partes móveis, o que confere aos sistemas fotovoltaicos alta confiabilidade e baixa necessidade de manutenção.

Em sua essência, o efeito fotovoltaico baseia-se na capacidade de certos materiais semicondutores, mais comumente o silício, de absorver fótons (partículas de luz) e liberar elétrons, gerando uma corrente elétrica. Para que isso aconteça de forma controlada e útil, as células solares são fabricadas com uma estrutura específica: a junção P-N.

Um material semicondutor puro, como o silício, é dopado com impurezas para criar duas regiões distintas:

- Região Tipo-P (Positiva): Dopada com elementos que possuem menos elétrons na camada de valência (como o boro), criando "lacunas"ou "buracos"que agem como portadores de carga positiva.
- Região Tipo-N (Negativa): Dopada com elementos que possuem mais elétrons na camada de valência (como o fósforo), criando um excesso de elétrons livres, que agem como portadores de carga negativa.

Quando as regiões P e N de um semicondutor são unidas, formam a junção P-N, criando uma região de depleção com um campo elétrico interno que evita a recombinação de elétrons e lacunas. Ao receber luz solar, os fótons energizam os elétrons, gerando pares elétron-lacuna. O campo elétrico da junção, então, separa esses portadores, enviando elétrons para a região N e lacunas para a região P. Essa separação de cargas estabelece uma diferença de potencial (tensão); ao conectar um circuito externo, os elétrons fluem, gerando corrente elétrica contínua (CC), o que caracteriza o efeito fotovoltaico.

2.2.2 Características Elétricas de uma Célula e Módulo Fotovoltaico

Para monitorar e otimizar o desempenho de um sistema fotovoltaico, é essencial compreender as características elétricas fundamentais que descrevem o comportamento de uma célula e, consequentemente, de um módulo fotovoltaico. Essas características são tipicamente apresentadas por meio das **curvas I-V (Corrente-Tensão)** e **P-V (Potência-Tensão)**, que ilustram a relação entre a corrente e a tensão de saída sob diferentes condições de operação.

A **tensão de circuito aberto (V_{OC})** é a tensão máxima que uma célula ou módulo fotovoltaico pode produzir quando não há fluxo de corrente, ou seja, quando seus terminais estão desconectados de qualquer carga. Nesse ponto, a corrente é nula ($I = 0$). A V_{OC} é inversamente proporcional à temperatura, o que significa que, à medida que a temperatura do módulo aumenta, sua tensão de circuito aberto tende a diminuir. Sua dependência da irradiância solar é relativamente baixa.

A **corrente de curto-circuito (I_{SC})** representa a corrente máxima que uma célula ou módulo pode gerar quando seus terminais estão diretamente conectados (curto-circuito), resultando em tensão nula ($V = 0$). A I_{SC} é diretamente proporcional à intensidade da luz solar (irradiância) incidente no módulo, aumentando com a maior incidência de fótons. Sua dependência da temperatura é menos acentuada, tendendo a um ligeiro aumento com o aumento da temperatura.

O **Ponto de Máxima Potência (MPP)** é a condição de operação ideal na qual a célula ou módulo fotovoltaico entrega a maior quantidade de potência elétrica. Neste ponto, o produto da tensão (V_{MPP}) pela corrente (I_{MPP}) atinge seu valor máximo (P_{MPP}). A identificação e o rastreamento contínuo do MPP são cruciais para maximizar a eficiência de um sistema fotovoltaico, uma tarefa tipicamente realizada por algoritmos de **MPPT (Maximum Power Point Tracking)** em controladores de carga ou inversores.

As **curvas I-V (Corrente-Tensão)** e **P-V (Potência-Tensão)** são representações gráficas do desempenho elétrico de uma célula ou módulo fotovoltaico.

- A **curva I-V** mostra a relação entre a corrente de saída (eixo y) e a tensão de saída (eixo x) do dispositivo. Nela, I_{SC} é o ponto em que a curva intercepta o eixo da corrente

(V=0), e V_{OC} é o ponto onde a curva intercepta o eixo da tensão (I=0). A Figura 2.1 é um exemplo de uma curva I-V.

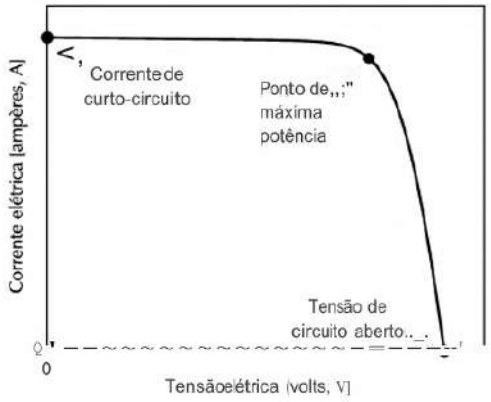


Figura 2.1 – Representação de uma curva I-V (Villalva, 2013)

- A **curva P-V** é derivada da curva I-V, plotando a potência de saída (produto de V por I) em função da tensão. O ponto mais alto desta curva representa o **MPP**. A Figura 2.2 apresenta um exemplo de P-V.



Figura 2.2 – Representação de uma curva P-V (Villalva, 2013)

É fundamental entender que essas curvas **variam significativamente** com as condições ambientais:

- **Irradiância Solar:** Um aumento na irradiância desloca a curva I-V para cima, aumentando a I_{SC} e, consequentemente, aP_{MPP}. A V_{OC} é menos afetada.
- **Temperatura do Módulo:** Um aumento na temperatura desloca a curva I-V para a esquerda e para baixo. Isso causa uma redução mais pronunciada na V_{OC} e uma pequena alteração na I_{SC}, resultando em uma diminuição da P_{MPP}.

A **eficiência (η)** de uma célula ou módulo fotovoltaico é uma medida de sua capacidade de converter a energia solar incidente em energia elétrica. É calculada como a razão entre a potência elétrica de saída no MPP (P_{MPP}) e a potência da luz solar incidente sobre a área do

módulo, geralmente expressa em porcentagem:

$$\eta = \frac{\mathrm{P_{MPP}}}{\text{Área do Módulo} \times \text{Irradiância}} \times 100\%$$

A eficiência é normalmente especificada sob **Condições Padrão de Teste (CST)**, que serão detalhadas em uma seção posterior.

Os fabricantes de módulos fotovoltaicos fornecem **coeficientes de temperatura** para a tensão, corrente e potência. Estes coeficientes quantificam a variação percentual de cada parâmetro por grau Celsius de mudança na temperatura da célula. Tipicamente, o coeficiente de temperatura da potência (%/°C) é negativo, indicando que a potência de saída do módulo diminui à medida que sua temperatura aumenta, um fator crítico para o desempenho em climas quentes.

2.2.3 Circuito equivalente

Para se entender o funcionamento específico de um módulo fotovoltaico, pode-se reduzi-lo a um circuito equivalente. No qual, conceitos básicos de circuitos elétricos servem para explicar o seu funcionamento. Existem vários tipos de circuitos equivalentes para representar módulos fotovoltaicos, um dos principais são o modelo de *Single Diode Model* (SDM) e *Double diode Model*(DDM).

2.2.3.1 *Single Diode Model* (SDM)

O *Single Diode Model* pode ser representado por dois componentes passivos (resistores) junto a um diodo ideal e uma fonte de corrente. A Figura 2.3 pode ser apresentada como um exemplo do circuito equivalente do tipo SDM.

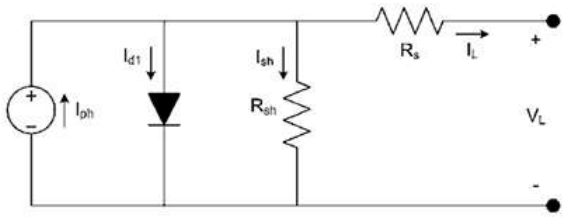


Figura 2.3 – Representação do modelo SDM (Selvaraj; Veerasamy; Krithiga, 2023)

O circuito equivalente do Modelo de Diodo Simples é composto por cinco elementos que buscam representar os fenômenos físicos e as perdas elétricas de uma célula ou módulo fotovoltaico:

1. **Fonte de Corrente Fotogerada (I_{ph}):** Representa a corrente gerada diretamente pela absorção de fótons da luz solar.

2. **Diodo (D_1):** Modeliza a junção P-N da célula fotovoltaica, que é responsável pelo efeito fotovoltaico. Ele representa a recombinação de portadores de carga (elétrons e lacunas) que ocorre dentro da camada de depleção e nas regiões neutras da célula.
3. **Resistência em Série (R_s):** Representa as perdas que ocorrem devido à resistência interna dos materiais semicondutores, dos contatos metálicos, das interconexões entre as células e da fiação do módulo.
4. **Resistência em paralelo (R_{sh}ouR_p):** também conhecida como resistência de *shunt* ou de derivação. Representa as correntes de fuga (ou correntes de desvio) que podem ocorrer por meio de imperfeições na junção P-N ou por curtos-circuitos parciais entre as camadas da célula.

A equação fundamental que descreve a corrente de saída (I) em função da tensão de saída (V) para o Modelo de Diodo Simples é dada por:

$$I = I_{ph} - I_0 \left[e^{\left(\frac{V+IR_s}{nV_t}\right)} - 1 \right] - \frac{V + IR_s}{R_{sh}}$$

Onde:

- I: Corrente de saída do módulo (A).
- V: Tensão de saída do módulo (V).
- I_{ph}: Corrente fotogerada (A).
- I_0: Corrente de saturação inversa do diodo (A).
- n: Fator de idealidade do diodo (adimensional, geralmente entre 1 e 2).
- R_s: Resistência em série (Ω).
- R_{sh}: Resistência em paralelo (Ω).
- V_t: Tensão térmica da célula (V), calculada como $V_t = \frac{N_s k T}{q}$.
  - N_s: Número de células conectadas em série no módulo.
  - k: Constante de Boltzmann (1.38×10^{-23} J/K).
  - T: Temperatura absoluta da célula (em Kelvin).
  - q: Carga elementar do elétron (1.602×10^{-19} C).

2.3 Fatores que acarretam a eficiência energética

A potência dos módulos fotovoltaicos não é constante, é variada de acordo com a irradiância sobre o módulo, a temperatura, fatores climáticos, sujeira e degradação do mesmo ao longo do tempo.

2.3.1 Irradiância Solar

A irradiância solar é a quantidade de potência da luz solar que incide sobre uma determinada área. É o fator mais óbvio e direto que afeta a geração de energia.

A intensidade da luz solar varia ao longo do dia (maior ao meio-dia solar) e das estações do ano (maior no verão, menor no inverno). Nuvens, neblina, chuva e poluição atmosférica podem reduzir significativamente a irradiância que atinge a superfície do módulo, diminuindo a corrente de curto-circuito (I SC) e, consequentemente, a potência gerada. A eficiência é máxima quando os raios solares incidem perpendicularmente à superfície do módulo. Desvios desse ângulo (devido à inclinação do módulo, orientação ou movimento do sol) reduzem a área efetiva de captação de luz.

2.3.2 Temperatura do Módulo

Ao contrário do que se pode pensar, temperaturas elevadas são prejudiciais à eficiência dos módulos fotovoltaicos, especialmente os baseados em silício cristalino.

O principal impacto da temperatura é na tensão de circuito aberto (V_{OC}), que diminui à medida que a temperatura da célula aumenta. Os fabricantes especificam coeficientes de temperatura (geralmente negativos para tensão e potência), indicando a porcentagem de perda de potência por cada grau Celsius acima de 25ºC.Um sistema de instalação com boa ventilação e espaçamento adequado entre o módulo e a superfície de montagem pode ajudar a mitigar o superaquecimento e manter a eficiência.

2.3.3 Sombreamento

O sombreamento, mesmo que parcial, é um dos fatores mais críticos e prejudiciais à eficiência de um arranjo fotovoltaico.

Em módulos com células conectadas em série, a célula sombreada se comporta como uma resistência (ou até mesmo uma carga), limitando a corrente de todo o string à sua própria capacidade reduzida. Isso leva a uma queda drástica na potência total do módulo ou arranjo. O sombreamento parcial pode causar o aquecimento excessivo das células sombreadas pois elas começam a dissipar energia em vez de produzi-la, podendo causar danos permanentes ao módulo. A presença de diodos de bypass (contornamento) em módulos modernos ajuda a minimizar o impacto do sombreamento, desviando a corrente em torno de grupos de células sombreadas.

2.3.4 Sujeira e poeira

O acúmulo de sujeira, poeira, folhas, dejetos de pássaros e outras partículas na superfície dos módulos bloqueia a luz solar de atingir as células.

A sujeira atua como uma barreira física, diminuindo a quantidade de fótons que chegam ao semicondutor, o que se traduz em perdas de corrente e, consequentemente, de potência. A manutenção e limpeza periódica dos módulos são essenciais para garantir que a superfície esteja desobstruída e maximizar a captação de luz. A frequência de limpeza depende da

localização e das condições ambientais (ex: áreas urbanas com poluição, zonas rurais com poeira).

Módulos fotovoltaicos são projetados para durar décadas, mas sofrem uma degradação natural em seu desempenho ao longo do tempo.

2.3.5 Degradação do Módulo ao Longo do Tempo

A degradação induzida pela luz ocorre nas primeiras horas ou dias de exposição à luz solar, especialmente em células de silício tipo-p dopadas com boro e oxigênio. Essa perda, geralmente entre 1% e 4%, é considerada no cálculo da potência nominal do módulo. Além dessas perdas iniciais, segundo (International Electrotechnical Commission, 2024) os módulos apresentam uma degradação linear esperada, geralmente de cerca de 0,5% a 0,8% ao ano, garantindo que a potência de saída permaneça acima de um determinado percentual após 25 anos.

2.4 MLPE - Eletrônica de Potência em nível de módulo

MLPE, ou Module-Level Power Electronics (Eletrônica de Potência em Nível de Módulo), refere-se a sistemas eletrônicos sofisticados integrados em cada painel solar de uma instalação fotovoltaica. O principal objetivo do MLPE é otimizar a produção de energia e a eficiência geral do sistema solar.

A tecnologia MLPE busca otimizar e monitorar a geração de energia módulo a módulo, buscando a maior potência para o funcionamento de cada módulo e, assim, garantindo o rendimento mesmo quando há diferenças na geração entre os módulos. A MLPE é acionada para garantir a máxima potência, evitando perdas no rendimento.

As vantagens da MLPE incluem:

- **Otimização de Desempenho em Condições Desfavoráveis:** É altamente eficaz em condições de sombreamento, onde conversores MLPE são utilizados para mitigar a perda de energia. Comparado a sistemas centralizados, pode gerar 25% a 35% mais energia em condições de sombreamento.
- **Monitoramento e Solução de Problemas a Nível de Módulo:** Permite o monitoramento e a solução de problemas em nível de módulo de forma muito eficiente.
- **Desligamento Rápido (Segurança):** Proporciona um desligamento rápido no nível do módulo em caso de perigo de incêndio ou outros eventos, cumprindo os requisitos do Código Elétrico Nacional (NEC).
- **Mitigação de Perdas:** Reduz perdas por incompatibilidade, deterioração e sombreamento parcial, que são problemas significativos em sistemas FV.

De acordo com o artigo (Sarwar MuhammadYaqoobJaved1, 2022), uma aplicação direta seria em uma instalação fotovoltaica residencial ou comercial que sofre com sombreamento parcial. Nesses cenários, painéis solares individuais podem ser sombreados por árvores, chaminés, prédios vizinhos ou até mesmo por sujeira e detritos. Sem o MLPE, um sistema fotovoltaico centralizado tradicional teria sua produção de energia significativamente reduzida, pois o desempenho de todo o "string"de painéis seria limitado pelo módulo de pior desempenho. Com a implementação do MLPE, através de microinversores ou otimizadores de potência instalados em cada painel, é possível otimizar a geração de energia individualmente em cada módulo. Isso garante que, mesmo sob sombreamento parcial, os painéis não sombreados continuem a produzir sua máxima potência, recuperando de 25% a 35% mais energia em comparação com sistemas centralizados.

2.5 Fundamentos de Sistemas Embarcados e IoT

Um sistema embarcado é um sistema que une elementos de Hardware/Software para realizar um processo ou uma tarefa específica. É projetado para ser integrado em outros sistemas ou subsistemas, visando controlar ou monitorar uma determinada função ou processo. Em grande parte do mundo, os sistemas embarcados são projetados para serem compactos, eficientes e confiáveis, visando uma longa duração em relação ao seu funcionamento.

Nos dias atuais, há uma ampla variação de áreas em que os sistemas embarcados estão presentes, desde a área automotiva, dispositivos médicos, aparelhos eletrônicos, entre outros. Eles desempenham um papel fundamental no cotidiano e são cada vez mais sofisticados e capazes de realizar tarefas que são cruciais para nossas vidas.

Sistemas embarcados, no geral, precisam de um poder computacional para realizar tais processos para os quais foram projetados. Esse poder computacional pode ser tanto usado por microcontroladores quanto por microprocessadores, dependendo do nível de processamento necessário.

2.5.1 Arquitetura de Sistemas Embarcados

Computadores embarcados, frequentemente referidos como microcontroladores ou sistemas em um único chip, são dispositivos projetados com um processador dedicado a executar uma aplicação muito específica. Diferentemente dos computadores de uso geral, esses sistemas integram, em menor escala, os componentes essenciais de um computador em um só chip para cumprir sua finalidade dedicada.

Em projetos de sistemas eletrônicos, esses dispositivos são empregados com a finalidade primária de executar uma ou múltiplas aplicações que funcionam como um sistema coeso. Em outras palavras, eles contêm um programa embarcado que permite a interação com o ambiente externo.

Em um microcontrolador existe uma série de periféricos necessários para que possam ser utilizados de forma correta. Com isso, teve-se a ideia de unir tudo o que é necessário para controlar em uma coisa só, chamado microcontrolador. Onde sua arquitetura se baseia em uma ULA (unidade de processamento), um conjunto de memórias voláteis e não voláteis e uma série de periféricos de acordo com a necessidade.

2.5.1.1 ULA (Unidade de processamento)

A unidade de processamento gerencia todas as tarefas e os recursos disponíveis em um sistema. A Unidade Lógica e Aritmética (ULA), que atua como o processador central, é um circuito digital sequencial. Isso significa que o seu estado atual, em conjunto com os dados de entrada, determina o próximo estado de operação do processador. Essa progressão entre estados é orquestrada por um circuito de sincronismo, mais conhecido como clock, e o intervalo de tempo entre dois estados consecutivos é chamado de ciclo.

O roteiro que indica ao processador o que fazer é definido por um conjunto de comandos, onde cada comando é uma instrução. Uma única instrução pode envolver múltiplas operações realizadas pelo microprocessador. O tempo que o processador leva para concluir uma instrução é determinado pela quantidade de ciclos necessários para executá-la, além da duração de cada ciclo. Algumas instruções podem operar com valores armazenados na memória, que são denominados dados. A figura 2.4 exemplifica como o processador executa uma tarefa de acordo com o sincronismo.

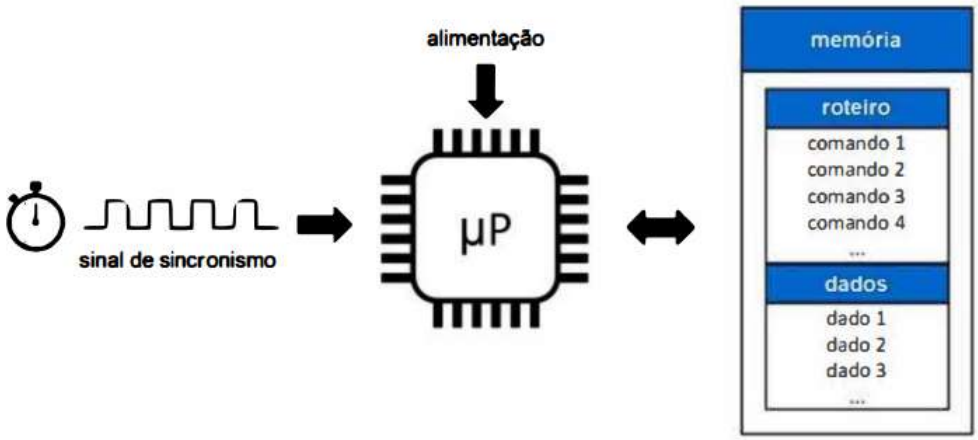


Figura 2.4 – Exemplo de uma tarefa para uma determinada função (Almeida, 2021)

A respeito do modo de funcionamento da grande maioria dos processadores atuais, existem duas arquiteturas que são baseadas: Von neumman e Harvard.

Proposta por John Von Neumann em 1945, corresponde ao modelo mais tradicional e amplamente utilizado em computadores de propósito geral, como PCs e servidores. Sua característica fundamental é a memória única compartilhada, onde tanto as instruções do programa quanto os dados são armazenados no mesmo espaço físico de memória. Isso

proporciona grande flexibilidade, pois a memória pode ser alocada dinamicamente conforme a necessidade de programas ou dados. No entanto, o acesso a essa memória compartilhada é feito por um único barramento, o que cria um "gargalo". O processador não consegue buscar uma instrução e um dado simultaneamente, precisando esperar pelo acesso à memória, o que limita a velocidade de processamento e é conhecido como o Gargalo de Von Neumann. Apesar dessa desvantagem de desempenho, a simplicidade de design e a flexibilidade a tornam ideal para sistemas que não exigem alta velocidade em tempo real.

Originalmente desenvolvida para o computador Harvard Mark I na década de 1940, esta arquitetura se diferencia por sua abordagem na organização da memória. A principal característica da arquitetura Harvard é a utilização de memórias fisicamente separadas para armazenar instruções e dados, cada uma com seu próprio barramento de acesso independente para a CPU. Essa separação permite que o processador acesse uma instrução da memória de programa e um dado da memória de dados simultaneamente. A maior vantagem é a velocidade, pois elimina o gargalo de Von Neumann, permitindo uma execução mais rápida das operações. Embora o design seja mais complexo e a divisão fixa da memória possa ser menos flexível, a arquitetura Harvard é amplamente utilizada em microcontroladores e processadores de sinais digitais (DSPs), onde alta velocidade de execução e previsibilidade em tempo real são cruciais. Muitas implementações modernas usam uma arquitetura Harvard modificada, que combina os benefícios de acesso paralelo com alguma flexibilidade para acessar a memória de instruções como dados, quando necessário.

2.5.1.2 Memória

A memória em sistemas embarcados é geralmente integrada diretamente no chip do microcontrolador e é projetada para atender a requisitos rigorosos de custo, consumo de energia e desempenho. Ela é categorizada em dois tipos principais: Memória de Programa e Memória de Dados.

A memória do tipo não volátil é onde o código do software (firmware) do microcontrolador é armazenado. Por ser uma memória não volátil, seu conteúdo permanece intacto mesmo quando a alimentação elétrica é desligada. Essa característica é vital, pois permite que o sistema embarcado inicie suas operações assim que é energizado. O tipo mais comum atualmente é a Memória Flash, que permite que o programa seja gravado e apagado eletricamente, facilitando o desenvolvimento e as atualizações de firmware. Outros tipos históricos incluem ROM, PROM, EPROM e EEPROM, sendo esta última (EEPROM) ainda relevante para armazenar parâmetros de configuração ou dados de calibração que precisam ser persistentes e ocasionalmente modificáveis, como configurações de rede ou limites operacionais de um sistema.

A Memória de Dados (Volátil) é utilizada para armazenar informações temporárias que o processador precisa acessar e modificar rapidamente durante a execução do programa.

Sendo uma memória volátil, todo o seu conteúdo é perdido quando a alimentação elétrica é removida. A RAM (Random Access Memory) é o principal tipo de memória de dados, funcionando como a memória de trabalho do processador. Nela são armazenadas as variáveis do programa, a pilha (stack) para variáveis locais de funções e endereços de retorno, e a área de dados dinâmicos (heap) para alocações de memória em tempo de execução. A SRAM (Static RAM) é o tipo de RAM mais comum em microcontroladores devido à sua velocidade e à não necessidade de refrescamento periódico, como a DRAM.

2.5.1.3 Periféricos

Para que microcontroladores e processadores possam interagir com o ambiente externo e cumprir suas funções dedicadas, eles contam com uma série de módulos de entrada e saída, conhecidos como periféricos. Esses módulos são componentes essenciais que integram o microcontrolador, permitindo-o monitorar dispositivos ou agentes externos e controlá-los. De acordo com a literatura especializada, como (Almeida, 2021), os periféricos podem ser classificados em diferentes categorias:

Os periféricos internos são aqueles que executam tarefas de forma autônoma, sem sobrecarregar continuamente o processador. Eles são projetados para realizar funções específicas e repetitivas que seriam ineficientes se processadas diretamente pela CPU a cada ciclo. Alguns exemplos comuns incluem:

- **ADC (Conversor Analógico-Digital):** Converte sinais analógicos (como tensão de sensores de temperatura ou corrente de painéis fotovoltaicos) em valores digitais que o processador pode entender e manipular.
- **PWM (*Pulse Width Modulation* - Modulação por Largura de Pulso):** Gera sinais digitais com ciclos de trabalho variáveis para controlar dispositivos analógicos, como a intensidade de luz de LEDs ou a velocidade de motores.
- ***Timers* (Temporizadores):** Permitem medir intervalos de tempo, gerar atrasos ou acionar eventos em períodos definidos, sendo cruciais para o agendamento de tarefas e o controle preciso.
- **Interrupções:** Mecanismos que permitem ao microcontrolador pausar sua execução normal para responder a eventos externos ou internos urgentes, garantindo uma resposta rápida e eficiente.
- ***Watchdog* (Cão de Guarda):** Um timer de segurança que reinicia o microcontrolador caso o software trave ou entre em um loop inesperado, garantindo a robustez do sistema.

Os periféricos de interface são responsáveis por condicionar os sinais que entram e saem dos terminais físicos do microcontrolador, também conhecidos como GPIOs (*General Purpose Input/Output*). As GPIOs são pinos versáteis que podem ser configurados como entradas (para ler estados digitais de botões, por exemplo) ou saídas (para ligar um LED).

Eles servem como a ponte primária entre o microcontrolador e componentes digitais simples do ambiente externo.

Periféricos externos são componentes adicionais que expandem as funcionalidades do microcontrolador além do que é nativamente oferecido pelo chip. Eles fornecem capacidades complementares que o microcontrolador, por si só, não possui ou possui em menor escala. Exemplos incluem displays LCD (para interfaces gráficas), teclados (para entrada de usuário), memórias externas (Flash ou RAM adicionais para armazenamento de grandes volumes de dados ou programas mais complexos) e outros módulos específicos. Esses periféricos se conectam ao microcontrolador através de barramentos e protocolos de comunicação, estendendo suas possibilidades de aplicação.

Os periféricos de comunicação são fundamentais para que o microcontrolador possa "conversar"com outros dispositivos e sistemas. Eles implementam protocolos e barramentos seriais que permitem a troca de dados. Alguns dos mais comuns são:

- I2C (*Inter-Integrated Circuit*): Um protocolo serial síncrono que permite a comunicação bidirecional entre múltiplos dispositivos com apenas duas linhas de sinal.
- SPI (*Serial Peripheral Interface*): Um protocolo serial síncrono de alta velocidade, ideal para comunicação com memórias Flash, displays e sensores que exigem rápida troca de dados.
- UART (*Universal Asynchronous Receiver/Transmitter*): Um protocolo de comunicação serial assíncrono amplamente utilizado para depuração e comunicação com módulos como GPS, Bluetooth ou outros microcontroladores.
- Ethernet, Wi-Fi e Bluetooth: Para conectividade em rede e Internet das Coisas (IoT), permitindo a comunicação com servidores, nuvens e outros dispositivos em redes locais ou globais.

2.5.2 Conceitos de Software Embarcado

Software embarcado é, essencialmente, um programa de computador dedicado a controlar um dispositivo ou máquina específica, que não é um computador de propósito geral como um PC ou smartphone. Ele é projetado para fazer um hardware funcionar, executando uma ou um conjunto limitado de funções para as quais aquele dispositivo foi construído.

2.5.2.1 Ciclo de Vida do Desenvolvimento de Software Embarcado

O desenvolvimento de software embarcado, embora compartilhe princípios com o desenvolvimento de software de propósito geral, possui um ciclo de vida com fases e considerações específicas que refletem a natureza restrita e crítica desses sistemas. A integração íntima com o hardware, os requisitos de tempo real e as limitações de recursos exigem uma

abordagem mais rigorosa e, muitas vezes, iterativa. A imagem 2.5 é um exemplo de um fluxograma que pode ser usado como desenvolvimento de softwares embarcados.

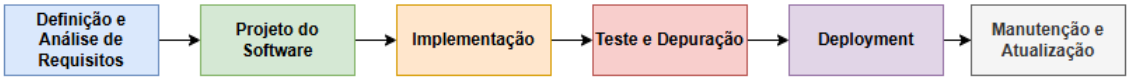


Figura 2.5 – Exemplo de um fluxo de um software embarcado (Fonte: Autoria Própria)

A figura 2.5 pode ser descrita pelos seguintes tópicos:

1. **Definição e Análise de Requisitos:** Estabelecer o que o sistema deve fazer (funcionalidades) e suas restrições (desempenho, memória, energia, ambiente).
2. **Projeto do Software:** Definir a arquitetura e estrutura do software (módulos, fluxo de dados, APIs), incluindo a escolha da abordagem (bare-metal ou RTOS).
3. **Implementação:** Escrever o código-fonte (em C/C++, MicroPython, etc.), desenvolvendo drivers e integrando bibliotecas.
4. **Teste e Depuração:** Verificar rigorosamente a funcionalidade e o desempenho do software (testes de unidade, integração, sistema) e corrigir erros, muitas vezes em hardware real.
5. **Deployment:** Gravar o firmware compilado na memória do dispositivo final.
6. **Manutenção e Atualização:** Fornecer suporte pós-lançamento, corrigir bugs e adicionar novas funcionalidades (incluindo possíveis atualizações remotas.).

2.5.2.2 Modelos de Implementação de Software Embarcado: Bare-Metal vs. RTOS

Em um desenvolvimento de software embarcado, escolher sua arquitetura no ínicio é uma decisão crucial, pois ela pode gerar um impacto significativo no desenvolvimento geral e na eficiência do produto final. Isso se torna verdade quando seu sistema embarcado requisita eficiência e uma taxa de resposta rápida.

Para sistemas embarcados, as duas principais abordagens são a programação bare-metal e o uso de Sistemas Operacionais de Tempo Real (RTOS). Embora ambas possam oferecer o desempenho otimizado necessário, cada uma possui suas próprias forças e limitações. Um RTOS, por exemplo, adiciona uma camada de sistema operacional que, apesar de consumir mais recursos, oferece maior estrutura e serviços integrados. Já um ambiente bare-metal não possui essa função do sistema operacional, proporcionando o benefício do controle direto sobre o hardware.

Em um ambiente bare-metal, a interação com o hardware é direta, uma vez que essa arquitetura elimina as camadas intermediárias de um sistema operacional, resultando em menor consumo de memória e energia. Essa abordagem permite a utilização precisa dos

recursos do sistema, conferindo maior exatidão em funcionalidades específicas. Consequentemente, desenvolvedores podem otimizar o código, aprimorando o desempenho.

Em contraste, no ambiente RTOS (Real-Time Operating System), o firmware deve responder dentro de prazos predefinidos, seja a eventos externos ou a interrupções de temporizadores. Ao projetar um sistema RTOS, é fundamental implementar um agendamento de tarefas. Uma das maneiras de fazer isso é utilizando um escalonador em conjunto com um laço de repetição infinito e flags. As flags são mecanismos lógicos, funcionando como um interruptor (ligado/desligado) e podem assumir os valores "1"ou "0". Os semáforos, por sua vez, possuem funcionalidade similar às flags, mas são empregados para controlar o acesso a recursos compartilhados em um ambiente multitarefa, conforme o livro (CERQUEIRA Marcos V B.; MASCHIETTO, 2021).

3 Metodologia

Este capítulo detalha a metodologia empregada para alcançar os objetivos deste trabalho. Inicialmente, procedeu-se ao levantamento e análise dos requisitos e à compreensão aprofundada do hardware a ser utilizado. Subsequentemente, a metodologia avançou para a fase de software embarcado, onde foram definidos os parâmetros a serem monitorados e os dados a serem adquiridos, com base nos sensores empregados. Abaixo estão detalhadas as seções deste capítulo:

- **Seção 3.1:** levantamento de requisitos funcionais e não funcionais a respeito, com base nos objetivos.
- **Seção 3.2:** entendimento do hardware utilizado e seu funcionamento.
- **Seção 3.3:** implementação do software embarcado com base nos parâmetros e dados necessários para o monitoramento.

Para um melhor entendimento, nota-se necessário um fluxograma para detalhar as etapas que foram feitas para a implementação da metodologia utilizada. A Figura 3.1 representa o fluxograma utilizado.

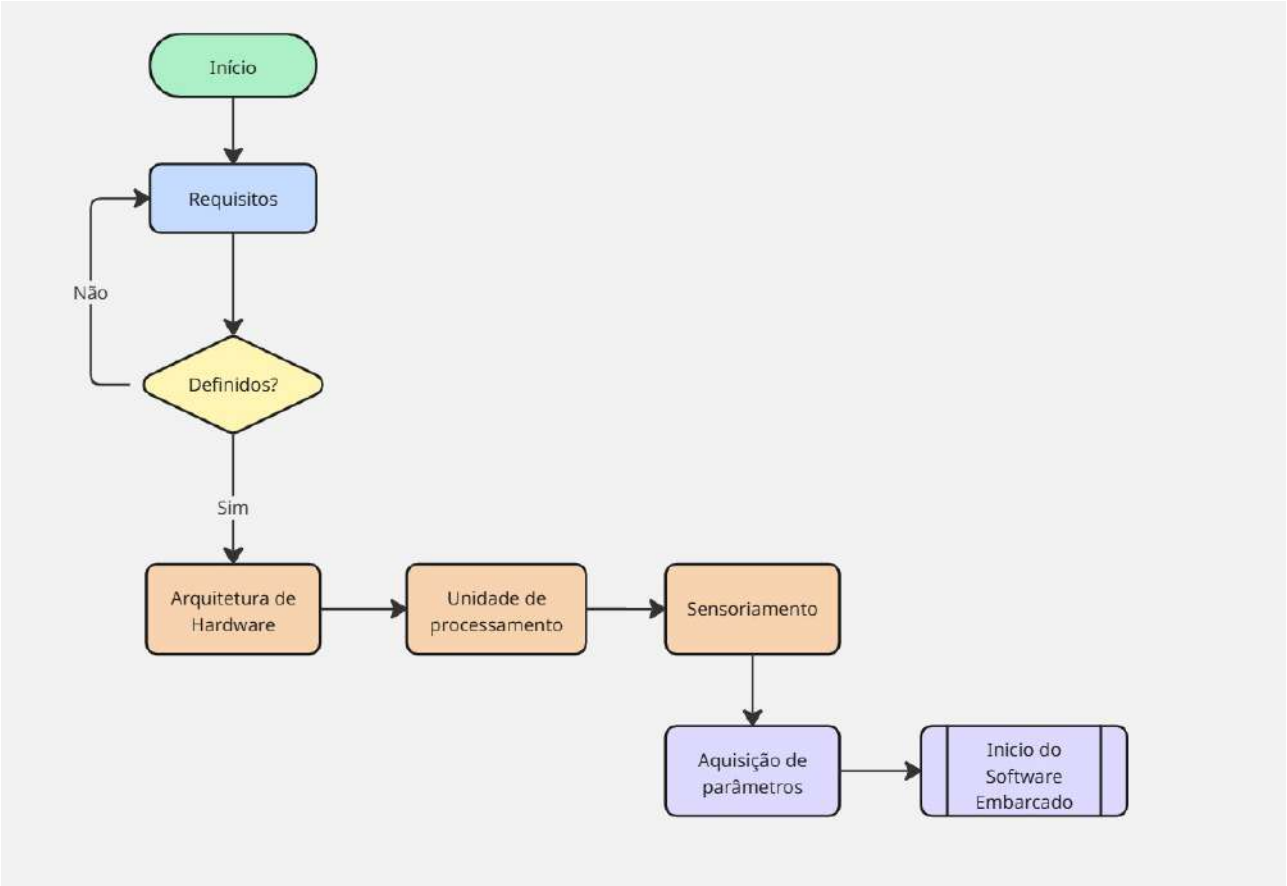


Figura 3.1 – Exemplo de um fluxo da metodologia utilizada (Fonte: Autoria Própria)

3.1 Análise de Requisitos

A análise de requisitos é uma etapa crucial no desenvolvimento de projetos, em que se identificam, compreendem e documentam as necessidades dele. Esta fase inicial é fundamental para garantir o sucesso do projeto, estabelecendo uma base sólida para a execução clara e objetiva do que é necessário para desenvolver, testar e validar o produto final. Durante esse processo, é essencial classificar os requisitos de acordo com sua prioridade, determinando se são **desejáveis, essenciais** ou **importantes**.

- **Essenciais:** são requisitos críticos e indispensáveis para o funcionamento básico do projeto.
- **Importantes:** representam funcionalidades ou características que agregam valor significativo e são altamente desejáveis para o sucesso do projeto, mas sua ausência não impede o funcionamento básico.
- **Desejáveis:** são requisitos que seriam bons de ter, mas não são cruciais para a primeira versão ou para o sucesso imediato do projeto.

É importante salientar que cada requisito será identificado por um código único, seguindo a numeração [RF1] para requisitos funcionais e [NF1] para requisitos não-funcionais.

3.1.1 Requisitos Funcionais

Requisitos funcionais definem as ações e funcionalidades que o projeto deve executar para atender e resolver os problemas e necessidades identificados. Essencialmente, tudo o que se refere a uma capacidade ou comportamento do sistema é considerado uma função. É crucial que esses requisitos sejam o menos ambíguos e o mais objetivos possível para garantir a alta qualidade do *software*.

RF1 - Medição de Tensão CC: O sistema deverá monitorar a tensão de saída de um módulo fotovoltaico a cada período de 1 ms.

- Faixa de operação: 0 a 80V.
- Prioridade: Essencial

RF2 - Medição de Corrente CC: O sistema deverá monitorar a corrente de saída de um módulo fotovoltaico a cada período de 1ms.

- Faixa de operação: 0 a 20A.
- Prioridade: Essencial

RF3 - Medição de Temperatura: O sistema deverá monitorar a temperatura real do módulo a cada 3s.

- Faixa de operação: 0 a 100 ºC.
- Prioridade: Essencial

RF4 - O sistema deve ser capaz de identificar anormalidades elétricas no painel fotovoltaico, como sobretensão, sobrecorrente e sobretemperatura.

- Sobretensão: acima de 60V
- Sobrecorrente: acima de 15A
- Sobretemperatura: acima de 70 ºC
- Prioridade: Essencial

RF5 - O sistema deve retirar o painel do array quando identificada uma das anormalidades do RF4. Um MOSFET em série deve seccionar em menos de 100ms. Somente após a verificação presencial poderá ser reconectado ao *array*.

.

- O sistema deve reportar o estado do MOSFET via RF6.
- Prioridade: Essencial

RF6 - O sistema deve fazer o envio dos dados de telemetria a cada evento detectado ou a cada 3s via MQTT.

- Eventos incluem: detecção de anormalidade (RF4)e abertura do MOSFET (RF5).
- A cada 3s: valores de tensão, corrente, temperatura e estado do MOSFET.
- Prioridade: Essencial

3.1.2 Requisitos Não Funcionais

Os requisitos não funcionais definem as qualidades e restrições do *software*/*firmware* ou aplicativo web, focando em como o sistema operará. Enquanto os requisitos funcionais descrevem o que o sistema fará, os não funcionais detalham aspectos como desempenho, segurança, usabilidade, confiabilidade e escalabilidade, essenciais para a experiência do usuário e a sustentabilidade do projeto.

NF1 - Processamento e transmissão de dados: O dispositivo deve processar os dados medidos e transmiti-los via MQTT.

- Taxa de aquisição: 1kHz para tensão e corrente (RF1, RF2); 1/3s para temperatura (RF3).
- Taxa de transmissão: dados periódicos a cada 3s; eventos críticos imediatamente.
- Protocolo: MQTT sobre TCP/IP via Wi-Fi (ESP32-C6).
- Prioridade: Essencial

NF2 - Sistema de alimentação: O sistema deve possuir circuito de alimentação integrado com armazenamento de energia para operação autossuficiente.

- Regulador buck de alta tensão: entrada 80VCC, saída 3,3V para ESP32-C6.
- Prioridade: Essencial

3.2 Arquitetura de Hardware

A arquitetura de *hardware* serve como um mapa detalhado dos componentes eletrônicos de um sistema, fundamental para o desenvolvimento do *software* embarcado. Ela não apenas lista os circuitos e elementos, mas também estabelece como o firmware irá interagir com cada um deles, seja para monitorar informações ou para acionar determinadas funções.

3.2.1 Unidade de Processamento

A unidade de processamento é responsável por fazer toda a aquisição de dados e processamento, no qual a escolha deve-se às funcionalidades que o sistema precisa. Uma forma de escolher a unidade de processamento é entender o que o mercado tem a oferecer e o que está sendo produzido ao nível do estado da arte. Nos últimos anos, uma empresa chamada "Espressif" tem chamado a atenção dos desenvolvedores de sistemas embarcados.

Empresa essa que desenvolve SoCs (System on Chips), os chamados "ESP32", que são famosos por serem versáteis e muito usados no mundo IoT. Um SoC significa que, além de um microcontrolador embarcado, possui uma série de periféricos disponíveis ao consumidor, como Wi-Fi, Bluetooth, protocolos de comunicação, entre outros. Dentro do catálogo da Espressif, uma das ESP32 chama a atenção: a ESP32-C6, um dispositivo que possui uma série de periféricos. Seguem as especificações da ESP32-C6.

- **Processador:** Um processador de arquitetura RISC-V, com velocidade de relógio (*clock*) de até 160 MHz, com memória cache de 32KB, ROM de 320 KB, SRAM de 512 KB.
- **Protocolos de comunicação sem fio:** Suporte ao Wi-Fi 6, Bluetooth 5, Thread e Zigbee. Todos podem ser usados em aplicações IoT.
- **Periféricos:** 30 pinos de GPIOs, divididos em um ADC do tipo SAR (*successive approximation*) de 7 canais, duas interfaces UART (*Universal asynchronous receiver/transmitter*), suporte a barramento I2C (*Inter-Integrated Circuit*) e também ao SPI(*Serial Peripheral Interface*), e como PWM(*Pulse Width Modulation*) e *timers*.

A Figura 3.2 mostra uma placa DevKit produzida pela própria Espressif. Essas placas são ferramentas cruciais no desenvolvimento de protótipos, permitindo que o desenvolvedor teste e refine aplicações antes de integrá-las a produtos específicos.

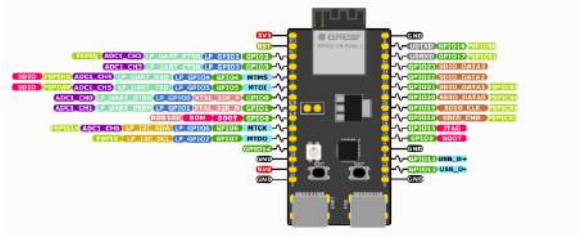


Figura 3.2 – Devkit da ESP32C6 (Fonte: Espressif)

3.2.2 Sensoriamento

Nesta seção, estão detalhadas como as principais medições do projeto serão realizadas. Serão abordados o funcionamento dos sensores e dos circuitos de condicionamento de sinal associados. Esse entendimento é fundamental para que o *software* embarcado seja projetado com as especificações exatas, garantindo o monitoramento preciso das grandezas elétricas e físicas necessárias.

3.2.2.1 Medição de Tensão

Para que a tensão de saída do módulo fotovoltaico possa ser medida, utilizou-se um transdutor de tensão, o LV 25-P. Sensor esse que possui o princípio de funcionamento baseado no efeito Hall. A Figura 3.3 mostra o circuito interno do LV25-P e suas conexões de entrada e saída

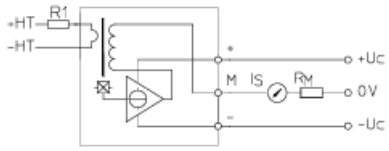


Figura 3.3 – Circuito interno LV25-P (Fonte: (LEM, 2022b))

Sua medição de tensão varia entre 10V e 500V, medições essas que estão dentro da faixa de operação dos módulos fotovoltaicos que serão utilizados. A tabela 3.1 mostra em detalhes as especificações eletrícas do LV25-P.

Tabela 3.1 – Características Técnicas do Lv25-P (Fonte: (LEM, 2022b))

I_{PN}	Corrente nominal do primário	10 mA		
I_P	Faixa de medição	0 ±14 mA		
V_P	Faixa de medição	10 ±500 V		
R_M	Resistência R_M	$R_{M_{min}}$	$R_{M_{max}}$	Ω
	with ±12V @ ±10mA	30	190	Ω
	with ±12V @ ±14mA	30	100	Ω
	with ±15V @ ±10mA	100	350	Ω
	with ±15V @ ±14mA	100	190	Ω
I_{SN}	Corrente nominal do secundário	25 mA		
K_N	Relação de conversão	2500 : 1000		
V_{CC}	Tensão de alimentação	±12	±15	V
V_D	Isolamento AC	2,5 KV		

Para que possa ser utilizado de forma correta, precisa de um circuito de condicionamento. O circuito de condicionamento foi baseado no trabalho de (Feijó, 2021), no qual foi utilizado um filtro anti-aliasing, para reduzir ruídos gerados por componentes de alta frequência. O subsistema que foi utilizado é apresentado pela Figura 3.4.

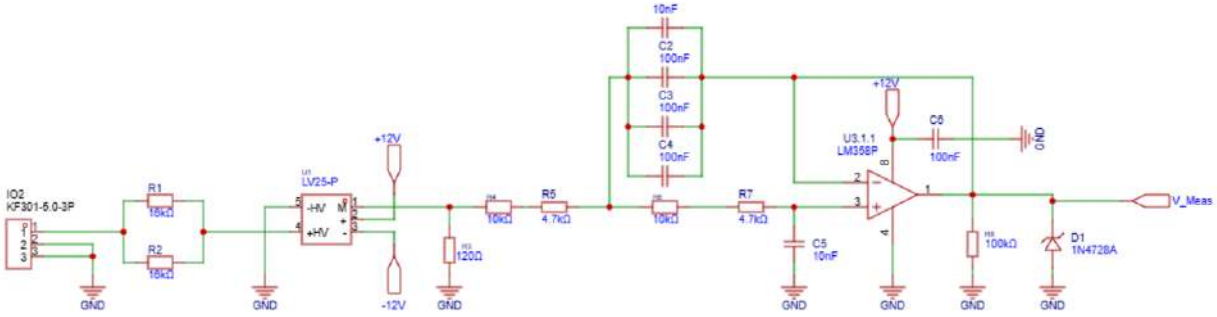


Figura 3.4 – Esquemático do LV25-P (Fonte: (Feijó, 2021))

O circuito é constituído por um limitador de corrente, que utiliza resistores de 16kΩ (3W), posicionado no primário do transdutor. No secundário, emprega-se um resistor shunt de 120Ω (1/4W), com tolerância de 1%. Este arranjo é, por sua vez, seguido por um filtro passa-baixas de segunda ordem.

3.2.2.2 Medição de Corrente

Para a medição de corrente utiliza-se um transdutor de corrente, o LA 55-P. Também baseado no efeito Hall, é capaz de medir correntes de 0 A até 70 A, (LEM, 2022a). A Figura 3.5 demonstra o diagrama interno do transdutor usado.

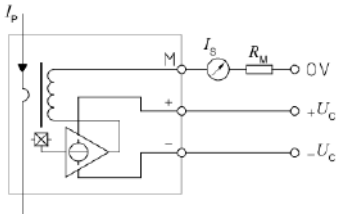


Figura 3.5 – Diagrama interno do LA55-P (Fonte: (LEM, 2022a))

I_P Denota a corrente que flui do primário do transdutor, enquanto +UC e - UC correspondem aos terminais de alimentação do mesmo. É importante notar que o cálculo das resistências foi previamente executado em ((Feijó, 2021)), servindo como fundamento para o desenvolvimento deste trabalho. A Tabela 3.2 apresenta as especificações do LA55-P.

O circuito (representado pela Figura 3.6) é composto por um resistor *shunt* (resistor de 150Ω por 1/4W, com tolerância de 1%) no secundário do transdutor, seguido por um filtro anti-aliasing de 2ª ordem.

3.2.2.3 Medição de Temperatura

Para monitorar e prevenir o superaquecimento do módulo fotovoltaico, empregou-se um termistor NTC (Coeficiente de Temperatura Negativo). Este componente possui a característica de ter sua resistência elétrica reduzida à medida que a temperatura aumenta.

Tabela 3.2 – Características Técnicas do LA55-P (Fonte: (LEM, 2022a))

I_{PN}	Corrente nominal do primário	50	A
I_{PM}	Faixa de medição	0 | ± 70	A
R_M	Resistência R_M ($T_A = 70°C$)	$R_{Mmin}\|R_{Mmax}$	Ω
	com ±12V a ±50A	10 | 100	Ω
	com ±12V a ±70A	10 | 50	Ω
	com ±15V a ±50A	50 | 160	Ω
	com ±15V a ±70A	50 | 90	Ω
I_{SN}	Corrente nominal do secundário	50	mA
K_N	Relação de conversão	1 : 1000	
U_C	Tensão de alimentação	±12 | ± 15	V
U_D	Isolamento AC, 50 Hz	2.5	kV

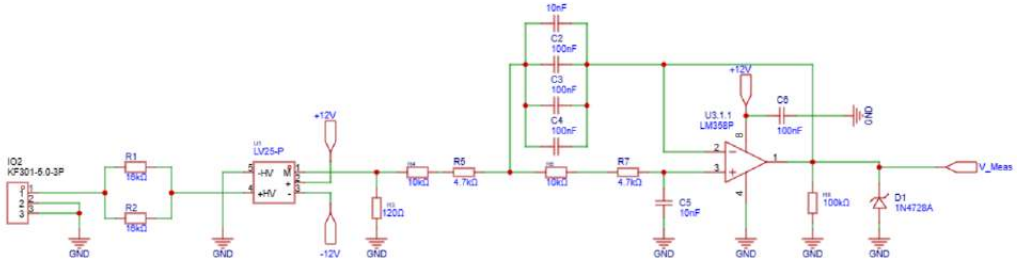


Figura 3.6 – Esquemático do LA55-P (Fonte: (Feijó, 2021))

Especificamente, foi utilizado o termistor NXRT15XV103FA1B040, fabricado pela Murata, cujas características elétricas detalhadas estão apresentadas na Tabela 3.3.

Tabela 3.3 – Características elétricas do Termistor NTC NXRT15XV103FA1B040(Fonte: (Murata, 2022))

Resistência (25°C)	10 kΩ
Tolerância da Resistência (25°C)	±1%
Corrente Máxima de Operação	0.12 mA
Potência	7.5 mW
Constante de Dissipação Típica (25°C)	1.5 mW/°C
Temperatura de Operação	−40°C a 125°C

O circuito utilizado para aquisição dos valores de temperatura utilizando o NXRT15XV103FA1B040 é representado pela Figura 3.7

3.2.3 Desconexão do módulo fotovoltaico

Para permitir o desligamento rápido do módulo fotovoltaico em caso de mau funcionamento ou eventos adversos, optou-se pela utilização de um MOSFET de canal N (NMOS), modelo STP75NF75-VB (SEMI, 2022). Este componente é crucial para realizar a interrupção controlada da corrente.

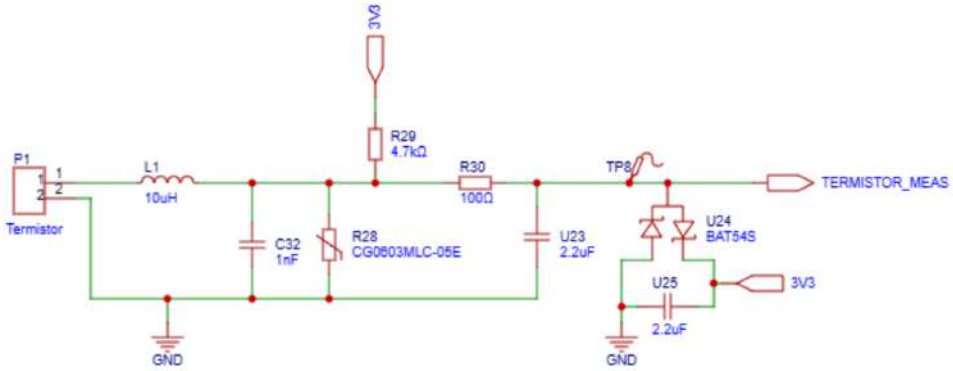


Figura 3.7 – Esquemático do NXRT15XV103FA1B040 (Fonte: Autoria própria)

Para o acionamento e desacionamento eficiente do MOSFET, empregou-se um *driver* de porta, o MIC4427 (MICREL, 2022), configurado em modo não inversor. Essa configuração proporciona um controle direto e intuitivo sobre o MOSFET. O MIC4427 opera com lógica TTL (*Transistor-Transistor Logic*), o que significa que uma entrada de nível lógico alto (1) resultará em uma saída de nível lógico alto, acionando o MOSFET; e uma entrada de nível lógico baixo (0) gerará uma saída de nível lógico baixo, desativando-o. A Figura 3.8 ilustra o circuito de acionamento e proteção responsável pelo desligamento rápido do módulo.

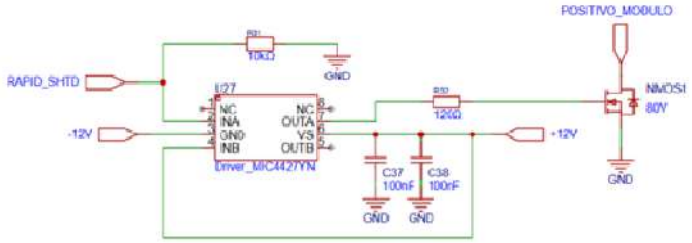


Figura 3.8 – Esquemático do Dispositivo contra surtos (Fonte: Autoria própria)

3.3 Software Embarcado

Para otimizar o desenvolvimento e a execução do *firmware* (*software* embarcado), é fundamental compreender o seu fluxograma de operação. Este fluxograma detalhará todas as etapas essenciais para o funcionamento do *firmware*, abrangendo desde a inicialização até as rotinas de aquisição, processamento de dados, comunicação e envio.

3.3.1 Fluxograma de firmware

O *firmware* foi estruturado em um modelo multitarefa, utilizando o FreeRTOS para garantir o determinismo da aquisição de sinais analógicos sem sofrer interferências da latência de rede. A arquitetura adota, dividindo o processamento em duas *Tasks* independentes. Para preservar os recursos de *hardware* do microcontrolador, o fluxo de comunicação entre

as tarefas possui caminhos distintos para dados de rotina e dados de emergência. A Figura 3.9 demonstra como foi implementado o *firmware* embarcado.

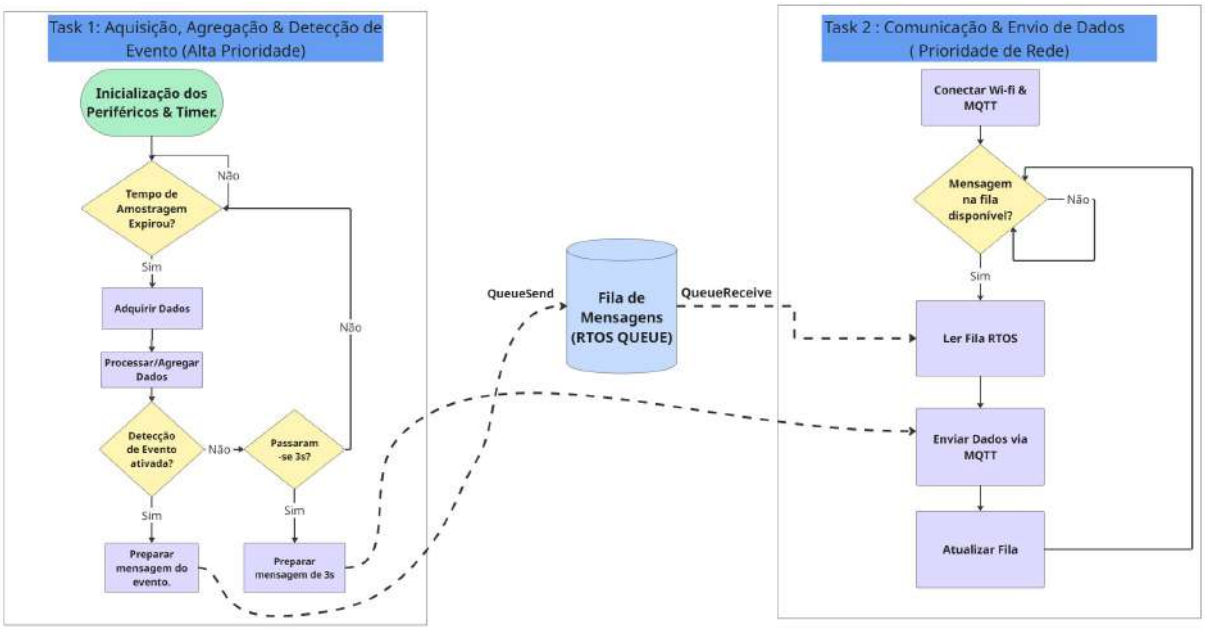


Figura 3.9 – Fluxograma de Firmware (Fonte: Autoria própria)

A primeira *task* "Aquisição, Agregação e Detecção de Eventos" é a rotina de mais alta prioridade do sistema, responsável pela interação direta com o *hardware* e pelo processamento computacional na borda (*Edge Computing*). O fluxo opera da seguinte maneira:

- **Controle Temporal e Aquisição:** O sistema aguarda rigorosamente a expiração do tempo de amostragem para iniciar a leitura dos dados do conversor A/D ("Adquirir Dados"), garantindo uma taxa de aquisição determinística.
- **Processamento e Agregação:** Imediatamente após a leitura, os dados alimentam a cascata de vetores ("Processar/Agregar Dados"). A primeira camada processa os dados instantâneos, condensando-os a cada 0,2 segundos.
- **Detecção e Despacho de Dados:** O algoritmo avalia as amostras em busca de distúrbios, atuando sob duas lógicas de roteamento distintas:
  1. **Gatilho Assíncrono (Detecção de Evento):** Se uma anomalia elétrica for identificada, o sistema ignora o tempo de agregação regular. Uma mensagem de alerta emergencial é preparada e despachada imediatamente para a fila do RTOS (`QueueSend`).
  2. **Gatilho Síncrono (Lotação do Buffer de 3s):** Na ausência de eventos críticos, a Segunda Camada atinge sua capacidade máxima a cada 3 segundos. Neste momento, a média global é validada. Crucialmente, para evitar o transbordamento da memória RAM (*Heap Overflow*) do ESP32, esta telemetria de rotina não é inserida na fila do RTOS.

A “Fila de Mensagens (RTOS QUEUE)” atua como um buffer circular estritamente reservado para ocorrências críticas. Sua principal função é garantir que alertas de anomalias nunca sejam perdidos, mesmo que haja instabilidade ou lentidão na conexão Wi-Fi. Ao isolar a telemetria periódica (3s) do tráfego desta fila, o sistema garante imunidade contra o esgotamento de memória RAM, mantendo a integridade do firmware em cenários de degradação de rede.

A segunda *task* “Comunicação e Envio de Dados (O Consumidor é Transmissor)", opera com prioridade secundária e gerencia exclusivamente a pilha de protocolos de rede (Wi-Fi e MQTT), trabalhando sob duas frentes de transmissão:

- **Gestão de Eventos:** A *Task* monitora continuamente a Fila de Mensagens. Havendo qualquer pacote de evento disponível, o sistema extrai o dado de forma atômica (`QueueReceive`), prioriza sua formatação e força a publicação imediata no *broker* MQTT.
- **Transmissão de Rotina (3s):** Em paralelo ao monitoramento da fila, a tarefa de rede verifica periodicamente o status das variáveis de telemetria consolidada da *Task* 1. Quando o ciclo de 3 segundos é concluído, os dados são lidos da memória de estado, empacotados e transmitidos, mantendo a atualização contínua da nuvem sem comprometer a estabilidade do fluxo de prioridades do microcontrolador.

3.3.2 Aquisição e Processamento

Para alcançar uma resposta ágil na aquisição de dados e garantir a acurácia das medições, optou-se pela implementação de um buffer circular com estrutura multicamadas. As medições serão realizadas a uma taxa de 960 sps (amostras por segundo), correspondente à frequência da interrupção do timer 1, conforme ilustrado na Figura 3.9.

Este método de aquisição foi inspirado na norma IEC 61400-4-30, que estabelece métodos de medição para sistemas elétricos. Embora esta norma seja predominantemente voltada para medições em corrente alternada (CA), o método de aquisição foi adotado devido à sua capacidade de suportar a tomada de decisões baseadas tanto em variações de curto prazo (como surtos e curtos-circuitos) quanto em análises de médio e longo prazo (como monitoramento de potência gerada e degradação dos módulos). A Figura 3.10 representa como funciona, em esquema de blocos, a agregação por multicamadas.

Tal que os níveis foram divididos em:

1. 1ª Camada:
   Buffer circular de 192 posições (12 ciclos de 16 amostras cada), inseridas a partir da leitura do módulo ADC, a uma taxa de 960 sps, totalizando 0,2 s.
2. 2ª camada:
   Buffer circular de 15 posições, calculado a partir da média aritmética da primeira

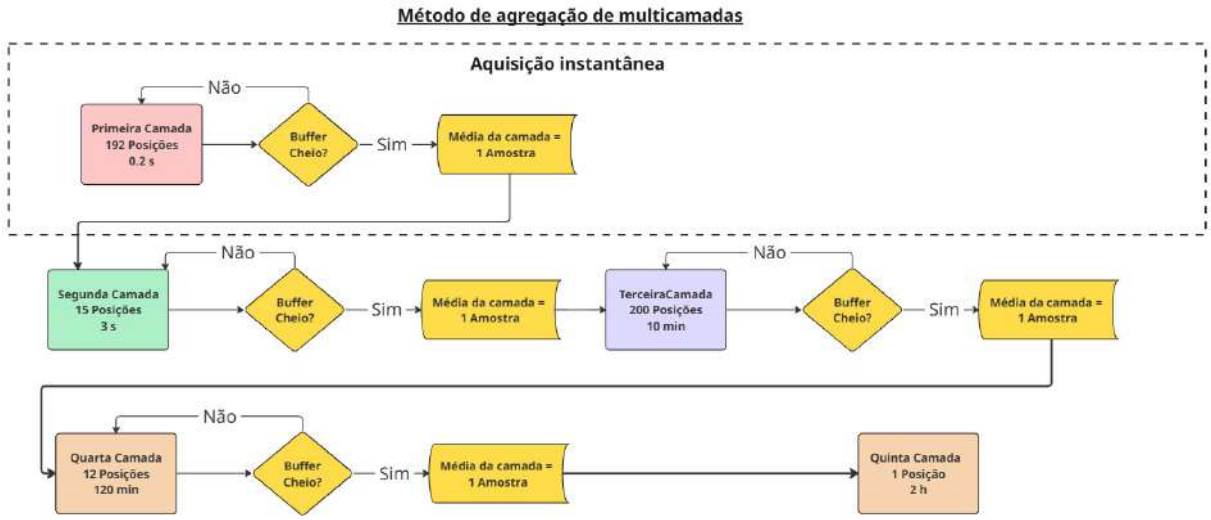


Figura 3.10 – Método de agregação em multicamadas, baseado na norma IEC 61400-4-30 (Fonte: autoria própria).

camada, a cada 0,2s, totalizando 3s.

3. 3ª camada:
   Buffer circular de 200 posições, calculado a partir da média aritmética da segunda camada, a cada 3 segundos, totalizando 10 min.
4. 4ª camada:
   Buffer circular de 12 posições, calculado a partir da média aritmética da terceira camada, a cada 10 minutos, totalizando 2h.
5. 5ª camada:
   Buffer circular de 1 posição, calculado a partir da média aritmética da quarta camada, totalizando 2h.

3.3.3 Detecção de Eventos

A confiabilidade de um sistema de monitoramento fotovoltaico está diretamente ligada à sua capacidade de identificar falhas elétricas reais, mitigando o risco de danos aos equipamentos e garantindo a segurança operacional. Sistemas de geração solar operam em ambientes externos dinâmicos e estão suscetíveis a distúrbios que podem levar os módulos a operarem fora de suas margens nominais.

Neste projeto, o *software* embarcado foi desenvolvido especificamente para focar no monitoramento e na detecção de dois dos eventos mais críticos em arranjos fotovoltaicos: a sobretensão e a sobrecorrente.

3.3.3.1 Parâmetros de Medição e Sensores de Tensão e Corrente

A base para a detecção de anomalias consiste na aquisição confiável das grandezas elétricas geradas pelo painel. Para isso, o sistema utiliza dois transdutores baseados no princípio do Efeito Hall de malha fechada, que garantem isolamento galvânico e alta precisão (Kamal; Dona, 2024).

- **Detecção de sobrecorrente**: realizada por meio do processamento dos dados do sensor de corrente LA-55P. O aumento anormal da corrente pode indicar falhas como curtos-circuitos no arranjo, problemas nos diodos de *bypass* ou degradação severa dos cabos (Pei; Hao, 2019).
- **Detecção de sobretensão:** realizada por meio do processamento dos dados do sensor de tensão LV25P. Tensões acima do limite seguro podem ser causadas por falhas na rede de conexão, defeitos na atuação do inversor ou variações abruptas de temperatura sob alta irradiância.

A conversão analógico-digital (ADC) dos sinais gerados por estes sensores é executada pelo microcontrolador ESP32-C6 a uma taxa de amostragem de aproximadamente 1 ms (frequência de 960Hz).

3.3.3.2 Proteção do Módulo: histerese com Janelamento Temporal

Para evitar que flutuações de sinal e ruídos de leitura gerem falsos alertas de falha ou o sistema de proteção atue de forma precipitada, a detecção de magnitude no *firmware* não utiliza um limite de ativação simples, mas sim um comparador digital por histerese (Kornkanok; Deeon; Wongcharoen, 2024). A histerese estabelece duas bandas operacionais para cada grandeza: um limiar superior crítico para o acionamento do evento (V_{max_H} ou I_{max_H}) e um limiar inferior de segurança para o retorno à normalidade (V_{max_L} ou I_{max_L}).

Essa abordagem cria uma "janela de inércia" no *software* (semelhante aos métodos de tolerância estatística propostos por (Nguyen-Van; Abe; Tanaka, 2018)), que impede o efeito de "repique" — situação em que o sistema de proteção alternaria rapidamente entre os estados de falha e operação normal caso o valor da tensão ou corrente flutuasse exatamente sobre um único ponto de detecção.

No entanto, mesmo com a lógica de histerese, o arranjo fotovoltaico pode sofrer distúrbios transitórios inofensivos. Picos de comutação do inversor, descargas atmosféricas distantes ou manobras bruscas na rede elétrica podem gerar micro picos que duram frações de segundo. Acionar o seccionamento do módulo por conta desses eventos geraria paradas operacionais redundantes sem representar um defeito real no equipamento (Silvestre *et al.*, 2014). Para filtrar esses falsos positivos e garantir que o módulo seja isolado apenas quando houver risco real, implementou-se uma dupla etapa de validação temporal.

A fim de elucidar como o microcontrolador interpreta os sinais e atua na proteção

do módulo fotovoltaico, a Figura 3.11 apresenta a detecção prática de uma sobrecorrente operando sob o método de histerese com janelamento de validação e atraso de acionamento.

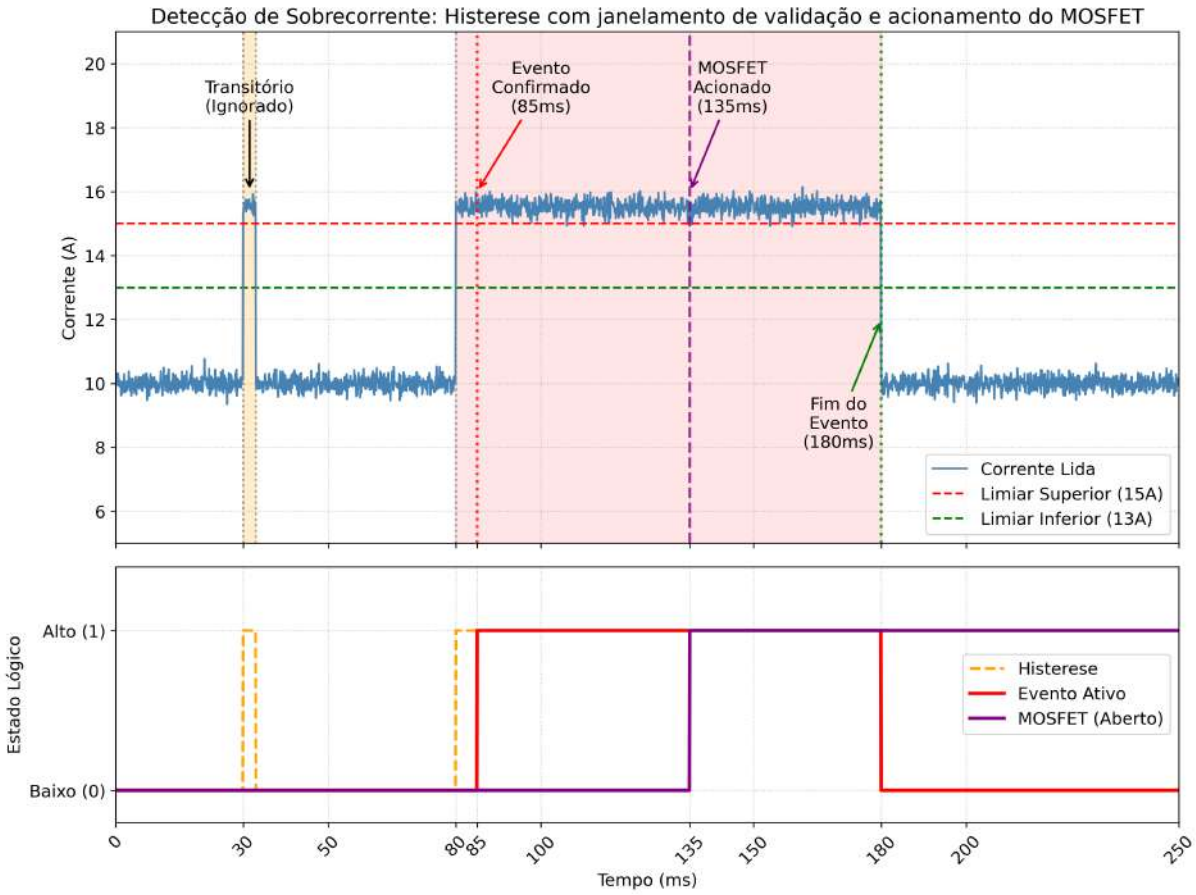


Figura 3.11 – Dinâmica de proteção: histerese com janelamento de validação e acionamento do MOSFET. Fonte: Autoria Própria

Conforme ilustrado na Figura 3.11, a lógica de proteção opera nas seguintes etapas:

1. **Rejeição de Transitórios (Janela de Validação):** Ao redor de 30 ms, ocorre um pico de corrente que ultrapassa o limiar superior de 15 A. O *firmware* inicia um contador interno configurado empiricamente para 5 ms (aproximadamente 5 amostras a 960 Hz). Como o distúrbio cessa antes desse tempo, o evento é classificado como transitório e ignorado, mantendo o módulo conectado e gerando energia.
2. **Confirmação da Anomalia:** Aos 80 ms, uma nova violação do limite de 15 A ocorre. Desta vez, a condição se mantém verdadeira durante toda a janela de 5 ms. Aos 85 ms, a anomalia é confirmada pelo *software*, alterando o estado do sistema para "Evento Ativo". Neste momento, a telemetria crítica começa a ser formatada para envio prioritário via protocolo MQTT à plataforma de monitoramento.
3. **Atuação da Proteção (Atraso do MOSFET):** Após a confirmação do evento, para permitir a coordenação das proteções, um segundo temporizador de 50 ms é acionado. Aos 135 ms, com o evento ainda ativo, o microcontrolador eleva o nível lógico dos

drivers, acionando fisicamente o MOSFET (estado “Alto”). Esta ação secciona o circuito, isolando o módulo fotovoltaico do restante do sistema e prevenindo danos por sobrecarga.

4. **Travamento de Segurança (*Latch*):** Aos 180 ms, a corrente decai cruzando o limiar inferior de histerese (13 A), o que sinaliza o fim do evento elétrico. Contudo, o sistema de segurança mantém o MOSFET permanentemente acionado (aberto). O módulo fotovoltaico permanece isolado até que a equipe técnica realize a inspeção física e envie o comando remoto de rearme via *dashboard*, assegurando total integridade à instalação.

3.3.4 Comunicação e Envio de Dados

A etapa de comunicação é fundamental para sistemas de Internet das Coisas (IoT), pois permite que os dados processados localmente pelo microcontrolador sejam transmitidos para uma plataforma em nuvem ou servidor local. No contexto deste projeto, a transmissão dos dados de telemetria (tensão, corrente e temperatura) e dos alertas de eventos críticos é realizada utilizando a conectividade Wi-Fi e o protocolo MQTT. Isso garante que o monitoramento do módulo fotovoltaico e a detecção de anomalias ocorram em tempo real e de forma confiável.

3.3.4.1 Wi-Fi

O principal avanço tecnológico que justifica a adoção do ESP32-C6 neste sistema é a sua conformidade nativa com a norma IEEE 802.11ax, amplamente conhecida como Wi-Fi 6. Operando com uma largura de banda de 20 MHz neste modo, o chip suporta esquemas de modulação e codificação que vão do MCS0 ao MCS9. A tecnologia Wi-Fi 6 introduz funcionalidades vitais para infraestruturas que lidam com alta densidade de nós sensores — um cenário comum em usinas solares com monitoramento individual por painel. reúso espacial para maximizar transmissões paralelas.

Apesar do suporte avançado ao 802.11ax, o sistema mantém retrocompatibilidade total com as redes tradicionais baseadas nos protocolos IEEE 802.11b/g/n, alcançando taxas de transmissão de dados de até 150 Mbps e suportando canais de 20 MHz e 40 MHz de largura.

3.3.4.2 MQTT

A transferência de dados entre o microcontrolador (ESP32-C6) e a nuvem foi implementada utilizando o protocolo MQTT (Message Queuing Telemetry Transport). A escolha deste protocolo para a metodologia justifica-se pelo seu baixo consumo de banda e por operar sobre o modelo de Publicação e Assinatura (*Publish/Subscribe*), ideal para dispositivos de IoT com recursos computacionais restritos.

Nesta topologia, o microcontrolador não se comunica diretamente com os painéis de visualização. Em vez disso, ele atua exclusivamente como um cliente MQTT, estabelecendo uma conexão TCP/IP contínua com um servidor central denominado *Broker*. No contexto deste projeto, a plataforma ThingsBoard atua simultaneamente como o *Broker* MQTT (recepcionando os pacotes no endereço 10.227.105.146 na porta padrão 1883) e como o motor de processamento de regras e interface gráfica.

No ecossistema MQTT, a informação é roteada por meio de "tópicos"(topics), que funcionam como endereços hierárquicos. Para garantir a compatibilidade nativa com a API do ThingsBoard, a comunicação foi estruturada em três vias principais de tópicos de sistema:

- **Tópico de Telemetria (`v1/devices/me/telemetry`):** Utilizado de forma unidirecional (Cliente → Servidor) para o envio de séries temporais. É por meio deste canal que o sistema publica as variáveis de rotina (tensão, corrente e temperatura da placa) e os dados durante um evento.
- **Tópico de Atributos (`v1/devices/me/attributes`):** Utilizado para propriedades de estado da máquina. O *firmware* é configurado para assinar (*subscribe*) este tópico logo após a conexão. Isso permite que o dispositivo receba atualizações instantâneas caso um parâmetro de configuração seja alterado remotamente pelo operador no *dashboard*.
- **Tópico de RPC (`v1/devices/me/rpc/request/+`):** Tópico bidirecional dedicado à Chamada de Procedimento Remoto (*Remote Procedure Call*). A assinatura deste canal garante a atuação do seccionador. Quando um comando (ex: “ON” ou “OFF”) é despachado pela interface, o servidor o encaminha por este tópico, e o ESP32C6 responde validando a execução no tópico correspondente de resposta (`v1/devices/me/rpc/response/`), fechando a malha de controle.

3.3.5 Atuação sobre o módulo

Conforme ilustrado na Figura 3.8, que apresenta o esquemático dos *drivers* de acionamento dos transistores MOSFET responsáveis pelo seccionamento do módulo, a atuação da proteção exige o envio de um sinal em nível lógico alto a partir dos pinos digitais (GPIOs) do microcontrolador. Portanto, para isolar o circuito da carga, o *firmware* comuta as respectivas GPIOs para nível lógico alto, saturando os *drivers* e acionando as chaves de potência.

O acionamento desta proteção ocorre exclusivamente mediante a detecção de eventos críticos, como sobrecorrente ou sobretensão, condições que indicam potenciais falhas ou anomalias indesejadas na operação do módulo fotovoltaico. Uma vez que o circuito é seccionado, o religamento automático pelo código é intencionalmente bloqueado. O rearme do sistema só pode ser executado remotamente por meio de comandos via *dashboard* na plataforma IoT. Essa premissa lógica foi estabelecida para garantir que uma análise técnica prévia seja realizada no módulo físico afetado, permitindo o desacionamento remoto da

proteção apenas quando a segurança da instalação for atestada.

Visando à robustez do sistema, era imperativo mitigar o risco de o microcontrolador ESP32-C6 reverter as GPIOs de seccionamento para o nível lógico baixo (estado padrão de inicialização) durante uma eventual reinicialização inesperada do sistema, o que provocaria o religamento acidental do módulo sob uma condição pré-existente de falha. Para contornar este comportamento, implementou-se uma rotina de persistência de estados utilizando a biblioteca nativa NVS (*Non-Volatile Storage*) para gerenciar a memória *Flash* do microcontrolador. Por meio deste recurso, o estado lógico atual da proteção é gravado continuamente em uma partição não volátil. Dessa forma, ao ser energizado após uma falha de alimentação, a primeira instrução de rotina do sistema é consultar o NVS e reaplicar imediatamente o último estado gravado nas GPIOs, assegurando a continuidade do isolamento elétrico até a intervenção do operador.

4 Resultados

4.1 Considerações Iniciais

Para a implementação do sistema, optou-se pelo microcontrolador ESP32-C6 (3.2) como unidade de processamento central. O ambiente de desenvolvimento integrado (IDE) utilizado para a codificação e depuração do firmware foi o VS Code, em conjunto com o ESP-IDF (*Espressif IoT Development Framework*), que forneceu as ferramentas de compilação e as bibliotecas necessárias. A linguagem de programação empregada foi "C", e a estrutura do *firmware* foi desenvolvida com base no *FreeRTOS* (Sistema Operacional de Tempo Real), visando gerenciar as tarefas concorrentes de forma eficiente.

Uma placa de circuito impresso (PCB) foi produzida a fim de testar em ambientes reais o funcionamento do *firmware*, porém, devido a falhas na produção inicial, foi necessária uma nova rodada de produção dessa placa. Assim, os testes que foram executados nesse documento foram feitos em bancada, em um ambiente controlado. A Figura 4.1, mostra a placa produzida, juntamente com uma outra placa feita em placa perfurada, pois a alimentação simétrica dos sensores de tensão (Figura 3.4) e sensor de corrente (Figura 3.6) não funcionou na primeira versão da placa. Assim, foi feito esse protótipo (placa verde acima da principal) de fonte apenas para gerar a alimentação negativa (-12V) para os sensores.

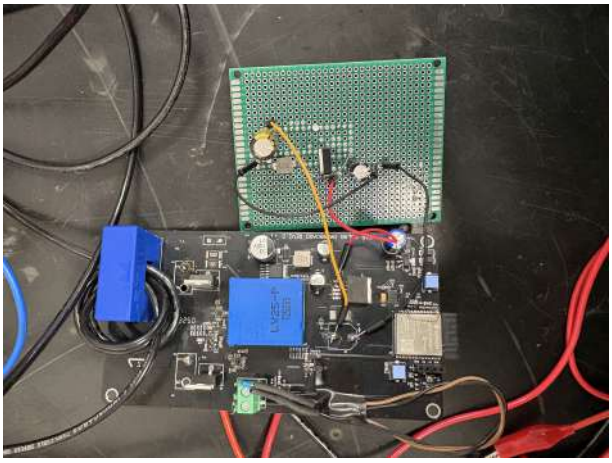


Figura 4.1 – Placa de circuito impresso produzida para o projeto. Fonte: autoria própria

4.2 Inicialização do firmware.

O desenvolvimento do *firmware* para o microcontrolador ESP32-C6 consolidou os requisitos de instrumentação, lógica de proteção e conectividade IoT em um sistema embar-

cado integrado e funcional. Para garantir que as exigências temporais da proteção elétrica não fossem comprometidas pela latência natural das comunicações de rede, a arquitetura de *software* foi estruturada sobre o sistema operacional de tempo real FreeRTOS. Essa abordagem permitiu abandonar a execução linear clássica em favor do paralelismo, segregando as responsabilidades críticas em tarefas (*Tasks*) independentes e hierarquizadas.

O ponto de entrada e orquestração do sistema ocorre na função principal do código-fonte (`app_main`), Apêndice A.1. Neste estágio de inicialização (comumente denominado *setup*), o microcontrolador executa rotinas preparatórias essenciais antes de iniciar a aquisição de dados. Primeiramente, inicializa-se o sistema de arquivos na memória não volátil (NVS), passo obrigatório tanto para o funcionamento da pilha de rede sem fio quanto para a segurança do *hardware* de proteção. É por meio da consulta inicial ao NVS que a rotina `VerifyRAPID_state()` assegura a persistência do estado lógico dos pinos atuadores, impedindo o religamento acidental do módulo fotovoltaico em caso de reinicializações por falha de alimentação.

Após a estabilização do *hardware* físico, criação das estruturas de sincronização (semáforos e filas) e estabelecimento da conexão Wi-Fi, sincronização do *timestamp* e do cliente MQTT, a função principal finaliza seu fluxo sequencial, delegando o processamento infinito a duas frentes de trabalho concorrentes. O escalonador do FreeRTOS assume o controle e instancia as duas rotinas fundamentais do projeto:

1. **Tarefa de Medição Principal (`Main_Measurement`):** configurada com maior prioridade no sistema, é estritamente dedicada a gerenciar o temporizador de *hardware*, comandar as leituras do conversor analógico-digital (ADC) e executar o algoritmo de histerese para detecção de anomalias em milissegundos, Apêndice A.2.
2. **Tarefa de Gestão de Rede (`conectarWifi`):** Operando em uma prioridade secundária, atua como uma consumidora de dados. Ela esvazia a fila de eventos críticos gerada pela medição, gerencia a máquina de estados para empacotamento de envio e controla todo o tráfego de telemetria e junto à plataforma ThingsBoard, Apêndice A.3.

A seguir, detalha-se o funcionamento computacional e o fluxo de dados no interior de cada uma dessas tarefas, bem como os impactos diretos dessa separação no desempenho e na confiabilidade do seccionamento elétrico. No Apêndice A.1, está o bloco da função "app_main", responsável por todo o gerenciamento do microcontrolador.

4.2.1 Aquisição e processamento

A tarefa `Main_Measurement`, responsável pela aquisição contínua dos parâmetros de tensão, corrente e temperatura, inicia sua execução instanciando um semáforo binário dedicado à sincronização com o temporizador de *hardware*. Este temporizador é previamente configurado para gerar interrupções assíncronas a uma frequência de amostragem de 960 Hz

(com período de aproximadamente 1 ms). Na sequência, a rotina realiza a configuração inicial do conversor analógico-digital (ADC), aplica os parâmetros de calibração aos canais de leitura e inicializa as variáveis de controle da lógica de proteção do módulo fotovoltaico (`RAPID_Init()` e `Initialize_Parameters()`), pode ser visualizado no Apêndice A.2.

Com a base de *hardware* estabelecida, o temporizador principal é acionado. A partir deste momento, a cada disparo da Rotina de Serviço de Interrupção (ISR), o microcontrolador realiza a leitura em modo *one-shot* dos canais do ADC correspondentes aos sensores de tensão e corrente, convertendo os sinais e armazenando os valores instantâneos na primeira camada do *buffer* de aquisição.

Após o estágio de configuração, o fluxo de execução entra no laço de repetição infinito (`while (1)`) característico de sistemas embarcados. Neste estado, a tarefa entra em bloqueio passivo, aguardando que o semáforo seja liberado pela interrupção do temporizador. Uma vez que o controle retorna ao laço principal, o algoritmo verifica sistematicamente o preenchimento das matrizes de agregação de dados.

Sempre que a rotina multicamadas conclui a média temporal equivalente a uma janela de 3 segundos, uma variável de sinalização (`event_flag_3s`) é ativada. Ao detectar esta condição, a `Main_Measurement` disponibiliza o semáforo de transmissão (`Transmit`). Esta ação de sincronização interprocessos alerta imediatamente a tarefa secundária, `conectarWifi`, de que os valores consolidados em regime permanente estão prontos para serem extraídos e despachados como telemetria de rotina para a plataforma IoT.

4.2.2 Comunicação

Atuando como consumidora na arquitetura paralela do sistema, a tarefa `conectarWifi`, código integral encontra-se detalhada no Apêndice A.3, gerencia a comunicação com a nuvem sem interferir no determinismo da aquisição de dados. Sua execução contínua no laço principal baseia-se em dois mecanismos distintos de sincronização do FreeRTOS (Real Time Engineers Ltd., 2024) semáforos para o tráfego de rotina e filas de mensagens (*Queues*) para eventos críticos transientes.

O fluxo de telemetria em regime permanente é estritamente controlado pelo semáforo binário `Transmit`. Quando o semáforo é obtido com sucesso, o microcontrolador aciona imediatamente a rotina `mqtt_enviar_rotina()`, que extrai os valores médios consolidados de tensão, corrente e temperatura, formata a carga útil em JSON e a publica no *Broker* MQTT (OASIS, 2019). Este método de bloqueio condicional garante que o rádio Wi-Fi seja exigido apenas no exato momento em que um novo bloco de dados estiver íntegro, evitando processamento desnecessário.

Por fim, a máquina de estados interna desta tarefa monitora continuamente os parâmetros elétricos para identificar o retorno à faixa de segurança nominal. Ao atestar fisicamente

o fim do distúrbio na placa, um comando final (`mqtt_alerta_off`) é transmitido, registrando a duração absoluta da falha em milissegundos e assegurando a rastreabilidade integral da ocorrência no banco de dados da plataforma de monitoramento (ThingsBoard Authors, 2024).

Para a sincronização do *timestamp*, no início do código quando se conecta ao WiFi, o microcontrolador atualiza seu *timestamp* interno para que possa enviar os pacotes de dados com a data e hora do pacote, Apêndice A.4. O *timestamp* é disponibilizado do tipo epoch,é uma representação numérica do tempo usada por computadores. Ele indica a quantidade exata de segundos (ou milissegundos) decorridos desde 1º de janeiro de 1970 às 00:00:00 UTC, marco conhecido como *Unix Epoch*. Com isso, é necessário que de tempos em tempos esse *timestamp* da ESP32 seja sincronizado novamente com o da rede. Por essa razão, o próprio *framework* da ESP32 disponibiliza APIs(*SNTP Time Synchronization*) para que possa ser setado o tempo de sincronização periódica do *timestamp* interno(Espressif Systems, 2024). Para fim de sincronização, foi definido para esse projeto que, a cada hora, o microcontrolador faça a sincronização novamente, disponível no Apêndice A.4.

4.3 Configuração do ADC

O ESP32-C6, um System-on-Chip (SoC) da Espressif Systems, integra um Conversor Analógico-Digital (ADC) com resolução de 12 bits. A compreensão de seu funcionamento, modos de operação e a necessidade de calibração é crucial para o desenvolvimento de sistemas embarcados precisos e confiáveis.

O ESP32-C6 emprega um ADC do tipo SAR (Successive Approximation Register), conhecido por seu balanço entre velocidade, precisão e consumo de energia. Este ADC é capaz de converter uma tensão analógica de entrada em um valor digital que varia de 0 a 4095 ($2^{12} - 1$), dada sua resolução de 12 bits.

A tensão de referência interna (Vref) do ADC é nominalmente de 1.1V (1100 mV). Para permitir que o ADC meça tensões de entrada que excedem essa Vref (um requisito comum em muitas aplicações), o ESP32-C6 incorpora um atenuador ajustável. Este atenuador escalona a tensão de entrada antes que ela seja convertida, expandindo a faixa de medição. As opções de atenuação disponíveis são 0 dB, 2,5 dB, 6 dB e 12 dB. Com a atenuação de 12 dB, por exemplo, a faixa de medição efetiva do ADC pode ser estendida para aproximadamente 3,3 V. Contudo, é crucial lembrar que a tensão de entrada nos pinos do ADC nunca deve ultrapassar 3.6V, pois isso pode causar danos permanentes ao chip.

4.3.1 One shot - Mode

O ADC (Conversor Analógico-Digital) no ESP32-C6 oferece diversas opções de operação, mas o *"Oneshot Mode"*(modo de leitura única) é a escolha mais comum para aplicações

de sensoriamento que exigem baixa frequência de amostragem e alta eficiência energética. As vantagens do "*Oneshot Mode*"se estendem ao controle determinístico que o desenvolvedor adquire sobre o momento exato da aquisição de dados, ideal para sistemas onde as leituras não precisam ser amostradas em altas frequências. Além disso, a simplicidade de implementação da interface de programação para esse modo facilita sua integração em diversas aplicações de IoT (Espressif-Adc, 2024).

Apesar de suas capacidades, o ADC do ESP32-C6, como muitos ADCs integrados em SoCs, apresenta certas não-idealidades que tornam a calibração um passo indispensável para a obtenção de medições precisas.

O ESP-IDF (*Espressif IoT Development Framework*) fornece uma API de calibração robusta para mitigar essas não-idealidades. A abordagem recomendada é o esquema de "*curve fitting*"(*ADC_CALI_SCHEME_CURVE_FITTING*), que utiliza dados de calibração pré-gravados em eFuses durante a fabricação do chip. Esta técnica ajusta a curva de resposta do ADC, mapeando o valor bruto lido para uma tensão real em milivolts, considerando a Vref específica do chip e suas características de não-linearidade para a atenuação configurada .

A utilização de funções como *adc_cali_create_scheme_curve_fitting* para inicializar o esquema de calibração e *adc_cali_raw_to_voltage* para converter as leituras brutas em milivolts é fundamental para assegurar a acurácia dos dados adquiridos. Sem essa calibração, as leituras do ADC seriam apenas aproximações, potencialmente levando a comportamentos erráticos ou a decisões incorretas em sistemas que dependem de medições de tensão precisas (Espressif, 2024).

Inicialmente, no Apêndice A.5 encontra-se a configuração, em que a estrutura `adc_oneshot_unit_init_cfg_t` é empregada para definir os parâmetros de inicialização da unidade ADC, permitindo a seleção do barramento e a desativação do modo Ultra-Low Power (ULP) para otimização do consumo energético. Em seguida, a função `adc_oneshot_new_unit` é responsável por instanciar uma nova unidade ADC configurada para operação no modo One-shot.

Complementando a configuração do hardware, a estrutura `adc_oneshot_chan_cfg_t` é utilizada para especificar parâmetros essenciais do canal ADC, incluindo sua resolução e o nível de atenuação desejado, o qual ajusta a faixa de tensão de entrada mensurável. Para assegurar a precisão das leituras, o processo de calibração é iniciado com a estrutura `adc_cali_curve_fitting_config_t`. Esta é fundamental para fornecer os parâmetros de configuração do ADC, permitindo ao sistema construir uma curva de calibração ideal que compensa as não-linearidades e as variações da tensão de referência (*Vref*) do *hardware*. Por fim, a função `adc_cali_raw_to_voltage` converte as leituras brutas obtidas do ADC em valores de tensão expressos em milivolts, aplicando as correções derivadas da curva de calibração.

4.4 Regime Permanente

O monitoramento contínuo do módulo fotovoltaico utiliza uma arquitetura de *buffers* em cascata para consolidar as leituras em regime permanente. Através da função `After_Acquisition`(Apêndice A.6), as médias matemáticas oriundas da primeira camada de aquisição são agrupadas em vetores intermediários. Quando a capacidade máxima destes vetores é atingida, o algoritmo calcula a média final correspondente a uma janela temporal de três segundos e ativa a sinalização global `event_flag_3s`.

Esta consolidação de dados é orquestrada pela rotina `Continuous_Energy` (Apêndice A.6), que atualiza simultaneamente a Tensão, a Corrente e a Temperatura. Assim que a sinalização é ativada, a tarefa principal de rede, gerenciada no arquivo principal `app_main` (Apêndice A.1), coleta estes parâmetros agregados e os transmite via MQTT para o *dashboard*. Este método reduz o tráfego de comunicação e garante a supervisão eficiente da geração de energia sob condições normais.

4.4.1 Medição de tensão

Para a validação das leituras de tensão, faz-se necessária a caracterização prévia do circuito de condicionamento de sinal associado ao transdutor de tensão por efeito Hall, modelo LV 25-P. Conforme ilustrado no diagrama esquemático da Figura 3.4, o conversor analógico-digital (ADC) do microcontrolador ESP32-C6 foi configurado para operar na faixa de 0 a 3,3 V, correspondendo linearmente à faixa de tensão condicionada.

A metodologia de caracterização consistiu na montagem de um ensaio experimental utilizando uma fonte de alimentação CC regulável de bancada (modelo Politerm POE 16E) para emular os níveis de tensão a serem monitorados pelo sistema. Como instrumento de calibração e referência, conectou-se um multímetro digital em paralelo com os terminais de entrada do transdutor LV 25-P, operando na função de voltímetro. Simultaneamente, a tensão de saída do circuito de condicionamento foi aferida diretamente no pino correspondente à entrada analógica do ADC do ESP32-C6, utilizando um segundo voltímetro de precisão para registrar o sinal enviado ao microcontrolador.

O ensaio foi conduzido aplicando-se varreduras incrementais na tensão da fonte, iniciando em 0 V até atingir o limite físico de entrega do equipamento de bancada, correspondente a 33 V. A disposição física dos equipamentos e a arquitetura da bancada de ensaios desenvolvida para esta etapa de caracterização são apresentadas na Figura 4.2.

Os dados empíricos obtidos durante o ensaio de calibração do transdutor de tensão LV 25-P estão consolidados na Tabela 4.1. O conjunto de dados correlaciona os níveis de tensão reais aplicados na entrada do transdutor com as respectivas tensões geradas na saída do circuito de condicionamento e lidas pelo canal do ADC do microcontrolador.

A partir do mapeamento desses pontos experimentais, os valores foram importados para

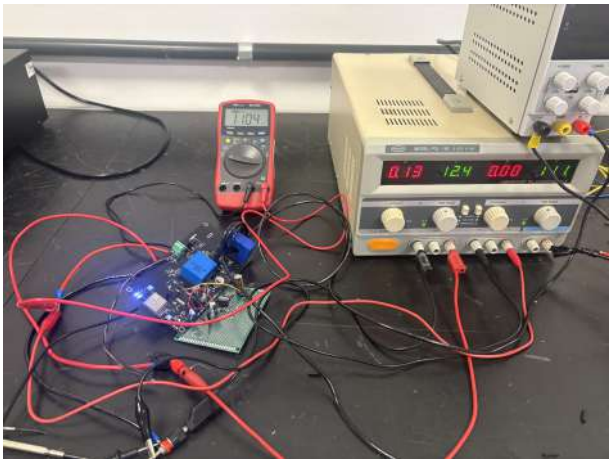


Figura 4.2 – Disposição experimental da bancada de testes para caracterização do circuito de tensão.

Tabela 4.1 – Relação entre a tensão lida pelo ADC e a tensão de referência.

Tensão no ADC (V)	**Tensão de Referência (V)**
0,002	0,0048
0,184	5
0,367	10
0,549	15
0,733	20
0,915	25
1,026	28
1,099	30
1,206	32,89

Fonte: autoria própria.

o *software* MATLAB com o propósito de modelar a curva de caracterização estática do sensor. Utilizando técnicas de regressão linear para aproximar o comportamento do transdutor e de seu circuito associado, determinou-se a curva de resposta apresentada na Figura 4.3. Este procedimento viabilizou a extração matemática do ganho do sensor (coeficiente angular) e do *offset* (coeficiente linear) do sistema de o código pode ser conferido no Apêndice B.1.

Identificados os coeficientes de calibração de ganho igual a 27.30 e *offset* de -0.01, estes parâmetros foram embarcados na lógica de processamento do *firmware* instalado no ESP32-C6.

Finalmente, para consolidar a validação do subsistema de medição de tensão, realizou-se um ensaio comparativo em tempo real. Os valores de tensão instantâneos medidos diretamente na bancada por meio do multímetro digital de referência foram confrontados com as respectivas informações recebidas e exibidas na plataforma de monitoramento remoto. Essa análise comparativa direta permitiu quantificar o erro de medição do protótipo e comprovar a acurácia do algoritmo de calibração desenvolvido. Para confirmar a leitura de tensão, a Figura 4.4 mostra que na plataforma está chegando 32.46V e no multímetro está 32.89.

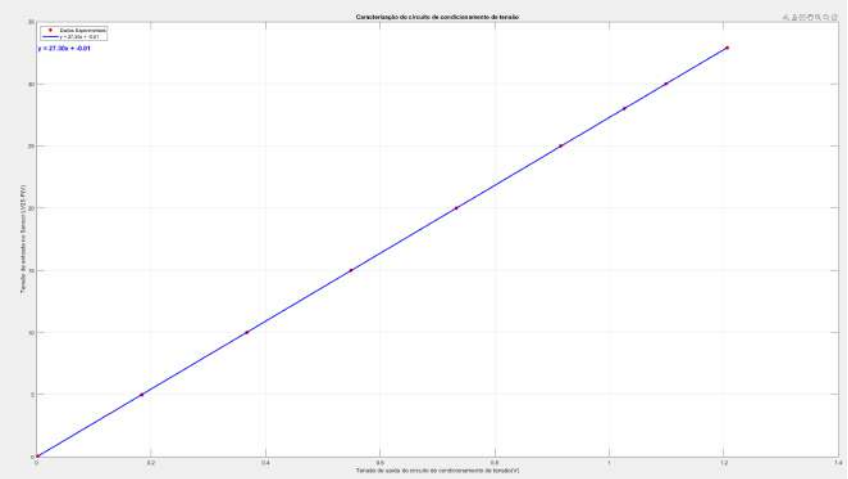


Figura 4.3 – Curva de caracterização do transdutor de tensão LV 25-P gerada via MATLAB.Fonte: autoria própria

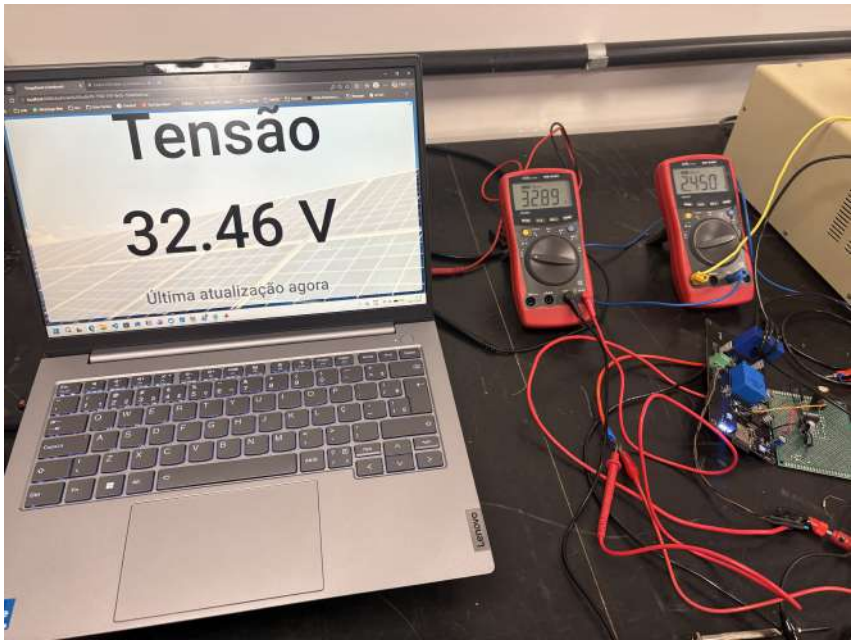


Figura 4.4 – Validação do envio de tensão em regime permanente.Fonte: autoria própria

Para avaliar a precisão do sistema, calculou-se o erro relativo percentual ($E_{\%}$) entre a tensão aferida pelo protótipo (V_{medido} = 32,46 V) e a referência do multímetro (V_{ref} = 32,89 V), utilizando a Equação 4.1:

$$E_{\%} = \left(\frac{|V_{medido} - V_{ref}|}{V_{ref}} \right) \times 100 \tag{4.1}$$

Aplicando os dados experimentais, obtém-se o erro de medição do sistema (Equação 4.2):

$$E_{\%} = \left(\frac{|32{,}46 - 32{,}89|}{32{,}89} \right) \times 100 \approx 1{,}31\% \tag{4.2}$$

O erro de apenas 1,31% atesta a precisão do circuito de condicionamento e valida a calibração implementada no *firmware*.

4.4.2 Medição de corrente

De modo análogo ao procedimento realizado para a grandeza de tensão, a etapa de validação da medição de corrente demandou a caracterização prévia de seu respectivo circuito de condicionamento de sinal.

Para a condução deste ensaio, fez-se necessária a inserção de uma carga no circuito para emular o comportamento elétrico e o respectivo dreno de corrente de um módulo fotovoltaico. As cargas projetadas para esta finalidade foram compostas por uma lâmpada automotiva a LED de 76 W e duas lâmpadas halógenas de 21 W cada. Estes três componentes foram associados em paralelo e submetidos a uma fonte de tensão ajustada em 12 V, resultando em uma de corrente nominal de aproximadamente 5,5 A.

Com o intuito de caracterizar o transdutor de corrente em uma faixa operacional mais ampla, aproximando-se dos valores máximos esperados no sistema fotovoltaico real, adotou-se a técnica de multiplicação da força magnetomotriz por meio do aumento do número de espiras no condutor primário. Como transdutores dessa categoria realizam a medição baseando-se no campo magnético gerado pelo fluxo de elétrons, o cabo de alimentação do banco de cargas foi transpassado quatro vezes pela janela central do sensor ($N = 4$). Como resultado, a corrente efetiva detectada pelo dispositivo é multiplicada pelo número de passagens. Esta abordagem viabilizou a simulação e a leitura de correntes de até 22 A (4 vezes 5,5 A) utilizando uma carga de potência elétrica consideravelmente menor. A disposição deste ensaio é ilustrada na Figura 4.5.

Durante o ensaio, a corrente real que circulava pela carga foi monitorada utilizando um multímetro digital associado em série, operando na função de amperímetro, servindo como instrumento de referência. Paralelamente, os níveis de tensão condicionados e entregues ao ADC do microcontrolador ESP32-C6 foram registrados a cada incremento. A Tabela 4.2 expõe a relação empírica entre a corrente nominal da carga, corrente efetiva no sensor e a tensão correspondente na entrada analógica.

De posse desses dados experimentais, os valores foram importados para o *software* MATLAB (código disponível no Apêndice B.2) para a execução da modelagem matemática. Aplicando-se técnicas de regressão linear para correlacionar a saída do ADC com a corrente efetiva, extraiu-se a curva de resposta do sensor, ilustrada na Figura 4.6.

Esta análise proporcionou a extração direta do ganho, equivalente a 6,72, e do *offset*, correspondente a -0,23. Parâmetros estes que foram embutidos no algoritmo do *firmware* para o processamento definitivo das leituras. No Apêndice A.9 é possível ver os valores.

Para consolidar a etapa, realizou-se um ensaio de validação, no qual os dados de

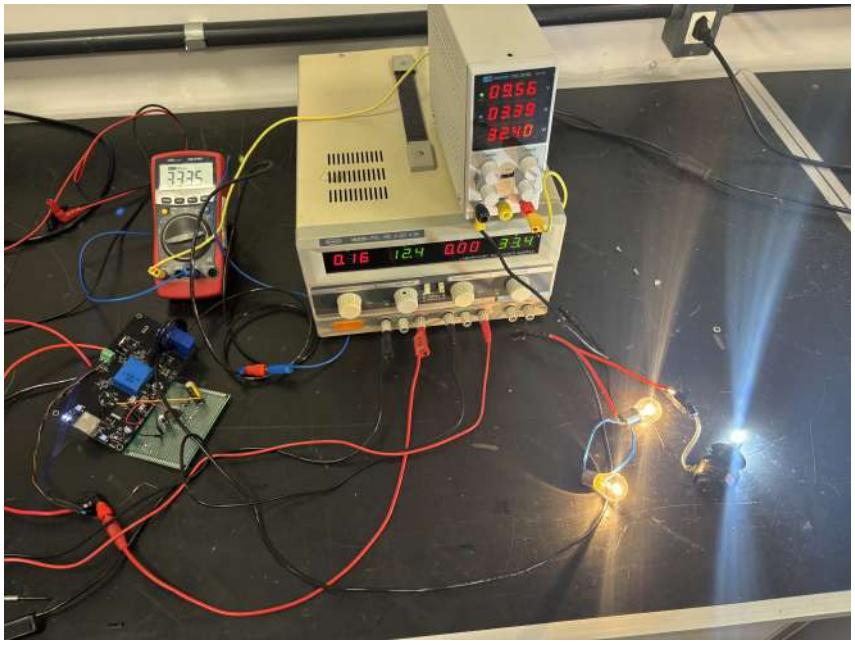


Figura 4.5 – Disposição do banco de cargas para a caracterização do sensor de corrente.Fonte: autoria própria

Tabela 4.2 – Relação empírica entre a tensão lida pelo ADC e as correntes de referência.

Tensão no ADC (V)	**Corrente Nominal na Carga (A)**	**Corrente Efetiva no Sensor (4×) (A)**
0,060	0,000	0,000
0,352	0,536	2,144
0,780	1,246	4,984
1,082	1,750	7,000
1,532	2,495	9,980
1,840	3,000	12,000
2,204	3,750	15,000
2,750	4,500	18,000
3,000	5,000	20,000

Fonte: autoria própria.

corrente enviados pelo protótipo e renderizados na plataforma de monitoramento foram confrontados com as aferições visuais do amperímetro de referência. Esta comparação atestou o funcionamento e a exatidão do circuito de condicionamento, bem como do algoritmo de conversão implementado. Na Figura 4.7 é possível ver que a placa mediu um valor de 9,81 A e o multímetro (configurado como amperímetro) mediu 2,458 A, que, multiplicado por 4 (número de voltas de fio no sensor), é equivalente a 9,832 A.

De forma análoga, a precisão do sistema de medição de corrente foi validada calculando-se o erro relativo percentual. O valor aferido pelo protótipo foi de $I_{medido} = 9{,}81$ A. Como a leitura do multímetro foi de 2,458 A em um condutor com 4 espiras, a corrente efetiva de referência é $I_{ref} = 2{,}458 \times 4 = 9{,}832$ A.

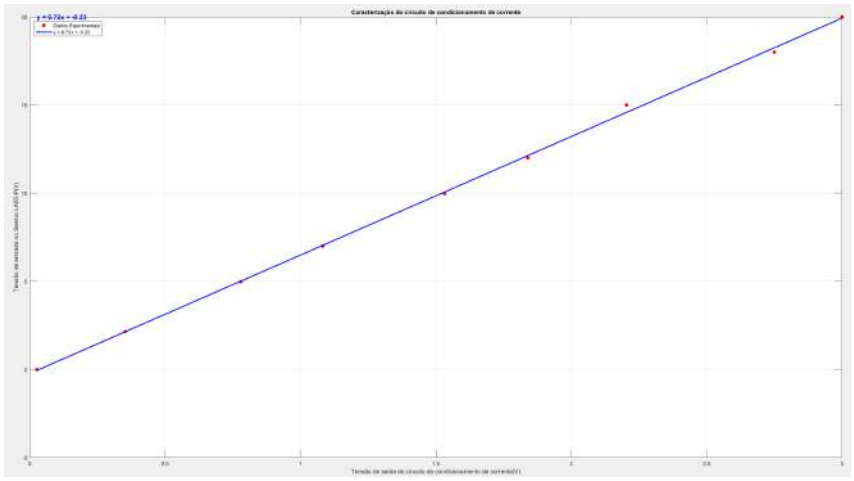


Figura 4.6 – Curva de caracterização do transdutor de corrente gerada via MATLAB.

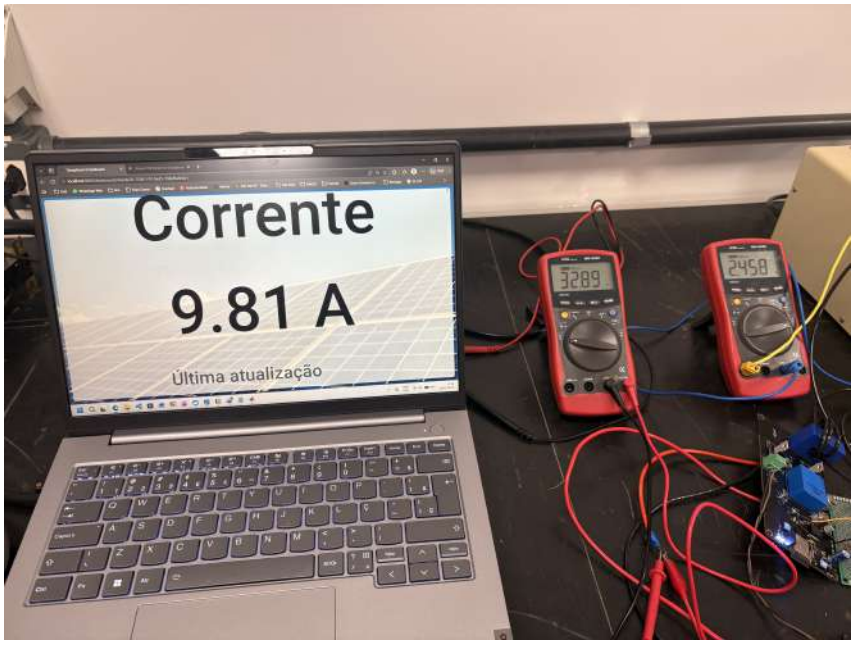


Figura 4.7 – Validação do envio de corrente em regime permanente.Fonte: autoria própria

Aplicando a equação de erro relativo, obtém-se o desvio do sistema para a corrente (Equação 4.4):

$$E_{\%} = \left(\frac{|I_{medido} - I_{ref}|}{I_{ref}} \right) \times 100 \tag{4.3}$$

$$E_{\%} = \left(\frac{|9{,}810 - 9{,}832|}{9{,}832} \right) \times 100 \approx 0{,}22\% \tag{4.4}$$

O erro resultante de apenas 0,22% evidencia uma excelente precisão na leitura, confirmando a eficácia do circuito de condicionamento e o sucesso da técnica de multiplicação por espiras.

4.4.3 Medição de temperatura

A etapa preliminar para a validação da medição de temperatura consiste na caracterização do sistema de condicionamento de sinal do sensor. O transdutor selecionado para esta aplicação é um termistor do tipo NTC (*Negative Temperature Coefficient*), componente cuja principal característica é apresentar uma resistência elétrica inversamente proporcional à temperatura a que é submetido.

Para a realização do ensaio de caracterização, o termistor de referência (termistor do alicate amperímetro digital HDC3000 da Minipa) e a placa de aquisição do projeto foram acondicionados no interior de uma caixa isolada, confeccionada em material PET-G, juntamente com um termistor de referência devidamente calibrado. A grandeza monitorada pelo sistema embarcado foi a tensão elétrica, em Volts, presente na entrada do canal do Conversor Analógico-Digital (ADC) correspondente ao sensor, que foi sendo enviada à plataforma a cada 3s.

O incremento térmico no interior da câmara foi promovido pela injeção de ar quente por meio de um soprador térmico. A temperatura de saída do equipamento foi fixada em 150 °C (valor mínimo suportado pelo aparelho), enquanto a vazão de ar foi ajustada de forma incremental. Devido à inércia térmica inerente aos processos de aquecimento, adotou-se um intervalo de estabilização de 5 minutos a cada incremento de vazão, assegurando uma elevação térmica gradual e homogênea na caixa. A montagem experimental deste ensaio é ilustrada na Figura 4.8.

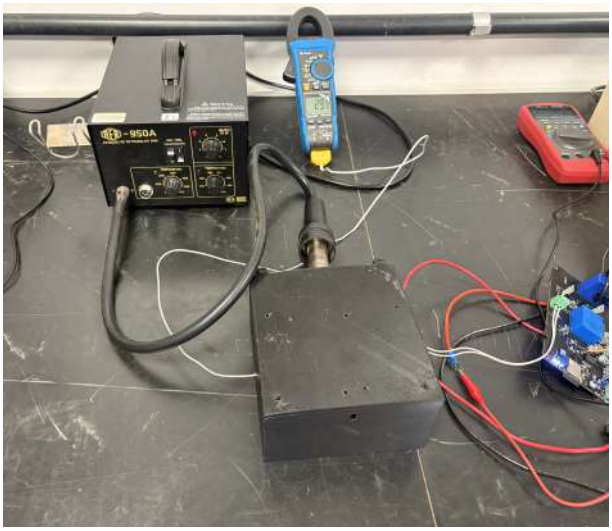


Figura 4.8 – Montagem experimental para caracterização do sistema de temperatura.

Durante o processo, foram registrados simultaneamente os valores de tensão lidos pelo ADC do microcontrolador e a temperatura correspondente aferida pelo sensor de referência. A Tabela 4.3 apresenta a correlação obtida entre esses dados empíricos.

Com o intuito de determinar os parâmetros de calibração do sensor (ganho e *offset*), os

Tabela 4.3 – Relação entre a tensão lida pelo ADC e a temperatura de referência.

Tensão no ADC (V)	Temperatura de Referência (°C)
1,8	23,7
1,52	32,4
1.44	40
1,24	45,2
1,11	50,7
1,03	53,8
0,95	56,5
0,86	60,3
0,80	63,3
0,64	70

dados experimentais foram processados por meio de uma rotina computacional desenvolvida no *software* MATLAB. A partir do processamento matemático dos dados, foi possível extrair a curva de resposta do sensor, ilustrada na Figura 4.9 e o código realizado se encontra no Apêndice B.3.

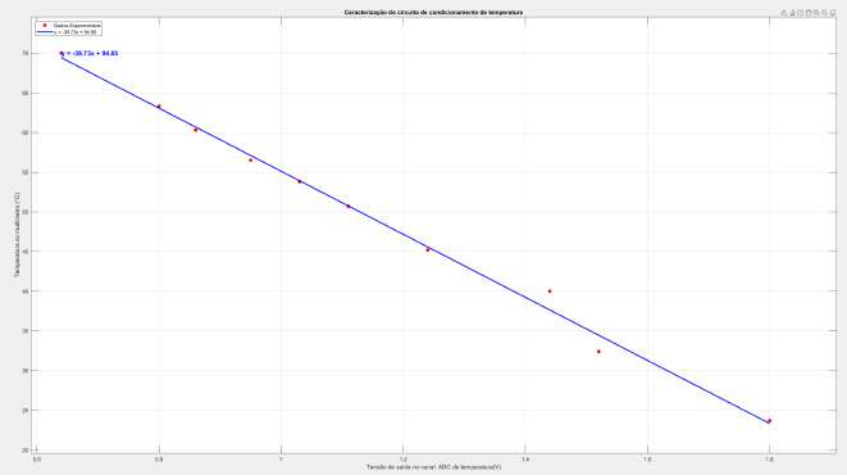


Figura 4.9 – Curva de resposta do sensor de temperatura gerada via MATLAB.Fonte: autoria própria

Por meio da análise da curva obtida, determinou-se que o sistema possui um ganho de -39.73 e um *offset* de 94.85, parâmetros estes que foram posteriormente implementados no *firmware* para o cálculo preciso da temperatura, visto no Apêndice A A.9.

A fim de confirmar que a temperatura que está chegando na plataforma é a mesma que está sendo lida, a Figura 4.10 representa o que está chegando na plataforma a cada 3 s e um termistor de referência. À esquerda da Figura é possível ver que na plataforma está chegando o valor de 20,24°C e no termistor de referência está medindo 19,9 °C.

Para o sensor de temperatura, a validação seguiu a mesma metodologia, comparando a temperatura calculada pelo microcontrolador (T_{medido} = 20,4 °C) com a leitura do termistor de referência (T_{ref} = 19,9 °C).

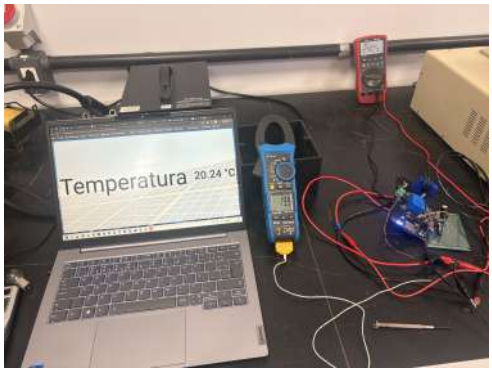


Figura 4.10 – Comparação entre a temperatura medida e temperatura lida. Fonte: autoria própria

Aplicando o cálculo de erro relativo percentual, tem-se a formulação e o respectivo resultado na Equação 4.6:

$$E_{\%} = \left(\frac{|T_{medido} - T_{ref}|}{T_{ref}} \right) \times 100 \tag{4.5}$$

$$E_{\%} = \left(\frac{|20{,}4 - 19{,}9|}{19{,}9} \right) \times 100 \approx 2{,}51\% \tag{4.6}$$

O erro de aproximadamente 2,51% (correspondente a um desvio absoluto de apenas 0,5 °C) está dentro da tolerância típica esperada para termistores do tipo NTC, validando de forma satisfatória a curva de calibração modelada.

4.5 Detecção de Eventos

Embora a telemetria em janelas de três segundos seja ideal para o acompanhamento macroscópico da eficiência fotovoltaica, o cálculo de médias temporais extensas atua intrinsecamente como um filtro passa-baixa. Caso ocorresse um surto de tensão ou um curto-circuito na instalação, o pico transiente e destrutivo seria matematicamente diluído ao ser dividido pelas milhares de amostras acumuladas na janela, o que inviabilizaria o acionamento preventivo da proteção elétrica.

Para mitigar esta vulnerabilidade topológica e garantir que o seccionamento do módulo ocorra na escala de milissegundos, o *software* implementou uma via de processamento concorrente estritamente focada na detecção de anomalias. Por meio da função `Instantaneous_Energy` (Apêndice A.7), cada amostra convertida individualmente é cruzada com os limiares de segurança e com a lógica de histerese antes de qualquer agregação estatística.

Se o valor instantâneo violar as tolerâncias permitidas — alterando o estado de operação das grandezas (`Voltage.state == false` ou `Current.state == false`) — o microcontrolador cessa temporariamente o foco no regime de médias e aciona a flag de contenção

`gravando_evento`. A partir deste instante crítico, o foco do sistema comuta imediatamente para o rastreamento da anomalia, engatilhando a captura de uma frequência de 960 Hz das variáveis instantâneas (`NUM_amostras`) para compor a amostragem da falha.

4.5.1 Evento de sobrecorrente.

Para o registro de um evento de sobrecorrente, é necessário que a grandeza medida ultrapasse os valores de referência pré-configurados no *firmware*, Apêndice A.9. Conforme o código apresentado no Apêndice A.8, foram estabelecidos dois limiares operacionais: um limite superior (*max_threshold*), fixado em 15 A para detectar a anomalia, e um limite inferior (*min_threshold*) de 13 A para determinar a normalização do sistema.

Além da lógica de histerese descrita anteriormente, implementou-se uma janela de tempo para validar a ocorrência do evento, atuando como um filtro contra falsos positivos decorrentes de picos transitórios. No respectivo apêndice, as variáveis referentes ao tempo de confirmação da falha (*tempo_confirmacao_ms*) e ao tempo de atraso para o acionamento do *MOSFET* de proteção do módulo (*tempo_acionamento_mosfet_ms*) foram configuradas em 5 ms e 50 ms, respectivamente. Para efeito de teste, foi inserida uma tensão de 12V nas cargas (três lâmpadas automotivas em paralelo, consumindo 5.5A) de teste, assim conseguindo uma corrente de 5,5A. Como foram dadas 4 voltas ao redor do primário do sensor LA55P 3.5, temos uma corrente de aproximadamente 20A que o sensor irá detectar. Após a detecção do evento,o sistema foi projetado para que o microcontrolador transmita os valores medidos de tensão e corrente com uma alta taxa de atualização, na ordem de 1 ms (referente à primeira camada de agregação de dados). Essa elevada resolução temporal é um recurso de diagnóstico fundamental, pois permite que o usuário analise o comportamento transitório das grandezas elétricas e consiga identificar com precisão a causa raiz que desencadeou a falha no sistema.

A Figura 4.11 evidencia o ocorrido: quando a corrente passa do limiar superior predefinido (15A), abre-se a janela de evento. Assim, o microcontrolador começa a transmitir os dados a uma frequência de aproximadamente 960 Hz. E, após essa corrente passar do limiar inferior (13A), o evento é encerrado e o microcontrolador volta a transmitir em regime permanente.



Figura 4.11 – Gráfico gerado após uma ocorrência de evento de sobrecorrente. Fonte: autoria própria

4.5.2 Evento de sobretensão

De maneira análoga, a detecção de um evento de sobretensão exige que o nível de tensão aferido ultrapasse as margens de segurança estabelecidas no *firmware*, Apêndice A.9. Conforme detalhado no Apêndice A.8, foram definidos dois limiares para o monitoramento desta grandeza: um limite superior (*max_threshold*), fixado em 60 V para a detecção da anomalia, e um limite inferior (*min_threshold*), estipulado em 55 V para caracterizar a normalização do sistema. Mas, para efeito de teste, foram diminuídos esses valores (*max_threshold*) para 30V e (*min_threshold*) para 25V, pois a fonte de bancada disponível tem limite de 32V.

Aplicando a mesma lógica de histerese adotada para a corrente, o algoritmo emprega temporizadores para validar a ocorrência da sobretensão, atuando como um filtro contra falsos positivos causados por transientes ou ruídos elétricos na rede. No mesmo apêndice, as variáveis responsáveis por ditar o tempo de confirmação da falha (*tempo_confirmacao_ms*) e o tempo de atraso para o acionamento do *MOSFET* de proteção do módulo (*tempo_acionamento_mosfet_ms*) foram configuradas, respectivamente, em 5 ms e 50 ms.

Para a validação prática do evento de sobretensão, provocou-se uma elevação intencional no nível de tensão aplicado à entrada do transdutor LV 25-P, mantendo-se a anomalia por um período contínuo de 5 s.

Durante a vigência de um evento desta natureza, o sistema foi projetado para que o microcontrolador transmita os valores medidos de tensão e corrente com uma alta taxa de atualização, na ordem de 1 ms (referente à primeira camada de agregação de dados). Essa elevada resolução temporal é um recurso de diagnóstico fundamental, pois permite que o usuário analise o comportamento transitório das grandezas elétricas e consiga identificar com precisão a causa raiz que desencadeou a falha no sistema. A Figura 4.12 mostra que, após a placa evidenciar uma tensão maior de 30V, abre-se a janela de evento. Durante aproximadamente 5 s, ele fica com o evento ativo e, quando é detectada uma tensão abaixo ou igual ao limiar inferior, o evento é finalizado e o sistema volta ao normal.

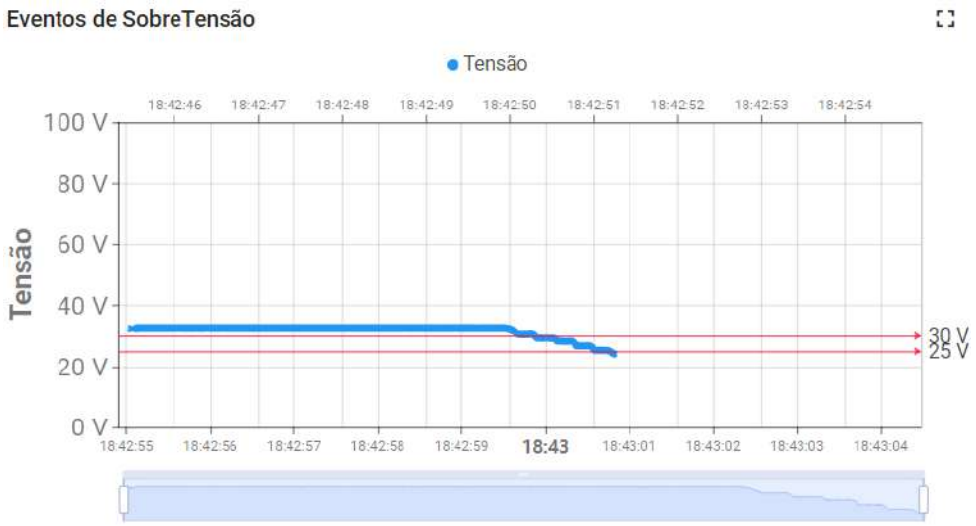


Figura 4.12 – Gráfico gerado após uma ocorrência de evento de sobretensão Fonte: autoria própria

4.5.3 Evento de sobretemperatura.

No que tange ao monitoramento térmico, o registro de um evento de sobretemperatura opera sob o mesmo princípio de acionamento por limiar, porém apresenta diferenças arquiteturais significativas em relação às grandezas elétricas. Neste caso, o microcontrolador atua apenas na aquisição e no envio periódico dos dados, transmitindo as leituras de temperatura para a plataforma de monitoramento em intervalos fixos de 3 segundos.

A detecção da anomalia térmica é processada diretamente pela própria plataforma. Quando o valor recebido pelo sistema ultrapassa o limite máximo pré-configurado (estipulado em 70 °C), o evento é imediatamente classificado e o alerta é ativado na interface. O encerramento deste evento ocorre de forma automática assim que a temperatura reportada decresce e atinge um limiar inferior de normalização (estipulado em 60 °C), restaurando o status visual de operação normal no painel.

Cabe ressaltar que este alerta possui caráter exclusivamente informativo. A sinalização de um evento de sobretemperatura gera apenas uma indicação visual e o respectivo registro no histórico da plataforma, não exercendo qualquer efeito de controle sobre a placa e não desencadeando o acionamento do MOSFET de proteção local. Para efeito de teste, foi aquecido o termistor com o soprador térmico e foi analisado na plataforma se o evento iria ser acionado. Após o acionamento, esperou alguns segundos (pois a temperatura é recebida pela plataforma de 3 em 3 segundos). Na Figura 4.13 é possível visualizar a ocorrência do evento.

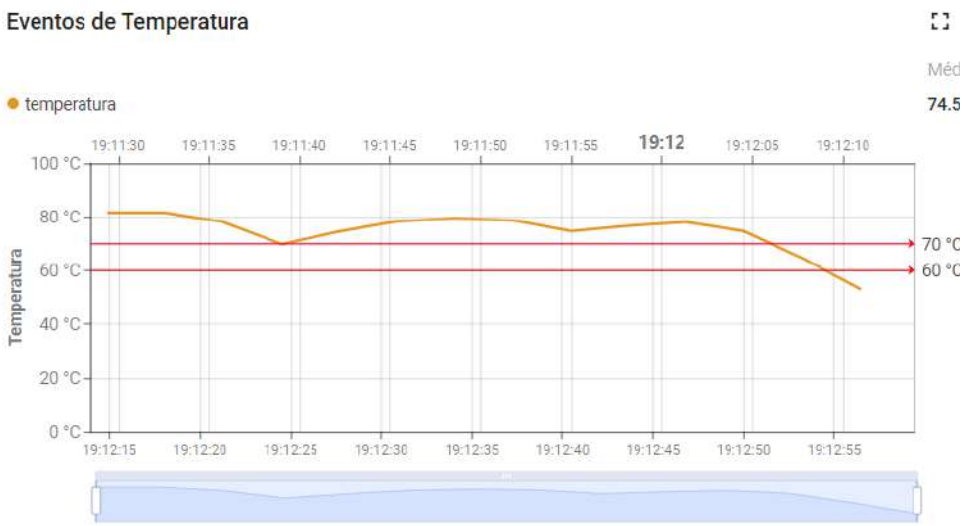


Figura 4.13 – Gráfico gerado após uma ocorrência de evento de sobretemperatura. Fonte: autoria própria

4.6 Envio de dados

O desempenho do sistema de proteção e monitoramento pode ser validado através da análise dos pacotes de dados (cargas úteis ou *payloads*) trafegados entre o microcontrolador ESP32-C6 e a plataforma ThingsBoard via protocolo MQTT. A estruturação desses pacotes foi dividida em três categorias principais, refletindo os diferentes estados operacionais do arranjo fotovoltaico: monitoramento em regime permanente, atuação remota e oscilografia de falhas transientes.

4.6.1 Monitoramento em Regime Permanente

Sob condições nominais de irradiância e temperatura, o sistema prioriza a economia de banda de rede enviando dados sumarizados. O pacote de telemetria de rotina, gerado a cada ciclo de agregação de 3 segundos, encapsula as médias aritméticas das grandezas elétricas e o estado lógico atual da chave de proteção.

O Código 4.1 apresenta a captura real de um envio de rotina, em que o módulo operava com tensão de 25,55 V, corrente de 2,92 A e temperatura na placa de 58,89°C, mantendo o seccionamento aberto (`"estado_MOSFET": True`). O timestamp atual do microcontrolador que foi sincronizado com a da internet atualizado durante a inicialização do WiFi.

```
1 {"timestamp": 1784409719, "tensao": 25.55, "corrente": 2.92, "temperatura":
      58.89, "estado_MOSFET": True}
```

Código 4.1 – Carga útil de telemetria em regime permanente enviada ao Broker.

4.6.2 Comunicação Bidirecional e Acionamento Remoto

A arquitetura IoT implementada permite que a proteção atue não apenas de forma autônoma, mas também responda a intervenções externas. Quando o operador altera o estado da proteção pelo *dashboard*, a plataforma ThingsBoard publica uma mensagem no tópico de Atributos (`v1/devices/me/attributes`).

Conforme demonstrado no Código 4.2, o microcontrolador intercepta o comando assíncrono contendo a chave `"new_state_MOSFET"`. A transição para `false` instrui para que os MOSFETs de segurança mantenham-se fechados, permitindo a geração contínua do módulo.

```
W (69924) MQTT: TOPICO DE ACAO RECEBIDO: v1/devices/me/attributes
W (69925) MQTT: PAYLOAD DE ACAO RECEBIDO: {"new_state_MOSFET":false}
```

Código 4.2 – Recepção de comando remoto para atuação do atuador de estado sólido.

4.6.3 Registro de Eventos Críticos

A principal funcionalidade de *software* do sistema embarcado é a sua capacidade de reagir a falhas elétricas em milissegundos e fornecer um diagnóstico claro da anomalia. Quando a lógica de histerese detecta uma violação nas tolerâncias operacionais, o fluxo de dados altera seu comportamento drasticamente, executando uma sequência de três etapas de notificação.

Primeiramente, assim que o atraso de validação confirma o evento (neste caso, um evento de sobrecorrente), um alerta de prioridade máxima é despachado instantaneamente (Código 4.3), sinalizando o estado de emergência para a plataforma antes mesmo do término da falha.

```
I (114503) MQTT: Payload Alerta ON: {"timestamp": 1784411046844, "status":
    "ALERTA", "tipo": "Corrente", "estado_MOSFET": True}
```

Código 4.3 – Transmissão imediata de alerta de evento.

Simultaneamente, o *firmware* aloca vetores dinâmicos para armazenar o comportamento transitório da curva elétrica até o evento ser cessado. Após o preenchimento de 200 amostras para tensão e corrente.O pacote de evento é publicado (Código 4.4).

```
I (129070) MQTT: Payload de evento: {"timestamp_alerta": 1784411049759, "
    tipo": "Corrente", "Tensao_array": [25.60, 25.57, ..., 25.60, 25.60], "
    Corrente_array": [16.00, 16.11, 16.11, 16.11, 16.11, ..., 16.11, 15.99,
    16.11, 15.99, 16.11, 16.00, 16.11, 16.11, ]}
```

Código 4.4 – Pacote dinâmico durante um evento evidenciando

Por fim, ao constatar que a anomalia física cessou (com as grandezas elétricas retornando à margem de segurança configurada no algoritmo), um pacote de encerramento é despachado (Código 4.5). Este documento digitaliza a assinatura do evento, registrando com exatidão a sua duração total (3181 ms), permitindo manutenções preditivas e maior rastreabilidade das falhas de sobrecorrente no arranjo fotovoltaico.

```
I (129309) MQTT: Payload Alerta Off: {"timestamp": 1784411050025, "status":
    "NORMAL", "duracao_real": 3181, "tipo": "Corrente"}
```

Código 4.5 – Sinalização de término do evento e retorno às margens nominais.

4.7 Validação da aquisição de 960 Hz.

Para quantificar o impacto da concorrência do sistema operacional e obter dados precisos sobre o temporizador sob carga de rede, implementou-se um diagnóstico temporal dividido em duas frentes: via *hardware* e via *software*.

4.7.0.1 Diagnóstico via *Hardware* (Osciloscópio)

A validação da taxa de amostragem foi feita alternando o estado de um pino digital (GPIO) durante a execução da rotina de interrupção (ISR) de aquisição do ADC. A análise do sinal quadrado resultante atestou o determinismo isolado do temporizador:

- **Frequência e Período:** O sistema cravou a operação em 960,61 Hz período de 1041 us, atendendo com exatidão aos requisitos do projeto.
- **Tempo de Execução (*Overhead*):** A leitura do ADC exigiu apenas 40,00. O uso de CPU dedicado à tarefa foi calculado em:

$$\text{Uso da CPU (\%)} = \left(\frac{40}{1041}\right) \times 100 \approx 3{,}84\% \tag{4.7}$$

Esse índice comprova a alta eficiência da rotina em nível de *hardware*, garantindo que mais de 96% do processamento permaneça livre para o FreeRTOS. A tensão de transição do sinal (3,28) também confirmou a conformidade com a arquitetura do componente. A Figura 4.14 mostra como ficou a forma de onda gerada por ficar alternando a GPIO para alto e baixo a cada interrupção do timer. Foi colocada a ponta de prova de um osciloscópio na GPIO correspondente.

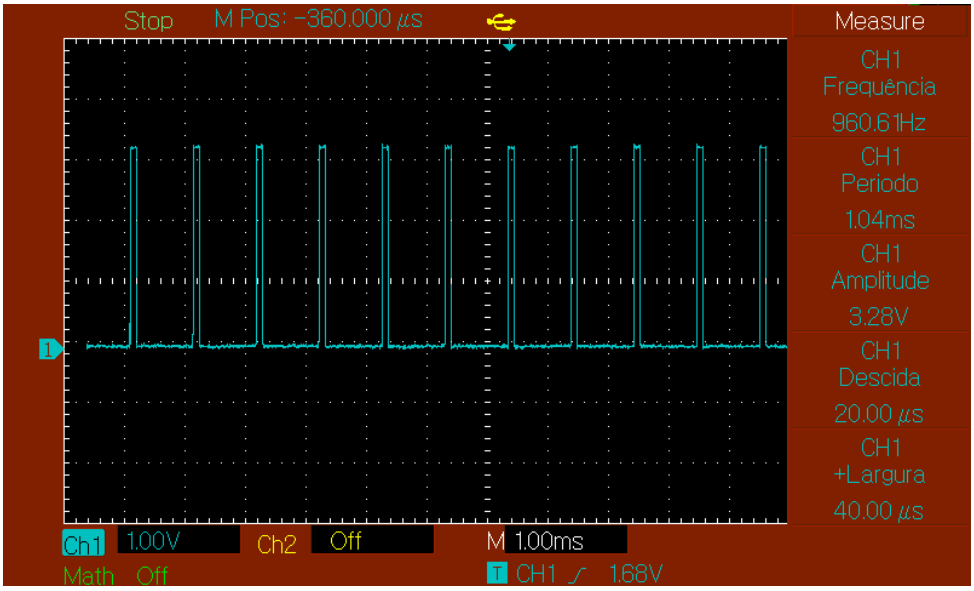


Figura 4.14 – Forma de onda lida pela mudança de estado do pino a cada ISR do Timer. Fonte: autoria própria

4.7.0.2 Diagnóstico via *Software* (*Firmware*)

A instrumentação no código focou no monitoramento do tempo de execução e do desvio do período, disponível no Apêndice A.2, descomentando as partes referentes a medição de tempo.

- **Tempo de Leitura:** Na ISR, para gerar a interrupção do timer de 960 Hz, utilizou-se `esp_timer_get_time()` antes e após a função de leitura para calcular a duração do processo, armazenando apenas o pior tempo de execução observado para identificar gargalos, disponível no Apêndice A.10.
- **Constância da Base de Tempo:** Na tarefa principal (`main.c`), calculou-se o delta de tempo entre as amostras, subtraindo o período ideal para encontrar possíveis atrasos. O sistema retém o desvio máximo e emite um relatório de desempenho a cada lote de 1000 amostras (aproximadamente a cada 1 s, redefinindo as variáveis para a janela de análise seguinte.

4.7.0.3 Análise dos relatórios gerados e justificativa de desempenho.

Durante os ensaios com o envio ativo de telemetria de rotina via MQTT, os relatórios de diagnóstico registraram picos no tempo de leitura do ADC na ordem de 363 us e desvios de período de até 3194 us, no Código 4.6.

```
I (13440) MQTT: Telemetria de rotina enviada.
W (14361) DIAGNOSTICO: --- Relatorio de Desempenho (1s) ---
I (14361) DIAGNOSTICO: 1. Atraso de envio: 3194 us
I (14361) DIAGNOSTICO: 2. Tempo de leitura do ADC : 363 us
```

Código 4.6 – Log para diagnóstico da duração de envio via MQTT.

O tempo da rotina de interrupção 363 us decorre da execução dos cálculos de conversão, agregação de camadas e verificação da detecção de eventos, diretamente após a leitura bruta dos dois canais analógicos (tensão e corrente). Embora seja um tempo considerável para o escopo de uma ISR em sistemas de tempo real, essa abordagem foi adotada arquiteturalmente para garantir a integridade matemática da potência instantânea do módulo fotovoltaico sem a necessidade de filas de processamento assíncrono complexas.

O desvio máximo de amostragem registrado de 3,3 ms é uma consequência intrínseca da topologia de núcleo único (*single-core*) do ESP32-C6 operando sob a pilha TCP/IP. Para assegurar a estabilidade da conexão Wi-Fi, o *driver* de rede demanda a CPU com prioridade absoluta para a manutenção do enlace e transmissão dos pacotes MQTT, suspendendo temporariamente a tarefa do temporizador.

Contudo, este atraso esporádico de 3 ms comprova-se ser aceitável e plenamente seguro para a aplicação proposta. A métrica fundamental para a viabilidade do projeto é o tempo de resposta do sistema de proteção elétrica face à inércia do mecanismo de atuação físico. O *Mosfet* de segurança só irá comutar após 50 ms (como foi predefinido anteriormente, Apêndice A.9). Desta forma, uma supressão transitória de 3 ms na taxa de amostragem induzida pela latência de rede configura-se como um desvio imperceptível para a malha de segurança. Outro fator é que, como se tem um método de agregação de camadas, como o envio em regime permanente é a cada 3s (terceira camada de agregação), esse desvio de 3ms não vai ser perceptível em regime permanente.

Ademais, outro fator que corrobora a mitigação desse impacto é a detecção de eventos adotada para a comunicação. O sistema não realiza o envio contínuo e ininterrupto (*streaming*) de todas as amostras em alta frequência para o servidor. Em operação nominal, a telemetria é despachada em intervalos predefinidos maiores (rotina de 3 s), momento em que ocorre o desvio pontual. A transmissão adensada, em que o microcontrolador é exigido a enviar pacotes de aquisição a 1 ms é disparada unicamente quando o algoritmo local detecta um evento de sobretensão ou sobrecorrente. Sendo a ocorrência de falhas uma condição de exceção, o barramento de rede e a CPU permanecem desobstruídos na esmagadora maioria do tempo operacional. Consequentemente, o sistema não sofre com congestionamentos crônicos e mantém capacidade plena para comandar rotinas de emergência em tempo hábil.

5 Conclusões

Este trabalho apresentou o desenvolvimento e a implementação de um *software* embarcado (*firmware*) focado na monitorização contínua e em tempo real de módulos fotovoltaicos. Através da utilização do *System-on-Chip* (SoC) ESP32-C6, foi possível conceber uma solução robusta, de elevada eficiência energética e com conectividade nativa, capaz de operar ao nível do módulo. O objetivo principal de adquirir dados telemétricos críticos — nomeadamente tensão, corrente e temperatura — e transmiti-los de forma confiável para uma plataforma externa foi alcançado com pleno êxito.

O processo de caracterização dos sensores e calibração do conversor analógico-digital (ADC) garantiu a precisão necessária para o correto funcionamento do sistema. Um dos pontos de maior relevo no desenvolvimento do *firmware* foi a integração do algoritmo de detecção de falhas baseado na lógica de histerese. As análises demonstraram que esta abordagem é altamente eficaz na identificação de eventos anômalos, como sobretensão e sobrecorrente, garantindo a estabilidade da monitorização ao mitigar oscilações transitórias e ruídos que poderiam gerar falsos alarmes.

Além disso, a adoção da comunicação Wi-Fi em conjunto com o protocolo MQTT provou ser a arquitetura ideal para o envio dos dados. A separação lógica entre os tópicos de telemetria periódica e os alertas de eventos críticos permitiu otimizar a largura de banda e assegurar que as falhas sejam notificadas quase instantaneamente, um requisito fundamental para a segurança da instalação.

Desta forma, conclui-se que o sistema proposto representa uma ferramenta valiosa para a manutenção preditiva e para o aumento da segurança em infraestruturas de geração de energia solar. Ao permitir a monitorização granular (painel a painel), facilita-se a identificação precoce de problemas físicos, degradação ou anomalias elétricas, contribuindo diretamente para a maximização do rendimento energético e para o prolongamento da vida útil dos equipamentos.

5.1 Sugestões para Trabalhos Futuros

Apesar dos resultados satisfatórios, o projeto abre margem para diversas evoluções e melhorias. Como propostas para trabalhos futuros, sugere-se:

- **Implementação de *Machine Learning* no *Edge*:** Integrar algoritmos leves de aprendizagem automática diretamente no ESP32-C6 (TinyML) para prever falhas incipientes no módulo fotovoltaico com base no histórico de degradação da curva de tensão e corrente.

- **Expansão da conectividade:** explorar a integração de transceivers LoRa ou protocolos como o Thread/Matter (suportados nativamente pelo ESP32-C6) para viabilizar a comunicação em grandes quintas solares situadas em zonas remotas, onde redes Wi-Fi convencionais não têm cobertura.

Referências

ABSOLAR. **Boletim de Dados do Setor Solar Fotovoltaico**. 2024. Acesso em: 18 maio 2025. Disponível em: https://www.absolar.org.br. Citado na p. 17.

ALMEIDA, R. M. A. de. **PROGRAMAÇÃO DE SISTEMAS EMBARCADOS**. [*S.l.*]: Elsevier, 2021. Citado nas pp. 9, 32 e 34.

ALVES, M.; SILVA, R.; PEREIRA, J. Análise de desempenho de sistemas fotovoltaicos com base em monitoramento elétrico. **Revista Brasileira de Energias Renováveis**, v. 9, n. 1, p. 55–64, 2020. Citado na p. 17.

(ANEEL), A. N. de E. E. **Relatório de Sustentabilidade e Inovação no Setor Elétrico**. 2022. Acesso em: 18 maio 2025. Disponível em: https://www.aneel.gov.br. Citado na p. 17.

CERQUEIRA MARCOS V B.; MASCHIETTO, L. G. Z. A. **Sistemas Operacionais Embarcados**. Porto Alegre: SAGAH, 2021. E-book., 2021. Acesso em: 29 jun. 2025. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9786556902616/. Citado na p. 37.

(EPE), E. de P. E. **Plano Decenal de Expansão de Energia 2032**. 2023. Acesso em: 18 maio 2025. Disponível em: https://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/Plano-Decenal-de-Expansao-de-Energia-PDE. Citado na p. 17.

ESPRESSIF. **Analog to Digital Converter (ADC) Calibration Driver**. 2024. Disponível em: https://docs.espressif.com/projects/esp-idf/en/stable/esp32c6/api-reference/peripherals/adc_calibration.html. Citado na p. 58.

ESPRESSIF-ADC. **Analog to Digital Converter (ADC) Oneshot Mode Driver**. 2024. Disponível em: https://docs.espressif.com/projects/esp-idf/en/stable/esp32c6/api-reference/peripherals/adc_oneshot.html. Citado na p. 58.

Espressif Systems. **ESP-IDF Programming Guide: ESP32-C6**. [*S.l.*], 2024. Disponível em: https://docs.espressif.com/projects/esp-idf/. Acesso em: 15 jul. 2026. Citado na p. 57.

FEIJó, R. **Desenvolvimento de um sistema de monitoramento para caixas de junção de sistemas fotovoltaicos**. 2021. Acesso em: 09 de julho 2025. Disponível em: https://bdm.unb.br/handle/10483/32519. Citado nas pp. 9, 42, 43 e 44.

FELIPE T.A. ; MELO, F. L. Development of an online smart monitoring and diagnosis system for photovoltaic distributed generation. **MDPI-Energies**, 2021. Citado na p. 20.

HUAWEI. **SUN2000-450W-P2 Smart PV Optimizer User Manual**. Shenzhen, China, 2023. Disponível em: https://solar.huawei.com/. Acesso em: 29 jul. 2026. Citado na p. 19.

International Electrotechnical Commission. **Terrestrial photovoltaic (PV) modules – Design qualification and type approval**. Geneva, 2024. Citado na p. 30.

KAMAL, A. I.; DONA, N. R. Utilization of the hall effect to detect leakage current in dc system branch loads using the positive-negative current difference method. **Jurnal Ilmiah Setrum**, v. 13, n. 1, p. 19–25, 2024. Citado na p. 49.

KORNKANOK, U.; DEEON, S.; WONGCHAROEN, S. Safety hysteresis comparator design for transient overvoltage detection. **Indonesian Journal of Electrical Engineering and Computer Science**, v. 34, n. 1, p. 69–80, 2024. Citado na p. 49.

LEM. **Current Transducer LA 55-P**. 2022. Acesso em: 09 de julho 2025. Disponível em: https://www.lem.com/sites/default/files/products_datasheets/la_55-p_v19.pdf. Citado nas pp. 9, 11, 43 e 44.

LEM. **Voltage Transducer LV 25-P**. 2022. Acesso em: 09 de julho 2025. Disponível em: https://www.lem.com/sites/default/files/products_datasheets/lv_25-p_v21.pdf. Citado nas pp. 9, 11 e 42.

MICREL. **MIC4426/4427/4428 Dual 1.5A-Peak Low-Side MOSFET Driver**. 2022. Disponível em: https://lcsc.com/datasheet/lcsc_datasheet_2410121904_Microchip-Tech-MIC4427YN_C185485.pdf. Citado na p. 45.

MURATA. **NXRT15XV103FA1B040**. 2022. Acesso em: 09 de julho 2025. Disponível em: https://www.mouser.com/datasheet/2/281/1/NXRT15XV103FA1B040-2936996.pdf. Citado nas pp. 11 e 44.

NGUYEN-VAN, T.; ABE, R.; TANAKA, K. Mppt and sppt control for pv-connected inverters using digital adaptive hysteresis current control. **Energies**, MDPI, v. 11, n. 8, p. 2075, 2018. Citado na p. 49.

OASIS. **MQTT Version 5.0. OASIS Standard**. 2019. Disponível em: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. Acesso em: 15 jul. 2026. Citado na p. 56.

PEI, T.; HAO, X. A fault detection method for photovoltaic systems based on voltage and current observation and evaluation. **Energies**, MDPI, v. 12, n. 9, p. 1712, 2019. Citado na p. 49.

PEREIRA IVONNE M. DUPONTA, P. C. C. S. C. J. R. I. Iot embedded linux system based on raspberry pi applied to real-time cloud monitoring of a decentralized photovoltaic plant. **Journal of the International Measurement Confederation**, 2017. Citado na p. 21.

PEREIRA, L. M.; COSTA, F. R. Aplicação de iot em monitoramento de usinas fotovoltaicas. *In*: **Canais do Congresso Brasileiro de Automática**. [*S.l.*: *s.n.*], 2021. p. 301–308. Citado na p. 17.

PEREIRA SANDRO C.S. JUCá B, P. C. C. R. I. Iot embedded systems network and sensors signal conditioning applied to decentralized photovoltaic plants. **Journal of the International Measurement Confederation**, 2019. Citado na p. 21.

Real Time Engineers Ltd. **FreeRTOS Real Time Kernel Reference Manual**. [*S.l.*], 2024. Disponível em: https://www.freertos.org/. Acesso em: 15 jul. 2026. Citado na p. 56.

SARWAR MUHAMMADYAQOOBJAVED1, M. H. J. M. a. S. Modular level power electronics (mlpe) based distributed pv system for partial shaded conditions. **MDPI**, 2022. Citado na p. 31.

SELVARAJ, S.; VEERASAMY, M.; KRITHIGA, G. Brief review on solar photovoltaic parameter estimation of single and double diode model using evolutionary algorithms. **International Journal of Engineering Technologies and Management Research**, v. 10, p. 64–78, 01 2023. DOI 10.29121/ijetmr.v10.i1.2023.1291. Citado nas pp. 9 e 27.

SEMI, V. **STP75NNF75-VB Datasheet N-Channel 80 V (D-S) MOSFET**. 2022. Disponível em: https://www.lcsc.com/datasheet/lcsc_datasheet_2410121922_VBsemi-Elec-STP75NF75-VB_C7463538.pdf. Citado na p. 44.

SILVA* FERNANDO C. MELO, J. H. R. L. C. F. A. M. The study and application of evaluation methods for photovoltaic modules under real operational conditions, in a region of the brazilian southeast. **Renewable Energy**, 2018. Citado na p. 21.

SILVESTRE, S.; SILVA, M. A. da; CHOUDER, A.; GUASCH, D.; KARATEPE, E. New procedure for fault detection in grid connected pv systems based on the evaluation of current and voltage indicators. **Energy Conversion and Management**, Elsevier, v. 86, p. 241–249, 2014. Citado na p. 49.

ThingsBoard Authors. **ThingsBoard IoT Platform Documentation**. [*S.l.*], 2024. Disponível em: https://thingsboard.io/docs/. Acesso em: 15 jul. 2026. Citado na p. 57.

VILLALVA, M. G. **Energia solar fotovoltaica : conceitos e aplicações**. [*S.l.*]: São Paulo, SP : Érica,, 2013. Citado nas pp. 9 e 26.

Apêndices

Apêndice A – Código-fonte principal do sistema embarcado

Esse apêndice compõe as principais funções para o entendimento do funcionamento da placa, para a visualização do código completo, está disponível no repositório do projeto. Link do repositório. O código a seguir à função *app_main*, localizada no main.c do projeto, é responsável pela chamada de funções prioritárias e início das duas tarefas(*tasks*) principais.

Arquivo: `main.c` – Funções principais para execução do *firmware*.

```
// ============================================================================
// Ponto de Entrada Principal (Main)
// ============================================================================
void app_main(void) {
    // 1. Inicializa o NVS (Non-Volatile Storage) - Obrigatório para o Wi-Fi
    esp_err_t ret = nvs_flash_init();
    if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
        ESP_ERROR_CHECK(nvs_flash_erase());
        ret = nvs_flash_init();
    }
    ESP_ERROR_CHECK(ret);
    RAPID_Init(); // Inicializa o sistema RAPID
    nvs_flash_init(); // Inicializa o NVS para salvar estados dos pinos
    VerifyRAPID_state(); // Verifica e aplica o estado salvo dos pinos no NVS

    // 2. Criação de Semáforos e Filas Globais
    Transmit = xSemaphoreCreateBinary();
    if (Transmit == NULL) {
        ESP_LOGE("Main", "Falha ao criar o semaphore de transmissão.");
        return;
    }
    Inicializar_Fila_Eventos();

    ESP_LOGI("Sistema", "Tamanho do heap disponível: %lu bytes", (unsigned long)esp_get_free_heap_size());

    // 3. Inicialização dos Serviços de Rede
    Wifi_Start();
    ///Sync_Time(); // Sincroniza o relógio via NTP a cada envio de rotina
```

```
29      mqtt_start(); // Arranca o cliente MQTT UMA  NICA  VEZ aqui!
30
31      // 4. Criação das Tarefas do FreeRTOS
32      // Core 0/1 geridos automaticamente a não ser que utilize
    xTaskCreatePinnedToCore
33      xTaskCreate(Main_Measurement, "Medicao_Principal", 4096, NULL, 2,
    &Handle_Main_Measurement);
34      xTaskCreate(conectarWifi, "Task_Rede_MQTT", 16384, NULL, 1, NULL);
35  }
```

Código A.1 – Função principal do firmware localizada no main.c.

Fonte: Elaborado pelo autor (2026).

```
1   void Main_Measurement(void *param) {
2
3       // Cria o semáforo binário para o Timer
4       SemaphoreHandle_t timer_semaphore = xSemaphoreCreateBinary();
5       if (timer_semaphore == NULL) {
6           ESP_LOGE("Tasks", "Falha ao criar o semaphore do timer.");
7           vTaskDelete(NULL);
8       }
9
10      // Inicialização do Hardware e Parâmetros
11      adc_setup();
12      MOSFET_Init();
13      Initialize_Parameters();
14
15      configure_led();
16      led_strip_set_pixel(led_strip, 0, 0, 0, 255); // Azul (A aguardar
    ligação)
17      led_strip_refresh(led_strip);
18
19      // Inicia o Timer de aquisição (Chama a Instant_Acquisition na ISR)
20      //config_gpios(); // Configura o pino do osciloscópio
21      iniciar_timer(timer_semaphore);
22
23      int64_t tempo_anterior = esp_timer_get_time();
24      //ESP_LOGI("Validação","Tempo inicial: %lld us", tempo_anterior);
25      int64_t max_desvio_us = 0;
26      uint32_t contador_amostras = 0;
27
28      while (1) {
29          // Aguarda o semáforo ser libertado pelo timer de hardware
30          if (xSemaphoreTake(timer_semaphore, portMAX_DELAY) == pdTRUE) {
31
```

```
32              int64_t tempo_atual = esp_timer_get_time();
33              int64_t delta_t = tempo_atual - tempo_anterior;
34              tempo_anterior = tempo_atual;
35
36              int64_t desvio = delta_t - 1000;
37              if (desvio < 0) desvio = -desvio;
38
39              if (desvio > max_desvio_us) {
40                  max_desvio_us = desvio;
41              }
42
43              contador_amostras++;
44
45              // A cada 1000 amostras (aprox. 1 segundo), imprime o
    relatório e reseta[cite: 9]
46              if (contador_amostras >= 1000) {
47                  //ESP_LOGW("DIAGNOSTICO", "--- Relatorio de Desempenho
    (1s) ---");
48                  //ESP_LOGI("DIAGNOSTICO", "1. Atraso de envio: %lld us",
    max_desvio_us);
49                  //ESP_LOGI("DIAGNOSTICO", "2. Tempo de leitura do ADC :
    %lld us", pior_tempo_execucao_adc);
50
51                  // Zera as variáveis para analisar o próximo segundo
52                  max_desvio_us = 0;
53                  pior_tempo_execucao_adc = 0;
54                  contador_amostras = 0;}
55
56
57              // Verifica se o buffer de amostras encheu
58              if (buffer_completo == true) {
59                  Continuous_Energy();
60
61                  // Rotina de 3 segundos para telemetria contínua
62                  if(event_flag_3s){
63                      xSemaphoreGive(Transmit); // Sinaliza a tarefa de rede
64                      event_flag_3s = false;
65                  }
66                  buffer_completo = false; // Reset da flag
67              }
68          }
69      }
70  }
```

Código A.2 – Código da task de aquisição, localizada no main.c.

Fonte: Elaborado pelo autor (2026).

```
//
    ================================================================================
// Tarefa 2: Gestão de Rede e Máquina de Estados MQTT (Fila e Oscilografia)
//
    ================================================================================
void conectarWifi(void *pvParameters) {
    PacoteEvento_t pacote_recebido;

    // NOVO: Variável para memorizar quem causou o problema
    char tipo_evento_ativo[32] = "";

    while (1) {
        // 1. RECEBE DA FILA
        if (xQueueReceive(fila_eventos, &pacote_recebido,
    pdMS_TO_TICKS(10)) == pdTRUE) {

            if (!evento_ativo_no_painel) {
                timestamp_inicio_real = timestamp_inicio_evento; // Usa o
    timestamp do início do evento registrado no Energy.c
                timestamp_inicio_real = pacote_recebido.timestamp;

                // NOVO: Guarda o nome do evento ativo ("Tensao" ou
    "Corrente")
                snprintf(tipo_evento_ativo, sizeof(tipo_evento_ativo),
    "%s", pacote_recebido.tipo);

                mqtt_alerta_on(timestamp_inicio_real,
    pacote_recebido.tipo);
                evento_ativo_no_painel = true;
                fim_detectado = false;
            }
            mqtt_enviar_alerta(pacote_recebido);
        }

        // 2. Detecta o fim do evento (quando ambos Voltage e Current
    voltam ao estado normal)
        if (evento_ativo_no_painel && !fim_detectado) {
            if (Voltage.state == true && Current.state == true) {
                struct timeval tv;
                gettimeofday(&tv, NULL);
                timestamp_fim_real = (int64_t)tv.tv_sec * 1000LL +
    (int64_t)tv.tv_usec / 1000LL;
                fim_detectado = true;
            }
        }
```

```
37
38          // 3. ENVIA O OFF (Agora enviando a variável tipo_evento_ativo)
39          if (evento_ativo_no_painel && fim_detectado) {
40              if (uxQueueMessagesWaiting(fila_eventos) == 0 &&
    !gravando_evento) {
41                  int64_t duracao_ms = timestamp_fim_real -
    timestamp_inicio_real;
42
43                  // NOVO: Passa o tipo para a função MQTT fechar o evento
    certo
44                  mqtt_alerta_off(timestamp_fim_real, duracao_ms,
    tipo_evento_ativo);
45
46                  evento_ativo_no_painel = false;
47              }
48          }
49
50          // 4. TRATAMENTO DA ROTINA
51          if (xSemaphoreTake(Transmit, 0) == pdTRUE) {
52              mqtt_enviar_rotina(Voltage.media, Current.media,
    Temperatura.real_value);
53          }
54      }
55  }
```

Código A.3 – Código da task de comunicação, localizada no main.c.

Fonte: Elaborado pelo autor (2026).

```
1  // --- FUN  O PRINCIPAL ---
2  void Wifi_Start()
3  {
4      // 1. Inicializa o NVS (Fundamental para a biblioteca de Wi-Fi do
    ESP32)
5      esp_err_t ret = nvs_flash_init();
6      if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret ==
    ESP_ERR_NVS_NEW_VERSION_FOUND) {
7          ESP_ERROR_CHECK(nvs_flash_erase());
8          ret = nvs_flash_init();
9      }
10     ESP_ERROR_CHECK(ret);
11
12     // 2. Criação do Grupo de Eventos e Loop de Sistema
13     s_wifi_event_group = xEventGroupCreate();
14     ESP_ERROR_CHECK(esp_netif_init());
15     ESP_ERROR_CHECK(esp_event_loop_create_default());
```

```
16      esp_netif_create_default_wifi_sta();
17
18      // 3. Inicialização e configuração do Wi-Fi
19      wifi_init_config_t wifi_cfg = WIFI_INIT_CONFIG_DEFAULT();
20      ESP_ERROR_CHECK(esp_wifi_init(&wifi_cfg));
21
22      // 4. Registro dos tratadores de eventos
23      ESP_ERROR_CHECK(esp_event_handler_register(WIFI_EVENT,
        ESP_EVENT_ANY_ID, &event_handler, NULL));
24      ESP_ERROR_CHECK(esp_event_handler_register(IP_EVENT,
        IP_EVENT_STA_GOT_IP, &event_handler, NULL));
25
26      wifi_config_t wifi_config = {
27          .sta = {
28              .ssid = WIFI_SSID,
29              .password = WIFI_PASSWORD,
30              .threshold.authmode = WIFI_AUTH_WPA2_PSK,
31          }
32      };
33
34      // 5. Início do Wi-Fi
35      ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA));
36      ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_STA, &wifi_config));
37      ESP_ERROR_CHECK(esp_wifi_start());
38      //esp_wifi_set_max_tx_power(8);// define minimo de pot ncia para 8
        dBm (aprox. 6,3 mW) para economizar energia
39
40      ESP_LOGI(TAG, "wifi_init_sta finalizado. Aguardando conexão...");
41
42      // 6. Aguarda até o Wi-Fi conectar com sucesso ou esgotar as tentativas
43      EventBits_t bits = xEventGroupWaitBits(s_wifi_event_group,
44              WIFI_CONNECTED_BIT | WIFI_FAIL_BIT,
45              pdFALSE,
46              pdFALSE,
47              portMAX_DELAY);
48
49      // 7. Lógica após tentativa de conexão (Sucesso ou Falha)
50      if (bits & WIFI_CONNECTED_BIT) {
51          ESP_LOGI(TAG, "Conectado com sucesso ao SSID:%s", WIFI_SSID);
52
53          // ============================================================
54          //                  INÍCIO DO SINCRONISMO NTP
55          // ============================================================
56          ESP_LOGI("timestamp", "Aguardando o roteador liberar a internet
            (DHCP/DNS)...");
57          vTaskDelay(3000 / portTICK_PERIOD_MS); // Dá 3 segundos cruciais
```

```
    para a rede estabilizar
58
59          ESP_LOGI("timestamp", "Inicializando SNTP com IPs fixos...");
60
61          esp_sntp_setoperatingmode(SNTP_OPMODE_POLL);
62
63          // Servidor 0: Nome de domínio padrão (Pode falhar se o DNS
    estiver lento)
64          esp_sntp_setservername(0, "pool.ntp.org");
65
66          // Servidor 1: IP BRUTO do NTP.br (Servidor oficial do Brasil -
    Não precisa de DNS)
67          esp_sntp_setservername(1, "200.160.7.186");
68
69          // Servidor 2: IP BRUTO do Google NTP (Backup de altíssima
    disponibilidade)
70          esp_sntp_setservername(2, "216.239.35.0");
71
72          sntp_set_sync_interval(3600000); // 3.600.000 ms = 1 hora de
    intervalo entre sincronizações automáticas
73
74
    //esp_sntp_set_time_sync_notification_cb(cb_notificacao_sincronismo);
75          esp_sntp_init();
76
77          setenv("TZ", "BRT3", 1);
78          tzset();
79
80          time_t now = 0;
81          struct tm timeinfo = { 0 };
82          int retry = 0;
83          const int retry_count = 15;
84
85          while (timeinfo.tm_year < (2020 - 1900) && ++retry <= retry_count)
    {
86              ESP_LOGI("timestamp", "Buscando hora na internet... (%d/%d)",
    retry, retry_count);
87              vTaskDelay(2000 / portTICK_PERIOD_MS);
88              time(&now);
89              localtime_r(&now, &timeinfo);
90          }
91
92          if (timeinfo.tm_year < (2020 - 1900)) {
93              ESP_LOGE("timestamp", "FALHA CRÍTICA: Placa conectou no Wi-Fi,
    mas sem acesso    internet ou porta UDP 123 bloqueada.");
94          } else {
```

```
95              char strftime_buf[64];
96              strftime(strftime_buf, sizeof(strftime_buf), "%d/%m/%Y -
    %H:%M:%S", &timeinfo);
97              ESP_LOGW("timestamp",
    "========================================");
98              ESP_LOGW("timestamp", "Relógio sincronizado! Tempo atual: %s",
    strftime_buf);
99              ESP_LOGW("timestamp",
    "========================================");
100         }
101         // ============================================================
102         // ============================================================
103
104     } else if (bits & WIFI_FAIL_BIT) {
105         ESP_LOGE(TAG, "Falha irreparável ao conectar no SSID:%s",
    WIFI_SSID);
106     } else {
107         ESP_LOGE(TAG, "EVENTO INESPERADO no Wi-Fi");
108     }
109 }
```

Código A.4 – Código da função $Wifi_Start$, *localizadanomain.c*.

Fonte: Elaborado pelo autor (2026).

Essa parte do código representa a parte responsável pelo monitoramento e captura de eventos.Incluem o *Monitoring.c e Energy.c*.

Arquivo: `Energy.c` – Processamento de energia e captura de eventos.

```
1  void adc_setup(void) {
2      // Configuração do ADC em modo one-shot
3      adc_oneshot_unit_init_cfg_t init_config = {
4          .unit_id = ADC_UNIT_1, // qual barramento do adc utlizado, nesse
    caso o 1
5          .ulp_mode = ADC_ULP_MODE_DISABLE, // Modo ultra low power
    desabilitado
6      };
7      adc_oneshot_new_unit(&init_config, &adc1_handle);
8
9      // Configurar canais
10     adc_oneshot_chan_cfg_t chan_cfg = { // configuração do ADC
11         .bitwidth = ADC_BITWIDTH_12, // resolução do adc
12         .atten = ADC_ATTEN_DB_12, // atenuação por ser 3.3V
```

```
    };
    adc_oneshot_config_channel(adc1_handle, ADC_CHANNEL_2, &chan_cfg); // Corrente
    adc_oneshot_config_channel(adc1_handle, ADC_CHANNEL_3, &chan_cfg); // Tensão
    adc_oneshot_config_channel(adc1_handle, ADC_CHANNEL_4, &chan_cfg); // Temperatura
    // Configura pull-downs para os pinos ADC usados
    gpio_set_pull_mode(GPIO_NUM_3, GPIO_PULLDOWN_ONLY);
    gpio_set_pull_mode(GPIO_NUM_2, GPIO_PULLDOWN_ONLY);
    gpio_set_pull_mode(GPIO_NUM_4, GPIO_PULLDOWN_ONLY);

    // Inicializa calibração
    adc_cali_curve_fitting_config_t cali_config = {
        .unit_id = ADC_UNIT_1,
        .atten = ADC_ATTEN_DB_12,
        .bitwidth = ADC_BITWIDTH_12,
    };
    adc_cali_create_scheme_curve_fitting(&cali_config, &adc_cali_handle); // cria a curva de calibração
}
```

Código A.5 – Configuração do ADC.

Fonte: Elaborado pelo autor (2026).

```
void After_Acquisition(Energy_ADC *p) {
    // Armazena as médias do primeiro nivel no segundo nivel
    p->SegundaCamada[p->index_segundo] = p->media;

    p->index_segundo++;

    // Verifica se o buffer do segundo nivel está completo
    if (p->index_segundo == SecondLevel_Length) {
        // ESP_LOGI("BUFFER2", "[%s]Buffer do segundo nivel completo.\n",p->tipo);

        // ---- Terceira Camada --------

        p->TerceiraCamada[p->index__Terceiro] = calcular_media(p->SegundaCamada, SecondLevel_Length); // buffer da terceira camada recebe a média do segundo nivel

        event_flag_3s = true; // Sinaliza que o terceiro nível foi atualizado, para verificação de eventos na tarefa principal

```

```
17          p->valor_3s = p->TerceiraCamada[p->index__Terceiro]; // Armazena o
    valor do terceiro nível para uso na tarefa principal
18
19          p-> index__Terceiro++;
20
21          p-> index_segundo = 0;
22      }
23          // Inicio da medição do Terceiro nivel, que é o buffer de 3
    segundos. Quando ele encher, será calculada a média e armazenada no
    quarto nível.
24
25      if (p->index__Terceiro == TerceiraCamada_Length) {
26
27          // ---- Quarta Camada --------
28         ESP_LOGI("BUFFER3", "[%s] Buffer do terciro nivel completo. Média:
    %.3f\n", p->tipo, calcular_media(p->TerceiraCamada,
    TerceiraCamada_Length));
29          // Quarto nivel
30              p->QuartaCamada[p->index__quarto] =
    calcular_media(p->TerceiraCamada, TerceiraCamada_Length);
31          // Incrementa os indices do Quarto nivel
32
33          p->index__quarto++;
34
35          p-> index__Terceiro = 0;
36
37          }
38
39      if (p->index__quarto == QuartaCamada_Length)    {
40
41          // --- Quinta Camada ----
42          // Calculo da média do terceiro nivel e armazenando no index do
    quarto
43          p->QuintaCamada[p->index__Quinto] =
    calcular_media(p->QuartaCamada, QuartaCamada_Length);
44          p->index__Quinto++;
45          p->index__quarto = 0;
46      }
47
48  }
```

Código A.6 – Método de agregação de camadas em regime permanente

Fonte: Elaborado pelo autor (2026).

```
1
```

```
void Instantaneous_Energy() { // primeiro nivel do buffer
    Instant_Acquisition(&Voltage);
    Instant_Acquisition(&Current);


    // 1. Início do evento: Verifica o gatilho, define o tipo UMA VEZ e começa a gravar
    if ((Voltage.state == false || Current.state == false) && !gravando_evento) {
        gravando_evento = true;
        indice_amostra = 0;

        // Define quem é o "culpado" pelo evento no momento do disparo
        if (Voltage.state == false && Current.state == false) {
            pacote_evento.tipo = "Tensao_e_Corrente"; // Opcional, caso ambos desarmem juntos
        } else if (Voltage.state == false) {
            pacote_evento.tipo = Voltage.tipo;
        } else if (Current.state == false) {
            pacote_evento.tipo = Current.tipo;
        }
        else {
            pacote_evento.tipo = "Desconhecido"; // Fallback, não deveria acontecer
        }

        struct timeval tv;
        gettimeofday(&tv, NULL);
        pacote_evento.timestamp = (int64_t)tv.tv_sec * 1000LL + (int64_t)tv.tv_usec / 1000LL;

        ESP_LOGW("GRAVADOR", "Iniciando captura de %d amostras. Motivo: %s", NUM_amostras, pacote_evento.tipo);
    }

    // 2. Se a gravação estiver ativada, salva os dados incondicionalmente
    if (gravando_evento) {
        // Grava sempre ambas as grandezas, sem checar os estados de novo e sem colocar zeros!
        pacote_evento.tensao[indice_amostra] = Voltage.instant_value;
        pacote_evento.corrente[indice_amostra] = Current.instant_value;

        indice_amostra++;

        // 3. Quando encher as posições (chegou em NUM_amostras)
        if (indice_amostra >= NUM_amostras) {
```

```
41              Enviar_Evento_Fila(&pacote_evento);
42              gravando_evento = false; // Desativa até o próximo evento
    acontecer
43              ESP_LOGW("GRAVADOR", "Captura concluída! Pacote na Fila.");
44          }
45      }
46  }
```

Código A.7 – Método de agregação de camadas inferiores e detecção de eventos

Fonte: Elaborado pelo autor (2026).

```
1
2  void Instant_Acquisition(Energy_ADC *p) {
3      // Leitura one-shot do ADC
4      adc_oneshot_read(adc1_handle, p->adc_channel, &adc_value_2);
5
6      // Calibração e conversão
7      int voltage_mv = 0;
8      adc_cali_raw_to_voltage(adc_cali_handle, adc_value_2, &voltage_mv);
9      p->instant_value = voltage_mv / 1000.0f;
10     p->instant_value = p->instant_value * p->ganho + p->offset;
11
12     // Armazena no buffer do primeiro nivel
13     p->PrimeiraCamada[p->index_primeiro] = p->instant_value;
14     p->index_primeiro = (p->index_primeiro + 1) % PrimeiraCamada_Length;
15
16     if (p->state == true) {
17         // --- ESTADO NORMAL: Monitorando limite ---
18         if (p->instant_value > p->max_threshold) {
19             p->contador_amostras++; // 1 amostra ~= 1 milissegundo
20
21             // Passo 1: Confirma se o valor ficou acima do limite pelo
    tempo exigido
22             if (p->contador_amostras >= p->tempo_confirmacao_ms) {
23
24                 // EVENTO DETECTADO! Registra o início do problema.
25                 struct timeval tv1;
26                 gettimeofday(&tv1, NULL);
27                 timestamp_inicio_evento = (int64_t)tv1.tv_sec * 1000LL +
    (int64_t)tv1.tv_usec / 1000LL;
28
29                 p->state = false; // ESTADO MUDA PARA "EM EVENTO"
30                 log_pendente = true;
31
```

```
32                  // Zera o cron metro exclusivo do MOSFET para começar a
    contar o atraso
33                  p->MOSFET_active_ms = 0;
34              }
35          }
36          else {
37              p->contador_amostras = 0; // Filtro de ruído: reseta o
    contador se a tensão cair
38          }
39      }
40
41      // --- ESTADO DE EVENTO ATIVO (p->state == false) ---
42      else {
43          // Passo 2: Contagem do atraso para acionar o MOSFET
44          // Defina o tempo que voc  quer esperar (ex: 200 ms)
45
46          if (p->MOSFET_active_ms < p->tempo_acionamento_mosfet_ms) {
47              p->MOSFET_active_ms++;
48
49              if (p->MOSFET_active_ms == p->tempo_acionamento_mosfet_ms) {
50                  // Passo 3: Tempo de atraso atingido. Aciona a proteção
    física!
51                  Positive_Set_State(true);
52                  Negative_Set_State(true);
53                  Save_MOSFET_STATE_FLASH(true, true); // Salva o estado do
    MOSFET na Flash
54              }
55          }
56
57          // Passo 4: Lógica de Histerese para desativar e voltar ao normal
58          if (p->instant_value < p->min_threshold) {
59              p->contador_amostras++; // Reaproveita o contador para validar
    a volta ao normal
60
61              // Exige que o sinal fique abaixo do limite por X
    milissegundos antes de rearmar
62              if (p->contador_amostras > 50) {
63                  p->state = true;          // VOLTA PARA "NORMAL"
64                  p->contador_amostras = 0;
65                  log_pendente = false;
66              }
67          } else {
68              p->contador_amostras = 0; // Zera se houver pico de ruído
    durante a queda
69          }
70      }
```

```
71
72      // Se o buffer do primeiro nivel está cheio, calcula a média e sinaliza
73      if (p->index_primeiro == 0) {
74          p->media = calcular_media(p->PrimeiraCamada,
    PrimeiraCamada_Length);
75          buffer_completo = true;
76      }
77  }
```

Código A.8 – Aquisição instântanea e detecção de eventos

Fonte: Elaborado pelo autor (2026).

```
1
2   Energy_ADC Voltage = {.adc_channel = ADC1_CHANNEL_3, .tipo = "Tensao",
3      .max_threshold = 30, .min_threshold = 25,  .offset = -0.01f, .ganho =
    27.30f,
4      .tempo_confirmacao_ms = 5, .contador_amostras = 0,
    .MOSFET_active_ms=0,.tempo_acionamento_mosfet_ms=50};
5   Energy_ADC Current = {.adc_channel = ADC1_CHANNEL_2, .tipo = "Corrente",
6      .max_threshold = 15, .min_threshold = 13, .offset = -0.23f, .ganho =
    6.72f,
7      .tempo_confirmacao_ms = 5, .contador_amostras =
    0,.MOSFET_active_ms=0,.tempo_acionamento_mosfet_ms=50};
8   Temp_ADC Temperatura = {.tipo = "Temperatura", .offset = 94.85f, .ganho =
    -39.73f};
```

Código A.9 – Configuração das estruturas de medição para as gradezas elétricas

Fonte: Elaborado pelo autor (2026).

Código responsável pela interupção do timer

```
1
2   void Timer_ISR(void *param) {
3
4       int64_t tempo_inicio = esp_timer_get_time();
5       Instantaneous_Energy();     // Executa aquisição instantânea de energia
6
7       int64_t tempo_fim = esp_timer_get_time();
8       int64_t duracao = tempo_fim - tempo_inicio;
9
10      // 3. Salva apenas o pior caso observado
11      if (duracao > pior_tempo_execucao_adc) {
12          pior_tempo_execucao_adc = duracao;
13      }
```

```
14      // Libera o semáforo para sinalizar a task principal
15      if (timer_semaphore != NULL) {
16          BaseType_t xHigherPriorityTaskWoken = pdFALSE;
17          xSemaphoreGiveFromISR(timer_semaphore, &xHigherPriorityTaskWoken);
    // Libera semáforo dentro da ISR
18          portYIELD_FROM_ISR(xHigherPriorityTaskWoken); // Troca de contexto
    se necessário
19      }
20  }
21
22
23  void iniciar_timer(SemaphoreHandle_t semaphore) {
24      timer_semaphore = semaphore; // Associa o semáforo recebido
    variável global
25      // Configuração dos parâmetros do timer
26      const esp_timer_create_args_t my_timer_args = {
27          .callback = &Timer_ISR,                // Função chamada a cada
    disparo do timer
28          .name = "Meu Timer",                   // Nome do timer (para
    debug)
29          .dispatch_method = ESP_TIMER_TASK      // Método de despacho
    (executa em task dedicada)
30      };
31
32      esp_timer_handle_t timer_handler;          // Handler para o timer
33
34      ESP_ERROR_CHECK(esp_timer_create(&my_timer_args, &timer_handler)); //
    Cria o timer
35      ESP_ERROR_CHECK(esp_timer_start_periodic(timer_handler, 1041));    //
    Inicia timer  (aprox. 1ms para 960Hz)
36  }
```

Código A.10 – Configuração do timer para a ISR para a aquisição instantâna

Fonte: Elaborado pelo autor (2026).

Apêndice B – Código em MATLAB para caracterização dos circuitos dos sensores

```
1
2 %% Gráfico para saber o ganho do sensor de tensão
3  x = [0.002,0.184,0.367,0.549,0.733,0.915,1.026,1.099,1.206];
4  y = [0.048,5,10,15,20,25,28,30,32.89];
5
6
7 % 2. Calcula a regressão linear (polin mio de grau 1)
8 p = polyfit(x, y, 1);
9 y_ajustado = polyval(p, x);
10
11 % 3. Cria o gráfico
12 figure; % Abre uma nova janela
13 plot(x, y, 'ro', 'MarkerFaceColor', 'r'); % Pontos originais
14 hold on;                                  % Mantém os pontos no gráfico
15 plot(x, y_ajustado, 'b-', 'LineWidth', 2); % Desenha a reta de ajuste
16
17 % 4. Prepara o texto da equação
18 % p(1) é a inclinação (a) e p(2) é o intercepto (b)
19 txt_equacao = sprintf('y = %.2fx + %.2f', p(1), p(2));
20
21 % 5. Apresenta a equação no gráfico (duas formas):
22
23 % Forma A: No meio do gráfico usando a função text(x, y, string)
24 % Usamos as coordenadas min(x) e max(y) para posicionar no canto superior
       esquerdo
25 text(min(x), max(y), txt_equacao, 'FontSize', 12, 'FontWeight', 'bold',
       'Color', 'blue');
26
27 % Forma B: Na legenda (mais organizado)
28 legend('Dados Experimentais', txt_equacao, 'Location', 'northwest');
29
30 % Customização final
31  xlabel ('Tensão de saída do circuito de condicionamento de tensão(V)');
32  ylabel('Tensão de entrada no Sensor LV25-P(V)');
33  title('Caracterização do circuito de condicionamento de tensão')
34 grid on;
```

Código B.1 – Código em MATLAB para gerar a curva de resposta do sensor de tensão.

Fonte: Elaborado pelo autor (2026).

```

%% Gráfico para saber o ganho do sensor de corrente
 x = [0.026,0.352,0.780,1.082,1.532,1.840,2.204,2.750,3.000];
 y = [0,2,5,7,10,12,15,18,20];


% 2. Calcula a regressão linear (polin mio de grau 1)
p = polyfit(x, y, 1);
y_ajustado = polyval(p, x);

% 3. Cria o gráfico
figure; % Abre uma nova janela
plot(x, y, 'ro', 'MarkerFaceColor', 'r'); % Pontos originais
hold on;                                  % Mantém os pontos no gráfico
plot(x, y_ajustado, 'b-', 'LineWidth', 2); % Desenha a reta de ajuste

% 4. Prepara o texto da equação
% p(1) é a inclinação (a) e p(2) é o intercepto (b)
txt_equacao = sprintf('y = %.2fx + %.2f', p(1), p(2));

% 5. Apresenta a equação no gráfico (duas formas):

% Forma A: No meio do gráfico usando a função text(x, y, string)
% Usamos as coordenadas min(x) e max(y) para posicionar no canto superior
    esquerdo
text(min(x), max(y), txt_equacao, 'FontSize', 12, 'FontWeight', 'bold',
    'Color', 'blue');

% Forma B: Na legenda (mais organizado)
legend('Dados Experimentais', txt_equacao, 'Location', 'northwest');

% Customização final
 xlabel ('Tensão de saída do circuito de condicionamento de corrente(V)');
 ylabel('Tensão de entrada no Sensor LA55-P(V)');
 title('Caracterização do circuito de condicionamento de corrente')
grid on;
```

Código B.2 – Código em MATLAB para gerar a curva de resposta do sensor de corrente.

Fonte: Elaborado pelo autor (2026).

```
%% Gráfico para saber o ganho do sensor de temperatura
 x = [1.8,1.52,1.44,1.24,1.11,1.03,0.95,0.86,0.80,0.64];
```

```
 y = [23.7,32.4,40,45.2,50.7,53.8,56.5,60.3,63.3,70];


% 2. Calcula a regressão linear (polin mio de grau 1)
p = polyfit(x, y, 1);
y_ajustado = polyval(p, x);

% 3. Cria o gráfico
figure; % Abre uma nova janela
plot(x, y, 'ro', 'MarkerFaceColor', 'r'); % Pontos originais
hold on;                                  % Mantém os pontos no gráfico
plot(x, y_ajustado, 'b-', 'LineWidth', 2); % Desenha a reta de ajuste

% 4. Prepara o texto da equação
% p(1) é a inclinação (a) e p(2) é o intercepto (b)
txt_equacao = sprintf('y = %.2fx + %.2f', p(1), p(2));

% 5. Apresenta a equação no gráfico (duas formas):

% Forma A: No meio do gráfico usando a função text(x, y, string)
% Usamos as coordenadas min(x) e max(y) para posicionar no canto superior
    esquerdo
text(min(x), max(y), txt_equacao, 'FontSize', 12, 'FontWeight', 'bold',
    'Color', 'blue');

% Forma B: Na legenda (mais organizado)
legend('Dados Experimentais', txt_equacao, 'Location', 'northwest');

% Customização final
 xlabel ('Tensão de saída no canal  ADC de temperatura( C )');
 ylabel('Temperatura no multímetro ( C )');
 title('Caracterização do circuito de condicionamento de temperatura')
grid on;
```

Código B.3 – Código em MATLAB para gerar a curva de resposta do sensor de temperatura.

Fonte: Elaborado pelo autor (2026).


UnB