UNIVERSIDADE FEDERAL DO PAMPA

Pedro Ramires da Silva Amalfi Costa

TULKAS: Desenvolvimento de metodologias e estudo para implementação de segurança em sistemas IoT

Alegrete
2025

Pedro Ramires da Silva Amalfi Costa

TULKAS: Desenvolvimento de metodologias e estudo para implementação de segurança em sistemas IoT

Dissertação apresentada ao Programa de Pós-Graduação Stricto Sensu em Engenharia de Software da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Mestre em Engenharia de Software.

Orientador: Prof. Dr. Claudio Schepke

Alegrete
2025

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

C837t Costa, Pedro Ramires da Silva Amalfi
TULKAS: Desenvolvimento de metodologias e estudo para implementação de segurança em sistemas IoT / Pedro Ramires da Silva Amalfi Costa.
96 p.

Dissertação(Mestrado)-- Universidade Federal do Pampa, MESTRADO EM ENGENHARIA DE SOFTWARE, 2025.
"Orientação: Claudio Schepke".

1. IoT. 2. Segurança da Informação. 3. STRIDE. 4. DREAD. 5. MQTT. I. Título.

PEDRO RAMIRES DA SILVA AMALFI COSTA

TULKAS: DESENVOLVIMENTO DE METODOLOGIAS E ESTUDO PARA IMPLEMENTAÇÃO DE SEGURANÇA EM SISTEMAS IOT

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Software da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Mestre em Engenharia de Software.

Dissertação defendida e aprovada em: 30/06/2025

Banca examinadora:

_____

Prof. Dr. Claudio Schepke

Orientador

(Unipampa)

_____

Prof. Dr. Rodrigo Brandão Mansilha

(Unipampa)

_____

Prof. Dr. Adriano José Vogel

(Dynatrace LLC e Sociedade Educacional Três de Maio - Setrem)

Assinado eletronicamente por **CLAUDIO SCHEPKE**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 30/06/2025, às 17:48, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **RODRIGO BRANDAO MANSILHA**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 30/06/2025, às 18:07, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **Adriano Vogel**, **Usuário Externo**, em 01/07/2025, às 13:35, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1770800** e o código CRC **C5926226**.

RESUMO

Esta dissertação apresenta a modelagem de ameaças e a análise de segurança de um subsistema de irrigação inteligente baseado em IoT, com ênfase nos dispositivos embarcados ESP32, sensores conectados e na comunicação entre esses dispositivos e o servidor de borda (Orange Pi). Projetado para a empresa Tulkas Desenvolvimento, o projeto tem como foco a realidade de pequenos produtores rurais, especialmente em regiões com conectividade limitada e com usuários de baixa familiaridade tecnológica. As metodologias STRIDE e DREAD foram aplicadas especificamente à camada sensorial e de comunicação segura via MQTT sobre TLS, com o objetivo de identificar vulnerabilidades críticas e realizar testes de segurança direcionados. Os resultados obtidos permitiram propor contramedidas viáveis para mitigar os riscos identificados, como criptografia de firmware, autenticação mútua e proteção física contra adulterações. A dissertação também discute a arquitetura parcial do sistema, decisões técnicas envolvidas na escolha dos componentes e os desafios enfrentados ao integrar mecanismos de segurança em um cenário rural. Como trabalho futuro, propõe-se a expansão da análise de ameaças para todo o sistema, incluindo a camada de nuvem e a interface de controle do usuário.

Palavras-chave: IoT. Segurança da Informação. STRIDE. DREAD. MQTT. Testes de Invasão.

ABSTRACT

This dissertation presents the threat modeling and security analysis of a small irrigation subsystem based on IoT, focusing on the embedded ESP32 device, connected sensors, and their communication with the edge server (Orange Pi). Developed by the company Tulkas Desenvolvimento, the solution targets small rural producers, particularly in areas with limited connectivity and low technological familiarity. We apply STRIDE and DREAD methodologies specifically to the sensing layer and secure communication via MQTT over TLS to identify critical vulnerabilities and conduct targeted security testing. Based on the results, feasible mitigation strategies were proposed, such as firmware encryption, mutual authentication, and physical protection of the devices. The work also discusses the partial system architecture, hardware and software design decisions, and the challenges of implementing security in rural environments. As in future work, we propose to extend the threat modeling to the entire system, including the cloud infrastructure and user interface.

Key-words: IoT. Information Security. STRIDE. DREAD. MQTT. Penetration Testing.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A agricultura é essencial para a subsistência humana. Com o avanço da tecnologia, surgem soluções capazes de automatizar tarefas no campo, consolidando o conceito de agricultura 4.0 — em que sensores, conectividade e inteligência computacional são integrados para otimizar os processos produtivos (ABBASI; MARTINEZ; AHMAD, 2022). A irrigação, elemento crítico na produção agrícola, pode ser significativamente aprimorada por meio de sistemas inteligentes, promovendo o uso eficiente da água e reduzindo desperdícios (BOURSIANIS et al., 2021).

A **Tulkas Desenvolvimento**, atuante no nordeste de São Paulo e sul de Minas Gerais, foi procurada por um cliente com dificuldades na irrigação hidropônica de sua propriedade rural, enfrentando perdas recorrentes causadas por quedas de energia e falhas no abastecimento de água. Como resposta, a empresa desenvolveu um sistema de irrigação inteligente baseado em IoT, com foco em confiabilidade, automação e acessibilidade a usuários com pouca familiaridade tecnológica (PATIL et al., 2016).

Embora o sistema tenha sido projetado para atuar em cenários rurais com infraestrutura limitada, a coleta e transmissão de dados sensoriais por dispositivos conectados introduz riscos importantes de segurança. A interceptação ou adulteração desses dados pode comprometer decisões automatizadas, levando a falhas de irrigação, desperdício de recursos ou até mesmo prejuízos à produção agrícola. Isso evidencia a importância de adicionar camadas de segurança à arquitetura, especialmente nos pontos mais vulneráveis do sistema, como os sensores e a comunicação com o servidor de borda.

Dessa forma, esta dissertação tem como objetivo identificar e mitigar vulnerabilidades de segurança no subsistema composto por dispositivos embarcados ESP32, sensores e a comunicação com Orange Pi via MQTT com TLS. O foco está na aplicação prática de metodologias de *Threat Modeling*, avaliadas no Capítulo 3, e posteriormente aplicadas ao sistema real. A análise concentra-se em ameaças relacionadas à interceptação de pacotes, adulteração de dados, ataques físicos e falhas de autenticação, propondo contramedidas viáveis para o contexto rural.

Além disso, este trabalho discute os desafios técnicos enfrentados na integração da segurança em um sistema de baixo custo e operante em ambiente não confiável. Embora não tenha sido possível mensurar o impacto das contramedidas em termos de desempenho, consumo energético ou tráfego de rede, esse aspecto é apontado como recomendação para trabalhos futuros.

Esta dissertação está estruturada da seguinte forma:

- **Capítulo 2 - Introdução ao Problema e Cenário Proposto**: descreve o contexto da agricultura inteligente, detalha a motivação do cliente, os objetivos do sistema, seus principais componentes físicos e lógicos, e apresenta uma visão geral da arquitetura proposta e dos riscos de segurança envolvidos.

- **Capítulo 3 - Revisão de Metodologias de Modelagem de Ameaças**: analisa criticamente diferentes metodologias de *Threat Modeling*, como STRIDE, DREAD, PASTA, OWASP IoT Top Ten, LINDDUN, entre outras, com o objetivo de selecionar aquelas mais adequadas ao cenário de sensores IoT com comunicação remota.
- **Capítulo 4 - Aplicação da Metodologia Selecionada**: documenta a aplicação prática das metodologias STRIDE e DREAD ao subsistema analisado (ESP32 e sensores), identificando ameaças e vulnerabilidades reais e propondo medidas de mitigação.
- **Capítulo 5 - Testes e Validação**: apresenta os testes de segurança realizados, incluindo simulações de ataques como interceptação de pacotes MQTT, adulteração de leituras sensoriais e análise de *firmware*, avaliando a eficácia das medidas adotadas.
- **Capítulo 6 - Conclusão e Recomendações**: sintetiza os resultados obtidos, destaca as contribuições da pesquisa e propõe melhorias futuras, incluindo a expansão da análise para o sistema completo e a avaliação quantitativa do impacto das técnicas de segurança.

Ao delimitar o escopo da análise de segurança a uma parte crítica da arquitetura — a coleta e transmissão de dados sensoriais em ambientes expostos —, esta dissertação contribui de forma prática e aplicável para a segurança em projetos de agricultura 4.0. Além disso, oferece uma base teórica e metodológica que pode orientar desenvolvedores e pesquisadores na adoção de boas práticas de proteção em sistemas IoT agrícolas.

2 INTRODUÇÃO AO PROBLEMA E CENÁRIO PROPOSTO

Este capítulo apresenta os componentes que integram a arquitetura do sistema desenvolvido para automatizar o processo de irrigação e auxiliar o produtor rural no monitoramento de suas culturas, sejam elas hidropônicas ou tradicionais. O sistema foi projetado para responder a uma demanda prática observada em propriedades rurais: a necessidade de garantir um controle confiável e contínuo da irrigação, mesmo em cenários com infraestrutura precária de energia ou conectividade.

Seu objetivo principal é mitigar riscos comuns enfrentados por produtores, como interrupções no fornecimento de água, falhas causadas por quedas de energia, excesso ou escassez de irrigação, além da ausência de informações em tempo real sobre as condições ambientais e o funcionamento do sistema. Dessa forma, a solução busca oferecer uma alternativa de baixo custo e fácil operação que permita ao agricultor manter a produtividade, reduzir desperdícios e prevenir perdas, por meio do uso de sensores IoT, automação e comunicação remota.

A arquitetura proposta foi pensada para operar em três camadas: dispositivos de campo (sensores e atuadores baseados em ESP32), um servidor de borda (utilizando uma Orange Pi) responsável pelo armazenamento e gerenciamento local dos dados, e uma camada em nuvem para visualização remota e alertas. Vale destacar que a implementação descrita nesta dissertação foca, principalmente, na camada de dispositivos IoT e na comunicação segura com o servidor de borda, onde foram aplicadas as técnicas de modelagem de ameaças e segurança.

2.1 Funcionamento Geral da Aplicação

Esta seção apresenta uma visão funcional simplificada do sistema, baseada nas interações que o usuário realiza com os recursos disponibilizados.

Conforme ilustrado na Figura 1, o usuário inicia a rotina de trabalho ao abrir a aplicação. Nesse momento, ele pode consultar alertas gerados por eventos detectados pelos sensores, visualizar e alterar a rotina de irrigação programada para a motobomba, além de acompanhar em tempo real os dados e estados dos sensores conectados. A aplicação foi projetada para ser flexível e adaptável à realidade de cada propriedade, não impondo um número fixo ou tipos específicos de sensores, o que permite personalização conforme as culturas monitoradas e os recursos do produtor.

Ao selecionar a opção “Conferir os Alerta”, o sistema executa automaticamente uma verificação periódica para identificar eventos críticos registrados na base de dados. Conforme mostrado na Figura 2, essa verificação depende da disponibilidade de conexão com a internet. Caso haja conectividade, os dados são buscados na nuvem; caso contrário, a aplicação acessa o servidor de borda para obter as informações.

Na ausência de conexão com a internet, um novo alerta é gerado automaticamente, informando o usuário sobre a indisponibilidade da rede. Essa funcionalidade é funda-

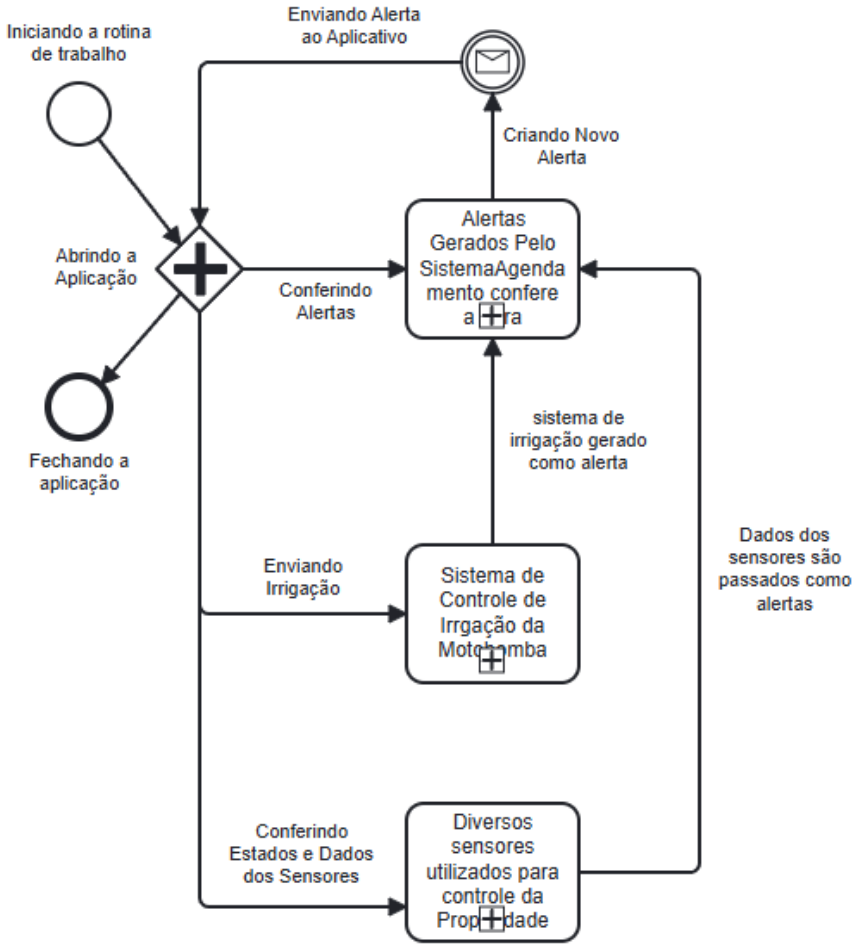

Figura 1 – Diagrama Principal da Aplicação

mental para evitar que falhas críticas passem despercebidas, especialmente em culturas sensíveis, onde a ausência de irrigação pode causar perdas irreversíveis.

Destaca-se ainda a existência de um processo automático (representado como um "script" na Figura 2) responsável por sincronizar periodicamente o banco de dados local do servidor de borda com a base de dados na nuvem, garantindo a consistência das informações e minimizando os impactos de falhas de comunicação temporárias.

Outra funcionalidade disponível é o agendamento manual da rotina de irrigação. Como ilustrado na Figura 3, o usuário define os horários e a duração em que a motobomba deve operar, programando o sistema conforme as necessidades específicas da sua cultura. Após o agendamento, o sistema verifica novamente a conectividade para decidir se a instrução será enviada diretamente à nuvem ou armazenada localmente no servidor de borda até que a conexão seja restabelecida.

A motobomba executa a rotina programada até que receba uma nova instrução. Como visto na Figura 4, essas instruções são armazenadas localmente, e a motobomba realiza testes periódicos para detectar falhas de comunicação, interrupções de energia, obs-

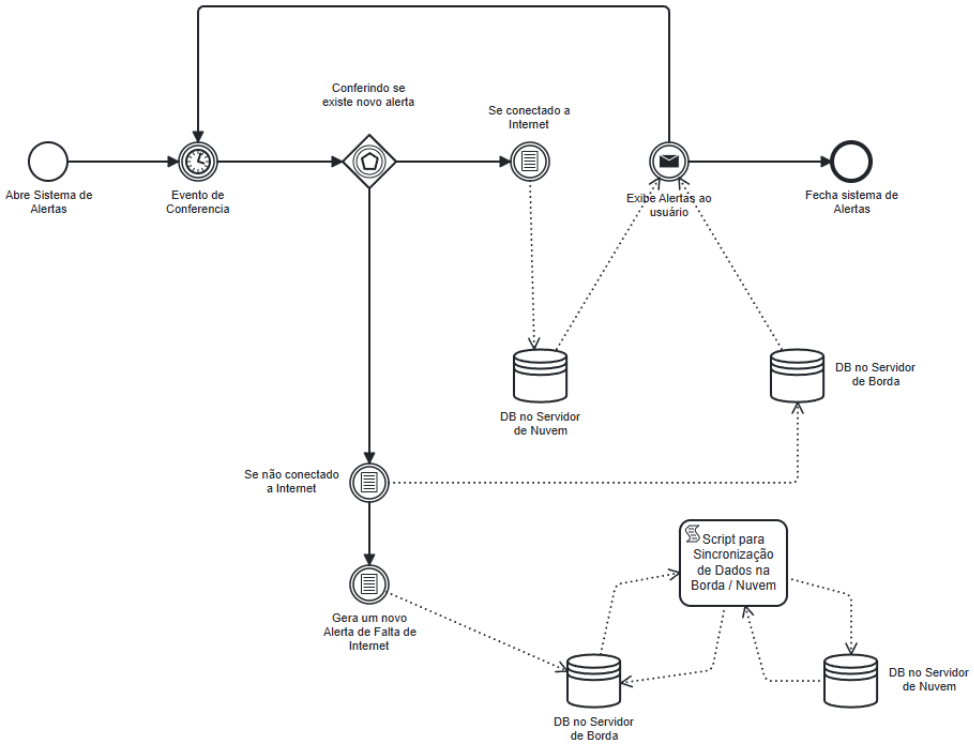

Figura 2 – Conferindo Alertas

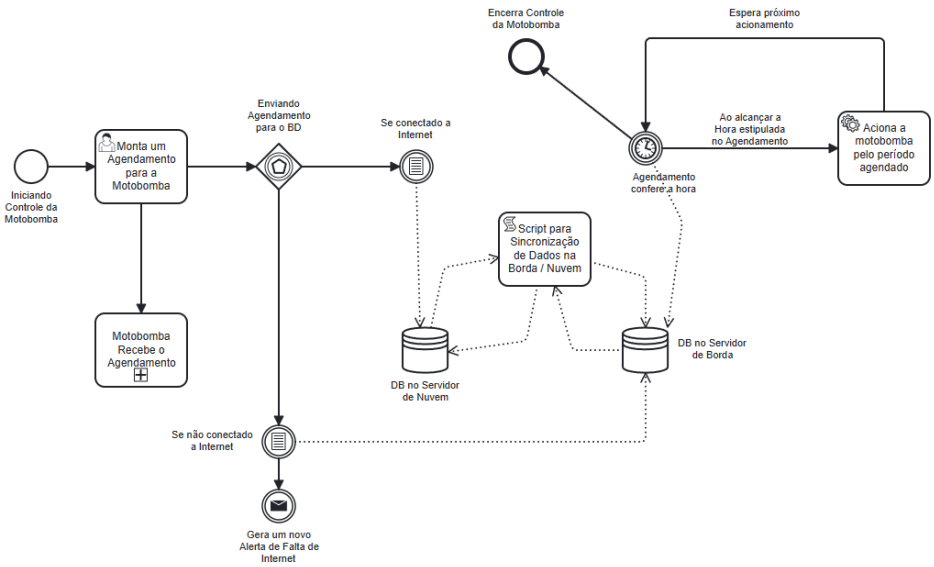

Figura 3 – Controle de Irrigação

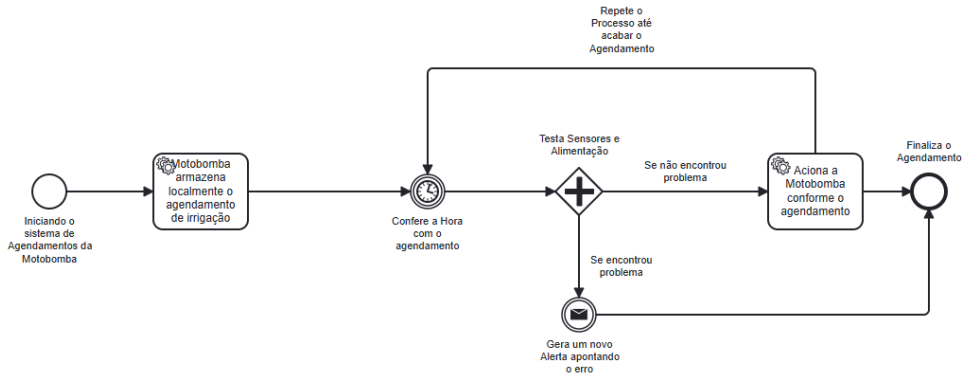

Figura 4 – Motobomba

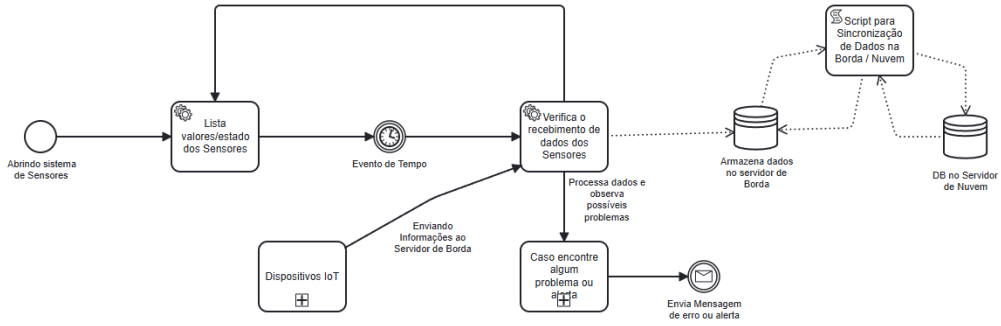

Figura 5 – Sensores

truções nas tubulações ou qualquer anomalia que possa comprometer a irrigação. Quando uma situação de risco é identificada, o sistema gera um novo alerta e o envia ao servidor de borda.

A última funcionalidade principal é a visualização do estado dos sensores conectados. Como apresentado na Figura 5, o sistema exibe os dados coletados em tempo quase real, permitindo ao usuário monitorar variáveis como umidade do solo, temperatura ambiente, pressão da água, entre outros. Esse monitoramento ocorre de forma periódica para garantir atualizações constantes.

Caso alguma leitura anormal seja detectada — como ausência de resposta de um sensor ou valores fora do esperado — o sistema gera um alerta automático para informar o usuário. As Figuras 6 e 7 ilustram esse processo de verificação contínua e resposta a falhas.

Esta explicação inicial tem como objetivo facilitar a compreensão do papel de cada componente apresentado nas seções seguintes, reforçando sua importância no funcionamento do sistema e destacando os pontos mais críticos do ponto de vista da segurança da informação.

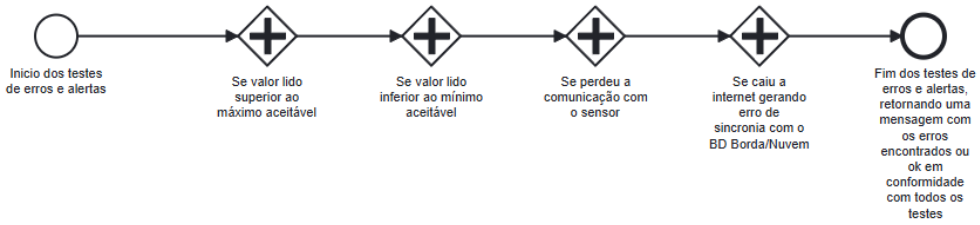

Figura 6 – Problemas

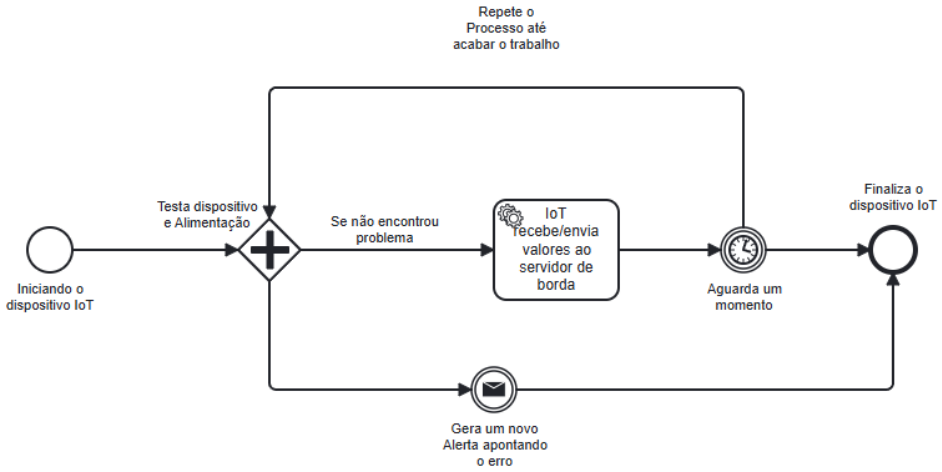

Figura 7 – Dispositivos IoT

2.2 Componentes do Sistema de Irrigação

Os componentes descritos nesta seção correspondem aos elementos efetivamente utilizados na implementação do sistema de irrigação automatizado desenvolvido para este trabalho. Trata-se de uma arquitetura real, implantada em ambiente de testes com o objetivo de automatizar a irrigação, monitorar variáveis ambientais e facilitar a gestão por parte do produtor rural, mesmo em locais com infraestrutura limitada.

Cada componente apresentado a seguir cumpre uma função específica dentro da arquitetura, sendo parte de um sistema integrado que contempla sensores IoT, dispositivos embarcados, servidor de borda, módulos de alimentação e interface de visualização. A interação entre esses elementos foi planejada para garantir o funcionamento contínuo, mesmo em cenários adversos, como instabilidade de rede ou quedas de energia.

Vale destacar, no entanto, que embora esta seção documente todos os elementos que compõem o sistema como um todo, a análise de segurança proposta nesta dissertação está restrita à camada de coleta de dados e envio para o servidor de borda. Mais especificamente, as técnicas de modelagem de ameaças (STRIDE e DREAD) foram aplicadas apenas ao ESP32, aos sensores conectados a ele e à comunicação com o servidor de borda (Orange Pi), utilizando o protocolo MQTT sobre TLS.

Esse recorte foi definido com base na criticidade dessa etapa do sistema, pois

é justamente nesse ponto que ocorrem a coleta, o processamento e o envio dos dados sensoriais, que são vitais para a tomada de decisão na irrigação. Futuramente, recomenda-se a expansão da análise de segurança para os demais módulos, como a interface de usuário e a nuvem.

A seguir, são detalhados os principais componentes utilizados na implementação do sistema.

2.2.1 Servidor de Nuvem

O **Servidor de Nuvem** é uma infraestrutura de computação remota que fornece serviços e recursos acessíveis pela internet. Ele permite armazenar, processar e gerenciar dados de forma distribuída, permitindo que aplicativos e dispositivos IoT acessem e compartilhem informações de maneira eficiente e escalável. (TANENBAUM; STEEN, 2024; ABBASI; MARTINEZ; AHMAD, 2022) No trabalho, foram utilizados os recursos de Docker e Django para a instanciação do servidor.

Docker é uma plataforma de software que permite a criação, o teste e a implantação de aplicativos em contêineres. Os contêineres permitem que um desenvolvedor empacote uma aplicação com todas as partes necessárias, como bibliotecas e outras dependências, e a distribua como uma única unidade. (Docker Inc., 2024) Isso garante que a aplicação seja executada de maneira consistente em qualquer ambiente.

Django é um framework de desenvolvimento web escrito em Python, projetado para promover a criação rápida, organizada e reutilizável de aplicações. Seguindo o princípio do "don't repeat yourself" (D.R.Y.), ele incentiva a reutilização de componentes e a manutenção de um código limpo e eficiente. Django é amplamente adotado na construção de aplicações web escaláveis e robustas. (FOUNDATION, 2024).

No contexto deste projeto, Django é utilizado tanto no servidor de nuvem quanto no servidor de borda, permitindo o desenvolvimento de interfaces web e a implementação da lógica de negócios necessária para a interação com dispositivos IoT e o processamento de dados local. Sua estrutura modular e suporte a múltiplos bancos de dados o tornam uma escolha adequada para ambientes distribuídos e aplicações embarcadas.

Embora o servidor de nuvem faça parte da infraestrutura completa do sistema, sua superfície de ataque não foi considerada no escopo desta dissertação. A segurança da comunicação entre os dispositivos IoT e a nuvem, bem como a proteção do banco de dados PostgreSQL e da aplicação web, ficam como direções para trabalhos futuros.

O **Servidor de Borda** é um dispositivo localizado próximo aos dispositivos IoT, responsável por intermediar a comunicação entre esses dispositivos e o Servidor de Nuvem. Ele ajuda a reduzir a latência, o tráfego de rede e aumenta a eficiência na gestão dos dados coletados pelos dispositivos. Este servidor de borda é composto por:

- **Orange Pi 3B RK3566**: O Orange Pi 3B RK3566 é um microcomputador com-

pacto baseado no processador Rockchip RK3566, que combina eficiência energética com desempenho adequado para aplicações embarcadas. Ele conta com conectividade Wi-Fi e Bluetooth integradas, múltiplas portas USB, saída HDMI e suporte para eMMC, cartão microSD e interfaces GPIO, facilitando a integração com sensores e atuadores. É compatível com diversos sistemas operacionais, como Debian, Ubuntu e Android, sendo uma opção versátil e acessível para soluções em IoT, automação residencial e servidores de borda. (Shenzhen Xunlong Software CO., Limited, 2025)

- **Armazenamento SSD KingSpec 256GB M.2 NVMe 2280 PCIe**: O SSD KingSpec 256GB M.2 NVMe 2280 PCIe é uma unidade de estado sólido de alto desempenho, ideal para armazenamento local em dispositivos de borda e sistemas embarcados. Utiliza o protocolo PCIe para transferência rápida de dados e o formato M.2 2280, que oferece fácil integração com placas-mãe compatíveis. A tecnologia NVMe garante baixíssima latência e alta taxa de leitura/gravação. O formato de partição pode ser configurado conforme o sistema operacional utilizado, e é possível aplicar criptografia para garantir a segurança dos dados armazenados. (KingSpec, 2025).

- **Linux - Orange Pi OS (Arch)**: O Linux é um sistema operacional de código aberto amplamente utilizado em servidores de borda devido à sua estabilidade e flexibilidade. O Orange Pi OS, baseado no Arch Linux, é uma distribuição otimizada para dispositivos Orange Pi, oferecendo suporte a hardware específico e uma ampla gama de pacotes de software. (Shenzhen Xunlong Software Co., Ltd., 2025).

- **Cliente MQTT - Eclipse Paho**: Um Cliente MQTT é uma aplicação ou dispositivo que se conecta a um broker MQTT para enviar e receber mensagens. O Eclipse Paho é uma implementação de código aberto do protocolo MQTT em várias linguagens de programação, fornecendo uma API fácil de usar para desenvolver clientes MQTT em dispositivos IoT. (Amazon Web Services, 2025).

2.2.2 Dispositivo IoT

Os dispositivos IoT (Internet das Coisas) são objetos físicos que estão conectados à internet e têm a capacidade de coletar e transmitir dados. Eles são fundamentais para a coleta de informações em tempo real e o monitoramento de sistemas físicos. No trabalho, foram usados ESP32 e Arduino.

ESP32 é um microcontrolador de alto desempenho com conectividade integrada Wi-Fi e Bluetooth, desenvolvido especialmente para aplicações de Internet das Coisas (IoT). Ele se destaca por seu baixo consumo de energia, múltiplos núcleos de processamento, pinos de entrada/saída versáteis e suporte a diversos protocolos de comunicação,

como SPI, I2C, UART e PWM. Além disso, possui recursos integrados como ADC, DAC, sensores de toque e temporizadores. Graças à sua versatilidade, baixo custo e ampla comunidade de suporte, o ESP32 é amplamente adotado em projetos de automação, monitoramento remoto e sistemas embarcados. (Espressif Systems, 2025).

Arduino é uma plataforma de prototipagem eletrônica de código aberto que combina hardware e software de fácil utilização. Ela disponibiliza uma ampla gama de placas com microcontroladores, além de um ambiente de desenvolvimento integrado (IDE) que simplifica a programação e a prototipagem de sistemas embarcados.

Neste projeto, está sendo utilizado o microcontrolador ESP32 S3 N16R8, que, embora não seja uma placa Arduino tradicional, é totalmente compatível com a IDE Arduino e suas bibliotecas. Essa escolha permite aproveitar os recursos avançados do ESP32, como Wi-Fi e Bluetooth integrados, mantendo a facilidade de programação e o suporte da comunidade proporcionados pela plataforma Arduino. (Arduino, 2025).

2.2.3 Infraestrutura Física de Proteção e Suporte

A confiabilidade de um sistema IoT implantado em ambientes externos vai além da robustez dos componentes eletrônicos e da lógica embarcada. Elementos como proteção contra intempéries, fixação estável e organização física adequada são essenciais para garantir o funcionamento contínuo, seguro e eficiente do sistema ao longo do tempo.

Nesta subseção, são apresentados os principais componentes físicos utilizados na estrutura do sistema proposto, com foco em itens de proteção ambiental (como caixas herméticas e anéis de vedação), mecanismos de fixação (estruturas metálicas e suportes ajustáveis), e organização térmica e física do servidor de borda. Cada escolha foi fundamentada com base em critérios como grau de proteção (IP67/IP68), resistência a corrosão, facilidade de manutenção e compatibilidade com os módulos eletrônicos embarcados.

Além de descrever a função de cada item, esta seção também apresenta suas respectivas fontes técnicas para comprovação das especificações adotadas.

A **Caixa Hermética** é construída em ABS ou Policarbonato, IP67/IP68 e tem como finalidade proteção contra intempéries (chuva, poeira, umidade) e entrada de insetos; isolamento dos circuitos eletrônicos. No contexto deste projeto, é utilizada para os módulos iot (sensor e atuador), garantindo confiabilidade, proteção ambiental e maior durabilidade do sistema. (Hummel do Brasil, 2025)

A **Vedação e Passagem de Cabos** é construída em anéis de vedação em borracha (neoprene ou silicone) e tem como finalidade garantir estanqueidade nos pontos de entrada de cabos e conectores, mantendo padrão IP67/IP68. No contexto deste projeto, é utilizada para os módulos iot (sensor e atuador), garantindo confiabilidade, proteção ambiental e maior durabilidade do sistema. (HellermannTyton, 2025)

O **Suporte do Painel Solar** é construída em estrutura metálica com inclinação ajustável e tem como finalidade fixação estável do painel solar, otimizando captação de

energia e resistência às intempéries. No contexto deste projeto, é utilizada para os módulos iot (sensor e atuador), garantindo confiabilidade, proteção ambiental e maior durabilidade do sistema. (Renovigi Energia Solar, 2025)

A **Estrutura de Fixação do Gabinete** é construída em estrutura metálica com tratamento anticorrosão e tem como finalidade suporte de fixação para a caixa hermética, passagem de cabos e suporte do painel solar. No contexto deste projeto, é utilizada para os módulos iot (sensor e atuador), garantindo confiabilidade, proteção ambiental e maior durabilidade do sistema. (Cemar, 2025)

A **Gabinete do Servidor de Borda** é construída em caixa acrílica ventilada e tem como finalidade proteção física contra impactos; facilita organização interna e dissipação térmica. No contexto deste projeto, é utilizada para os servidor de borda, garantindo confiabilidade, proteção ambiental e maior durabilidade do sistema. (Treedix, 2025)

2.2.4 Sensores e Componentes de Energia

Os seguintes sensores e componentes de energia foram utilizados no desenvolvimento da solução proposta:

O **No-Break (UPS) 600VA** é utilizado para garantir o fornecimento contínuo de energia ao servidor de borda durante quedas ou oscilações na rede elétrica. Com capacidade para manter a alimentação de dispositivos por alguns minutos, permite o desligamento seguro do sistema ou a continuidade de operações críticas em curtos períodos de interrupção. Também oferece proteção contra surtos e ruído elétrico, aumentando a confiabilidade da infraestrutura local. (SMS Tecnologia Eletrônica, 2025)

O **Módulo Fonte AC/DC 5V 3A** é responsável pela conversão da corrente alternada da rede elétrica em corrente contínua estável de 5V, com capacidade de fornecimento de até 3 amperes. Esse componente é utilizado para alimentar diretamente o *servidor de borda*, garantindo energia contínua e suficiente para o funcionamento dos seus sistemas embarcados e processos de coleta e análise de dados locais. (RS Components, 2025).

As **Baterias de Li-Ion 18650 - Vtc6 3000mah 30a** recarregáveis fornecem energia ao sistema em conjunto com o painel solar. As baterias são ideais por sua alta densidade energética e compatibilidade com módulos de carregamento Li-Ion. (Murata Manufacturing Co., Ltd., 2025).

Os **Painéis Solares Fotovoltaicos** são utilizados como fonte de energia alternativa para alimentar o sistema em ambientes remotos. Os painéis convertem energia solar em eletricidade, carregando as baterias 18650 através de um controlador de carga. (ALLOYSEED, 2025).

O **Módulo de Alimentação com suporte para a Bateria de lítio 18650 para ESP32** é utilizado neste projeto como uma solução prática e eficiente para fornecer energia ao microcontrolador ESP32 em ambientes sem acesso à rede elétrica. Este

módulo, frequentemente chamado de *shield* de alimentação, permite a conexão direta de uma célula 18650 e possui circuito de carregamento integrado com entrada Micro USB, USB ou USB-C, também possui proteçãoo contra sobrecarga/descarga de alimentação. Neste projeto será utilizado para receber energia via painel solar e alimentar o ESP32 e sensores. (diymore, 2025).

O módulo fornece saídas reguladas de 5V/3A, 3.3V e 3V/1A, adequadas para alimentar o ESP32 e seus periféricos. Além disso, conta com porta USB Tipo-A, permitindo seu uso como banco de potência (*power bank*) para outros dispositivos. Essa configuração garante autonomia energética ao sistema, sendo ideal para aplicações em campo, como monitoramento ambiental e automação agrícola.

O **Regulador de Tensão Step-Up - MT3608** é responsável pela elevação da tensão da bateria (3.7V) para 5V, garantindo operação contínua mesmo com queda de tensão da célula. (Aerosemi Technology Co., Ltd., 2025).

O **Sensor de Corrente ACS712 5A** permite medir a corrente elétrica que passa por um condutor, sendo utilizado no projeto para monitorar o funcionamento da bomba d'água. A versão 5A é adequada para aplicações de baixa potência, como sistemas de irrigação compactos. (Allegro MicroSystems, 2025).

O **Sensor de Fluxo de Água YF-S201** monitora a passagem da água pelo encanamento. Com ele, é possível verificar se há fluxo efetivo durante o acionamento da bomba e detectar possíveis obstruções ou falhas no sistema. (Seeed Studio, 2025).

O **Sensor digital de Temperatura e Umidade DHT22** é utilizado para medir a temperatura e a umidade relativa do ar com maior precisão e faixa de operação do que o modelo DHT11. Capaz de operar entre -40 °C e +80 °C, e com precisão de ±0,5 °C e ±2% de umidade, o DHT22 é adequado para aplicações que exigem dados ambientais mais confiáveis e consistentes. (Aosong Electronics Co., Ltd., 2025)

Neste projeto, o sensor foi incluído com o objetivo de coletar dados do ambiente, considerando que a temperatura e a umidade do ar influenciam diretamente na taxa de evaporação da água no solo. Esses dados, embora inicialmente apenas monitorados, poderão ser utilizados em desenvolvimentos futuros para implementar funcionalidades como alertas inteligentes e agendamento autônomo da irrigação, otimizando ainda mais o uso dos recursos hídricos.

O **Sensor Capacitivo de Umidade do Solo V2** é um componente projetado para medir o teor de umidade no solo utilizando variações de capacitância em vez de condutividade elétrica. Diferente dos sensores resistivos, que estão sujeitos à corrosão por eletroólise ao longo do tempo, este modelo capacitivo oferece maior resistência à oxidação devido à sua construção com materiais isolantes e revestimento protetivo, o que o torna ideal para uso prolongado em ambientes úmidos e agressivos. No contexto deste projeto, a leitura analógica é recebida pelo ESP32, que interpreta o nível de umidade do solo e decide de forma autônoma sobre a ativação da irrigação. A escolha deste sensor

se deu não apenas pela durabilidade, mas também pela boa estabilidade de leitura e baixa interferência eletromagnética, características importantes para ambientes agrícolas ou externos.(DFRobot, 2025).

O **Relé de Estado Sólido Fotek SSR-40DA** é um atuador utilizado para controlar a comutação de cargas elétricas em corrente alternada, como motores, válvulas e bombas, a partir de sinais de controle em corrente contínua. Este modelo suporta correntes de até 40A em 24–380V AC e opera com sinais de controle entre 3–32V DC, sendo ideal para aplicações que exigem isolamento galvânico e comutação silenciosa. Neste projeto, o relé é responsável por acionar a válvula solenoide do sistema de irrigação de forma automatizada, conforme os dados processados pelo microcontrolador. (Fotek Controls Co., Ltd., 2025)

A **Válvula Solenoide Emicol 320300** é utilizada para controlar o fluxo de água no sistema hidráulico, funcionando como um atuador eletromecânico que libera ou bloqueia a passagem de fluido com base em sinais elétricos. No contexto deste projeto, a válvula simula o acionamento de motobombas em sistemas de irrigação, permitindo que o ESP32, por meio do relé de estado sólido, atue automaticamente no controle da irrigação conforme os níveis de umidade do solo. (Emicol Eletro Eletrônica Ltda., 2025)

A segurança do servidor de borda como um todo — incluindo sistema operacional, aplicações em contêiner, banco de dados local e lógica da aplicação — não foi alvo direto da modelagem de ameaças. A análise se concentrou exclusivamente na comunicação recebida pelo servidor via MQTT, proveniente dos sensores conectados ao ESP32.

2.2.5 Componentes de Software do Sistema

A arquitetura de software foi projetada para garantir modularidade, escalabilidade e interoperabilidade entre os módulos embarcados, o servidor de borda e o servidor em nuvem. Abaixo, são descritos os principais componentes de software utilizados neste projeto:

O **Django** é um framework web de alto nível escrito em Python, escolhido para o desenvolvimento do *backend web* da aplicação. Oferece recursos como ORM, autenticação, painel administrativo e segurança integrada, acelerando o desenvolvimento e facilitando a manutenção da aplicação. Está presente tanto no servidor de nuvem quanto no servidor de borda, proporcionando uma interface unificada para gerenciamento dos dados e do sistema. (FOUNDATION, 2024)

O **SQLite** é um banco de dados leve e incorporado, utilizado localmente no servidor de borda. Sua principal vantagem é a simplicidade de configuração e operação, dispensando um processo servidor-cliente, o que o torna ideal para aplicações embarcadas ou de borda com recursos limitados. (SQLite Consortium, 2025)

O **PostgreSQL** é o banco de dados relacional adotado no servidor de nuvem. Robusto, seguro e com suporte a extensões avançadas, ele é apropriado para cenários de

produção que exigem alta confiabilidade, escalabilidade e integridade dos dados. (PostgreSQL Global Development Group, 2025)

O **Gunicorn** (Green Unicorn) é um servidor WSGI compatível com aplicações Python, utilizado para hospedar a aplicação Django em produção. Atua como interface entre o código Python e servidores web como o Nginx, sendo utilizado em ambos os servidores para garantir desempenho e estabilidade. (Gunicorn Project, 2025)

O **Nginx** é um servidor web leve e de alto desempenho utilizado como *proxy reverso* e servidor de arquivos estáticos. Sua função é encaminhar requisições HTTP/HTTPS para o Gunicorn, além de fornecer arquivos da interface web com maior eficiência. Presente tanto no servidor de nuvem quanto no servidor de borda. (F5, Inc., 2025)

O **NanoMQ** é o broker MQTT adotado neste projeto. Com foco em dispositivos embarcados e aplicações em tempo real, sua leveza e suporte a múltiplos protocolos o tornam ideal para operar no servidor de borda, garantindo comunicação eficiente com os módulos sensores e atuadores via MQTT. (NanoMQ Project, 2025)

A biblioteca **Paho MQTT**, escrita em Python, é utilizada como cliente MQTT no servidor de borda. Sua responsabilidade é publicar e assinar tópicos MQTT, permitindo a recepção dos dados dos sensores e envio de comandos aos atuadores de forma assíncrona e leve. (Eclipse Foundation, 2025)

O **RabbitMQ** é o broker de mensagens baseado no protocolo AMQP, utilizado para mensageria assíncrona entre os serviços do sistema, como processamento de dados, notificações e agendamentos. Sua robustez e escalabilidade o tornam ideal para aplicações distribuídas e resilientes, estando presente tanto na borda quanto na nuvem. (VMware, Inc., 2025)

O **Celery** é o gerenciador de tarefas assíncronas responsável por executar atividades em segundo plano, como análise de dados, geração de alertas e sincronizações entre a borda e a nuvem. Integra-se ao RabbitMQ, que atua como middleware de mensagens. (Celery Project, 2025)

O **Redis** é utilizado como *backend* de resultados para o Celery, armazenando o estado e retorno das tarefas processadas de forma assíncrona. Sua estrutura baseada em chave-valor permite acesso extremamente rápido aos dados em memória. (Redis Ltd., 2025)

O **Docker** é a tecnologia de containerização utilizada para empacotar os serviços da aplicação em ambientes isolados e portáveis. Através dos containers, garante-se a replicabilidade do ambiente em diferentes infraestruturas, tanto em nuvem quanto na borda. (Docker Inc., 2024)

O **Docker Compose** facilita a orquestração de múltiplos containers Docker, permitindo o gerenciamento centralizado dos serviços da aplicação (banco de dados, backend, brokers de mensagem, etc.) por meio de um único arquivo de configuração. (Docker Inc., 2025)

O **Orange Pi OS**, baseado no Debian, é o sistema operacional instalado no computador de borda. Sua compatibilidade com ferramentas Linux e a leveza em comparação com distribuições genéricas o tornam ideal para execução em dispositivos ARM com recursos computacionais reduzidos. (Shenzhen Xunlong Software Co., Ltd., 2025)

A **Plataforma Arduino** foi utilizada para o desenvolvimento do firmware dos microcontroladores ESP32-S3. Sua comunidade ativa, simplicidade e compatibilidade com bibliotecas diversas permitem a leitura de sensores, controle de atuadores e comunicação via MQTT de forma ágil e confiável.

Dois firmwares distintos foram desenvolvidos: um para o **Módulo Sensor**, focado na aquisição de dados ambientais (umidade, temperatura, fluxo de água, tensão da bateria), e outro para o **Módulo Atuador**, responsável por controlar relés e válvulas com base nos comandos recebidos.

Os componentes de software descritos acima foram todos utilizados na implementação real do sistema, garantindo seu funcionamento modular e escalável. Contudo, a análise de segurança desta dissertação restringe-se aos módulos embarcados (ESP32 e sensores) e à comunicação segura com o servidor de borda via MQTT/TLS. Aspectos como autenticação de usuários, validação de comandos na aplicação web e segurança da base de dados em nuvem estão fora do escopo desta análise, mas são recomendados como extensão futura.

2.3 Considerações sobre Segurança no Cenário Proposto

A segurança em sistemas IoT é um aspecto crítico, especialmente em aplicações agrícolas que operam em ambientes abertos e com infraestrutura limitada. Nesta seção, são descritas as principais diretrizes de segurança consideradas no projeto, embora nem todas tenham sido aplicadas integralmente em todos os componentes do sistema. A modelagem de ameaças e a implementação de contramedidas foram concentradas exclusivamente no microcontrolador ESP32, nos sensores conectados e na comunicação com o servidor de borda via MQTT sobre TLS.

As medidas descritas a seguir representam tanto mecanismos efetivamente aplicados quanto recomendações para a proteção futura de todo o sistema:

- **Autenticação**: Para o escopo abordado, a autenticação entre o ESP32 e o servidor de borda foi implementada com o uso de certificados digitais e verificação de identidade via TLS. Para os demais módulos, a autenticação permanece como medida futura.
- **Criptografia**: Foi adotada a criptografia de dados em trânsito entre o ESP32 e o servidor de borda, utilizando MQTT sobre TLS. Essa camada garante a confidencialidade dos dados sensoriais coletados e impede sua interceptação durante o transporte.

- **Controle de Acesso**: O controle de acesso aos dispositivos foi parcialmente implementado no nível embarcado, com a validação de mensagens recebidas pelo ESP32. Mecanismos mais abrangentes de controle de usuários na aplicação web e servidores permanecem fora do escopo da análise de segurança desta dissertação.
- **Segurança Física**: A segurança física dos módulos embarcados foi abordada por meio da utilização de caixas herméticas, vedação contra umidade e estruturas de fixação resistentes. A redundância energética, provida por baterias recarregáveis e painéis solares, também contribui para a resiliência do sistema em caso de falhas intencionais ou acidentais na rede elétrica.

A inclusão dessas medidas, mesmo que parcialmente, fortalece a integridade da coleta de dados e a confiabilidade da automação agrícola, reduzindo o risco de falhas operacionais e comprometimento dos dados sensoriais. A continuidade desse trabalho poderá contemplar a aplicação das medidas descritas aos demais módulos da arquitetura, como servidores e interfaces de usuário.

3 REVISÃO DE METODOLOGIAS DE MODELAGEM DE AMEAÇAS

No cenário atual, onde os sistemas digitais desempenham um papel central em diversas áreas, a segurança da informação tornou-se um fator crítico para organizações de todos os portes. O avanço das ameaças cibernéticas exige a adoção de metodologias eficazes que permitam a identificação, análise e mitigação de riscos em sistemas computacionais. Nesse contexto, *Threat Modeling* ou Modelagem de Ameaças, surge como uma abordagem estruturada para compreender e antecipar potenciais ataques cibernéticos, sendo amplamente empregada na proteção de sistemas baseados em Internet das Coisas (IoT) (OWASP, 2024). A Modelagem de Ameaças possibilita que desenvolvedores e profissionais de segurança identifiquem vulnerabilidades em sistemas e planejem estratégias para mitigá-las. Além de identificar pontos fracos em infraestruturas existentes, essa abordagem também garante que a segurança seja incorporada desde as fases iniciais do desenvolvimento de novos sistemas.

Diversas metodologias de Modelagem de Ameaças foram desenvolvidas ao longo dos anos, cada uma com abordagens específicas para análise e mitigação de riscos. Entre as mais amplamente adotadas, destacam-se STRIDE (*Spoofing*, *Tampering*, *Repudiation*, *Information Disclosure*, *Denial of Service*, e *Elevation of Privilege*), DREAD (*Damage potential*, *Reproducibility*, *Exploitability*, *Affected users*, *Discoverability*) e PASTA (*Process for Attack Simulation and Threat Analysis*). Além dessas, outras abordagens como o *NIST Cybersecurity Framework*, *OWASP IoT Top Ten*, *LINDDUN* e *VAST* também são frequentemente mencionadas na literatura.

Com o intuito de compreender melhor a aplicabilidade dessas metodologias no contexto da segurança em IoT, foi conduzida uma revisão sistemática da literatura, com o objetivo de identificar as abordagens mais utilizadas atualmente na proteção de dispositivos IoT. Além disso, buscou-se obter percepções práticas sobre cenários reais, nos quais essas metodologias foram aplicadas, analisando seus impactos e resultados. Esses achados forneceram subsídios para validar a adoção da metodologia mais relevante no contexto da **automatização da irrigação dos dispositivos IoT da empresa TULKAS**, garantindo uma abordagem fundamentada para a segurança da informação. Esta revisão sistemática tem como propósito fornecer uma análise abrangente sobre a modelagem de ameaças e suas metodologias, destacando suas características, benefícios, limitações e aplicações práticas. Por meio da análise da literatura existente, busca-se identificar as melhores práticas e recomendações para a implementação eficiente do *Threat Modeling*, oferecendo uma base sólida para pesquisadores e profissionais de segurança da informação.

3.1 Etapas do processo de revisão sistemática

Esta seção aborda as etapas e processos utilizados na revisão sistemática a fim de encontrar, discutir e responder possíveis problemas que podem ser levados em consideração antes de escolher um dos diversos métodos que podem ser escolhidos, bem como seus

resultados ao longo do seu desenvolvimento.

3.1.1 Identificação de Termos e Sinônimos

- Threat Modeling
  - Threat Modeling
  - Threat Analysis
  - Threat Identification
  - Threat Assessment
- Security Framework
  - Security Framework
  - Cybersecurity Framework
  - Information Security Framework
- Cybersecurity Threats
  - Cybersecurity Threats
  - Information Security Threats
  - Digital Security Threats
- Risk Assessment
  - Risk Assessment
  - Risk Analysis
  - Risk Evaluation
  - Risk Management
- IoT Security
  - IoT Security
  - Internet of Things Security
  - Smart Device Security
  - Embedded Systems Security
- Attack Modeling
  - Attack Modeling
  - Adversarial Modeling

  - Cyber Attack Analysis

- Security Methodologies
  - Security Methodologies
  - Threat Modeling Approaches

3.1.2 Metodologias Específicas

- STRIDE
- DREAD
- PASTA
- LINDDUN
- Trike
- VAST
- OWASP IoT Top Ten
- OCTAVE
- NIST Cybersecurity Framework

3.1.3 Strings de Busca

As strings de busca podem ser construídas utilizando combinações desses termos e sinônimos. Por exemplo:

```
("IoT security" OR "Internet of Things security") AND ("threat modeling"
OR "threat analysis" OR "security framework" OR "risk assessment") AND
("STRIDE" OR "DREAD" OR "PASTA" OR "LINDDUN" OR "Trike" OR "OCTAVE" OR
"NIST" OR "OWASP" OR "VAST")
```

Esses termos e sinônimos abrangem uma vasta gama de conceitos relacionados à modelagem de ameaças e suas metodologias, garantindo uma revisão sistemática abrangente e eficaz. Percebe-se que são utilizados os parênteses para agrupar palavras de uma mesma categoria, podendo utilizar as palavras OR para ampliar as possibilidades de informações encontradas, enquanto o AND tem a função de limitar ou restringir os dados de uma consulta. Após essa análise, foi utilizada a mesma String em 3 bibliotecas de armazenamento de artigos (ACM, Springer e IEEE) obtendo os seguintes resultados (Tabela 1).

Biblioteca	Artigos ou Papers encontrados
ACM	173
Springer	409
IEEE	18
TOTAL	**600**

Tabela 1 – Bibliotecas

Vale ressaltar que esses números foram obtidos a partir de uma consulta no dia 14/02/2025, podendo ter variações em seus resultados em datas posteriores. Outro elemento que vale ser informado é que, para cada biblioteca, existem ferramentas de filtro distintas e não padronizadas em seus mecanismos de busca, o que garante uma interação distinta para uma busca mais refinada.

3.1.4 Critérios de Inclusão

Em seguida, são aplicados filtros adicionais para facilitar a localização de artigos mais relevantes ao tema pesquisado. Para isso, são utilizados critérios que adicionam os artigos encontrados a uma base de dados própria do pesquisador. No caso em questão, foram utilizados os seguintes critérios:

- Artigos publicados nos últimos 5 anos (2020 a 2025);
- Estudos sobre metodologias de *Threat Modeling* aplicadas à segurança em IoT;
- Trabalhos que descrevem ou analisam metodologias como STRIDE, DREAD, PASTA, LINDDUN, Trike, VAST, OWASP ou NIST Cybersecurity Framework;
- Estudos que apresentam análise de riscos, identificação de ameaças ou estratégias de mitigação relacionadas à segurança de dispositivos IoT;
- Artigos que incluem aplicações práticas, estudos de caso ou implementações reais de modelagem de ameaças em IoT;
- Estudos escritos em inglês para garantir a padronização da revisão.

3.1.5 Critérios de Exclusão

Logo em seguida, são utilizados critérios que possam diminuir um pouco mais a quantidade de artigos para serem analisados pelo pesquisador. Neste caso, foram utilizados os critérios abaixo para diminuir os artigos utilizados:

- Artigos sem acesso ao texto completo;
- Estudos que não abordam diretamente metodologias de *Threat Modeling* aplicadas à segurança em IoT;

- Trabalhos que mencionam *Threat Modeling*, mas não descrevem ou analisam metodologias específicas;
- Publicações que não apresentam avaliação comparativa, estudos de caso ou implementações práticas;
- Estudos que tratam de segurança em IoT, mas sem relação com modelagem de ameaças;
- Trabalhos publicados em idiomas diferentes do inglês;
- Artigos duplicados ou versões prévias de estudos já incluídos.

3.1.6 Critérios de Qualidade

Com uma base menor, mas ainda grande, são utilizados os critérios de qualidade para medir de forma qualitativa o quão interessante e relevante é aquele artigo para o seu estudo. Neste caso, foram utilizados 4 critérios de qualidade que podem ser vistos abaixo:

- Estudos com metodologia bem definida, clara e replicável;
- Inclusão de métricas de avaliação de desempenho das metodologias de *Threat Modeling*;
- Amostras representativas e estatisticamente significativas para garantir validade dos resultados;
- Clareza na apresentação dos resultados, incluindo vantagens, limitações e aplicações práticas das metodologias analisadas.

3.1.7 Resultado

Como resultado, foram encontrados artigos relevantes para análise, que possuem alta qualidade de informação e relevância para o objetivo proposto inicialmente. Os dados extraídos dos artigos selecionados foram organizados na Tabela 2, apresentando as metodologias identificadas e os respectivos estudos que as mencionam. No entanto, para facilitar a compreensão da frequência com que cada metodologia foi citada, a Figura 8 apresenta um gráfico ilustrando a distribuição das metodologias de *Threat Modeling* mais mencionadas na literatura analisada. Essa análise gráfica complementa a tabela de artigos ao fornecer uma visão **quantitativa e comparativa** sobre o uso das metodologias, facilitando a interpretação das tendências na área de segurança em IoT.

A Figura 8 evidencia que as metodologias **STRIDE e NIST** são as mais utilizadas para modelagem de ameaças em sistemas de **IoT**, sendo mencionadas em **13 e 12 artigos**, respectivamente. Isso demonstra que essas abordagens são amplamente aceitas e aplicadas na literatura para análise de riscos em dispositivos conectados.

Metodologia	Título
STRIDE	A Comparative Analysis of Threat Modelling Methods: STRIDE, DREAD, VAST, PASTA, OCTAVE, and LINDDUN
	Threat Modelling in Virtual Assistant Hub Devices
	Securing RC Based P2P Networks: A Blockchain-Based Access Control Framework Utilizing Ethereum Smart Contracts for IoT and Web 3.0
	Internet of Things Security: Modelling Smart Industrial Thermostat for Threat Vectors and Common Vulnerabilities
	A Study on Blockchain Architecture Design Decisions and Their Security Attacks and Threats
	STRIPED: A Threat Analysis Method for IoT Systems
	Threats and Risk on Using Digital Technologies for Remote Health Care Process
	An Ensemble Approach for IoT Firmware Strength Analysis using STRIDE Threat Modeling and Reverse Engineering
	A Comprehensive Approach for Applying Threat Modeling to Internet of Things Systems
	Threat Modeling with Mitre ATT&CK Framework Mapping for SD-IOT Security Assessment and Mitigations
	STRIDE-based Cyber Security Threat Modeling for IoT-enabled Precision Agriculture Systems
	Narrow-band Internet of Things Protocol Standards: Survey of Security and Privacy Control Effectiveness
	Security Analysis of Large Scale IoT Network for Pollution Monitoring in Urban India
NIST	Threat Modeling and IoT Attack Surfaces
	Towards a Fog Based Computing Conceptual Framework for Biometric Digital Identification Systems
	Refining Mosca's Theorem: Risk Management Model for the Quantum Threat Applied to IoT Protocol Security
	On Reducing Underutilization of Security Standards by Deriving Actionable Rules: An Application to IoT
	CPSIOTSEC'20: 2020 Joint Workshop on CPS&IoT Security and Privacy
	Emerging Cybersecurity Capability Gaps in the Industrial Internet of Things: Overview and Research Agenda
	The Challenges and Processes of Achieving Optimal Implementation of Zero Trust Architecture in Workplace
	Construction of a Privacy Management Framework for AIoT Enterprises Based on Risk Analysis
	Risk Assessment for Emerging Domains (IoT, Cloud Computing, and AI)
	A Review of Security Standards and Frameworks for IoT-Based Smart Environments
	Explaining Cyber Risks in Transportation 5.0: A Data Driven Approach
	Dynamic Security Management of Systems on Chip via Embedded FPGA in 22nm CMOS Technology
DREAD	Categorizing IoT Services According to Security Risks
	A Comparative Analysis of Threat Modelling Methods: STRIDE, DREAD, VAST, PASTA, OCTAVE, and LINDDUN
	Threats and Risk on Using Digital Technologies for Remote Health Care Process
	A Comprehensive Approach for Applying Threat Modeling to Internet of Things Systems
LINDDUN	A Comparative Analysis of Threat Modelling Methods: STRIDE, DREAD, VAST, PASTA, OCTAVE, and LINDDUN
	Threat Modelling in Virtual Assistant Hub Devices
OCTAVE	A Comparative Analysis of Threat Modelling Methods: STRIDE, DREAD, VAST, PASTA, OCTAVE, and LINDDUN
	Risk Assessment in Smart Aging Care Systems: An Elderly-Centered Perspective
OWASP	Compliance Evaluation of Cryptographic Security Requirements on IoT Gateways
	Risk Assessment for Emerging Domains (IoT, Cloud Computing, and AI)
PASTA	A Comparative Analysis of Threat Modelling Methods: STRIDE, DREAD, VAST, PASTA, OCTAVE, and LINDDUN
VAST	A Comparative Analysis of Threat Modelling Methods: STRIDE, DREAD, VAST, PASTA, OCTAVE, and LINDDUN

Tabela 2 – Lista de artigos categorizados por metodologia de *Threat Modeling*

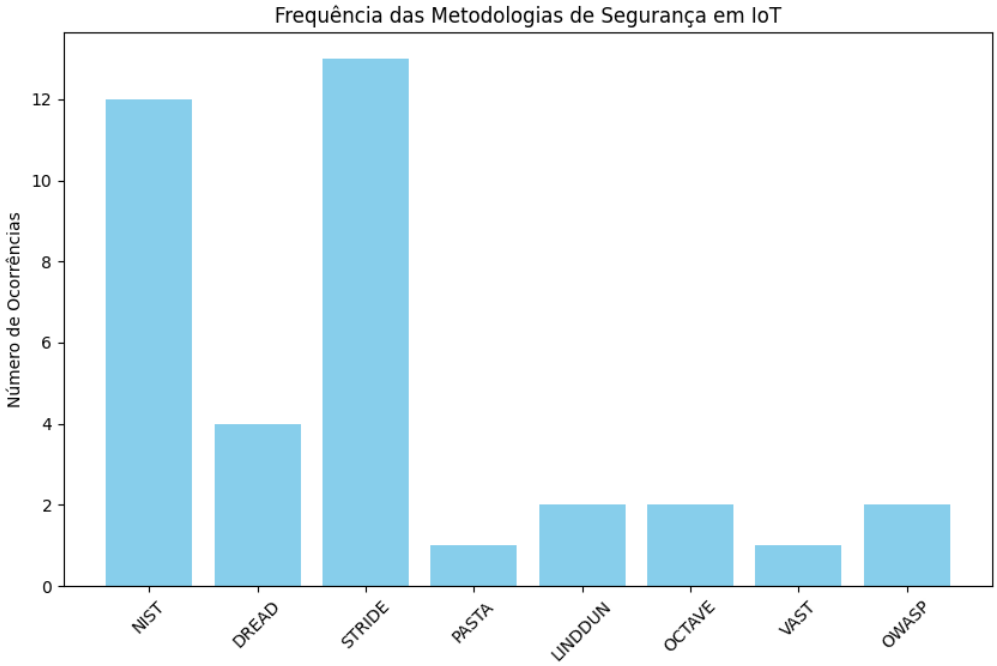

Figura 8 – Frequência das metodologias de *Threat Modeling* encontradas na revisão sistemática.

Outras metodologias, como **DREAD, OWASP, OCTAVE e LINDDUN**, aparecem com menor frequência, indicando que são aplicadas em contextos mais específicos ou como complementos a métodos mais consolidados. Já metodologias como **PASTA e VAST** são raramente citadas na literatura revisada, sugerindo que sua adoção no contexto de IoT ainda é limitada.

3.2 Utilização do Modelo STRIDE na Segurança de Sistemas IoT

A crescente adoção de sistemas baseados na Internet das Coisas (IoT) impõe desafios significativos de segurança. Modelos de análise de ameaças, como STRIDE, são empregados para categorizar e mitigar riscos nesses ambientes. STRIDE é composto por seis categorias de ameaças: *Spoofing* (falsificação de identidade), *Tampering* (modificação indevida), *Repudiation* (negação de ações realizadas), *Information Disclosure* (divulgação de informações sensíveis), *Denial of Service* (negação de serviço) e *Elevation of Privilege* (elevação de privilégios). Vários estudos aplicam essa abordagem para identificar vulnerabilidades em diferentes domínios da IoT, conforme discutido a seguir.

3.2.1 Artigos da base de dados da IEEE

No trabalho de (CHORFA; YOUSSEF; JEMAI, 2023), STRIDE foi utilizado para modelar ameaças em redes definidas por software para IoT (SD-IoT). O estudo classificou vetores de ataque dentro do modelo STRIDE e mapeou as táticas, técnicas e procedimentos (TTPs) associadas a cada ameaça por meio do framework Mitre ATT&CK. A pesquisa demonstrou que a modelagem sistemática permitiu a identificação de vulnerabilidades críticas, especialmente no contexto de redes veiculares definidas por software.

Já em (ASIF et al., 2021), STRIDE foi empregado para analisar os riscos de segurança em sistemas de agricultura de precisão baseados em IoT. A abordagem estruturada identificou 58 ameaças potenciais, categorizadas conforme os seis componentes do modelo. O estudo também propôs contramedidas, como criptografia de dados e autenticação multifator, para mitigar os riscos associados.

O artigo de (IHITA et al., 2021) aplicou STRIDE na análise de segurança de uma rede de sensores IoT para monitoramento de poluição em áreas urbanas. A pesquisa avaliou vulnerabilidades em diferentes camadas da infraestrutura, como redes Wi-Fi, comunicação móvel e painéis de controle. Foram identificadas ameaças como *spoofing* de dispositivos, ataques de negação de serviço e vazamento de informações devido a falhas na criptografia.

No estudo conduzido por (YOCAM, 2020), a metodologia STRIDE foi utilizada para avaliar a segurança de protocolos de comunicação NB-IoT (Narrowband IoT). A pesquisa comparou os padrões SigFox, LoRa e 3GPP, identificando suas vulnerabilidades e propondo melhorias em autenticação e criptografia para reduzir o impacto de ataques

como falsificação de identidade e interceptação de dados.

O trabalho de (YAQUB et al., 2022) combinou STRIDE e engenharia reversa para analisar a segurança de firmwares de dispositivos IoT. Foram investigadas 4.364 imagens de firmware, sendo que 58,8% delas apresentavam vulnerabilidades que poderiam ser exploradas por meio de ataques de *spoofing*, manipulação de dados, negação de serviço e elevação de privilégios. A abordagem proposta permitiu mapear as principais falhas e sugerir estratégias para seu endurecimento.

Por fim, em (SILVA et al., 2022), foi aplicada uma abordagem abrangente de modelagem de ameaças baseada em STRIDE e em DREAD. O estudo identificou 29 tipos de ataques e 20 contramedidas específicas para sistemas IoT. A abordagem foi aplicada a um sistema de fechadura inteligente (*smart lock*), onde vulnerabilidades severas foram detectadas, permitindo a definição de medidas preventivas para mitigar riscos.

A análise dos artigos revela que STRIDE é um modelo amplamente utilizado na segurança de sistemas IoT. Sua aplicação permite uma categorização sistemática das ameaças, possibilitando a identificação de vulnerabilidades e a proposição de contramedidas adequadas. Diferentes domínios da IoT, como redes definidas por software, agricultura de precisão, monitoramento ambiental, comunicação de baixa potência e segurança de firmwares, se beneficiam dessa abordagem estruturada para mitigar riscos e fortalecer a resiliência dos sistemas.

3.2.2 Artigos da base de dados da ACM

O estudo de (AHMADJEE et al., 2022) investiga decisões arquiteturais em sistemas blockchain e os riscos de segurança associados. Os autores realizam uma análise sistemática e mapeiam ameaças de segurança utilizando o modelo STRIDE. Cada ameaça identificada é categorizada conforme seu impacto na segurança da blockchain, abordando aspectos como falsificação de identidade (*spoofing*), manipulação de dados (*tampering*) e elevação de privilégios (*elevation of privilege*). O estudo destaca como decisões equivocadas na configuração de mecanismos de consenso, gerenciamento de chaves e contratos inteligentes podem introduzir vulnerabilidades críticas. O mapeamento das ameaças visa fornecer diretrizes para arquitetos de software tomarem decisões mais seguras no desenvolvimento de sistemas baseados em blockchain.

(KHATIWADA et al., 2023) analisam ameaças em tecnologias digitais utilizadas no contexto de saúde remota, com foco em dispositivos médicos da Internet das Coisas Médicas (IoMT). O estudo utiliza o modelo STRIDE para identificar vulnerabilidades na geração, armazenamento e transmissão de dados médicos sensíveis. A análise aponta que ameaças de manipulação de dados (*tampering*) são as mais prevalentes, seguidas por riscos de divulgação indevida de informações (*information disclosure*) e negação de serviço (*denial of service*). Além de STRIDE, os autores aplicam o modelo DREAD para quantificar os riscos e propor contramedidas para os ataques mais críticos. Os resultados evidenciam

a necessidade de fortalecer mecanismos de autenticação, criptografia e monitoramento contínuo para mitigar riscos em ambientes de saúde remota.

Os dois estudos analisados demonstram a relevância do modelo STRIDE para a identificação e mitigação de ameaças em diferentes contextos tecnológicos. Enquanto (AHMADJEE et al., 2022) aplicam STRIDE para avaliar vulnerabilidades em arquiteturas blockchain, (KHATIWADA et al., 2023) utilizam o modelo para mapear ameaças em dispositivos IoMT e sistemas de saúde digital. Ambos os trabalhos reforçam a importância da modelagem de ameaças na fase de projeto, permitindo a implementação de medidas de segurança mais eficazes e reduzindo os riscos de ataques.

3.2.3 Artigos da base de dados da Springer

O estudo de (LECLAIR, 2021) investiga a modelagem de ameaças em dispositivos assistentes virtuais, como Amazon Alexa e Google Home. O autor compara diferentes abordagens para identificar qual metodologia de modelagem de ameaças é mais eficiente nesse contexto. Os resultados indicam que a combinação de STRIDE com o método LINDDUN GO oferece a melhor cobertura de ameaças, especialmente no que se refere à privacidade dos usuários. STRIDE foi utilizado para categorizar as principais ameaças, evidenciando vulnerabilidades relacionadas à falsificação de identidade (*spoofing*), manipulação de dados (*tampering*) e negação de serviço (*DoS*). A principal limitação identificada foi a dificuldade de priorizar ameaças, dado que STRIDE não fornece uma classificação de riscos baseada na gravidade ou probabilidade de exploração.

O trabalho de (GHOSH et al., 2024) propõe um modelo de controle de acesso descentralizado utilizando *smart contracts*, na blockchain Ethereum, para redes *peer-to-peer* (P2P). Para garantir a segurança da infraestrutura, os autores adotam uma abordagem híbrida baseada nos princípios de *Confidencialidade, Integridade e Disponibilidade* (CIA) e STRIDE. O modelo STRIDE foi empregado para identificar ameaças potenciais dentro do sistema, com destaque para a *elevação de privilégio* em contratos inteligentes maliciosos e tentativas de *spoofing* para burlar as permissões de acesso. O estudo demonstra como STRIDE pode ser integrado a um ambiente descentralizado, auxiliando na criação de mecanismos de resposta e mitigação de ataques em redes blockchain.

(NAIK et al., 2024) realizam uma análise comparativa entre diferentes métodos de modelagem de ameaças, incluindo STRIDE, DREAD, VAST, PASTA, OCTAVE e LINDDUN. STRIDE é avaliado como um dos métodos mais consolidados, especialmente útil para análise de segurança de software. Os autores destacam que STRIDE é amplamente utilizado devido à sua simplicidade e estruturação clara, mas apontam suas limitações, como a ausência de uma avaliação quantitativa de riscos. Além disso, o estudo enfatiza a integração de STRIDE com diagramas de fluxo de dados (*DFD*), que permitem uma identificação mais precisa das fronteiras e pontos vulneráveis dos sistemas analisados.

A análise dos três estudos evidencia a relevância contínua do modelo STRIDE na

modelagem de ameaças. Em diferentes cenários, como segurança de assistentes virtuais, redes P2P e avaliações comparativas de métodos, STRIDE demonstrou sua eficácia na identificação de riscos e vulnerabilidades. No entanto, suas limitações, como a falta de um mecanismo de priorização de ameaças, indicam a necessidade de complementação com outras abordagens, como LINDDUN e CVSS. Dessa forma, STRIDE continua sendo uma ferramenta fundamental para a segurança de sistemas computacionais, especialmente quando integrado a outras metodologias.

3.3 Justificativa da Escolha das Metodologias Aplicadas

Com base na revisão sistemática conduzida, observou-se que as metodologias **STRIDE** e **DREAD** são amplamente adotadas em cenários de segurança envolvendo dispositivos IoT. A metodologia STRIDE destacou-se pela sua capacidade de categorizar ameaças de forma estruturada e aplicável a diferentes camadas de sistemas conectados, como sensores, microcontroladores e protocolos de comunicação. Já a metodologia DREAD, embora menos frequente na literatura, demonstrou ser útil para avaliar e priorizar riscos com base em critérios quantitativos, especialmente quando utilizada de forma complementar ao STRIDE.

A aplicação prática destas metodologias no presente trabalho se justifica pela natureza do subsistema analisado: um conjunto de sensores conectados a um microcontrolador ESP32, com comunicação via protocolo MQTT sobre TLS com um servidor de borda. Este contexto é semelhante aos descritos nos estudos analisados — como aqueles que tratam de segurança em agricultura de precisão, redes definidas por software e sistemas de monitoramento ambiental — nos quais STRIDE foi utilizado para identificar ameaças como *spoofing*, *tampering* e *denial of service*.

Além disso, o uso de DREAD neste projeto contribuiu para a priorização das vulnerabilidades identificadas por meio do STRIDE, permitindo uma escolha mais criteriosa das contramedidas a serem implementadas, com foco em viabilidade técnica e impacto real. A combinação dessas duas metodologias possibilitou uma abordagem prática, com base teórica sólida, para a análise de ameaças ao subsistema embarcado de irrigação.

Dessa forma, a escolha de STRIDE e DREAD foi fundamentada não apenas em sua recorrência na literatura, mas também na compatibilidade entre suas características e os objetivos deste projeto. A próxima etapa, descrita no Capítulo 4, apresenta a aplicação prática dessas metodologias no cenário real desenvolvido, com destaque para as ameaças mapeadas, os testes de segurança realizados e as estratégias de mitigação propostas.

4 APLICAÇÃO DA METODOLOGIA SELECIONADA

A segurança de software tornou-se uma prioridade devido ao aumento dos ataques cibernéticos e à crescente complexidade dos sistemas. A Modelagem de Ameaças (*Threat Modeling*) é uma abordagem sistemática que permite identificar e mitigar ameaças ao longo de todo o ciclo de vida do desenvolvimento de software. Essa técnica envolve a identificação de possíveis vetores de ataque, a análise de vulnerabilidades existentes e a definição de medidas de mitigação adequadas (SHOSTACK, 2014).

Tradicionalmente aplicada a sistemas operacionais, redes e aplicações web, a modelagem de ameaças tem sido cada vez mais empregada em sistemas embarcados e soluções de Internet das Coisas (IoT), devido à sua natureza distribuída e muitas vezes exposta a ambientes fisicamente inseguros. Entre as metodologias mais conhecidas, destacam-se **STRIDE**, que categoriza ameaças como *Spoofing*, *Tampering* e *Denial of Service* (INFOMACH, 2024), e **DREAD**, que classifica os riscos com base em critérios como potencial de dano (*Damage Potential*) e reprodutibilidade (*Reproducibility*) (CONVISO, 2024).

Neste capítulo, é apresentada a aplicação prática das metodologias STRIDE e DREAD no contexto de um sistema de irrigação automatizado baseado em IoT. É importante ressaltar que a análise de segurança desenvolvida nesta dissertação se restringe ao subsistema composto por:

- o microcontrolador **ESP32 S3 N16R8**;
- os sensores conectados ao ESP32;
- a comunicação entre o ESP32 e o servidor de borda (Orange Pi), realizada via protocolo MQTT sobre TLS.

Esse recorte foi definido com base em sua criticidade: é nesta camada do sistema que ocorre a coleta, processamento e transmissão dos dados sensoriais, os quais influenciam diretamente as decisões de irrigação automatizada. Outros módulos, como a aplicação web, banco de dados e infraestrutura em nuvem, não foram incluídos na análise de ameaças desta dissertação.

4.1 Modelagem de Ameaças

A modelagem de ameaças é uma prática estruturada que visa identificar e avaliar possíveis ameaças à segurança de um sistema, com o objetivo de determinar as medidas necessárias para mitigá-las (OWASP, 2024). Segundo o NIST, uma ameaça é qualquer evento ou condição com potencial para impactar negativamente as operações organizacionais ou a integridade dos sistemas (NIST, 2006).

No contexto deste projeto, a modelagem de ameaças foi aplicada ao subsistema embarcado e sua comunicação com a borda. Para representar graficamente os componentes envolvidos, as fronteiras de confiança e os canais de comunicação, foi utilizado o **C4**

Model, uma abordagem moderna de modelagem arquitetural que permite visualizar os níveis de abstração da solução em camadas.

O processo de modelagem adotado seguiu as seguintes etapas:

1. **Identificação de Ativos**: Determinar ativos que precisam ser protegidos: sensores, dados de umidade do solo, comandos de irrigação e interfaces de comunicação.

2. **Modelagem Arquitetural com C4 Model**: Representar graficamente os elementos do subsistema analisado, como ESP32, sensores, MQTT broker e servidor de borda, destacando as fronteiras de confiança e os canais seguros.

3. **Identificação de Ameaças (STRIDE)**: Analisar os componentes e interações descritos no modelo arquitetural, categorizando as ameaças.

4. **Classificação de Ameaças (DREAD)**: Avaliar e priorizar as ameaças identificadas com base em critérios quantitativos, como impacto, probabilidade e facilidade de exploração.

5. **Mitigação de Ameaças**: Propor contramedidas viáveis para reduzir ou eliminar os riscos mais críticos identificados nas etapas anteriores.

6. **Revisão e Atualização**: Refletir sobre a efetividade das medidas implementadas e identificar possíveis evoluções futuras do modelo à medida que o sistema for expandido.

As subseções a seguir detalham a aplicação prática dessas etapas com base no escopo definido, apresentando os diagramas arquiteturais (via C4 Model), as ameaças identificadas e as contramedidas adotadas.

4.1.1 Camadas e Componentes do Sistema

Para ilustrar as etapas de desenvolvimento e facilitar a compreensão das metodologias STRIDE e DREAD, foi criado um cenário de monitoramento de ambiente IoT. Esse cenário envolve a coleta e transmissão de dados de sensores ambientais (temperatura, umidade do ar e do solo) para um servidor de borda (gateway), utilizando comunicação via MQTT sobre Wi-Fi com TLS.

O foco da análise de segurança está exclusivamente nos módulos embarcados (ESP32 e sensores) e na comunicação segura com o servidor de borda. A seguir, são listados os componentes relevantes para esse escopo. Outros elementos presentes na arquitetura do sistema completo são omitidos nesta análise, pois não foram considerados na modelagem de ameaças.

Tabela 3 – Sensores, Atuadores e Componentes de Apoio Utilizados nos Módulos IoT

ID	Recurso	Componente	Funcionalidade
1	Sensor e Atuador	Sensor de temperatura e umidade do ar (DHT22)	Monitoramento climático local (temperatura e umidade relativa do ar). Presente em ambos os módulos.
2	Sensor	Sensor capacitivo de umidade do solo (Capacitive Soil Moisture Sensor v2.0)	Medição da umidade do solo para controle da irrigação. Utilizado no módulo sensor.
3	Atuador	Sensor de fluxo de água (YF-S201)	Detecção do fluxo no sistema hidráulico para monitoramento e detecção de falhas. Presente no módulo atuador.
4	Atuador	Sensor de corrente elétrica (ACS712)	Monitoramento do consumo elétrico dos atuadores (ex.: motobomba, válvula) para detecção de funcionamento e anomalias. Presente no módulo atuador.
5	Atuador	Relé de estado sólido (SSR Fotek SSR-40DA)	Atuador para acionamento de cargas elétricas (motores, válvulas, etc.) no módulo atuador.
6	Atuador	Válvula solenoide (Emicol 320300)	Controle do fluxo de água no sistema hidráulico, simulando o acionamento de motobombas ou válvulas. Presente no módulo atuador.
7	Microcontrolador	ESP32 S3	Microcontrolador dos módulos sensor e atuador.
8	Software do Microcontrolador	Plataforma Arduino	Firmware responsável pela lógica dos módulos sensor e atuador.
9	Energia	Bateria Li-Ion 18650 (3.7V 2200–3400mAh)	Armazenamento de energia para operação autônoma dos módulos IoT.
10	Energia	Módulo de gerenciamento de bateria (18650 Battery Shield)	Responsável por carregar a bateria, proteger contra sobrecarga/descarga e fornecer alimentação primária ao sistema.
11	Energia	Regulador de Tensão Step-Up (MT3608)	Elevação da tensão da bateria (3.7V) para 5V, garantindo operação estável dos componentes eletrônicos.
12	Energia	Painel Solar Fotovoltaico (10W, 5V)	Geração de energia elétrica para recarga da bateria e operação autônoma dos módulos IoT em campo.
13	Comunicação	Ponto de Acesso Wi-Fi (2.4GHz)	Estabelece rede local sem fio entre os módulos IoT e o servidor de borda.

A estrutura geral desses componentes é posteriormente representada visualmente por meio da modelagem C4, na Seção 4.1.5, facilitando a identificação dos contêineres envolvidos e suas fronteiras de segurança, que incluem:

- **Dispositivos**: Os dispositivos são responsáveis por coletar dados ambientais, processá-los localmente e transmiti-los para o *gateway*. A Tabela 3 detalha os sensores, atuadores, microcontroladores, componentes de alimentação e comunicação utilizados nos módulos IoT.

- **Comunicação**: A comunicação entre os módulos IoT e o servidor de borda ocorre via Wi-Fi, utilizando o protocolo MQTT com suporte a TLS para garantir a confidencialidade dos dados transmitidos. A Tabela 4 descreve os componentes utilizados neste processo.

- ***Gateway***: O gateway (servidor de borda) é responsável por receber e processar as mensagens enviadas pelos módulos IoT. No escopo desta dissertação, considera-se apenas os componentes que participam diretamente da comunicação segura com os dispositivos. A Tabela 5 lista esses componentes.

Tabela 4 – Componentes de Comunicação

ID	Recurso	Componente	Funcionalidade
1	Protocolo de Comunicação	Wi-Fi (2.4GHz)	Protocolo sem fio utilizado pelos dispositivos para enviar dados ao servidor de borda.
2	Infraestrutura de Rede	Ponto de Acesso Wi-Fi	Estabelece a rede local sem fio, permitindo a comunicação entre os módulos IoT e o servidor de borda.
3	Middleware de Mensagens	NanoMQ (Broker MQTT)	Gerencia a troca de mensagens via protocolo MQTT entre os dispositivos e o servidor.
4	Biblioteca Cliente MQTT	Paho MQTT (Python)	Implementa o cliente MQTT no servidor de borda para receber e processar as mensagens dos dispositivos.

Tabela 5 – Componentes Relevantes do Gateway (Servidor de Borda)

ID	Recurso	Componente	Funcionalidade
1	Hardware do Gateway	Orange Pi 3B	Dispositivo que recebe os dados dos sensores via MQTT.
2	Sistema Operacional	Orange Pi OS (Debian)	Sistema Linux leve para execução dos serviços embarcados.
3	Broker MQTT	NanoMQ	Recebe os dados publicados pelos dispositivos IoT via MQTT com TLS.
4	Cliente MQTT	Paho MQTT (Python)	Subscreve aos tópicos MQTT para análise e armazenamento dos dados.
5	Alimentação	Fonte 5V 3A + No-Break (UPS)	Garante energia contínua e estabilidade durante quedas de rede elétrica.
6	Proteção Física	Gabinete Acrílico Ventilado	Protege fisicamente o hardware do gateway.

Tabela 6 – Dependências Externas Relevantes à Modelagem de Ameaças

ID	Recurso/Dependência	Pontos de Atenção
1	Sensores	Possibilidade de manipulação física ou lógica das leituras por agentes mal-intencionados.
2	Microcontrolador ESP32 S3	Controle de interfaces, exposição de GPIOs e uso indevido de comandos por falhas de firmware.
3	Software do Microcontrolador (Arduino)	Risco de *spoofing*, falta de verificação de integridade e firmware desprotegido.
4	Protocolo de Comunicação (Wi-Fi/MQTT)	Sujeito a interceptação de dados, ataques de negação de serviço (DoS), *jamming* e manipulação de mensagens.
5	Gateway (Orange Pi 3B)	Controle das portas de entrada de dados e segurança na comunicação TLS com o ESP32.
6	Sistema Operacional do Gateway (Orange Pi OS)	Gerenciamento de atualizações de segurança e autenticação básica no acesso ao broker MQTT.
7	Broker MQTT (NanoMQ)	Autenticação, criptografia, controle de tópicos e proteção contra injeção de mensagens MQTT.
8	Biblioteca Cliente MQTT (Paho)	Necessidade de atualização periódica e verificação contra vulnerabilidades conhecidas em bibliotecas de terceiros.
9	Fonte de Alimentação e No-Break	Falhas no fornecimento comprometem a disponibilidade do servidor de borda.

4.1.2 Dependências Externas

Para o correto funcionamento do sistema, algumas dependências externas precisam ser gerenciadas, especialmente aquelas relacionadas aos dispositivos embarcados e à comunicação com o servidor de borda. Embora permaneçam sob o controle da organização, essas dependências nem sempre estão sob o controle direto da equipe de desenvolvimento.

A Tabela 6 apresenta as principais dependências relevantes ao escopo desta análise, bem como os pontos de atenção associados.

Tabela 7 – Pontos de Entrada

ID	Recurso	Ponto de Entrada	Nível de Confiança
1	Sensor	Manipulação das leituras por agentes físicos ou substituição por dispositivos falsos	Baixo
2	ESP32 S3	Interfaces GPIO, UART, ou porta serial expostas fisicamente	Baixo
3	Firmware Arduino	Possibilidade de spoofing via rede ou comandos forjados	Médio
4	Comunicação Wi-Fi	Injeção de pacotes, ataques DDoS, jamming ou Man-in-the-Middle	Médio
5	Broker MQTT (NanoMQ)	Publicações não autorizadas, falta de autenticação forte ou uso de tópicos não controlados	Médio
6	Orange Pi (gateway)	Porta Ethernet exposta, acesso físico ao hardware ou acesso ao sistema operacional	Alto

Tabela 8 – Pontos de Saída

ID	Recurso	Ponto de Saída
1	Broker MQTT (NanoMQ)	Dados publicados nos tópicos MQTT pelos sensores (umidade, temperatura, fluxo, tensão)
2	Orange Pi	Armazenamento local temporário ou exibição via terminal/logs internos da aplicação

4.1.3 Pontos de Entrada e Saída

Os pontos de entrada e saída do sistema representam interfaces críticas que precisam ser monitoradas e protegidas para garantir a segurança da solução. **Pontos de entrada** são interfaces físicas ou lógicas pelas quais comandos e dados podem ingressar no sistema, como sensores, protocolos de rede ou conexões físicas. Já os **pontos de saída** correspondem aos canais pelos quais os dados produzidos ou processados são enviados a outros sistemas, como tópicos MQTT ou registros locais. Mesmo que os pontos de saída não sejam diretamente exploráveis como vetores de ataque, eles podem expor informações sensíveis que auxiliam em ataques futuros, como dados de sensores, endereços IP e estrutura dos tópicos MQTT. A Tabela 7 e Tabela 8 listam apenas os pontos de entrada e saída relevantes ao escopo desta dissertação.

4.1.4 Ativos e Níveis de Confiança

Os ativos do sistema são os componentes físicos e lógicos que precisam ser protegidos contra acessos indevidos, modificação não autorizada ou interceptação. No contexto desta dissertação, foram considerados como ativos os sensores, microcontroladores ESP32, firmware embarcado, comunicação via MQTT e o broker NanoMQ. Esses ativos foram selecionados com base na análise STRIDE/DREAD aplicada especificamente aos módulos IoT e ao canal de comunicação com o servidor de borda. A Tabela 9 apresenta os ativos relevantes, suas descrições e os níveis de confiança associados. A Tabela 10 define os níveis de confiança considerados.

Tabela 9 – Ativos do Sistema e seus Níveis de Confiança

ID	Ativo	Descrição	Nível de Confiança
1	Sensores	Dispositivos analógicos conectados ao ESP32 para coleta de dados ambientais (solo, ar, fluxo, tensão)	3, 4, 5, 8
2	ESP32 S3	Microcontrolador que executa a lógica de coleta e envio de dados para o gateway	3, 4, 5, 8
3	Firmware do ESP32	Código embarcado no dispositivo, desenvolvido em Arduino IDE	2, 3, 8
4	Comunicação Wi-Fi + MQTT	Canal de comunicação sem fio entre o ESP32 e o broker NanoMQ	2, 3
5	Broker MQTT (NanoMQ)	Serviço que recebe as mensagens publicadas pelo ESP32	2, 6
6	Orange Pi (gateway)	Hardware que hospeda o broker e sistema operacional base	2, 6

Tabela 10 – Níveis de Confiança

ID	Entidade	Descrição
2	Desenvolvedor	Responsável pelo firmware e configuração dos dispositivos e do broker
3	Instalador/Técnico de Campo	Realiza a instalação física do sistema, com acesso parcial aos dispositivos
4	Atacante Físico Local	Indivíduo com acesso físico aos sensores e ao ESP32 em campo
5	Atacante Remoto na Rede	Agente malicioso com acesso à rede Wi-Fi onde estão os dispositivos IoT
6	Processo do Broker	Componente de software que gerencia as mensagens publicadas pelo ESP32
8	Sistema Embarcado no ESP32	Ambiente de execução do firmware, incluindo acesso ao barramento de sensores e memória flash

4.1.5 Modelagem da Arquitetura com C4 Model

A arquitetura do sistema desenvolvido na empresa Tulkas foi modelada utilizando o C4 Model, especificamente no nível de containers. Esse modelo fornece uma representação clara das tecnologias utilizadas, das fronteiras de confiança e da comunicação entre os principais blocos funcionais que compõem a solução. A solução geral é composta por três grandes domínios: os dispositivos IoT (ESP32-S3), o servidor de borda (Orange Pi) e o servidor em nuvem (IaaS). Cada domínio é subdividido em containers que representam instâncias de software isoladas, com responsabilidades bem definidas.

No servidor de borda, destacam-se os seguintes containers:

- **NanoMQ** – Broker MQTT utilizado para comunicação com os dispositivos IoT via tópicos de leitura e comando;

- **Django App (Borda)** – Backend local acessado via HTTPS LAN, responsável por manter uma interface mesmo em caso de falha de conexão com a nuvem;

- **SQLite** – Banco de dados leve, embarcado na instância da borda, utilizado para persistência local das leituras;

- **Celery Worker (MQTT)** – Responsável por consumir tópicos MQTT e processar comandos sensoriais;
- **RabbitMQ** e **Redis** – Utilizados para mensageria assíncrona e armazenamento temporário de resultados de tarefas.

No servidor em nuvem, os containers refletem a versão escalável e redundante da mesma estrutura:

- **Django App (Nuvem)** – Responsável pela interface principal acessada via navegador pelo produtor rural;
- **PostgreSQL** – Banco de dados robusto para armazenamento completo dos dados do sistema;
- **Celery Worker (AMQP)**, **RabbitMQ** e **Redis** – Responsáveis pela orquestração assíncrona, enfileiramento de tarefas e manipulação de resultados, replicando a lógica da borda;
- **Nginx** – Servidor proxy reverso, que encaminha requisições HTTP para os aplicativos Django.

Os dispositivos ESP32-S3 (módulo sensor e módulo atuador) comunicam-se diretamente com o container NanoMQ via protocolo MQTT sobre Wi-Fi. O módulo sensor envia leituras de temperatura, umidade do solo e do ar, enquanto o módulo atuador realiza comandos de controle, como acionamento de relés, e também envia medições (fluxo, corrente elétrica, etc.).

Embora o C4 Model represente toda a arquitetura da solução desenvolvida, a análise de segurança com as metodologias STRIDE e DREAD foi aplicada exclusivamente ao subsistema composto pelos dispositivos ESP32, sensores conectados e comunicação com o servidor de borda via MQTT sobre TLS. Essa delimitação foi adotada por questões de escopo e viabilidade, concentrando os testes e a modelagem de ameaças na camada mais crítica e vulnerável do sistema: a coleta e transmissão de dados sensoriais em campo. Os demais containers e domínios do sistema são apresentados no diagrama da Figura 9, para fins de contextualização da solução como um todo.

4.2 Aplicação da Metodologia STRIDE

Com base no cenário descrito anteriormente (Seção 4.1.1), cada componente do sistema foi classificado de acordo com sua função e camada (hardware, software ou comunicação). A análise considerou pontos críticos do sistema à luz da metodologia STRIDE — *Spoofing*, *Tampering*, *Repudiation*, *Information Disclosure*, *Denial of Service* e *Elevation of Privilege*.

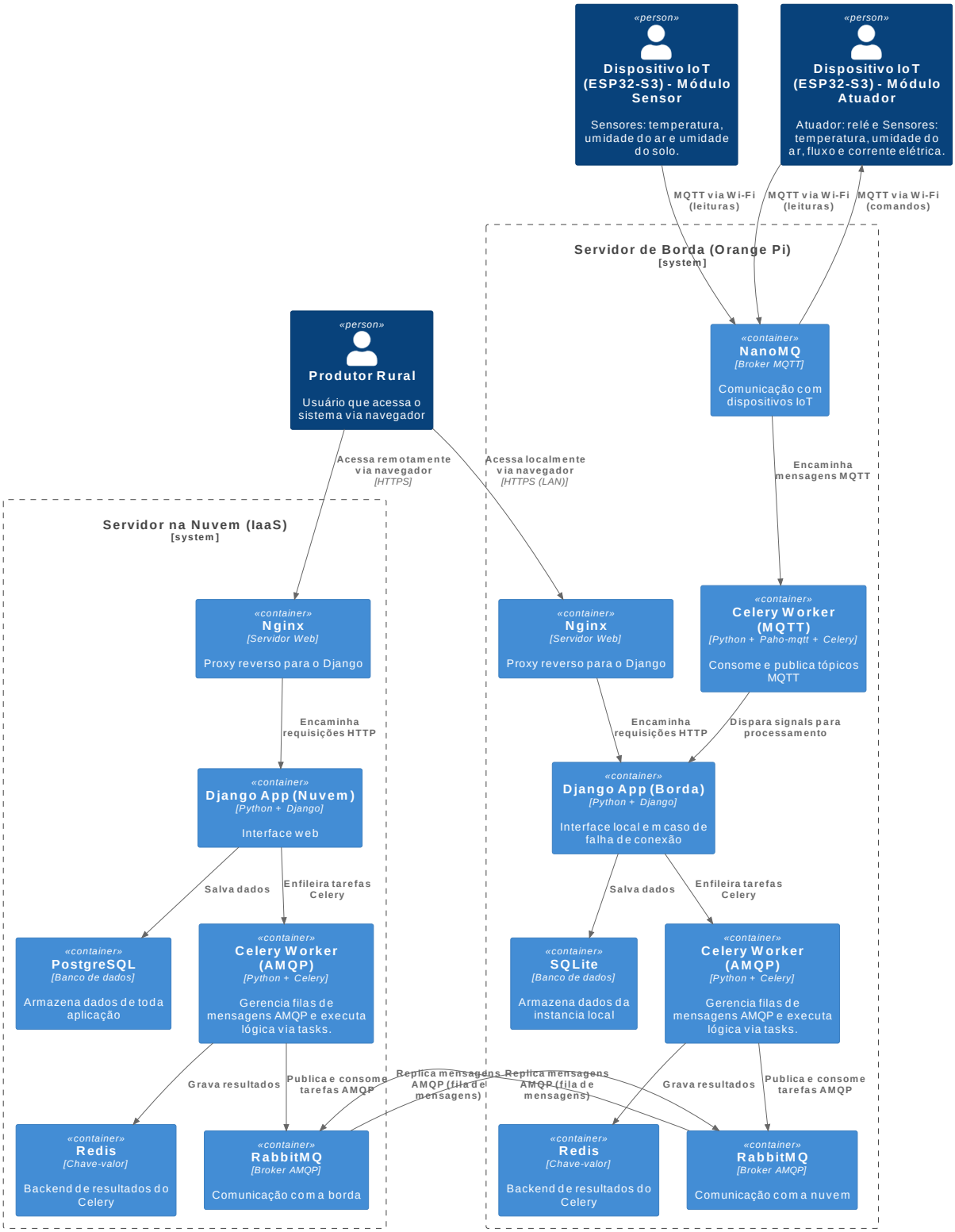

Figura 9 – Diagrama de contêineres da solução proposta

Tabela 11 – Lista de Ameaças STRIDE (Apenas Elementos Testados)

Tipo de Ameaça	**Pontos de Entrada / Ativos Afetados**	**Controles de Segurança Recomendados**
Spoofing	Leituras falsas nos sensores (simuladas com potenciômetro), Publicação de pacotes MQTT falsos sem autenticação, Spoofing via Wi-Fi (pacotes maliciosos)	Certificados digitais para autenticação mútua, validação da origem dos dados no broker, filtragem por tópicos e identificação do dispositivo
Tampering	Alteração dos valores sensoriais via falha analógica (injeção), Dump de firmware via UART, Tentativas de modificação física do ESP32	Flash Encryption, bloqueio de UART, controle físico de acesso, Secure Boot (recomendado para trabalhos futuros)
Information Disclosure	Interceptação de dados MQTT sem TLS (via tcpdump), Leitura do conteúdo da memória flash (sem criptografia), Exposição dos dados publicados em tópicos MQTT abertos	Uso de MQTT com TLS, ativação de Flash Encryption, controle de acesso aos tópicos do broker
Denial of Service	Flood de mensagens MQTT geradas por dispositivos falsos ou comprometidos, Simulação de saturação por publicação frequente	Rate limiting no broker MQTT, verificação da integridade das mensagens, bloqueio de publicações fora de padrão, criação de tópicos reservados seguros
Elevation of Privilege	Modificação do firmware para burlar validações, Bypass da verificação de boot por falha física no circuito (glitch no reset)	Implementação de Secure Boot, assinatura do firmware, proteção física dos pontos de reset, queima de eFuses para travamento do bootloader

A aplicação foi restrita ao subsistema composto pelos dispositivos ESP32-S3, sensores conectados e comunicação com o servidor de borda via protocolo MQTT sobre TLS, conforme definido no escopo. A modelagem de ameaças levou em consideração:

- Os ativos identificados (Tabela 9);
- Os pontos de entrada e saída do sistema (Tabelas 7 e 8);
- O fluxo de dados representado no C4 Model (Figura 9);
- E os testes de segurança realizados em laboratório.

A metodologia STRIDE permitiu classificar e mapear as ameaças de forma sistemática, possibilitando a proposição de contramedidas viáveis para o contexto rural e embarcado da aplicação. A Tabela 11 apresenta apenas as ameaças que foram efetivamente testadas e analisadas com base nos experimentos realizados no sistema embarcado (ESP32 e sensores) e na comunicação com o servidor de borda via MQTT. Elementos não testados foram excluídos desta versão para manter alinhamento com o escopo prático.

4.3 Análise e Classificação de Ameaças com DREAD

A análise DREAD foi utilizada para priorizar as ameaças identificadas no subsistema composto pelos dispositivos ESP32-S3, sensores conectados e a comunicação com o servidor de borda via protocolo MQTT sobre TLS. O método DREAD avalia cada ameaça com base nos seguintes critérios:

- **Damage Potential (Potencial de Dano)**: impacto da exploração da ameaça.

Tabela 12 – Análise e Classificação de Ameaças com DREAD (Apenas Elementos Testados)

Ameaça aos Recursos	**D**	**R**	**E**	**A**	**D**	**Total**
Interceptação de dados MQTT sem TLS	9	9	8	9	9	**44**
Publicação de mensagens falsas no broker	8	9	9	9	8	**43**
Leitura falsa nos sensores (spoofing físico com potenciômetro)	8	8	8	9	8	**41**
Dump de firmware via UART (sem proteção)	9	7	7	8	6	**37**
Ataque por flood MQTT (DoS)	8	7	6	8	7	**36**
Glitch físico para burlar boot/reset	9	6	5	7	6	**33**
Modificação de firmware sem Secure Boot	8	6	5	7	6	**32**
Acesso físico ao ESP32 e sensores	7	5	5	6	6	**29**

- **Reproducibility (Reprodutibilidade)**: facilidade com que o ataque pode ser repetido com sucesso.
- **Exploitability (Explorabilidade)**: dificuldade para explorar a vulnerabilidade.
- **Affected Users (Usuários Afetados)**: quantidade e criticidade dos usuários ou sistemas impactados.
- **Discoverability (Detectabilidade)**: facilidade de descoberta da vulnerabilidade por atacantes.

Cada critério é pontuado de 0 a 10. A soma total define a prioridade de mitigação. A Tabela 12 apresenta somente as ameaças efetivamente testadas e analisadas nos experimentos realizados nesta dissertação.

4.4 Contramedidas e Mitigação

Com base nas análises realizadas pelas metodologias STRIDE e DREAD — e nos testes de segurança realizados na prática, a Tabela 13 apresenta as principais vulnerabilidades identificadas no subsistema composto por ESP32, sensores e a comunicação com o servidor de borda via MQTT. Foram mantidas na tabela apenas as ameaças diretamente observadas ou simuladas durante os experimentos. Cada vulnerabilidade é acompanhada das técnicas de mitigação aplicadas ou recomendadas, bem como sua situação atual quanto à cobertura de segurança:

- **Ameaças Não Mitigadas**: ainda sem contramedidas eficazes, representando alto risco caso exploradas.
- **Ameaças Parcialmente Mitigadas**: mitigadas em parte, mas que ainda demandam reforço.
- **Ameaças Totalmente Mitigadas**: com controles eficazes no escopo avaliado.

Tabela 13 – Contramedidas e Mitigação (Somente Elementos Testados)

Vulnerabilidade	Ameaças Associadas	Técnicas de Mitigação	Status
Comunicação MQTT/Wi-Fi	Spoofing, Divulgação de dados, DoS, Injeção de pacotes	TLS ativo, autenticação no broker, segregação de tópicos, rate limiting	Parcialmente Mitigada
Sensores (Leituras)	Spoofing físico, adulteração, excesso de dados	Verificação de consistência, comparação entre sensores, filtros por faixa de valor	Parcialmente Mitigada
ESP32 (Firmware)	Dump de firmware, modificação, engenharia reversa	Flash Encryption, proteção da UART, recomendação de Secure Boot	Parcialmente Mitigada
ESP32 (Hardware)	Extração via acesso físico, interferência no boot	Queima de eFuses (em fase final), encapsulamento, reforço físico (protetores, gabinete)	Parcialmente Mitigada
Broker MQTT (NanoMQ)	Injeção de mensagens, tópicos sem controle	Autenticação, ACLs, segregação por tópicos e logs de publicações	Parcialmente Mitigada
Ataque de Glitch (Reset)	Bypass de proteção por falha analógica	Reforço de circuito de alimentação e reset, proteção física de trilhas sensíveis	Parcialmente Mitigada
Flood MQTT	Negação de serviço por excesso de publicações	Monitoramento de frequência, bloqueio temporário de dispositivos ofensores	Parcialmente Mitigada

4.5 Considerações sobre o Capítulo

A partir da aplicação das metodologias STRIDE e DREAD, foram identificadas e classificadas diversas ameaças que afetam o subsistema composto por ESP32, sensores conectados e comunicação com o servidor de borda via MQTT. As tabelas anteriores apresentaram os pontos de vulnerabilidade, os controles de segurança recomendados e o nível de prioridade de mitigação.

Com base nessas análises, foi possível selecionar um conjunto de ameaças críticas que representam alto risco para a integridade, disponibilidade e confiabilidade da solução embarcada. Essas ameaças serviram de base para a definição dos testes de segurança documentados no Capítulo 5, os quais buscaram validar na prática a exploração de vulnerabilidades como:

- Interceptação de dados sensoriais via MQTT sem TLS;
- Falsificação de leituras por manipulação física dos sensores;
- Dump de firmware via UART sem Flash Encryption;
- Injeção de falhas físicas (glitch) para interferência no processo de boot;
- Ataques por flooding de mensagens MQTT.

Esses experimentos práticos visam verificar a viabilidade real de exploração das ameaças mapeadas e avaliar a efetividade das contramedidas propostas. No próximo capítulo, são descritos os procedimentos adotados, os ambientes utilizados e os resultados obtidos durante os testes de vulnerabilidade.

5 TESTES E VALIDAÇÃO

Este capítulo apresenta os testes de segurança realizados sobre a arquitetura de irrigação automatizada proposta. Os testes foram conduzidos com base na modelagem de ameaças elaborada no Capítulo 4, que utilizou as metodologias STRIDE e DREAD como referência. O objetivo é validar as medidas de mitigação propostas e avaliar a robustez do sistema frente a ataques físicos, lógicos e de comunicação. Os testes foram organizados por categorias de ameaça e por componente do sistema, visando uma análise abrangente e prática da segurança implementada.

A coleta e transmissão de dados sensoriais, embora pareça inofensiva à primeira vista, pode expor o sistema a diversos riscos quando não protegida adequadamente. Informações como umidade do solo, fluxo de água e tensão da bateria — se interceptadas, manipuladas ou falsificadas — podem comprometer diretamente a tomada de decisão automatizada, ocasionando falhas graves no fornecimento de água, desperdício de recursos e prejuízos à produção agrícola. Além disso, um atacante pode explorar essas informações para inferir o funcionamento interno do sistema, identificar padrões operacionais e conduzir ataques mais elaborados, como sabotagens ou negação de serviço.

Mesmo em sistemas de pequeno porte e operando em ambientes isolados, como propriedades rurais, a adoção de camadas de segurança se mostra essencial. Isso se deve à crescente conectividade dos dispositivos, ao uso de protocolos amplamente conhecidos (como Wi-Fi e MQTT) e à possibilidade de acesso físico aos equipamentos embarcados. A ausência de proteção adequada expõe esses dispositivos a ataques triviais — como spoofing de sensores e leitura de firmware via UART — que poderiam ser mitigados com medidas relativamente simples, como criptografia, autenticação forte e uso de boot seguro. Portanto, a segurança não deve ser tratada como um recurso opcional, mas sim como um requisito fundamental desde as fases iniciais de projeto.

A segurança é um pilar fundamental no desenvolvimento de sistemas computacionais, especialmente em contextos distribuídos e sensíveis como a Internet das Coisas (IoT). Um sistema seguro deve considerar elementos centrais como a **confidencialidade** (garantia de que informações sejam acessadas apenas por quem tem permissão), **integridade** (assegurar que os dados não sejam alterados indevidamente), **disponibilidade** (garantir acesso contínuo aos recursos), **autenticidade** (verificação da identidade das partes envolvidas), **confiabilidade** (garantia de funcionamento correto e previsível mesmo diante de falhas) e **responsabilidade** (possibilidade de rastrear ações realizadas no sistema e atribuí-las aos seus autores) (PFLEEGER; PFLEEGER; MARGULIES, 2015).

A partir dessa base conceitual, este capítulo apresenta uma análise da segurança em sistemas distribuídos, conforme discutido por autores clássicos como Tanenbaum (TANENBAUM; STEEN, 2024), que destaca a complexidade de proteger serviços e dados diante de ameaças como divulgação indevida, alteração maliciosa e negação de uso. O autor enfatiza que, além da autenticação e autorização, é fundamental considerar a integridade

das operações, a privacidade dos dados, o design seguro em múltiplas camadas e o uso de mecanismos como criptografia, controle de acesso, auditoria e monitoramento contínuo.

Esses conceitos fornecem a base para o aprofundamento nas práticas contemporâneas de segurança voltadas a dispositivos IoT, como as preconizadas pela OWASP[1]. Na sequência, são exploradas as diretrizes dessa organização, que categorizam as vulnerabilidades mais críticas em sistemas embarcados e propõem boas práticas para mitigação. Por fim, são apresentados os cenários de invasão aplicados ao sistema desenvolvido, incluindo a modelagem das ameaças, a análise de risco com base no CVSS v3.1 (*Common Vulnerability Scoring System*) e a execução de testes práticos que simulam essas ameaças, a fim de validar a robustez da arquitetura proposta. Cada teste é documentado com seus métodos, resultados e potenciais implicações para a segurança do sistema.

5.1 Cenários de Ameaça por Componente

Esta seção apresenta os principais cenários de ameaça associados aos componentes físicos e embarcados que foram efetivamente analisados e testados no sistema de irrigação automatizada desenvolvido. Cada elemento — do sensor ao microcontrolador, passando pelo circuito de alimentação e o gateway — possui características próprias de exposição a riscos físicos e lógicos. Assim, é necessário avaliar individualmente suas vulnerabilidades, impactos e possíveis contramedidas.

Esses cenários complementam a modelagem STRIDE e a priorização DREAD discutidas no Capítulo 4, mas agora com foco prático, baseando-se nos testes descritos neste capítulo. Cada análise inclui o tipo de ataque, sua classificação CVSS v3.1 e recomendações de mitigação. Cabe destacar que apenas os componentes testados na prática estão incluídos nesta seção. Elementos como o backend web (Django) e a infraestrutura de nuvem (Docker, Redis, RabbitMQ) não foram alvo de testes nesta fase do projeto e, portanto, foram excluídos da análise prática de vulnerabilidades.

5.1.1 Sensor de Temperatura e Umidade (DHT22)

Por ser um sensor digital simples, o DHT22 é suscetível a manipulações físicas (remoção ou cobertura) e a falhas causadas por ruído na linha de dados. Ataques físicos simples podem induzir medições falsas, afetando a lógica de irrigação.
Vulnerabilidades: Acesso físico, ruído eletromagnético, falsificação por aquecimento.
Mitigações: Alojamento protegido, verificação por sensores redundantes, detecção de outliers.
CVSS: 4.6 — AV:P/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:H

[1] OWASP IoT Top 10, disponível em: <https://owasp.org/www-project-internet-of-things/>

5.1.2 Sensor de Fluxo de Água

Como se baseia em impulsos magnéticos, pode ser interferido por campos externos ou indução. Também pode sofrer de DoS físico (bloqueio mecânico).
Vulnerabilidades: Interferência magnética, obstrução física.
Mitigações: Alojamento, checagem cruzada com válvula e tempo estimado.
CVSS: 5.1 — AV:P/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:H

5.1.3 Sensor de Corrente (ACS712)

Por usar detecção Hall, pode ser afetado por campos magnéticos externos ou manipulação de carga falsa.
Vulnerabilidades: Indução externa, sabotagem com cargas falsas.
Mitigações: Blindagem, checagem de comportamento fora do padrão, limites mínimos e máximos.
CVSS: 4.9 — AV:P/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:L

5.1.4 Sensor Capacitivo de Umidade do Solo

Sensível à interferência eletromagnética e à degradação em ambientes úmidos.
Vulnerabilidades: Ruído na leitura, falsos positivos, manipulação física (remoção).
Mitigações: Leitura média, isolamento do sensor, checagem periódica de falhas.
CVSS: 5.0 — AV:P/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:H

5.1.5 Painel Solar + Circuito de Alimentação (MT3608 + Bateria 18650)

Vulnerável a sabotagem física e variações bruscas de tensão (glitching) que podem forçar resets ou comportamento instável.
Vulnerabilidades: Brownout, glitching físico, sabotagem da entrada.
Mitigações: Capacitores estabilizadores, watchdog no firmware, proteção contra sobrecarga.
CVSS: 5.3 — AV:P/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

5.1.6 ESP32-S3

Principal alvo de ameaças: boot inseguro, acesso físico à UART, firmware sem proteção, sensores manipuláveis fisicamente, Wi-Fi sem criptografia.
Vulnerabilidades: spoofing, fuzzing, glitching, leitura da flash via UART, substituição de firmware sem autenticação, injeção de falhas analógicas, interceptação de pacotes MQTT.
Mitigações: boot seguro (Secure Boot V2), queima de eFuses, criptografia da flash (Flash Encryption), UART desativado, MQTT sobre TLS, validação do firmware por as-

sinatura digital.
CVSS: 8.8 — AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H

5.1.7 Módulo de Atuação (Relé + Válvula Solenoide)

Podem ser ativados indevidamente se o controle via GPIO for comprometido ou mal validado no firmware.
Vulnerabilidades: Trigger externo via GPIO exposto, falha lógica no código embarcado.
Mitigações: Buffer de isolamento, validação de comandos antes da ativação, lógica redundante de controle.
CVSS: 6.5 — AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:H/A:H

5.1.8 Servidor de Borda (Orange Pi 3B)

Durante os testes de segurança, o Orange Pi foi alvo de intercepções via rede utilizando a ferramenta `tcpdump`. Essas capturas permitiram observar pacotes MQTT trafegando em texto claro antes da ativação do TLS, evidenciando uma vulnerabilidade de *information disclosure*. Também foram simulados envios de pacotes forjados, caracterizando um risco de *tampering* no broker. Por fim, verificou-se a possibilidade de ataque de *denial of service* (DoS) com sobrecarga de mensagens MQTT.

Mitigações recomendadas:

- Ativação obrigatória de TLS no broker (porta 8883);
- Autenticação com usuário e senha;
- Configuração de ACLs por tópico;
- Monitoramento de logs e pacotes anômalos;
- Restrição física ao acesso à Ethernet, USB e GPIOs do Orange Pi.

CVSS estimado: 7.4 — AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:H/A:H

5.1.9 Broker MQTT

Durante os testes, foi possível forjar pacotes MQTT e publicar mensagens falsas nos tópicos do sistema, inclusive simulando sensores inexistentes. A ausência inicial de autenticação e TLS permitiu ataques de *Spoofing* e *Denial of Service*.

Mitigações aplicadas:

- Habilitação de TLS com certificado digital;
- Autenticação mútua com usuário e senha;
- ACLs por tópico;

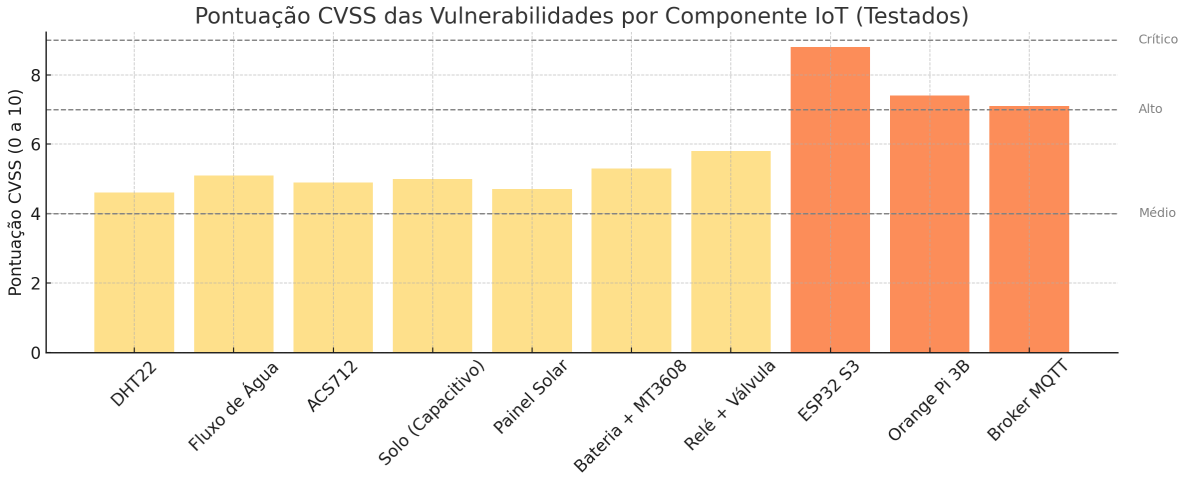

Figura 10 – Pontuação CVSS das Vulnerabilidades por Componente IoT

- Configuração de rate limiting;
- Logs de atividade e alertas de comportamento suspeito.

CVSS estimado: 7.1 — AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:H/A:H

5.1.10 Pontuação CVSS das Vulnerabilidades por Componente IoT

A Figura 10 apresenta a pontuação CVSS associada às vulnerabilidades identificadas individualmente em cada componente do sistema, considerando apenas os dispositivos efetivamente testados nesta dissertação. Isso inclui sensores físicos, circuitos de alimentação, microcontrolador ESP32-S3, gateway (Orange Pi) e o broker MQTT.

Observa-se que o ESP32 S3 permanece como o componente com maior pontuação de risco, devido ao seu papel central e à exposição tanto física quanto lógica. Em seguida, aparecem o gateway de borda e o broker MQTT, que foram alvos de testes relacionados à interceptação, publicação de pacotes forjados e ataques por negação de serviço. Essa visualização reforça a importância de aplicar medidas de segurança tanto nos dispositivos embarcados quanto nos elementos intermediários da arquitetura de IoT distribuída.

5.2 Análise Sistêmica das Interações

A integração dos sensores com o ESP32, comunicação MQTT e controle final por meio da Orange Pi configura um ambiente de múltiplos pontos de ataque. A ausência inicial de criptografia entre ESP32 e broker permitiu interceptações de dados sensoriais, e ataques físicos aos sensores mostraram-se capazes de induzir decisões incorretas de irrigação. Esta seção detalha ameaças compostas que emergem da interação entre diferentes componentes do sistema testado. Os cenários foram modelados com base na análise STRIDE e priorizados segundo a metodologia DREAD. Cada ameaça foi associada a um

escopo testado, incluindo manipulação física, injeção de leituras falsas, interceptação de pacotes MQTT e sabotagem por falha elétrica.

5.2.1 Ataques Compostos e Riscos Sistêmicos

1. Replay Attack na Comunicação MQTT

Com a ausência de criptografia e autenticação robusta, um invasor pode capturar pacotes MQTT válidos e reenviá-los, forçando comandos indevidos no sistema de irrigação.

CVSS estimado: 7.4 — AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:H/A:L

2. Injeção de Dados Falsos via Sensores (Sensor Spoofing)

Ao manipular fisicamente os sensores (umidade do solo, temperatura, fluxo), um atacante pode induzir o sistema a irrigar em excesso ou suspender a irrigação incorretamente.

CVSS estimado: 6.3 — AV:P/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:L

3. Sabotagem da Fonte de Alimentação

A manipulação da linha de alimentação (painel solar, MT3608, módulo de carga) pode ser usada para induzir falhas no boot do ESP32, resetar o sistema ou provocar estados inconsistentes.

CVSS estimado: 5.9 — AV:P/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

5.2.2 Avaliação Consolidada de Risco

Com base nos cenários descritos, a arquitetura apresenta os seguintes níveis de risco sistêmico:

- **Alto Risco (CVSS >= 7.0):** Comunicação MQTT não segura (possibilidade de replay e interceptação de pacotes). - **Médio Risco (CVSS 5.0 a 6.9):** Sensor spoofing físico e sabotagem da alimentação. - **Baixo Risco (CVSS < 5.0):** Não identificado entre os cenários testados.

5.2.3 Implicações para o Projeto

A análise evidencia a importância de adotar mecanismos de segurança em todos os níveis da arquitetura testada: proteção física dos sensores, comunicação segura com criptografia e autenticação, verificação de integridade do firmware e monitoramento da alimentação elétrica para evitar falhas induzidas. Essas práticas minimizam a superfície de ataque e aumentam a resiliência do sistema frente a ameaças reais, alinhando-se às recomendações da OWASP IoT Top 10 e às boas práticas descritas por Tanenbaum (TANENBAUM; STEEN, 2024).

A Figura 11 apresenta os cenários de ataque compostos com maior potencial de comprometimento sistêmico dentre os que foram efetivamente testados. Mesmo sem alcançar o nível crítico, esses ataques demonstram como vulnerabilidades físicas e lógicas podem se combinar para aumentar significativamente os riscos operacionais de um sistema

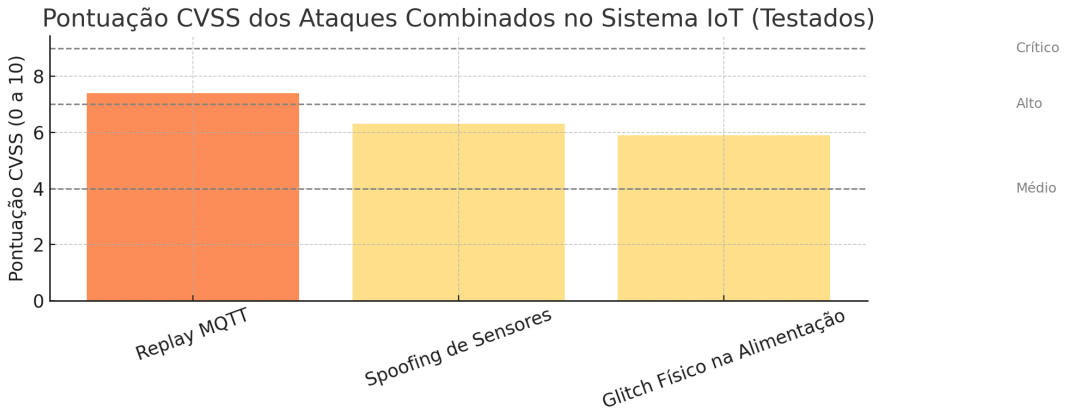

Figura 11 – Pontuação CVSS dos Ataques Combinados no Sistema IoT (Testados)

IoT distribuído. A comunicação MQTT não segura, a manipulação física dos sensores e a sabotagem da alimentação configuram pontos de atenção prioritários no fortalecimento da segurança da solução.

5.3 Dump de Firmware e Recuperação de Informações em Dispositivos ESP32

Este relatório documenta as etapas realizadas para a extração da memória flash de um microcontrolador ESP32 utilizando a ferramenta `esptool.py` em ambiente Windows. O processo foi dividido em blocos de 16KB e permitiu recuperar parte dos dados da flash, com destaque para a identificação de informações sensíveis como credenciais de Wi-Fi.

5.3.1 Resumo da Metodologia

Durante os testes, foram utilizados scripts no formato `.bat` para automatizar a leitura da flash, bloco a bloco, com tentativas de repetição em caso de falhas. Foram também testadas variações de tamanho de bloco e velocidade de comunicação (`baudrate`), porém os melhores resultados foram obtidos com:

- Porta utilizada: `COM8`
- Baudrate: `38400`
- Tamanho de cada bloco: `0x4000` (16KB)
- Total de blocos lidos com sucesso: **62**
- Ferramenta usada: `python -m esptool read_flash`

A partir do bloco 62, a leitura foi consistentemente bloqueada, possivelmente devido a mecanismos de proteção contra leitura no ESP32. Esse comportamento está documentado em artigos técnicos e fóruns especializados[2] como uma funcionalidade de

[2] <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/security/flash-encryption.html>

segurança disponível em versões recentes do chip.

5.3.2 Script de Dump Utilizado

```
python -m esptool --port COM8 --baud 38400 read_flash 0x%OFFSET%
   0x4000 part_%N%.bin
```

5.3.3 Unificação dos Blocos

Os arquivos binários de cada bloco (`part_#.bin`) foram concatenados em um único arquivo:

```
copy /b part_*.bin esp32_dump_consolidado.bin
```

5.3.4 Extração de Strings

Foi utilizada a ferramenta `strings` para extrair trechos legíveis de texto do arquivo final:

```
strings esp32_dump_consolidado.bin > esp32_strings.txt
```

5.3.5 Informações Identificadas

Durante a análise, foram encontrados padrões que indicam dados sensíveis armazenados na flash:

- Endereços IP internos, como `192.168.1.188`
- SSIDs de redes Wi-Fi como `Zoro`, `Pedro`, `Rise`
- Palavras-chave como `senha`, `pass`, `token`, `user`

Estes dados estavam armazenados em texto claro e podem ser recuperados mesmo sem acesso completo à memória flash.

5.3.6 Conclusão

Apesar da extração não ter sido completa — interrompida no bloco 62 devido à falha contínua de leitura —, foi possível recuperar informações sensíveis como nomes de redes Wi-Fi (SSIDs), endereços IP internos e palavras-chave como `senha`, `pass`, `token` e `user`, todas visíveis em texto claro. Foram testados diversos tamanhos de blocos e baudrates para aumentar a taxa de sucesso da extração. O formato de 16KB por bloco foi o que permitiu maior extração, enquanto blocos maiores resultaram em travamentos frequentes. A extração foi interrompida provavelmente por uma medida de segurança do

chip ESP32, como a **criptografia da flash** ou a **proteção por eFuses**, que impedem a leitura de determinadas regiões da memória flash via UART (Espressif Systems, 2024).

Mesmo com essas proteções parcialmente aplicadas, os dados obtidos já revelam pontos críticos da infraestrutura embarcada. Informações como nomes de redes e endereços IP internos facilitam ataques de footprinting, engenharia social e podem orientar ações mais sofisticadas, como spoofing de dispositivos ou ataques por força bruta em ambientes com conectividade restrita. Esse experimento reforça a importância de implementar medidas de proteção embarcada como:

- Criptografia completa da flash com chaves armazenadas em eFuses;
- Habilitação do *Secure Boot* para validação do firmware na inicialização;
- Armazenamento seguro de senhas e tokens criptografados em NVS;
- Desativação da interface UART e de logs em produção;
- Queima de eFuses para impedir regravação ou extração do firmware.

Mesmo quando a leitura total do firmware não é possível, dumps parciais ainda podem revelar informações estratégicas e auxiliar na elaboração de ataques direcionados. Portanto, a aplicação rigorosa das boas práticas de segurança em sistemas embarcados é essencial para proteger dispositivos conectados contra ameaças reais.

5.4 Manipulação de Leituras Analógicas por Injeção de Falhas no ESP32

Além dos testes de extração de firmware e análise de comunicação, também foi conduzido um experimento voltado à identificação de vulnerabilidades físicas relacionadas à integridade dos dados sensoriais recebidos pelo microcontrolador ESP32. O objetivo foi verificar se seria possível manipular artificialmente os valores de leitura dos sensores, induzindo o sistema a tomar decisões incorretas com base em dados falsificados, explorando uma classe de ataques conhecida como *fault injection*.

Para simular essa manipulação, foi conectado um potenciômetro de 10 kΩ entre os terminais de 3,3 V e GND da placa, com o pino central (wiper) ligado diretamente à entrada analógica anteriormente utilizada por um sensor capacitivo de umidade do solo. Essa abordagem permitiu alterar dinamicamente a tensão de entrada lida pelo conversor analógico-digital (ADC) do ESP32, replicando valores válidos de umidade, mesmo sem a presença física do sensor.

A variação da resistência no potenciômetro resultou em tensões dentro da faixa esperada para sensores analógicos (geralmente entre 1,5 V e 2,8 V), levando o microcontrolador a interpretar valores legítimos de umidade do solo. Essa técnica é uma forma de injeção de falhas analógicas (*analog fault injection*), que se encaixa no escopo de ataques

por canal lateral (*side-channel attacks*), ainda que não envolva manipulação de clock ou glitches no sistema de alimentação.

A ausência de qualquer mecanismo de verificação de integridade das leituras — como validação cruzada com sensores redundantes, filtros temporais, ou assinaturas digitais de sensores inteligentes — demonstrou que o sistema é suscetível a esse tipo de adulteração. Essa falha pode ser explorada por um invasor com acesso físico ao dispositivo, comprometendo a confiabilidade dos dados coletados e a lógica de tomada de decisão do sistema automatizado de irrigação.

Dentre as principais contramedidas para mitigar esse tipo de ataque, destacam-se:

- A blindagem física dos fios e conectores dos sensores;
- A implementação de filtros temporais e limites plausíveis para variação de dados;
- O uso de sensores digitais com comunicação autenticada;
- O monitoramento contínuo de padrões de leitura suspeitos.

Esse experimento reforça a importância de considerar não apenas ameaças em nível lógico e de software, mas também ataques físicos simples e eficazes que exploram a ausência de verificação na origem dos dados sensoriais em sistemas IoT. Esse tipo de adulteração se enquadra na categoria **Tampering** da metodologia STRIDE, e destaca a necessidade de mecanismos de verificação e integridade para garantir decisões confiáveis na automação agrícola.

5.5 Intercepção de Pacotes MQTT e Análise de Segurança

Durante os testes de captura de pacotes MQTT utilizando a ferramenta *tcpdump*, foi possível interceptar dados sensoriais enviados pelo ESP32 ao servidor de borda. A comunicação se deu do IP `192.168.1.153` (dispositivo ESP32) para o IP `192.168.1.188` (Orange Pi), utilizando a porta `1884`.

A seguir, apresenta-se um exemplo real do conteúdo extraído do pacote interceptado, o qual continha dados de leitura dos sensores embarcados no dispositivo:

```
{
  "umidade_ar": 50.10,
  "temperatura": 22.00,
  "fluxo": 0.00,
  "tensao": 2.25,
  "solo": 3
}
```

Esse tipo de informação, quando não protegida por criptografia (ex.: MQTT sobre TLS), pode ser facilmente acessada por um atacante conectado à mesma rede, representando um risco à confidencialidade dos dados transmitidos. Adicionalmente, a análise do dump de memória do ESP32 contribuiu diretamente para o sucesso da interceptação dos pacotes, uma vez que revelou informações sensíveis como o SSID da rede Wi-Fi, senha de acesso, IP do servidor de borda, porta de comunicação MQTT personalizada (1884), client ID MQTT e o nome do tópico utilizado para publicação das leituras (ex.: `iot/esp32/leitura`). De posse desses dados, um invasor pode configurar filtros altamente direcionados no Wireshark para isolar pacotes específicos, sem necessidade de capturar ou analisar grandes volumes de tráfego.

Apesar da porta utilizada para comunicação MQTT (1884) não ser a padrão (1883), essa alteração representa apenas uma forma superficial de ofuscação, já que a porta foi facilmente descoberta na análise da memória flash. Portanto, não se configura como uma medida de segurança eficaz.

Para mitigar esse tipo de exposição, é recomendável adotar medidas como:

- Utilização de MQTT sobre TLS (porta 8883), garantindo criptografia de ponta a ponta;
- Autenticação mútua entre cliente e broker com certificados digitais;
- Segmentação da rede por VLANs, isolando dispositivos IoT do tráfego da rede principal;
- Armazenamento seguro das credenciais e configurações no ESP32, com criptografia de flash e queima de eFuses;
- Obfuscação ou cifragem do payload MQTT para dificultar a leitura mesmo em tráfego não criptografado;
- Monitoramento da rede para detectar tentativas de sniffing ou conexões não autorizadas.

Essas medidas contribuem para dificultar a escuta não autorizada e garantir a confidencialidade e integridade dos dados sensoriais transmitidos. A ausência de criptografia e autenticação robusta expõe o sistema a ameaças de **divulgação de informação (Information Disclosure)** e **falsificação de identidade (Spoofing)**, conforme descrito no modelo STRIDE. A mitigação dessas vulnerabilidades é essencial para assegurar a proteção de sistemas IoT distribuídos em ambientes abertos ou compartilhados.

5.6 Leitura da Flash via UART e Verificação de Criptografia

Com o dispositivo ESP32-S3 conectado fisicamente ao computador pela interface USB (porta `COM8`), foi executado o teste de leitura da memória flash com o objetivo

de verificar se o conteúdo do firmware estava protegido por criptografia. Utilizou-se a ferramenta `esptool.py` para acessar o modo de bootloader via UART, explorando a combinação de `GPIO0` em nível lógico baixo durante o reset.

O seguinte comando foi executado para extrair os primeiros 64 KB da flash:

```
python -m esptool --port COM8 read_flash 0x00000 0x10000 flash_teste.bin
```

Após a extração, foi utilizado o utilitário `strings64.exe` (Sysinternals) para verificar se o arquivo binário `flash_teste.bin` continha strings legíveis. O resultado do comando demonstrou diversas informações sensíveis armazenadas em texto claro, como SSIDs e senhas de redes Wi-Fi:

```
sta.ssid
caju
sta.pswd
dttv4c4pr3t4
sta.ssid
teste123
sta.pswd
12345678
```

A presença dessas strings comprova que o conteúdo da flash não está criptografado, representando uma vulnerabilidade crítica caso o dispositivo seja acessado fisicamente. Esta falha se enquadra nas categorias de **divulgação de informação (Information Disclosure)** e **elevação de privilégio (Elevation of Privilege)** segundo o modelo STRIDE, pois permite a extração de credenciais e possivelmente a modificação não autorizada do firmware.

Recomendação: ativar a criptografia da flash por meio da configuração de eFuses específicos, utilizando ferramentas como `espefuse.py`, e também implementar o *Secure Boot* para evitar execução de firmwares não autorizados. Tais medidas aumentam significativamente a resiliência do dispositivo contra ataques físicos e engenharia reversa (Espressif Systems, 2024), mesmo em sistemas de baixo custo como aplicações agrícolas distribuídas.

5.7 Análise de eFuses no ESP32-S3

Para verificar o estado das proteções físicas do dispositivo ESP32-S3, foi utilizado o utilitário `espefuse.py` para inspecionar os valores programados nos fusíveis eletrônicos (eFuses). A análise foi realizada com o comando:

```
python -m espefuse --port COM8 summary
```

A saída indicou que os principais mecanismos de segurança embarcados no ESP32-S3 estavam desativados. A Tabela 14 resume os campos relevantes observados:

Tabela 14 – Estado dos principais eFuses de segurança no ESP32-S3

eFuse	Valor	Status
SPI_BOOT_CRYPT_CNT	0b000	Flash Encryption **desativada**
SECURE_BOOT_EN	False	Secure Boot **desativado**
DIS_DOWNLOAD_MODE	False	UART Download **habilitado**
SOFT_DIS_JTAG / DIS_PAD_JTAG	0 / False	JTAG **habilitado**
BLOCK_KEY0 - BLOCK_KEY5	Zeros	Chaves de criptografia **não configuradas**
SECURE_VERSION	0	Anti-rollback **inativo**

A análise comprova que o dispositivo encontra-se em estado vulnerável a ataques físicos. Com o modo de download habilitado, a flash desprotegida e a interface JTAG ativa, é possível realizar leitura completa da memória, engenharia reversa e substituição do firmware sem qualquer impedimento técnico.

Recomendações:

- Habilitar a criptografia da flash por meio da configuração apropriada do eFuse `SPI_BOOT_CRYPT_CNT`, preferencialmente utilizando o processo seguro de `idf.py flash encrypt`.
- Ativar o *Secure Boot* com `Secure Boot v2`, incluindo a gravação de uma chave segura nos blocos `BLOCK_KEYx` e a queima do eFuse `SECURE_BOOT_EN`.
- Desabilitar o modo de download via UART (`DIS_DOWNLOAD_MODE`) e a interface JTAG (`DIS_PAD_JTAG`) permanentemente após o desenvolvimento.
- Implementar verificação de versão segura com `SECURE_VERSION` para mitigar ataques por downgrade de firmware.

Tais medidas reduzem significativamente a superfície de ataque físico e garantem que apenas firmwares autorizados e protegidos possam ser executados no dispositivo (Espressif Systems, 2024). Esse diagnóstico evidencia que, mesmo após o desenvolvimento funcional do sistema, a ausência de configurações mínimas de segurança física pode comprometer por completo a integridade e confidencialidade da solução, especialmente em ambientes expostos. Essa constatação reforça a necessidade de incorporar camadas de proteção embarcada como parte fundamental da arquitetura, e não apenas como um complemento posterior ao projeto.

5.8 Teste de Substituição de Firmware via UART

Com o objetivo de verificar a existência de proteções contra regravação não autorizada do firmware, foi realizado um teste de sobrescrita total da memória flash do

dispositivo ESP32-S3 por meio da interface UART. O procedimento consistiu em forçar o microcontrolador a entrar no modo de bootloader por hardware (colocando o pino `GPIO0` em nível lógico baixo e reiniciando o dispositivo) e, em seguida, utilizar a ferramenta `esptool.py` para gravar um novo firmware sem qualquer autenticação. O comando utilizado foi:

```
python -m esptool --chip esp32s3 --port COM7 write_flash -z 0x0
ESP32S3_WiFi_Test_Functional.bin
```

O procedimento foi concluído com sucesso, resultando na substituição completa do *firmware* previamente instalado. O dispositivo reiniciou normalmente com o novo código, confirmando que não havia nenhuma barreira de segurança ativa, como verificação de integridade, assinatura digital ou autenticação de origem do binário. Esse comportamento evidencia que os mecanismos de segurança esperados para impedir regravações maliciosas — como o *Secure Boot* e a proteção por eFuses — estavam desativados.

Recomendação: ativar o *Secure Boot v2* com assinatura obrigatória de firmware, programar os eFuses responsáveis por desabilitar o modo de download via UART (`DIS_DOWNLOAD_MODE`) e proteger a memória flash contra gravações arbitrárias (Espressif Systems, 2024).

Esse tipo de ataque permite que um agente malicioso substitua completamente o comportamento do sistema, inclusive desativando futuras atualizações remotas, instalando backdoors ou corrompendo a lógica de controle da irrigação. Portanto, a ausência do Secure Boot e de proteções físicas representa um risco crítico que compromete toda a integridade da solução embarcada.

5.9 Simulação de Glitch Físico na Inicialização do ESP32-S3

Com o intuito de avaliar a robustez do sistema frente a falhas físicas simples, realizou-se um teste de simulação de glitch manual durante o processo de inicialização do ESP32-S3. O procedimento consistiu em manipular os botões físicos da placa — `EN` (reset) e `BOOT` (GPIO0) — de forma propositalmente imprecisa, com o objetivo de reproduzir ruídos ou instabilidades elétricas que poderiam ocorrer em ambiente hostil ou ser exploradas como vetores de ataque do tipo *fault injection*.

Os seguintes cenários foram testados:

- **Segurar o botão `BOOT`, pressionar e soltar `EN`, e então soltar `BOOT`**: o ESP32-S3 entrou corretamente no modo de bootloader, conforme esperado. A mensagem exibida foi:

  ```
  rst:0x1 (POWERON),boot:0x0 (DOWNLOAD(USB/UART0))
  waiting for download
  ```

- **Pressionar `EN`, e somente depois pressionar/soltar `BOOT`**: o ESP32-S3 apresentou falha severa durante o processo de boot, com erro de watchdog e `Guru Meditation Error`, conforme demonstrado:

  ```
  Guru Meditation Error: Core  1 panic'ed (Interrupt wdt timeout on CPU1).
  Backtrace: 0x4038644c:... |<-CORRUPTED
  ```

- **Soltar o botão `BOOT` muito rapidamente ou durante o reset**: causou novo travamento, seguido de `core dump` e reinicialização inesperada:

  ```
  Re-entered core dump! Exception happened during core dump!
  Rebooting...
  ```

Tais resultados demonstram que, mesmo sem o uso de equipamentos especializados (como glitchers de clock ou crowbars de tensão), é possível induzir falhas operacionais por meio de manipulações físicas simples nos pinos de boot. Esse tipo de vulnerabilidade pode ser explorado para forçar o microcontrolador a entrar em estados não previstos de execução, possivelmente contornando proteções de segurança baseadas em estado de boot.

Recomendação: implementar medidas de mitigação como verificação de integridade no código de boot, uso de *brown-out detectors*, e rotinas que verifiquem anomalias no processo de inicialização. Em ambientes sensíveis, é aconselhável complementar a proteção com hardware que detecte alterações bruscas de alimentação ou manipulação externa de GPIOs críticos.

Esse tipo de falha ilustra a importância de complementar as proteções lógicas com mecanismos de detecção física e validação contínua da integridade do processo de boot. Mesmo em sistemas com criptografia de flash e Secure Boot ativados, ataques baseados em glitch podem induzir falhas no carregamento ou permitir a ativação indevida de modos privilegiados, comprometendo a confiabilidade do dispositivo em ambientes não supervisionados.

5.10 Análise de Armazenamento de Dados Sensíveis no Firmware

Foi realizada a análise do código-fonte utilizado no dispositivo ESP32-S3 com o objetivo de identificar práticas inseguras relacionadas ao armazenamento de credenciais e dados sensíveis. O código analisado implementa a conexão à rede Wi-Fi e a publicação de dados via protocolo MQTT, conforme o trecho a seguir:

```
const char* ssid = "caju";
const char* password = "v4c4pr3t4";
const char* mqtt_BROKER = "192.168.1.188";
```

```
const char* mqtt_CLIENT_ID = "esp32-sensor-001";
WiFi.begin(ssid, password);
```

A análise revelou que as credenciais de rede e informações de conexão MQTT estão armazenadas diretamente no código, como literais em variáveis constantes. Essa prática resulta na gravação dessas informações na memória flash do dispositivo, tornando-as facilmente recuperáveis em caso de acesso físico — como foi confirmado nos testes anteriores de dump de flash e análise com a ferramenta `strings`.

Além disso, não foi utilizada a função `WiFi.setStorage(WIFI_STORAGE_RAM)`, o que faz com que o ESP32-S3 utilize o modo padrão `WIFI_STORAGE_FLASH`. Esse modo persiste automaticamente as credenciais na memória flash após o primeiro uso, mesmo que tenham sido inseridas em tempo de execução.

Riscos identificados:

- Exposição de SSID e senha do Wi-Fi em texto claro na memória flash.
- Retenção não controlada de credenciais sensíveis mesmo após reinicializações.
- Ausência de técnicas de proteção em tempo de execução, como apagamento de variáveis sensíveis.

Recomendações:

- Utilizar `WiFi.setStorage(WIFI_STORAGE_RAM)` para evitar a persistência automática de credenciais.
- Evitar o uso de literais fixos no código-fonte; optar por entrada segura via interface serial, NVS com criptografia, ou configuração dinâmica (por exemplo, via BLE).
- Implementar práticas de *zeroization*, apagando explicitamente variáveis sensíveis após o uso.

Essas medidas contribuem para reduzir significativamente a superfície de ataque, principalmente em cenários onde o acesso físico ao dispositivo é possível (Espressif Systems, 2024). A presença de credenciais sensíveis no firmware compromete diretamente a confidencialidade da coleta e transmissão de dados, expondo o sistema a ataques de interceptação, controle indevido e substituição de sensores legítimos por dispositivos maliciosos. Essa vulnerabilidade reforça a necessidade de boas práticas de desenvolvimento seguro desde as primeiras linhas de código.

5.11 Resumo dos Testes de Segurança Realizados

Ao longo deste capítulo, foram conduzidos diversos testes práticos com o objetivo de identificar vulnerabilidades presentes na implementação original do sistema embarcado

Tabela 15 – Resumo dos Testes de Segurança Realizados no ESP32-S3

Teste	**Resultado Observado**	**Risco Identificado**	**Contramedidas Recomendadas**
Dump de Flash	Firmware legível, credenciais expostas	Acesso direto a SSID, senhas e variáveis internas	Ativar Flash Encryption
Leitura dos eFuses	Proteções desativadas (Flash Encryption, Secure Boot, UART, JTAG)	Regravação e leitura sem restrições físicas	Programar eFuses e ativar boot seguro
Substituição de Firmware via UART	Upload permitido sem autenticação	Invasor pode sobrescrever firmware original	Ativar Secure Boot + desabilitar UART download
Simulação de Glitch Manual	Boot falhou com manipulação simples de pinos	Potencial para ataques de fault injection	Verificação de integridade no boot + detecção de anomalias
Análise de Código (Wi-Fi / MQTT)	Credenciais em texto fixo e armazenadas na flash	Dados sensíveis persistem mesmo após reset	Armazenar em RAM ou via NVS criptografada + apagar após uso

baseado no ESP32-S3. As atividades envolveram tanto o acesso físico ao hardware quanto a análise lógica do código-fonte, explorando diferentes vetores de ataque conhecidos para dispositivos IoT.

Para consolidar os resultados obtidos, a Tabela 15 apresenta um resumo comparativo dos testes realizados, destacando os comportamentos observados, os riscos de segurança associados e as contramedidas técnicas recomendadas. Essa síntese tem como objetivo facilitar a visualização do estado inicial de segurança do sistema e fundamentar as melhorias que serão aplicadas na etapa seguinte do projeto.

A tabela permite observar de forma integrada que muitas das vulnerabilidades não estão isoladas, mas interligadas — por exemplo, o armazenamento de credenciais em texto claro contribui diretamente para os riscos identificados na extração de firmware, e a ausência de proteção por eFuses permite que outras técnicas de ataque sejam efetivas. Isso reforça a necessidade de uma abordagem combinada de segurança, contemplando tanto proteções de hardware quanto boas práticas no desenvolvimento do firmware.

5.12 Considerações Finais sobre a Avaliação de Segurança

Os testes apresentados neste capítulo demonstram que, na configuração inicial, o dispositivo ESP32-S3 apresenta diversas vulnerabilidades relacionadas ao acesso físico, persistência de dados sensíveis e ausência de proteções por hardware. Foi possível extrair informações críticas, substituir o firmware e causar falhas no boot por meio de manipulações simples, sem o uso de equipamentos avançados. Esses resultados reforçam

a importância da configuração adequada dos mecanismos de segurança embarcados no chip, como a criptografia da memória flash, o uso do *Secure Boot*, a restrição de interfaces de depuração (UART e JTAG) e o controle sobre como as credenciais são armazenadas e gerenciadas no firmware.

No capítulo final, será realizada uma nova rodada de testes, com as medidas de segurança implementadas conforme as recomendações descritas. O objetivo será demonstrar, de forma prática, a efetividade dessas contramedidas por meio da comparação dos dados extraídos — evidenciando a redução (ou eliminação) de informações sensíveis acessíveis por meios não autorizados. Portanto, a abordagem prática adotada neste capítulo não apenas expôs vulnerabilidades críticas presentes em sistemas embarcados mal configurados, mas também forneceu evidências técnicas que fundamentam a adoção de uma estratégia de segurança em múltiplas camadas. O próximo capítulo validará essa abordagem ao demonstrar, por meio de testes reais, a eficácia das medidas implementadas para mitigar os riscos identificados.

6 CONCLUSÃO E RECOMENDAÇÕES

Os testes realizados no Capítulo 5 evidenciaram vulnerabilidades críticas na configuração original do sistema embarcado, incluindo extração de dados sensíveis da memória flash, substituição não autorizada de *firmware* e interceptação de pacotes MQTT em texto claro. Esses resultados reforçaram a necessidade da aplicação prática das contramedidas previstas nas modelagens STRIDE e DREAD, visando reduzir os riscos identificados.

Este capítulo apresenta a implementação das principais medidas de segurança no ESP32-S3, com foco na proteção da memória flash, do processo de inicialização e da comunicação com o servidor de borda. Também são discutidos os resultados dos testes realizados após a aplicação dessas técnicas, bem como as limitações da solução atual e direções para trabalhos futuros.

6.1 Transição para a Plataforma ESP-IDF

A implementação das medidas de segurança exigiu a migração do desenvolvimento da aplicação da plataforma Arduino para o ambiente nativo *ESP-IDF* (*Espressif IoT Development Framework*). Essa mudança foi necessária devido às limitações da IDE Arduino no suporte a mecanismos avançados de segurança embarcada.

A ESP-IDF oferece suporte completo a recursos críticos como:

- Criptografia da memória flash (*Flash Encryption*);
- Inicialização segura do sistema (*Secure Boot*);
- Gerenciamento de chaves criptográficas por eFuses;
- Controle mais refinado das permissões de acesso e inicialização;
- Monitoramento do sistema por meio de ferramentas nativas do framework.

Embora a transição tenha exigido adaptações estruturais no código-fonte, a lógica principal do sistema foi mantida: leitura dos sensores, formatação dos dados em JSON e envio via MQTT ao servidor de borda. A adoção da ESP-IDF foi fundamental para viabilizar a aplicação das técnicas de proteção testadas neste trabalho, garantindo maior controle sobre o hardware e maior aderência às boas práticas de segurança recomendadas pela própria fabricante do microcontrolador.

6.2 Ativação da Criptografia da Memória Flash

A primeira medida de proteção implementada no ESP32-S3 foi a ativação da **criptografia da memória flash** (*Flash Encryption*), recurso nativo da plataforma que impede a leitura não autorizada dos dados armazenados, como firmware, credenciais e variáveis de configuração. Optou-se inicialmente pelo *modo de desenvolvimento*, que permite

testar a aplicação e realizar atualizações sem queimar eFuses de forma permanente. Após a validação funcional do sistema, planeja-se a migração para o *modo de produção*, onde as proteções tornam-se irreversíveis.

A ativação da criptografia resultou em bloqueio efetivo das tentativas de leitura direta da memória via UART. Diferentemente dos testes realizados com a flash desprotegida, não foi possível extrair informações sensíveis nem utilizar ferramentas como `strings` para inspecionar o conteúdo dos arquivos binários gerados. Isso comprova a efetividade da proteção, mesmo em modo de testes.

Impacto e Limitações: embora a criptografia tenha se mostrado eficaz contra ataques físicos, seu uso em modo de desenvolvimento ainda permite regravações legítimas. Além disso, a técnica não protege contra injeção de *firmware* malicioso, sendo necessário combiná-la com o *Secure Boot*, tratado na seção seguinte. Os comandos e configurações detalhadas para ativação da *Flash Encryption* estão descritos no Apêndice A.

6.3 Validação da Criptografia por Dump de Memória Flash

Para validar a eficácia da criptografia da memória flash, foi realizado um novo teste de leitura direta utilizando a ferramenta `esptool.py` com o dispositivo já protegido. O conteúdo foi extraído em blocos de 16 KB e analisado por meio da ferramenta `strings`.

Nos testes anteriores, sem proteção, o dump da memória revelou credenciais de Wi-Fi, tópicos MQTT, payloads em JSON e variáveis internas da aplicação, todos em texto claro. Já com **Flash Encryption** ativada, o conteúdo extraído apresentou-se totalmente embaralhado, e nenhuma string sensível foi identificada — incluindo nos primeiros setores da flash. Além disso, a leitura da memória foi interrompida após 25 blocos (de um total esperado de 256), sugerindo um bloqueio adicional de leitura por software ou hardware.

Resultado: a criptografia da flash impediu completamente a extração de informações úteis, mesmo com acesso físico ao dispositivo, validando sua efetividade como barreira contra engenharia reversa.

Impacto: a aplicação da *Flash Encryption* protegeu o *firmware* e os dados sensoriais, mitigando um dos principais vetores de ataque físico documentados no Capítulo 5. Os scripts e ferramentas utilizados para este teste encontram-se descritos no Apêndice A.

6.4 Implementação do Boot Seguro (Secure Boot V2)

O mecanismo de **Secure Boot V2**, disponibilizado pela Espressif, foi ativado com sucesso no ESP32-S3, adicionando uma camada de proteção contra a execução de firmwares adulterados ou não autorizados. Com essa funcionalidade, o dispositivo passa a verificar a assinatura digital do bootloader e da aplicação no momento da inicialização, bloqueando automaticamente qualquer binário não assinado com a chave previamente registrada.

Nesta etapa, o sistema foi configurado em **modo de desenvolvimento**, o que permite atualizações seguras do *firmware* enquanto as eFuses ainda não estão permanentemente queimadas. Isso viabilizou a regravação de imagens assinadas com a mesma chave e a execução de testes com segurança, antes da ativação definitiva.

A geração das chaves, a assinatura automática do *firmware* e os parâmetros de configuração foram realizados por meio da ferramenta `menuconfig`, seguindo as diretrizes oficiais da ESP-IDF. Os passos detalhados estão documentados no Apêndice A.

Status atual: o dispositivo está operando com o *Secure Boot V2* ativo e funcional, porém com as eFuses de proteção ainda desabilitadas, permitindo regravações controladas.

Próximos passos: a ativação definitiva do Secure Boot, com a queima das eFuses, está prevista para a fase final do projeto. Essa medida tornará irreversível a exigência por *firmware* assinado, elevando o nível de segurança para a produção. Antes disso, será necessário garantir:

- A validação completa da versão final do *firmware*;
- A preparação de mecanismos de atualização OTA com verificação de assinatura;
- O backup seguro das chaves privadas utilizadas na assinatura.

A combinação entre **Flash Encryption** e **Secure Boot V2** oferece proteção abrangente contra extração de firmware, substituição maliciosa e execução de código não confiável, reforçando a confiabilidade da solução embarcada.

6.5 Resultados da Proteção com Secure Boot

Após a ativação combinada da **Flash Encryption** e do **Secure Boot V2**, foi realizado um novo teste de leitura da memória flash para verificar a efetividade das proteções implementadas. Utilizou-se o mesmo procedimento dos testes anteriores: leitura da memória por blocos de 16 KB via interface serial, utilizando a ferramenta `esptool`.

Diferente do cenário anterior, no qual apenas a criptografia da flash estava habilitada, esta nova extração não revelou qualquer string sensível ou identificável. O conteúdo dos blocos lidos consistia unicamente em dados embaralhados e mensagens internas do *bootloader*, sem qualquer ocorrência de JSON, tópicos MQTT ou credenciais de rede. Alguns trechos ilustrativos extraídos são:

```
E (%lu) %s: Initialization of Flash Encryption key failed (%d)
I (%lu) %s: Partition Table:
I (%lu) %s: Defaulting to factory image
```

Comparativo entre os Cenários

- **Com Flash Encryption apenas:** ainda foi possível recuperar nomes de tópicos, amostras de payloads MQTT e informações de rede.
- **Com Flash Encryption + Secure Boot:** nenhuma informação útil foi extraída; todo o conteúdo encontrava-se criptografado e embaralhado.

Esse resultado demonstra que a proteção conjunta impede não apenas a leitura dos dados armazenados, mas também a execução de *firmwares* adulterados, mesmo com acesso físico ao dispositivo.

Conclusão Parcial

A implementação dessas duas camadas de segurança representa um salto significativo na resiliência do sistema embarcado. A confidencialidade dos dados foi preservada, e a integridade do *firmware* garantida. Essa abordagem se mostra especialmente adequada para aplicações em campo, como sistemas de irrigação remotos, nos quais o risco de acesso físico indevido não pode ser ignorado.

6.6 Manipulação de Leituras Analógicas por Injeção de Falhas no ESP32

Nesta etapa, foi realizado um experimento para avaliar a vulnerabilidade do ESP32-S3 à manipulação de sinais analógicos utilizados na leitura de sensores. O teste consistiu na substituição do sensor capacitivo de umidade do solo por um **potenciômetro de 10 k**Ω, conectado entre os pinos de alimentação (3,3 V e GND), com o terminal central ligado à entrada analógica `GPIO36`.

Com essa configuração, foi possível controlar manualmente a tensão de entrada no conversor analógico-digital (ADC) e simular artificialmente diferentes níveis de umidade. O ESP32 interpretou os valores como leituras legítimas do sensor, o que demonstra a ausência de verificação de integridade nos dados recebidos.

Implicações de Segurança

Esse tipo de manipulação caracteriza uma **injeção de falha física**, permitindo que um invasor induza o sistema a tomar decisões incorretas, como acionar ou inibir a irrigação fora dos parâmetros reais. Em ambientes remotos, essa vulnerabilidade pode ser explorada para sabotar a operação ou provocar desperdícios.

Medidas de Mitigação

As seguintes estratégias são recomendadas para mitigar o risco identificado:

- **Sensor Fusion:** validar os dados com múltiplos sensores e checagens cruzadas;

- **Limites de Segurança:** definir faixas plausíveis de leitura e rejeitar valores abruptos;
- **Monitoramento de Sinal:** detectar sinais estáticos ou com ruídos incompatíveis;
- **Sensores Digitais:** preferir sensores com comunicação autenticada (I2C, SPI com CRC);
- **Blindagem Física:** dificultar o acesso a conexões expostas por encapsulamento;
- **Auditoria de Leituras:** registrar e analisar padrões de leitura suspeitos.

Conclusão da Etapa

O teste comprovou que sensores analógicos não protegidos são suscetíveis à manipulação local. A substituição por um potenciômetro foi suficiente para enganar o sistema e influenciar seu comportamento. A adoção de sensores digitais, filtros de validação e proteção física é fundamental para garantir a integridade dos dados sensoriais em aplicações críticas de IoT.

6.7 Intercepção de Pacotes MQTT e Análise de Segurança

Com o objetivo de proteger os dados sensoriais transmitidos entre o ESP32-S3 e o servidor de borda (Orange Pi), foi implementado o protocolo *Transport Layer Security* (TLS) na comunicação MQTT, utilizando a porta padrão segura 8883.

Procedimento de Configuração

No servidor, foi gerado um certificado digital autofirmado utilizando `OpenSSL`. O broker Mosquitto foi configurado para aceitar conexões seguras com os seguintes parâmetros:

```
listener 8883
protocol mqtt
cafile /etc/mosquitto/certs/mosquitto.crt
certfile /etc/mosquitto/certs/mosquitto.crt
keyfile /etc/mosquitto/certs/mosquitto.key
```

No lado do cliente, o *firmware* do ESP32 foi adaptado com a biblioteca `WiFiClientSecure` e o certificado do broker embarcado diretamente no código:

```
espClient.setCACert(ca_cert);
```

Validação com Captura de Pacotes

A ferramenta `tcpdump` foi utilizada no Orange Pi para capturar o tráfego na porta 8883. Os pacotes foram analisados no *Wireshark*, revelando que, com o TLS ativado, os dados sensoriais estavam encapsulados como **TLS Application Data**, impedindo a visualização de informações como tópicos e payloads MQTT.

Em comparação, na comunicação sem TLS (porta 1884), os dados eram transmitidos em texto claro, incluindo tópicos MQTT e leituras dos sensores (Figura 12).

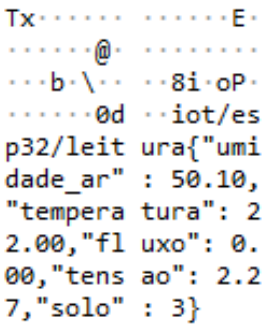

Figura 12 – Pacote MQTT capturado sem TLS, com dados visíveis.

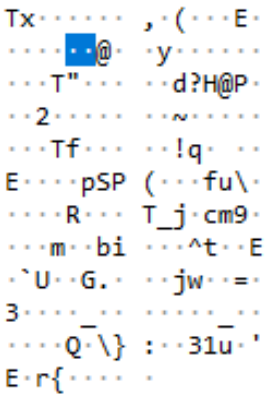

Figura 13 – Pacote MQTT com TLS, dados criptografados.

Comparativo entre os Modos de Comunicação

Tabela 16 – Comparação entre comunicação MQTT com e sem TLS

Aspecto	**Sem TLS**	**Com TLS**
Porta utilizada	1884	8883
Conteúdo dos pacotes	Visível (JSON)	Criptografado (TLS)
Tópico MQTT	Visível	Oculto
Privacidade dos dados	Nenhuma	Garantida
Análise no Wireshark	Legível	Protegido

Conclusão da Etapa

A utilização de TLS no canal MQTT demonstrou ser uma medida eficaz para proteger a confidencialidade dos dados sensoriais em trânsito. Os testes comprovaram que, sem TLS, os dados podem ser interceptados e lidos com facilidade, enquanto com TLS os pacotes são protegidos contra ataques de interceptação, como sniffing ou *Man-in-the-Middle*.

6.8 Leitura da Flash via UART e Verificação de Criptografia

Com o objetivo de validar a proteção contra extração de dados por acesso físico, foi realizado um teste de leitura direta da memória flash do ESP32-S3 via interface UART. Dois dispositivos foram utilizados: um sem proteções habilitadas e outro com **Flash Encryption**, **Secure Boot** e **eFuses** devidamente configurados.

Dispositivo Sem Proteção

No primeiro cenário, a memória flash foi completamente lida com sucesso utilizando a ferramenta `esptool.py`:

```
esptool.py --chip esp32 --port COM7 read_flash 0x00000 0x400000 esp32_dump.bin
```

A análise do conteúdo com `strings` revelou informações sensíveis armazenadas em texto claro, incluindo:

- Tópico MQTT: `iot/esp32/leitura`;
- Endereço IP do broker: `192.168.1.188`;
- Payloads JSON de sensores;
- Trechos do código-fonte e variáveis de configuração.

Esse resultado comprova a exposição dos dados em dispositivos sem criptografia habilitada.

Dispositivo com Proteção Ativada

No segundo cenário, foi utilizado um ESP32 com as proteções ativadas. A leitura foi repetida com o mesmo comando, gerando o arquivo `esp32_flash_seguro.bin`. A análise com `strings` revelou apenas mensagens genéricas do bootloader, como:

```
"boot", "RF data", "ota"
```

Nenhuma credencial, tópico ou valor sensorial foi identificado. Isso demonstra a efetividade combinada da criptografia de flash, do Secure Boot e da queima de eFuses para impedir a leitura ou engenharia reversa do firmware.

Comparação dos Resultados

Tabela 17 – Comparação entre leitura da flash com e sem proteção

Informação Extraída	Sem Proteção	Com Proteção
Tópico MQTT	Visível	Oculto
Endereço IP do Broker	Visível	Oculto
Payload JSON	Visível	Oculto
Trechos do Código	Parcialmente visíveis	Indecifráveis
Mensagens Internas	Legíveis	Genéricas apenas

Conclusão da Etapa

Os testes demonstram de forma conclusiva que a combinação entre **Flash Encryption**, **Secure Boot** e **queima de eFuses** bloqueia efetivamente a leitura da flash via UART. Mesmo com acesso físico ao dispositivo e ferramentas adequadas, não foi possível obter dados úteis no segundo cenário, validando a eficácia das medidas de segurança implementadas.

6.9 Análise de eFuses no ESP32-S3

Esta etapa teve como objetivo verificar se o ESP32-S3 utilizado estava devidamente protegido contra leitura não autorizada da memória flash após a ativação das técnicas de segurança. Para isso, foi analisado o estado das eFuses — regiões de memória OTP (One Time Programmable) responsáveis por definir parâmetros de segurança permanentes no chip.

Leitura da Flash com Proteções Ativas

Com o dispositivo conectado à porta `COM10`, foi realizado o seguinte comando para leitura completa da flash:

```
esptool.py --chip esp32 --port COM10 read_flash 0x00000 0x400000 tentativa_bloqueio.bin
```

O arquivo foi gerado com sucesso, mas os dados estavam criptografados e inutilizáveis, mesmo com ferramentas como `strings`, confirmando a proteção ativa da flash.

Verificação das eFuses

A seguir, foi executado o comando:

```
espefuse.py -p COM10 summary
```

Os campos relevantes confirmaram a proteção:

- `FLASH_CRYPT_CNT = 1`: criptografia da flash ativada.
- `DISABLE_DL_DECRYPT = True`: bloqueia leitura durante boot UART.
- `JTAG_DISABLE = True`: desativa acesso de depuração físico.
- `CONSOLE_DEBUG_DISABLE = True`: bloqueia interpretador via UART.
- `WR_DIS / RD_DIS`: impedem regravação/leitura de regiões protegidas.

Conclusão da Etapa

A combinação entre Flash Encryption ativada e eFuses programadas garantiu que os dados permanecessem confidenciais, mesmo após a extração completa da memória. O teste validou a eficácia das proteções físicas e lógicas do ESP32-S3, reforçando sua adequação a ambientes expostos a risco físico, como o sistema de irrigação proposto.

6.10 Teste de Substituição de Firmware via UART

Para verificar a eficácia das proteções contra gravação não autorizada de firmware, foi realizado um teste de substituição via UART em um ESP32-S3 com **Flash Encryption**, **Secure Boot** e **eFuses** ativados.

Utilizou-se uma aplicação simples baseada no exemplo `hello_world` da plataforma ESP-IDF, compilada com:

```
idf.py build
```

Em seguida, foi tentada a gravação direta via UART:

```
esptool.py --chip esp32 --port COM10 write_flash -z 0x10000 build/hello_world.bin
```

A escrita foi concluída sem erros, mas a aplicação não executou. O monitor serial exibiu:

```
rst:0x1 (POWERON_RESET),boot:0x17 (SPI_FAST_FLASH_BOOT)
flash read err, 1000
ets_main.c 371
```

Essa falha de leitura indica que o firmware gravado não possuía criptografia nem assinatura válidas, sendo rejeitado pelo bootloader seguro.

Conclusão da Etapa

O teste confirma que, com as medidas de segurança ativas, o microcontrolador rejeita qualquer tentativa de sobrescrever o firmware original com código não autorizado. Essa proteção é fundamental para prevenir ataques de injeção de firmware malicioso em ambientes IoT expostos.

6.11 Simulação de Glitch Físico na Inicialização do ESP32-S3

Ataques físicos do tipo *glitch* consistem na introdução de ruídos ou instabilidades intencionais na alimentação elétrica de um dispositivo, visando corromper etapas críticas do boot e, potencialmente, burlar proteções de segurança.

Para simular esse cenário, foram provocadas instabilidades no sinal de alimentação (`EN`) do ESP32-S3 durante o processo de inicialização. O dispositivo testado possuía **Flash Encryption** e **Secure Boot** previamente habilitados.

Durante a manipulação manual dos botões `EN` e `BOOT`, foram observadas falhas de leitura na flash:

```
rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
flash read err, 1000
ets_main.c 371
```

Em alguns casos, o dispositivo entrou involuntariamente no modo de bootloader via UART:

```
boot:0x3 (DOWNLOAD_BOOT(UART0/UART1/SDIO_REI_REO_V2))
waiting for download
```

Conclusão da Etapa

Apesar das falhas provocadas, o ESP32 não executou código não autorizado e não revelou informações sensíveis. O teste confirma que, mesmo diante de perturbações físicas simples, os mecanismos de segurança aplicados impedem a leitura ou execução indevida do firmware. Esse comportamento reforça a importância da combinação entre *Flash Encryption* e *Secure Boot* para a proteção de sistemas embarcados.

6.12 Análise de Armazenamento de Dados Sensíveis no Firmware

Durante o desenvolvimento de sistemas embarcados, é comum encontrar credenciais sensíveis diretamente embutidas no código-fonte, como SSID, senha de Wi-Fi, endereço do broker MQTT e tópicos de publicação. Essa prática, embora conveniente, representa uma séria vulnerabilidade.

Para avaliar esse risco, foi realizada a extração da memória flash de um ESP32-S3 sem mecanismos de segurança ativados. A análise com a ferramenta `strings` revelou diversas informações em texto claro, incluindo:

- Endereço do broker MQTT;
- Tópicos e payloads em JSON;

- Identificadores de dispositivo e variáveis de depuração.

Em seguida, o mesmo procedimento foi repetido em um ESP32-S3 com **Flash Encryption** ativado e eFuses programadas. O conteúdo extraído apresentou-se completamente cifrado, sem nenhuma informação útil visível.

Conclusão da Etapa

Os resultados comprovam que o armazenamento de dados sensíveis em texto claro compromete diretamente a segurança do sistema. O uso combinado de *Flash Encryption* e eFuses é eficaz para mitigar esse risco, assegurando que, mesmo com acesso físico ao dispositivo, as credenciais permaneçam protegidas.

6.13 Conclusão e Recomendações Finais

Este capítulo demonstrou a aplicação prática das camadas de segurança propostas no ESP32-S3, validando sua eficácia na proteção de dados sensoriais, firmware e canais de comunicação. Os testes realizados evidenciaram que, na ausência dessas medidas, o sistema permanece vulnerável a extração de credenciais, regravação de firmware, ataques físicos e interceptação de tráfego.

Após a aplicação de mecanismos como **Flash Encryption**, **Secure Boot** e **queima de eFuses**, foi possível bloquear tentativas de leitura da memória flash, impedir substituições não autorizadas de firmware via UART e ocultar completamente dados sensíveis mesmo com acesso físico ao dispositivo.

Tabela 18 – Comparação entre testes com e sem medidas de segurança

Tipo de Teste	**ESP32 sem Proteção**	**ESP32 com Proteção**
Leitura da Flash	Totalmente acessível	Dados criptografados
Extração de Credenciais	Visíveis em texto claro	Não identificados
Substituição de Firmware	Permitida via UART	Bloqueada por eFuses
Ataque Físico (Glitch)	Boot comprometido	Inicialização protegida
Leitura de Strings Internas	Variáveis sensíveis visíveis	Dados embaralhados

As medidas adotadas alinham-se às recomendações da OWASP IoT Top 10 e às diretrizes discutidas por Tanenbaum e Van Steen (TANENBAUM; STEEN, 2024), que reforçam a importância de uma abordagem contínua e multicamada na segurança de sistemas distribuídos.

Limitação Identificada: esta dissertação não avaliou o impacto das técnicas de segurança sobre o desempenho do sistema. Questões como tempo de boot, consumo energético adicional e uso de CPU com TLS ou criptografia embarcada não foram mensurados. Essa lacuna será registrada como limitação do estudo e motivação para trabalhos futuros.

Recomendações:

- Aplicar Flash Encryption e Secure Boot em todo firmware antes da implantação;
- Realizar backup seguro das chaves e queimar eFuses somente após testes finais;
- Adotar sensores digitais sempre que possível, evitando spoofing analógico;
- Proteger fisicamente os dispositivos IoT contra acesso externo;
- Implementar atualizações OTA assinadas e autenticação mútua na comunicação MQTT;
- Evitar armazenamento de credenciais em texto claro no código-fonte.

6.14 Perspectivas Futuras

Embora este trabalho tenha focado no dispositivo embarcado (ESP32-S3), reconhece-se que a segurança de um sistema IoT depende também da proteção das demais camadas da arquitetura.

Como extensão natural da pesquisa, propõe-se:

- Testar a **interface web** e APIs REST desenvolvidas em Django, com base nas diretrizes OWASP Top 10;
- Avaliar a **segurança do servidor de borda (Orange Pi)** e da infraestrutura Docker utilizada;
- Investigar a **síncronia de dados entre borda e nuvem**, incluindo mecanismos de integridade e autenticação;
- Realizar **avaliações de overhead** gerado pelas camadas de segurança aplicadas (tempo de boot, consumo energético, latência MQTT);
- Explorar metodologias emergentes como o **NIST Cybersecurity Framework** em combinação com STRIDE.

Em síntese, este estudo contribui com um conjunto validado de boas práticas para fortalecer a segurança de sistemas embarcados conectados à internet, servindo como base técnica e metodológica para aplicações reais e pesquisas futuras.

REFERÊNCIAS

ABBASI, R.; MARTINEZ, P.; AHMAD, R. The digitization of agricultural industry – a systematic literature review on agriculture 4.0. **Smart Agricultural Technology**, v. 2, p. 100042, 2022. ISSN 2772-3755. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2772375522000090>. Citado 2 vezes nas páginas 17 e 24.

Aerosemi Technology Co., Ltd. **MT3608: High Efficiency 1.2MHz 2A Step-Up Converter**. 2025. <https://www.olimex.com/Products/Breadboarding/BB-PWR-3608/resources/MT3608.pdf>. Acesso em: 3 jun. 2025. Citado na página 28.

AHMADJEE, S. et al. A Study on Blockchain Architecture Design Decisions and their Security Attacks and Threats. **ACM Transactions on Software Engineering and Methodology**, Association for Computing Machinery, v. 31, n. 2, p. 36e, 2022. Disponível em: <https://doi.org/10.1145/3502740>. Citado 2 vezes nas páginas 40 e 41.

Allegro MicroSystems. **ACS712 Hall-Effect-Based Linear Current Sensor Datasheet**. 2025. <https://www.allegromicro.com/-/media/files/datasheets/acs712-datasheet.ashx>. Accessed: 26 maio 2025. Citado na página 28.

ALLOYSEED. **10W 5V Solar Panel Charger with Micro USB for Security Cameras and Wireless Doorbells**. 2025. <https://solarpanel.vip/products/10w-solar-panel-sola>. Acesso em: 3 jun. 2025. Citado na página 27.

Amazon Web Services. **What is MQTT? - MQTT Protocol Explained**. 2025. <https://aws.amazon.com/what-is/mqtt/>. Acesso em: 3 jun. 2025. Citado na página 25.

Aosong Electronics Co., Ltd. **DHT22 (AM2302) Humidity and Temperature Sensor Datasheet**. 2025. <https://www.aosong.com/uploadfiles/2025/04/20250417105409216.pdf>. Acesso em: 3 jun. 2025. Citado na página 28.

Arduino. **Arduino IDE**. 2025. <https://www.arduino.cc/en/software>. Acesso em: 3 jun. 2025. Citado na página 26.

ASIF, M. R. A. et al. STRIDE-based Cyber Security Threat Modeling for IoT-enabled Precision Agriculture Systems. In: **2021 3rd International Conference on Sustainable Technologies for Industry 4.0 (STI)**. [S.l.: s.n.], 2021. p. 1–6. Citado na página 39.

BOURSIANIS, A. D. et al. Smart irrigation system for precision agriculture—the arethou5a iot platform. **IEEE Sensors Journal**, v. 21, n. 16, p. 17539–17547, 2021. Citado na página 17.

Celery Project. **Celery - Distributed Task Queue**. 2025. <https://docs.celeryq.dev/>. Acesso em: 6 jun. 2025. Citado na página 30.

Cemar. **Coluna Metálica de Fixação Externa – Suporte Técnico para IoT**. 2025. <https://es.scribd.com/doc/221230139/Cemar-Quadros-e-Paineis-Para-Instalacoes-Eletricas>. Acesso em: 3 jun. 2025. Citado na página 27.

CHORFA, W.; YOUSSEF, N. B.; JEMAI, A. Threat Modeling with Mitre ATT&CK Framework Mapping for SD-IoT Security Assessment and Mitigations. In: **2023 IEEE Symposium on Computers and Communications (ISCC)**. [S.l.: s.n.], 2023. p. 1323–1326. Citado na página 39.

CONVISO. **Threat Modeling**. 2024. Online. Accessed: 2024-06-17. Disponível em: <https://www.convisoappsec.com/glossario/threat-modeling>. Citado na página 43.

DFRobot. **Capacitive Soil Moisture Sensor V2.0 – Analog Corrosion Resistant Sensor for Arduino**. 2025. <https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0193_Web.pdf>. Datasheet técnico. Acesso em: 3 jun. 2025. Citado na página 29.

diymore. **ESP32/ESP32S 16340/18650 Lithium Battery Shield V3/V8/V9**. 2025. <https://pt.aliexpress.com/item/4000409263677.html>. Acesso em: 3 jun. 2025. Citado na página 28.

Docker Inc. **What is a Container?** 2024. <https://docs.docker.com/get-started/overview/>. Accessed: 26 maio 2025. Citado 2 vezes nas páginas 24 e 30.

Docker Inc. **Docker Compose - Define and run multi-container applications with Docker**. 2025. <https://docs.docker.com/compose/>. Acesso em: 6 jun. 2025. Citado na página 30.

Eclipse Foundation. **Eclipse Paho MQTT Python Client**. 2025. <https://projects.eclipse.org/projects/iot.paho/releases/2.0.0-python-client>. Acesso em: 6 jun. 2025. Citado na página 30.

Emicol Eletro Eletrônica Ltda. **Válvula de Água Emicol 320300 – Especificações Técnicas**. 2025. <https://www.frigelar.com.br/valvula-agua-emicol-compativel-lavadora-brastemp-mondial-simples-suporte-ajustavel-rosca-fina-1e-x-1s-p/kit10691>. Acesso em: 3 jun. 2025. Citado na página 29.

Espressif Systems. **ESP-IDF Security Features: Flash Encryption**. 2024. <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/security/flash-encryption.html>. Acesso em: 11 jun. 2025. Citado 5 vezes nas páginas 63, 66, 67, 68 e 70.

Espressif Systems. **ESP32-S3 Series Datasheet**. 2025. <https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf>. Acesso em: 3 jun. 2025. Citado na página 26.

F5, Inc. **NGINX - High Performance Load Balancer, Web Server and Reverse Proxy**. 2025. <https://www.nginx.com/>. Acesso em: 6 jun. 2025. Citado na página 30.

Fotek Controls Co., Ltd. **Solid State Relay SSR-40DA Datasheet**. 2025. <https://cdn.sparkfun.com/datasheets/Components/General/SSR40DA.pdf>. Acesso em: 3 jun. 2025. Citado na página 29.

FOUNDATION, D. S. **Django Documentation**. 2024. <https://docs.djangoproject.com/en/5.2/>. Accessed: 26 maio 2025. Citado 2 vezes nas páginas 24 e 29.

GHOSH, S. et al. Securing RC Based P2P Networks: A Blockchain-based Access Control Framework utilizing Ethereum Smart Contracts for IoT and Web 3.0. **arXiv**, v. 2412.03709v1, December 2024. Disponível em: <https://arxiv.org/abs/2412.03709>. Citado na página 41.

Gunicorn Project. **Gunicorn - Python WSGI HTTP Server for UNIX**. 2025. <https://gunicorn.org/>. Acesso em: 6 jun. 2025. Citado na página 30.

HellermannTyton. **Soluções de Vedação e Passagem de Cabos – Linha IP67/IP68**. 2025. <https://www.hellermanntyton.com.br/produtos/sistemas-de-protecao-de-cabos>. Acesso em: 3 jun. 2025. Citado na página 26.

Hummel do Brasil. **Caixas Industriais com Grau de Proteção IP66/IP67**. 2025. <https://hummel.com.br/caixas/>. Acesso em: 3 jun. 2025. Citado na página 26.

IHITA, G. V. et al. Security Analysis of Large Scale IoT Network for Pollution Monitoring in Urban India. In: **2021 IEEE 7th World Forum on Internet of Things (WF-IoT)**. [S.l.: s.n.], 2021. p. 283–288. Citado na página 39.

INFOMACH. **Conheça o modelo de ameaça STRIDE**. 2024. Online. Accessed: 2024-06-17. Disponível em: <https://www.infomach.com.br/conheca-o-modelo-de-ameaca-stride/>. Citado na página 43.

KHATIWADA, P. et al. Threats and Risk on Using Digital Technologies for Remote Health Care Process. In: **International Conference on Sustainable Information Engineering and Technology (SIET 2023)**. Association for Computing Machinery, 2023. Disponível em: <https://doi.org/10.1145/3626641.3627604>. Citado 2 vezes nas páginas 40 e 41.

KingSpec. **NE-XXX 2280 Specifications**. 2025. <https://www.kingspec.com/product/m2-nvme-pcie-gen3-ssd-ne-2280mm.html>. Accessed: 26 maio 2025. Citado na página 25.

LECLAIR, B. Threat Modelling in Virtual Assistant Hub Devices Compared With User Risk Perceptions. September 2021. Citado na página 41.

Murata Manufacturing Co., Ltd. **US18650VTC6 Lithium-Ion Rechargeable Battery Datasheet**. 2025. <https://www.murata.com/-/media/webrenewal/products/batteries/cylindrical/datasheet/us18650vtc6-product-datasheet.ashx>. Acesso em: 3 jun. 2025. Citado na página 27.

NAIK, N. et al. A Comparative Analysis of Threat Modelling Methods: STRIDE, DREAD, VAST, PASTA, OCTAVE, and LINDDUN. **International Conference on Computing, Communication, Cybersecurity & AI (C3AI 2024)**, 2024 2024. Citado na página 41.

NanoMQ Project. **NanoMQ - Blazing Fast and Lightweight MQTT Broker for IoT Edge**. 2025. <https://nanomq.io/>. Acesso em: 6 jun. 2025. Citado na página 30.

NIST. **Minimum Security Requirements for Federal Information and Information Systems**. [S.l.], 2006. Accessed: October 15, 2024. Disponível em: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.200.pdf>. Citado na página 43.

OWASP. **Threat Modeling Process**. 2024. <https://owasp.org/www-community/Threat_Modeling_Process>. Accessed: October 15, 2024. Citado 2 vezes nas páginas 33 e 43.

PATIL, S. R. et al. A comparative review on ceph and swift open source cloud storage platform. In: **2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)**. [S.l.: s.n.], 2016. p. 213–218. Citado na página 17.

PFLEEGER, C. P.; PFLEEGER, S. L.; MARGULIES, J. **Security in Computing**. 5th. ed. [S.l.]: Prentice Hall, 2015. Citado na página 55.

PostgreSQL Global Development Group. **PostgreSQL - The World's Most Advanced Open Source Relational Database**. 2025. <https://www.postgresql.org/>. Acesso em: 6 jun. 2025. Citado na página 30.

Redis Ltd. **Redis - In-memory data structure store, used as a database, cache, and message broker**. 2025. <https://redis.io/>. Acesso em: 6 jun. 2025. Citado na página 30.

Renovigi Energia Solar. **Suporte com Ajuste de Inclinação para Módulos Solares – Manual Técnico**. 2025. <https://renovigi.com.br/produto/suportes/suporte-com-ajuste-de-inclinacao-1un>. Acesso em: 3 jun. 2025. Citado na página 27.

RS Components. **5V 3A Power Supply Adapter with USB-C Connector**. 2025. <https://docs.rs-online.com/275b/A700000006857541.pdf>. Datasheet técnico. Acesso em: 3 jun. 2025. Citado na página 27.

Seeed Studio. **G3/4 Water Flow Sensor - FS300A**. 2025. <https://wiki.seeedstudio.com/G3-4_Water_Flow_sensor/>. Acesso em: 3 jun. 2025. Citado na página 28.

Shenzhen Xunlong Software CO., Limited. **Orange Pi 3B Product Page**. 2025. <http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/Orange-Pi-3B.html>. Accessed: 26 maio 2025. Citado na página 25.

Shenzhen Xunlong Software Co., Ltd. **Orange Pi OS**. 2025. <https://www.orangepi.org/html/softWare/orangePiOS/index.html>. Acesso em: 3 jun. 2025. Citado 2 vezes nas páginas 25 e 31.

SHOSTACK, A. **Threat Modeling: Designing for Security**. [S.l.]: Wiley, 2014. Citado na página 43.

SILVA, L. P. da et al. A Comprehensive Approach for Applying Threat Modeling to Internet of Things Systems. In: **2022 IEEE 8th World Forum on Internet of Things (WF-IoT)**. [S.l.: s.n.], 2022. p. 1–8. Citado na página 40.

SMS Tecnologia Eletrônica. **Manual do Usuário – Nobreak Station II 600VA**. 2025. <https://smslegrand.com/uploads/1/0/5/5/105549193/manual_sms_station_ii.pdf>. Acesso em: 3 jun. 2025. Citado na página 27.

SQLite Consortium. **SQLite - Lightweight, Reliable, Embedded SQL Database Engine**. 2025. <https://www.sqlite.org/>. Acesso em: 6 jun. 2025. Citado na página 29.

TANENBAUM, A. S.; STEEN, M. V. **Distributed Systems: Principles and Paradigms**. 3rd. ed. Boston: Pearson, 2024. Citado 4 vezes nas páginas 24, 55, 60 e 83.

Treedix. **Caixa de Acrílico com Ventilador de Resfriamento para Raspberry Pi 5**. 2025. <https://www.amazon.com.br/dp/B0CN2C6BLM>. Acesso em: 3 jun. 2025. Citado na página 27.

VMware, Inc. **RabbitMQ - Messaging that just works**. 2025. <https://www.rabbitmq.com/>. Acesso em: 6 jun. 2025. Citado na página 30.

YAQUB, M. S. et al. An Ensemble Approach for IoT Firmware Strength Analysis using STRIDE Threat Modeling and Reverse Engineering. In: **2022 24th International Multitopic Conference (INMIC)**. [S.l.: s.n.], 2022. p. 1–6. Citado na página 40.

YOCAM, E. Narrow-band Internet of Things Protocol Standards: Survey of Security and Privacy Control Effectiveness. In: **2020 IEEE International Conference on Communications (ICC)**. [S.l.: s.n.], 2020. p. 1–6. Citado na página 39.

A APÊNDICE – COMANDOS E CONFIGURAÇÕES PARA O ESP32

A.1 Dump da Memória Flash com `esptool.py`

```
python -m esptool --port COM8 --baud 38400 read_flash 0x%OFFSET% 0x4000
part_%N%.bin
copy /b part_*.bin esp32_dump_consolidado.bin
strings esp32_dump_consolidado.bin > esp32_strings.txt
```

A.2 Leitura Completa da Flash com e sem Proteções

```
esptool.py --chip esp32 --port COM7 read_flash 0x00000 0x400000 esp32_
dump.bin
esptool.py --chip esp32 --port COM10 read_flash 0x00000 0x400000
tentativa_bloqueio.bin
strings tentativa_bloqueio.bin > saida_strings_seguro.txt
```

A.3 Verificação dos eFuses com `espefuse.py`

```
python -m espefuse --port COM10 summary
```

A.4 Substituição de Firmware via UART

```
idf.py build
esptool.py --chip esp32 --port COM10 write_flash 0x10000 build/hello_world.bin
```

A.5 Geração de Chave de Assinatura do Secure Boot

```
espsecure.py generate_signing_key secure_boot_signing_key.pem
```

A.6 Comandos `openssl` para Gerar Certificados TLS no Orange Pi

```
openssl req -x509 -newkey rsa:2048 -keyout mosquitto.key -out mosquitto.crt
-days 365 -nodes
```

A.7 Configuração TLS do Mosquitto Broker

```
listener 8883
protocol mqtt
cafile /etc/mosquitto/certs/mosquitto.crt
certfile /etc/mosquitto/certs/mosquitto.crt
keyfile /etc/mosquitto/certs/mosquitto.key
allow_anonymous true
```

```
require_certificate false
```

A.8 Captura de Pacotes com `tcpdump`

```
sudo tcpdump -i wlan0 port 8883 -w mqtt_tls_capture.pcap
```

A.9 Exemplo de Certificado Embutido no Código Arduino (ESP32)

```
espClient.setCACert(ca_cert);
```