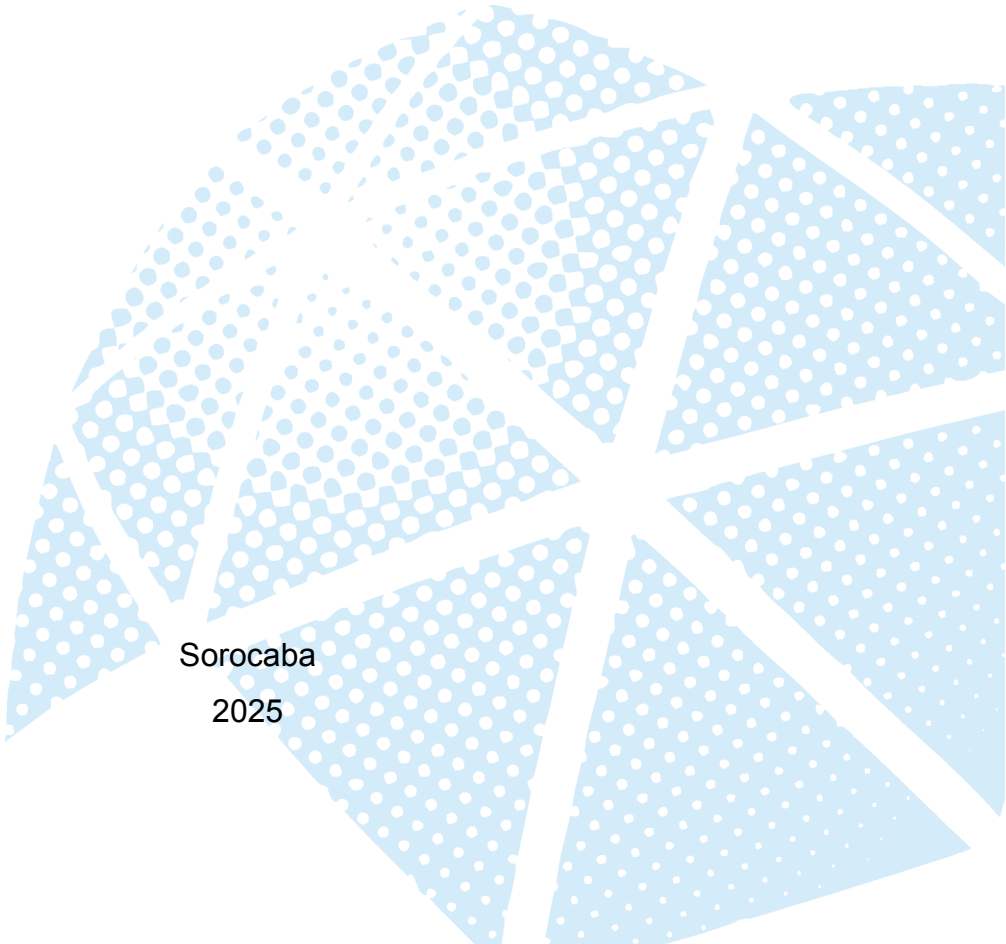


UNIVERSIDADE ESTADUAL PAULISTA - UNESP
Instituto de Ciência e Tecnologia - Campus de Sorocaba

GUILHERME PULIZZI COSTA

Ecosistema para monitoramento ambiental de galpões no agronegócio



Sorocaba
2025

GUILHERME PULIZZI COSTA

Ecossistema para monitoramento ambiental de galpões no agronegócio

Dissertação apresentada à Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba, para obtenção do título de Mestre em Engenharia Elétrica.

Área de Concentração: Sistemas Eletrônicos

Orientador(a): Prof. Dr. Flávio José de Oliveira Moraes

Coorientador(a): Prof. Dr. Luiz Fernando Pinto de Oliveira

Sorocaba

2025

C837e Costa, Guilherme
Ecossistema para Monitoramento Ambiental de Galpões no Agronegócio /
Guilherme Costa. -- Sorocaba, 2025
83 p. : tabs., fotos

Dissertação (mestrado) - Universidade Estadual Paulista (UNESP), Instituto
de Ciência e Tecnologia, Sorocaba
Orientador: Flávio José de Oliveira Morais
Coorientador: Luiz Fernando Pinto de Oliveira

1. Internet das coisas. 2. Monitoramento ambiental. 3. Interface de
programação de aplicativos (software de computador). 4. Algoritmos de
computador. 5. Armazens gerais. I. Título.

Impacto potencial desta pesquisa

Esta pesquisa apresenta um ecossistema baseado em *Internet of Things (IoT)* para monitoramento e controle ambiental em galpões agrícolas, priorizando baixo custo, automação inteligente e eficiência energética. A proposta busca democratizar o acesso à tecnologia e impulsionar a inovação científica no agronegócio brasileiro.

Potential impact of this research

This research presents an Internet of Things (IoT)-based ecosystem for environmental monitoring and control in agricultural barns, prioritizing low cost, smart automation, and energy efficiency. The proposal seeks to democratize access to technology and drive scientific innovation in Brazilian agribusiness.

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: Ecossistema para monitoramento ambiental de galpões no agronegócio

AUTOR: GUILHERME PULIZZI COSTA

ORIENTADOR: FLÁVIO JOSÉ DE OLIVEIRA MORAIS

COORIENTADOR: LUIZ FERNANDO PINTO DE OLIVEIRA

Aprovado como parte das exigências para obtenção do Título de Mestre em Engenharia Elétrica, área: Sistemas Eletrônicos pela Comissão Examinadora:

Prof. Dr. FLÁVIO JOSÉ DE OLIVEIRA MORAIS (Participação Virtual)
Coordenadoria do Curso de Engenharia de Biossistemas / Faculdade de Ciências e Engenharia de Tupã - UNESP

Prof. Dr. LUÍS FERNANDO CAPARROZ DUARTE (Participação Virtual)
DAELE-CP / UTFPR

Prof. Dr. MARIO MOLLO NETO (Participação Virtual)
Docente do Departamento de Engenharia de Biossistemas / Faculdade de Ciências e Engenharia - FCE - UNESP - Tupã/SP

Sorocaba, 30 de maio de 2025

Dedico este trabalho,
À minha avó Cerlei, por ter me criado com amor e sido meu maior alicerce.
Ao Texugo, meu cachorro leal, que esteve ao meu lado nas incansáveis horas de trabalho.
À minha namorada Natalia, pelo amor, paciência e apoio incondicional.
A mim mesmo, pela determinação, pela persistência e por nunca deixar de acreditar.

Agradecimentos

Agradeço à Universidade Estadual Paulista (UNESP), pela infraestrutura, pelo ambiente de pesquisa e pelas oportunidades que possibilitaram o desenvolvimento deste trabalho de mestrado. O suporte institucional foi essencial para a realização deste projeto.

Ao meu orientador, Prof. Flavio, pela orientação técnica e científica, pelas discussões construtivas e pela confiança depositada em minha condução ao longo da pesquisa. Sua contribuição foi determinante para o amadurecimento deste trabalho.

Ao meu colega e amigo Geovane Sakata, pela troca constante de ideias, apoio nos momentos de desafio e colaboração direta em etapas importantes deste estudo. Sua parceria contribuiu de forma concreta para o progresso da pesquisa.

Ao Prof. Cleber Amorim e ao Prof. Mario Mollo, pelos *feedbacks* valiosos e contribuições pontuais durante a elaboração deste trabalho, que ajudaram a aprimorar o conteúdo técnico e estrutural da pesquisa.

Ao meu amigo Ricardo Riedo, pelos ensinamentos técnicos, especialmente nas áreas de desenvolvimento de software, que tiveram impacto na construção prática e funcional do sistema proposto. Sua generosidade em compartilhar conhecimento foi fundamental.

Por fim, agradeço a mim mesmo, pela dedicação constante, pela resiliência diante dos desafios e pelo empenho em conduzir este trabalho com propósito e responsabilidade, buscando sempre contribuir de forma relevante para o avanço da ciência brasileira.

*"Ideas don't come out fully formed.
They only become clear as you work on them.
You just have to get started."
(Mark Zuckerberg, Harvard Commencement Speech, 2017)*

Resumo

Na era atual das práticas de gestão agrícola, conhecida como Agricultura 5.0, ambientes internos ideais estão associados a temperaturas confortáveis, umidade regulada e boa qualidade do ar — variáveis essenciais para melhorar os rendimentos. Diante desse cenário, surge a necessidade de ecossistemas inovadores que automatizem o monitoramento ambiental interno de forma acessível e escalável. Este trabalho apresenta o escopo do desenvolvimento e validação de um ecossistema baseado em *Internet of things (IoT)*, projetado para monitorar e controlar as condições ambientais em galpões agrícolas. O objetivo é apresentar um sistema de monitoramento ambiental de baixo custo e fácil acesso para construções rurais e áreas de armazenamento agrícola, promovendo o bem-estar de animais, humanos e culturas, e contribuindo para o desenvolvimento sustentável da indústria agrícola. O sistema integra sensores sem fio, algoritmos preditivos, uma interface web e infraestrutura em nuvem para otimizar temperatura e umidade. Uma avaliação de prova de conceito foi realizada para verificar se a arquitetura modular oferece escalabilidade, enquanto a interface web responsiva garante acessibilidade em diferentes dispositivos. Os resultados mostram precisão dos dados acima de 95%, eficiência preditiva de 96% e aumentos nos rendimentos da produção. Essa solução demonstra vantagens econômicas e operacionais em relação às tecnologias existentes, promovendo a sustentabilidade e a automação nas práticas de gestão agrícola em galpões e armazéns, em alinhamento com os Objetivos de Desenvolvimento Sustentável (ODS) das Nações Unidas.

Palavras-chave: INTERNET DAS COISAS; MONITORAMENTO AMBIENTAL; AUTOMAÇÃO AGRÍCOLA; ALGORITMOS PREDITIVOS.

Abstract

In the current era of agricultural management practices, known as agricultural 5.0, optimal indoor environments are associated with comfortable temperatures, regulated humidity, and good air quality—essential variables to improve yields. Given this scenario, there is a need for innovative ecosystems that automate indoor environmental monitoring in an affordable and scalable way. This work presents the scope of the development and validation of an Internet of things (IoT)-based ecosystem designed to monitor and control environmental conditions in agricultural barns. The objective is to present a cost-effective and easily accessible environmental monitoring system for barn buildings and agricultural storage areas, promoting the welfare of animals, humans, and crops, and contributing to the sustainable development of the agricultural industry. The system integrates wireless sensors, predictive algorithms, a web interface and cloud infrastructure to optimize temperature and humidity. A proof-of-concept assessment was performed to determine whether the modular architecture offers scalability, while the responsive web interface ensures cross-device accessibility. The results show data accuracy above 95%, prediction efficiency of 96%, and increases in production yields. This solution demonstrates economic and operational advantages over existing technologies, promoting sustainability and automation in agricultural management practices in hangars and barns, in alignment with the United Nations' Sustainable Development Goals (SDGs).

Keywords: INTERNET OF THINGS; ENVIRONMENTAL MONITORING; AGRICULTURAL AUTOMATION; PREDICTIVE ALGORITHMS.

Lista de Figuras

Figura 2.1 – Exemplo de rede de sensores sem fio com múltiplos saltos (<i>Multi-hop</i>). Os nós sensores trocam informações entre si até que os dados cheguem ao nó sorvedouro, que então os encaminha para a estação base.	17
Figura 2.2 – Arquitetura de uma rede <i>Multilayer Perceptron (MLP)</i> com uma camada de entrada, uma camada intermediária (oculta) e uma camada de saída.	24
Figura 2.3 – Representação de um neurônio artificial, com entradas ponderadas, função de soma e função de ativação. Adaptado de [1].	25
Figura 2.4 – Representação do processo de aprendizado supervisionado. Adaptado de [2].	26
Figura 3.1 – Arquitetura do sistema para coleta de dados dos sensores e controle dos atuadores.	36
Figura 3.2 – Fluxo de trabalho da interação entre o sistema de sensores e o servidor.	38
Figura 3.3 – Arquitetura de banco de dados para gerenciamento de dados do ecossistema.	43
Figura 3.4 – Arquitetura detalhada do ecossistema e fluxo de dados.	44
Figura 3.5 – Fluxo de funcionamento da <i>API</i> de recebimento e processamento de dados.	47
Figura 3.6 – Fluxo da <i>API</i> de Treinamento para algoritmos preditivos.	49
Figura 4.1 – <i>Printed Circuit Board (PCB)</i> — vistas superior e inferior.	54
Figura 4.2 – Implantação do sistema: (a, b) mostram diferentes visões do hardware instalado e funcionando dentro do ambiente controlado, (a) Hardware instalado e operando no ambiente controlado (visão 1), (b) Hardware instalado e operando no ambiente controlado (visão 2).	55
Figura 4.3 – Perfil de consumo de corrente do <i>ESP32</i> durante o acionamento, comunicação e retorno ao modo de <i>deep sleep</i>	56
Figura 4.4 – Perfil de consumo de corrente do <i>MSP430FR2355</i> em operação de ultra-baixo consumo.	56
Figura 4.5 – Visão geral das interações entre as APIs: (a) API de Processamento de Dados recebendo dados dos sensores da <i>SSN</i> e respondendo com comandos de controle dos atuadores ou sinalizadores de treinamento; (b) Chamadas e respostas da API de Treinamento, ilustrando o processo de treinamento e o retorno do status de treinamento atualizado para o sistema.	58
Figura 4.6 – Design responsivo adaptando-se a múltiplos tipos de dispositivos.	59

Figura 4.7 – Gráfico dinâmico exibindo tendências nos dados dos sensores e meteorológicos ao longo de um intervalo de tempo selecionado.	60
Figura 4.8 – Funcionalidades de visualização de dados e relatórios, incluindo o monitoramento de erros.	61
Figura 4.9 – Interface de controle dos atuadores com configurações manuais e automáticas.	62
Figura 4.10 – Interface para configuração dos algoritmos de predição, destacando opções flexíveis de <i>machine learning</i>	63
Figura 4.11 – Estratégia de <i>caching</i> para melhorar o tempo de resposta e reduzir os custos com <i>AWS</i>	64
Figura 4.12 – Distribuição dos erros no sistema de monitoramento durante o período de coleta de dados.	67
Figura 4.13 – Acurácia do modelo <i>Random Forest</i> ao longo do tempo (média: 96,46%).	68
Figura 4.14 – Acurácia do modelo <i>RNN</i> ao longo do tempo (média: 96,24%). . . .	69

Lista de Tabelas

Tabela 3.1 – Requisitos do Trabalho de Pesquisa	31
Tabela 3.2 – Riscos do projeto e estratégias de tratamento.	32
Tabela 3.3 – Limitações do sistema e soluções propostas.	33
Tabela 3.4 – Cotação do preço unitário do sensor <i>DHT22</i> para pequenas quantidades (em 2 de abril de 2025).	34
Tabela 3.5 – Cotação do preço unitário do ESP32 para pequenas quantidades (em 2 de abril de 2025).	35
Tabela 3.6 – Comparação de consumo de energia entre MSP430FR2355 e ESP32 em diferentes modos de operação.	35
Tabela 3.7 – Cotação do preço unitário do <i>MSP430FR2355</i> para pequenas quantidades (em 2 de abril de 2025).	36
Tabela 3.8 – Cotação de sensores de gás para aplicações de monitoramento ambiental (em 2 de abril de 2025).	37
Tabela 4.1 – Resumo da fatura da <i>AWS</i> para outubro de 2024.	66
Tabela 4.2 – Métricas de desempenho dos modelos preditivos (conjunto de teste: 07/2024 – 09/2024).	67

Sumário

1	Introdução	13
1.1	Objetivo	14
2	Fundamentação Teórica	16
2.1	Realidade Agronegócio Nacional	16
2.2	Rede de Sensores Sem Fio	16
2.3	Rede de sensores sem fio aplicados na agricultura de precisão	17
2.4	Arquitetura de Software e Interfaces: Fundamentos para Sistemas Web Baseados em APIs	18
2.4.1	Arquitetura de Software Limpa: Separação de Responsabilidades e Independência Tecnológica	18
2.4.2	Projeto de Software Orientado ao Domínio: Modelagem Centrada no Problema	19
2.4.3	Interfaces de Programação de Aplicações: Contrato entre Componentes Distribuídos	20
2.4.4	Interface Web: Acesso, Visualização e Controle	21
2.5	<i>Internet of Things (IoT)</i> e o paradigma da Tecnologia 5.0	22
2.6	Armazenamento em nuvem e o paradigma da Tecnologia 5.0	22
2.7	Aprendizagem de Máquina	23
2.8	Aplicação de Controle de Ambiência no Agronegócio	27
2.9	Fundamentos de Gestão de Projetos: Abordagens Tradicional e <i>SCRUM</i>	28
2.9.1	Modelo Tradicional: Abordagem Estruturada do PMBOK	28
2.9.2	Modelo <i>Ágil</i> : Flexibilidade com o Guia SBOK™	28
2.9.3	Integração das Abordagens: Híbrido como Alternativa	29
3	Materiais e Métodos	30
3.1	Definição de Requisitos, Riscos e Limitações	30
3.2	Requisitos de <i>Hardware</i>	33
3.3	Seleção dos Componentes de <i>Hardware</i>	34
3.4	Fluxo de Trabalho do <i>Hardware</i>	37
3.5	Montagem do Hardware	38
3.6	Desenvolvimento do Firmware	39
3.7	Seleção de Algoritmos Preditivos	39
3.8	Seleção do Provedor de Serviços em Nuvem	40
3.8.1	<i>Amazon Web Services (AWS)</i>	41
3.8.2	<i>Microsoft Azure</i>	41
3.8.3	<i>Google Cloud Platform (GCP)</i>	41
3.9	Definição da Arquitetura de Banco de Dados	41
3.10	Arquitetura e Requisitos da <i>API</i>	43
3.10.1	<i>API</i> de Recebimento e Processamento de Dados	44

3.10.2	<i>Database</i> e Armazenamento em Nuvem	48
3.10.3	<i>API</i> de Treinamento de Algoritmos Preditivos	48
3.10.4	<i>Desenvolvimento da API</i>	51
3.11	Arquitetura e Requisitos da Interface <i>Web</i>	52
3.11.1	<i>Front-End</i> (Interface do Usuário)	52
3.11.2	<i>Back-End</i> (Interação com o Banco de Dados)	52
3.11.3	Requisitos Funcionais	52
3.11.4	Requisitos Não Funcionais	53
4	Resultados e Discussão	54
4.1	Resultados da Validação do <i>Hardware</i>	54
4.2	Resultados da Validação de Software	57
4.2.1	Validação da <i>API</i>	57
4.2.2	Resultados da Validação da Interface <i>Web</i>	58
4.3	Ecossistema Geral	64
4.4	Validação dos Algoritmos Preditivos	67
5	Trabalhos Futuros	70
6	Conclusão	72
	Referências	73

1 Introdução

Na fase atual da gestão agrícola, conhecida como Agricultura 5.0, a integração de tecnologias da *Indústria 5.0*, como a *Internet das Coisas (IoT)*, análise de dados e automação, está revolucionando o agronegócio ao aumentar a eficiência e a precisão no gerenciamento ambiental [3, 4, 5]. Essas inovações possibilitam uma utilização mais inteligente dos recursos e maiores rendimentos em diversas operações agrícolas [6, 7, 8, 9, 10].

Os sistemas de *IoT* têm impacto especial na produção animal, onde monitoram e controlam condições ambientais para reduzir os riscos de doenças e melhorar o bem-estar animal, aumentando, assim, a produtividade [6]. Instalações agrícolas inteligentes, como aviários, estábulos leiteiros e estufas, integram *IoT* com *machine learning*, *blockchain* e arquiteturas de computação em nuvem e na borda (*edge computing*) para melhorar a confiabilidade dos sistemas e manter condições ideais para animais e culturas [11, 12, 13, 14]. Estruturas de *IoT* validadas em fazendas reais demonstraram alta eficácia no monitoramento de temperatura e umidade — variáveis-chave para o bem-estar animal —, além de permitirem a detecção de concentrações de gases nocivos como a amônia [15].

De forma semelhante, também na produção vegetal, a *IoT* facilita a irrigação e adubação precisas, alinhando os recursos às necessidades das culturas para maximizar os rendimentos [4].

A aplicação da *IoT* ao longo de todo o ciclo de vida dos produtos agrícolas, como proposto em estruturas de *green IoT*, potencializa a sustentabilidade e a produtividade em diversas práticas agrícolas [16]. Além disso, tecnologias sem fio otimizam as condições de armazenamento ao monitorar, por exemplo, temperatura, umidade e níveis de gases, preservando a qualidade das colheitas e minimizando perdas [17, 18, 19].

Os avanços nos sistemas de *IoT* permitem o monitoramento de condições ambientais, fornecendo *insights* baseados em dados para mitigar os riscos associados às práticas agrícolas e abrindo caminho para tecnologias emergentes como *Digital Twins*, que criam réplicas virtuais dos ambientes agrícolas reais [20, 21].

No entanto, a efetividade desses sistemas depende não apenas da coleta de dados, mas também de interfaces e experiências de usuário (*UX/UI*) intuitivas, que permitam aos agricultores e demais envolvidos visualizar informações de forma clara, interpretar tendências e tomar decisões informadas [22, 15]. Nesse contexto, ambientes internos ideais ou instalações de confinamento — associados a temperaturas confortáveis, umidade regulada e boa qualidade do ar, variáveis essenciais para melhorar os rendimentos — apresentam desafios intrínsecos à implementação de sistemas de monitoramento ambiental acessíveis e escaláveis. Esses desafios incluem complexidade, processamento de dados, portabilidade, integração de sensores, calibração, requisitos de energia, acessibilidade limitada e custos de manutenção [23]. Embora a literatura registre avanços no monitoramento de parâmetros ambientais como temperatura, umidade e comportamento animal, muitas implementações

ainda são limitadas em automação, escalabilidade e eficiência energética, evidenciando a necessidade de soluções agrícolas inteligentes mais integradas e sustentáveis [24].

Na África Ocidental, por exemplo, a produção agrícola enfrenta inúmeros desafios, incluindo controle ambiental inadequado e infraestrutura subdesenvolvida — fatores que dificultam a produtividade e a sustentabilidade. Uma revisão centrada no Senegal identificou restrições estruturais e climáticas como os principais entraves para a melhoria do desempenho nos sistemas agrícolas e pecuários, especialmente em contextos extensivos e agropecuários [25]. Esses achados ressaltam a necessidade de soluções tecnológicas acessíveis, adaptáveis e de baixo custo, capazes de apoiar produtores agrícolas em diferentes regiões e escalas operacionais.

Para enfrentar esse problema, este estudo apresenta o *FarmSync*, um ecossistema baseado em *IoT* projetado para monitorar e controlar parâmetros de condições ambientais, começando pelas variáveis microclimáticas de temperatura e umidade em galpões agrícolas. O objetivo e a inovação são demonstrar que é possível implementar um sistema de monitoramento ambiental de baixo custo e fácil acesso para instalações agrícolas, promovendo o bem-estar de culturas, animais e humanos, ao mesmo tempo em que contribui para o desenvolvimento sustentável da indústria agrícola. Projetado para ser flexível, o *FarmSync* permite integração com qualquer hardware, desde que este consiga se comunicar com *Interfaces de Programação de Aplicações (APIs)* por meio de requisições *HTTP*. Ao aproveitar infraestrutura em nuvem, algoritmos preditivos e atuadores automatizados, o sistema foi concebido para otimizar o consumo de energia e manter condições ambientais ideais conforme as variáveis microclimáticas evoluem. Além disso, uma interface web permite visualização de dados, emissão de relatórios e controle direto dos atuadores.

O *FarmSync* responde à necessidade por práticas agrícolas sustentáveis, alinhando-se aos princípios *ESG* e aos Objetivos de Desenvolvimento Sustentável (*ODS*) das Nações Unidas, como *Fome Zero*, *Ação Climática* e *Consumo Responsável* [26]. Ao suprir lacunas nos sistemas de gestão de galpões, o ecossistema proposto, movido por *IoT*, utiliza ciência para desenvolver eficiência energética, democratizar a tecnologia e capacitar agricultores de pequeno e médio porte.

1.1 Objetivo

O objetivo principal desta pesquisa é investigar a viabilidade de um ecossistema baseado em *IoT* para monitoramento ambiental e controle preditivo em galpões agrícolas. Especificamente, este estudo visa:

1. Avaliar a efetividade de redes de sensores de baixo custo na monitoração precisa de parâmetros ambientais-chave, como temperatura e umidade, em ambientes agrícolas confinados.
2. Desenvolver e validar modelos preditivos de *machine learning* capazes de prever condições microclimáticas com base em dados de sensores e informações meteorológicas externas.
3. Avaliar a capacidade de estratégias de controle preditivo para automatizar respostas de atuadores, mantendo estados ambientais ideais nos galpões.
4. Analisar a escalabilidade e confiabilidade de uma infraestrutura em nuvem projetada para armazenar, processar e gerenciar dados ambientais, dando suporte à expansão do sistema para múltiplos galpões e instalações agrícolas.

-
5. Desenvolver e validar uma interface web centrada no usuário, que permita o monitoramento, a análise de dados históricos e o controle manual ou automatizado dos atuadores ambientais.
 6. Quantificar métricas de desempenho do sistema, estabelecendo como metas uma acurácia de coleta de dados de pelo menos 95% e uma eficiência dos algoritmos preditivos de pelo menos 80%. Essas metas são sustentadas por estudos anteriores que demonstraram acurácia acima de 90% em dados de sensores em ambientes agrícolas [27] e previsões superiores a 80% na previsão de temperatura e umidade usando *machine learning* [28].

2 Fundamentação Teórica

2.1 Realidade Agronegócio Nacional

A agricultura familiar representa cerca de 77% dos estabelecimentos agropecuários no país, mas ocupa apenas 23% da área total disponível para a produção agrícola, de acordo com o Censo Agropecuário 2017 do IBGE [29]. Essa disparidade demonstra a concentração fundiária, que ainda persiste como um dos principais desafios estruturais para os pequenos produtores, impactando diretamente a produtividade e, consequentemente, sua capacidade de gerar renda.

Em relação à renda, os pequenos produtores enfrentam dificuldades financeiras. De acordo com o Censo Agropecuário de 2017, a renda média mensal desses agricultores varia entre R\$1.000 e R\$2.500, dependendo da região e do tipo de cultivo [29].

Diversos fatores contribuem para essa limitação de renda, como a falta de acesso a crédito adequado, a escassez de assistência técnica qualificada e as adversidades climáticas que tornam a produção agrícola altamente vulnerável. Essas dificuldades aumentam a volatilidade dos rendimentos e dificultam a adoção de práticas mais eficientes e sustentáveis [30].

Além disso, a dependência dos pequenos produtores de mercados locais e regionais, assim como a falta de infraestrutura para agregar valor aos produtos, reduz a produtividade. Políticas públicas, como o Programa Nacional de Fortalecimento da Agricultura Familiar (Pronaf), fornecem suporte financeiro e capacitação técnica. Contudo, muitos produtores ainda enfrentam dificuldades para acessar esses benefícios, devido à burocracia ou falta de informação [30].

2.2 Rede de Sensores Sem Fio

No decorrer dos últimos anos, vem ocorrendo um grande avanço tecnológico nas áreas de circuitos integrados, sensores e comunicação sem fio. Nesse sentido, foi formulada uma rede de sensores sem fio (*RSSF*), que pode ser aplicada em diferentes contextos, utilizando princípios de monitoramento, rastreamento, coordenação e processamento [31].

Desse modo, as *RSSFs* possibilitam, por exemplo, interconectar sensores a fim de monitorar e controlar as condições de um determinado ambiente. Esse novo conceito de rede de sensores tem a capacidade de revolucionar a coleta e o processamento de informações.

Uma rede de sensores sem fio normalmente é composta por um número de nós sensores (responsáveis pela captação de dados) e um nó sorvedouro (responsável por receber todos os dados captados dos nós sensores). Nesse sentido, todos os dados captados pelos nós

sensores são encaminhados pela rede até uma estação base, que é responsável por agrupar e organizar os dados provenientes do conjunto nó sensor e nó sorvedouro [32].

A estação base pode executar duas funções: encaminhar os dados captados pela rede a um computador controlador ou a algum dispositivo responsável pela manipulação dos dados e execução das tarefas de gerenciamento e controle; ou ela mesma processar os dados e realizar o gerenciamento da rede e dos atuadores do sistema [33].

A Figura 2 mostra um exemplo de uma rede de sensores sem fio com múltiplos saltos (*Multi-hop*), uma arquitetura amplamente utilizada em aplicações atuais. Nesse modelo, os nós sensores trocam informações entre si até que os dados sejam encaminhados ao nó sorvedouro, que por sua vez os envia para uma estação base [34].

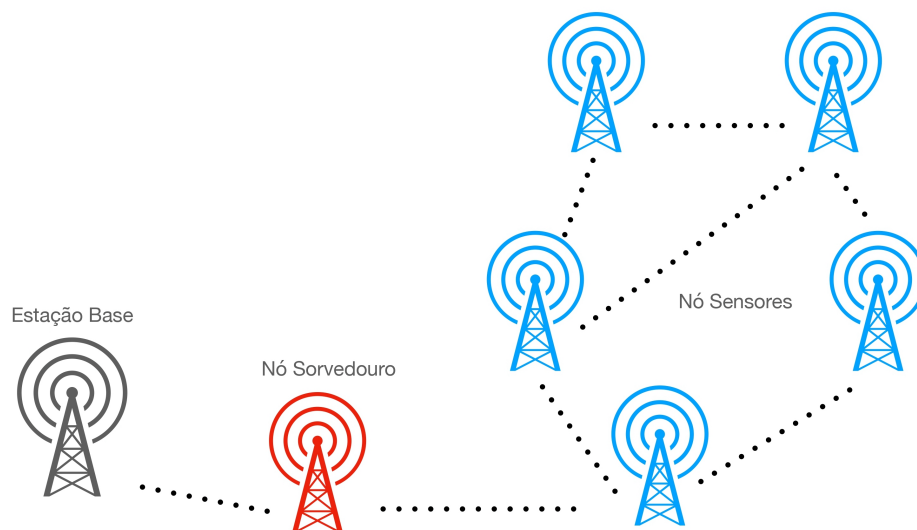


Figura 2.1: Exemplo de rede de sensores sem fio com múltiplos saltos (*Multi-hop*). Os nós sensores trocam informações entre si até que os dados cheguem ao nó sorvedouro, que então os encaminha para a estação base.

2.3 Rede de sensores sem fio aplicados na agricultura de precisão

No contexto ambiental, a utilização de redes de sensores sem fio tem chamado a atenção de diversos pesquisadores. Monitorar a biodiversidade em um determinado ecossistema com o uso de uma *RSSF* torna-se mais simples a partir do momento em que esse tipo de rede soluciona o problema da presença de grandes quantidades de fios espalhados pelo ambiente [35, 32, 36].

A agricultura de precisão apresenta-se como a área ambiental mais estudada para a aplicação de redes de sensores sem fio. A literatura investiga formas de aliar o conceito de *RSSF* ao monitoramento de variáveis relacionadas ao manejo agrícola, como temperatura, umidade, pressão, velocidade, presença, entre outras [37].

O uso de *RSSF* para evitar a propagação de fungos em plantações de batata, no cultivo e armazenamento de grãos de milho, e em plantações de eucalipto e pinhão, por meio do monitoramento de variáveis como temperatura e umidade, já é uma realidade em muitas fazendas ao redor do mundo [38].

Esse fato tem proporcionado diversos benefícios aos produtores, como a maximização dos lucros ao evitar grandes prejuízos causados por determinados fungos; o aumento da segurança do trabalho, uma vez que, no caso do monitoramento de grãos em grandes silos, elimina-se a necessidade de trabalhadores supervisionarem pessoalmente os armazéns, evitando riscos à integridade física; além da praticidade não apenas no controle da qualidade do produto, mas também no acesso remoto às informações, comprovando a eficiência do uso dessas redes em diversos aspectos [32].

2.4 Arquitetura de Software e Interfaces: Fundamentos para Sistemas Web Baseados em APIs

O desenvolvimento de sistemas distribuídos modernos, especialmente aqueles orientados a sensores e automação, exige uma abordagem disciplinada e escalável de engenharia de software. Neste contexto, o uso de boas práticas arquiteturais, como as propostas por [39], [40] e [41], oferece uma base conceitual sólida para a construção de aplicações resilientes, manuteníveis e de alto desempenho.

2.4.1 Arquitetura de Software Limpa: Separação de Responsabilidades e Independência Tecnológica

A arquitetura de software diz respeito à maneira como um sistema é organizado internamente, definindo como seus diferentes componentes interagem entre si [41]. Para que um sistema seja sustentável ao longo do tempo — ou seja, fácil de modificar, testar e evoluir — é essencial que ele seja construído com base em princípios sólidos de organização [40].

Uma das abordagens mais reconhecidas nesse sentido é a **Arquitetura Limpa**, proposta por [39]. Essa arquitetura defende uma divisão clara entre as diferentes camadas que compõem um sistema, com especial ênfase na independência da lógica de negócio — isto é, das regras centrais que definem o funcionamento do sistema — em relação aos detalhes técnicos como banco de dados, frameworks de programação, bibliotecas externas ou mesmo a interface visual.

Na prática, isso significa que as decisões mais importantes de um sistema, aquelas que refletem a sua essência e propósito, não devem estar atreladas a tecnologias específicas. Por exemplo, imagine um sistema que controla a temperatura de um ambiente agrícola. A lógica que define quando um ventilador deve ser ativado com base na temperatura e umidade captadas por sensores deve estar isolada e bem protegida no núcleo do software. Já a forma como esses dados são recebidos (por rede Wi-Fi, USB ou Bluetooth), onde são armazenados (em nuvem, em banco local ou em arquivos) e como são apresentados ao usuário (interface web, aplicativo ou display físico) são detalhes secundários e podem mudar com o tempo sem afetar a estrutura principal do sistema [41, 40].

A Arquitetura Limpa propõe uma estrutura em camadas concêntricas, em que a lógica de negócio ocupa o centro e os detalhes técnicos ficam nas camadas mais externas [39]. Entre essas camadas, existem regras bem definidas: apenas as camadas externas podem depender das internas, nunca o contrário. Essa regra é conhecida como Regra da Dependência.

“A arquitetura limpa nos ensina que frameworks são apenas detalhes, e que

dependências devem sempre apontar para dentro, protegendo o coração da aplicação.” [39]

Esse princípio permite que partes do sistema sejam substituídas ou evoluídas de forma independente. Por exemplo, é possível trocar a base de dados utilizada ou mudar o tipo de dispositivo responsável pela coleta dos dados sem que isso impacte na lógica central do sistema [41].

Além disso, essa separação clara contribui significativamente para a testabilidade do sistema. Quando as regras de negócio estão isoladas, é possível testá-las com facilidade e de forma automatizada, sem depender de componentes externos como banco de dados ou interfaces gráficas [39, 40]. Isso torna o sistema mais confiável e reduz o tempo necessário para identificar e corrigir falhas.

Outro ponto importante é que a Arquitetura Limpa facilita a colaboração entre profissionais de diferentes áreas. Especialistas em hardware, por exemplo, podem trabalhar nas camadas responsáveis pela comunicação com sensores e atuadores, enquanto desenvolvedores de software concentram-se na implementação da lógica central do sistema [40]. Essa modularidade melhora a organização do trabalho e reduz os riscos de interferência indesejada entre áreas.

A adoção da Arquitetura Limpa em sistemas de automação permite garantir sua integridade mesmo diante da evolução tecnológica. A lógica que define o controle de atuadores, baseada na leitura de sensores e em algoritmos inteligentes, deve permanecer protegida no núcleo da aplicação, assegurando robustez, longevidade e facilidade de manutenção.

2.4.2 Projeto de Software Orientado ao Domínio: Modelagem Centrada no Problema

O desenvolvimento de sistemas que representam contextos complexos, como ambientes agrícolas, financeiros ou logísticos, demanda mais do que apenas boas práticas de codificação. Nesses casos, é essencial que o software seja projetado com base em uma compreensão profunda do domínio em que será aplicado. Essa é a premissa do *Domain-Driven Design* (DDD), ou "projeto orientado ao domínio", proposto por [40].

A abordagem do DDD propõe que o modelo conceitual do sistema — ou seja, a representação das entidades, regras, comportamentos e relações que compõem o problema — seja construído de forma colaborativa entre desenvolvedores e especialistas do domínio. Para isso, é fundamental o uso de uma **linguagem ubíqua**: um vocabulário comum, baseado nos termos e conceitos reais do contexto em que o sistema será aplicado. Essa linguagem é adotada tanto na comunicação da equipe quanto na própria implementação do código, promovendo alinhamento e clareza ao longo de todo o ciclo de vida do projeto.

No DDD, o modelo de domínio não é apenas um artefato de documentação, mas a base sobre a qual todas as decisões de projeto são tomadas. A estrutura do sistema é organizada em torno de elementos como **entidades** (objetos com identidade própria), **objetos de valor** (representações imutáveis de conceitos), **serviços** (operações relevantes que não pertencem a uma entidade específica) e **agregados** (conjuntos de objetos tratados como uma unidade lógica).

“O design orientado ao domínio é a ponte entre conhecimento e código, transformando a complexidade em estrutura compreensível e evolutiva.” [40]

Essa forma de organização favorece a separação de responsabilidades, a testabilidade e a capacidade de adaptação a mudanças no conhecimento do domínio. Sistemas construídos sob a perspectiva do DDD tendem a ser mais coesos, pois cada parte reflete de maneira explícita um conceito do mundo real, e mais fáceis de evoluir, já que a lógica de negócio é isolada dos detalhes técnicos.

Outro conceito importante no DDD é o de **contexto delimitado** (*bounded context*), que define fronteiras claras dentro das quais um modelo de domínio é válido. Essa delimitação permite que diferentes partes do sistema utilizem abordagens ou terminologias distintas sem gerar conflitos, o que é especialmente útil em sistemas distribuídos ou compostos por múltiplos subsistemas.

A integração dessa abordagem com arquiteturas modernas, como a Arquitetura Limpa, reforça os benefícios de modularidade e independência entre camadas, garantindo que a lógica central permaneça fiel ao problema que se propõe a resolver. Ao adotar o DDD, os sistemas ganham não apenas em organização estrutural, mas também em alinhamento com os objetivos reais do negócio que representam.

2.4.3 Interfaces de Programação de Aplicações: Contrato entre Componentes Distribuídos

No desenvolvimento de sistemas distribuídos, é comum que diferentes componentes precisem se comunicar para formar um sistema coeso e funcional. Para garantir essa comunicação, utilizam-se as chamadas **Interfaces de Programação de Aplicações** (IPAs), conhecidas internacionalmente como *Application Programming Interfaces* (APIs). Essas interfaces definem um conjunto padronizado de regras e formatos para que aplicações distintas possam trocar dados de maneira previsível, segura e desacoplada [41].

As IPAs baseadas no estilo arquitetural denominado REST, sigla para **Transferência Representacional de Estado** (*Representational State Transfer*), tornaram-se amplamente adotadas por sua simplicidade, escalabilidade e independência de plataforma. Elas operam sobre o Protocolo de Transferência de Hipertexto (HTTP — *Hypertext Transfer Protocol*), o mesmo protocolo utilizado para navegação na internet, e utilizam métodos padronizados como **GET** (para leitura de dados), **POST** (para criação), **PUT** (para atualização) e **DELETE** (para exclusão) [41]. Essa padronização permite que dispositivos heterogêneos — como sensores ambientais, servidores em nuvem e interfaces gráficas — interajam de forma unificada e previsível.

Segundo Eric Evans, em sua obra *Domain-Driven Design: Tackling Complexity in the Heart of Software*, a integração bem estruturada entre módulos de software depende diretamente da clareza nos contratos de comunicação. Nesse contexto, as IPAs atuam como fronteiras explícitas entre componentes, garantindo modularidade e permitindo que cada parte do sistema evolua de maneira independente, desde que respeite os contratos estabelecidos. Essa característica torna-se ainda mais relevante em sistemas orientados ao domínio, nos quais os serviços devem refletir regras e comportamentos reais do mundo modelado [40].

Além de promover interoperabilidade, as IPAs favorecem a reutilização de funcionalidades já existentes. Um mesmo serviço pode ser consumido por diferentes aplicações — como sistemas de monitoramento, aplicativos móveis ou interfaces de visualização — sem a necessidade de reimplementação. Isso reduz custos de desenvolvimento, melhora o tempo de entrega e aumenta a coesão entre os módulos do sistema [41, 39].

“Interfaces baseadas em serviços promovem reutilização, integração e modularidade — princípios centrais da engenharia de software moderna.” [41]

Outro benefício relevante das IPAs é sua capacidade de facilitar a documentação e o monitoramento dos serviços. Ferramentas como o padrão OpenAPI (Interface Aberta de Programação de Aplicações) permitem a geração automática de descrições formais das rotas disponíveis, métodos aceitos e estruturas de dados esperadas, o que facilita a integração com novos módulos e a manutenção da consistência durante a evolução do sistema [41].

Portanto, o uso de interfaces de programação bem definidas é um dos pilares do desenvolvimento de sistemas distribuídos modernos. Elas garantem flexibilidade arquitetural, promovem a separação de responsabilidades e possibilitam a construção de soluções escaláveis e aderentes aos princípios da engenharia de software contemporânea [39].

2.4.4 Interface Web: Acesso, Visualização e Controle

A interface web representa a camada de interação entre o sistema e seus usuários, sendo responsável por apresentar as informações de forma acessível, compreensível e funcional. Conforme discutido por [41], a usabilidade de uma aplicação depende não apenas da apresentação visual, mas também da clareza na disposição das informações, da consistência das interações e da capacidade de o usuário atingir seus objetivos com eficiência.

Em sistemas de monitoramento e automação, como os voltados ao controle ambiental, a interface web desempenha um papel essencial ao permitir a visualização em tempo real de variáveis críticas — como temperatura, umidade, pressão e funcionamento de atuadores — além de possibilitar ações diretas de controle. Para isso, [40] ressalta a importância da comunicação clara entre quem desenvolve e quem utiliza o sistema, o que reforça a necessidade de interfaces bem estruturadas.

A arquitetura adotada no desenvolvimento da interface web geralmente segue o princípio da separação entre a camada de apresentação e a de processamento. Em vez dos termos estrangeiros amplamente usados como *frontend* e *backend*, pode-se falar, respectivamente, em "camada de apresentação" e "camada de negócios". Essa separação, conforme argumenta [39], favorece o desacoplamento entre camadas, permitindo que alterações na interface visual não afetem a lógica central do sistema, e vice-versa.

Além disso, a interface deve ser concebida com foco em acessibilidade e adaptação a diferentes dispositivos, como computadores, tablets e telefones móveis. Isso está alinhado com os princípios de engenharia de software voltados à experiência do usuário, conforme apresentado por [41], que destaca a importância de sistemas que sejam intuitivos e responsivos.

Recursos atuais de desenvolvimento de interfaces permitem a utilização de bibliotecas e ferramentas reutilizáveis, que aceleram a construção de componentes visuais padronizados. A integração dessas tecnologias com os serviços definidos pelas Interfaces de Programação de Aplicações (IPAs) — detalhadas na seção anterior — transforma a interface web em um instrumento robusto de consulta, análise e controle [40, 39].

Em síntese, a interface web exerce a função de elo entre a complexidade interna do sistema e a simplicidade exigida pela experiência do usuário. Sua concepção cuidadosa é essencial para assegurar não apenas a funcionalidade do sistema como um todo, mas também a satisfação de quem interage com ele [41, 40, 39].

2.5 Internet of Things (IoT) e o paradigma da Tecnologia 5.0

Nos últimos anos, um novo paradigma denominado *Internet of Things (IoT)*, introduzido por Kevin Ashton em 1998, destacou-se tanto em aplicações acadêmicas quanto industriais. A computação onipresente, os protocolos de internet, sensores, tecnologias de comunicação e dispositivos embarcados estão cada vez mais integrados, formando um sistema em que o mundo real e o digital se interceptam e interagem de forma simbiótica [42, 43].

Na *IoT*, o objeto inteligente é o elemento central. Ao incorporar inteligência a objetos do cotidiano, estes passam a ser capazes não apenas de controlar e interagir com o mundo físico, mas também de se comunicarem entre si, utilizando a internet para a troca de dados e informações.

O conceito de *IoT*, composto por uma rede global de equipamentos (sensores, atuadores, dispositivos inteligentes, sistemas embarcados, etc.) habilitados para coletar dados, comunicar-se e trocar informações entre si, proporciona novas formas de interação. A *IoT* provê uma infraestrutura completa para a propagação dessas tecnologias tanto de forma local (*in situ*) quanto remota (*ex situ*) [44, 45].

Para que haja interação entre tais dispositivos, cada aparelho eletrônico possui uma identificação única armazenada em um banco de dados. Assim, ao se conectar a uma rede com acesso a esse banco de dados, o sistema é capaz de identificar e registrar os dispositivos presentes.

Os principais impactos gerados pela *IoT* refletem-se na vida cotidiana e no comportamento dos usuários. Para o usuário final, os efeitos mais evidentes estarão presentes não apenas nos ambientes de trabalho, mas também em residências. Moradias, instalações de armazenagem de grãos assistidas, escritórios inteligentes e ambientes de aprendizado aprimorado são alguns dos cenários previstos. Do ponto de vista empresarial, destacam-se aplicações nas áreas de automação e manufatura industrial, logística, gestão de processos de negócios, e transporte inteligente de pessoas, mercadorias e animais [46, 47].

Dessa forma, a tecnologia *IoT* apresenta características marcantes como alto grau de adaptação, autonomia e segurança dos dados. Contudo, os grandes desafios dessa tecnologia residem em prover aos dispositivos um alto grau de inteligência, bem como arquivar e processar a enorme quantidade de dados que eles geram [43].

2.6 Armazenamento em nuvem e o paradigma da Tecnologia 5.0

Neste cenário, a computação em nuvem (*cloud*) surge para agregar valor aos dispositivos da *IoT*, fornecendo armazenamento e processamento robustos para o grande volume de dados gerado por inúmeras aplicações [48, 49].

Segundo Puthal, Sahoo, Mishra e Swain [50], a computação em nuvem é um modelo que permite, sob demanda, o acesso conveniente à rede para um conjunto compartilhado de recursos de computação configuráveis (servidores, serviços, armazenamento, rede e aplicações), que podem ser fornecidos e liberados rapidamente e com facilidade [51].

Esse modelo é atrativo para as empresas, pois elimina a necessidade de planejamento antecipado de provisionamento por parte dos usuários, permitindo que a *cloud* cresça conforme a empresa se expande, otimizando o uso dos recursos de armazenamento.

Com a computação em nuvem, não é mais necessário instalar aplicativos ou fazer download de documentos em computadores locais, uma vez que os dados não estão armazenados em um local específico. Em vez disso, eles estão disponíveis na rede e podem ser acessados por qualquer dispositivo conectado à internet.

Em relação à confiabilidade e responsabilidade, o provedor de *clouds* deve garantir a proteção dos dados, impedindo acessos não autorizados e preservando a privacidade das informações, principalmente quando utilizadas fora do ambiente corporativo [52].

De forma geral, cada sistema de *cloud* possui sua própria política de privacidade de dados. Quando ocorre a movimentação de dados entre sistemas distintos, é necessário garantir a privacidade mesmo em ambientes com modelos de dados diferentes e aplicações multiusuário. Agrawal et al. [52] propõem uma abordagem segura para preservação da privacidade: em vez de empregar criptografia — técnica de alto custo computacional — é adotada uma estratégia baseada na distribuição de dados em múltiplos sítios do provedor e técnicas de acesso sigiloso [50].

Os serviços em nuvem atuam como processadores de dados da *Internet of Things*, originando um novo paradigma denominado *CloudIoT*. Embora a *CloudIoT* ainda seja um conceito recente e pouco explorado, já se observa que, no contexto de comunicação e dados, ela permite a entrega de aplicações ubíquas e personalizadas com baixo custo, tanto para coleta quanto para distribuição de dados. Além disso, a nuvem oferece um meio eficaz e econômico para conectar e gerenciar dispositivos de qualquer lugar, a qualquer momento, por meio de aplicações embarcadas [53, 50].

Na área de comunicação, os dispositivos da *IoT* geralmente não possuem capacidade de processamento local suficiente. Por isso, a nuvem fornece recursos computacionais virtualmente ilimitados e um modelo de uso sob demanda.

No campo do armazenamento, a *CloudIoT* satisfaz as necessidades da *IoT*, uma vez que a computação em nuvem possui capacidade para armazenar grandes volumes de dados. Nessas condições, esse conjunto massivo de informações também pode ser caracterizado como *Big Data*.

2.7 Aprendizagem de Máquina

A inteligência artificial (IA) pode ser entendida como uma abordagem para desenvolver sistemas computacionais que simulam o pensamento humano. Esse conceito envolve a análise de como as pessoas tomam decisões e resolvem problemas, decompondo esses processos em etapas fundamentais que podem ser replicadas por um programa de computador. Dessa forma, a IA oferece uma maneira estruturada e eficiente de projetar soluções complexas para a tomada de decisão [54].

Dentro dessa ampla área de estudo, que vem se desenvolvendo cada vez mais, existe um segmento denominado *Deep Learning*. Esse campo oferece uma ferramenta que ensina o computador a realizar tarefas que, até então, eram exclusivas dos seres humanos, baseadas na percepção. Os algoritmos de *Deep Learning* possuem uma arquitetura inspirada na IA, buscando emular o neocórtex do cérebro humano, área responsável pela absorção de aprendizado [55].

Esses algoritmos são projetados para automatizar a extração de características predominantes nos dados apresentados, permitindo que realizem altos níveis de abstração das informações e, assim, adquiram aprendizado de maneira autônoma [56].

Alguns algoritmos de aprendizado em *Deep Learning* operam com base em redes neurais. Uma rede neural comum para casos como esse, é chamada de *Multilayer Perceptron*

(*MLP*). Ela é composta por conjuntos de neurônios artificiais organizados em camadas: uma camada de entrada (com variáveis como $x_1, x_2, \dots$, que representam as características do problema), uma ou mais camadas intermediárias (ou ocultas), e uma camada de saída (com variáveis como y_1, y_2 , que representam os valores previstos). Um exemplo de MLP está representado na Figura 2.2 [1].

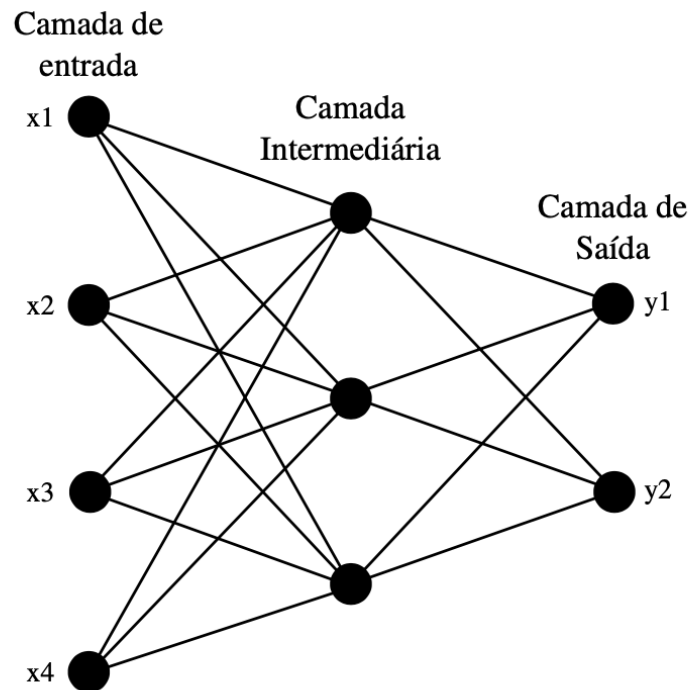


Figura 2.2: Arquitetura de uma rede *Multilayer Perceptron* (MLP) com uma camada de entrada, uma camada intermediária (oculta) e uma camada de saída.

Desenvolvido a partir do conceito de neurônio biológico, o neurônio artificial busca executar a mesma função: receber informações, processá-las e, em seguida, repassá-las de forma processada para outro neurônio. Cada entrada x_i é associada a um peso sináptico w_i , que define a importância daquela informação. A saída do neurônio é obtida pela soma ponderada das entradas, acrescida de um viés (bias), e transformada por uma função de ativação $g(\cdot)$, resultando em uma saída y . A representação esquemática pode ser vista na Figura 2.3 [57].

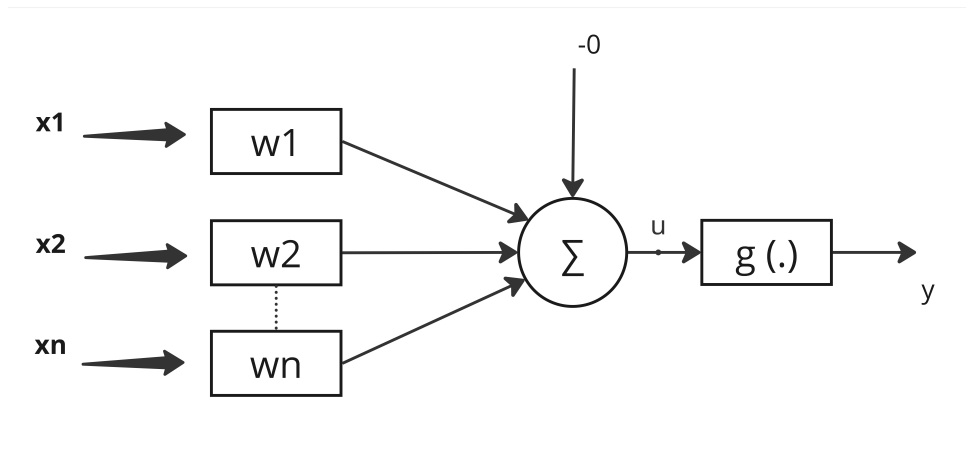


Figura 2.3: Representação de um neurônio artificial, com entradas ponderadas, função de soma e função de ativação. Adaptado de [1].

As redes *MLP* utilizam funções de ativação não lineares nas camadas intermediárias, como, por exemplo, a função sigmoide. Cada nó realiza uma função específica, e a função final é uma combinação das funções implementadas nos nós das camadas anteriores. Embora o princípio permaneça o mesmo, à medida que o processamento avança de uma camada para outra, as operações realizadas e suas funções se tornam mais complexas [58].

Cada classe que a *MLP* precisa segmentar é representada por um nó na camada de saída. O treinamento da rede ocorre quando o vetor que contém as respostas desejadas para cada objeto de entrada possui o valor 1 na posição associada à classe do objeto e 0 nas demais posições. A comparação entre o vetor dos nós da camada de saída e o vetor de valores desejados representa o erro cometido pela rede [59].

Existem várias formas de aplicar redes neurais, sendo uma delas o aprendizado de máquina supervisionado. Como o nome sugere, esse tipo de aprendizado ocorre com supervisão, ou seja, durante o treino, a máquina recebe o gabarito dos dados de treino para que a rede possa abstrair conhecimento e, posteriormente, gerar suas próprias respostas para novos dados. A Figura 2.4 ilustra esse processo de aprendizado supervisionado [2].

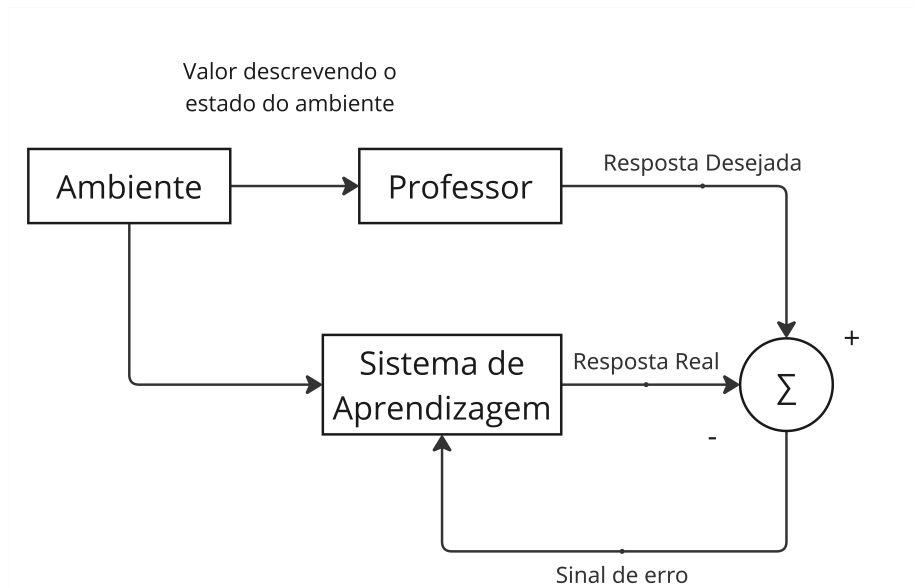


Figura 2.4: Representação do processo de aprendizado supervisionado. Adaptado de [2].

Um exemplo de algoritmo utilizado para treinamento com aprendizado supervisionado é o *backpropagation*, que é o principal algoritmo para o treinamento de redes *MLP*. Seu funcionamento envolve iterações em duas fases: propagação (para frente) e retropropagação (para trás) [60].

Na primeira fase, a de propagação, cada dado de entrada é apresentado à rede. O objetivo é receber essas informações nos neurônios das camadas intermediárias. Esses dados são ponderados pelos pesos associados às conexões de entrada dos neurônios, que representam o aprendizado adquirido pela rede. O valor resultante é então processado pelas funções de ativação, e a saída produzida é usada como entrada para o próximo neurônio na camada seguinte [61].

Esse processo é repetido até os neurônios da camada de saída. Os valores gerados são comparados com a saída desejada (gabarito), e a diferença entre os valores indica o erro cometido pela rede. Na fase de retropropagação, esse valor de erro é usado para ajustar os pesos da rede, com o objetivo de reduzir o erro a cada iteração [58].

Random Forest (RF) e *Recurrent Neural Networks (RNN)* são técnicas de aprendizado de máquina.

Random Forest é um método de aprendizado em conjunto, utilizado principalmente para tarefas de classificação e regressão. Ele opera construindo várias árvores de decisão durante o treinamento e resulta na moda das classes (para classificação) ou na média das previsões das árvores individuais (para regressão). O RF é particularmente útil para lidar com dados não lineares e conjuntos de dados de alta dimensionalidade. O algoritmo é conhecido por sua capacidade de lidar com dados ausentes e por ser robusto contra o *overfitting*, especialmente quando aplicado a conjuntos de dados grandes e complexos. A análise de importância de atributos realizada pelo RF é outro ponto positivo, permitindo identificar os fatores mais influentes nas previsões [62, 63].

As *Recurrent Neural Networks (RNNs)*, por outro lado, são uma classe de algoritmos de *deep learning* projetados para processar dados sequenciais. Elas são especialmente úteis para previsões de séries temporais, pois levam em consideração as dependências temporais entre os pontos de dados. As RNNs possuem uma arquitetura que permite reter

informações de entradas anteriores, tornando-as ideais para prever eventos futuros com base em dados históricos. No monitoramento ambiental, como na previsão de poluentes, as RNNs superam os modelos tradicionais ao prever com maior precisão a qualidade do ar com base em tendências passadas [63].

Ambos os algoritmos, quando utilizados em conjunto, podem complementar seus pontos fortes. Enquanto o RF fornece interpretabilidade para lidar com conjuntos de dados complexos, a RNN se destaca na previsão de dados sequenciais [64].

2.8 Aplicação de Controle de Ambiência no Agronegócio

Este tópico explora o uso de tecnologias para monitoramento e controle em fazendas de aves, destacando *IoT*, sensores ambientais e aprendizado de máquina. Três estudos são apresentados, mostrando como essas soluções otimizam o bem-estar animal e a eficiência produtiva.

O artigo de Ali et al. [65] descreve o desenvolvimento de um sistema inteligente para monitoramento de fazendas de criação de aves, com o objetivo de controlar o ambiente do galinheiro utilizando sensores e tecnologia *IoT*. A plataforma em nuvem Blynk foi empregada para gerenciar os dados, recebidos por meio do módulo Wi-Fi do Arduino WeMo D1. Os dados de peso, temperatura e umidade são enviados e armazenados na nuvem, permitindo o acesso remoto e o monitoramento em tempo real das atividades no galinheiro.

Diversos sensores foram utilizados para coleta de dados relevantes, como temperatura, umidade e peso das aves. O sensor *DHT11* monitora temperatura e umidade de forma contínua, garantindo que os produtores mantenham um ambiente adequado. Uma célula de carga foi empregada para estimar o peso das aves, permitindo acompanhar o crescimento e determinar o momento ideal para o abate.

O sistema oferece aos produtores acesso remoto às condições do ambiente, facilitando a tomada de decisões e permitindo ajustes proativos. Com a integração dos sensores e o uso de *IoT*, o sistema contribui para ações preventivas, melhora da eficiência produtiva e garantia do bem-estar animal.

O projeto de Budiarto et al. [66] apresenta uma arquitetura de monitoramento e controle para fazendas de frangos, utilizando o *Raspberry Pi 3* como servidor central. Este dispositivo coleta e processa dados de sensores como o *DHT22* (temperatura e umidade), sensor de peso (quantidade de ração) e o *MQ135* (nível de amônia). Com base nessas leituras, decisões automatizadas são tomadas.

O monitoramento da umidade, temperatura e alimentação evita escassez e garante condições adequadas para o desenvolvimento dos frangos. O sensor *MQ135*, por sua vez, assegura o controle da qualidade do ar nas instalações, aspecto fundamental para a saúde das aves.

O sistema também incorpora comunicação via *GPRS* (Serviço de Rádio de Pacote Geral), permitindo a transmissão dos dados coletados em tempo real para servidores ou dispositivos móveis. Isso possibilita o controle ambiental e de suprimento alimentar com maior precisão e agilidade.

O estudo de Ribeiro et al. [67] abordou o controle ambiental de criadouros de aves utilizando técnicas de aprendizado de máquina supervisionado, especificamente Redes Neurais Artificiais (*RNA*). As decisões relacionadas à temperatura e ventilação foram

calibradas com base no conhecimento de especialistas, buscando otimizar o ambiente dos galpões. Com essa abordagem, foi possível ajustar automaticamente as ações do sistema com base nas condições ideais para o bem-estar e desenvolvimento das aves.

2.9 Fundamentos de Gestão de Projetos: Abordagens Tradicional e Scrum

O gerenciamento de projetos envolve o planejamento, a execução, o controle e o encerramento de iniciativas com objetivos bem definidos, dentro de restrições de tempo, custo e qualidade. A aplicação estruturada de boas práticas e metodologias de gestão tem se mostrado essencial em ambientes complexos e tecnologicamente intensivos, como no caso de sistemas baseados em Internet das Coisas (IoT). Neste capítulo, apresenta-se uma visão abrangente das duas principais abordagens: a tradicional, baseada no Guia PMBOK®, e a *ágil*, fundamentada no Guia SBOK™ [68, 69].

2.9.1 Modelo Tradicional: Abordagem Estruturada do PMBOK

Segundo o Project Management Institute [68], a gestão tradicional de projetos – amplamente conhecida como abordagem em cascata – organiza os projetos em cinco grupos de processos: Iniciação, Planejamento, Execução, Monitoramento e Controle, e Encerramento. Esses processos são aplicados por meio de áreas de conhecimento, como gestão de escopo, tempo, custo, qualidade, riscos, recursos humanos e aquisições.

Essa abordagem é caracterizada pelo detalhamento inicial extensivo, em que requisitos, entregas e cronogramas são definidos previamente. Isso proporciona alto controle e previsibilidade, sendo eficaz em projetos com baixa variabilidade. No entanto, sua rigidez pode ser desafiadora em ambientes que exigem respostas rápidas a mudanças.

Grupos de Processos e Áreas de Conhecimento

Cada grupo de processos contempla atividades distintas. Na iniciação, define-se a justificativa do projeto e nomeia-se o gerente de projetos. O planejamento estabelece o escopo, cronograma, recursos e riscos. A execução realiza as tarefas planejadas. O monitoramento acompanha o desempenho e realiza ajustes. O encerramento formaliza a conclusão do projeto [68].

As áreas de conhecimento fornecem subsídios para que esses processos sejam realizados com qualidade e consistência. Por exemplo, a gestão de escopo garante que somente o trabalho necessário seja realizado; a gestão de riscos identifica e mitiga ameaças; e a gestão de comunicação assegura o alinhamento entre os envolvidos.

2.9.2 Modelo *Ágil*: Flexibilidade com o Guia SBOK™

Em contraposição ao modelo tradicional, as metodologias *ágeis* surgiram como resposta à necessidade de adaptabilidade e entrega contínua de valor. O *Scrum*, conforme definido no Guia SBOK™, [69] – *Scrum Body of Knowledge* – oferece uma estrutura leve e iterativa para desenvolvimento de produtos e gestão de projetos.

Papéis, Eventos e Artefatos do Scrum

O *Scrum* define três papéis principais: o Dono do Produto (responsável pelo valor do produto), o Scrum Master (facilitador do processo) e o Time de Desenvolvimento (que constrói as entregas). Esses papéis interagem em ciclos curtos chamados de sprints, que duram entre uma a quatro semanas.

A cada sprint, ocorre um conjunto de eventos (reuniões) como planejamento, revisão, retrospectiva e reunião diária. Os principais artefatos são o backlog do produto, backlog do sprint e incremento. Esses elementos promovem transparência, inspeção e adaptação [69].

Princípios e Benefícios do Modelo *Scrum*

O modelo *Scrum* apoia-se em princípios como colaboração, auto-organização, entrega orientada a valor, priorização adaptativa e desenvolvimento iterativo. Tais diretrizes são particularmente eficazes em projetos sujeitos a mudanças frequentes ou de requisitos pouco definidos.

A natureza empírica do processo – baseada em transparência, inspeção e adaptação – favorece o aprendizado contínuo e a melhoria progressiva. Isso torna o Scrum adequado para projetos de software, inovação tecnológica e desenvolvimento de soluções complexas.

2.9.3 Integração das Abordagens: Híbrido como Alternativa

Embora tradicionalmente tratadas como opostas, as abordagens do PMBOK® e do Scrum podem ser integradas em um modelo híbrido. Projetos que exigem um planejamento inicial mais robusto, mas que têm fases exploratórias ou iterativas, podem se beneficiar da combinação das abordagens.

Por exemplo, a iniciação e o planejamento do projeto podem seguir os moldes do PMBOK, enquanto a execução e monitoramento se baseiam em ciclos ágeis. Essa complementaridade proporciona estabilidade sem abrir mão da capacidade de adaptação, sendo cada vez mais comum em projetos multidisciplinares.

3 Materiais e Métodos

O *FarmSync* adota uma abordagem integrada que combina as diretrizes do *Project Management Institute (PMI)* para uma gestão de projetos estruturada com a metodologia *Agile* para o desenvolvimento flexível da *API* e da interface *web*. O *PMI* garante processos claros para o gerenciamento de escopo, tempo, custo e riscos, enquanto o *Agile* permite o desenvolvimento incremental, alinhando-se à acumulação gradual de dados dos sensores necessária para o treinamento dos algoritmos preditivos [68, 69].

Os algoritmos preditivos foram desenvolvidos por meio de um processo sistemático: coleta de dados, limpeza, seleção de características (*feature selection*), particionamento do conjunto de dados (*dataset*), treinamento e validação do modelo.

A integração dos princípios de engenharia de software descritos por Ian Sommerville, como entrega incremental (*incremental delivery*) e design evolutivo (*evolutionary design*), assegurou qualidade, adaptabilidade e refinamento iterativo da *API* e da interface, possibilitando a construção de um sistema escalável [70, 71].

3.1 Definição de Requisitos, Riscos e Limitações

Os requisitos do projeto são categorizados em aspectos funcionais, não funcionais e técnicos. Os requisitos funcionais focam no monitoramento das condições ambientais com sensores *IoT*, automação de atuadores (ventiladores e controladores climáticos, por exemplo) e fornecimento de uma interface *web* para visualização dos dados e geração de relatórios. O sistema também deve executar algoritmos preditivos para ajustar as condições com base nos dados coletados.

Os requisitos não funcionais incluem escalabilidade, eficiência energética, alta usabilidade e a integração de tecnologias de baixo custo para garantir viabilidade financeira. Os requisitos técnicos envolvem sensores *IoT*, microcontroladores, *firmware*, protocolos de comunicação, uma *API* baseada em nuvem, um banco de dados para armazenamento dos dados e uma interface *web* para gerenciamento e visualização.

As principais limitações incluem os custos de implementação, restrições energéticas, dependência de conectividade com a internet e as capacidades de processamento dos microcontroladores. A Tabela 3.1 apresenta um resumo dos requisitos do projeto.

Categoria	Detalhes
Requisitos Funcionais	- Monitorar condições ambientais. - Automatizar atuadores. - Interface <i>web</i> para visualização e relatórios. - Executar algoritmos preditivos.
Requisitos Não Funcionais	- Escalável e energeticamente eficiente. - Interface <i>web</i> com boa usabilidade. - Tecnologia de baixo custo.
Requisitos Técnicos	- Sensores <i>IoT</i>, microcontroladores. - <i>API</i> baseada em nuvem e banco de dados. - Interface <i>web</i> e integração com atuadores.

Tabela 3.1: Requisitos do Trabalho de Pesquisa

O sistema está sujeito a uma série de riscos potenciais que podem afetar seu desempenho e confiabilidade, especialmente em implantações no mundo real. Esses riscos incluem problemas de conectividade em ambientes rurais, restrições energéticas em áreas fora da rede elétrica, limitações de poder de processamento e desafios relacionados à escalabilidade e manutenção do sistema. Para enfrentá-los, são identificadas estratégias de tratamento apropriadas — como mitigação, transferência ou aceitação — que estão detalhadas na Tabela 3.2, juntamente com as ações propostas para minimizar seus impactos.

Risco	Estratégia
Falha de conectividade com a internet em áreas rurais	Mitigação: implementar design modular com cache local de dados e avaliar protocolos como LoRaWAN ou redes em malha.
Limitações de energia em instalações remotas	Mitigação: projetar para baixo consumo, usar modos de sono em microcontroladores e otimizar o ciclo de uso dos sensores. Considerar energia solar ou fontes renováveis.
Escalabilidade da infraestrutura em nuvem	Transferência: delegar o gerenciamento da infraestrutura para provedores escaláveis como AWS ou Azure, com funções serverless e bancos gerenciados.
Limitações de processamento dos microcontroladores	Mitigação: transferir cálculos pesados para a <i>API</i> na nuvem e manter o <i>firmware</i> local leve.
Falha de hardware (sensores ou atuadores)	Aceitação: risco inerente a sistemas <i>IoT</i> , minimizado com componentes modulares e fácil substituição.
Inconsistência ou ruído nos dados dos sensores	Mitigação: aplicar filtros e validação na <i>API</i> (médias móveis, detecção de outliers) para limpeza dos dados.
Paradas inesperadas do sistema	Mitigação: utilizar <i>watchdog timers</i> , mensagens de status e logs para detectar falhas e promover recuperação.
Falhas de segurança na comunicação ou no armazenamento em nuvem	Mitigação: usar HTTPS, autenticação por tokens, controle de acesso por funções e gateways seguros de <i>API</i> .

Tabela 3.2: Riscos do projeto e estratégias de tratamento.

Além dos riscos potenciais, a versão atual do sistema apresenta limitações conhecidas que surgiram durante a fase de concepção e projeto do trabalho. Essas limitações refletem o estágio atual de desenvolvimento e indicam áreas para melhorias futuras. A Tabela 3.3 resume essas limitações juntamente com as soluções propostas ou estratégias futuras para tratá-las nas próximas fases do projeto.

Limitação	Solução Proposta / Estratégia Futura
Testes limitados em ambientes reais	Ampliar o escopo de testes para incluir fazendas operacionais. Coletar feedback dos usuários para identificar lacunas de desempenho e desafios não previstos.
Escalabilidade em grandes propriedades	Implementar clusters distribuídos de sensores e avaliar o uso de <i>edge computing</i> para reduzir a carga central. Otimizar os recursos em nuvem para fluxos de dados extensos.
Suporte limitado à expansão automatizada da infraestrutura em nuvem	Embora a infraestrutura atual suporte crescimento modular, ela carece de mecanismos de escalonamento automatizado. Melhorias futuras incluem políticas de <i>auto-scaling</i> , práticas de <i>infrastructure-as-code</i> e <i>deployment pipelines</i> .
Dependência de conexão estável com a internet	Desenvolver recursos de cache de dados e sincronização offline. Avaliar alternativas como LoRaWAN e redes em malha.
Latência nas decisões de controle via nuvem	Adicionar inteligência em nível de borda para decisões sensíveis ao tempo, como o acionamento de atuadores. Utilizar modelos de decisão híbridos (local + nuvem).
Ausência de suporte nativo a <i>apps</i> móveis	Melhorar a responsividade da interface <i>web</i> em dispositivos móveis. Considerar o desenvolvimento futuro de um <i>PWA</i> ou <i>app</i> nativo para maior acessibilidade.
Atualizações de <i>firmware</i> manuais	Desenvolver um mecanismo de atualização remota (<i>OTA</i>) para reduzir a complexidade da manutenção.
Uso de hardware simples com sensores limitados	O uso inicial de componentes básicos como sensores <i>DHT11</i> e microcontroladores de baixo consumo foi intencional, visando validar a viabilidade do sistema em um cenário de baixo custo. Futuras versões usarão sensores mais precisos e especializados (ex.: BME280 e amônia), aproveitando a arquitetura modular do FarmSync.

Tabela 3.3: Limitações do sistema e soluções propostas.

3.2 Requisitos de *Hardware*

Os requisitos de *hardware* para o sistema de monitoramento ambiental foram definidos conforme os objetivos e requisitos do sistema. O *hardware* deve ser capaz de monitorar variáveis microclimáticas como temperatura e umidade.

Além disso, o sistema requer conectividade sem fio para viabilizar a comunicação com

a *cloud*, permitindo a transmissão dos dados coletados para uma *API* responsável pelo processamento e armazenamento dessas informações.

O *hardware* também deve ser compatível com o acionamento de atuadores, a fim de ajustar automaticamente as condições microclimáticas com base nas previsões geradas pelo sistema.

3.3 Seleção dos Componentes de *Hardware*

Buscando atender aos princípios de custo-benefício, os componentes de *hardware* foram selecionados prioritariamente para garantir baixo custo e validar a funcionalidade do ecossistema, com foco na testagem de suas capacidades centrais. O sistema foi projetado para ser flexível e adaptável, permitindo a substituição ou atualização de toda a arquitetura de *hardware* conforme as necessidades específicas do projeto. Essa estrutura possibilita a integração de qualquer *hardware* que suporte comunicação com *APIs* via *HTTPS REST*.

Sensor *DHT22*: selecionado para o monitoramento de temperatura e umidade devido ao seu equilíbrio entre simplicidade, acessibilidade e confiabilidade na detecção ambiental, especialmente em contextos agrícolas [6, 72].

Foi obtida uma cotação junto a dois dos principais distribuidores de componentes eletrônicos — Mouser e Digi-Key — indicando preços unitários para pequenas quantidades variando entre USD 9,59 e USD 9,95, conforme mostrado na Tabela 3.4. O custo-benefício do *DHT22* foi um fator determinante em sua seleção, visto que o sistema foi projetado com foco em acessibilidade e replicabilidade, especialmente para aplicações de pequena escala ou com recursos limitados. Seu preço acessível, aliado à precisão adequada para o monitoramento de temperatura e umidade, tornou o *DHT22* uma escolha apropriada para validação da camada de sensoriamento ambiental do sistema *FarmSync* [73, 74].

Marketplace	Modelo	Quantidade	Preço Unitário (USD)	Referência
<i>Mouser Electronics</i>	<i>DHT22</i>	1	USD 9,59	[75]
<i>Digi-Key Electronics</i>	<i>DHT22</i>	1	USD 9,95	[76]

Tabela 3.4: Cotação do preço unitário do sensor *DHT22* para pequenas quantidades (em 2 de abril de 2025).

O *ESP32* foi selecionado como microcontrolador central devido à sua conectividade *Wi-Fi* integrada, baixo consumo de energia e ampla adoção na comunidade de desenvolvimento *IoT*, o que o torna ideal para a transmissão de dados dos sensores para a nuvem via requisições *HTTP* [6, 7].

A relação custo-benefício em implantações de pequena escala foi um fator decisivo em sua escolha. Foram obtidas cotações junto a dois distribuidores, *Mouser* e *Digi-Key*, com preços unitários variando entre USD 2,50 e USD 4,84 para pequenas quantidades, conforme detalhado na Tabela 3.5. A combinação entre acessibilidade e funcionalidade tornou o *ESP32* uma escolha adequada para validar a viabilidade do ecossistema *FarmSync* [73, 74].

Marketplace	Modelo	Quantidade	Preço Unitário (USD)	Referência
<i>Mouser Electronics</i>	<i>ESP32-WROOM-32E-N4</i>	1	USD 2,50	[77]
<i>Digi-Key Electronics</i>	<i>ESP32-WROOM-32E-N4</i>	1	USD 4,84	[78]

Tabela 3.5: Cotação do preço unitário do ESP32 para pequenas quantidades (em 2 de abril de 2025).

MSP430FR2355: O microcontrolador *MSP430FR2355* foi selecionado para complementar o *ESP32*, atendendo a necessidades específicas de consumo ultrabaixo de energia e monitoramento contínuo eficiente. Enquanto o *ESP32* é responsável pela conectividade e transmissão dos dados, o *MSP430FR2355* atua como controlador principal na aquisição de dados dos sensores, especialmente em cenários que exigem operação de longo prazo com fonte de energia limitada. Seus modos de ultrabaixo consumo e sua arquitetura permitem funcionamento com consumo mínimo, tornando-o ideal para ambientes com baterias ou fora da rede elétrica [79, 80].

Como mostrado na Tabela 3.6, o *MSP430FR2355* apresenta consumo de energia inferior em modos ativo e de espera profunda, quando comparado ao *ESP32*. Esses valores, obtidos a partir das folhas de dados oficiais da *Texas Instruments* [81] e da *Espressif Systems* [82], reforçam a adequação do *MSP430* para o monitoramento contínuo e eficiente de sensores em ambientes com disponibilidade energética limitada.

Modo de operação	MSP430FR2355	ESP32
Ativo	3,40 mA (24 MHz)	20–31 mA (80 MHz)
<i>Deep standby</i>	0,62 μ A	10–150 μ A

Tabela 3.6: Comparação de consumo de energia entre MSP430FR2355 e ESP32 em diferentes modos de operação.

Além disso, o *MSP430FR2355* oferece maior escalabilidade para a integração de múltiplos sensores, proporcionando flexibilidade para expansões futuras do sistema. Embora seu preço unitário (USD 3,96) não seja o mais baixo entre os microcontroladores, suas funcionalidades de economia de energia e capacidade de integração de sensores justificaram sua escolha. Uma cotação obtida em 2 de abril de 2025, junto às distribuidoras *Mouser* e *Digi-Key*, confirmou o mesmo valor entre ambas, conforme mostrado na Tabela 3.7 [73, 74].

Distribuidor	Modelo	Quantidade	Preço Unitário (USD)	Ref.
<i>Mouser Electronics</i>	<i>MSP430FR2355</i>	1	USD 3,96	[83]
<i>Digi-Key Electronics</i>	<i>MSP430FR2355</i>	1	USD 3,96	[84]

Tabela 3.7: Cotação do preço unitário do *MSP430FR2355* para pequenas quantidades (em 2 de abril de 2025).

Atuadores: Para os testes iniciais, ventiladores $80 \times 80 \times 25$ GC 5V 0.15A foram utilizados para controlar o fluxo de ar, validando as previsões do sistema. Para aplicações em ambientes reais, atuadores maiores e mais robustos podem ser integrados para atender demandas mais elevadas.

A Figura 3.1 ilustra os componentes de hardware selecionados e suas interações. O sensor *DHT22* mede temperatura e umidade e envia os dados ao *MSP430FR2355* para processamento. O *MSP430FR2355* comunica-se com o *ESP32*, que transmite os dados para o servidor e repassa comandos de volta ao *MSP430FR2355* para o controle dos atuadores.

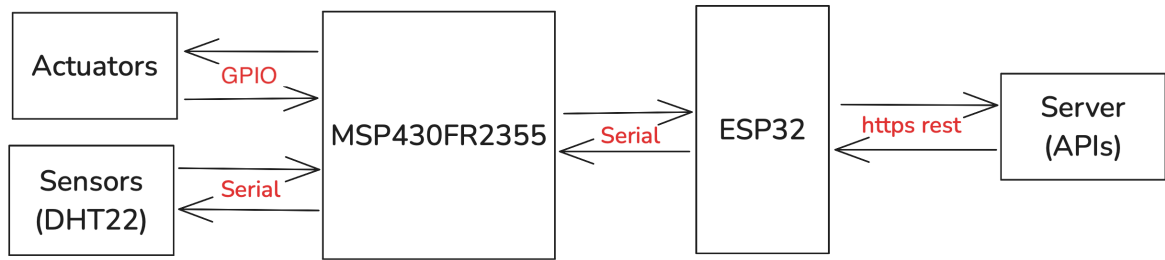


Figura 3.1: Arquitetura do sistema para coleta de dados dos sensores e controle dos atuadores.

Nesta fase inicial de desenvolvimento, apenas sensores de temperatura e umidade foram integrados ao sistema. Essa escolha foi baseada em revisão da literatura sobre aplicações de *IoT*, monitoramento ambiental e *machine learning* para o bem-estar animal e vegetal, em que essas duas variáveis surgem como parâmetros universalmente monitorados [85, 86, 87, 88, 89, 14, 19]. Focar nessas variáveis permitiu validar a funcionalidade central do ecossistema com baixos custos, alinhando-se ao objetivo principal desta pesquisa: validar a funcionalidade e viabilidade de um ecossistema de monitoramento e controle ambiental baseado em *IoT* para galpões agrícolas.

Além disso, embora sensores de gás capazes de detectar compostos como amônia, metano e dióxido de carbono sejam relevantes para aplicações específicas (como em estábulos leiteiros), seu custo ainda é significativamente mais elevado em comparação aos sensores ambientais básicos [27, 8, 15, 14]. Para manter os princípios de viabilidade econômica nesta versão inicial e garantir o cumprimento dos objetivos de validação do sistema, a integração de sensores de gás foi adiada. Uma cotação dos principais sensores de gás identificados em estudos recentes sobre monitoramento ambiental foi realizada nos dois principais distribuidores de componentes eletrônicos, *Digi-Key* e *Mouser* [73, 74, 27, 8, 90, 24, 5].

Os resultados, resumidos na Tabela 3.8, apoiam essa decisão ao evidenciar o custo elevado desses sensores.

Distribuidor	Modelo	Tipo	Gás	Preço (USD)
<i>Mouser Electronics</i>	SRAQ-G001 (MG811)	Dióxido de Carbono (CO ₂)	CO ₂	USD 135,74
<i>Digi-Key Electronics</i>	SRAQ-G001 (MG811)	Dióxido de Carbono (CO ₂)	CO ₂	USD 146,93
<i>Digi-Key Electronics</i>	MIKROE-1630 (MQ-135)	Qualidade do Ar (NH ₃ , CO, CH ₄)	Gases múltiplos	USD 20,00
<i>Mouser Electronics</i>	MIKROE-1630 (MQ-135)	Qualidade do Ar (NH ₃ , CO, CH ₄)	Gases múltiplos	USD 20,00
<i>Mouser Electronics</i>	SEN0159 (MG811)	Dióxido de Carbono (CO ₂)	CO ₂	USD 48,90
<i>Digi-Key Electronics</i>	SEN0159 (MG811)	Dióxido de Carbono (CO ₂)	CO ₂	USD 48,90

Tabela 3.8: Cotação de sensores de gás para aplicações de monitoramento ambiental (em 2 de abril de 2025).

A integração entre esses componentes é responsável por proporcionar um ecossistema flexível e escalável, que contribui para a posterior inclusão de sensores, atuadores e demais elementos de hardware.

3.4 Fluxo de Trabalho do *Hardware*

O fluxo de trabalho do *hardware* é ilustrado na Figura 3.2, que detalha a interação entre a rede de sensores e o servidor para monitorar as condições microclimáticas e controlar os atuadores. O processo se inicia com a *Sensor System Network (SSN)*, que verifica periodicamente se é o momento de medir a temperatura e a umidade. Esse intervalo de tempo é configurável com base nos requisitos do sistema. Caso a condição seja atendida, o sistema coleta os dados ambientais e os envia ao servidor. Ao receber os dados, o servidor processa as informações e envia uma resposta de volta à rede de sensores. Se a resposta for válida, o sistema ajusta o funcionamento dos atuadores e aciona outros *endpoints* para treinar os algoritmos de *machine learning* com base nos comandos do servidor. No caso de uma resposta inválida, um alerta é disparado por meio de um LED.

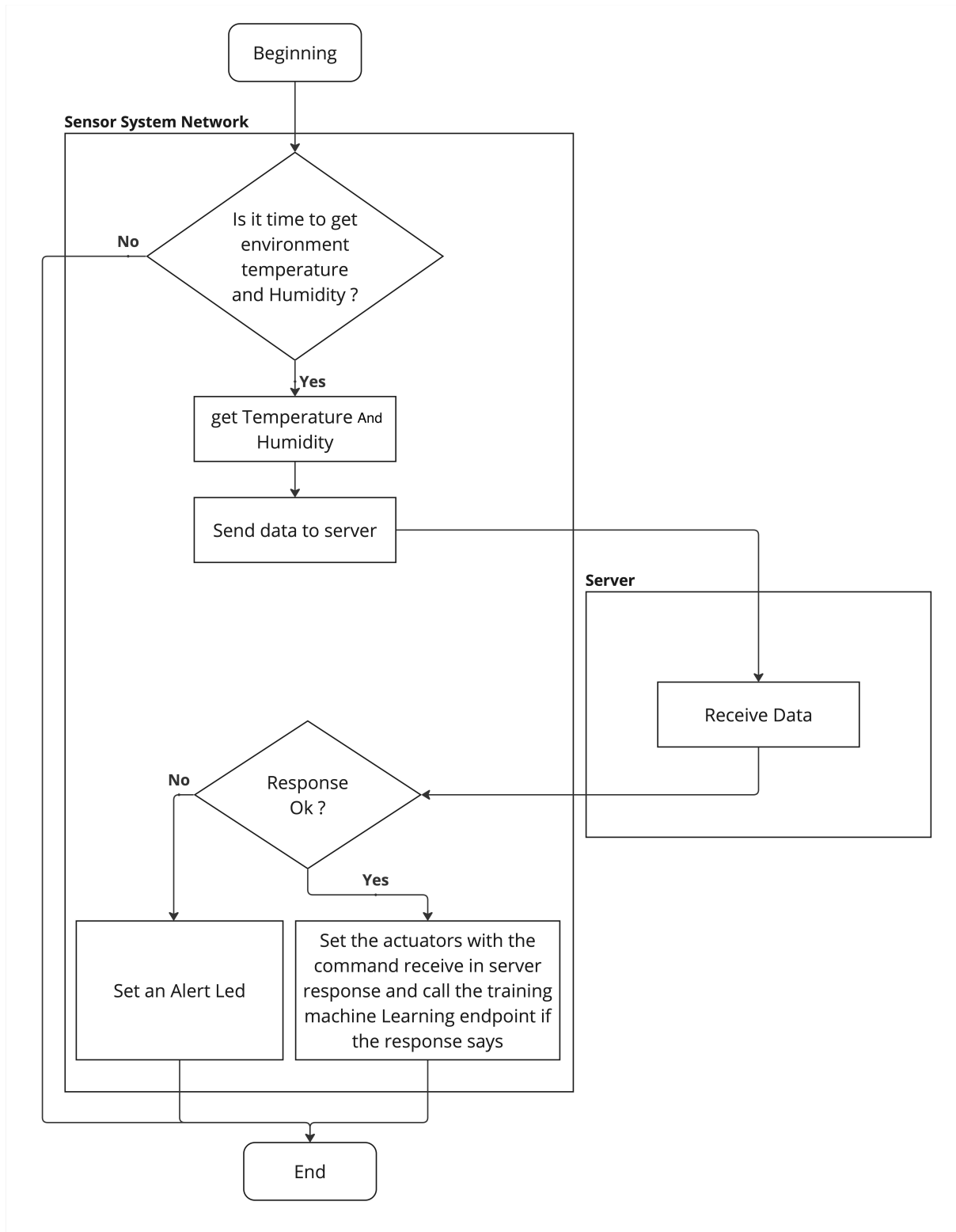


Figura 3.2: Fluxo de trabalho da interação entre o sistema de sensores e o servidor.

3.5 Montagem do Hardware

Os testes iniciais serão realizados utilizando uma *protoboard*, conectando os componentes do sensor e dos microcontroladores. Essa montagem será posicionada em um ambiente

de sala de estar para monitorar a temperatura e a umidade, servindo como uma prova de conceito. Além dos testes com a *proto-board*, também será desenvolvido um circuito impresso (*PCB – Printed Circuit Board*) para integrar os mesmos componentes de forma mais compacta e estável. Os dados coletados durante essa fase serão utilizados para validar a funcionalidade e a confiabilidade do sistema, assegurando que ele atenda aos requisitos e objetivos do trabalho de pesquisa.

3.6 Desenvolvimento do Firmware

O firmware do microcontrolador MSP430 foi programado para realizar a leitura periódica dos sensores conectados ao sistema, incluindo sensores de temperatura e umidade. Após a aquisição, os dados são formatados e enviados ao microcontrolador ESP32 por meio de comunicação serial (UART). O firmware também gerencia os sinais de entrada e saída relacionados aos atuadores da rede.

O firmware do ESP32 foi estruturado para interpretar os dados recebidos via UART, identificar o módulo sensor responsável pela transmissão, o tipo de sensor utilizado e o valor medido. Cada conjunto de dados é associado ao identificador do galpão em que a rede de sensores está instalada, ao identificador do módulo sensor e ao identificador individual de cada sensor.

Em seguida, o ESP32 realiza a comunicação com as APIs do sistema por meio de requisições HTTP, utilizando o protocolo HTTPS. As informações são transmitidas ao sistema central para armazenamento e processamento.

A frequência de envio das informações é configurada diretamente no firmware do ESP32. Esse parâmetro pode ser ajustado conforme o cenário experimental ou operacional. Essa configuração permite o controle da taxa de atualização dos dados enviados à API e está diretamente relacionada à eficiência da operação do sistema.

Além disso, o firmware do ESP32 deve estar preparado para interpretar as respostas recebidas da API de recebimento e processamento de dados. Caso a resposta contenha a sinalização para iniciar o treinamento dos algoritmos preditivos, o firmware aciona a API específica de treinamento, enviando os parâmetros necessários para a execução do processo. Essa funcionalidade permite a reconfiguração dinâmica do modelo preditivo conforme novos dados são coletados e processados.

3.7 Seleção de Algoritmos Preditivos

A seleção dos algoritmos preditivos para o projeto foi guiada pelas características dos dados microclimáticos e pela necessidade de prever condições, especialmente a temperatura. Os dados do sistema consistem principalmente em medições de temperatura e umidade coletadas por sensores em uma rede de microcontroladores. Esses dados apresentam um forte componente temporal com variações contínuas ao longo do tempo, exigindo algoritmos capazes de capturar essas dependências. Além disso, parâmetros externos como previsões meteorológicas e condições atmosféricas foram integrados para aprimorar a capacidade preditiva. Com base nesses fatores, os algoritmos selecionados precisaram atender aos seguintes critérios:

- Capturar padrões temporais nos dados.
- Processar múltiplas variáveis microclimáticas.

- Fornecer previsões para tomada de decisão.
- Escalar com o aumento no volume de dados.

Para atender a esses requisitos, foram selecionados os seguintes algoritmos:

- *Recurrent Neural Networks (RNNs)*: as *RNNs* foram escolhidas por sua capacidade de lidar com dados de séries temporais e capturar dependências de longo prazo [91].
- *Random Forest (RF)*: o *Random Forest* foi selecionado por sua versatilidade no tratamento de dados complexos e não lineares, característica comum em sistemas de monitoramento ambiental. Além disso, ele reduz o impacto de *outliers*, melhora a acurácia ao agregar previsões de múltiplas árvores de decisão e identifica as variáveis mais importantes [92].

3.8 Seleção do Provedor de Serviços em Nuvem

A seleção do provedor de serviços em nuvem para este trabalho de pesquisa baseou-se em uma análise dos requisitos técnicos, funcionais e econômicos. Dada a necessidade de uma infraestrutura confiável e escalável para suportar a coleta, o processamento e a análise preditiva de dados, foram estabelecidos critérios específicos para guiar a avaliação. Esses critérios garantiram que o provedor escolhido pudesse atender às demandas de operações intensivas em dados, mantendo ao mesmo tempo a viabilidade econômica e a segurança. Os seguintes aspectos foram considerados:

- Escalabilidade: capacidade de expandir recursos de armazenamento e processamento à medida que o volume de dados aumenta e novas funcionalidades são adicionadas.
- Efetividade de custo: modelo de precificação que equilibra os custos de armazenamento, processamento e transferência de dados, garantindo a viabilidade econômica do sistema.
- Suporte a *Machine Learning*: ferramentas para desenvolver, treinar e implementar modelos de *machine learning* para análise preditiva e ajustes automatizados.
- Segurança e conformidade: medidas de segurança, como criptografia e gerenciamento de chaves, além de conformidade com regulamentações internacionais de proteção de dados.
- Alta disponibilidade e redundância: garantias de disponibilidade dos serviços por meio de infraestrutura redundante para assegurar a confiabilidade.

Para determinar o provedor de nuvem mais adequado, foi realizada uma avaliação detalhada com base nos critérios definidos. A análise concentrou-se em escalabilidade, custo-benefício, suporte a *machine learning*, segurança e alta disponibilidade. Três grandes provedores de nuvem foram analisados, cada um oferecendo vantagens distintas e ferramentas especializadas para lidar com o processamento de dados orientados por *IoT* e análises preditivas.

3.8.1 *Amazon Web Services (AWS)*

A *AWS* oferece uma infraestrutura escalável e suporte a *IoT* por meio do *AWS IoT Core*. Inclui serviços como o *AWS SageMaker* para *machine learning* e o *DynamoDB* para gerenciamento de grandes volumes de dados. A *AWS* é reconhecida por sua presença global, alta segurança e disponibilidade de 99,99%, suportada por 108 zonas de disponibilidade em 34 regiões [93, 94].

3.8.2 *Microsoft Azure*

O *Azure* oferece o *Azure IoT Hub* para gerenciamento de dispositivos *IoT* e integra-se ao *Azure Machine Learning*. Também suporta bancos de dados *SQL* e *NoSQL*, oferecendo recursos de segurança e conformidade com regulamentações de proteção de dados [95].

3.8.3 *Google Cloud Platform (GCP)*

A *GCP* oferece o *Google Cloud IoT Core* e integra-se ao *TensorFlow*, um framework de *machine learning*. Possui ferramentas de análise de dados como o *BigQuery*, além de oferecer escalabilidade [96].

3.9 Definição da Arquitetura de Banco de Dados

A arquitetura de banco de dados do sistema foi projetada para suportar o armazenamento, recuperação e análise dos dados do ecossistema de forma escalável. O modelo proposto considera os requisitos de desempenho, flexibilidade de estruturação dos dados, capacidade de escalonamento horizontal e facilidade de integração com serviços externos.

Com base nesses requisitos, foram definidos os padrões e atributos essenciais para a escolha da tecnologia de banco de dados:

- Escalabilidade horizontal: capacidade de manter a performance mesmo com o aumento do volume de dados e do número de ambientes monitorados.
- Baixa latência de leitura e escrita: essencial para suportar o processamento em tempo quase real dos dados captados e das decisões automatizadas.
- Modelo orientado a documentos ou chave-valor: maior flexibilidade na estrutura dos dados, adequada a registros heterogêneos, como sensores, previsões e logs de controle.
- Suporte a múltiplos ambientes simultâneos: a arquitetura deve permitir o monitoramento de diferentes instalações, com isolamento lógico e configuração independente.
- Custo proporcional ao uso: o modelo de precificação do banco de dados deve ser compatível com orçamentos de pesquisa e permitir adaptação ao volume variável de dados.

A Figura 3.3 apresenta a arquitetura conceitual do banco de dados, detalhando as entidades e seus relacionamentos para suporte ao monitoramento ambiental:

- *HangarData* (1-1): Armazena informações sobre os ambientes controlados (por exemplo, galpões ou instalações específicas). Cada ambiente possui uma configuração única registrada na tabela *APIConfiguration*.
- *ModuleSensorData* (1-n): Representa os módulos de sensores que monitoram cada ambiente. Um único ambiente pode conter vários módulos, cada um capaz de coletar dados de múltiplos sensores, vinculados à tabela *SensorData*.
- *SensorType* (1-n): Define os tipos de sensores (por exemplo, temperatura e umidade) utilizados no sistema. Os módulos podem incluir diversos tipos de sensores, e um mesmo tipo pode ser compartilhado entre diferentes módulos.
- *SensorData* (1-n, 1-2): Armazena os dados brutos coletados pelos sensores, com os seguintes vínculos:
 - *ModuleSensorData* (1-n): Múltiplos dados coletados por módulo.
 - *WeatherData* (1-2): Dados meteorológicos atuais e previsão de 24 horas.
 - *PredictData* (1-1): Previsões para a próxima medição no ambiente.
- *WeatherData* (1-2): Armazena dados meteorológicos externos, vinculados à *SensorData* para condições atuais e previstas.
- *PredictData* (1-1): Armazena as previsões geradas pelos algoritmos de aprendizado de máquina com base nos dados dos sensores, vinculadas diretamente aos dados correspondentes.
- *ActuatorData* (1-n): Mantém informações sobre o controle de atuadores (por exemplo, ventiladores). Cada módulo pode gerenciar vários atuadores com base nos dados dos sensores.
- *ControlLogs* (1-n): Registra falhas operacionais no ecossistema. Cada módulo pode gerar múltiplos registros de falha para acompanhamento.
- *APIConfiguration* (1-1): Armazena as configurações específicas da *API* para cada ambiente controlado, vinculadas à tabela *HangarData*.

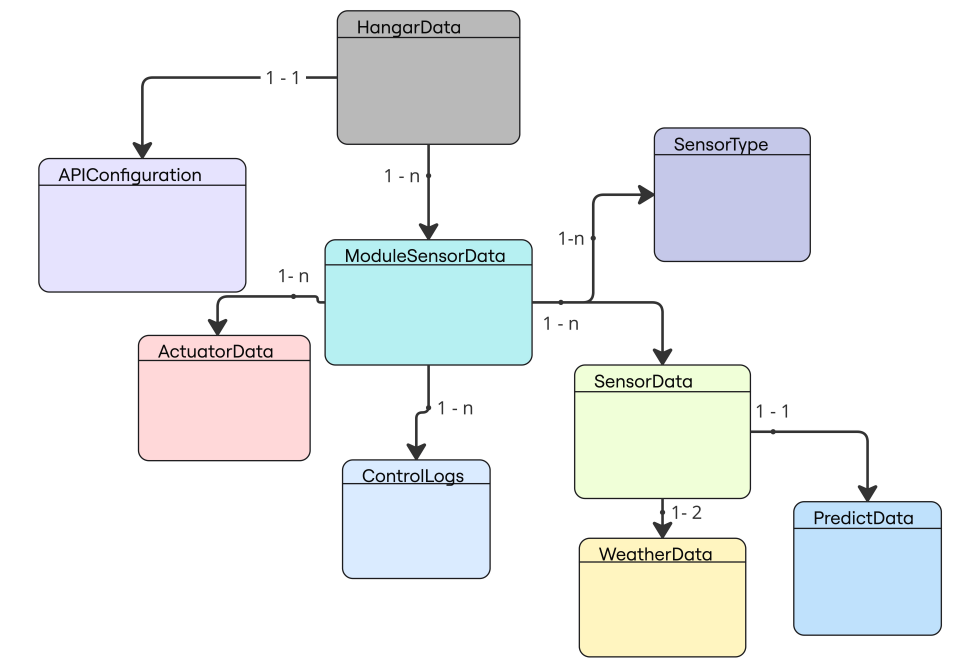


Figura 3.3: Arquitetura de banco de dados para gerenciamento de dados do ecossistema.

3.10 Arquitetura e Requisitos da API

O sistema proposto de monitoramento e controle ambiental integra múltiplos componentes para coletar, processar e analisar dados microclimáticos. A rede de sensores constitui o núcleo desta arquitetura. A Figura 3.4 ilustra a interação entre esses componentes, destacando o fluxo de dados entre a rede de sensores, as APIs e os serviços de armazenamento e predição.

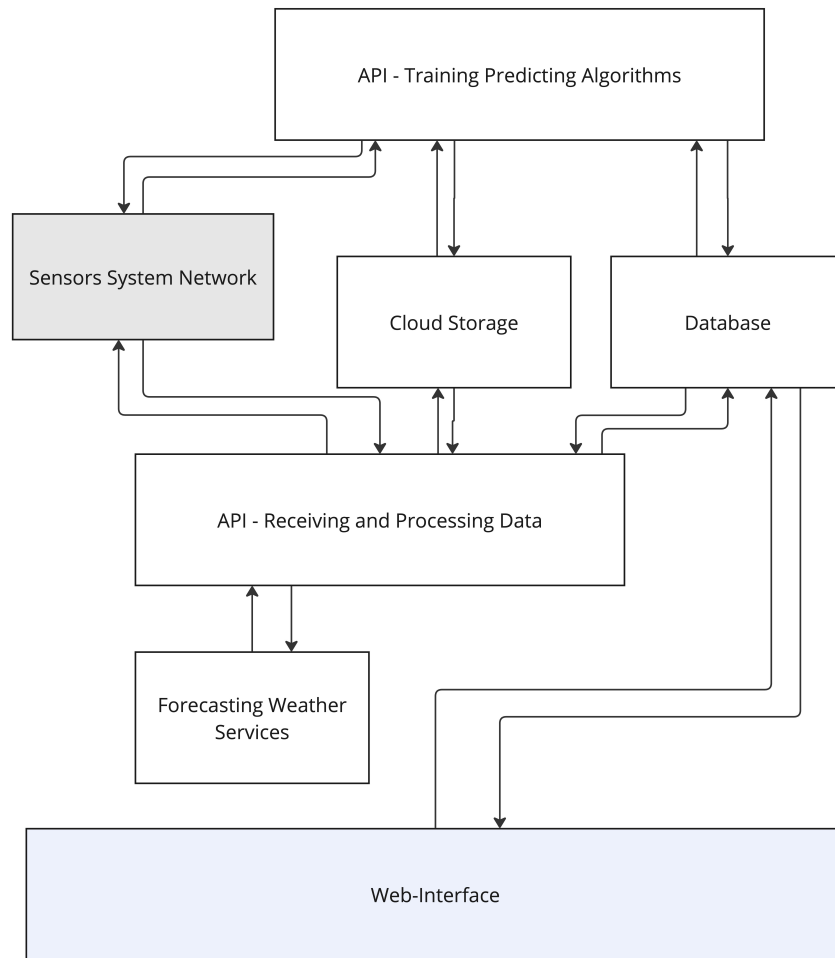


Figura 3.4: Arquitetura detalhada do ecossistema e fluxo de dados.

A rede de sensores atua como a espinha dorsal do sistema, sendo responsável pela coleta de dados das variáveis microclimáticas, como temperatura e umidade. Esses dados são enviados para a *API* para posterior processamento e integração com fontes externas.

3.10.1 *API* de Recebimento e Processamento de Dados

A *API* recebe dados da rede de sensores e os integra com informações externas, como previsões do tempo, incluindo umidade, velocidade do vento e pressão atmosférica, obtidas de um serviço meteorológico como a *WeatherAPI* [97]. Esse conjunto de dados é armazenado no banco de dados e utilizado para o treinamento de algoritmos preditivos. A *API* também executa as seguintes tarefas principais:

- **Enriquecimento de Dados:** combina os dados dos sensores com informações meteorológicas externas para fornecer um conjunto de dados para análise preditiva.
- **Tomada de Decisão:** com base nas previsões, a *API* determina se os atuadores (por exemplo, ventiladores) devem ser ativados ou desativados para manter as condições ideais.
- **Treinamento de Algoritmos:** caso necessário, a *API* configura o sistema para realizar o treinamento dos algoritmos, e outra *API* é chamada pela *SSN* para executar o treinamento.

- **Armazenamento de Dados:** organiza e armazena todos os dados processados em um banco de dados para uso futuro em previsões ou visualização na interface *web*.
- **Registro e Classificação de Erros:** quaisquer erros encontrados durante o processo são registrados, categorizados e salvos para análise futura.

Em resumo, a *API* de Recebimento e Processamento de Dados atua como o cérebro do processo de tomada de decisão do sistema. Ela coleta dados dos sensores, os enriquece com previsões meteorológicas externas e armazena tudo na nuvem. Ao analisar esses dados, a *API* ajuda a determinar se dispositivos como ventiladores devem ser ativados para manter um ambiente estável. Também registra todos os dados coletados e o comportamento operacional do sistema, tornando essas informações disponíveis para análises posteriores, treinamento de modelos e diagnósticos.

A comunicação com a *API* de recebimento e processamento de dados é realizada por meio de requisições HTTP do tipo *POST*. Para autenticação, o cabeçalho da requisição deve conter um token de acesso válido, fornecido previamente, no campo **AppToken**. Esse token identifica a rede de sensores autorizada a realizar a transmissão.

O corpo da requisição (*body*) deve ser enviado no formato **JSON**, contendo os seguintes campos obrigatórios:

- **HangarId:** identificador numérico do galpão onde os dados foram capturados.
- **Module:** identificador do módulo de sensores responsável pela coleta.
- **Sensor:** identificador individual do sensor que gerou os dados.
- **Temperature:** valor de temperatura capturado, em graus Celsius.
- **Humidity:** valor de umidade relativa do ar, em porcentagem.
- **ActuatorData:** valor associado ao estado do atuador, quando aplicável.

Exemplo de requisição:

```
POST /sensor-data HTTP/1.1
Host: api.exemplo.com
Content-Type: application/json
AppToken: abc123xyz456
```

```
{
  "HangarId": 2,
  "Module": 5,
  "Sensor": 3,
  "Temperature": 24.78,
  "Humidity": 48.95,
  "ActuatorData": 7.12
}
```

Como resposta, a *API* retorna um objeto **JSON** contendo a confirmação de recebimento e instruções operacionais adicionais, como controle de atuadores ou sinalização de treinamento de algoritmos:

```
{
  "FanControl": true,
  "SetToTrain": true,
  "date": "2024-06-01 14:22:43.528194",
  "id": "a3d91b8e-0f23-4c8f-b25f-672fa9d2c674",
  "message": "Sensor data received and processed successfully."
}
```

O campo `FanControl` indica a necessidade de ativação ou desativação de dispositivos de controle ambiental. O campo `SetToTrain` informa se os dados enviados deverão ser utilizados para o treinamento de modelos preditivos. A resposta inclui ainda a identificação única da inserção no banco de dados e a data de processamento.

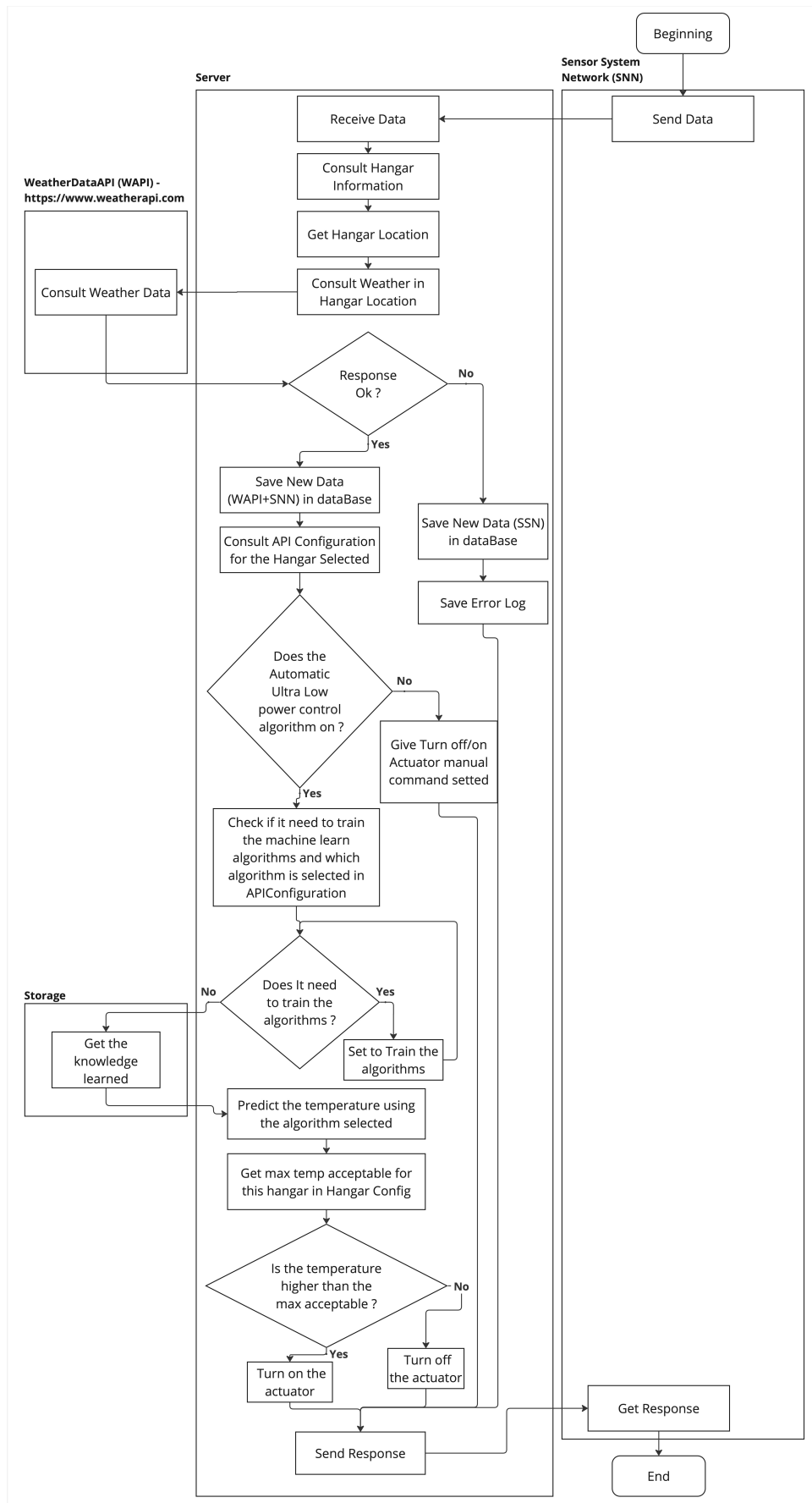


Figura 3.5: Fluxo de funcionamento da API de recebimento e processamento de dados.

3.10.2 *Database* e Armazenamento em Nuvem

O sistema depende tanto de um *database* quanto de um armazenamento em nuvem para gerenciar e processar os dados. O *database* lida com dados estruturados, armazenando leituras dos sensores e informações meteorológicas externas. Isso garante que os dados estejam disponíveis para os algoritmos preditivos e possam ser acessados por meio da interface *web* para fins de monitoramento e análise histórica.

Enquanto isso, o armazenamento em nuvem fornece um espaço escalável para armazenar modelos de aprendizado de máquina e dados relacionados. Isso permite que o sistema retenha o conhecimento preditivo, possibilitando a atualização e o re-treinamento dos modelos conforme novos dados se tornam disponíveis, sem sobrecarregar o *database*. Juntos, esses componentes asseguram a gestão eficiente dos dados, apoiando tanto as operações quanto a evolução de longo prazo do sistema.

3.10.3 *API* de Treinamento de Algoritmos Preditivos

A *API* de Treinamento utiliza dados da rede de sensores e do serviço de previsão do tempo para treinar algoritmos de aprendizado de máquina. Esses algoritmos predizem as condições ambientais avaliando dados das variáveis microclimáticas, como temperatura e umidade, para a próxima medição da rede de sensores. A *API* automatiza todo o processo de treinamento, incluindo preparação dos dados, seleção dos algoritmos e armazenamento do conhecimento aprendido, conforme ilustrado na Figura 3.6.

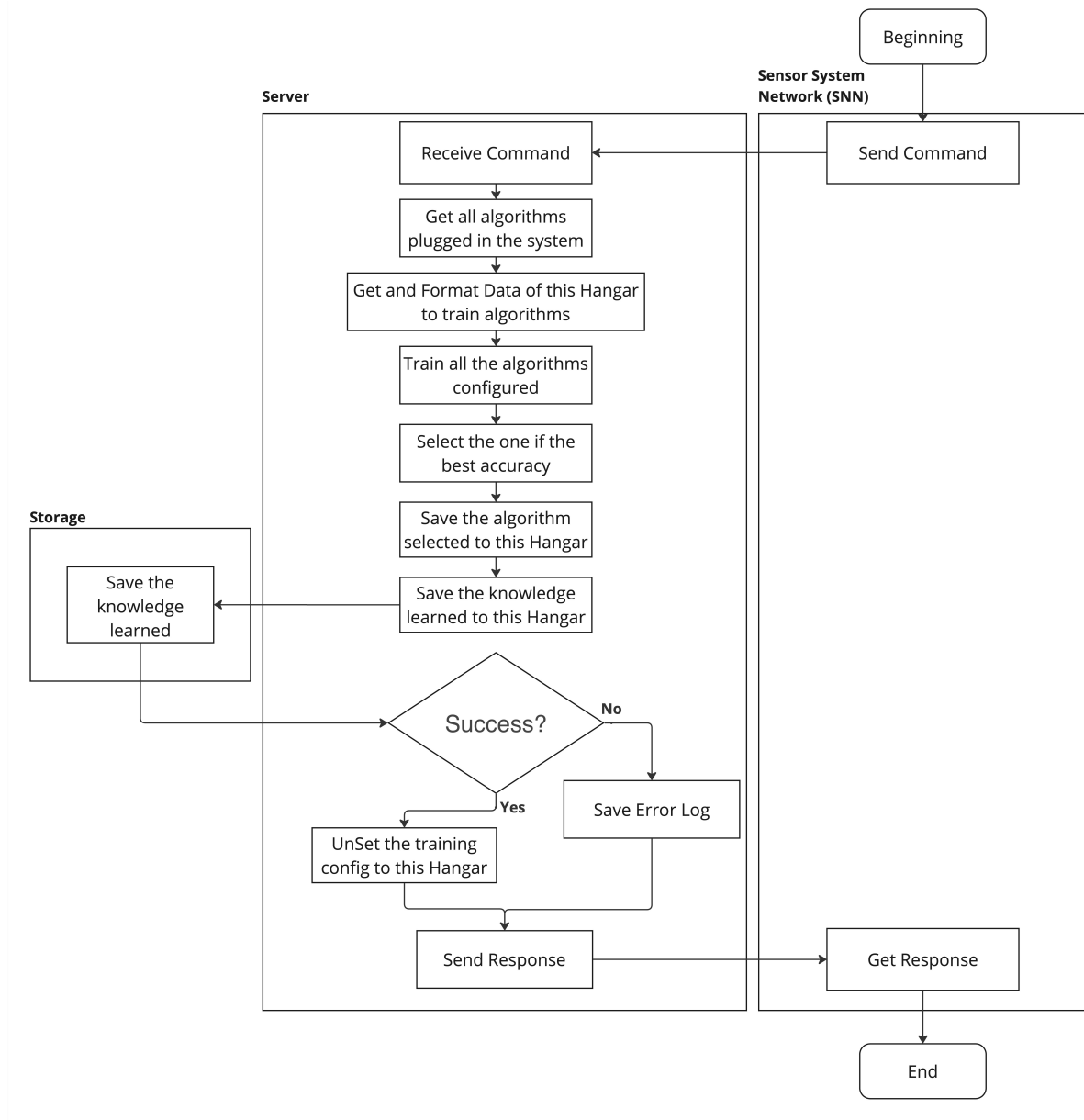


Figura 3.6: Fluxo da API de Treinamento para algoritmos preditivos.

O processo de formatação e pré-processamento dos dados garante que o conjunto esteja estruturado e otimizado para o treinamento, melhorando a acurácia e a confiabilidade dos modelos. Esta etapa inclui a organização dos dados brutos dos sensores, a integração com fontes externas e a preparação dos dados para os algoritmos de *machine learning*. As principais tarefas envolvidas são:

- **Combinação de fontes de dados:** Os dados dos sensores são enriquecidos com dados meteorológicos (por exemplo, umidade, velocidade do vento e pressão atmosférica) obtidos de serviços externos, como a *WeatherAPI* [97].
- **Limpeza e validação:** A API verifica valores ausentes ou inconsistentes e aplica técnicas de imputação para garantir a integridade dos dados.
- **Divisão dos dados:** O conjunto de dados é dividido em subconjuntos de treinamento, validação e teste para avaliação do modelo.

Com os dados formatados, o fluxo de treinamento na *API* começa com a normalização e a seleção de características. O conjunto é então dividido em subconjuntos para treinamento e validação, permitindo que o modelo aprenda padrões e seja avaliado quanto à sua acurácia. Por fim, o modelo treinado é testado com dados inéditos antes da implantação. A *API* segue estas etapas principais:

1. Recuperar Algoritmos: A *API* recupera todos os algoritmos configurados, incluindo modelos de regressão e redes neurais.
2. Preparar Dados para Treinamento: A *API* formata os dados de acordo com os requisitos de cada algoritmo.
3. Treinar Algoritmos: A *API* treina todos os algoritmos com os dados preparados e os avalia utilizando métricas como acurácia, R-quadrado e Erro Absoluto Médio (*MAE*).
4. Selecionar o Melhor Algoritmo: A *API* compara os algoritmos treinados e seleciona aquele com maior acurácia para o ambiente monitorado.
5. Salvar Modelo Treinado: O algoritmo selecionado e seu modelo treinado são armazenados tanto na nuvem quanto no *database*.

A *API* foi projetada para lidar com processos de treinamento bem-sucedidos e com falhas, garantindo confiabilidade e rastreabilidade no desenvolvimento dos modelos. Ela adota uma abordagem estruturada para gerenciar os resultados da seguinte forma:

- Treinamento Bem-Sucedido: O modelo treinado é vinculado ao ambiente específico e seu conhecimento é salvo para uso em previsões futuras. A configuração de treinamento é então desativada para o ambiente.
- Treinamento com Falha: Caso o processo falhe, a *API* cria um registro de erro detalhando os problemas encontrados.

A comunicação com a *API* de treinamento é realizada por meio de requisições HTTP do tipo *POST*. A autenticação é feita por meio de um token previamente autorizado, transmitido no cabeçalho da requisição por meio do campo **AppToken**.

O corpo da requisição (*body*) deve ser enviado no formato JSON, contendo os seguintes campos:

- **HangarId**: identificador do galpão que será considerado no processo de treinamento.
- **Module**: identificador do módulo de sensores cujos dados servirão como base para o treinamento.
- **idSendData**: identificador único do conjunto de dados que será utilizado para o treinamento dos modelos.

Exemplo de requisição:

```
POST /train-model HTTP/1.1
Host: api.exemplo.com
Content-Type: application/json
AppToken: abc123xyz456

{
  "HangarId": 1,
  "Module": 1,
  "idSendData": "62a5d825-c92d-468e-af76-7b71598ac096"
}
```

A resposta da API informa se o treinamento foi concluído com sucesso. Também retorna o identificador do processo e uma mensagem de confirmação:

```
{
  "SetToTrain": false,
  "id": "62a5d825-c92d-468e-af76-7b71598ac096",
  "message": "Success Training the Algorithms:"
}
```

O campo `SetToTrain` indica se o processo de treinamento deve ser acionado novamente ou não. O campo `id` corresponde ao identificador do conjunto de dados utilizado, e o campo `message` fornece uma descrição textual do resultado da operação.

3.10.4 Desenvolvimento da API

As APIs foram desenvolvidas em *Python*, escolhido por sua flexibilidade e compatibilidade com uma ampla gama de bibliotecas de ciência de dados e *machine learning*. O *Python* simplifica a integração com serviços em nuvem como a *AWS*, utilizando bibliotecas como a *boto3* para o gerenciamento de recursos. A implementação se baseou em diversas bibliotecas e ferramentas fundamentais:

- *scikit-learn* e *TensorFlow/Keras*: utilizadas para fornecer a infraestrutura necessária para construção, treinamento e testes dos modelos preditivos.
- *decouple*: utilizada para facilitar o gerenciamento de variáveis de ambiente, garantindo o tratamento seguro de configurações sensíveis.
- *requests*: responsável por realizar requisições *HTTP*, permitindo a comunicação com serviços externos como a *WeatherAPI* [97].
- *pandas* e *numpy*: utilizadas para manipulação, limpeza e análise de dados, permitindo o processamento de grandes volumes de informação.
- *pickle*: utilizada para serialização e desserialização de objetos, como o salvamento de modelos treinados para reutilização futura.

3.11 Arquitetura e Requisitos da Interface Web

A interface *web* foi projetada para permitir o monitoramento, visualização e gerenciamento dos dados microclimáticos capturados pelos sensores. A arquitetura segue o modelo *cliente-servidor*, com um *front-end* para interação do usuário e um *back-end* para processamento dos dados e comunicação com o banco de dados.

3.11.1 *Front-End* (Interface do Usuário)

O *front-end* foi construído utilizando *HTML*, *CSS* e *JavaScript*, com o uso do framework *Bootstrap* para criar layouts responsivos. Isso garante acessibilidade e usabilidade em diferentes dispositivos, incluindo computadores, tablets e smartphones. Para visualização de dados, a biblioteca *Chart.js* é utilizada para gerar gráficos dinâmicos e interativos, como gráficos de linha e barra, permitindo aos usuários monitorar variáveis como temperatura e umidade ao longo do tempo. As principais funcionalidades do *front-end* incluem:

- *Filtragem de dados*: o sistema permite que o usuário filtre os dados com base em parâmetros específicos e intervalos de datas [98].
- *Exportação de dados*: os dados visualizados podem ser exportados em formatos como *Excel* ou *CSV* para análise offline [99].

3.11.2 *Back-End* (Interação com o Banco de Dados)

O *back-end* é responsável por gerenciar o fluxo de dados e as operações do sistema por meio da interação direta com o banco de dados *DynamoDB*. Essa interação garante o gerenciamento das configurações e o controle dos atuadores, permitindo o funcionamento do ecossistema *IoT*. O *back-end* executa as seguintes funções principais no banco de dados:

- Recuperação dos dados capturados pelos sensores.
- Gerenciamento das configurações do sistema.
- Controle dos atuadores na rede.

Para aumentar a eficiência, o *back-end* utiliza uma estratégia de *caching*, armazenando temporariamente em memória os dados acessados com frequência. Essa abordagem reduz a frequência de consultas ao banco de dados, diminui os custos de utilização dos serviços da *AWS* e melhora o tempo de resposta em consultas repetidas.

3.11.3 Requisitos Funcionais

Os requisitos funcionais definem as capacidades essenciais que o sistema deve oferecer para garantir o monitoramento e o controle das condições ambientais. Esses requisitos abrangem visualização de dados, gerenciamento de dados históricos, monitoramento de falhas, configuração do sistema e controle de atuadores. Ao implementar essas funcionalidades, o sistema permite que os usuários acessem e interajam com os dados atuais e históricos. Os principais requisitos funcionais são os seguintes:

- **Visualização de Dados:** O sistema deve permitir que os usuários acessem e visualizem dados capturados pelos sensores, como temperatura e umidade.
- **Visualização de Dados Históricos:** Os usuários devem poder visualizar dados históricos por meio de gráficos interativos e dinâmicos. Os dados devem ser filtráveis por parâmetros (por exemplo, temperatura e umidade) e intervalos de datas.
- **Exportação de Dados:** A interface deve permitir a exportação dos dados atuais e históricos em formatos como *Excel* e *CSV*.
- **Monitoramento de Falhas:** O sistema deve detectar e exibir falhas no ecossistema em formato de relatório para análise do usuário.
- **Configuração do Galpão:** Os usuários devem poder configurar parâmetros do galpão, como limites aceitáveis de temperatura e umidade.
- **Controle de Atuadores:** A interface deve permitir que os usuários controlem os atuadores (por exemplo, *ventiladores* e *aquecedores*) dentro da rede de sensores.

3.11.4 Requisitos Não Funcionais

Os requisitos não funcionais definem aspectos de desempenho e usabilidade que garantem que o sistema opere corretamente e permaneça acessível em diferentes dispositivos. Esses requisitos focam em fatores como responsividade, escalabilidade e experiência do usuário. O principal requisito não funcional é o seguinte:

- *Responsiveness:* A interface *web* deve se adaptar a diferentes tamanhos de tela e dispositivos (por exemplo, computadores, tablets e smartphones) sem comprometer a usabilidade ou a experiência do usuário.

Para atender a esses requisitos funcionais e não funcionais, a interface *web* foi desenvolvida utilizando *Python* e o framework *Django*, escolhidos por sua escalabilidade, modularidade e capacidade de integração [100]. Os recursos de segurança do *Django*, suas ferramentas nativas e o suporte à integração com bibliotecas de terceiros aumentam a confiabilidade do sistema e facilitam a comunicação com serviços em nuvem.

4 Resultados e Discussão

4.1 Resultados da Validação do *Hardware*

O sistema de monitoramento ambiental baseado em *IoT* foi testado em um ambiente de confinamento controlado para validar sua funcionalidade. A primeira versão do teste de conceito, montada em uma *proto board*, incluiu o microcontrolador *MSP430FR2355*, o microcontrolador *ESP32*, o sensor *DHT22* e os atuadores que, neste caso, eram ventiladores. Durante os testes, o sistema coletou com sucesso dados de temperatura e umidade do sensor e os transmitiu para a *cloud* por meio do microcontrolador *ESP32*. O conjunto de dados foi enviado para a *API*, responsável por processá-los e armazená-los na *cloud*. Também testamos os atuadores, como os ventiladores. Nesse processo, o sistema ativou e desativou os ventiladores com base nos comandos emitidos pelo sistema. Vale ressaltar que a transição da *proto board* para a *PCB* (Figura 4.1) não introduziu nenhum problema funcional.

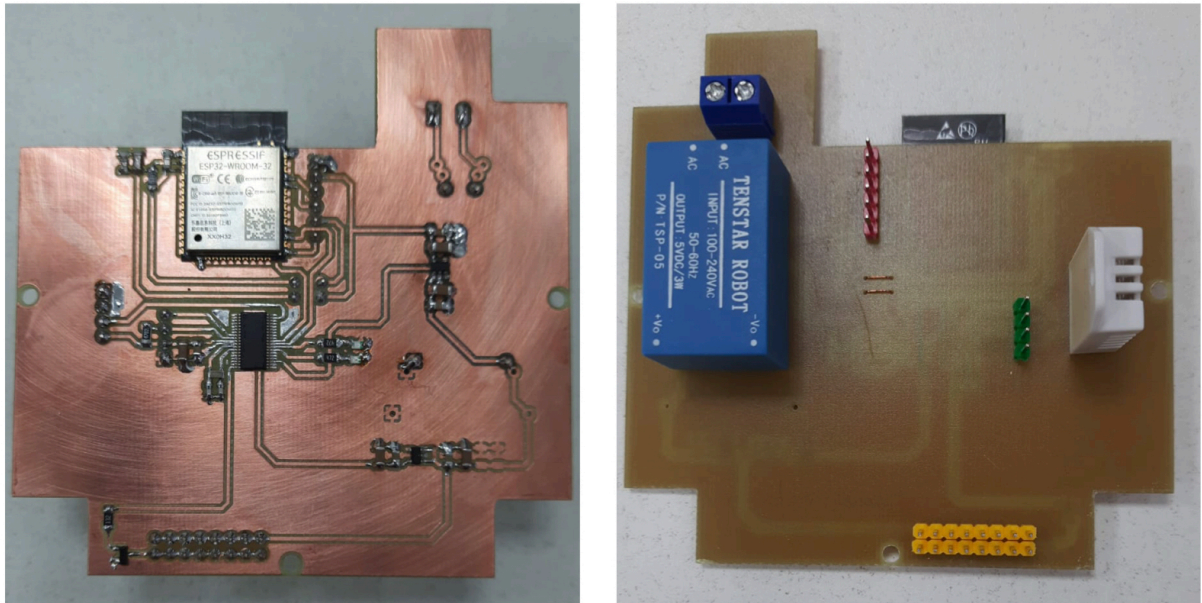


Figura 4.1: *Printed Circuit Board (PCB)* — vistas superior e inferior.

Após a montagem da *PCB*, o hardware foi implantado *in situ* dentro do ambiente de testes controlado para validar seu desempenho em condições reais. Durante a implantação, o sistema monitorou continuamente a temperatura e a umidade ambiente utilizando o sensor *DHT22* e ativou com sucesso os atuadores. A Figura 4.2a,b ilustra o hardware instalado e operando dentro da sala.

O hardware é alimentado por uma fonte de 5 V *DC*, obtida a partir de um módulo conversor *AC-DC HLK-PM01* conectado à rede elétrica principal (110/220 V *AC*). A fonte de 5 V alimenta um regulador linear (*TPS7A4533*), que fornece uma tensão estável de 3,3 V para o módulo *ESP32*, o microcontrolador *MSP430FR2355* e outros componentes de 3,3 V. Para conectividade em rede, o microcontrolador *ESP32* estabelece comunicação *Wi-Fi* com a rede sem fio local.

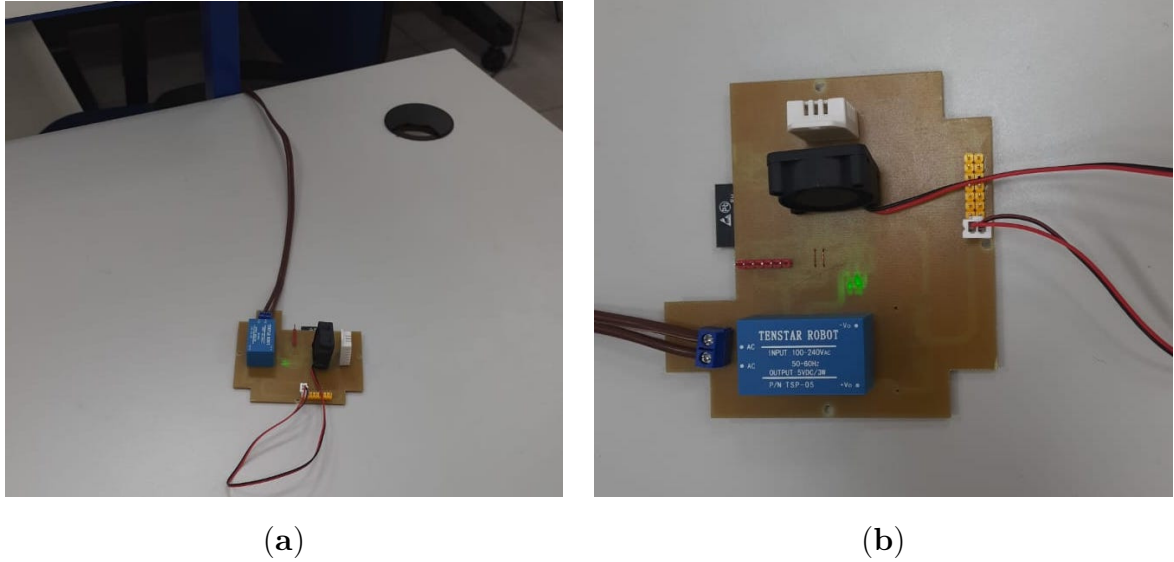


Figura 4.2: Implantação do sistema: (a, b) mostram diferentes visões do hardware instalado e funcionando dentro do ambiente controlado, (a) Hardware instalado e operando no ambiente controlado (visão 1), (b) Hardware instalado e operando no ambiente controlado (visão 2).

Para otimizar o consumo de energia, o sistema foi projetado de forma que o *ESP32* permaneça em modo de *deep sleep* durante a maior parte do ciclo de operação. O microcontrolador *MSP430FR2355*, operando continuamente em um modo de consumo ultra-baixo, é responsável por gerenciar a aquisição dos dados dos sensores. Quando um evento de transmissão de dados é agendado, o *MSP430* aciona o *ESP32* para que este acorde, estabeleça conexão com a *Wi-Fi* e transmita os dados coletados para o servidor em nuvem por meio de uma requisição *HTTPS REST API*. Imediatamente após concluir a comunicação, o *ESP32* retorna ao modo de *deep sleep*.

A Figura 4.3 ilustra esse comportamento, apresentando o perfil de consumo de corrente durante um ciclo típico de operação. O gráfico destaca a corrente mínima durante as fases de inatividade e o pico temporário durante a comunicação ativa. Esse projeto reduz o consumo médio de energia do sistema, tornando-o adequado para implantações com restrição energética, ao mesmo tempo que assegura uma transmissão de dados periódica e confiável para a nuvem. A medição foi realizada com o osciloscópio digital *Tektronix TBS 1052B*.

Além da medição do consumo de corrente do *ESP32*, também foram realizadas medições no microcontrolador *MSP430FR2355*. Como seu consumo é muito baixo, não foi possível visualizá-lo com precisão no osciloscópio. Por isso, utilizou-se um amperímetro digital *Siglent SDM3065X* e um resistor de $100\ \Omega$ em série para medir a corrente de forma indireta, via queda de tensão. Para aumentar a sensibilidade da leitura, foi adotado um ganho resistivo com resistor de $10\ k\Omega$, resultando em um ganho de $6x$. Em teste com

tensão de 6 V e carga de $6\ \Omega$, a corrente medida foi de $166,67\text{ mA}$. Já em operação real, o *MSP430* consumiu cerca de $0,001\text{ mA}$, enquanto o *ESP32* apresentou consumo ativo de $30,00\text{ mA}$. A Figura 4.4 mostra o perfil de corrente do *MSP430*, confirmando seu baixo consumo e justificando sua escolha para tarefas no sistema.

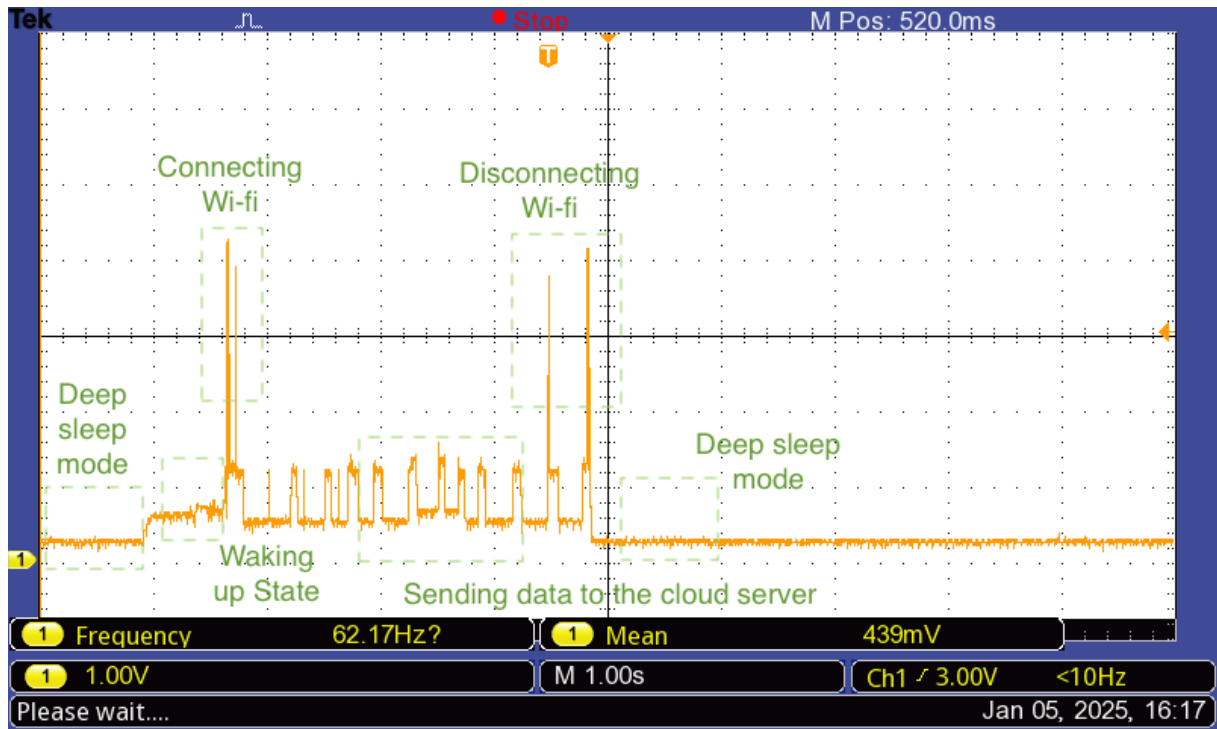


Figura 4.3: Perfil de consumo de corrente do *ESP32* durante o acionamento, comunicação e retorno ao modo de *deep sleep*.

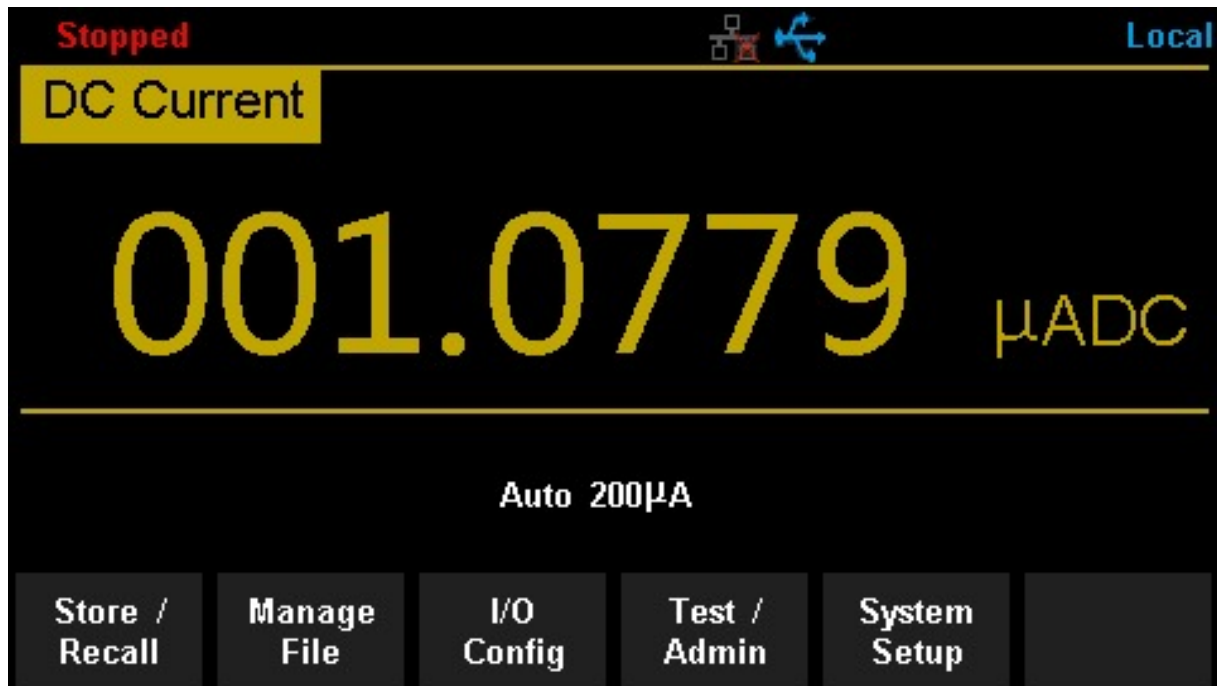


Figura 4.4: Perfil de consumo de corrente do *MSP430FR2355* em operação de ultra-baixo consumo.

Nesta versão inicial do sistema, a comunicação sem fio é implementada utilizando a tecnologia *Wi-Fi* por meio do microcontrolador *ESP32*. Como resultado, a cobertura é, inerentemente, limitada pelo alcance padrão das redes *Wi-Fi*. Em condições típicas de ambientes internos, o alcance da comunicação atinge aproximadamente *50 m*, o que é suficiente para pequenas instalações agrícolas, galpões de confinamento ou ambientes controlados. No entanto, para instalações maiores ou implantações em áreas com obstruções físicas significativas, estratégias de extensão de cobertura, como repetidores *Wi-Fi*, redes em malha (*mesh networking*) ou a integração de tecnologias de comunicação de longo alcance, como *LoRa*, poderão ser consideradas em versões futuras do sistema.

4.2 Resultados da Validação de Software

4.2.1 Validação da API

O processo de validação concentrou-se em duas *APIs*: a *API de Processamento e Predição de Dados* e a *API de Treinamento*. Ambas as *APIs* apresentaram o comportamento esperado, demonstrando sua capacidade de manipular dados, integrar-se com a rede de sensores e executar suas respectivas tarefas. A *API de Processamento e Predição de Dados* processou dados ambientais provenientes da rede de sensores e os complementou com dados meteorológicos externos. Ela determinou os estados apropriados dos atuadores (por exemplo, ligar ou desligar os atuadores) e sinalizou cenários que requeriam ações adicionais, como o treinamento de algoritmos de aprendizado de máquina.

Como mostrado na Figura 4.5a, o sistema transmitiu e recebeu dados dos sensores com sucesso. Em cenários onde o treinamento era necessário, a *API* configurou o sinalizador *SetToTrain* como *true*, indicando a necessidade de treinamento. Após a conclusão do processo de treinamento pela *API de Treinamento*, os dados subsequentes enviados à *API de Recebimento e Processamento de Dados* redefiniram o sinalizador *SetToTrain* como *false*, conforme esperado, seguindo a configuração de treinamento apenas durante os intervalos de tempo predefinidos especificados para cada galpão. Esse mecanismo garante que o sistema opere evitando treinamentos desnecessários, ao mesmo tempo em que mantém capacidades preditivas adaptadas às exigências de cada galpão.

A *API de Treinamento* foi responsável pela preparação e pelo treinamento dos algoritmos de aprendizado de máquina. Ela processou os dados formatados dos sensores e das previsões meteorológicas para treinar múltiplos algoritmos, selecionando o modelo com os melhores indicadores de desempenho. O modelo treinado foi salvo para uso futuro tanto em armazenamento na nuvem quanto no banco de dados. A Figura 4.5b apresenta as entradas e saídas durante a operação da *API*. Após a conclusão do treinamento, a *API* redefiniu o sinalizador *SetToTrain* como *false*, garantindo que a *API de Recebimento e Processamento de Dados* continuasse funcionando sem reativar o treinamento de forma desnecessária.

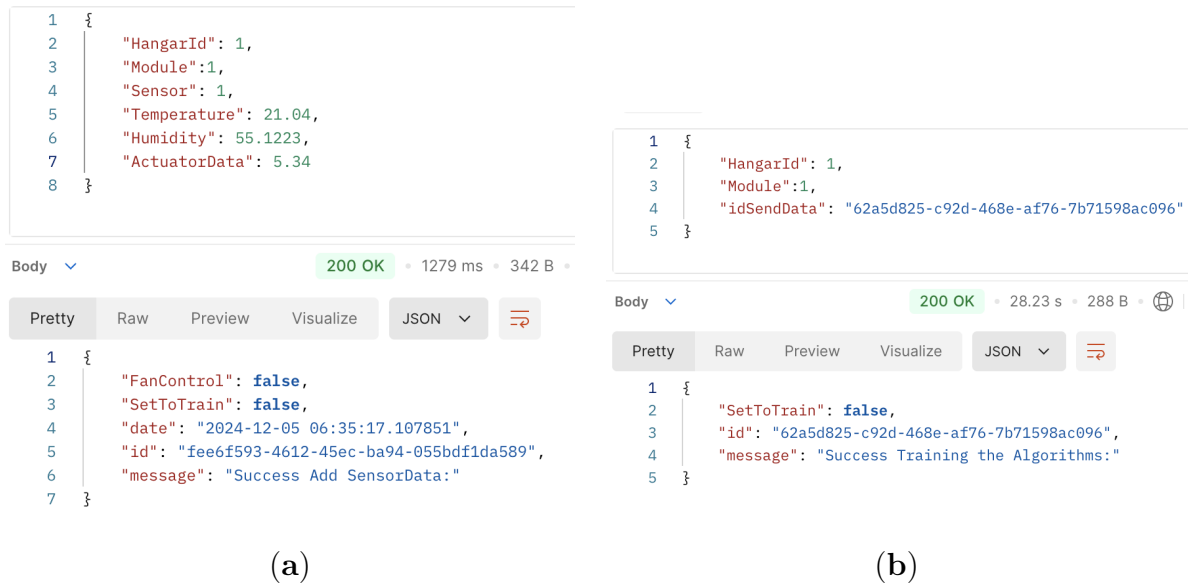


Figura 4.5: Visão geral das interações entre as APIs: (a) API de Processamento de Dados recebendo dados dos sensores da *SSN* e respondendo com comandos de controle dos atuadores ou sinalizadores de treinamento; (b) Chamadas e respostas da API de Treinamento, ilustrando o processo de treinamento e o retorno do status de treinamento atualizado para o sistema.

4.2.2 Resultados da Validação da Interface Web

A interface web foi testada, demonstrando sua capacidade de atender aos requisitos funcionais e não funcionais definidos. Os resultados destacam sua usabilidade, responsividade e integração com os serviços de *back-end*. A interface foi testada quanto à responsividade e acessibilidade em diversos dispositivos, incluindo computadores, tablets e smartphones. Conforme mostrado na Figura 4.6, o layout adapta-se dinamicamente a diferentes tamanhos de tela.

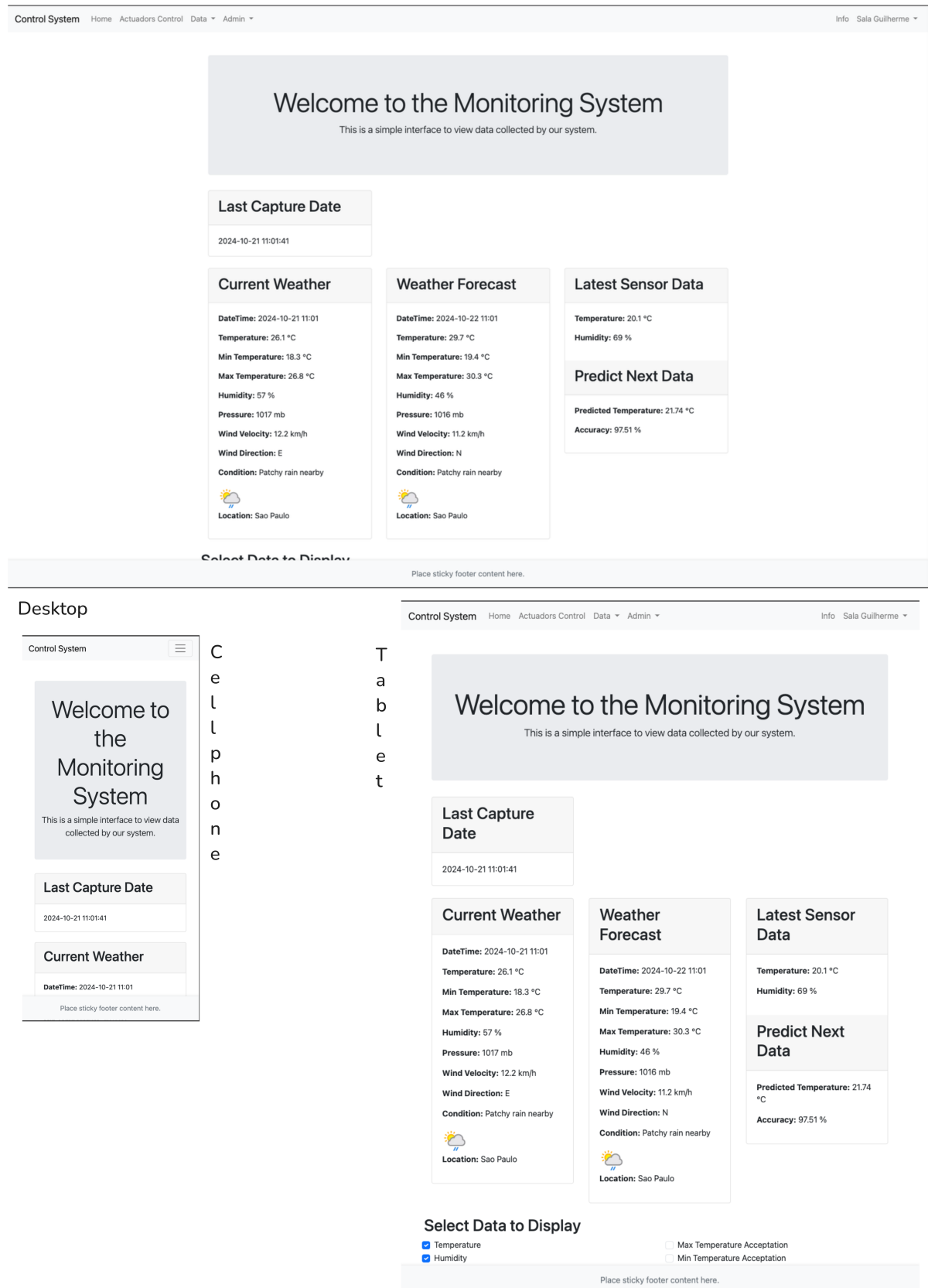


Figura 4.6: Design responsivo adaptando-se a múltiplos tipos de dispositivos.

Além da responsividade, a interface melhora a experiência do usuário ao fornecer uma plataforma para o monitoramento e controle das condições ambientais, integrando acesso

aos dados mais recentes dos sensores, condições climáticas e previsões do sistema, conforme ilustrado na Figura 4.6. Adicionalmente, os usuários podem interagir com um gráfico dinâmico, conforme mostrado na Figura 4.7, que visualiza tendências dos parâmetros ambientais ao longo de intervalos de tempo personalizáveis. Esse gráfico compara os valores internos do ambiente — como temperatura e umidade captados pela rede de sensores — com os valores externos, obtidos por meio de um serviço de previsão do tempo, permitindo uma análise contextualizada das condições dentro e fora do galpão.

Select Data to Display

- ☒ Temperature
- ☒ Humidity
- ☒ Weather Temperature
- ☐ Weather Humidity

Start Date:

Aug. 6, 2024, 9:38 p.m.

End Date:

Aug. 16, 2024, 9:38 p.m.

Filter

Sensor Data Chart

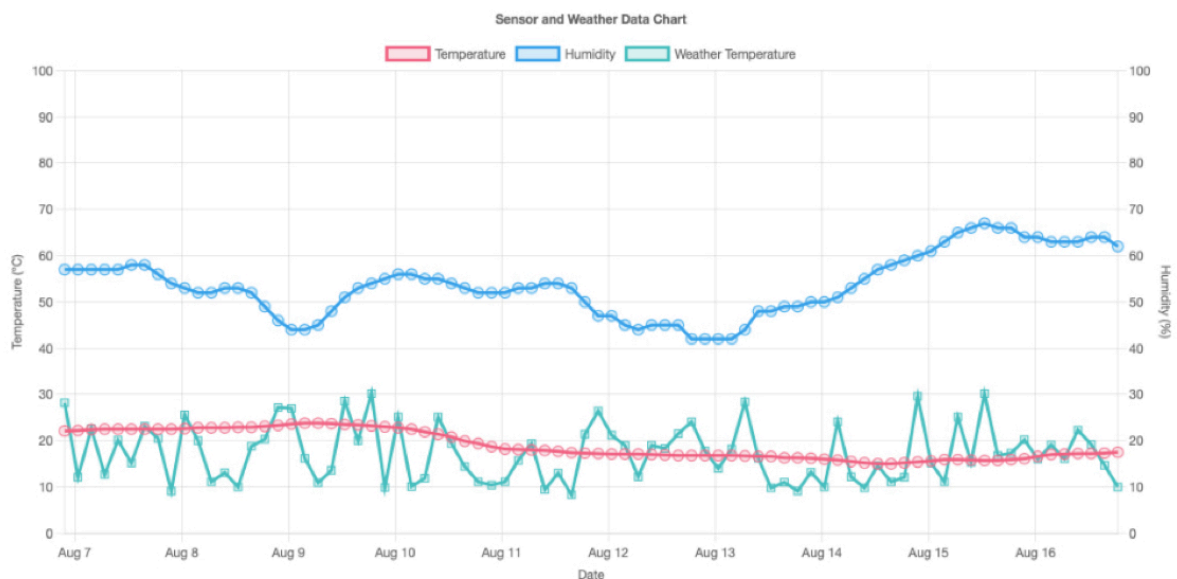


Figura 4.7: Gráfico dinâmico exibindo tendências nos dados dos sensores e meteorológicos ao longo de um intervalo de tempo selecionado.

Ao oferecer recursos de monitoramento e gerenciamento, a interface viabiliza a visualização de dados, geração de relatórios e monitoramento de erros, permitindo o acompanhamento das condições ambientais, com acesso às últimas informações dos sensores, dados meteorológicos e previsões do sistema. Além disso, a interface inclui funcionalidades de relatórios, permitindo aos usuários visualizar informações sobre dados dos sensores, dados meteorológicos, registros de erros, análises preditivas e desempenho dos atuadores, conforme ilustrado na Figura 4.8. Opções de filtro permitem a recuperação de intervalos de dados específicos, enquanto a funcionalidade de exportação possibilita salvar os relatórios em formatos *Excel* ou *CSV* para análise offline.

The screenshot displays the 'Control System' web interface. On the left is a sidebar menu with options: Home, Actuators Control, Data (expanded), Admin, Info, and Sala Guilherme. The 'Data' menu is open, showing 'Sensor Data', 'Weather Data', 'Predict Data', and 'Actuator Data'. The main content area is titled 'Sensor Data' and includes filters for 'Start Date and Time' (01/10/2024 09:23), 'End Date and Time' (04/12/2024 09:23), and 'Module ID' (All Modules). There are buttons for 'Filter' and 'Export All Data'. Below the filters, it shows 'Total: 113'. A table displays sensor data with columns: Module ID, Sensor ID, Temperature, and Humidity. The table contains three rows of data, all showing Module ID 1, Sensor ID 1, Temperature 20.10, and Humidity 69.00. A large 'Welcome to the Monitoring System' message is overlaid on the bottom left, with a placeholder for sticky footer content.

Module ID	Sensor ID	Temperature	Humidity
1	1	20.10	69.00
1	1	20.10	69.00
1	1	20.10	69.00

Figura 4.8: Funcionalidades de visualização de dados e relatórios, incluindo o monitoramento de erros.

A interface web inclui um recurso de controle dos atuadores que suporta configurações tanto manuais quanto automáticas. No modo manual, os usuários podem ligar ou desligar os atuadores e visualizar seu status atual. No modo automático, os usuários podem habilitar ou desabilitar a capacidade do sistema de controlar os atuadores com base em condições predefinidas e algoritmos preditivos. O sistema analisa os limites máximos e mínimos de temperatura configurados para o galpão, bem como a temperatura preditiva fornecida pelos algoritmos, para determinar se os atuadores devem ser ativados ou desativados, conforme mostrado na Figura 4.9.

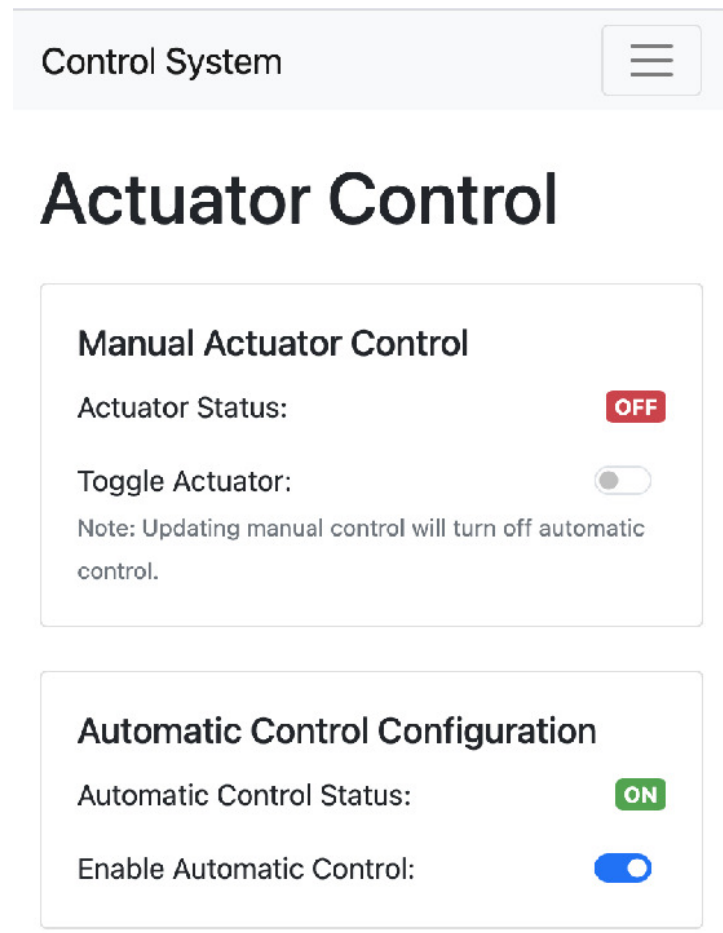


Figura 4.9: Interface de controle dos atuadores com configurações manuais e automáticas.

A Interface de Configuração dos Algoritmos, conforme ilustrado na Figura 4.10, oferece uma plataforma flexível para configurar tanto os parâmetros gerais quanto os específicos dos algoritmos do sistema preditivo. As configurações gerais incluem a ativação das etapas de treinamento de *machine learning*, a frequência de treinamento e a exibição do algoritmo selecionado de acordo com os critérios definidos de melhor desempenho, com base em métricas como *Mean Absolute Error (MAE)* e *R-squared (R^2)*.

Control System

General Settings

Prediction Algorithm Using

MachineLearnTraining False Flag to activate all the machine learning algorithms training	Algorithm Accuracy 97.50792767242184 The accuracy of the algorithm selected
Period_To_Train 10 Period to train the algorithms (Use always days)	Algorithm Selected Random Forest Algorithm name that is used to predict data
Mean Absolute Error (MAE) 0.5159405050505046 Mean Absolute Error (MAE) of the algorithm selected	Algorithm R2 0.9255899728247063 Algorithm R2

Random Forest

RF_TestSize 0.25 Test size to check the training of random forest algorithm	RF_RandomState 42 Random State to train random forest algorithm
--	--

Edit

Recurrent Neural Network

RNN_batch_size 32 batch_size to neural network	RNN_patience_rlr 0.2 patience_rlr to neural network
RNN_Config_StepsBack 15 How many history RNN will need to make a prediction	RNN_units_first_layer 100 units_first_layer to neural network
RNN units second layer	RNN middle layers number

Place sticky footer content here.

Figura 4.10: Interface para configuração dos algoritmos de predição, destacando opções flexíveis de *machine learning*.

Os usuários podem personalizar configurações para os algoritmos *Random Forest* e *RNN*, ajustando parâmetros para otimizar as previsões conforme contextos operacionais específicos. Essa interface elimina a necessidade de ajustes diretos no código, permitindo a adaptação às diferentes condições.

Para otimizar o desempenho, o *back-end* implementa *caching*, reduzindo a frequência de consultas ao banco de dados *DynamoDB*. Essa abordagem melhorou os tempos de resposta para dados acessados com frequência em aproximadamente 90%, além de minimizar o uso de recursos da *AWS* (Figura 4.11).

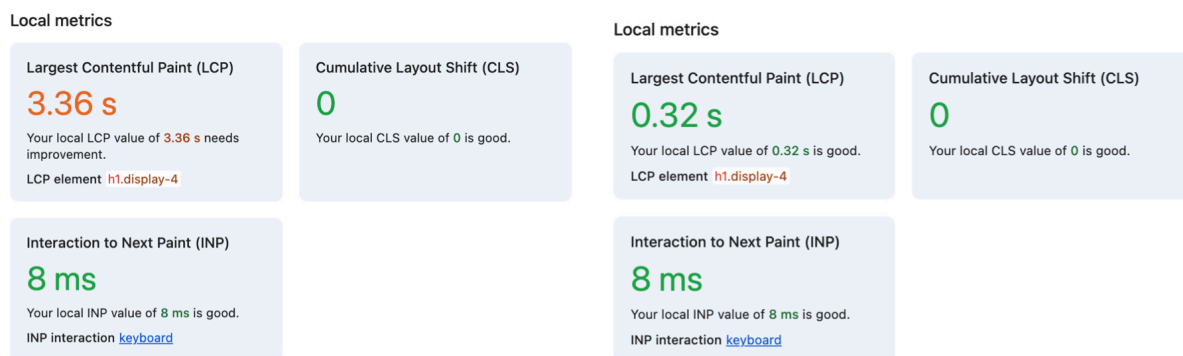


Figura 4.11: Estratégia de *caching* para melhorar o tempo de resposta e reduzir os custos com *AWS*.

4.3 Ecossistema Geral

O ecossistema geral do sistema de monitoramento foi avaliado para analisar seu desempenho na coleta e no processamento de dados ao longo de um período. Entre 30 de julho de 2024 e 21 de outubro de 2024, o sistema coletou dados ambientais a cada 3 horas, totalizando 624 leituras.

Para sustentar esse ecossistema, a *AWS* foi selecionada como provedora de serviços em nuvem após uma análise comparativa. Seu modelo de precificação sob demanda permite o pagamento apenas pelos recursos efetivamente utilizados, otimizando a eficiência de custos. Além disso, sua infraestrutura global garante 99,99% de disponibilidade e redundância, atendendo aos requisitos de escalabilidade, eficiência e confiabilidade do projeto [93].

Dentro desse ecossistema, o banco de dados *NoSQL DynamoDB* foi escolhido como solução de armazenamento, em função de sua integração nativa com os serviços da *AWS*, suporte a escalonamento automático e modelo de dados flexível, que atende aos requisitos de alta frequência de escrita, leitura eficiente e gerenciamento de múltiplos ambientes controlados.

Esta subseção fornece uma visão geral dos erros encontrados durante esse período e avalia o alinhamento do sistema com os objetivos definidos de precisão e eficiência.

O intervalo de coleta de dados foi configurado para três horas a fim de equilibrar os custos operacionais e a precisão dos dados, evitando custos mais elevados associados a intervalos mais curtos. Essa configuração foi definida com base em critérios gerais de eficiência energética e frequência de variação ambiental. No entanto, o sistema foi projetado para ser flexível, permitindo que o intervalo de coleta seja ajustado conforme a necessidade específica de cada aplicação. Dependendo da cultura ou do tipo de produção

mantida no galpão, pode haver a necessidade de coletas mais frequentes — para ambientes com alta sensibilidade térmica — ou menos frequentes, em cenários com menor variabilidade ambiental.

O uso do *DynamoDB* em um período de 30 dias incluiu 24.600 escritas e 24.600 leituras, calculado a *USD* 1,25 por milhão de escritas e *USD* 0,25 por milhão de leituras, resultando em um custo mensal estimado de *USD* 19,19 [101]. A distribuição dos custos entre as tabelas é a seguinte:

- **Tabela de Dados dos Sensores:** 14.760 writes + 14.760 reads = 11,51 *USD*/mês.
- **Tabela de Dados Meteorológicos:** 4.920 writes + 4.920 reads = 3,84 *USD*/mês.
- **Tabela de Predições:** 2.460 writes + 2.460 reads = 1,92 *USD*/mês.
- **Tabela de Dados dos Atuadores:** 1.230 writes + 1.230 reads = 0,96 *USD*/mês.
- **Tabela de Configuração dos Galpões:** 615 reads = 0,08 *USD*/mês.
- **Tabela de Configuração da API:** 615 reads = 0,08 *USD*/mês.

O custo total por ciclo (uma instância de coleta de dados) considerando todas as tabelas é de aproximadamente: $0,048 + 0,016 + 0,008 + 0,004 + 0,00034 + 0,00034 = 0,077$ *USD*/ciclo.

Reduzir o intervalo para 2 horas aumentaria o número de operações em 50%, elevando os custos para aproximadamente *USD* 28,79 por mês. Embora esse valor ainda fique abaixo do custo mensal de *USD* 30,24 de uma instância *AWS EC2* (t4g.nano na região leste dos Estados Unidos), a pequena margem introduz um risco financeiro maior devido a possíveis flutuações no volume de dados e cobranças adicionais da *AWS* [101].

O intervalo de 3 horas foi escolhido para manter a eficiência de custos e a previsibilidade financeira, atendendo aos requisitos de precisão e operação do sistema, garantindo a sustentabilidade sem se aproximar dos custos de configurações alternativas.

A Tabela 4.1 resume a fatura da *AWS* para o sistema entre 1º e 21 de outubro de 2024. A maior parte do custo (*USD* 29,78) foi atribuída às operações do *DynamoDB*, enquanto serviços como *EC2 Container Registry (ECR)* e *Data Transfer* geraram encargos menores de *USD* 0,03 e *USD* 0,02, respectivamente. Não foram observadas cobranças significativas para outros serviços como *API Gateway*, *CloudWatch*, *Lambda* ou *S3*.

Descrição	Quantidade de Uso	Valor (USD)
<i>DynamoDB</i>	-	29,78
<i>EC2 Container Registry (ECR)</i>	-	0,03
<i>Data Transfer</i>	-	0,02
<i>API Gateway</i>	-	0,00
<i>CloudWatch</i>	-	0,00
<i>Elastic Compute Cloud</i>	-	0,00
<i>Key Management Service</i>	-	0,00
<i>Lambda</i>	-	0,00
<i>Simple Storage Service (S3)</i>	-	0,00
<i>Virtual Private Cloud (VPC)</i>	-	0,00
Impostos Totais	-	4,12

Tabela 4.1: Resumo da fatura da AWS para outubro de 2024.

A *AWS Free Tier* ajudou a compensar os custos gerais, especialmente no que diz respeito às unidades de leitura e escrita do *DynamoDB*. No entanto, as despesas totais excederam as estimativas iniciais devido a atividades adicionais, como testes no ambiente de desenvolvimento, uso da interface web para visualização e geração de relatórios, além da inclusão de impostos e outras cobranças incidentais.

Durante o período de coleta de dados, foram registrados 21 erros de processamento, resultando em uma taxa de erro de aproximadamente 3,37%. Esses erros foram analisados para identificar padrões comuns de falhas e melhorar a confiabilidade do sistema. Os erros foram categorizados em quatro tipos principais com base em suas causas e impacto no processamento de dados, conforme ilustrado na Figura 4.12:

- **missing_data_error**: Ocorre quando o sistema falha ao registrar ou recuperar os dados necessários dos sensores, resultando em entradas de dados incompletas.
- **sensor_data_error**: Acontece quando as leituras dos sensores são inválidas ou estão fora da faixa esperada, possivelmente devido a falhas de hardware ou interferência ambiental.
- **weather_forecast_fetch_error**: Representa falhas na obtenção de dados de previsão do tempo a partir da *API* externa, possivelmente devido a problemas de conectividade ou indisponibilidade da *API*.
- **loadingfile_error**: Indica problemas no acesso ou no processamento de arquivos necessários para as operações do sistema, como o conhecimento usado por algoritmos de *machine learning*.

A distribuição dos erros reflete a robustez do sistema, que manteve uma taxa de acurácia dos dados superior a 96%. Esse desempenho supera a meta de 95% de acurácia estabelecida nos objetivos.

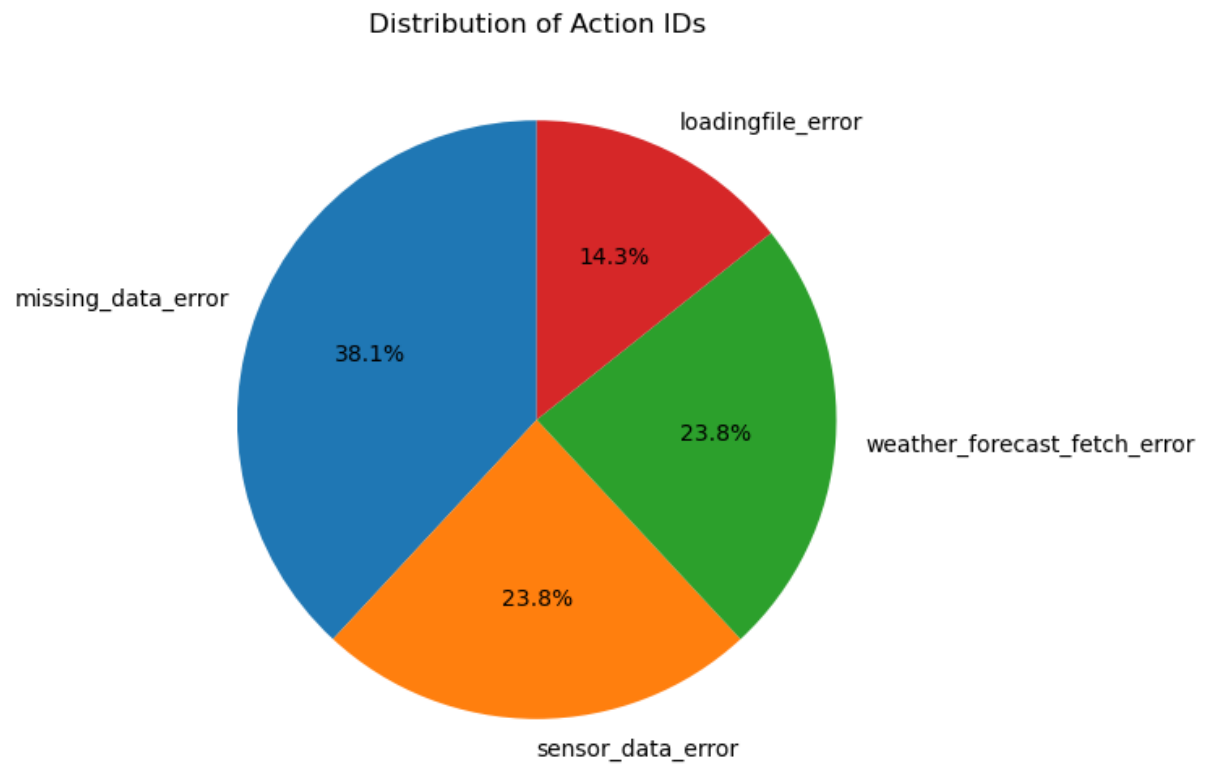


Figura 4.12: Distribuição dos erros no sistema de monitoramento durante o período de coleta de dados.

4.4 Validação dos Algoritmos Preditivos

Para atender às considerações relacionadas à análise estatística, os modelos *Random Forest (RF)* e *Recurrent Neural Network (RNN)* foram treinados e validados utilizando dados históricos dos sensores coletados pelo sistema entre julho de 2024 e setembro de 2024. A avaliação utilizou métricas padrão para medir o desempenho preditivo, incluindo *Mean Absolute Error (MAE)*, *Root Mean Squared Error (RMSE)*, erros relativos e o coeficiente de determinação *R-squared (R^2)*. Os resultados estão apresentados na Tabela 4.2.

Métrica	Random Forest	Recurrent Neural Network
Mean Absolute Error (MAE)	0,4114	1,0342
Mean Squared Error (MSE)	0,3550	1,7452
Root Mean Squared Error (RMSE)	0,5959	1,3211
Relative Absolute Error (RAE)	0,2175	0,4483
Relative Squared Error (RSE)	0,0599	0,2110
R-squared (R^2)	0,9401	0,7890
Acurácia (%)	98,05	95,20

Tabela 4.2: Métricas de desempenho dos modelos preditivos (conjunto de teste: 07/2024 – 09/2024).

Com base nessas métricas, o modelo *Random Forest* demonstrou desempenho preditivo superior, especialmente em relação às taxas de erro e ao valor de R^2 , que indica maior poder explicativo. Isso justificou sua seleção como modelo principal para uso.

Para validar a acurácia preditiva do sistema na previsão de temperatura em condições reais de operação, ambos os modelos foram avaliados com dados coletados em outubro de 2024. Nesta fase, os modelos já estavam treinados com dados históricos, e as previsões foram geradas continuamente ao longo do mês. Ao final do período, os valores de temperatura predita foram comparados com as leituras reais registradas nos respectivos intervalos de tempo, a fim de avaliar sua confiabilidade prática. Conforme mostrado na Figura 4.13, o modelo *Random Forest* atingiu uma acurácia geral de 96,46%, superando a meta deste trabalho de pesquisa, que era de 80%.

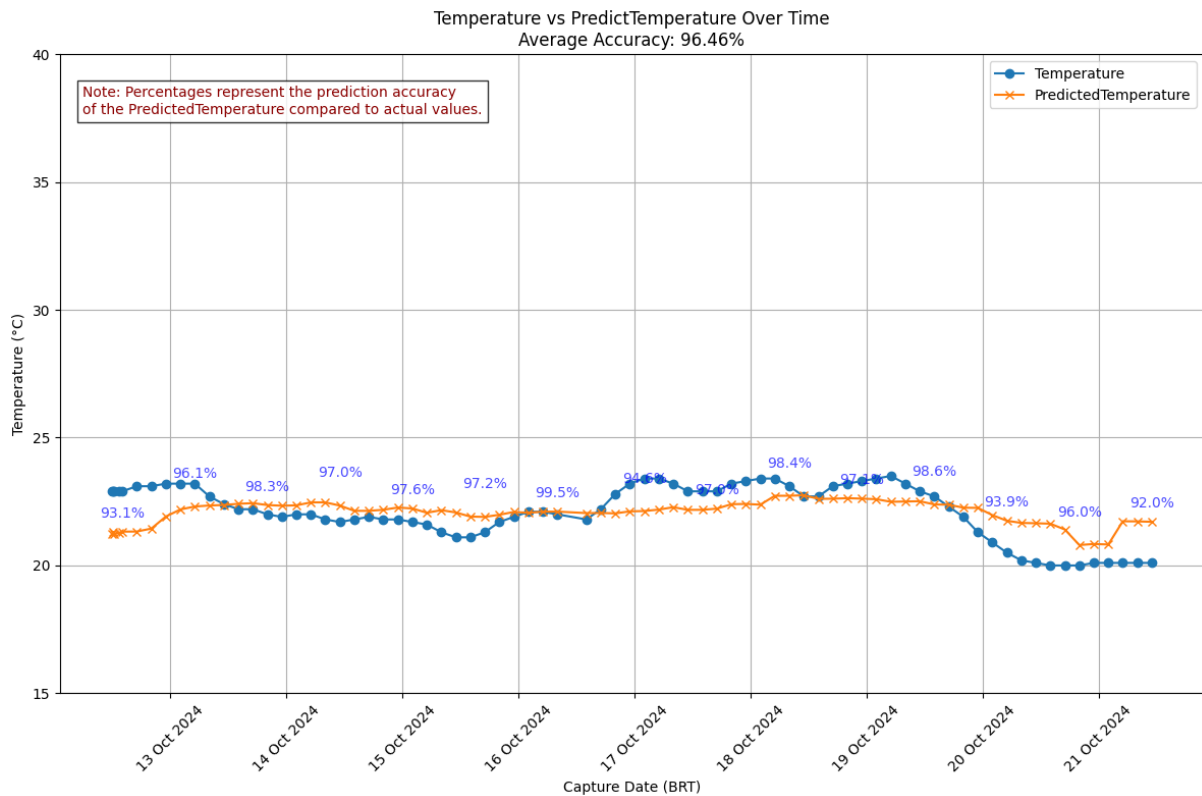


Figura 4.13: Acurácia do modelo *Random Forest* ao longo do tempo (média: 96,46%).

De forma semelhante, conforme ilustrado na Figura 4.14, o modelo *RNN* alcançou uma acurácia geral de 96,24%, confirmando sua robustez para predição, embora com métricas de treinamento ligeiramente inferiores em comparação ao *Random Forest*.

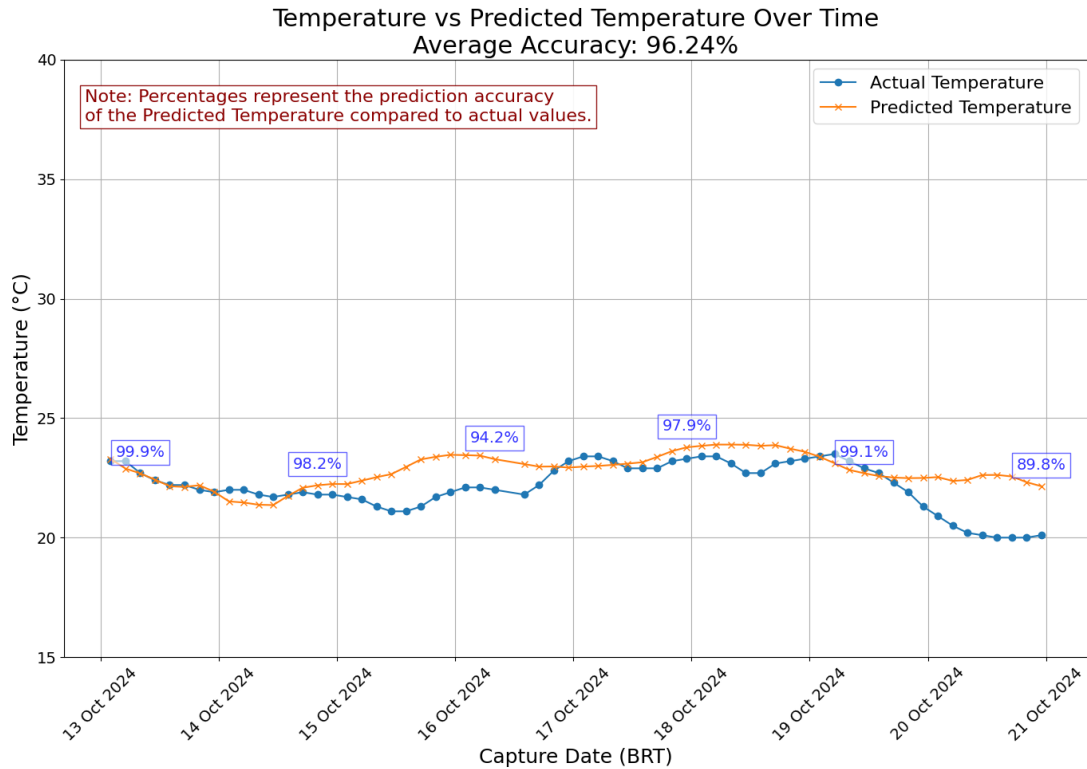


Figura 4.14: Acurácia do modelo *RNN* ao longo do tempo (média: 96,24%).

Embora ambos os modelos tenham mantido alta acurácia durante sua utilização operacional, o desempenho ligeiramente inferior observado em outubro, em comparação com a fase de testes, pode ser atribuído à natureza dinâmica e imprevisível das condições ambientais — especialmente fatores climáticos externos que não estavam totalmente representados no conjunto de dados de treinamento. Esses resultados reforçam a importância da avaliação contínua dos modelos e demonstram a capacidade do sistema de fornecer previsões confiáveis mesmo sob condições variáveis de campo.

5 Trabalhos Futuros

Este estudo validou com sucesso um ecossistema de monitoramento e controle ambiental baseado em *IoT* para aplicações agrícolas, demonstrando sua acurácia, eficiência preditiva e viabilidade para implantação em cenários reais. No entanto, diversas melhorias ainda são necessárias para aumentar a escalabilidade, otimizar o desempenho e adaptar o sistema a diferentes contextos agrícolas.

Embora o sistema tenha sido inicialmente testado em um ambiente controlado, pesquisas futuras envolverão implantações de longo prazo em campo, abrangendo diferentes ambientes agrícolas, como galpões, instalações de armazenamento e diversas regiões geográficas. Essas implantações permitirão avaliar a robustez do sistema sob condições operacionais e ambientais variadas, ao mesmo tempo em que fornecerão dados reais essenciais para o refinamento dos modelos preditivos e das estratégias de controle dos atuadores.

As investigações futuras abordarão as seguintes direções de pesquisa:

- **Expansão da Sensoriamento Ambiental:** Expandir a rede de sensores por meio da integração de sensores de gás para detecção de amônia (NH_3), dióxido de carbono (CO_2), monóxido de carbono (CO) e metano (CH_4), com o objetivo de aprimorar a modelagem ambiental e melhorar as previsões relacionadas ao bem-estar animal e à gestão das instalações.
- **Mecanismos de Atualização Remota:** Investigar a viabilidade e o impacto da implementação de mecanismos de atualização de *firmware over-the-air* (OTA), facilitando a escalabilidade e manutenção da rede de sensores em ambientes agrícolas.
- **Otimização da Arquitetura do Sistema:** Explorar arquiteturas alternativas de *backend*, incluindo a migração das *APIs* para linguagens mais rápidas como *Golang*, visando melhorar a escalabilidade do sistema, a capacidade de resposta em tempo real e a eficiência operacional diante do aumento no volume de dados.
- **Avaliação de Usabilidade:** Realizar estudos de campo com usuários e pesquisas de design participativo para avaliar e aprimorar a usabilidade da interface web, garantindo que ela apoie efetivamente a tomada de decisão operacional por parte de gestores de instalações agrícolas.
- **Expansão da Modelagem Preditiva:** Estender os modelos preditivos para incluir parâmetros ambientais adicionais, como umidade, índices de qualidade do ar e concentrações específicas de gases, ampliando assim as capacidades proativas de controle do sistema.
- **Análise de Infraestruturas Alternativas:** Conduzir estudos de viabilidade técnica e econômica para avaliar o uso de serviços em nuvem de menor custo, bem como arqui-

teturas híbridas ou locais (*edge computing*), como alternativas à atual infraestrutura baseada na *AWS*. Essa análise visa identificar opções que mantenham a confiabilidade e escalabilidade do sistema, mas com custos operacionais mais acessíveis para adoção por produtores de pequeno e médio porte.

Além disso, avanços recentes em tecnologias de processamento de imagem [102, 103, 104, 105] destacam o potencial de integração de sensores de imagem de baixo custo e algoritmos leves de análise visual. Iterações futuras do sistema *FarmSync* explorarão a incorporação de fluxos de dados visuais para permitir tarefas de classificação e monitoramento, avançando em direção ao desenvolvimento de um arcabouço de **gêmeo digital** — uma representação virtual das condições e comportamentos do ambiente físico — para a gestão de instalações agrícolas.

Essas direções de pesquisa visam fortalecer a confiabilidade operacional do sistema, ampliar sua aplicabilidade em diferentes ambientes agrícolas e contribuir para o avanço de práticas agrícolas sustentáveis e baseadas em tecnologia. Ao integrar modelagem preditiva, aquisição multissensorial de dados e análise visual, o sistema *FarmSync* aspira evoluir para uma plataforma de **gêmeo digital**, oferecendo inteligência e adaptabilidade para operações agrícolas.

6 Conclusão

Este estudo desenvolveu e validou um ecossistema baseado em *IoT* para monitoramento ambiental e controle preditivo em galpões agrícolas, alcançando mais de 95% de acurácia nos dados e 96% de eficiência preditiva com os modelos *Random Forest* e *RNN*, superando a meta inicial de desempenho de 80%. A arquitetura modular de hardware e software garante escalabilidade, enquanto a interface web responsiva permite o gerenciamento contínuo dos dados em diversos dispositivos.

A estrutura de custos do sistema favorece a escalabilidade, aumentando proporcionalmente com a expansão da produção do usuário, e a eficiência energética é obtida por meio do gerenciamento inteligente dos atuadores, otimizando o consumo de energia ao ativá-los apenas quando necessário. Essas características reforçam a viabilidade econômica e operacional do sistema, especialmente quando comparado a soluções convencionais de monitoramento agrícola baseadas em nuvem.

Embora o sistema tenha atingido os objetivos de validação em condições controladas, pesquisas futuras se concentrarão na implantação e validação do sistema em ambientes agrícolas reais. Trabalhos adicionais explorarão a expansão das capacidades de sensoria-mento ambiental com a integração de sensores de gases, a implementação de mecanismos de atualização *over-the-air* (OTA) para melhorar a escalabilidade e a manutenção, e a otimização do desempenho do banco de dados para implantações em larga escala. Além disso, avanços em sensores de imagem de baixo custo e algoritmos de processamento visual serão investigados para estender as capacidades do sistema em direção à análise ambiental baseada em imagens, apoiando o desenvolvimento de uma estrutura de **gêmeo digital** para a gestão de instalações agrícolas.

Referências

- [1] BRAGA, A. d. P.; CARVALHO, A.; LUDERMIR, T. B. Redes neurais artificiais. *Teoria e Aplicações*, Livros Técnicos e Científicos, 2000.
- [2] HAYKIN, S. Neural networks: A guided tour. *Nonlinear Biomedical Signal Processing*, Wiley Online Library, v. 1, p. 53–68, 2000.
- [3] SUNG, W.-T.; LU, C.-Y. Smart warehouse management based on iot architecture. In: *2018 international symposium on computer, consumer and control (IS3C)*. [S.l.]: IEEE, 2018. p. 169–172.
- [4] FUBING. Research on the agriculture intelligent system based on iot. In: *2012 International Conference on Image Analysis and Signal Processing*. [S.l.]: IEEE, 2012. p. 1–4.
- [5] LAMSAL, R. R. et al. Implementing internet of things for real-time monitoring and regulation of off-season grafting and post-harvest storage in citrus cultivation: A case study from the hilly regions of nepal. *AgriEngineering*, MDPI, v. 6, n. 3, p. 2082–2100, 2024.
- [6] BANERJEE, S. et al. Iot instrumented food and grain warehouse traceability system for farmers. In: *2020 international conference on artificial intelligence and signal processing (AISP)*. [S.l.]: IEEE, 2020. p. 1–4.
- [7] KODALI, R. K.; JOHN, J.; BOPPANA, L. Iot monitoring system for grain storage. In: *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. [S.l.]: IEEE, 2020. p. 1–6.
- [8] MARAVEAS, C.; BARTZANAS, T. Application of internet of things (iot) for optimized greenhouse environments. *AgriEngineering*, MDPI, v. 3, n. 4, p. 954–970, 2021.
- [9] YADAV, S. et al. Disruptive technologies in smart farming: an expanded view with sentiment analysis. *AgriEngineering*, MDPI, v. 4, n. 2, p. 424–460, 2022.
- [10] VILLAGRAN, E. et al. Integration of iot technologies and high-performance phenotyping for climate control in greenhouses and mitigation of water deficit: A study of high-andean oat. *AgriEngineering*, MDPI, v. 6, n. 4, p. 4011–4040, 2024.
- [11] OLIVEIRA, F. et al. Dependability evaluation of a smart poultry house: Addressing availability issues through the edge, fog, and cloud computing. *IEEE Transactions on Industrial Informatics*, IEEE, v. 20, n. 2, p. 1304–1312, 2023.

-
- [12] THWIN, K. M. M.; HORANONT, T.; PHATRAPORNNANT, T. Machine-learning microclimate forecasting for adaptive equipment control via web integration in open-ventilated greenhouses. *AgriEngineering*, MDPI, v. 6, n. 3, p. 2845–2869, 2024.
- [13] BÖKLE, S. et al. A concept of a decentral server infrastructure to connect farms, secure data, and increase the resilience of digital farming. *Smart Agricultural Technology*, Elsevier, v. 10, p. 100701, 2025.
- [14] GEHLOT, A. et al. Dairy 4.0: Intelligent communication ecosystem for the cattle animal welfare with blockchain and iot enabled technologies. *Applied Sciences*, MDPI, v. 12, n. 14, p. 7316, 2022.
- [15] PROVOLO, G. et al. An internet of things framework for monitoring environmental conditions in livestock housing to improve animal welfare and assess environmental impact. *Animals*, MDPI, v. 15, n. 5, p. 644, 2025.
- [16] RUAN, J. et al. A life cycle framework of green iot-based agriculture and its finance, operation, and management issues. *IEEE communications magazine*, IEEE, v. 57, n. 3, p. 90–96, 2019.
- [17] ZHANG, Q. et al. Research on warehouse environment monitoring system based on wireless sensor network. In: *2014 9th IEEE Conference on Industrial Electronics and Applications*. [S.l.]: IEEE, 2014. p. 1639–1644.
- [18] LYDIA, J. et al. Automated food grain monitoring system for warehouse using iot. *Measurement: Sensors*, Elsevier, v. 24, p. 100472, 2022.
- [19] MOTA, A.; BRIGA-SÁ, A.; VALENTE, A. Development of a wireless system to control a trombe wall for poultry brooding. *AgriEngineering*, MDPI, v. 3, n. 4, p. 853–867, 2021.
- [20] AIRA, J.; OLIVARES, T.; DELICADO, F. M. Driftgly: A low-cost iot-based ecosystem for monitoring agrochemical spray drifts. *IEEE Transactions on Instrumentation and Measurement*, IEEE, 2024.
- [21] SYMEONAKI, E.; MARAVEAS, C.; ARVANITIS, K. G. Recent advances in digital twins for agriculture 5.0: applications and open issues in livestock production systems. *Applied Sciences*, MDPI, v. 14, n. 2, p. 686, 2024.
- [22] SEBALD, C. et al. Usability testing of novel iot-infused digital services on farm equipment reveals farmer’s requirements towards future human–machine interface design guidelines. *AgriEngineering*, Multidisciplinary Digital Publishing Institute, v. 6, n. 2, p. 1660–1673, 2024.
- [23] ARULMOZHI, E. et al. Development and validation of low-cost indoor air quality monitoring system for swine buildings. *Sensors*, MDPI, v. 24, n. 11, p. 3468, 2024.
- [24] TERENCE, S. et al. Systematic review on internet of things in smart livestock management systems. *Sustainability*, MDPI, v. 16, n. 10, p. 4073, 2024.
- [25] EESWARAN, R. et al. Current and future challenges and opportunities for livestock farming in west africa: Perspectives from the case of senegal. *Agronomy*, MDPI, v. 12, n. 8, p. 1818, 2022.

-
- [26] United Nations. United Nations Official Website. n.d. Accessed: 2024-09-01. Disponível em: <<https://www.un.org/en/>>.
- [27] LELIVELD, L. M. et al. Real-time automatic integrated monitoring of barn environment and dairy cattle behaviour: Technical implementation and evaluation on three commercial farms. *Computers and Electronics in Agriculture*, Elsevier, v. 216, p. 108499, 2024.
- [28] OJO, R. O. et al. Internet of things and machine learning techniques in poultry health and welfare management: A systematic literature review. *Computers and Electronics in Agriculture*, Elsevier, v. 200, p. 107266, 2022.
- [29] IBGE. Censo agropecuário 2017: Agricultura familiar. 2017. Agência de Notícias - IBGE. Disponível em: <https://agenciadenoticias.ibge.gov.br/agencia-sala-de-imprensa/2013-agencia-de-noticias/releases/25789-censo-agro-2017-populacao-ocupada-nos-estabelecimentos-agropecuarios-cai-8-8>. Acesso em: 17 out. 2024.
- [30] ZANDER, K. *Agricultura Familiar no Brasil: Desafios e Oportunidades*. São Paulo: Cortez, 2013.
- [31] LOUREIRO, A. A. et al. Redes de sensores sem fio. In: SN. *Simpósio Brasileiro de Redes de Computadores (SBRC)*. [S.l.], 2003. v. 21, p. 179–226.
- [32] CARVALHO, F. B. S. de et al. Aplicacoes ambientais de redes de sensores sem fio. *Revista de Tecnologia da Informação e Comunicação*, v. 2, n. 1, p. 14–19, 2012.
- [33] SOUSA, M. P.; LOPES, W. T. A. Desafios em redes de sensores sem fio. *dados*, v. 2, p. 3, 2011.
- [34] SOTOMAIOR, V. *Rede de Sensores Sem Fios Para Veículo Frigorífico e/ou Industria Alimentar Indoor*. Dissertação (Mestrado) — Universidade do Porto (Portugal), 2013.
- [35] AKYILDIZ, I. F. et al. A survey on sensor networks. *IEEE Communications magazine*, IEEE, v. 40, n. 8, p. 102–114, 2002.
- [36] KARL, H.; WILLIG, A. *Protocols and architectures for wireless sensor networks*. [S.l.]: John Wiley & Sons, 2007.
- [37] BOGENA, H. et al. Potential of wireless sensor networks for measuring soil water content variability. *Vadose Zone Journal*, Soil Science Society of America, v. 9, n. 4, p. 1002–1013, 2010.
- [38] BAGGIO, A. Wireless sensor networks in precision agriculture. In: CITESEER. *ACM workshop on real-world wireless sensor networks (REALWSN 2005)*, Stockholm, Sweden. [S.l.], 2005. v. 20, p. 1567–1576.
- [39] MARTIN, R. C. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. [S.l.]: Pearson Education, 2018.
- [40] EVANS, E. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. [S.l.]: Addison-Wesley, 2003.
- [41] SOMMERVILLE, I. *Software Engineering*. 8. ed. [S.l.]: Pearson Education, 2007.

-
- [42] MATTERN, F.; FLOERKEMEIER, C. From the internet of computers to the internet of things. *From active data management to event-based systems and more: Papers in honor of Alejandro Buchmann on the occasion of his 60th birthday*, Springer, p. 242–259, 2010.
- [43] VANELLI, B. et al. Internet of things data storage infrastructure in the cloud using nosql databases. *IEEE Latin America Transactions*, IEEE, v. 15, n. 4, p. 737–743, 2017.
- [44] BANDYOPADHYAY, D.; SEN, J. Internet of things: Applications and challenges in technology and standardization. *Wireless personal communications*, Springer, v. 58, p. 49–69, 2011.
- [45] CONDRY, M. W.; NELSON, C. B. Using smart edge iot devices for safer, rapid response with industry iot control operations. *Proceedings of the IEEE*, IEEE, v. 104, n. 5, p. 938–946, 2016.
- [46] ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer networks*, Elsevier, v. 54, n. 15, p. 2787–2805, 2010.
- [47] ZHANG, Q. et al. Research on warehouse environment monitoring system based on wireless sensor network. In: IEEE. *2014 9th IEEE Conference on Industrial Electronics and Applications*. [S.l.], 2014. p. 1639–1644.
- [48] BATISTA, E. d. S. et al. Comportamento de uso do ninho e desempenho produtivo de poedeiras alojadas em diferentes densidades e tamanhos de grupo. *Revista Educação Agrícola Superior*, v. 27, n. 2, p. 119–123, 2012.
- [49] CASTILHO, V. et al. Bem-estar de galinhas poedeiras em diferentes densidades de alojamento. *Revista Brasileira de Engenharia de Biosistemas*, v. 9, n. 2, p. 122–131, 2015.
- [50] PUTHAL, D. et al. Cloud computing features, issues, and challenges: a big picture. In: IEEE. *2015 International conference on computational intelligence and networks*. [S.l.], 2015. p. 116–123.
- [51] COOPER, B. F. et al. Building a cloud for yahoo! *IEEE Data Eng. Bull.*, v. 32, n. 1, p. 36–43, 2009.
- [52] AGRAWAL, D. et al. Database management as a service: Challenges and opportunities. In: IEEE. *2009 IEEE 25th international conference on data engineering*. [S.l.], 2009. p. 1709–1716.
- [53] FORTINO, G.; RUSSO, W. et al. Towards a cloud-assisted and agent-oriented architecture for the internet of things. In: CITESEER. *WOA@ AI* IA*. [S.l.], 2013. p. 60–65.
- [54] LEVINE, R. I. *Inteligência artificial e sistemas especialistas*. [S.l.]: McGraw-Hill, 1988.
- [55] AREL, I.; ROSE, D. C.; KARNOWSKI, T. P. Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *IEEE computational intelligence magazine*, IEEE, v. 5, n. 4, p. 13–18, 2010.

-
- [56] NAJAFABADI, M. M. et al. Deep learning applications and challenges in big data analytics. *Journal of big data*, Springer, v. 2, p. 1–21, 2015.
- [57] SILVA, I. N. d.; SPATTI, D. H.; FLAUZINO, R. A. Redes neurais artificiais para engenharia e ciências aplicadas. 2010.
- [58] FACELI, K. et al. Inteligência artificial: uma abordagem de aprendizado de máquina. 2021.
- [59] SANTOS, V. N. d. *Reconhecimento de objetos em uma cena utilizando redes neurais convolucionais*. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2018.
- [60] ALENCAR, F. E. S. Desenvolvimento de um sistema para classificação automática de imagens dermatoscópicas para dispositivos móveis. *Universidade do Estado do Rio Grande do Norte, Biblioteca Central Orlando Teixeira, BR-RN*, 2015.
- [61] SANTOS, C. A. S. d. Reconhecimento de imagens de marcas de gado utilizando redes neurais convolucionais e máquinas de vetores de suporte. Universidade Federal do Pampa, 2017.
- [62] RODRIGUEZ-GALIANO, V. et al. Random forest classification of mediterranean land cover using multi-seasonal imagery and multi-seasonal texture. *Remote Sensing of Environment*, Elsevier, v. 121, p. 93–107, 2012.
- [63] FENG, R. et al. Recurrent neural network and random forest for analysis and accurate forecast of atmospheric pollutants: a case study in hangzhou, china. *Journal of cleaner production*, Elsevier, v. 231, p. 1005–1015, 2019.
- [64] ELIJAH, O. et al. An overview of internet of things (iot) and data analytics in agriculture: Benefits and challenges. *IEEE Internet of things Journal*, IEEE, v. 5, n. 5, p. 3758–3773, 2018.
- [65] ALI, M. L.; RAHMAN, M. A.; TAUJUDDIN, N. S. A. M. Smart chicken farm monitoring system. *evol electr electron eng* 1 (1): 317–325. 2020.
- [66] BUDIARTO, R.; GUNAWAN, N. K.; NUGROHO, B. A. Smart chicken farming: Monitoring system for temperature, ammonia levels, feed in chicken farms. In: IOP PUBLISHING. *IOP Conference Series: Materials Science and Engineering*. [S.l.], 2020. v. 852, n. 1, p. 012175.
- [67] RIBEIRO, R. et al. Generating action plans for poultry management using artificial neural networks. *Computers and Electronics in Agriculture*, Elsevier, v. 161, p. 131–140, 2019.
- [68] INSTITUTE, P. M. *A Guide to the Project Management Body of Knowledge (PM-BOK Guide)*. 7th. ed. Newtown Square, PA: Project Management Institute, 2021.
- [69] SCHWABER, K.; SUTHERLAND, J. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. [s.n.], 2020. Disponível em: <<https://scrumguides.org>>.

-
- [70] BISHOP, C. M. *Pattern Recognition and Machine Learning*. New York: Springer, 2006. (Information Science and Statistics). ISBN 978-0387310732.
- [71] SOMMERVILLE, I. *Software Engineering*. 11th. ed. Harlow, England: Pearson, 2021.
- [72] SHINDE, V. R.; TASGAONKAR, P. P.; GARG, R. D. Environment monitoring system through internet of things (iot). In: *2018 international conference on information, communication, engineering and technology (ICICET)*. [S.l.]: IEEE, 2018. p. 1–4.
- [73] DIGI-KEY Electronics Brazil. <https://www.digikey.com.br/en>. Accessed: 2 April, 2025.
- [74] MOUSER Electronics Brazil. <https://br.mouser.com>. Accessed: 2 April, 2025.
- [75] SOLDERED 101262 — DHT22 Sensor at Mouser Electronics. <https://l1nq.com/gtIMi>. Accessed: 2 April, 2025.
- [76] ADAFRUIT 385 — DHT22 Sensor at Digi-Key Electronics. <https://www.digikey.com.br/en/products/detail/adafruit-industries-llc/385/5774230>. Accessed: 2 April, 2025.
- [77] ESP32-WROOM-32U-16MB — Mouser Electronics. <https://br.mouser.com/ProductDetail/Esspressif-Systems/ESP32-WROOM-32E-N4?qs=Li> Accessed: 2 April, 2025.
- [78] ESP32-WROOM-32E-N4 — Digi-Key Electronics. <https://www.digikey.com.br/en/products/detail/esspressif-systems/ESP32-WROOM-32E-N4/11613125>. Accessed: 2 April, 2025.
- [79] ZHAO, J. et al. Design of wireless temperature and humidity data collection system based on msp430 and cc2530. In: *2012 3rd International Conference on System Science, Engineering Design and Manufacturing Informatization*. [S.l.]: IEEE, 2012. v. 2, p. 193–195.
- [80] ZHOU, Y. et al. Wireless temperature & humidity monitor and control system. In: *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*. [S.l.]: IEEE, 2012. p. 2246–2250.
- [81] Texas Instruments. Msp430fr2355 mixed-signal microcontroller datasheet. 2023. <https://www.ti.com/lit/ds/symlink/msp430fr2355.pdf>. Accessed: 2 April, 2025.
- [82] SYSTEMS, E. Esp32 series datasheet. 2023. <https://www.espressif.com>. Accessed: 2 April, 2025.
- [83] MSP430FR2355TPT — Texas Instruments at Mouser Electronics. <https://br.mouser.com/ProductDetail/Texas-Instruments/MSP430FR2355TPT?qs=gTYE2QTfZfTjQOKK> Accessed: 2 April, 2025.
- [84] MSP430FR2355TPT — Texas Instruments at Digi-Key Electronics. <https://www.digikey.com.br/en/products/detail/texas-instruments/MSP430FR2355TPT/9597611>. Accessed: 2 April, 2025.

-
- [85] DUAN, S. et al. Forecasting of grain pile temperature from meteorological factors using machine learning. *IEEE access*, IEEE, v. 7, p. 130721–130733, 2019.
- [86] ARAÚJO, S. O. et al. Intelligent data-driven decision support for agricultural systems-id3sas. *IEEE Access*, IEEE, v. 11, p. 115798–115815, 2023.
- [87] NAYYAR, A.; PURI, V. Smart farming: Iot based smart sensors agriculture stick for live temperature and moisture monitoring using arduino, cloud computing & solar technology. In: *Proc. of The International Conference on Communication and Computing Systems (ICCCS-2016)*. [S.l.: s.n.], 2016. p. 9781315364094–121.
- [88] ZHANG, H.; ZHANG, L. A reliable data-driven control method for planting temperature in smart agricultural systems. *IEEE Access*, IEEE, v. 11, p. 38182–38193, 2023.
- [89] CODELUPPI, G. et al. Ai at the edge: a smart gateway for greenhouse air temperature forecasting. In: *2020 IEEE international workshop on metrology for agriculture and forestry (MetroAgriFor)*. [S.l.]: IEEE, 2020. p. 348–353.
- [90] RAYHANA, R.; XIAO, G.; LIU, Z. Internet of things empowered smart greenhouse farming. *IEEE journal of radio frequency identification*, IEEE, v. 4, n. 3, p. 195–211, 2020.
- [91] FENG, R. et al. Recurrent neural network and random forest for analysis and accurate forecast of atmospheric pollutants: a case study in hangzhou, china. *Journal of cleaner production*, Elsevier, v. 231, p. 1005–1015, 2019.
- [92] RODRIGUEZ-GALIANO, V. et al. Random forest classification of mediterranean land cover using multi-seasonal imagery and multi-seasonal texture. *Remote Sensing of Environment*, Elsevier, v. 121, p. 93–107, 2012.
- [93] SERVICES, A. W. Global infrastructure. 2024. Available at: <https://aws.amazon.com/about-aws/global-infrastructure/> (Accessed: 16 Oct. 2024).
- [94] SERVICES, A. W. Security documentation. 2024. Available at: <https://aws.amazon.com/security/> (Accessed: 16 Oct. 2024).
- [95] Microsoft Azure. Azure iot hub. 2024. Available at: <https://azure.microsoft.com/en-us/services/iot-hub/> (Accessed: 16 Oct. 2024).
- [96] Google Cloud Platform. Google cloud iot core. 2024. Available at: <https://cloud.google.com/iot-core> (Accessed: 16 Oct. 2024).
- [97] WeatherAPI. WeatherAPI Documentation. 2024. Accessed: 2024-07-01. Disponível em: <<https://www.weatherapi.com>>.
- [98] Bootstrap Table Contributors. Filter control extension. 2025. Available at: <https://bootstrap-table.com/docs/extensions/filter-control/> (Accessed: 29 March. 2025).
- [99] Bootstrap Table Contributors. Export extension. 2025. Available at: <https://bootstrap-table.com/docs/extensions/export/> (Accessed: 29 March. 2025).

-
- [100] DIVYASHREE, N.; KS, N. P. Design and development of we-cdss using django framework: Conducing predictive and prescriptive analytics for coronary artery disease. *IEEE Access*, IEEE, v. 10, p. 119575–119592, 2022.
 - [101] SERVICES, A. W. Aws pricing. 2024. Available at: <https://aws.amazon.com/pt/pricing/order> (Accessed: 12 Jan. 2025).
 - [102] WANG, N. et al. Structured doubly stochastic graph-based clustering. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, 2025.
 - [103] LAN, Y. et al. Exploiting diffusion prior for real-world image dehazing with unpaired training. *arXiv preprint arXiv:2503.15017*, 2025.
 - [104] SU, Y. et al. Real scene single image dehazing network with multi-prior guidance and domain transfer. *IEEE Transactions on Multimedia*, IEEE, 2025.
 - [105] SU, Y. Z. et al. Physical model and image translation fused network for single-image dehazing. *Pattern Recognition*, Elsevier, v. 142, p. 109700, 2023.