INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLGIA DE SANTA CATARINA CÂMPUS ITAJAÍ

YAN SCHNEIDER DE QUADROS

SISTEMA DE MONITORAMENTO DOS AQUÁRIOS DO INSTITUTO FEDERAL DE SANTA CATARINA - CÂMPUS ITAJAÍ

ITAJAÍ

2026

Yan Schneider de Quadros

Sistema de monitoramento dos aquários do instituto federal de Santa Catarina Câmpus-Itajaí

Monografia submetida ao curso de Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina - Campus Itajaí como requisito para a obtenção do diploma de bacharel em engenharia elétrica

Orientador: Prof. Dr. Roddy Alexander Romero Antayhua

ITAJAÍ

2026

Ficha de Identificação da obra elaborada pelo autor, através do cadastro de ficha de identificação disponível no portal discente do Sistema Integrado de Gestão Acadêmica - SIGAA, do IFSC.

Quadros, Yan Schneider de
Sistema de monitoramento dos aquários do instituto federal de santa catarina - câmpus itajaí / Yan Schneider de Quadros; Orientador(a): Dr. Roddy Alexander Romero Antayhua. - Itajaí, SC, 2026.
114 p.

Trabalho de conclusão de curso (Graduação) - Instituto Federal de Santa Catarina, Campus Itajaí. Curso de Bacharelado Em Engenharia Elétrica.

Inclui referências.

1. Automação. 2. Piscicultura. 3. Esp32. 4. Internet das Coisas (iot). I. Antayhua, Dr. Roddy Alexander Romero. II. Instituto Federal de Santa Catarina. Curso de Bacharelado Em Engenharia Elétrica. III. Título.

YAN SCHNEIDER DE QUADROS

SISTEMA DE MONITORAMENTO DOS AQUÁRIOS DO INSTITUTO FEDERAL DE SANTA CATARINA CÂMPUS-ITAJAÍ

Este trabalho foi julgado adequado para obtenção do título em Engenharia elétrica pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

Itajaí, 11 de março de 2026.

gov.br
Documento assinado digitalmente
RODDY ALEXANDER ROMERO ANTAYHUA
Data: 17/03/2026 09:30:11-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Eng. Roddy Alexander Romero Antayhua

Orientador

IFSC-Câmpus Itajaí

gov.br
Documento assinado digitalmente
SERGIO AUGUSTO BITENCOURT PETROVCIC
Data: 17/03/2026 11:08:51-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Eng. Sérgio Augusto Bitencourt Petrovcic

IFSC-Câmpus Itajaí

gov.br
Documento assinado digitalmente
EDUARDO OLIVEIRA RODRIGUES
Data: 17/03/2026 11:17:56-0300
Verifique em https://validar.iti.gov.br

Prof. Esp. Eng. Eduardo Oliveira Rodrigues

IFSC-Câmpus Itajaí

RESUMO

Este projeto apresenta o desenvolvimento de um sistema automatizado para monitoramento de parâmetros físico-químicos em aquários do laboratório de piscicultura do IFSC – Câmpus Itajaí. O objetivo central foi substituir a coleta manual por uma infraestrutura de monitoramento contínuo, mitigando erros humanos e lacunas de dados em períodos de recesso. A arquitetura baseia-se em um microcontrolador ESP32 integrado a sensores de pH, temperatura e turbidez, com suporte para futura expansão de sensores de amônia, nitrito, oxigênio dissolvido e alcalinidade. Os dados são transmitidos via protocolo *HTTP* para um servidor local, que hospeda um *backend* em *Node.js* com persistência em banco de dados PostgreSQL e uma interface web responsiva desenvolvida em *Next*.js. Um diferencial crítico do sistema é o módulo de calibração interativo, que utiliza regressões matemáticas e travas de segurança baseadas no coeficiente de determinação (R^2) para assegurar a confiabilidade das leituras. Os resultados demonstram que o sistema é funcional, escalável e capaz de garantir a continuidade do histórico de dados, viabilizando análises avançadas por meio da exportação de relatórios em formato XLSX.

Palavras-chave: Automação; Piscicultura; ESP32; *Internet das Coisas (IoT).*

ABSTRACT

This project presents the development of an automated system for monitoring physicochemical parameters in aquaria at the pisciculture laboratory of IFSC – Campus Itajaí. The primary objective was to replace manual data collection with a continuous monitoring infrastructure, mitigating human errors and data gaps during recess periods. The architecture is based on an ESP32 microcontroller integrated with pH, temperature, and turbidity sensors, featuring pre-configured logical support for the future expansion of ammonia, nitrite, dissolved oxygen, and alkalinity sensors. Data is transmitted via the HTTP protocol to a local server, which hosts a Node.js backend with PostgreSQL database persistence and a responsive web interface developed in Next.js. A critical differentiator of the system is the interactive calibration module, which employs mathematical regressions and safety locks based on the coefficient of determination (R2) to ensure the scientific reliability of the readings. The results demonstrate that the system is functional, scalable, and capable of ensuring data history continuity, enabling advanced analysis through the export of reports in XLSX format.

Keywords: Automation; Pisciculture; ESP32; Internet of Things (IoT).

LISTA DE IMAGENS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ADC	Conversor Analógico-Digital (Analog-to-Digital Converter)
API	Interface de Programação de Aplicações (Application Programming Interface)
BIOS	Sistema Básico de Entrada e Saída (Basic Input/Output System)
BNC	Conector coaxial padrão (Bayonet Neill-Concelman)
CV-RMSE	Coeficiente de Variação da Raiz do Erro Quadrático Médio
DC	Corrente Contínua (Direct Current)
ESP32	Microcontrolador Wi-Fi/Bluetooth da Espressif Systems
GET	Método de requisição HTTP para obtenção de dados
HTTP	Protocolo de Transferência de Hipertexto (Hypertext Transfer Protocol)
I2C	Protocolo de comunicação serial (Inter-Integrated Circuit)
ID	Identificador único (Identifier)
IDE	Ambiente de Desenvolvimento Integrado (Integrated Development Environment)
IFSC	Instituto Federal de Santa Catarina
IoT	Internet das Coisas (Internet of Things)
IP	Protocolo de Internet (Internet Protocol)
JSON	Notação de Objetos JavaScript (JavaScript Object Notation)
JWT	Token Web JSON (JSON Web Token)
LED	Diodo Emissor de Luz (Light Emitting Diode)
MAC	Endereço de Controle de Acesso à Mídia (Media Access Control

	Address)
MIME	Extensões Multipropósito de Correio da Internet (Multipurpose Internet Mail Extensions)
MIT	Licença de software de código aberto (Massachusetts Institute of Technology License)
MUI	Material UI — biblioteca de componentes React
NTU	Unidade Nefelométrica de Turbidez (Nephelometric Turbidity Unit)
OD	Oxigênio Dissolvido
ORM	Mapeamento Objeto-Relacional (Object-Relational Mapping)
PCB	Placa de Circuito Impresso (Printed Circuit Board)
pH	Potencial Hidrogeniônico
POST	Método de requisição HTTP para envio de dados
PUT	Método de requisição HTTP para atualização de dados
PVC	Policloreto de Vinila (Polyvinyl Chloride)
R^2	Coeficiente de Determinação (estatística de qualidade de regressão)
REST	Transferência de Estado Representacional (Representational State Transfer)
SD	Cartão de memória flash (Secure Digital)
SQL	Linguagem de Consulta Estruturada (Structured Query Language)
TCC	Trabalho de Conclusão de Curso
URL	Localizador Uniforme de Recursos (Uniform Resource Locator)
UUID	Identificador Único Universal (Universally Unique Identifier)
VCA	Tensão em Corrente Alternada

Wi-Fi	Tecnologia de rede local sem fio (Wireless Fidelity)
XLSX	Formato de arquivo de planilha eletrônica (Excel Open XML Spreadsheet)

SUMÁRIO

1 INTRODUÇÃO

A piscicultura tem apresentado um crescimento contínuo no Brasil, consolidando-se como uma alternativa promissora em relação a outras fontes de proteína animal. Segundo a Associação Brasileira de Piscicultura (2025), a produção nacional em 2024 registrou um aumento de 9,21% em comparação ao ano anterior. Corroborando esse cenário positivo, destaca-se também que o volume de exportações cresceu expressivos 102%, marca não observada desde 2021.

No cenário regional, o estado do Paraná se sobressai como a principal produtora do país, liderando o ranking nacional e Santa Catarina ocupando a quarta posição, com uma produção de 59.900 toneladas de peixes cultivados (EMBRAPA, 2025).

Segundo o Serviço Nacional de Aprendizagem Rural - SENAR (2017), a piscicultura é a atividade voltada à criação de peixes em ambientes controlados, como viveiros escavados, açudes e tanques-rede. Para o sucesso da produção, é essencial avaliar aspectos como o volume e a qualidade da água, bem como as limitações específicas de cada sistema de cultivo.

Nesse contexto, a aquicultura demanda um controle rigoroso dos parâmetros físico-químicos da água, essenciais para a manutenção do equilíbrio do ecossistema e o desenvolvimento saudável dos organismos cultivados. O monitoramento constante da qualidade da água é, portanto, imprescindível para assegurar condições adequadas ao cultivo e sustentar a biomassa presente nos viveiros (LOPES, 2018).

A qualidade da água se configura como um fator determinante para o crescimento e desenvolvimento dos peixes. De acordo com Moro et al. (2021), negligências nesse aspecto podem gerar perdas significativas e prejuízos incalculáveis aos produtores.

Diante desse panorama, o presente trabalho tem como objetivo desenvolver um sistema automatizado para o monitoramento e registro dos parâmetros da água nos aquários internos do IFSC - Campus Itajaí. A proposta visa garantir maior precisão, confiabilidade e frequência nas coletas de dados, contribuindo para a melhoria da gestão aquícola institucional.

1.1 OBJETIVOS

1.1.1 Objetivo geral

Automatizar a leitura dos parâmetros físico-químicos dos aquários do laboratório de piscicultura do IFSC-Campus Itajaí, através de um sistema de monitoramento e armazenamento das informações coletadas. Isso visa garantir a confiabilidade e a frequência dessas leituras para, com base nos dados obtidos, atuar de maneira eficaz na estabilidade do ecossistema dos aquários.

1.1.2 Objetivo específico

- Estudar a viabilidade de automação com sensores de baixo custo.
- Definir os pontos de coleta otimizando o uso de sensores.
- Escolher a plataforma de microcontroladores e a arquitetura de rede mais adequada.
- Integrar o sistema com um banco de dados para registro e análise das informações coletadas.
- Desenvolver uma página web para visualizar os dados coletados.

1.2 CONTEXTO E JUSTIFICATIVA DO PROJETO

O ponto de partida para este trabalho foi a análise do processo de monitoramento existente no laboratório de piscicultura do campus. Atualmente, a leitura dos parâmetros físico-químicos da água é realizada de forma manual, com os dados sendo transcritos para formulários pelos estudantes do curso de Recursos Pesqueiros. Este método, embora estabelecido, apresenta desafios significativos, como a suscetibilidade a erros humanos de medição e registro, a possibilidade de omissão de dados e, crucialmente, a interrupção da coleta durante fins de semana e períodos de recesso acadêmico, o que compromete a continuidade e a integridade dos dados.

2. REVISÃO DE LITERATURA

Neste capítulo, o referencial teórico aborda os conceitos fundamentais da aquicultura, os processos de medição da qualidade da água e a automatização de aquários por meio do *IoT (Internet das Coisas)*, juntamente com aplicação em literaturas atuais, sendo elementos centrais para o presente trabalho.

2.1 AQUICULTURA

A aquicultura consiste na criação de peixes em ambientes controlados, como açudes, viveiros, tanques ou lagos. Nesses sistemas, todo o processo é gerenciado, desde o crescimento inicial até a fase adulta, quando os peixes estão aptos para beneficiamento (CRISTIANO, 2024). Segundo Oliveira (2025), em ambientes controlados, vários fatores são essenciais para o desenvolvimento saudável dos peixes, incluindo temperatura da água, oxigênio dissolvido, pH, turbidez, alcalinidade, nitrito e amônia.

2.2 QUALIDADE DA ÁGUA

Segundo Cristiano (2024), a qualidade da água define o êxito ou insucesso de uma criação de peixes. Um ambiente aquático bem monitorado e gerenciado promove o metabolismo otimizado dos peixes, gerando uma produção mais eficiente e sustentável.

2.2.1 Oxigênio dissolvido

Um dos parâmetros fundamentais para indicar a qualidade da água é o nível de oxigênio disponível para os peixes, sendo a principal fonte de respiração. O baixo nível de oxigênio levará a dificuldade de respiração, causando alta mortalidade dos peixes e importantes fitoplânctons (CRISTIANO, 2024).

2.2.2 Potencial hidrogeniônico (pH)

Para assegurar uma criação eficiente de peixes, a água deve apresentar níveis de pH estáveis, evitando picos de acidez ou basicidade. Viveiros com água

excessivamente ácida ou básica demandam gestão precisa do pH para promover o desenvolvimento vigoroso dos peixes, mesmo que seja desafiador aproximá-lo do ótimo. Faixas de 7,0 a 8,3 são ideais para piscicultura, mas opera-se entre 6 e 9 sem comprometer a produção. Níveis extremos como pH < 4 ou pH > 11 causam mortalidade nos tanques (CRISTIANO, 2024).

2.2.3 Alcalinidade

Na piscicultura, a alcalinidade é essencial para regular e fixar o pH, neutralizando flutuações na água. O ideal é a alcalinidade total acima de 20 a 40 mg de Carbonato de Cálcio por Litro, para proporcionar estabilidade e evitar quedas de pH (OLIVEIRA, 2025).

2.2.4 Temperatura

Peixes regulam seu metabolismo pela temperatura ambiente, tornando seu controle essencial para o crescimento e funções fisiológicas (OLIVEIRA, 2025). Assim, a temperatura altera as propriedades físicas e químicas da água, influenciando diretamente o metabolismo dos peixes (JACOMINI et al., 2025).

2.2.5 Amônia

De acordo com Jacomini et al. (2025), a amônia é um composto nitrogenado que se apresenta dissolvida na água, originando-se principalmente da excreção dos peixes e da decomposição de alimentos não ingeridos. Valores aceitáveis ficam abaixo de 0,1 mg/L, o que previne a toxicidade aos peixes.

2.2.6 Nitrito

Os nitritos surgem como produtos intermediários na conversão de amônia em nitrato. Trata-se de uma substância tóxica, produzida pela transformação da amônia pela ação de bactérias, cuja concentração segura na água é de até 0,03 mg/L; picos ocorrem devido à ração acumulada no fundo, degradada por peixes e micróbios (Oliveira, 2025; Jacomini et al., 2025).

2.2.7 Turbidez

A turbidez mede a quantidade de partículas suspensas na água, reduzindo sua transparência e a penetração luminosa. Relacionada à clareza visual, é influenciada por sedimentos, algas ou matéria orgânica, impactando a fotossíntese do fitoplâncton e a eficiência alimentar dos peixes. (OLIVEIRA, 2025). Sua medida é realizada através do disco de Secchi, sendo recomendado a transparência na faixa de 40 a 60 cm (JACOMINI et al., 2025).

2.3 INTERNET DAS COISAS *(IoT)*

A *IoT* é compreendida como uma rede de objetos físicos incorporado a sensores, softwares e outras tecnologias com objetivo de conectar e trocar dados com outros dispositivos e sistemas pela internet. Segundo Cristiano (2024), a inclusão de tecnologia no agronegócio tem o potencial de elevar a qualidade da produção, reduzindo perdas e diminuindo custos. Além de que automação ajuda a eliminar atividades redundantes que geram estresse quando executadas por pessoas (SANTOS et al.,2019).

Para que um sistema de monitoramento inteligente opere de forma robusta, ele deve ser estruturado em camadas interdependentes. García et al. (2023), descrevem que essas soluções envolvem desde a camada de hardware (sensores e microcontrolador) até a camada de aplicação e interface com o usuário.

2.4 APLICAÇÃO NA LITERATURA

Abreu (2023) investigou o desenvolvimento de uma solução voltada ao monitoramento e controle automatizado em sistemas de aquicultura, focando na integração de sensores para a manutenção de parâmetros ideais de cultivo. O protótipo, baseado em sistemas embarcados, permitiu o acompanhamento remoto das condições da água e a automação de dispositivos de controle térmico e oxigenação. O trabalho concluiu que a implementação dessas tecnologias favorece a eficiência produtiva ao reduzir riscos associados a oscilações bruscas nas variáveis físico-químicas do ambiente aquático, destacando-se como uma alternativa de baixo custo para produtores de pequena escala.

Gabriel Albino de Oliveira (2025) descreveu o desenvolvimento de um sistema

automatizado, focado na otimização do controle de alimentação, iluminação e temperatura. Ele integra sensores a hardware e software para operação autônoma e remota, com uma interface gráfica em *Next.js* que possibilita configurações de agendamentos e monitoramento em tempo real. Testes em aquários de 15 e 80 litros confirmaram a eficácia no controle térmico e na capacidade de funcionamento offline, garantindo a continuidade de funções críticas mesmo sem internet.

Matos e Sena (2025), destacam a importância de manutenções regulares, dosagem precisa de ração e controle do ciclo de luz para manter a estabilidade do conforto, alertando que falhas humanas, como superalimentação ou trocas descontraídas de água, desequilibram o ecossistema. Seu estudo desenvolve o sistema Aquamático, com hardware embarcado no ESP32 e aplicativo em *React Native*, para monitoramento de pH, temperatura e nível de água, além de automação de iluminação, alimentação e trocas parciais de água. Os testes confirmaram a solução integrada, promovendo um ambiente mais saudável, prático e eficiente para aquaristas.

Andrade et al. (2023) descrevem a criação e validação de um sistema acessível de monitoramento inteligente para aquicultura de precisão, baseado em *IoT*, equipado com sensores submersíveis que avaliam em tempo real temperatura, pH e oxigênio dissolvido (OD) — este último previsto via rede neural artificial a partir de dados de temperatura e pH. O protótipo foi avaliado em cultivos de tilápia-do-nilo (Oreochromis niloticus) em pisciculturas do Estado do Pernambuco (PE), durante quatro meses completos. Os resultados indicaram elevada acurácia para temperatura (CV-RMSE ≤ 2,93%) e pH (CV-RMSE ≤ 16,01%), apesar de maior variabilidade nas partículas de OD (CV-RMSE de 58,01% a 81,73%), revelando seu potencial para aquicultura 4.0 com dados em nuvem, visualização remota e alertas automáticos.

Grzeszczak (2021) desenvolve um sistema de automação para mudanças baseado em *IoT*, permitindo o controle remoto e monitoramento de configurações básicas como temperatura e pH, além de eventos chave para o equilíbrio do micro ecossistema, via *smartphone*. O protótipo inclui um automatizador com unidade central que coleta dados de sensores e módulos, enviando-os a um servidor em nuvem via *Firebase* (plataforma Google) para integração com o aplicativo móvel. Os usuários podem ajustar aquecimento/resfriamento para adequar a temperatura à fauna e flora, programar filtragem/oxigenação, definir fotoperíodos para plantas e configurar horários e porções de alimentação.

O presente trabalho se diferencia dos estudos relacionados em três aspectos principais. Primeiro, pelo módulo de calibração remota com validação por coeficiente de determinação R^2, que garante a qualidade matemática do ajuste antes de ativar os coeficientes no *firmware* — mecanismo ausente em todos os trabalhos anteriores. Segundo, pela arquitetura, que permite o cadastro e o monitoramento simultâneo de múltiplos aquários a partir de uma única instância do servidor, atendendo diretamente à demanda do laboratório de piscicultura do IFSC — Câmpus Itajaí. Terceiro, pela funcionalidade de exportação de histórico em formato XLSX, que viabiliza análises externas pelos pesquisadores.

O levantamento bibliográfico evidencia que, embora o tema de monitoramento automatizado em aquicultura seja objeto de crescente interesse acadêmico, ainda há espaço para contribuições que abordem a calibração formal dos sensores, a escalabilidade do sistema para múltiplos pontos de monitoramento e a validação estatística da qualidade das leituras. O presente trabalho busca preencher essas lacunas, consolidando em uma única plataforma o monitoramento contínuo, a calibração validada e o gerenciamento centralizado de dispositivos para o contexto institucional do IFSC.

3 MATERIAIS E MÉTODOS

Este capítulo descreve os materiais utilizados e os procedimentos metodológicos adotados para o desenvolvimento e implementação do sistema de monitoramento de qualidade da água, aplicado aos aquários de peixes do Laboratório de Piscicultura do Instituto Federal de Santa Catarina – Campus Itajaí. São apresentados os componentes de hardware empregados na construção do dispositivo físico, as tecnologias de software responsáveis pelo processamento, armazenamento e visualização dos dados, bem como a estratégia metodológica definida para integração das camadas do sistema. A estrutura foi organizada de modo a detalhar desde a seleção dos sensores e microcontrolador até a arquitetura de comunicação, infraestrutura de rede, procedimentos de montagem e métodos de funcionamento.

3.1 MATERIAIS

Nesta seção, serão detalhados os componentes e tecnologias utilizados para desenvolver o sistema de monitoramento dos parâmetros. Serão explicadas as escolhas feitas para atender à demanda e garantir a confiabilidade do sistema.

Na primeira etapa, serão descritos os componentes do dispositivo físico, incluindo o microcontrolador, os sensores e os acessórios necessários para seu pleno funcionamento. Também será apresentado um levantamento de custos do dispositivo físico, que é fundamental para a relevância deste trabalho. Na segunda etapa, serão abordados o servidor e as tecnologias empregadas para sua implementação.

3.1.1 Dispositivo físico

O dispositivo físico é a primeira camada do sistema, atuando com a ponta para aquisição dos dados, sua principal função é fazer as leituras dos sensores, processar esse sinal para garantir a qualidade e transmiti-los de forma segura para o servidor *Backend*.

3.1.1.1 Microcontrolador

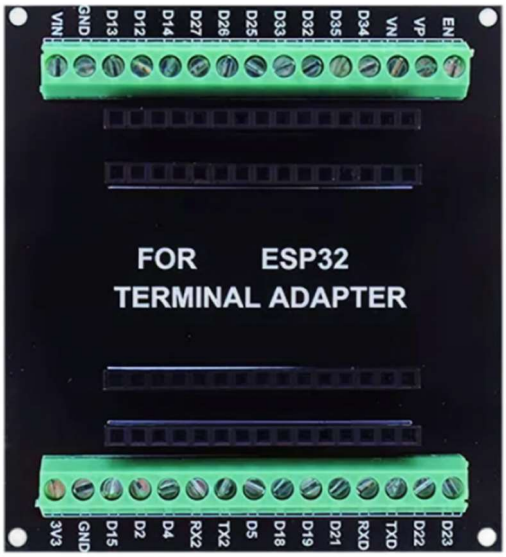
O microcontrolador selecionado para este projeto foi o ESP32 (Imagem 1). De acordo com a MakerHero (2026), a escolha desta plataforma justifica-se pelo seu elevado poder de processamento aliado ao baixo custo e à vasta utilização pela comunidade, o que assegura uma ampla gama de bibliotecas e documentação técnica. Além disso, o dispositivo destaca-se pela conectividade Wi-Fi e Bluetooth integradas, consolidando-se como uma solução robusta e versátil para o ecossistema de *IoT*. Para o desenvolvimento do *firmware*, foi utilizada a *IDE* do Arduino, aproveitando a compatibilidade da plataforma com a linguagem C/C++.

Imagem 1 - Placa ESP32 Wi-Fi/Bluetooth

Fonte: MakerHero (2026).

Além disso, para facilitar a conexão entre o ESP32 e sensores, foi utilizado um modulo de expansão compatível com a versão utilizada (Imagem 2).

Imagem 2 - Módulo de expansão

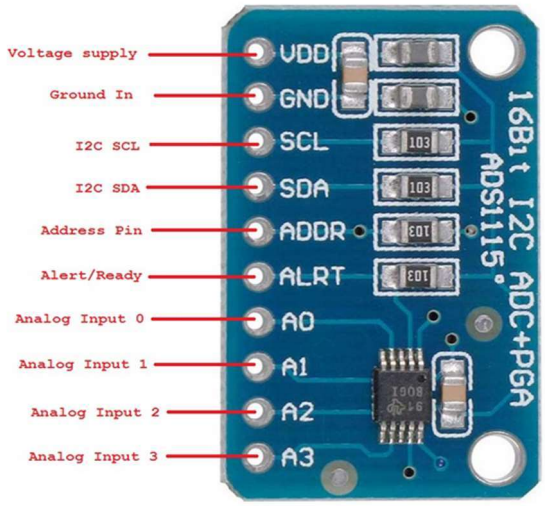


Fonte: MakerHero (2026).

3.1.1.2 Conversor analógico digital.

Para as leituras dos sensores analógicos (turbidez e pH), optou-se por não utilizar o conversor analógico digital (ADC) interno do ESP-32, que é notoriamente suscetível a ruídos e não linearidade, além de ficar limitado à tensão de operação de 3,3 V do microcontrolador (AISLAN, 2024). Com isso, foi utilizado o conversor externo de 16 bits ADS1115 (Imagem 3), que comunica com o ESP-32 via protocolo I2C. Esta decisão de projeto foi crucial para realizar a adequação dos níveis de tensão, já que os sensores analógicos operam na faixa de 0 a 5 V e as portas do ESP32 suportam o limite máximo de 3.3 V."

Imagem 3 - Modulo ADS1115



Fonte: Aislan (2024).

3.1.1.3 Sensor de pH

O monitoramento do pH é realizado pelo sensor PH-4502C (Imagem 4), composto por uma sonda com eletrodo de vidro e um módulo condicionador de sinal. O dispositivo opera em uma faixa de 0 a 14 pH, apresentando um tempo de resposta de 60 segundos (AISLAN, 2024). A principal função do módulo é amplificar o sinal da sonda e convertê-lo em uma tensão analógica de saída (0 a 5 V), garantindo a compatibilidade para a leitura pelo conversor ADS1115.

Imagem 4 - Sensor de pH-4502c

Fonte: Aislan (2024).

3.1.1.4 Sensor de turbidez

O monitoramento da transparência da água é realizado pelo sensor ST100 (Imagem 5), que utiliza o princípio da nefelometria para medir a luz espalhada por partículas suspensas. O dispositivo possui um emissor infravermelho e um fototransistor posicionados em ângulo de 180° (STRAUB, 2021). Quando a água apresenta impurezas, a luz é desviada para o receptor, gerando um sinal elétrico proporcional à concentração de sedimentos (MAKERHERO, 2021). Esse sinal é tratado por um módulo eletrônico que ajusta a tensão para uma leitura precisa através do conversor ADS1115.

Imagem 5 - Sensor de turbidez ST100

Fonte: MAKERHERO (2021).

3.1.1.5 Sensor de temperatura

Para a medição de temperatura, utilizou-se o sensor DS18B20 (Imagem 6), que integra um conversor ADC interno de 9 a 12 bits. Este componente realiza a leitura térmica por meio de um diodo de silício, processa o sinal e armazena o valor digital em um registrador interno (MAXIM INTEGRATED, 2018). O dispositivo é alimentado pela tensão de 5V da fonte e a versão aplicada possui encapsulamento à prova d'água, sendo ideal para imersão em líquidos, além de comunicação com o *ESP-32* ocorre via protocolo *1-Wire*, que permite a leitura de múltiplos sensores utilizando apenas um pino digital (MAKERHERO, 2023).

Imagem 6 - Sensor de temperatura DS18B20

Fonte: MAKERHERO (2023).

3.1.1.6 Componentes de montagem

- Placa perfurada.
- Caixa PVC.
- Cano PVC ½.
- Parafusos e Porcas.
- Prensa-cabos para vedação.
- Fonte de 5V DC.
- Plug Industrial 2p+t 16 A.
- Cabo PP 3x1,5 mm.
- Botão de impulso com led.
- Botão seletor 2 posições.

3.1.2 Custos dos Componentes

Um dos objetivos do sistema desenvolvido é oferecer monitoramento de baixo custo para análise de qualidade da água em laboratórios de piscicultura. Para quantificar esse aspecto, a Tabela 1 apresenta o levantamento de custos dos componentes eletrônicos e materiais necessários para a construção de uma unidade de monitoramento, com base em preços pesquisados em lojas brasileiras.

Quadro 1- Levantamento de custos dos componentes

N°	Componente	Descrição	Qtd.	Preço	Total
1	ESP32-WROOM-32	Microcontrolador	1	R$ 45,00	R$ 45,00
2	ADS1115	Conversor ADC	1	R$ 22,00	R$ 22,00
3	Sensor pH PH-4502C	Kit módulo + eletrodo BNC	1	R$ 75,00	R$ 75,00
4	Sensor DS18B20	Temperatura à prova d'água,	1	R$ 18,00	R$ 18,00
5	Sensor turbidez ST100	Kit módulo + sonda	1	R$ 45,00	R$ 45,00
6	Resistor 4,7 kΩ	Pull-up para barramento	1	R$ 0,10	R$ 0,10
7	Placa PCI Dupla Face 4×6 cm	Placa ilhada para soldagem	1	R$ 7,00	R$ 7,00
8	Borne Fêmea	Conector borne	1	R$ 3,00	R$ 3,00
9	Cabo de rede	Fio para interligação interna	1	R$ 2,00	R$ 2,00
10	Fonte chaveada 5V 2A	Alimentação	1	R$ 35,00	R$ 35,00
11	Caixa ABS 250×200×100 mm	Caixa plástica para proteção	1	R$ 68,00	R$ 68,00
TOTAL ESTIMADO POR UNIDADE					**R$ 320,10**

Fonte: Autor (2026).

3.1.3 Tecnologias da informação

3.1.3.1 *Backend*

O servidor foi desenvolvido em *TypeScript*, uma linguagem que adiciona tipagem estática ao *JavaScript*, permitindo a detecção de erros durante o desenvolvimento e aumentando a segurança do código (MICROSOFT, 2024). Como ambiente de execução, utilizou-se o *Node.js*, que possibilita gerenciar múltiplas requisições simultâneas de forma eficiente (NODE.JS FOUNDATION, 2024). A estruturação das rotas e a comunicação via protocolo HTTP foram viabilizadas pelo framework Express, garantindo uma arquitetura leve para o sistema (EXPRESS, 2024).

Para o armazenamento e integridade das informações coletadas pelos sensores, utilizou-se o *PostgreSQL*, um sistema de banco de dados relacional reconhecido por sua robustez (POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2024). A interface entre o servidor e o banco de dados é mediada pelo Prisma ORM,

ferramenta que abstrai consultas SQL complexas, automatiza o gerenciamento de entidades e facilita a manutenção do esquema de dados (PRISMA DATA, 2024)

3.1.3.2 *Frontend*

A camada de apresentação foi desenvolvida com o *framework Next.js 14*, baseado na biblioteca React.js, utilizando *TypeScript* para assegurar a integridade e a facilidade de manutenção do código. A interface utiliza a biblioteca *Material UI*, fundamentada nas diretrizes do *Material Design*, o que garante uma experiência de usuário responsiva e visualmente consistente (MUI, 2024).

Para atender aos requisitos funcionais de monitoramento e análise, integraram-se bibliotecas especializadas:

- Visualização e Relatórios: o *Recharts* é empregado para a geração de gráficos interativos das leituras dos sensores, enquanto a biblioteca xlsx possibilita a exportação de relatórios em formato Excel.
- Comunicação e Dados: a troca de informações com o *backend* é realizada via *Axios*, e a manipulação temporal dos dados utiliza as bibliotecas *date-fns e Day.js.*
- Segurança: a integridade das sessões é gerenciada pela biblioteca *jwt-decode*, que valida os tokens de autenticação no lado do cliente.

3.1.4 Validação Estatística da Calibração (R^2).

A conversão dos sinais elétricos brutos em valores físico-químicos reais exige uma validação estatística rigorosa para assegurar que o monitoramento seja confiável. Para este fim, o sistema utiliza o Coeficiente de Determinação, representado por R^2, como métrica principal de qualidade do ajuste entre os dados coletados e o modelo matemático aplicado.

O coeficiente R^2 indica a proporção da variabilidade dos dados que pode ser explicada pelo modelo de regressão adotado (DEVORE, 2018). Matematicamente, ele varia de 0 a 1 (ou 0% a 100%):

- R^2=1,00 (100%): indica que o modelo matemático passa exatamente por todos os pontos coletados, representando um ajuste perfeito.
- R^2=0,00 (0%): indica que o modelo não consegue explicar nenhuma variação nos dados, sendo completamente ineficaz para a medição.

Um ponto técnico crítico na calibração é a quantidade de amostras utilizadas. Em uma regressão linear (reta), o uso de apenas dois pontos sempre resultará em um R^2 de 100%, pois uma linha sempre ligará dois pontos perfeitamente, o que pode mascarar erros de leitura.

Para garantir a real acurácia do monitoramento, o sistema permite o uso de dois a quatro pontos. Esta técnica, conhecida como super amostragem, permite que o cálculo de R^2 revele se o sensor realmente mantém sua linearidade e precisão em diferentes faixas da escala. Caso a regressão com múltiplos pontos mantenha um índice elevado, confirma-se que o *hardware* está operando com alta fidelidade científica; caso o índice caia abaixo de 70%, o sistema obriga o usuário a revisar a coleta para garantir que a automação final.

3.2 MÉTODOS

3.2.1 Definição da estratégia de monitoramento

Inicialmente foi realizado um levantamento para determinar a quantidade e distribuição dos aquários no laboratório de piscicultura. O laboratório consiste em sete bancadas, cada bancada é responsável por oito aquários, sendo 5 bancadas de água doce e 2 de água salgada, resultando em 56 aquários (Imagem 7).

Imagem 7 - Aquários do IFSC- Campus Itajaí

Fonte: Autor (2026).

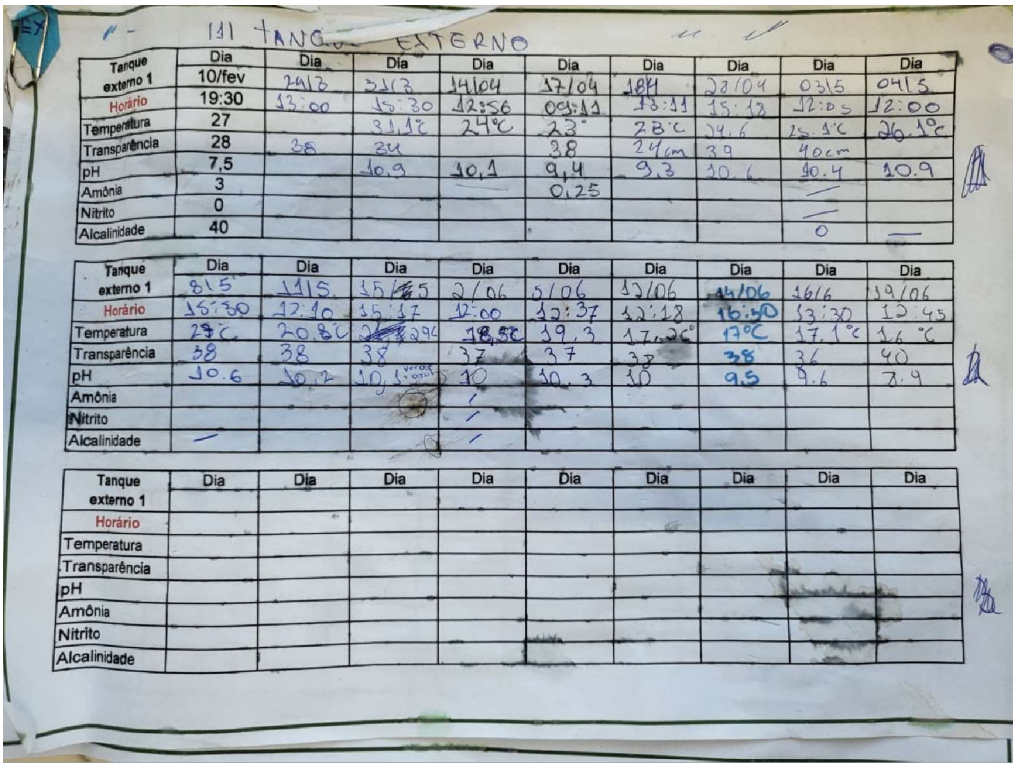
A bancada funciona como um sistema de circuito fechado. A água flui dos aquários e acumula-se em um reservatório de retorno, sendo posteriormente bombeada de volta para os aquários superiores (Imagem 8).

Imagem 8 - Bancada de teste do IFSC- Campus Itajaí

Fonte: Autor (2026).

Com isso, os alunos e técnicos do curso de recurso pesqueiros do IFSC-Campus Itajaí, são responsáveis pelo controle dos parâmetros físico-químicos que compõem a qualidade da água, para garantir o crescimento e funcionamento do ecossistema presente nos aquários, é realizado uma vez por dia a medição manual de pelo menos 1 aquário de cada bancada. Na Imagem 9 podemos verificar o processo de preenchimento de uma tabela com os dados medidos.

Imagem 9 - Tabela do Laboratório de piscicultura



Fonte: Autor (2026).

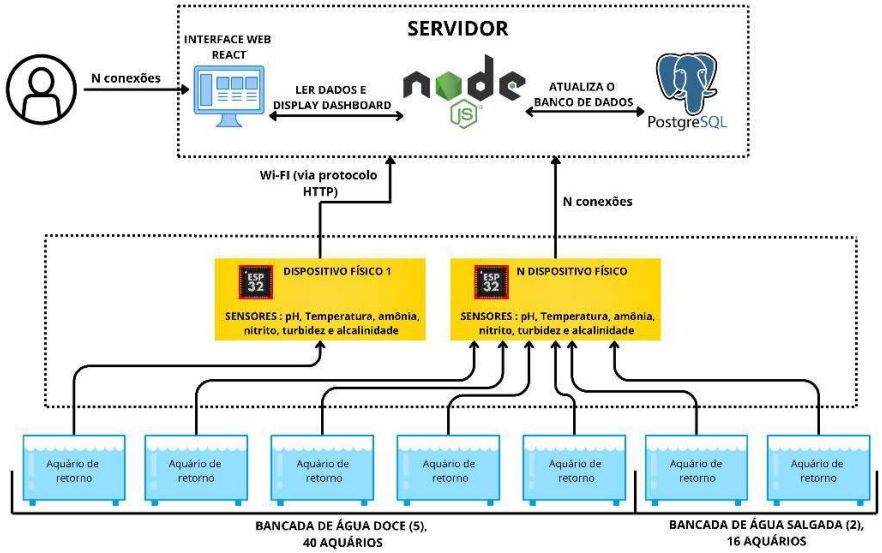
Com base no levantamento realizado no Laboratório de Piscicultura, observou-se que a rotina de monitoramento manual, embora essencial, é intermitente e suscetível a erros de preenchimento, ocorrendo apenas uma vez ao dia por bancada. A análise do sistema de circuito fechado das bancadas foi o fator determinante para a definição da arquitetura do projeto: ao monitorar o reservatório centralizado, o sistema garante uma leitura representativa de todos os oito aquários daquela unidade, garantindo a estabilidade do ecossistema com eficiência de custos.

Dessa forma, o sistema foi estruturado em três camadas interdependentes que buscam automatizar esse processo.

1. Camada de Aquisição (Dispositivo Físico): responsável pela coleta de dados em tempo real no reservatório através do ESP32 e sensores.
2. Camada de Processamento (Servidor *Backend*): responsável por receber, validar e armazenar os dados de forma segura utilizando *Node.js* e *PostgreSQL*.
3. Camada de Apresentação (*Interface Web*): responsável por exibir os gráficos e gerar relatórios automatizados, substituindo as tabelas manuais da figura 9 por um painel digital.

Na Imagem 10 podemos observar o diagrama do sistema.

Imagem 10 - Diagrama do sistema



Fonte: Autor (2026).

O sistema foi projetado com a capacidade de monitorar um conjunto abrangente de parâmetros físico-químicos essenciais para a aquicultura: temperatura, pH, amônia, nitrito, alcalinidade, turbidez e oxigénio dissolvido.

Contudo, devido a restrições orçamentais e à dificuldade de aquisição de sensores de baixo custo com a precisão e durabilidade necessárias para todos os parâmetros, o escopo prático deste trabalho focou-se na implementação completa da infraestrutura de software e apenas três sensores foram implementados:

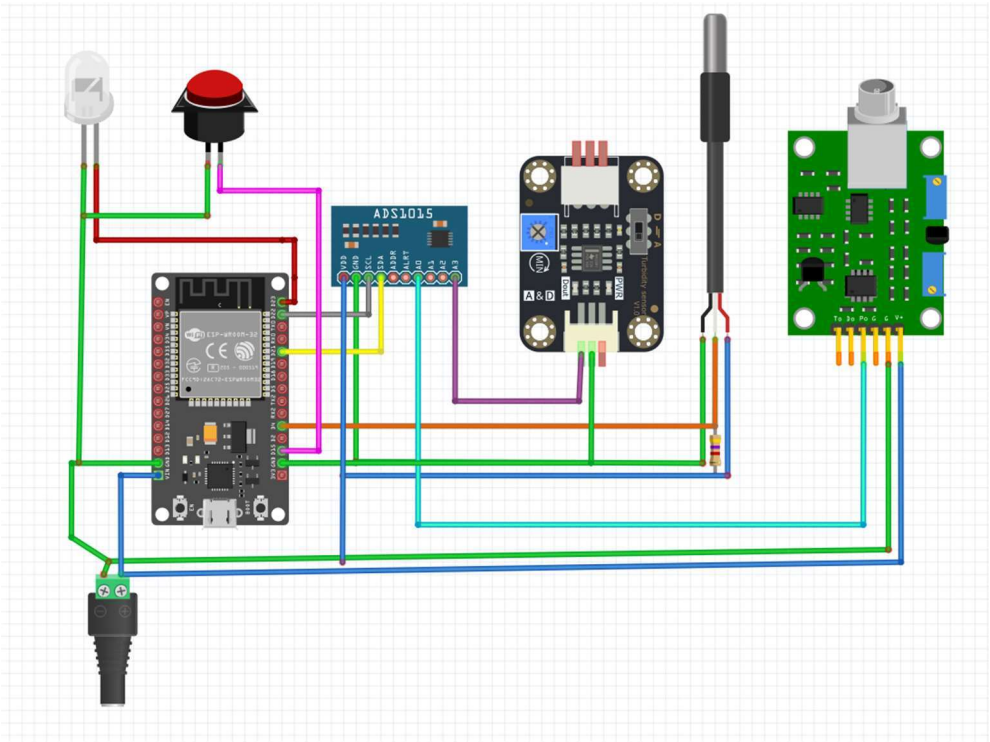
- Temperatura (Digital): essencial para o metabolismo dos peixes e para a correção de outras medições.
- pH (Analógico): indicador chave da acidez da água.
- Turbidez (Analógico): mede a transparência da água, relacionada à matéria em suspensão.

Os demais sensores, futuramente, podem ser integrados ao sistema, que já possui a arquitetura preparada para recebê-los, sem a necessidade de alterações significativas na sua estrutura base.

3.2.2 Montagem

Para facilitar e garantir o funcionamento dos sensores, todos os componentes eletrônicos foram soldados em uma placa perfurada e feita comunicação com ESP32 através de cabos. O diagrama de ligação dos sensores pode ser visto na Imagem 11.

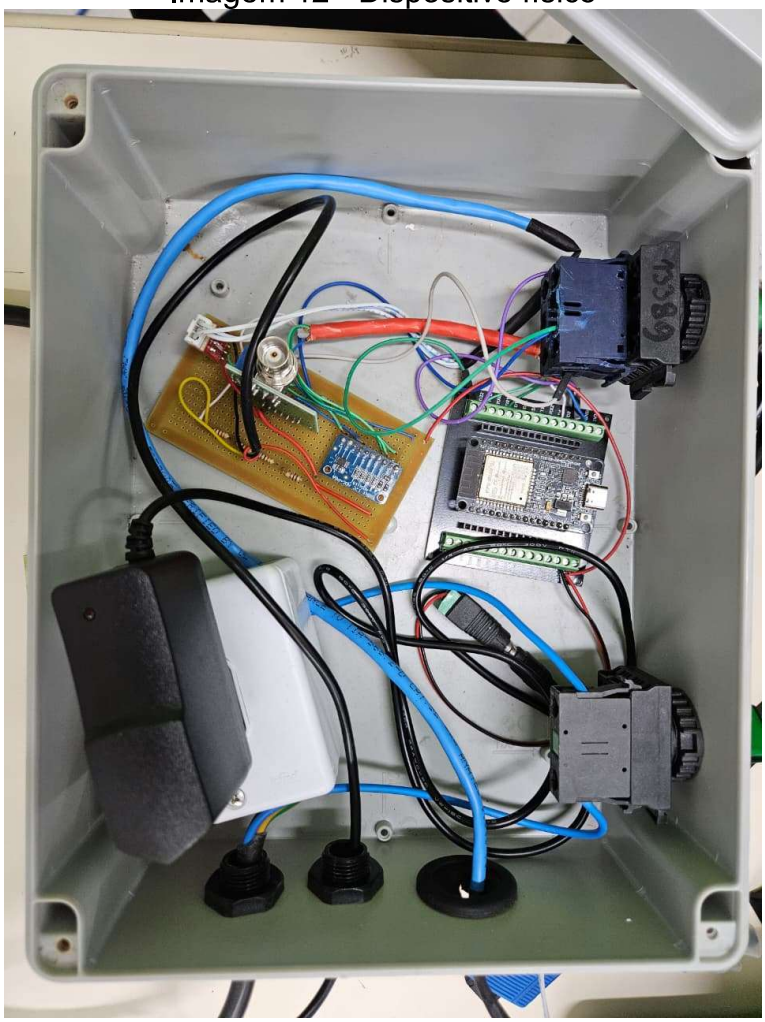
Imagem 11 - Diagrama dos sensores



Fonte: Autor (2026).

Todos os componentes eletrônicos, foram colocados dentro de uma caixa PVC, para o cabo de alimentação e saída dos sensores, foram colocados 3 prensa-cabos, para facilitar na manutenção e substituição dos sensores, além de vedar contra possíveis respingos de água. A caixa PVC com os componentes, pode ser observada na Imagem 12.

Imagem 12 - Dispositivo físico



Fonte: Autor (2026).

A alimentação do dispositivo foi realizada usando as tomadas industriais já instaladas no laboratório. A alimentação de energia chega até uma tomada dentro da caixa, nessa tomada é ligado uma fonte, para converter e alimentar o microcontrolador e todos os sensores.

3.2.3 Descrição do firmware embarcado

3.2.3.1 Configurações gerais e bibliotecas utilizadas

O *firmware* é desenvolvido para a plataforma *ESP32*, utilizando o *framework* Arduino. O bloco inicial do código carrega as bibliotecas necessárias para comunicação sem fio, requisições *HTTP*, serialização de dados e leitura dos sensores físicos. As constantes de rede e do servidor são definidas de forma centralizada,

facilitando a manutenção e eventual alteração do ambiente de implantação (Imagem 13).

Imagem 13 - Inclusão de bibliotecas e definição das configurações de rede

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiMulti.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <Adafruit_ADS1X15.h>

const char* ssid     = "JUCA";
const char* password = "qwe12345";
const char* api_host = "http://192.168.0.132:3001";
```

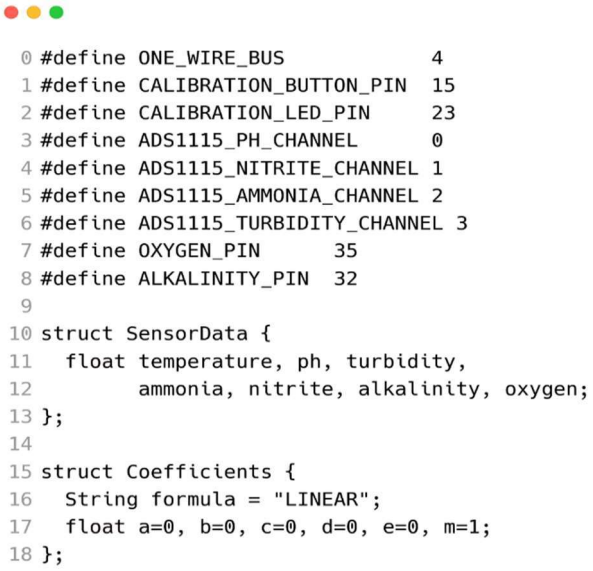
Fonte: Autor (2026).

As bibliotecas WiFi e *WiFiMulti* gerenciam a conexão sem fio com suporte a múltiplas redes. A *HTTPClient* é utilizada para requisições *HTTP* ao servidor *backend.* A *ArduinoJson* é responsável pela serialização e desserialização dos dados no formato *JSON*, padrão de comunicação adotado na *API*. As bibliotecas *DallasTemperature* e *Adafruit_ADS1X15* são específicas para os sensores de temperatura DS18B20 e o conversor analógico-digital ADS1115, respectivamente.

3.2.3.2 Definição de pinos e estrutura de dados

A seguir (Imagem 14), são definidos os pinos físicos do *ESP32* utilizados por cada sensor, além das estruturas que organizam os dados lidos e os coeficientes de calibração de cada sensor. O uso de *structs* facilita a passagem de múltiplas informações entre funções e mantém o código organizado e legível.

Imagem 14 - Definição de pinos e estruturas de dados e coeficientes



```
#define ONE_WIRE_BUS            4
#define CALIBRATION_BUTTON_PIN  15
#define CALIBRATION_LED_PIN     23
#define ADS1115_PH_CHANNEL      0
#define ADS1115_NITRITE_CHANNEL 1
#define ADS1115_AMMONIA_CHANNEL 2
#define ADS1115_TURBIDITY_CHANNEL 3
#define OXYGEN_PIN      35
#define ALKALINITY_PIN  32

struct SensorData {
  float temperature, ph, turbidity,
        ammonia, nitrite, alkalinity, oxygen;
};

struct Coefficients {
  String formula = "LINEAR";
  float a=0, b=0, c=0, d=0, e=0, m=1;
};
```

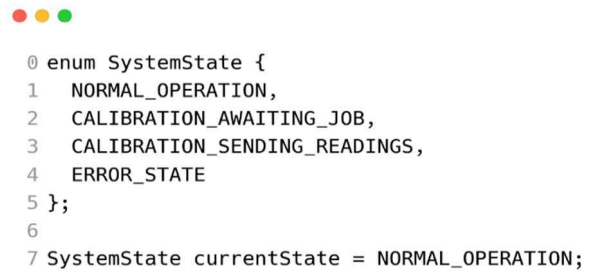
Fonte: Autor (2026).

O *struct SensorData* agrupa todas as leituras dos sensores em um único objeto, que é posteriormente serializado e enviado ao servidor. O *struct Coefficients* armazena os coeficientes matemáticos utilizados na conversão da tensão elétrica lida pelo sensor para o valor físico correspondente, suportando fórmulas lineares e polinomiais de até quarto grau.

3.2.3.3 Máquina de estados principal

O comportamento do dispositivo é governado por uma máquina de estados finita com quatro estados distintos. Essa abordagem permite que o *firmware* reaja de forma previsível a diferentes situações operacionais, como operação normal, calibração dos sensores e recuperação de falhas de comunicação (Imagem 15).

Imagem 15 - Definição da máquina de estados do sistema



```
enum SystemState {
  NORMAL_OPERATION,
  CALIBRATION_AWAITING_JOB,
  CALIBRATION_SENDING_READINGS,
  ERROR_STATE
};

SystemState currentState = NORMAL_OPERATION;
```

Fonte: Autor (2026).

O estado *NORMAL_OPERATION* corresponde ao funcionamento rotineiro, em que o dispositivo realiza leituras periódicas e as envia ao servidor. Os estados *CALIBRATION_AWAITING_JOB* e *CALIBRATION_SENDING_READINGS* gerenciam o processo de calibração remota, em que o dispositivo aguarda um comando do servidor e, ao recebê-lo, passa a enviar leituras brutas de tensão continuamente. O estado *ERROR_STATE* é ativado quando ocorre falha de comunicação, fazendo o LED piscar lentamente e tentando recuperar a conexão a cada 30 segundos.

3.2.3.4 Amostragem estável com filtragem

Para garantir leituras confiáveis, o firmware coleta 10 amostras de cada sensor, ordena os valores e descarta os extremos (2 mínimos e 2 máximos), calculando a média dos valores centrais. Essa técnica, conhecida como média aparada, é fundamentada nos princípios de filtragem por estatísticas de ordem, nos quais estimadores robustos são obtidos ao se eliminarem os valores mais extremos de um conjunto de amostras antes do cálculo da média (DINIZ; SILVA; NETTO, 2014). Sua aplicação em *firmware* reduz o impacto de ruídos elétricos e picos momentâneos no sinal, tornando as leituras mais estáveis e representativas do valor real da grandeza monitorada (Imagem 16).

Imagem 16 - Funções de amostragem com filtragem de ruído

```
#define NUM_SAMPLES 6

float getStableADCRead(int pin) {
  int r[NUM_SAMPLES];
  for(int i = 0; i < NUM_SAMPLES; i++) {
    r[i] = analogRead(pin);
    delay(10);
  }
  std::sort(r, r + NUM_SAMPLES);
  long s = 0;
  for(int i = 1; i < NUM_SAMPLES-1; i++) { s += r[i]; }
  return s / (float)(NUM_SAMPLES - 2);
}

int16_t getStableADSRead(int channel) {
  int16_t r[NUM_SAMPLES];
  for(int i = 0; i < NUM_SAMPLES; i++) {
    r[i] = ads.readADC_SingleEnded(channel);
    delay(10);
  }
  std::sort(r, r + NUM_SAMPLES);
  long s = 0;
  for(int i = 1; i < NUM_SAMPLES-1; i++) { s += r[i]; }
  return s / (float)(NUM_SAMPLES - 2);
}
```

Fonte: Autor (2026).

3.2.3.5 Conversão de tensão para grandeza física

Após a leitura da tensão bruta, o *firmware* aplica os coeficientes de calibração para converter o valor elétrico na grandeza física correspondente. A função *applyCoefficients* suporta quatro tipos de fórmulas, selecionadas conforme o comportamento de cada sensor: linear, polinomial de segundo, terceiro ou quarto grau (Imagem 17).

Imagem 17 - Função de conversão de tensão para grandeza física

```
float applyCoefficients(float voltage, Coefficients coeffs) {
  if (coeffs.formula == "LINEAR")
    return (coeffs.m * voltage) + coeffs.c;
  if (coeffs.formula == "POLYNOMIAL_2_DEGREE")
    return (coeffs.a * pow(voltage,2)) + (coeffs.b * voltage) + coeffs.c;
  if (coeffs.formula == "POLYNOMIAL_3_DEGREE")
    return (coeffs.a * pow(voltage,3)) + (coeffs.b * pow(voltage,2))
           + (coeffs.c * voltage) + coeffs.d;
  if (coeffs.formula == "POLYNOMIAL_4_DEGREE")
    return (coeffs.a * pow(voltage,4)) + (coeffs.b * pow(voltage,3))
           + (coeffs.c * pow(voltage,2)) + (coeffs.d * voltage) + coeffs.e;
  return voltage;
}
```

Fonte: Autor (2026).

3.2.3.6 Compensação de temperatura no pH e turbidez

A temperatura da água afeta a precisão das leituras, exigindo ajustes no firmware para garantir a confiabilidade dos dados:

- pH: o firmware utiliza a Lei de Nernst para aplicar uma correção de 0,003 pH/°C para cada unidade de pH distante do ponto neutro (pH 7,0). Esse fator é o padrão técnico para compensar a variação de sensibilidade do eletrodo de vidro.
- Turbidez: embora o código suporte compensação térmica, o coeficiente está definido como 0,0 NTU. Como referência, a documentação do sensor SEN0189 da DFRobot, que opera sob o mesmo princípio físico de dispersão óptica do ST100 utilizado neste trabalho, indica que o sensor produz 4,1 ± 0,3 V para água limpa na faixa de 10 °C a 50 °C, sugerindo que a variação introduzida pela temperatura já se encontra dentro da margem de incerteza inerente ao próprio hardware. Nesse contexto, a determinação de um coeficiente de compensação térmica confiável para o ST100 exigiria uma calibração empírica controlada, com amostras de turbidez conhecida em diferentes temperaturas, o que está além do escopo deste trabalho.

O código com essas implementações de ajuste de temperatura, pode ser visualizada na imagem 18.

Imagem 18 - Leitura do pH e turbidez com compensação de temperatura

```
0 float readPHEscalado(float currentTemperature) {
1   float ph = applyCoefficients(readRawPHVoltage(), phCoefficients);
2   float tempDiff = currentTemperature - 25.0;
3   return ph + (0.003 * tempDiff * (7.0 - ph));
4 }
5
6 float readTurbidityEscalado(float currentTemperature) {
7   float voltage = readRawTurbidityVoltage();
8   float tempDiff = currentTemperature - 25.0;
9   float voltageCorrected = voltage - (0.0 * tempDiff); // fator ajustável
10  float ntu = applyCoefficients(voltageCorrected, turbidityCoefficients);
11  if (ntu < 0)    ntu = 0;
12  if (ntu > 3000) ntu = 3000;
13  return ntu;
14 }
```

Fonte: Autor (2026).

3.2.3.7 Busca de configuração

Periodicamente, o dispositivo consulta o servidor para obter sua configuração atualizada, incluindo o intervalo de leitura e quais sensores estão habilitados. Essa abordagem permite alterar o comportamento do dispositivo remotamente, sem necessidade de reprogramação do *firmware*. O código com está lógica de programação, pode ser vista na (Imagem 19).

Imagem 19 - Requisição GET ao servidor para atualizar configurações

```
bool updateDeviceConfiguration() {
  HTTPClient http;
  String url = String(api_host) + "/api/devices/" + deviceId + "/config";
  http.begin(url);
  int httpCode = http.GET();

  if (httpCode == HTTP_CODE_OK) {
    String payload = http.getString();
    JsonDocument doc;
    deserializeJson(doc, payload);

    timeInterval = String(doc["timeInterval"].as<const char*>()).toFloat();

    activeSensors.temperature = doc["activeSensors"]["temperature"];
    activeSensors.ph          = doc["activeSensors"]["ph"];
    activeSensors.turbidity   = doc["activeSensors"]["turbidity"];
    // ... demais sensores
    return true;
  }
  if (httpCode == HTTP_CODE_NOT_FOUND) handleFactoryReset();
  currentState = ERROR_STATE;
  return false;
}
```

Fonte: Autor (2026).

3.2.3.8 Busca de coeficientes de calibração

Antes de cada ciclo de leitura, o *firmware* busca no servidor os coeficientes de calibração mais recentes para cada sensor ativo. Isso garante que qualquer nova calibração realizada pela plataforma web seja imediatamente aplicada ao dispositivo, sem necessidade de intervenção física. O código responsável por essa lógica pode ser visualizado na imagem 20.

Imagem 20 - Requisição GET para obtenção dos coeficientes de calibração

```
bool fetchCoefficients(String sensorType) {
  HTTPClient http;
  String url = String(api_host) + "/api/calibrations/latest"
               + "?deviceId=" + deviceId
               + "&sensor_type=" + sensorType;
  http.begin(url);
  int httpCode = http.GET();

  if (httpCode == HTTP_CODE_OK) {
    JsonDocument doc;
    deserializeJson(doc, http.getString());

    Coefficients* target = nullptr;
    if (sensorType == "PH")        target = &phCoefficients;
    else if (sensorType == "TURBIDITY") target = &turbidityCoefficients;
    // ... demais sensores

    target->formula = doc["formula"].as<String>();
    JsonObject coeffs = doc["calculated_coefficients"];
    if (coeffs.isNull()) coeffs = doc["coefficients"];

    target->a = coeffs["a"] | 0.0;
    target->b = coeffs["b"] | 0.0;
    target->c = coeffs["c"] | 0.0;
    target->m = coeffs["m"] | 1.0;
    return true;
  }
  return false;
}
```

Fonte: Autor (2026).

3.2.3.9 Envio dos dados ao servidor

Após coletar e processar todas as leituras, o *firmware* monta um objeto *JSON* contendo apenas os valores dos sensores habilitados e o envia ao servidor por meio de uma requisição *HTTP POST*. Sensores desativados são omitidos do *payload*, evitando o envio de dados inválidos ou desnecessários, conforme imagem 21.

Imagem 21 - Serialização dos dados em JSON e envio ao servidor

```
void sendSensorData(SensorData data) {
  HTTPClient http;
  String url = String(api_host) + "/api/devices/log/" + deviceId;
  http.begin(url);
  http.addHeader("Content-Type", "application/json");

  JsonDocument doc;
  if (activeSensors.temperature) doc["temperature"] = String(data.temperature, 2);
  if (activeSensors.ph)          doc["ph"]          = String(data.ph, 2);
  if (activeSensors.turbidity)   doc["turbidity"]   = String(data.turbidity, 2);
  if (activeSensors.ammonia)     doc["ammonia"]      = String(data.ammonia, 2);
  if (activeSensors.nitrite)     doc["nitrite"]      = String(data.nitrite, 2);
  if (activeSensors.alkalinity)  doc["alkalinity"]   = String(data.alkalinity, 2);
  if (activeSensors.oxygen)      doc["oxygen"]       = String(data.oxygen, 2);

  String payload;
  serializeJson(doc, payload);
  int httpCode = http.POST(payload);

  if (httpCode == HTTP_CODE_NOT_FOUND) handleFactoryReset();
  http.end();
}
```

Fonte: Autor (2026).

3.2.3.10 Provisionamento do dispositivo

O provisionamento é o processo de associar o dispositivo físico a um registro no sistema. Na primeira execução, o *firmware* verifica se há um identificador salvo na memória não volátil. Caso não haja, o dispositivo entra em loop de provisionamento: anuncia seu endereço *MAC* ao servidor e aguarda que um administrador o adote pela plataforma web. A lógica de provisionamento automático está contida imagem 22.

Imagem 22 - Rotina de provisionamento

```
void handleProvisioning() {
  String macAddress = WiFi.macAddress();

  while (true) {
    // Anuncia presença ao servidor
    HTTPClient http_ping;
    http_ping.begin(String(api_host) + "/api/devices/ping");
    http_ping.addHeader("Content-Type", "application/json");
    http_ping.POST("{\"macAddress\":\"" + macAddress + "\"}");
    http_ping.end();
    delay(5000);

    // Verifica se foi adotado
    HTTPClient http_status;
    http_status.begin(String(api_host) + "/api/provision/status/" + macAddress);
    if (http_status.GET() == HTTP_CODE_OK) {
      JsonDocument doc;
      deserializeJson(doc, http_status.getString());
      if (doc["status"] == "claimed") {
        deviceId = doc["deviceId"].as<String>();
        preferences.begin("device-config", false);
        preferences.putString("deviceId", deviceId);
        preferences.end();
        ESP.restart(); // Reinicia com ID salvo
      }
    }
    http_status.end();
    delay(5000);
  }
}
```

Fonte: Autor (2026).

3.2.3.11 Modo de calibração

O modo de calibração é ativado pelo pressionamento curto do botão físico. Nesse modo, o dispositivo consulta o servidor em busca de um trabalho de calibração pendente para um determinado sensor. Ao receber o comando, começa a enviar leituras brutas de tensão a cada 2 segundos, permitindo que o servidor colete pontos suficientes para calcular a curva de calibração (Imagem 23).

Imagem 23 - Máquina de estados da calibração

```
void handleCalibration() {
  static unsigned long lastCheck = 0;

  if (currentState == CALIBRATION_AWAITING_JOB) {
    if (millis() - lastCheck > 3000) {
      HTTPClient http;
      http.begin(String(api_host) + "/api/calibration-command/" + deviceId);
      if (http.GET() == HTTP_CODE_OK) {
        JsonDocument doc;
        deserializeJson(doc, http.getString());
        activeCalibrationId = doc["id"].as<String>();
        sensorToCalibrate   = doc["sensor_type"].as<String>();
        currentState = CALIBRATION_SENDING_READINGS;
      }
      http.end();
      lastCheck = millis();
    }
  }
  else if (currentState == CALIBRATION_SENDING_READINGS) {
    if (millis() - lastCheck > 2000) {
      float voltage = 0;
      if (sensorToCalibrate == "PH")       voltage = readRawPHVoltage();
      else if (sensorToCalibrate == "TURBIDITY") voltage = readRawTurbidityVoltage();
      // ... demais sensores

      HTTPClient http_post;
      http_post.begin(String(api_host) + "/api/calibrations/"
                      + activeCalibrationId + "/reading");
      http_post.addHeader("Content-Type", "application/json");
      http_post.POST("{\"voltage\":" + String(voltage, 4) + "}");
      http_post.end();
      lastCheck = millis();
    }
  }
}
```

Fonte: Autor (2026).

3.2.3.12 Inicialização

A função setup é executada uma única vez na inicialização do ESP32. Ela configura os pinos físicos, conecta ao Wi-Fi, verifica se o dispositivo já foi

provisionado, inicializa os periféricos de *hardware* (sensor de temperatura e conversor ADS1115) e realiza a primeira busca de configuração junto ao servidor (Imagem 24).

Imagem 24 - Função setup

```
void setup() {
  Serial.begin(115200);
  pinMode(CALIBRATION_BUTTON_PIN, INPUT_PULLUP);
  pinMode(CALIBRATION_LED_PIN, OUTPUT);

  wifiMulti.addAP(ssid, password);
  while (wifiMulti.run() != WL_CONNECTED) {
    delay(500); wifi_retries++;
    if (wifi_retries > 20) { currentState = ERROR_STATE; return; }
  }

  preferences.begin("device-config", true);
  deviceId = preferences.getString("deviceId", "");
  preferences.end();

  if (deviceId == "") handleProvisioning();

  Wire.begin();
  tempSensor.begin();
  ads.begin();
  ads.setGain(GAIN_TWOTHIRDS);

  updateDeviceConfiguration();
  digitalWrite(CALIBRATION_LED_PIN, HIGH);
}
```

Fonte: Autor (2026).

3.2.3.13 Laço principal e controle do LED

O laço principal loop é executado continuamente após o setup. Ele gerencia o estado do LED indicador, interpreta os eventos do botão físico (curto: para alternar entre calibração e operação normal; pressionar mais de 4 segundos para forçar uma leitura imediata) e despacha a execução para a função correspondente ao estado atual do sistema (Imagem 25).

Imagem 25 - Laço principal

```
void loop() {
  // LED: sólido = normal | pisca rápido = calibração | pisca lento = erro
  switch (currentState) {
    case NORMAL_OPERATION:
      digitalWrite(CALIBRATION_LED_PIN, HIGH);
      break;
    case CALIBRATION_AWAITING_JOB:
    case CALIBRATION_SENDING_READINGS:
      if (millis() - lastBlinkTime > 500) {
        lastBlinkTime = millis();
        ledState = !ledState;
        digitalWrite(CALIBRATION_LED_PIN, ledState);
      }
      break;
    case ERROR_STATE:
      if (millis() - lastBlinkTime > 1500) {
        lastBlinkTime = millis();
        ledState = !ledState;
        digitalWrite(CALIBRATION_LED_PIN, ledState);
      }
      if (millis() - lastConfigCheckTime > 30000) {
        if (updateDeviceConfiguration()) currentState = NORMAL_OPERATION;
        lastConfigCheckTime = millis();
      }
      return; // Não executa o restante no modo de erro
  }

  // Lógica do botão com debounce
  int reading = digitalRead(CALIBRATION_BUTTON_PIN);
  if (buttonState == LOW && !longPressTriggered
      && (millis() - buttonPressTime > 4000)) {
    if (currentState == NORMAL_OPERATION) handleNormalOperation(true);
    longPressTriggered = true;
  }

  // Despacho para o estado atual
  switch (currentState) {
    case NORMAL_OPERATION:             handleNormalOperation(); break;
    case CALIBRATION_AWAITING_JOB:
    case CALIBRATION_SENDING_READINGS: handleCalibration();     break;
  }
}
```

Fonte: Autor (2026).

3.2.3.14 Ciclo de operação normal

A função *handleNormalOperation* concentra toda a lógica do ciclo de coleta e envio de dados. Ela verifica se o intervalo configurado foi atingido (ou se houve solicitação de leitura forçada), busca a configuração e os coeficientes atualizados,

realiza a leitura de todos os sensores ativos e envia o resultado ao servidor. A temperatura é sempre lida primeiro, pois é utilizada na compensação das demais grandezas (Imagem 26).

Imagem 26 - Função de operação normal

```
void handleNormalOperation(bool forceRead) {
  unsigned long intervalo = (unsigned long)(timeInterval * 3600000.0);
  if (!forceRead && (millis() - lastSensorReadTime < intervalo)) return;

  updateDeviceConfiguration();

  if (activeSensors.ph)         fetchCoefficients("PH");
  if (activeSensors.turbidity)  fetchCoefficients("TURBIDITY");
  if (activeSensors.ammonia)    fetchCoefficients("AMMONIA");
  if (activeSensors.nitrite)    fetchCoefficients("NITRITE");
  if (activeSensors.oxygen)     fetchCoefficients("OXYGEN");
  if (activeSensors.alkalinity) fetchCoefficients("ALKALINITY");

  float temp = readTemperature(); // Lida primeiro para compensação

  SensorData data;
  data.temperature = activeSensors.temperature ? temp          : NAN;
  data.ph          = activeSensors.ph          ? readPHEscalado(temp)          : NAN;
  data.turbidity   = activeSensors.turbidity   ? readTurbidityEscalado(temp)   : NAN;
  data.ammonia     = activeSensors.ammonia     ? readAmmoniaEscalado()         : NAN;
  data.nitrite     = activeSensors.nitrite     ? readNitriteEscalado()         : NAN;
  data.alkalinity  = activeSensors.alkalinity  ? readAlkalinityEscalado()      : NAN;
  data.oxygen      = activeSensors.oxygen      ? readOxygenEscalado()          : NAN;

  sendSensorData(data);
  lastSensorReadTime = millis();
}
```

Fonte: Autor (2026).

3.2.4 Estrutura de dados *(JSON)*

O empacotamento dos dados é padronizado através de objetos *JSON*, um formato leve que permite serializar múltiplos parâmetros dos sensores em uma única estrutura de texto, facilitando a interpretação nativa pelo ambiente *Node.js* e a persistência no banco de dados.

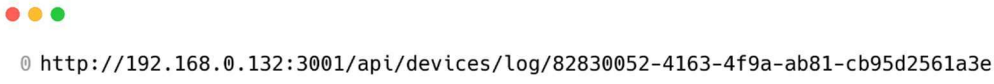
Abaixo, detalham-se os pacotes de dados planejados para a operação do sistema.

3.2.4.1 Envio de dados dos sensores

Este pacote é gerado pela função *sendSensorData()* (Imagem 28). e enviado via método *POST* para o servidor. O endereço de destino para o envio de dados é

observado na imagem 27.

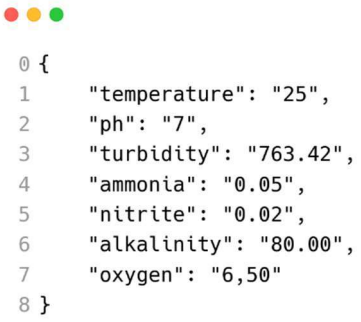
Imagem 27 - Endpoint para envio dos dados



```
http://192.168.0.132:3001/api/devices/log/82830052-4163-4f9a-ab81-cb95d2561a3e
```

Fonte: Autor (2026).

Imagem 28 - Estrutura de dados para envio dos sensores (POST)



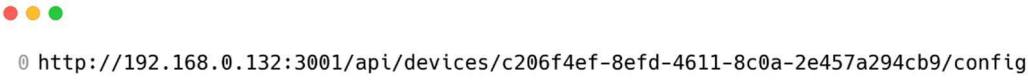
```
{
    "temperature": "25",
    "ph": "7",
    "turbidity": "763.42",
    "ammonia": "0.05",
    "nitrite": "0.02",
    "alkalinity": "80.00",
    "oxygen": "6,50"
}
```

Fonte: Autor (2026).

3.2.4.2 Sincronização de configurações

Ao iniciar ou em ciclos de recuperação, o ESP32 consome este *JSON* da imagem 30 via método *GET* para saber quais são as configurações escolhidas para o dispositivo, o exemplo do endereço pode ser visto na imagem 29:

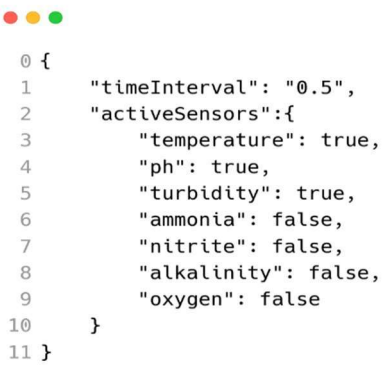
Imagem 29 - Endpoint para buscar os dados



```
http://192.168.0.132:3001/api/devices/c206f4ef-8efd-4611-8c0a-2e457a294cb9/config
```

Fonte: Autor (2026).

Imagem 30 - Estrutura de dados recebidas do servidor (GET)



```
{
    "timeInterval": "0.5",
    "activeSensors":{
        "temperature": true,
        "ph": true,
        "turbidity": true,
        "ammonia": false,
        "nitrite": false,
        "alkalinity": false,
        "oxygen": false
    }
}
```

Fonte: Autor (2026).

3.2.4.3 Coeficientes de calibração

Para converter a tensão analógica do ADS1115 em unidades físicas, o servidor envia os coeficientes resultantes da regressão matemática. O ESP32 utiliza esses valores na função *applyCoefficients()* seguindo um exemplo de fórmula que pode ser utilizada:

$$V = (a * V^2) + (b * V) + c$$

Equação 1: Equação de 2° grau genérica.

O endereço de busca dos dados pode ser visto na imagem 31. E a estrutura de dados na imagem 32.

Imagem 31 - Endpoint para buscar os dados de coeficiente

```
http://192.168.0.132:3001/api/calibrations/latest?deviceId=4e931c46...&sensor_type=TURBIDITY
```

Fonte: Autor (2026).

Imagem 32 - Estrutura de dados recebidas do servidor (GET)


```
{
  "formula": "POLYNOMIAL_2_DEGREE",
  "calculated_coefficients": {
     "a": -1120.4,
     "b": 5742.3,
     "c": -4352.9,
     "d": 0.0,
     "e": 0.0,
     "m": 1.0
   }
}
```


Fonte: Autor (2026).

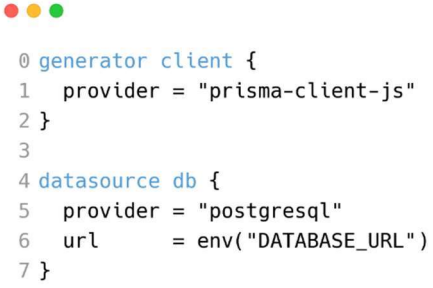
3.2.5 Modelo de dados

A persistência dos dados é gerenciada pelo Prisma ORM, que abstrai a comunicação com o banco de dados PostgreSQL por meio de um arquivo de schema declarativo. Nesse arquivo, os modelos definem as tabelas, enquanto os campos tipados estabelecem as colunas e restrições. A adoção dessa ferramenta justifica-se pela geração automática de um Prisma Client fortemente tipado em TypeScript, o que elimina erros de sintaxe em nomes de campos e permite a validação das queries ainda em tempo de compilação (PRISMA DATA, 2026).

3.2.5.1 Configuração do provedor e fonte de dados

O bloco inicial do *schema* define o gerador de cliente Prisma e a fonte de dados. O gerador instrui o Prisma a produzir o cliente *JavaScript (prisma-client-j*s), utilizado pelo *backend Node.js* para executar as *queries*. A *datasource* aponta para um banco *PostgreSQL*, cuja *URL* de conexão é lida de uma variável de ambiente, mantendo as credenciais fora do código-fonte e permitindo a troca de ambiente (desenvolvimento, homologação, produção) sem alteração de código (Imagem 33).

Imagem 33 - Configuração do gerador de cliente e da fonte de dados PostgreSQL

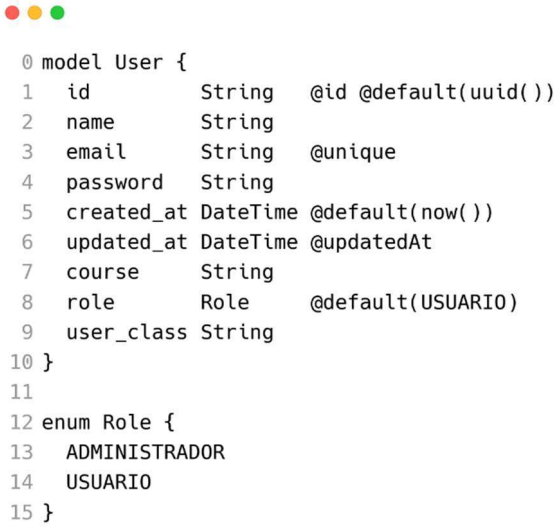


Fonte: Autor (2026).

3.2.5.2 Modelo *User* - usuários do sistema

O modelo *User* representa os usuários com acesso à plataforma de monitoramento. O campo role, baseado no *enum role*, controla o nível de acesso: administradores possuem permissões ampliadas para gerenciar dispositivos e calibrações, enquanto usuários comuns têm acesso apenas à visualização dos dados. O identificador é gerado automaticamente como *UUID*, garantindo unicidade global sem depender de sequências numéricas do banco. O campo *email* recebe a anotação *@unique*, impedindo duplicidade de cadastro. Os campos *created_at e updated_at* são preenchidos e atualizados automaticamente pelo *Prisma,* conforme imagem 34.

Imagem 34 - Modelo User



```
model User {
  id         String   @id @default(uuid())
  name       String
  email      String   @unique
  password   String
  created_at DateTime @default(now())
  updated_at DateTime @updatedAt
  course     String
  role       Role     @default(USUARIO)
  user_class String
}

enum Role {
  ADMINISTRADOR
  USUARIO
}
```

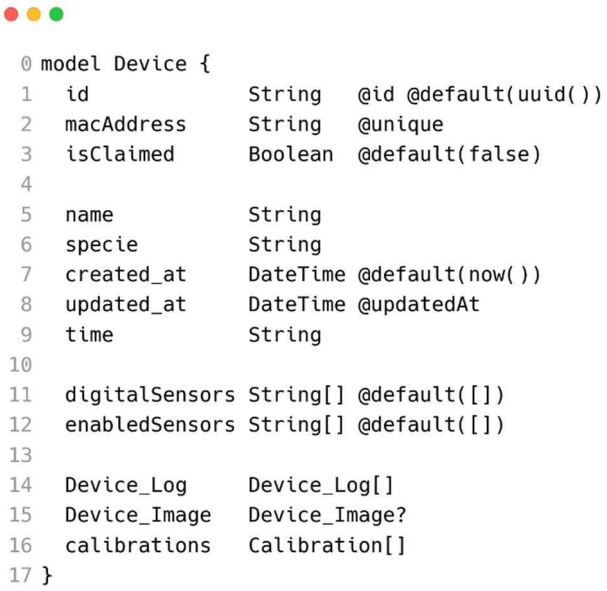
Fonte: Autor (2026).

3.2.5.3 Modelo *Device* - dispositivos de monitoramento

O modelo *Device* representa cada microcontrolador ESP32 cadastrado no sistema. O campo macAddress é único por definição de hardware e serve como identificador físico durante o processo de provisionamento. O campo *isClaimed* indica se o dispositivo já foi associado a um registro pelo administrador. Os *arrays digitalSensors* e *enabledSensors* armazenam, respectivamente, quais sensores estão fisicamente conectados ao dispositivo e quais estão habilitados para envio de dados, permitindo controle granular por sensor sem necessidade de tabelas auxiliares.

O campo *specie* registra a espécie aquática monitorada no tanque associado, contextualizando os parâmetros ideais de qualidade da água para cada caso. Os três relacionamentos declarados *(Device_Log, Device_Image e calibrations)* conectam o dispositivo a seus registros históricos, imagem de identificação e histórico de calibrações, respectivamente. A estrutura do modelo *device* pode ser observada na imagem 35.

Imagem 35 - Modelo Device



```
model Device {
  id             String   @id @default(uuid())
  macAddress     String   @unique
  isClaimed      Boolean  @default(false)

  name           String
  specie         String
  created_at     DateTime @default(now())
  updated_at     DateTime @updatedAt
  time           String

  digitalSensors String[] @default([])
  enabledSensors String[] @default([])

  Device_Log     Device_Log[]
  Device_Image   Device_Image?
  calibrations   Calibration[]
}
```

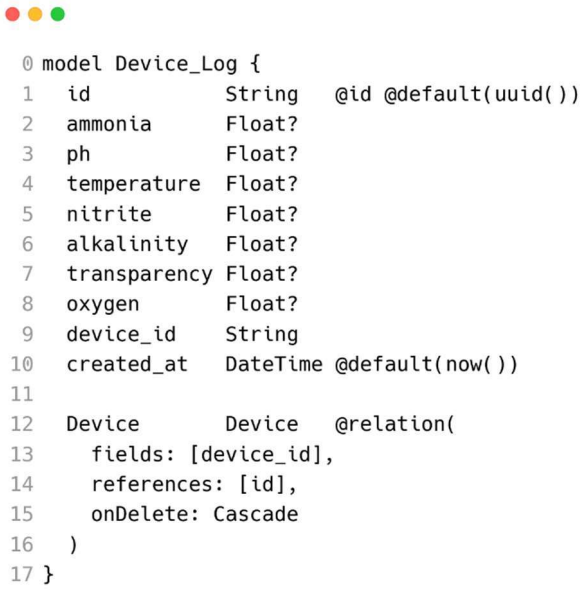
Fonte: Autor (2026).

3.2.5.4 Modelo *Device_Log* — histórico de leituras

O modelo *Device_Log* é a tabela central de dados do sistema. Cada registro representa uma leitura realizada pelo dispositivo em um determinado momento, armazenando os valores de todos os parâmetros físico-químicos coletados. Uma decisão de projeto fundamental nesse modelo é que todos os campos de sensor são opcionais *(Float?):* como dispositivos diferentes podem ter conjuntos distintos de sensores ativos, a ausência de um valor indica apenas que aquele sensor estava desabilitado no momento da coleta, sem comprometer a integridade do registro nem forçar valores padrão que distorceriam o histórico de monitoramento.

A anotação *onDelete: Cascade* garante que, ao remover um dispositivo do sistema, todos os seus logs históricos sejam excluídos automaticamente, mantendo a consistência referencial do banco de dados sem necessidade de lógica adicional na camada de aplicação. A estrutura do modelo *device_log* pode ser observada na imagem 36.

Imagem 36 - Modelo Device_Log



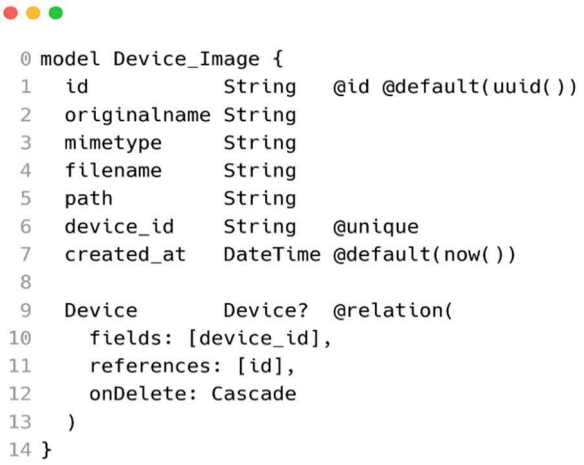
```
model Device_Log {
  id           String   @id @default(uuid())
  ammonia      Float?
  ph           Float?
  temperature  Float?
  nitrite      Float?
  alkalinity   Float?
  transparency Float?
  oxygen       Float?
  device_id    String
  created_at   DateTime @default(now())

  Device       Device   @relation(
    fields: [device_id],
    references: [id],
    onDelete: Cascade
  )
}
```

Fonte: Autor (2026).

3.2.5.5 Modelo *Device_Image* — imagem do dispositivo

O modelo *Device_Image* gerencia as fotos que identificam cada dispositivo na tela. Para manter o sistema rápido, o banco de dados salva apenas as informações sobre a imagem (como o nome e o tipo do arquivo), enquanto a foto real fica guardada em uma pasta separada no servidor. Graças a uma regra de unicidade, o sistema garante que cada dispositivo tenha apenas uma foto vinculada. A estrutura do modelo *device_Image* pode ser observada na imagem 37.

Imagem 37 - Modelo Device_Image



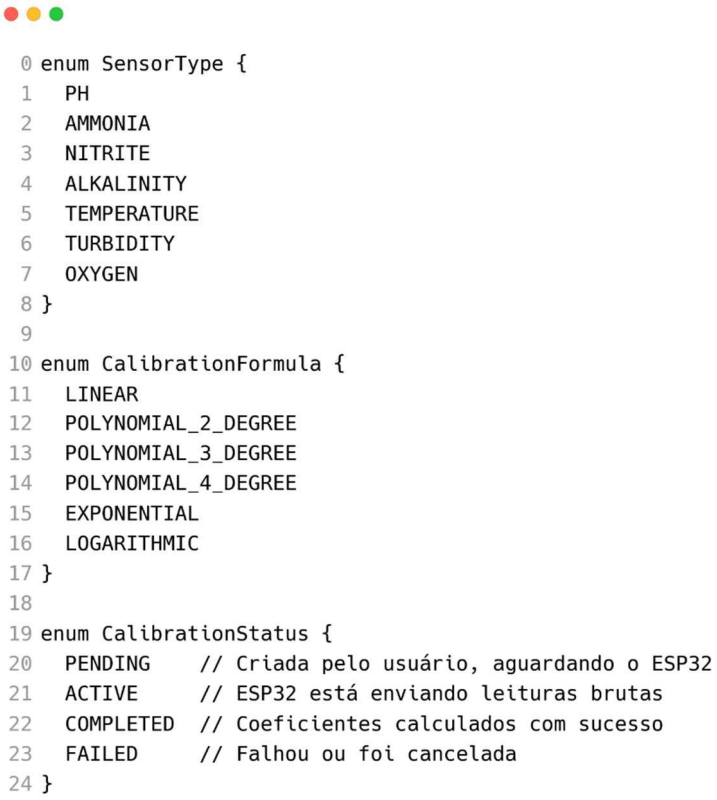
```
model Device_Image {
  id           String   @id @default(uuid())
  originalname String
  mimetype     String
  filename     String
  path         String
  device_id    String   @unique
  created_at   DateTime @default(now())

  Device       Device?  @relation(
    fields: [device_id],
    references: [id],
    onDelete: Cascade
  )
}
```

Fonte: Autor (2026).

3.2.5.6 Enumeradores do sistema de calibração

Três enumeradores estruturam o sistema de calibração, restringindo os valores aceitos nos campos correspondentes. O *SensorType* lista os sete parâmetros que podem ser calibrados. O *CalibrationFormula* define os modelos matemáticos suportados para a regressão entre tensão e grandeza física. O *CalibrationStatus* representa o ciclo de vida de uma calibração: nasce como *PENDING*, torna-se *ACTIVE* quando o ESP32 inicia o envio de leituras brutas e transita para *COMPLETED* ou *FAILED* ao término do processo, refletindo diretamente a máquina de estados implementada no *firmware* (Imagem 38).

Imagem 38 - Enumeradores que definem os tipos de sensor, fórmulas e estados



```
enum SensorType {
  PH
  AMMONIA
  NITRITE
  ALKALINITY
  TEMPERATURE
  TURBIDITY
  OXYGEN
}

enum CalibrationFormula {
  LINEAR
  POLYNOMIAL_2_DEGREE
  POLYNOMIAL_3_DEGREE
  POLYNOMIAL_4_DEGREE
  EXPONENTIAL
  LOGARITHMIC
}

enum CalibrationStatus {
  PENDING    // Criada pelo usuário, aguardando o ESP32
  ACTIVE     // ESP32 está enviando leituras brutas
  COMPLETED  // Coeficientes calculados com sucesso
  FAILED     // Falhou ou foi cancelada
}
```

Fonte: Autor (2026).

3.2.5.7 Modelo *Calibration* — processo de calibração

O modelo *Calibration* centraliza todo o ciclo de vida de uma calibração de sensor. Ele registra tanto os parâmetros de entrada definidos pelo usuário quanto os resultados produzidos pelo algoritmo de regressão ao final do processo. O uso do tipo

JSON para os campos *collected_points e calculated_coefficients* oferece flexibilidade para armazenar estruturas de dados variáveis, como *arrays* de pares (tensão, referência) e conjuntos de coeficientes de diferentes graus, sem necessidade de tabelas auxiliares.

O campo *points_to_collect* define quantos pares serão necessários para o cálculo da regressão. O campo *collected_points* acumula esses pares em formato JSON à medida que o ESP32 os envia. Ao atingir o número definido, o servidor executa a regressão com a fórmula especificada em *formula_to_apply* e persiste os coeficientes resultantes em *calculated_coefficients*, juntamente com o coeficiente de determinação R^2 em *r_squared*, que quantifica a qualidade do ajuste da curva aos dados coletados. A estrutura do modelo *Calibration* pode ser observada na imagem 39.

Imagem 39 - Modelo Calibration

```
model Calibration {
  id        String @id @default(uuid())
  device_id String
  Device    Device @relation(
    fields: [device_id],
    references: [id],
    onDelete: Cascade
  )

  isActive             Boolean           @default(false)
  status               CalibrationStatus @default(PENDING)
  sensor_type          SensorType
  formula_to_apply     CalibrationFormula
  points_to_collect    Int

  // Dados coletados durante o processo
  collected_points     Json?   // Pares [referência, tensão] de cada ponto
  last_voltage_reading Float?  // Última tensão enviada pelo ESP32

  // Resultados gerados pelo servidor
  calculated_coefficients Json? // Coeficientes {a, b, c, d, e, m}
  r_squared               Float?

  created_at DateTime @default(now())
  updated_at DateTime @default(now())
}
```

Fonte: Autor (2026).

A visão geral dos modelos do banco de dados pode ser observada na tabela 2.

Quadro 2 - Visão geral dos modelos do banco de dados e suas responsabilidades.

Modelo*(prisma)*	Tabela no Banco	Responsabilidade Principal
User	*User*	Usuários do sistema com controle de acesso por papel (Role)
Device	*Device*	Dispositivos ESP32 com sensores configuráveis e estado de provisionamento
Device_Log	*Device_Log*	Histórico de telemetria — uma linha por leitura realizada pelo dispositivo
Device_Image	*Device_Image*	Metadados da imagem de identificação visual do dispositivo
Calibration	*Calibration*	Ciclo completo de calibração: parâmetros, pontos coletados e coeficientes calculados

Fonte: Autor (2026).

3.2.6 Arquitetura do *Backend*

O servidor *backend* foi estruturado seguindo o padrão arquitetural *Controller-Service*, amplamente adotado em aplicações *Node.js*, por promover a separação de responsabilidades e facilitar a manutenção do código (NODE.JS FOUNDATION, 2024). Os *Controllers* são responsáveis exclusivamente por receber as requisições *HTTP*, validar os parâmetros de entrada e devolver as respostas ao solicitante. Toda a lógica de negócio, regras de provisionamento, ciclo de calibração e registro de telemetria é encapsulada nos *Services*, que também centralizam a comunicação com o banco de dados por meio do *Prisma ORM* (PRISMA DATA, 2024). Essa divisão permite que cada camada seja testada e modificada de forma isolada, sem impacto nas demais.

Embora o sistema conte com quatro serviços para sua operação, a lógica de negócio central e as inovações deste projeto concentram-se em dois eixos principais: o DeviceService e o CalibrationService. Os serviços de UserService e LoginService, embora essenciais para a segurança, desempenham funções de suporte voltadas à gestão de acesso e autenticação.

A escolha por detalhar os dois serviços principais justifica-se por eles serem os responsáveis pela inteligência do sistema: enquanto o DeviceService gerencia a comunicação e o estado dos dispositivos ESP32, o CalibrationService orquestra os cálculos matemáticos e os ajustes necessários para a precisão dos sensores. A visão geral das principais funções do backend pode ser observada na tabela 3.

Quadro 3 - Principais funções dos Services do backend e suas responsabilidades.

Serviço	Função	Responsabilidade
DeviceService	*findOrCreateUnclaimedDevice*	Registra o dispositivo pelo MAC na primeira conexão
DeviceService	*claimDevice*	Vincula o dispositivo após adoção pelo administrador
DeviceService	*getProvisioningStatus*	Informa ao firmware se o dispositivo já foi adotado
DeviceService	*getDeviceConfig*	Entrega ao firmware o intervalo de leitura e sensores ativos
DeviceService	*createLog*	Persiste as leituras dos sensores no banco de dados
DeviceService	*getChartData*	Retorna o histórico de 30 dias de um sensor para os gráficos
CalibrationService	*createPendingCalibration*	Cria nova sessão de calibração com transação atômica
CalibrationService	*findPendingJobForDevice*	Despacha o trabalho de calibração ao firmware
CalibrationService	*saveVoltageReading*	Armazena a tensão bruta enviada pelo firmware
CalibrationService	*addReferencePoint*	Associa a tensão ao valor de referência confirmado pelo usuário
CalibrationService	*finalizeCalibration*	Executa a regressão e persiste os coeficientes calculados
CalibrationService	*getLatestCoefficients*	Entrega ao firmware os coeficientes de calibração vigentes

Fonte: Autor (2026).

3.2.6.1 *DeviceService* — gerenciamento do ciclo de vida dos dispositivos

O *DeviceService* centraliza toda a lógica de negócio relacionada aos dispositivos ESP32, cobrindo o ciclo completo desde o primeiro contato com a rede até o registro contínuo de telemetria.

3.2.6.1.1 Provisionamento

Quando o ESP32 é energizado pela primeira vez, anuncia seu endereço *MAC* ao servidor via *HTTP POST*. A função *findOrCreateUnclaimedDevice* utiliza a operação *upsert* do *Prisma* para localizar um registro existente com aquele *MAC* ou criar um, evitando duplicidades mesmo que o dispositivo realize múltiplos anúncios. O registro é criado com *isClaimed* igual a false, sinalizando que aguarda adoção. Quando o administrador adota o dispositivo, a função *claimDevice* preenche os campos de identificação e altera o *flag* para *TRUE*, concluindo o vínculo entre o hardware físico e seu registro lógico no banco. A programação da lógica do provisionamento pode ser vista na imagem 40.

Imagem 40 - Funções de provisionamento

```
static async findOrCreateUnclaimedDevice(macAddress: string) {
    return await prisma.device.upsert({
        where:  { macAddress },
        update: { updated_at: new Date() },
        create: {
            macAddress,
            name:      `Novo Dispositivo (${macAddress})`,
            specie:    "Não definida",
            isClaimed: false,
            time:      "30"
        }
    });
}

static async claimDevice(macAddress: string,
    details: { name: string, specie: string, time: string }) {
    return await prisma.device.update({
        where: { macAddress, isClaimed: false },
        data:  {
            name:      details.name,
            specie:    details.specie,
            time:      String(details.time),
            isClaimed: true
        }
    });
}
```

Fonte: Autor (2026).

3.2.6.1.2 Consulta periódica

Após anunciar seu *MAC*, o *firmware* do ESP32 consulta periodicamente o servidor para saber se já foi adotado. A função *getProvisioningStatus* verifica o registro no banco e retorna um objeto de estado com três possibilidades: *not_found*, quando o MAC ainda não chegou ao servidor; *waiting_claim*, quando o dispositivo existe, mas ainda não foi adotado e *claimed*, quando o vínculo está completo, neste último caso, o *deviceId* gerado pelo banco é incluído na resposta para que o firmware o salve em memória não volátil. A lógica pode ser observada na imagem 41.

Imagem 41 - Função que entrega ao firmware se o dispositivo já foi adotado

```
0 static async getProvisioningStatus(macAddress: string) {
1     const device = await prisma.device.findUnique({
2         where: { macAddress }
3     });
4
5     if (!device)          return { status: 'not_found' };
6     if (device.isClaimed) return { status: 'claimed', deviceId: device.id };
7                           return { status: 'waiting_claim' };
8 }
```

Fonte: Autor (2026).

3.2.6.1.3 Configuração remota

A cada ciclo de operação, o *firmware* consulta o servidor para obter sua configuração atualizada. A função *getDeviceConfig* recupera o intervalo de leitura e a lista de sensores habilitados, transformando o *array* de identificadores em um mapa de booleanos consumido pelo *firmware*. Essa abordagem permite que qualquer alteração feita na *interface web*, como habilitar um sensor ou alterar a frequência de leitura, seja refletida no comportamento do *hardware* sem qualquer reprogramação física do dispositivo. A lógica pode ser observada na imagem 42.

Imagem 42 - Função que entrega o intervalo de leitura e os sensores ativos

```
static async getDeviceConfig(deviceId: string) {
    const device = await prisma.device.findUnique({
        where:  { id: deviceId },
        select: { time: true, enabledSensors: true }
    });

    if (!device) throw new Error("Dispositivo não encontrado.");

    const allSensorKeys = [
        "temperature","ph","turbidity",
        "ammonia","nitrite","alkalinity","oxygen"
    ];

    const activeSensors: { [key: string]: boolean } = {};
    allSensorKeys.forEach(key => {
        activeSensors[key] = device.enabledSensors.includes(key.toUpperCase());
    });

    return { timeInterval: device.time, activeSensors };
}
```

Fonte: Autor (2026).

3.2.6.1.4 Registro de telemetria

Quando o ESP32 envia uma leitura via *HTTP POST*, a função *createLog* converte cada campo de *string* para *Float* antes de persistir o registro. Campos ausentes no *payload* são gravados como *null*, decisão consistente com o design do modelo *Device_Log,* preservando a estrutura do registro sem forçar valores padrão que poderiam distorcer o histórico de monitoramento. A lógica que persiste a leitura dos sensores pode ser observada na imagem 43.

Imagem 43 - Função que persiste a leitura dos sensores

```
static async createLog(device_id: string, deviceLog: any) {
    const dataToCreate = {
        device_id,
        ammonia:      deviceLog.ammonia      ? parseFloat(deviceLog.ammonia)      : null,
        ph:           deviceLog.ph           ? parseFloat(deviceLog.ph)           : null,
        temperature:  deviceLog.temperature  ? parseFloat(deviceLog.temperature)  : null,
        nitrite:      deviceLog.nitrite      ? parseFloat(deviceLog.nitrite)      : null,
        alkalinity:   deviceLog.alkalinity   ? parseFloat(deviceLog.alkalinity)   : null,
        transparency: deviceLog.turbidity    ? parseFloat(deviceLog.turbidity)    : null,
        oxygen:       deviceLog.oxygen       ? parseFloat(deviceLog.oxygen)       : null,
    };

    return await prisma.device_Log.create({ data: dataToCreate });
}
```

Fonte: Autor (2026).

3.2.6.2 *CalibrationService* - processo de calibração

O *CalibrationService* implementa toda a lógica do ciclo de calibração dos sensores. O processo é colaborativo entre *firmware* e operador: o ESP32 envia leituras brutas de tensão enquanto o administrador, pela *interface web*, associa cada tensão ao valor de referência medido com solução padrão. Ao final, o servidor executa a regressão matemática e persiste os coeficientes calculados.

3.2.6.2.1 Criação da sessão

A função *createPendingCalibration* executa uma transação atômica que, antes de criar a sessão, cancela automaticamente qualquer calibração anterior em estado *PENDING* ou *ACTIVE* para o mesmo dispositivo, garantindo que exista sempre apenas uma sessão ativa por vez e evitando inconsistências em acessos simultâneos. A lógica pode ser observada na imagem 44.

Imagem 44 - Criação de sessão de calibração

```
static async createPendingCalibration(data: ICreateCalibration) {
    return await prisma.$transaction(async (tx) => {

        // Cancela sessões anteriores em aberto
        await tx.calibration.updateMany({
            where: {
                device_id: data.device_id,
                status: { in: ['PENDING', 'ACTIVE'] }
            },
            data: { status: 'FAILED' }
        });

        // Cria a nova sessão
        return await tx.calibration.create({
            data: { ...data, status: 'PENDING' }
        });
    });
}
```

Fonte: Autor (2026).

3.2.6.2.2 Coleta de pontos em duas etapas

A coleta de cada ponto ocorre em duas etapas. Na primeira, o *firmware* envia continuamente a tensão bruta; *saveVoltageReading* sobrescreve o campo *last_voltage_reading* com o valor mais recente. Na segunda, quando o operador confirma o valor de referência pela *interface web*, *addReferencePoint* lê a última tensão salva, cria o par (tensão, referência) e o acrescenta ao *array collected_points,* construindo progressivamente o conjunto de dados para a regressão. A lógica pode ser observada na imagem 45.

Imagem 45 - Coleta de pontos em duas etapas

```
// Etapa 1 — Firmware envia a tensão bruta lida pelo sensor
static async saveVoltageReading(calibrationId: string, voltage: number) {
    return await prisma.calibration.update({
        where: { id: calibrationId },
        data:  { last_voltage_reading: voltage }
    });
}

// Etapa 2 — Usuário confirma o valor de referência pela interface web
static async addReferencePoint(calibrationId: string, referenceValue: number) {
    const calibration = await prisma.calibration.findUnique({
        where: { id: calibrationId }
    });

    if (!calibration?.last_voltage_reading)
        throw new Error("Nenhuma leitura de tensão encontrada para associar.");

    const existingPoints = (calibration.collected_points as any[]) || [];
    const newPoint       = {
        voltage:   calibration.last_voltage_reading,
        reference: referenceValue
    };

    return await prisma.calibration.update({
        where: { id: calibrationId },
        data:  { collected_points: [...existingPoints, newPoint] }
    });
}
```

Fonte: Autor (2026).

3.2.6.2.3 Cálculo dos coeficientes

Com todos os pontos coletados, *finalizeCalibration* recupera os pares (tensão, referência), válida a quantidade mínima exigida pela fórmula escolhida e executa a regressão matemática via biblioteca *regression.js*. Os coeficientes são mapeados para os campos nomeados esperados pelo firmware. Uma transação atômica desativa a calibração anterior do mesmo sensor no dispositivo e persiste os novos coeficientes com status *COMPLETED* e *isActive* igual a *TRUE*, tornando-os imediatamente disponíveis para consulta pelo *firmware*. A lógica pode ser observada na imagem 46.

Imagem 46 - Validação dos pontos, regressão e dos coeficientes

```
static async finalizeCalibration(calibrationId: string) {
    const calibration = await prisma.calibration.findUnique({
        where: { id: calibrationId }
    });

    const dataPoints: DataPoint[] =
        (calibration.collected_points as any[]).map(p => [p.voltage, p.reference]);

    // Valida o mínimo de pontos por fórmula
    const minPoints = { LINEAR: 2, POLYNOMIAL_2_DEGREE: 3,
                        POLYNOMIAL_3_DEGREE: 4, POLYNOMIAL_4_DEGREE: 5 };
    if (dataPoints.length < minPoints[calibration.formula_to_apply])
        throw new Error("Pontos insuficientes para a fórmula selecionada.");

    // Executa a regressão
    let result, finalCoefficients = {};
    switch (calibration.formula_to_apply) {
        case 'LINEAR':
            result = regression.linear(dataPoints, { precision: 10 });
            finalCoefficients = { m: result.equation[0], c: result.equation[1] };
            break;
        case 'POLYNOMIAL_2_DEGREE':
            result = regression.polynomial(dataPoints, { order: 2, precision: 10 });
            finalCoefficients = { a: result.equation[2], b: result.equation[1],
                                  c: result.equation[0] };
            break;
        // ... graus 3 e 4 seguem o mesmo padrão
    }

    // Transação: desativa calibração anterior e salva a nova
    return await prisma.$transaction(async (tx) => {
        await tx.calibration.updateMany({
            where: { device_id: calibration.device_id,
                     sensor_type: calibration.sensor_type, isActive: true },
            data:  { isActive: false }
        });

        return await tx.calibration.update({
            where: { id: calibrationId },
            data:  { status: 'COMPLETED', isActive: true,
                     calculated_coefficients: finalCoefficients,
                     r_squared: result.r2 }
        });
    });
}
```

Fonte: Autor (2026).

3.2.6.3 Biblioteca *regression.js*

Para a execução dos cálculos de regressão matemática no servidor, foi utilizada a biblioteca *regression.js*, uma dependência de código aberto para o ecossistema *Node.js* que implementa os principais métodos de ajuste de curvas utilizados em instrumentação e análise de dados. A biblioteca é distribuída sob a licença MIT e amplamente empregada em aplicações científicas e industriais que

demandam análise estatística no servidor (REGRESSION.JS, 2020).

A escolha da regression.js justifica-se por três razões principais. Primeiro, ela oferece suporte nativo a regressões lineares e polinomiais de múltiplos graus, cobrindo todos os modelos matemáticos previstos no módulo de calibração do sistema. Segundo, a biblioteca calcula automaticamente o coeficiente de determinação R^2, principal indicador de qualidade utilizado pelo sistema para aceitar ou rejeitar uma calibração. Terceiro, sua *API* é simples e diretamente integrável ao ambiente *TypeScript/Node.js* sem dependências adicionais.

Os principais métodos empregados no *CalibrationService* são *regression.linear()* para sensores com comportamento linear, como o sensor de pH PH-4502C, e *regression.polynomial()* com ordem configurável para sensores com curva de resposta não linear, como o sensor de turbidez ST100. Ambos recebem como entrada um *array* de pares [tensão, referência] e retornam um objeto com o *array equation*, contendo os coeficientes ordenados e o valor R^2.

Um aspecto técnico relevante é a ordem dos coeficientes retornados. Para regressões polinomiais, o *array equation* é ordenado do menor para o maior grau: índice 0 corresponde ao coeficiente constante c, índice 1 ao coeficiente linear b e índice 2 ao coeficiente quadrático a. O *CalibrationService* realiza o mapeamento explícito desses índices para os campos nomeados esperados pelo firmware, garantindo que a função *applyCoefficients* no ESP32 aplique os valores na ordem correta. Na tabela 4 apresenta o resumo das regressões utilizadas.

Quadro 4 - Métodos da biblioteca regression.js utilizados no CalibrationService

Método	Aplicação no Sistema	Retorno Principal
regression.linear(points)	Calibração do sensor de pH (PH-4502C)	*equation*: [m, c] — coeficientes da reta y = mx + c
regression.polynomial(points, {order:2})	Calibração do sensor de turbidez (ST100)	*equation*: [c, b, a] — coeficientes do polinômio 2° grau
regression.polynomial(points, {order:3})	Disponível para sensores futuros de 3° grau	*equation:* [d, c, b, a] — coeficientes do polinômio 3° grau
regression.polynomial(points, {order:4})	Disponível para sensores futuros de 4° grau	*equation*: [e, d, c, b, a] — coeficientes do polinômio 4° grau

R^2 (campo do retorno)	Validação da calibração — trava de $R^2 > 0,70$	Coeficiente de determinação entre 0 e 1

Fonte: Autor (2026).

3.2.6.4 *Controllers* — Camada de Roteamento *HTTP*

Os *Controllers* definem as rotas *HTTP* expostas pela *API REST* e fazem a ponte entre as requisições externas e os métodos dos Services. Cada rota valida os parâmetros de entrada, aciona a função de serviço correspondente e retorna o código de status *HTTP* adequado. Rotas sensíveis são protegidas por middlewares de autenticação que verificam a autenticidade e o nível de privilégio do solicitante antes de prosseguir.

3.2.6.4.1 Rotas do DeviceController

As rotas de dispositivos cobrem o ciclo completo de provisionamento e operação do hardware. Um comportamento crítico é o tratamento do código 404 na rota de recepção de telemetria: se o servidor não encontrar o dispositivo no banco de dados, retorna esse status, o que aciona automaticamente a função *handleFactoryReset()* no *firmware*, reiniciando o processo de provisionamento sem necessidade de intervenção física. A tabela 6 mostra o resumo das rotas do *DeviceController.*

Quadro 5 - Rotas do DeviceController e suas funções no ciclo de operação

Método	Rota	Descrição
POST	*/api/devices/ping*	ESP32 anuncia seu *MAC*; cria ou atualiza registro na fila de adoção
GET	*/api/provision/status/:mac*	*Firmware* consulta se já foi adotado e recebe seu *ID* único
PUT	*/api/devices/claim/:mac*	Administrador finaliza o vínculo e configura o dispositivo
GET	*/api/devices/:id/config*	*Firmware* busca intervalo de leitura e lista de sensores ativos
POST	*/api/devices/log/:id*	*Firmware* envia leitura dos sensores; 404 aciona *factory reset*

Fonte: Autor (2026).

3.2.6.4.2 Rotas do CalibrationController

As rotas de calibração orquestram a comunicação entre a *interface web* e o firmware durante todo o ciclo de calibração. A separação entre as rotas de leitura de tensão *(POST /reading)* e de confirmação de referência *(PUT /reference)* reflete a natureza assíncrona do processo: o *firmware* opera de forma contínua enviando tensões, enquanto o operador age no seu próprio ritmo, confirmando o valor apenas quando considera que o sensor estabilizou na amostra tampão. Essa arquitetura elimina a necessidade de sincronização rígida entre hardware e interface web. A tabela 6 mostra o resumo das rotas do *CalibrationController.*

Quadro 6 - Rotas do CalibrationController e suas funções no ciclo de calibração

Método	Rota	Descrição
POST	*/api/calibrations*	Administrador cria sessão de calibração pela *interface web*
GET	*/api/calibration-command/:id*	*Firmware* busca trabalho pendente; transiciona status para *ACTIVE*
POST	*/api/calibrations/:id/reading*	*Firmware* envia tensão bruta continuamente durante a calibração
PUT	*/api/calibrations/:id/reference*	Administrador confirma o valor de referência para o ponto atual
POST	*/api/calibrations/:id/finalize*	Dispara o cálculo da regressão e persiste os coeficientes
GET	*/api/calibrations/latest*	*Firmware* busca coeficientes vigentes antes de cada ciclo de leitura

Fonte: Autor (2026).

3.2.7 Sincronização e tratamento de respostas *HTTP*

O controle de fluxo e a tomada de decisão do *firmware* dependem diretamente do feedback do servidor *Node.js* após cada requisição. Ao realizar o envio de dados via método *POST* ou a busca de configurações via *GET*, o ESP32 analisa o código de status retornado:

3.2.7.1 Resposta *HTTP 200*

Confirma que o pacote de dados foi recebido, processado e persistido com sucesso no banco de dados *PostgreSQL*. Somente após receber este código, o dispositivo atualiza o registro de tempo da última leitura e reinicia o contador do intervalo de espera *(timeInterval).*

3.2.7.2 *Resposta HTTP 404*

Caso o servidor retorne este código, o *firmware* identifica que o *ID* do dispositivo foi deletado do servidor. Como medida de segurança e integridade, o sistema aciona automaticamente a função *handleFactoryReset(),* limpando a memória não volátil e reiniciando o processo de provisionamento.

3.2.7.3 Perda de conexão

Caso o ESP32 não receba uma resposta *HTTP 200* ou perca o sinal Wi-Fi, o sistema transita para o *ERROR_STATE*. O LED de sinalização passa a piscar lentamente (intervalo de 1,5s), fornecendo diagnóstico visual imediato da falha de comunicação.

Durante o estado de erro, a lógica principal de leitura dos sensores e processamento de coeficientes é suspensa. Esta pausa economiza processamento e impede o envio de pacotes de dados incompletos que poderiam comprometer a integridade do histórico no banco de dados.

O dispositivo executa uma tentativa de reconexão a cada 30 segundos através da função *updateDeviceConfiguration*(). Ao obter sucesso na comunicação o microcontrolador redefine seu estado para *NORMAL_OPERATION* e retoma o envio de dados automaticamente, eliminando a necessidade de intervenção técnica manual.

3.2.8 Servidor e infraestrutura de rede

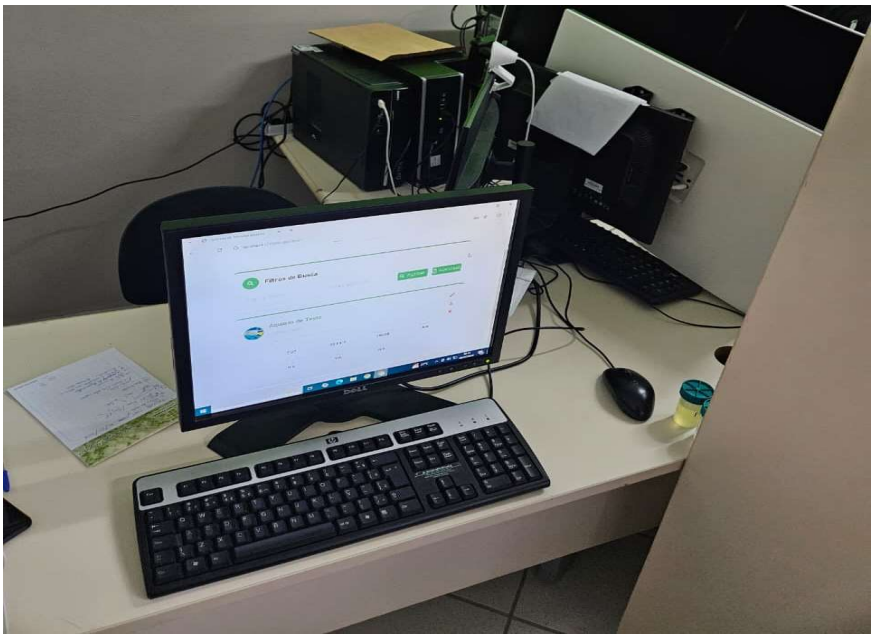
A gestão centralizada do sistema é realizada por um computador dedicado, cedido pelo IFSC, que hospeda de forma integrada as camadas de *backend, frontend* e o banco de dados relacional. A conectividade entre o microcontrolador ESP32 e a

central de processamento ocorre por meio da infraestrutura de rede sem fio (Wi−Fi) interna da instituição.

3.2.8.1 Estabilidade e acesso à rede

Para assegurar a estabilidade das requisições e a localização persistente do serviço, o servidor foi configurado com um endereço IP fixo. Essa configuração permite que o dispositivo de campo envie os dados de forma contínua e que qualquer usuário autenticado, desde que conectado à rede local do *campus*, acesse a interface de monitoramento e os relatórios de qualidade da água em tempo real. O computador utilizado como servidor, pode ser visto na imagem 47.

Imagem 47 - Computador utilizado para gestão do sistema



Fonte: Autor (2026)

3.2.8.2 Protocolos automáticos

Visando assegurar a continuidade do monitoramento e a resiliência do sistema contra interrupções no fornecimento de energia elétrica, foram implementadas rotinas automáticas:

- Recuperação de Hardware: o *firmware* da *BIOS* do computador servidor foi configurado com a funcionalidade *Restore on AC Power Loss*. Esta parametrização garante que a placa-mãe realize o religamento automático

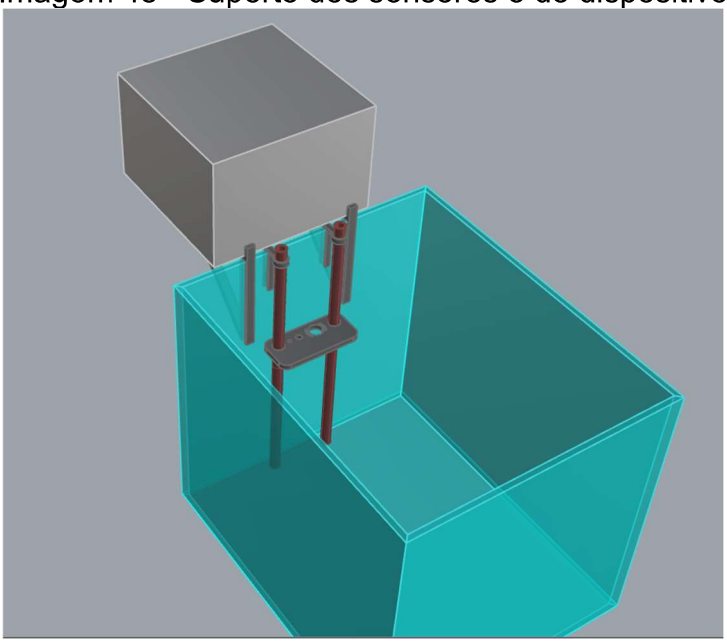
assim que a alimentação da energia for restabelecida, eliminando a necessidade de acionamento manual.

- Automação de *Software*: foram desenvolvidos *scripts* de inicialização integrados ao agendador de tarefas do sistema operacional *Windows*. Estes *scripts* automatizam o carregamento das instâncias do banco de dados *PostgreSQL* e do *servidor Node.js* imediatamente após o *boot*, garantindo que os serviços de *backend* e *frontend* estejam operacionais de forma autônoma.
- Esta abordagem combinada assegura que o sistema retome a aquisição e o armazenamento de dados sem intervenção humana, minimizando lacunas no histórico de parâmetros.

3.2.9 Suporte

Para a fixação do dispositivo e dos sensores, foi desenvolvido no *software Rhino 8* em sua versão estudantil, uma plataforma em que os sensores são alocados, ajustando conforme o nível de água no aquário. Imagem da modelagem pode ser observada na figura 48.

Imagem 48 - Suporte dos sensores e do dispositivo



Fonte: Autor (2026).

[1] Disponível em: https://github.com/yanschneider98/TCC

4 DESENVOLVIMENTO

Este capítulo detalha o processo de concepção e implementação do sistema de monitoramento para os aquários do laboratório de piscicultura do IFSC. O objetivo é escopo do protótipo funcional desenvolvido.

4.1. DISPOSITIVO FÍSICO

Para o dispositivo foi criado um software genérico, nele é contido a informação de endereço do servidor, além da implementação dos sensores, apesar de não ter todos os sensores instalados nesse primeiro momento, o software através da interface web, consegue indicar quais sensores o usuário pode ativar para fazer as leituras.

Para a interação do usuário, a caixa foi equipada com dois controles externos:

- Um seletor Ligar/Desligar, que permite ao utilizador cortar a energia do circuito de forma segura, sem a necessidade de desconectar o cabo de alimentação.
- Uma botoeira multifuncional com LED integrado, que serve tanto como meio de entrada de comandos quanto como principal indicador visual do estado operacional do dispositivo.

4.1.1 Máquina de estados e feedback visual

O *firmware* utiliza uma Máquina de Estados para monitorar a conectividade e a integridade da comunicação, fornecendo diagnóstico imediato via LED:

- Estado de Erro (Piscando Lento - 1,5s): Transição disparada por falha de conexão Wi-Fi ou erro de comunicação com o servidor. O sistema entra em ciclo de tentativa de reconexão automática.
- Estado de Operação Normal (Aceso): Validado pelo recebimento de um código *HTTP 200*. Indica que as leituras e envios estão ocorrendo nos intervalos programados.
- Estado de Calibração (Piscando Rápido - 500ms): Acionado por clique curto na botoeira, coloca o hardware em espera para comandos vindos da interface web.

4.1.2 Interação manual: leitura forçada

Adicionalmente, foi implementada a função de "leitura forçada". Se o utilizador mantiver a botoeira pressionada por mais de quatro segundos em modo normal, o firmware dispara uma leitura imediata de todos os sensores ativos e tenta o envio instantâneo para o servidor, ignorando o cronograma pré-programado. Essa funcionalidade é essencial para diagnósticos rápidos e verificação de resposta do sistema sob demanda. A imagem com o dispositivo montado e operando pode ser observada na imagem 49.

Imagem 49 - Dispositivo com o led aceso

Fonte: Autor (2026).

4.1.3 Suporte de fixação

Para garantir a correta instalação e operação dos sensores no ambiente do laboratório, foi projetado e construído um suporte de fixação customizado. Este suporte, fabricado através do recurso da impressora 3D do IFSC, permite acoplar o

dispositivo de hardware e os seus sensores de forma segura ao tanque de retorno de água da bancada. Foi incorporado um mecanismo de ajuste de altura, assegurando que os eletrodos dos sensores permaneçam sempre submersos no nível correto, independentemente de pequenas variações no fluxo de água, garantindo assim a consistência das medições. O suporte desenvolvido para os sensores, pode ser observado na imagem 50.

Imagem 50 - Suporte dos sensores

Fonte: Autor (2026).

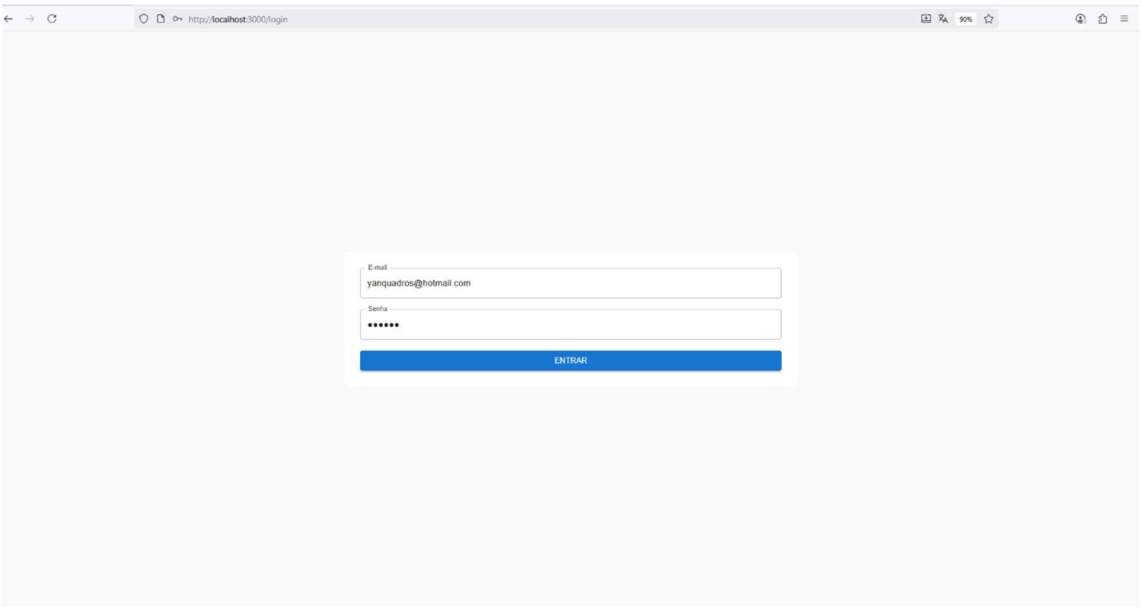
4.2 IMPLEMENTAÇÃO DO SERVIDOR E DA INTERFACE WEB

A construção do servidor e da *interface web* focou-se em criar uma plataforma centralizada, segura e intuitiva para a gestão e visualização de todos os dados coletados. A seguir, detalham-se as principais funcionalidades desenvolvidas.

4.2.1. Sistema de autenticação e níveis de permissão

Para garantir a segurança e a integridade das informações, foi implementado um sistema de autenticação onde apenas usuários previamente cadastrados podem acessar à aplicação através de uma página de login, conforme a Imagem 51.

Imagem 51 - Tela de login



Fonte: Autor (2026).

Uma vez autenticado, o sistema aplica um controle de acesso baseado em dois níveis de permissão:

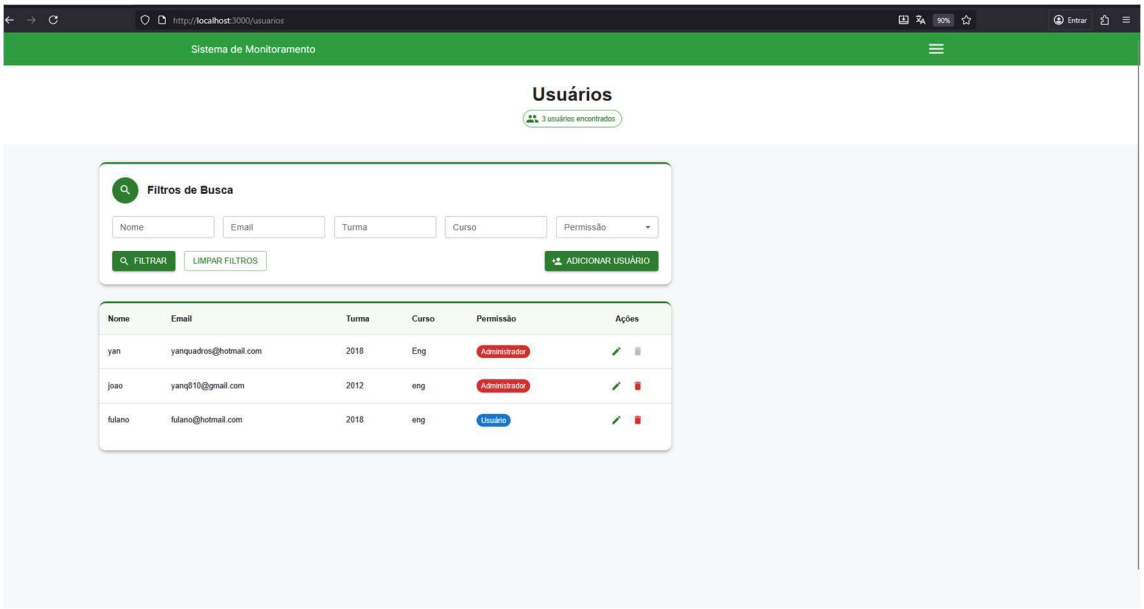
- Administrador: possui controle total sobre a plataforma. Este utilizador é o único com permissão para criar, editar e excluir outros utilizadores e dispositivos, além de ser responsável por todo o processo de calibração e pela gestão de quais sensores estão habilitados para enviar dados. A tela inicial do sistema com o usuário do tipo administrador, pode ser vista na imagem 52.

Imagem 52 - Tela principal com usuário administrador

Fonte: Autor (2026).

Uma vez autenticado com o usuário do tipo administrador, será possível acessar página de usuário, nela é possível fazer as alterações de usuários. conforme a Imagem 53.

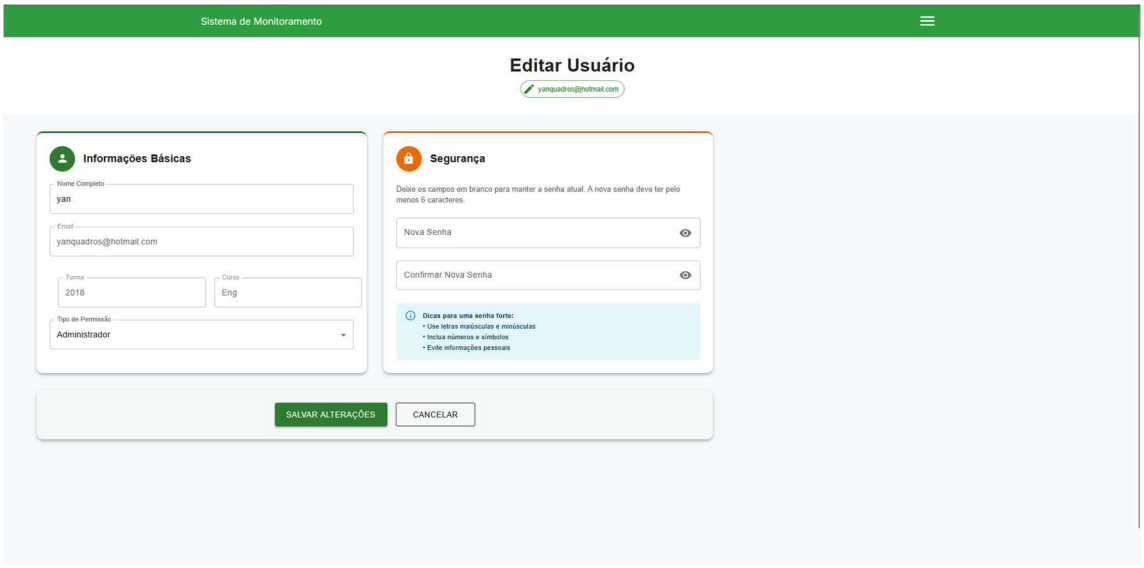
Imagem 53 - Tela de usuários



Fonte: Autor (2026).

Ao selecionar para editar um usuário, é possível fazer alteração de senha, nome e tipo de usuário, conforme imagem 54.

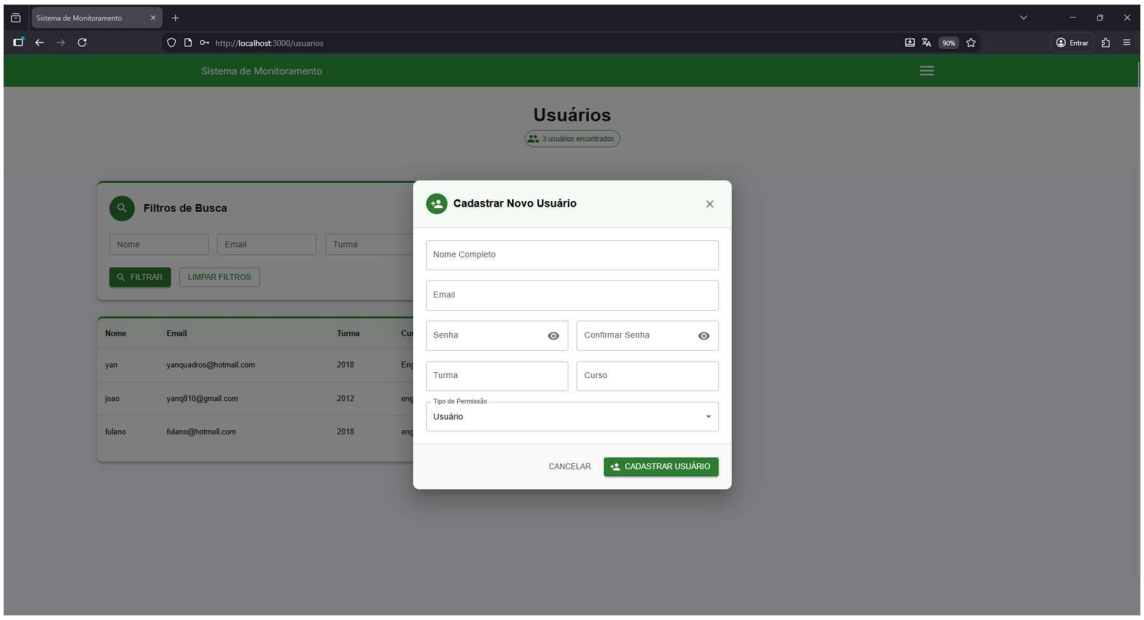
Imagem 54 - Tela de edição de usuário



Fonte: Autor (2026).

É possível também na tela de usuário, cadastrar um novo usuário, informando, nome, Email, senha, turma, curso e tipo de permissão, conforme imagem 55.

Imagem 55 - Tela de cadastro de usuário.

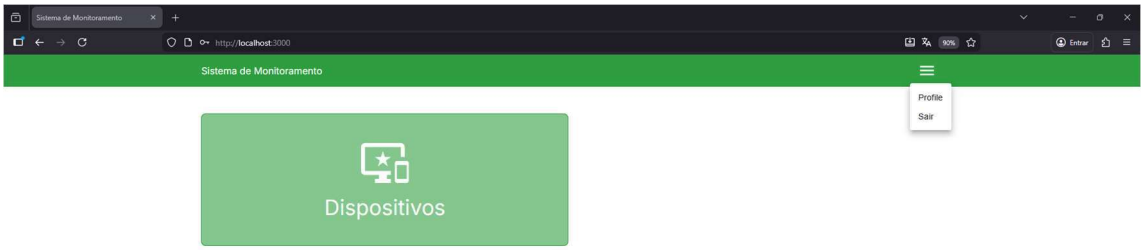


Fonte: Autor (2026).

- Utilizador Comum: possui acesso de apenas leitura. Pode visualizar os dados de todos os dispositivos cadastrados e os relatórios, mas não pode realizar alterações. A sua única permissão de escrita é para gerir as suas próprias

informações de perfil, como a alteração da sua senha. A tela inicial do sistema com o usuário do tipo comum, pode ser vista na imagem 56.

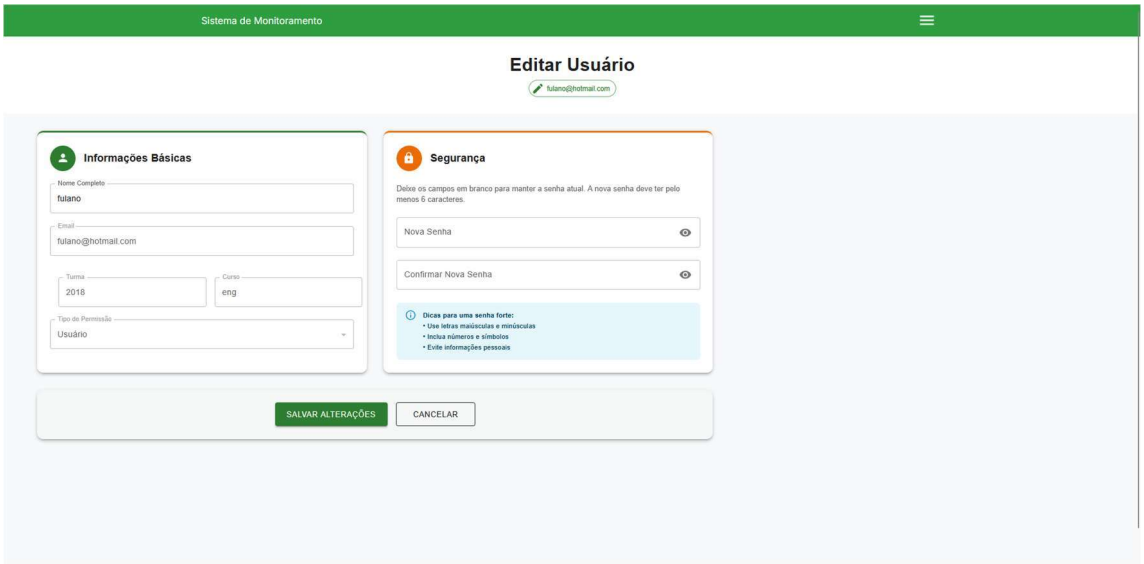
Imagem 56 - Tela principal com usuário comum



Fonte: Autor (2026).

Ao acessar a barra de menu no canto superior direito (Imagem 56), na opção *Profile*, é possível acessar a página, para gerenciar as próprias informações do usuário logado, permitindo trocar apenas o nome e a senha (Imagem 57).

Imagem 57 - Tela de edição de usuário



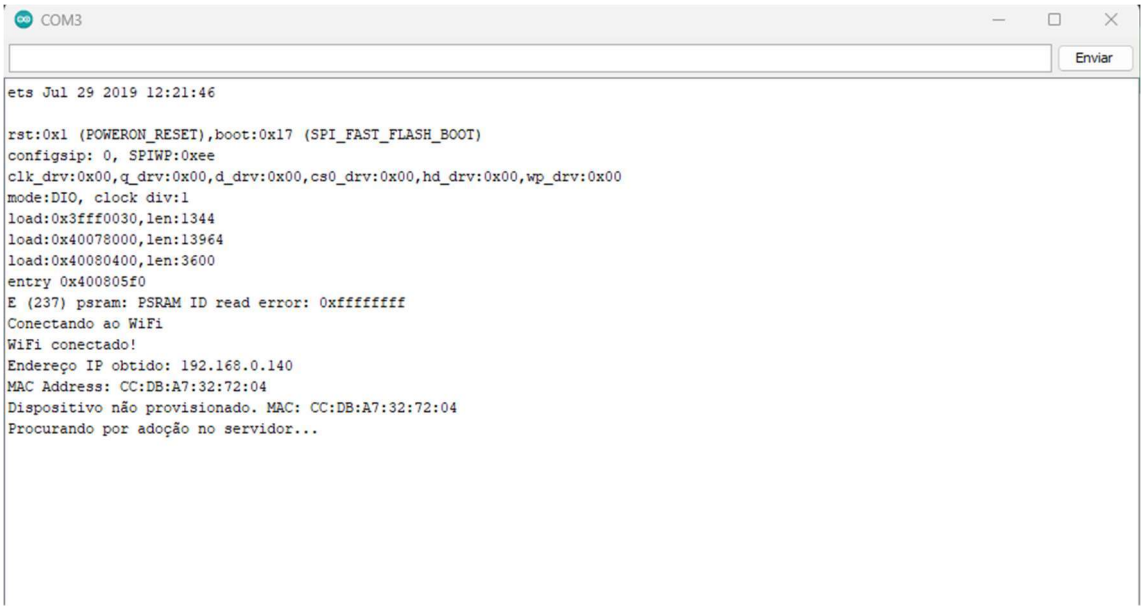
Fonte: Autor (2026).

4.2.2. Fluxo de provisionamento automático de dispositivos

Para maximizar a usabilidade e eliminar a necessidade de reprogramar o hardware para cada novo dispositivo, foi desenvolvido um fluxo de "adoção" totalmente gerido pela interface web.

O dispositivo já possui em seu programação o endeço do servidor e a credenciais para se autenticar no *WIFI*, dessa forma ao ligar um dispositivo novo e não configurado, ele se conecta ao servidor e envia seu *MAC*, o servidor ao receber essa informação, faz uma busca nos dispositivos cadastrados e verifica se o idenficador não está cadastrado. Caso o *MAC* esteja cadastrado, o servidor informa o endereço único criado pelo sistema. O monitor serial do ESP-32 ao ser iniciado pela primeira vez, pode ser vista na imagem 58.

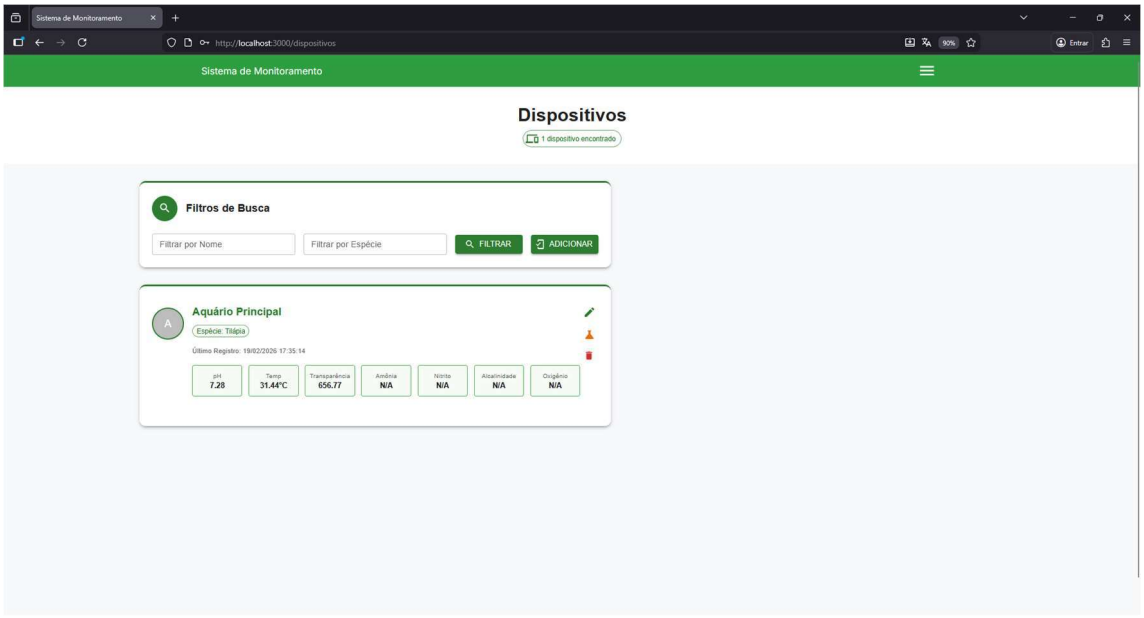
Imagem 58 - Monitor serial do ESP-32



Fonte: Autor (2026).

Conforme a imagem 59, ao acessar a página de dispositivos, é possível visualizar dispositivos já cadastrados e adicionar dispositivo, essa opção de adicionar dispositvo só está visível ao usuário do tipo administrador.

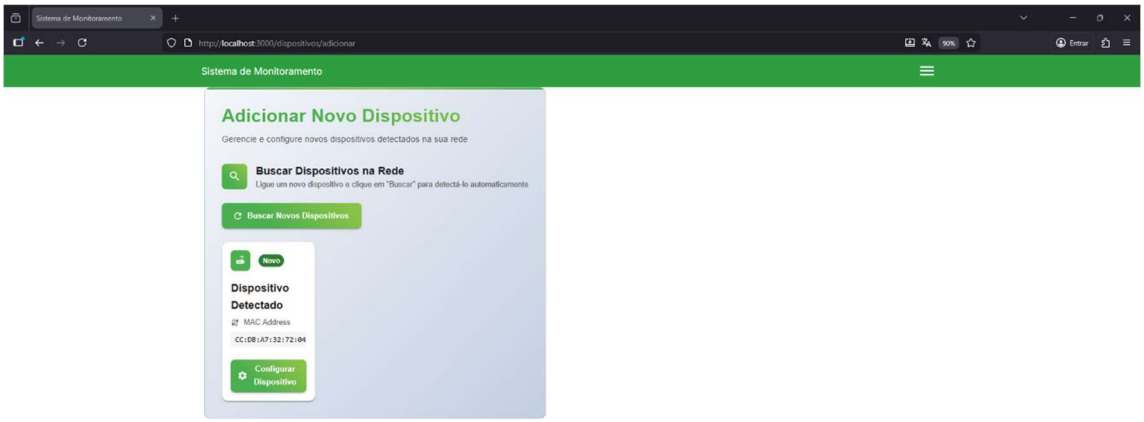
Imagem 59 - Tela de dispositivos com o usuário do tipo Administrador



Fonte: Autor (2026).

Na página de "Adicionar Dispositivos", o administrador visualiza uma lista de equipamentos que estão "Aguardando Adoção" (Imagem 60).

Imagem 60 - Tela de adicionar dispositivo

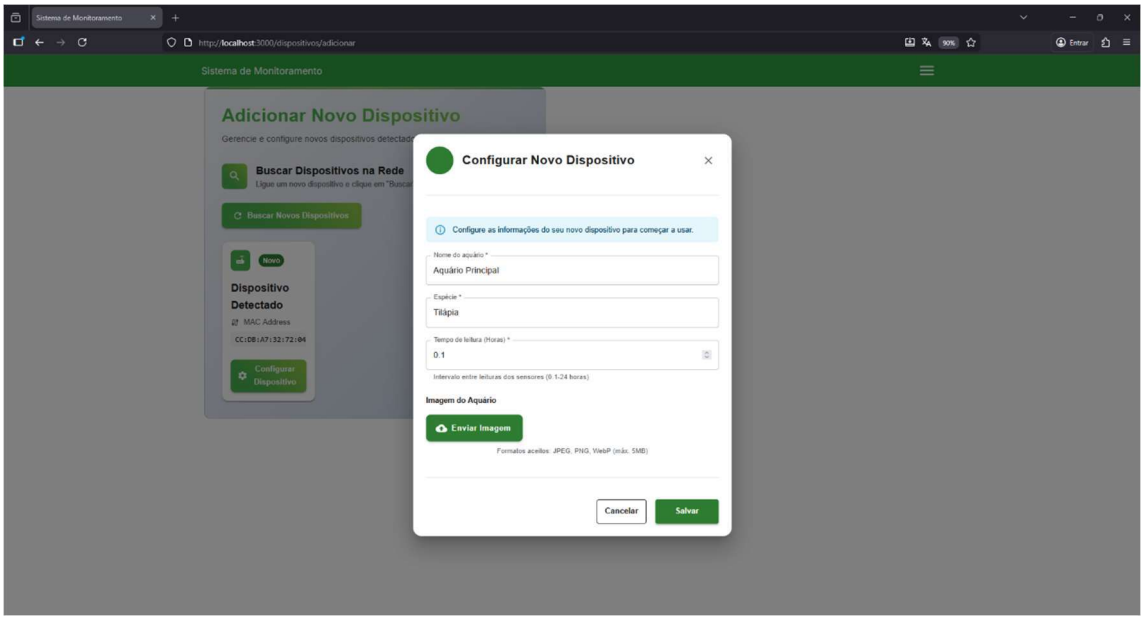


Fonte: Autor (2026)

Ao selecionar o dispositivo, é apresentada uma interface para configurar os detalhes do novo dispositivo, como atribuir-lhe um nome ("Bancada 01"), definir o tipo

de espécie ("Tilápia"), inserir uma fotografia e, crucialmente, definir o intervalo de tempo entre as leituras (ex: a cada 1 hora) (Imagem 61).

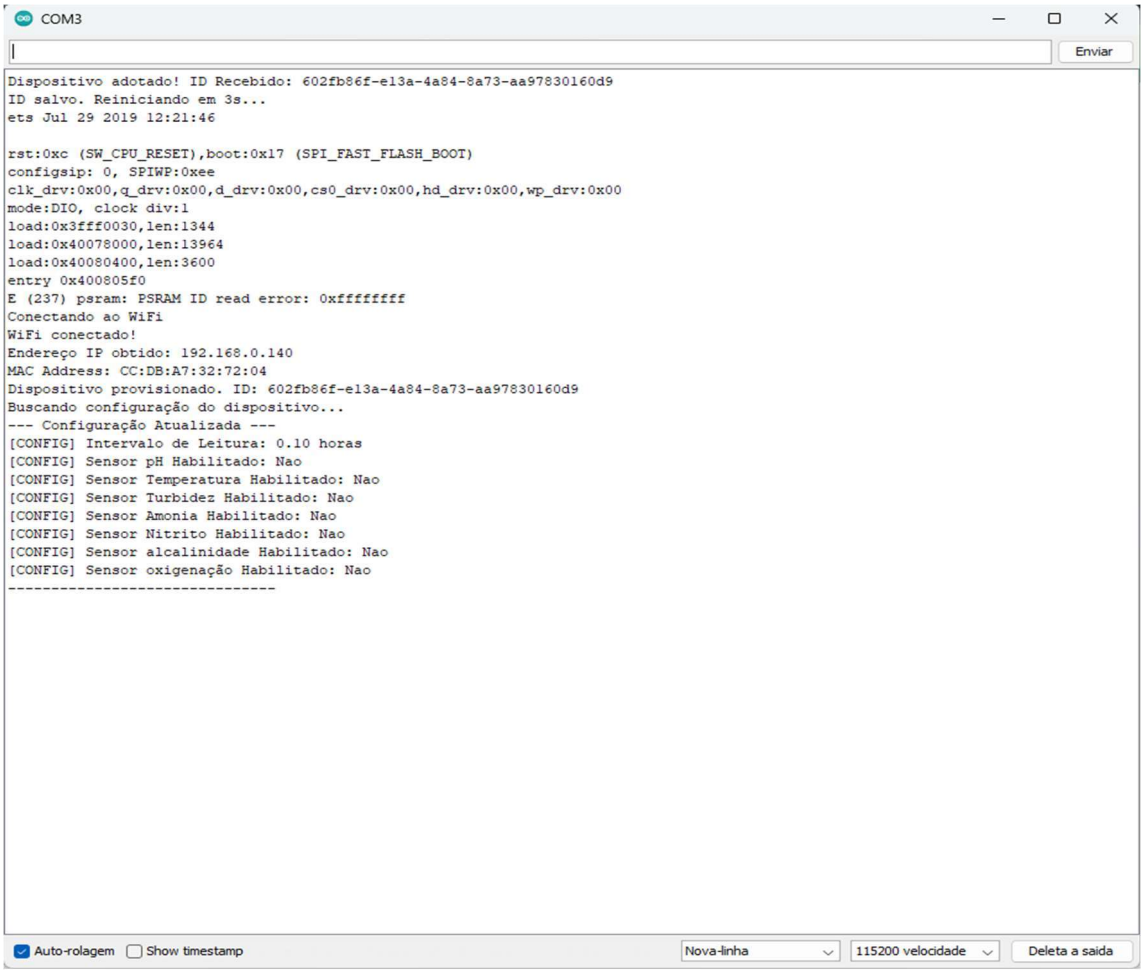
Imagem 61 - Tela de cadastro de dispositivo



Fonte: Autor (2026).

Após o cadastro, o servidor atribui um identificador único (ID) ao dispositivo. Na sua próxima comunicação, o *hardware* recebe este ID, salva-o permanentemente na sua memória interna não volatil e reinicia, finalizando o processo de provisionamento e entrando em modo de operação normal. Este procedimento é crítico: uma vez salvo, o dispositivo deixa de solicitar o provisionamento e passa a realizar apenas a sincronização de configurações, garantindo que mesmo após quedas de energia, o vínculo entre o hardware físico e sua identidade lógica no banco de dados permaneça íntegro. O monitor serial do ESP-32 ao receber adoção, pode ser vista na imagem 62.

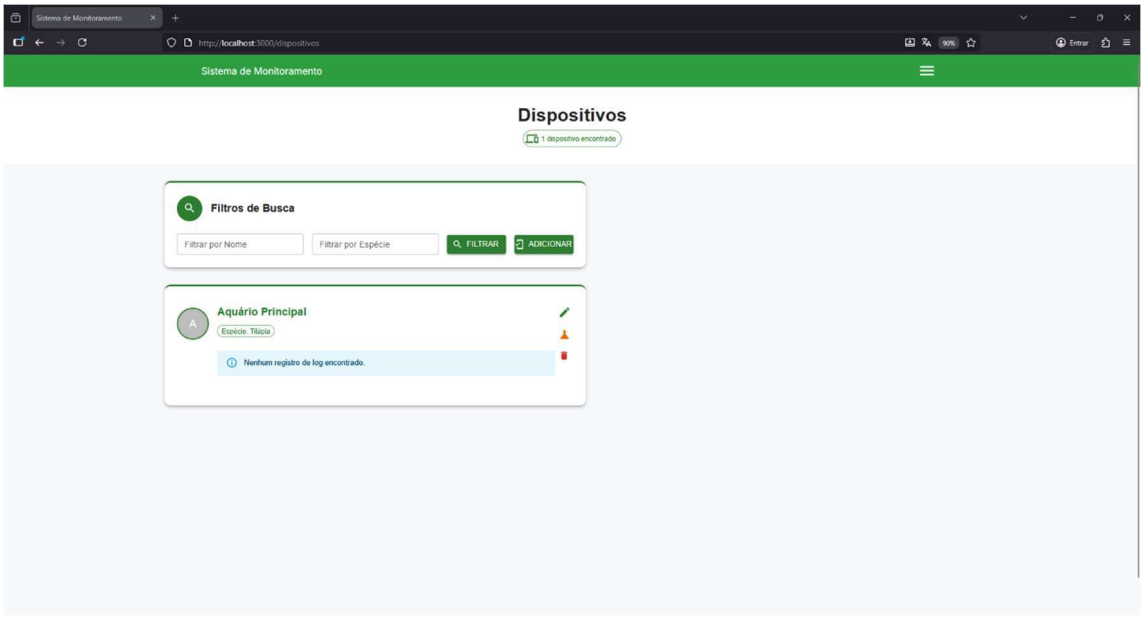
Imagem 62 - Monitor serial do ESP-32 ao receber adoção



Fonte: Autor (2026).

Após o registro, o dispositivo é listado na página de dispositivos, essa página é equipada com ferramentas de busca e filtros por categorias. Para garantir a integridade das configurações, o sistema restringe operações críticas, como edição, parametrização de calibração e remoção do dispositivo exclusivamente a usuários com privilégios administrativos, conforme ilustrado na imagem 63.

Imagem 63 - Tela de dispositivos

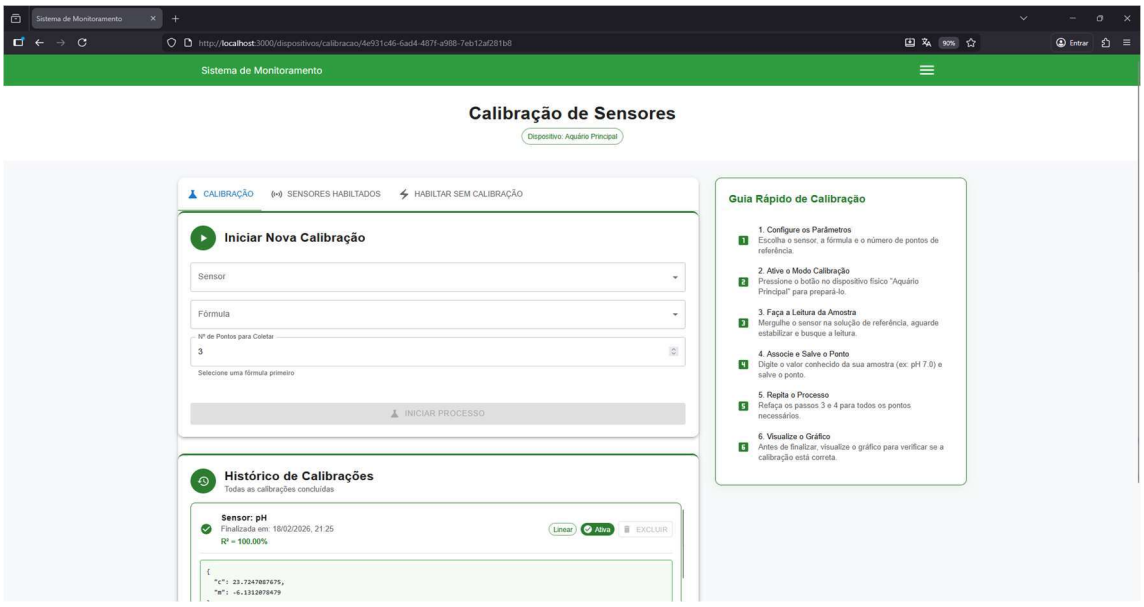


Fonte: Autor (2026).

4.2.3. Módulo de calibração e gestão de qualidade

Foi desenvolvida uma interface dedicada e interativa para a calibração dos sensores, o acesso a esta funcionalidade é realizado de forma intuitiva através do botão identificado pelo ícone de um tubo de ensaio na cor laranja, conforme ilustrado na Imagem 63. A página dedicada a calibração pode ser vista na imagem 64.

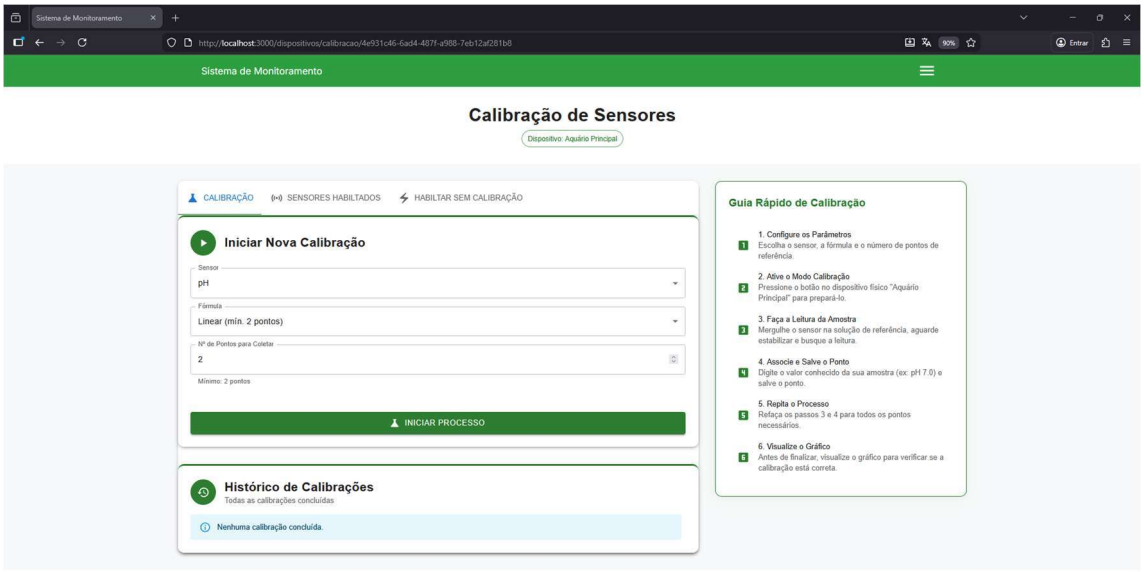
Imagem 64 - Página de calibração.



Fonte: Autor (2026).

O processo foi projetado para ser guiado: o administrador seleciona o sensor, a fórmula matemática desejada (Linear, Polinomial de 2º Grau até Polinomial de 4° Grau) e a quantidade de pontos de referência que irá utilizar. Ao lado direito da tela é possível visualizar um guia rápido de Calibração (Imagem 65).

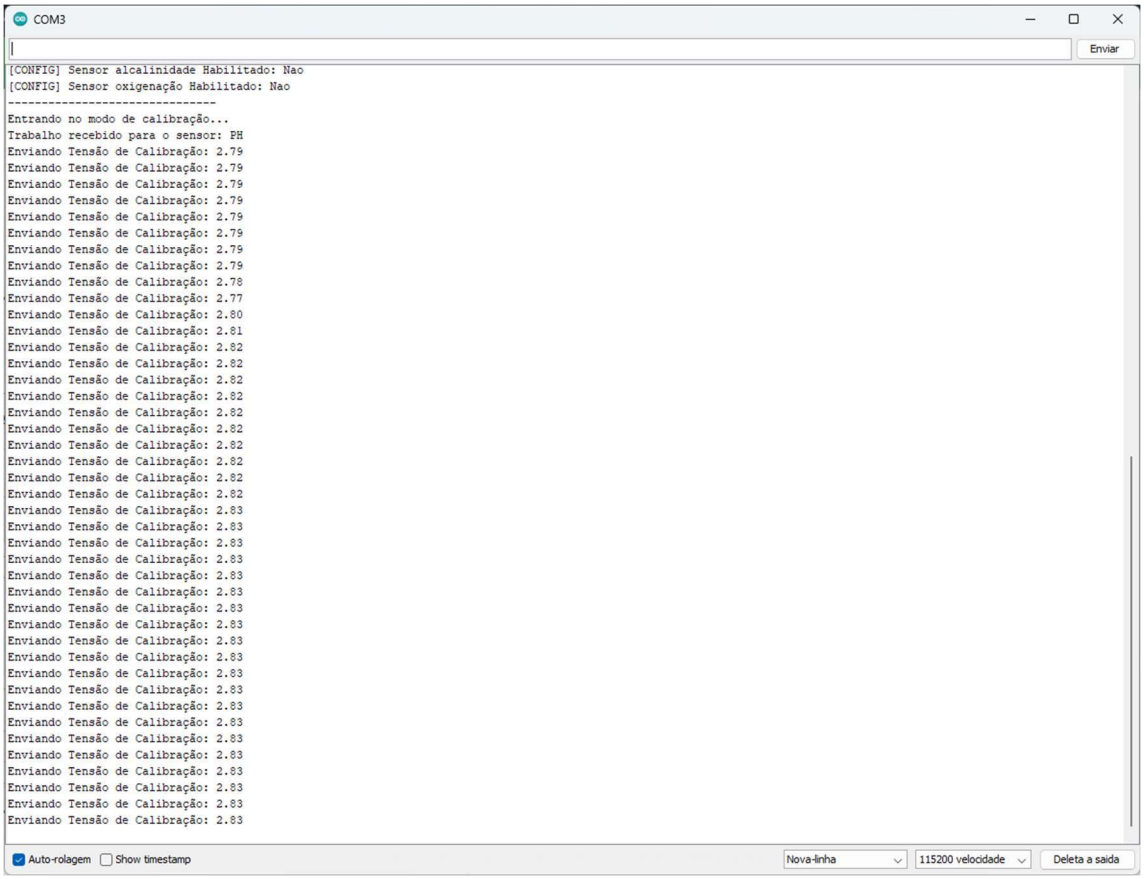
Imagem 65 - Página de calibração em andamento



Fonte: Autor (2026).

Após escolher os parâmetros para realizar a calibração, é necessário apertar e soltar o botão no dispositivo físico, dessa forma o dispositivo entra em modo calibração, busca as informações no servidor, com a requisição de trabalho obtida, o dispositivo começa a capturar a tensão lida no sensor e envia para o servidor. O administrador então remove o sensor suporte e coloca em uma amostra tampão (exemplo: "7.0" para uma solução de pH neutro). O monitor serial do ESP-32 ao receber instruções de calibração, pode ser vista na imagem 66.

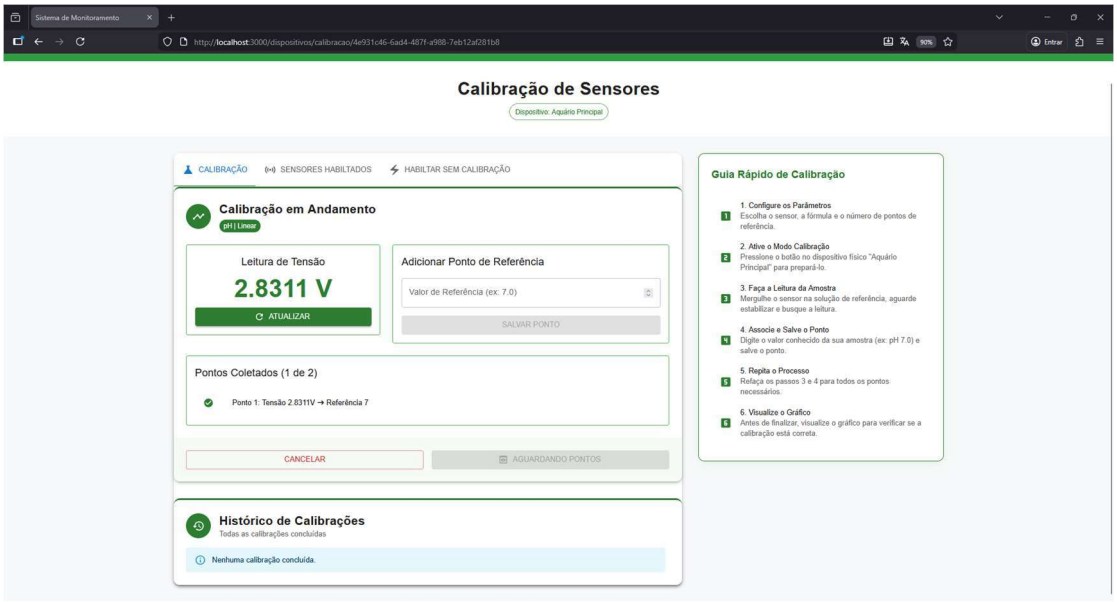
Imagem 66 - Monitor serial ESP32 em modo calibração.



Fonte: Autor (2026).

É necessário aguardar que a tensão lida no sensor, fique estável, garantido que o sensor chegou na tensão correta para aquela amostra. Após isso, o administrador então insere o valor de referência conhecido e salva (tensão vs. referência), essa tela pode ser vista na imagem 67.

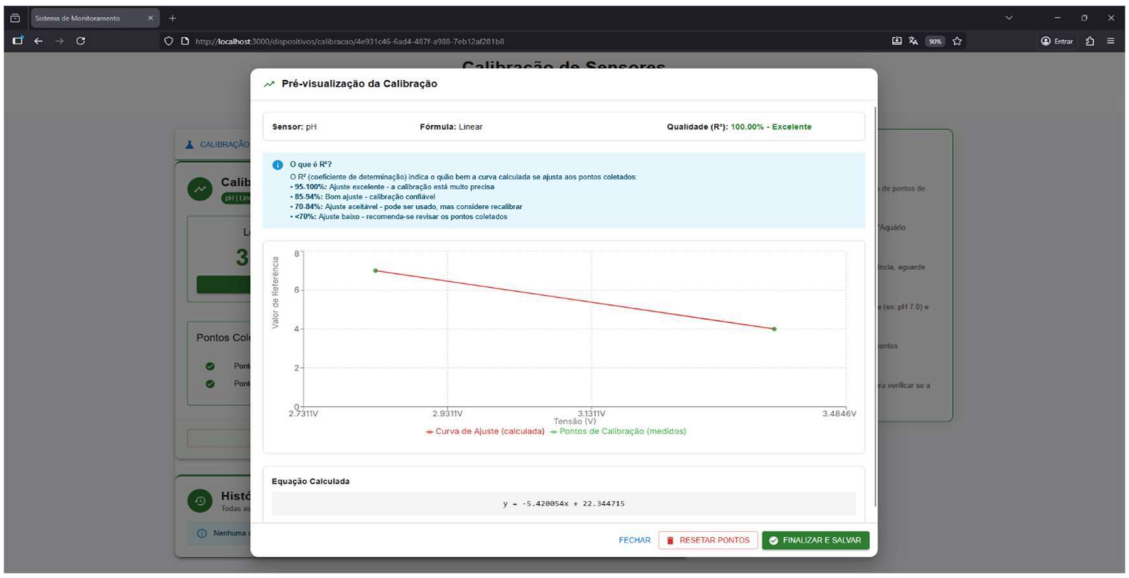
Imagem 67 - Página de calibração em andamento



Fonte: Autor (2026).

Com todos os pontos coletados, o sistema exibe um gráfico de pré-visualização que plota os pontos do utilizador e a curva de regressão calculada. A interface também exibe o coeficiente de determinação (R^2), que mede a qualidade do ajuste. O botão "Salvar Calibração" só é habilitado se o R^2 for superior a 70%, garantindo que apenas calibrações confiáveis sejam salvas no sistema. Gráfico pré-visualização da calibração pode ser observado na imagem 68.

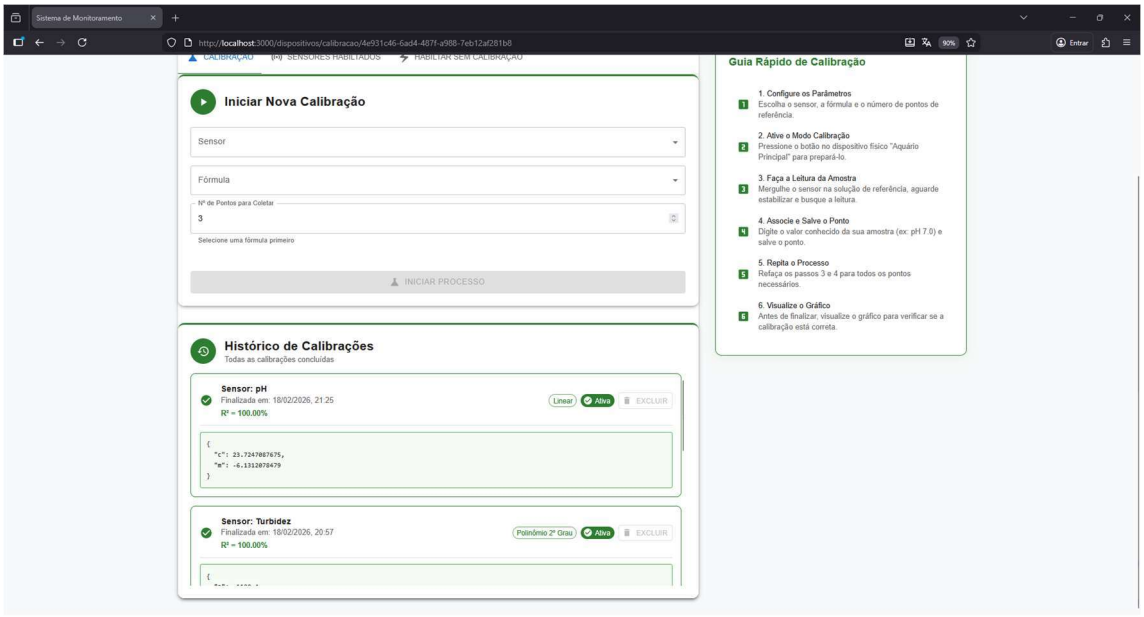
Imagem 68 - Página de pré-visualização da calibração



Fonte: Autor (2026).

Todas as calibrações válidas (com $R^2 > 70\%$) são guardadas em um histórico. O sistema oferece ao administrador a flexibilidade de visualizar este histórico e, a qualquer momento, reativar uma calibração antiga que considere mais confiável, conforme imagem 69.

Imagem 69 - Histórico de calibração.



Fonte: Autor (2026).

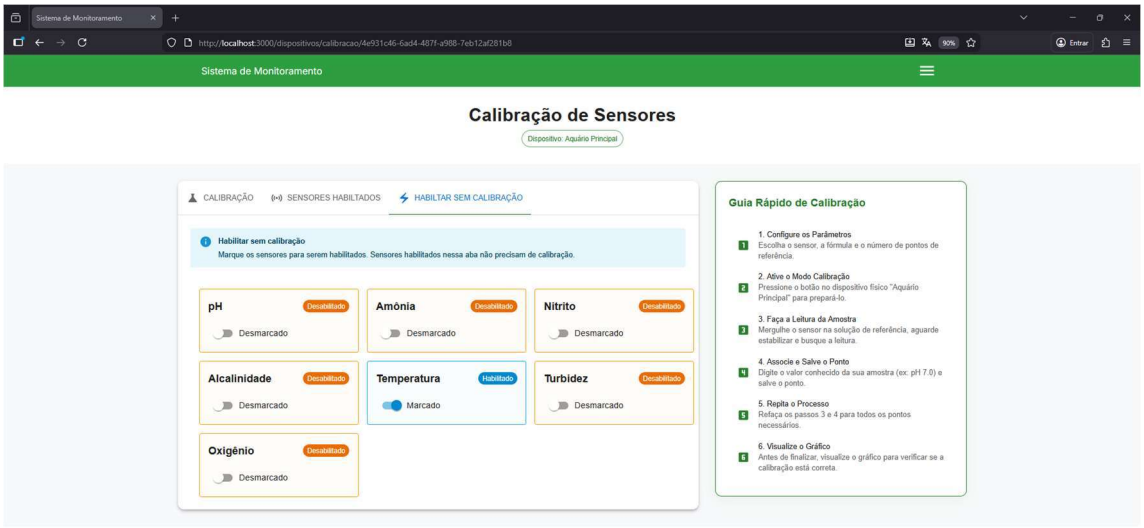
Ao finalizar a calibração, é necessário apertar o botão do dispositivo para voltar ao modo ao modo de operação normal, ao sair, o dispositivo faz uma nova coleta de informações no servidor.

4.2.4. Gestão de sensores

A *interface web* atua como o cérebro estratégico do monitoramento, funcionando como um painel de controle remoto que dita o comportamento do hardware em tempo real. Em vez de o ESP32 decidir sozinho o que ler, ele consulta a interface para saber quais tarefas deve executar.

Ainda na página de calibração, existe duas abas, uma delas é “Habilitar sem calibração” nela é permitido habilitar um sensor sem calibração, como por exemplo o sensor de temperatura, que por padrão, não precisa de regressão complexa para funcionar perfeitamente. A aba de habilitar sensores sem calibração pode ser observada na imagem 70.

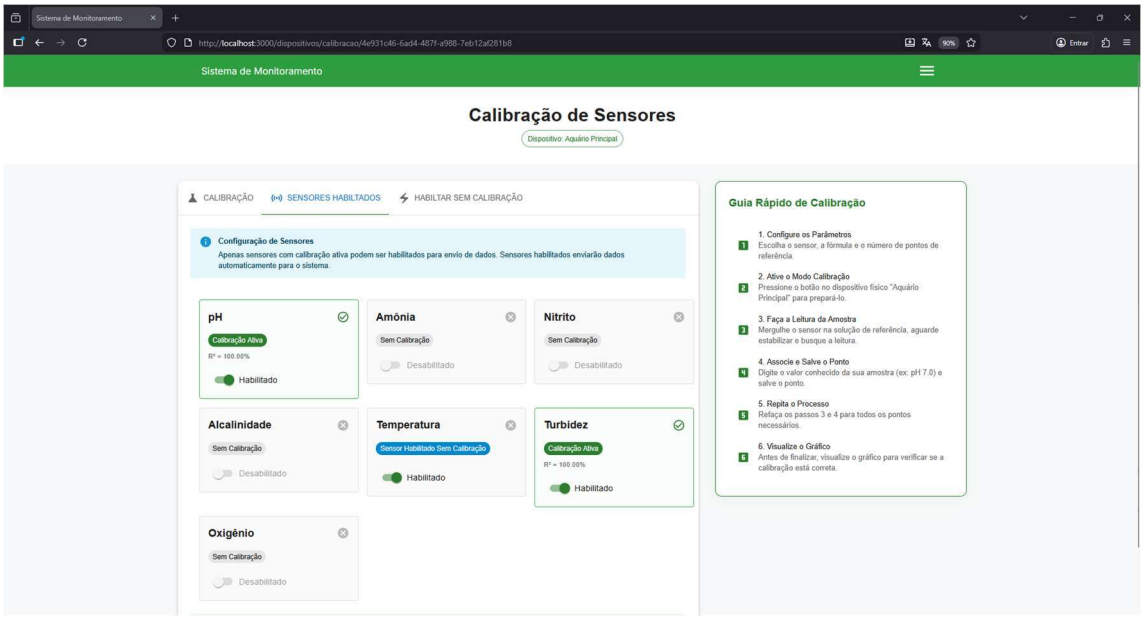
Imagem 70 - Aba de habilitar sem calibração



Fonte: Autor (2026).

Já na aba de “sensores habilitados”, é possível habilitar os sensores que tenha uma calibração ativa ou se estão habilitados na aba de “Habilitar sem calibração”, dessa forma a interface web tem o controle total sobre quais sensores e quando eles são habilitados, evitando assim de que o dispositivo físico preencha o banco de com valores sem sentidos físicos. A aba de habilitar sensores pode ser observada na imagem 71.

Imagem 71 - Aba de habilitar sensores



Fonte: Autor (2026).

Para garantir que as informações de sensores e coeficientes de calibração estejam sempre atualizado com a interface web, o dispositivo, sempre antes de fazer envio de dado, busca as configurações atulizadas do backend. Monitor serial do ESP-32 ao realizando processo de leitura dos sensores (Imagem 72).

Imagem 72 - Monitor serial ESP-32 realizando leitura de dados.

```
E (237) psram: PSRAM ID read error: 0xffffffff
Conectando ao WiFi
WiFi conectado!
Endereço IP obtido: 192.168.0.141
MAC Address: CC:DB:A7:32:72:04
Dispositivo provisionado. ID: 1041ac4b-54f1-47b4-9261-926be4255673
Buscando configuração do dispositivo...
--- Configuração Atualizada ---
[CONFIG] Intervalo de Leitura: 0.10 horas
[CONFIG] Sensor pH Habilitado: Sim
[CONFIG] Sensor Temperatura Habilitado: Sim
[CONFIG] Sensor Turbidez Habilitado: Sim
[CONFIG] Sensor Amonia Habilitado: Nao
[CONFIG] Sensor Nitrito Habilitado: Nao
[CONFIG] Sensor alcalinidade Habilitado: Nao
[CONFIG] Sensor oxigenação Habilitado: Nao
-------------------------------
Leitura forçada...

--- Nova Leitura ---
--- Coeficientes para PH ---
[COEFFS] Formula: LINEAR
[COEFFS] m=-5.68, c=21.97
--------------------------------
--- Coeficientes para TURBIDITY ---
[COEFFS] Formula: null
[COEFFS] m=1.00, c=0.00
--------------------------------
Temperatura: 25.19
pH: 7.00
Turbidez: 4.04
Amonia: Inativo
Nitrito: Inativo
Alcalinidade: Inativo
Oxigenio: Inativo
```

Fonte: Autor (2026).

4.3. CALIBRAÇÃO DOS SENSORES

A calibração é a etapa que correlaciona os sinais de tensão lidos pelo conversor ADS1115 com os valores reais dos parâmetros físico-químicos da água.

4.3.1 Calibração do sensor de pH

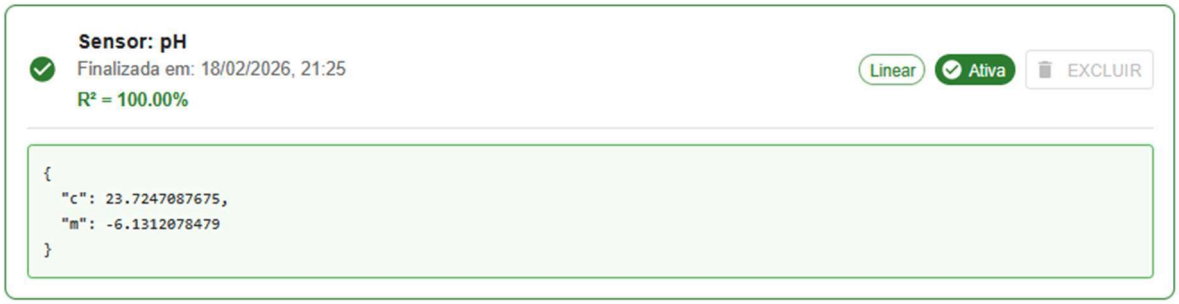
Para o sensor de pH, foi adotado o método de calibração em dois pontos, utilizando soluções tampão de referência com valores nominais de 4,0 e 7,0. A partir das tensões coletadas em cada amostra, realizou-se uma regressão linear para determinar a curva de resposta do sensor. Esta relação permite converter a tensão bruta em pH através da equação:

$$pH = (m.V) + c$$

Equação 2: Equação linear.

A calibração ativa do sensor pode ser vista na imagem 73:

Imagem 73 - Coeficiente de calibração do pH



Fonte: Autor (2026).

4.3.2 Calibração sensor de turbidez

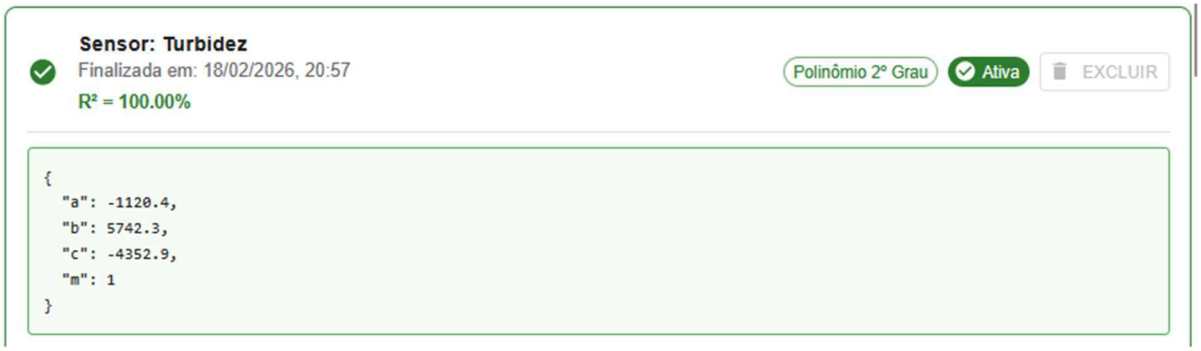
Para o sensor de turbidez (ST100), devido à indisponibilidade de amostras padrão de formazina (NTU) para calibração prática, utilizou-se a curva de transferência característica fornecida pela fabricante *DFRobot* para sensores baseados no modelo SEN0189 (DFROBOT, 2024). O comportamento do sensor é descrito por uma equação polinomial de segunda ordem, que correlaciona a tensão de saída (V) com a turbidez em unidades nefelométricas (NTU):

$$T(NTU) = -1120{,}4 * V^2 + 5742{,}3 * V - 4352{,}9$$

Equação 3: Equação polinomial de 2° grau.

Conforme a documentação técnica, esta fórmula é válida para leituras de tensão entre 2,5V e 4,2V, a imagem da calibração ativa pode ser vista na imagem 74:

Imagem 74 - Coeficiente de calibração da turbidez.



Fonte: Autor (2026).

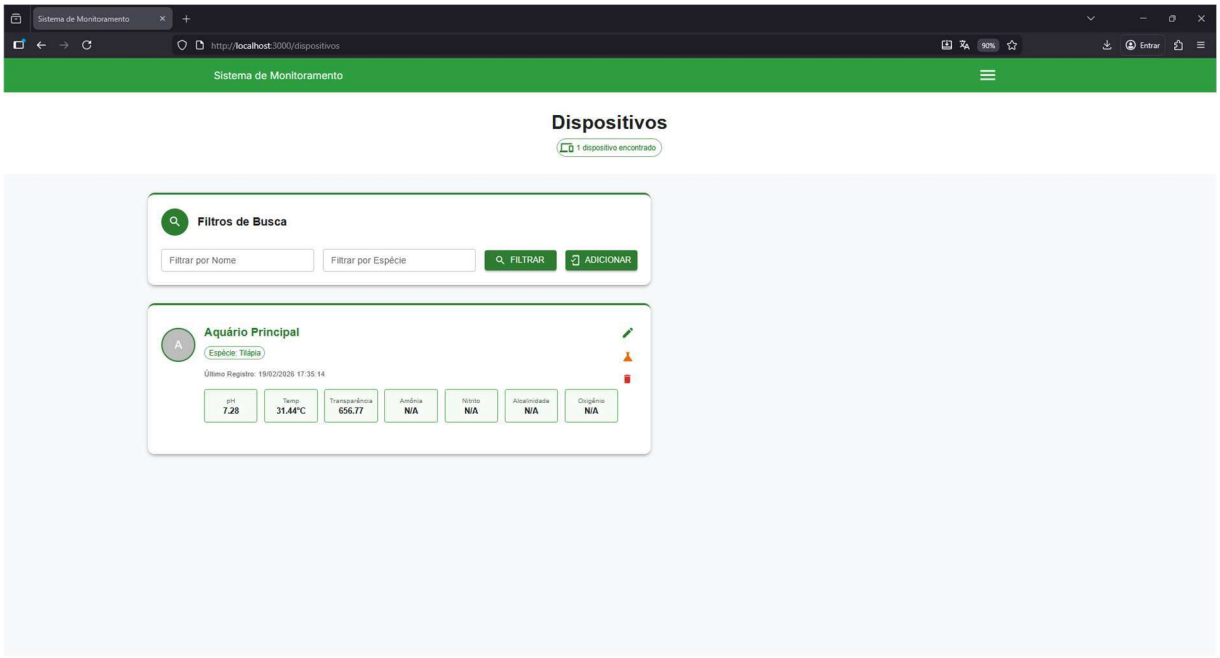
4.3.3 Calibração sensor de temperatura

Diferente dos transdutores analógicos, o sensor DS18B20 não requer calibração pelo usuário, sendo habilitado diretamente no sistema. Trata-se de um dispositivo digital pré-calibrado de fábrica através de um processo de ajuste a laser (laser-trimming), garantindo alta precisão sem a necessidade de ajustes externos (MAXIM INTEGRATED, 2018). O componente possui um conversor ADC interno próprio e utiliza o protocolo de comunicação *1-Wire*, o que assegura que a informação térmica seja transmitida ao ESP32 de forma puramente digital, eliminando erros de ganho ou ruídos elétricos que afetariam uma cadeia de medição analógica.

4.4. VISUALIZAÇÃO DE DADOS E RELATÓRIOS

Uma vez que os dados são recebidos e armazenados, a *interface web* torna-se a principal ferramenta de visualização e análise. Na imagem 75, está a página principal com os dispositivos cadastrados.

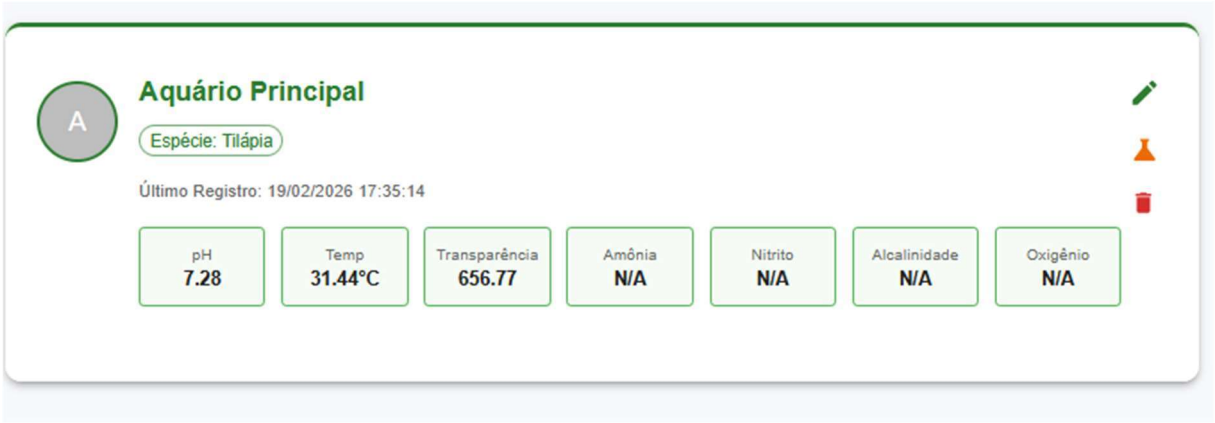
Imagem 75 - Páginas de dispositivos.



Fonte: Autor (2026).

Cada dispositivo possui uma página dedicada que exibe a sua última leitura de todos os sensores ativos (Imagem 76) e uma tabela paginada com o histórico completo de todos os registos recebidos.

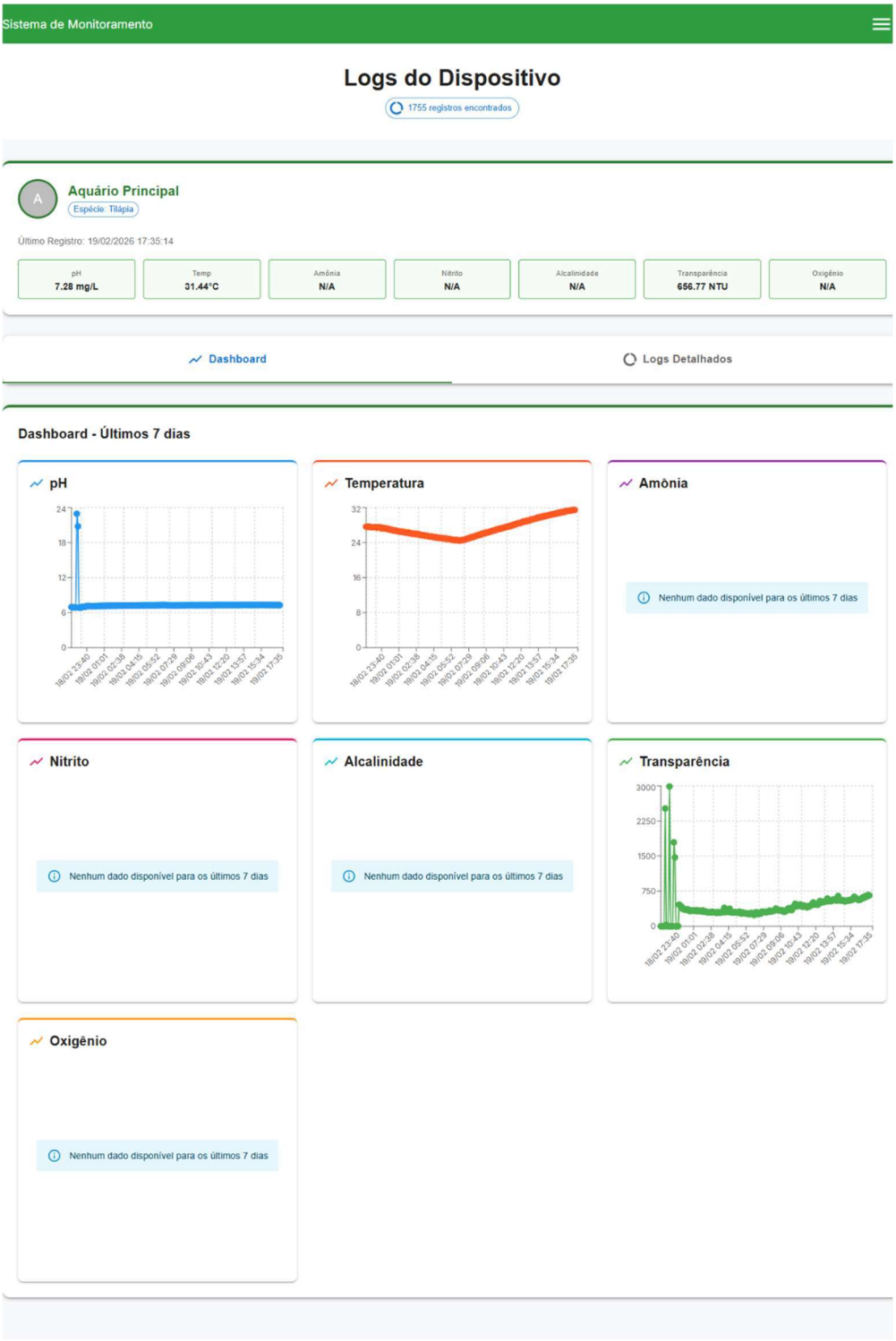
Imagem 76 - Página de dispositivo último registro



Fonte: Autor (2026).

Ao abrir a tela de dispositivo, é possível visualizar um dashboard com o histórico dos últimos 7 dias (Imagem 77).

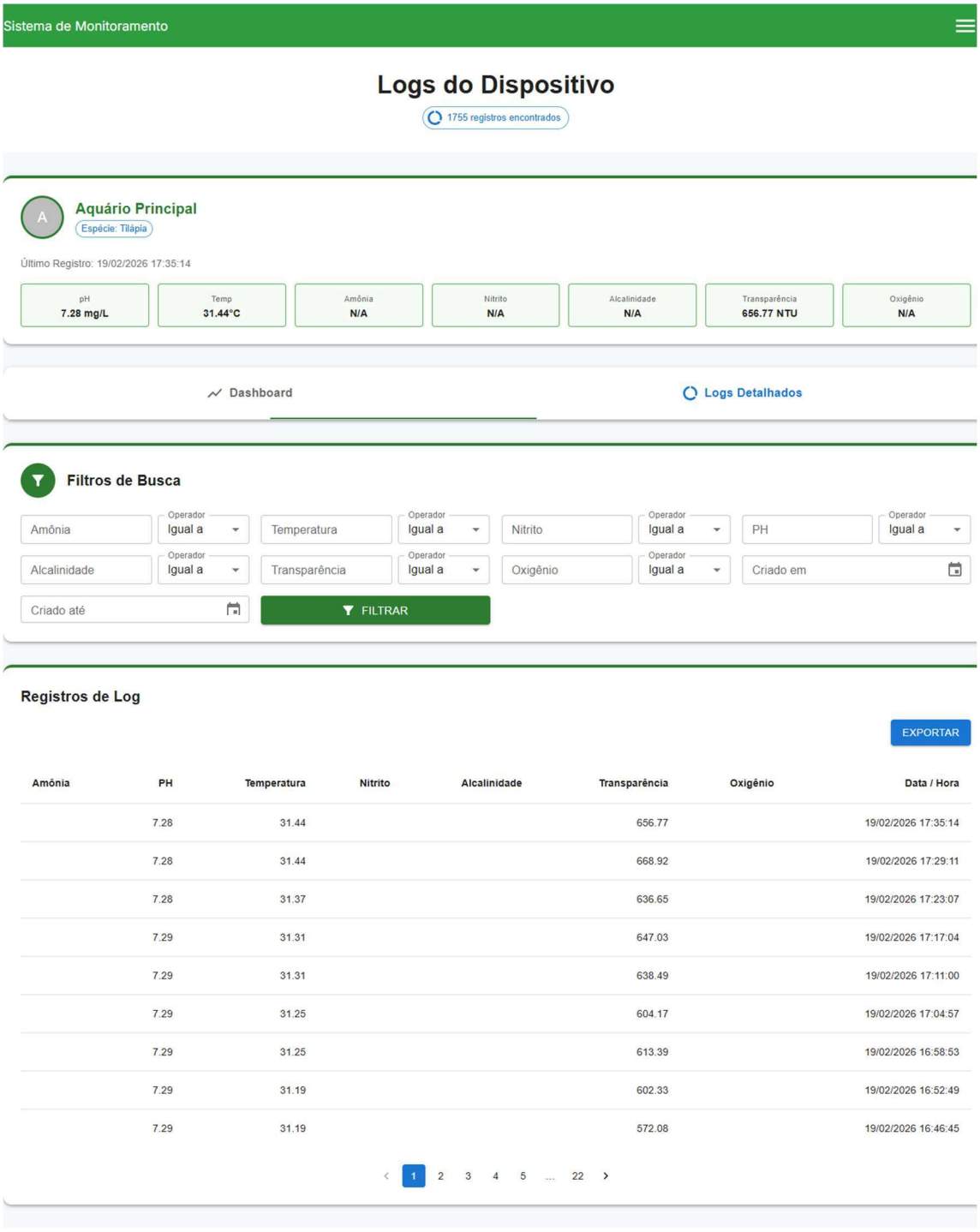
Imagem 77 - Dashboard dos dispositivos



Fonte: Autor (2026).

Já na aba de logs detalhados é possível visualizar o histórico dos registros por ordem do recente para o mais antigo. A parte de histórico foi página para facilitar a visualização (Imagem 78).

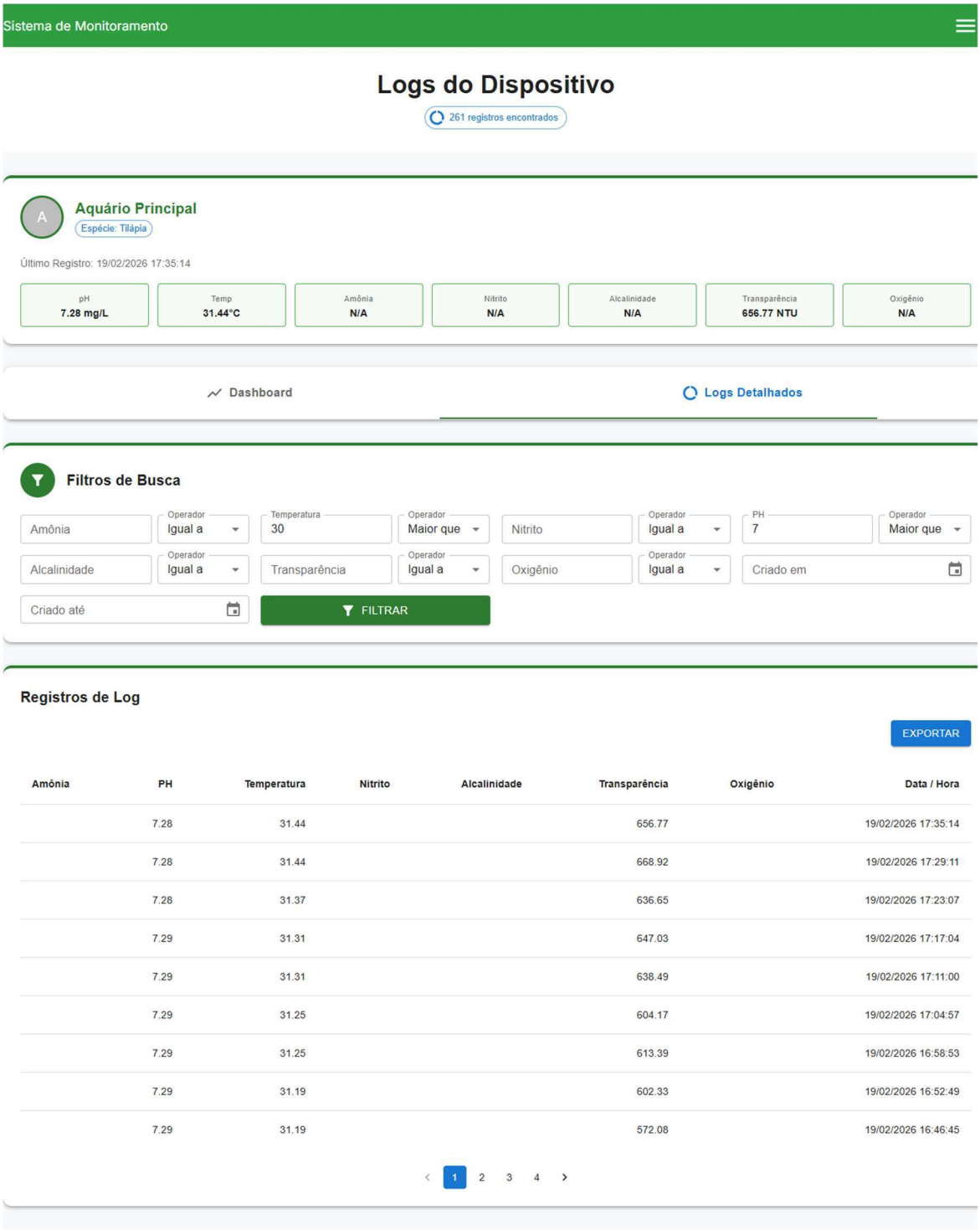
Imagem 78 - Página de log detalhado.



Fonte: Autor (2026).

Ainda nessa tela é possível realizar filtro para buscar algum dado específico, exemplo pH>8 e temperatura >30°C (Imagem 79).

Imagem 79 - Página de log detalhado com filtro.

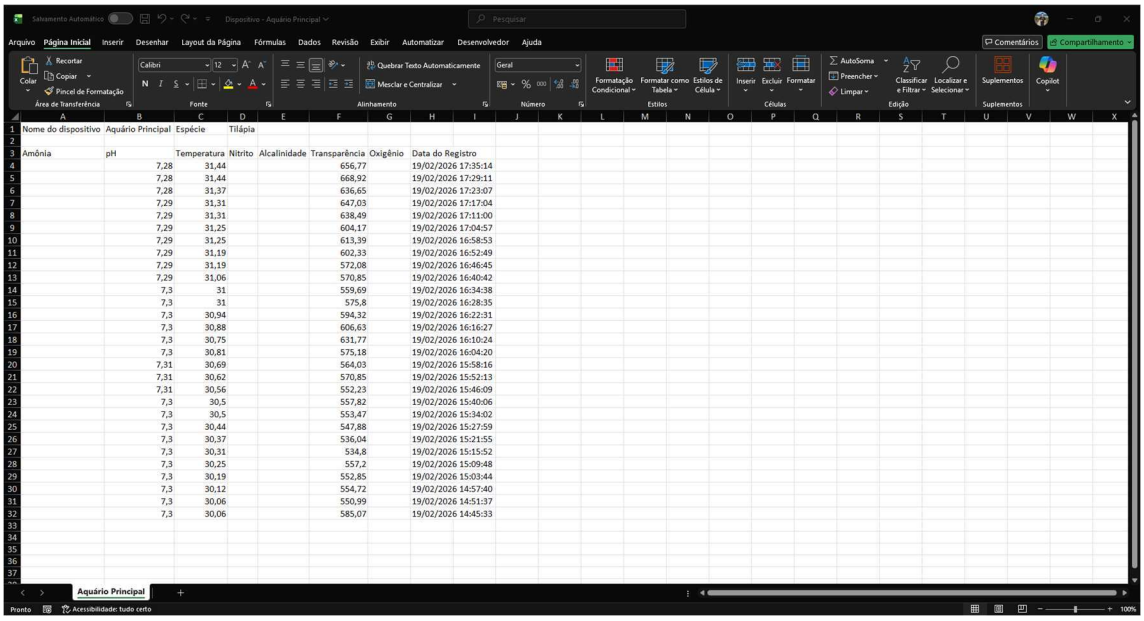


Fonte: Autor (2026).

Além disso foi implementada uma funcionalidade que permite ao usuário exportar o histórico de logs de um dispositivo no formato “.xlsx”. Este recurso possibilita análises mais aprofundadas e a criação de gráficos customizados em ferramentas externas, como o *Microsoft Excel*, atendendo a uma necessidade direta

dos utilizadores do laboratório. O arquivo criado pode ser observado na imagem 80.

Imagem 80 - Arquivo .xlsx.



Nome do dispositivo	Aquário Principal	Espécie	Tilápia					
Amônia	pH		Temperatura	Nitrito	Alcalinidade	Transparência	Oxigênio	Data do Registro
	7,28		31,44			656,77		19/02/2026 17:35:14
	7,28		31,44			668,92		19/02/2026 17:29:11
	7,28		31,37			636,65		19/02/2026 17:23:07
	7,29		31,31			647,03		19/02/2026 17:17:04
	7,29		31,31			638,49		19/02/2026 17:11:00
	7,29		31,25			604,17		19/02/2026 17:04:57
	7,29		31,25			613,39		19/02/2026 16:58:53
	7,29		31,19			602,33		19/02/2026 16:52:49
	7,29		31,19			572,08		19/02/2026 16:46:45
	7,29		31,06			570,85		19/02/2026 16:40:42
	7,3		31			559,69		19/02/2026 16:34:38
	7,3		31			575,8		19/02/2026 16:28:35
	7,3		30,94			594,32		19/02/2026 16:22:31
	7,3		30,88			606,63		19/02/2026 16:16:27
	7,3		30,75			631,77		19/02/2026 16:10:24
	7,3		30,81			575,18		19/02/2026 16:04:20
	7,31		30,69			564,03		19/02/2026 15:58:16
	7,31		30,62			570,85		19/02/2026 15:52:13
	7,31		30,56			552,23		19/02/2026 15:46:09
	7,3		30,5			557,82		19/02/2026 15:40:06
	7,3		30,5			553,47		19/02/2026 15:34:02
	7,3		30,44			547,88		19/02/2026 15:27:59
	7,3		30,37			536,04		19/02/2026 15:21:55
	7,3		30,31			534,8		19/02/2026 15:15:52
	7,3		30,25			557,2		19/02/2026 15:09:48
	7,3		30,19			552,85		19/02/2026 15:03:44
	7,3		30,12			554,72		19/02/2026 14:57:40
	7,3		30,06			550,99		19/02/2026 14:51:37
	7,3		30,06			585,07		19/02/2026 14:45:33

Fonte: Autor (2026).

A aplicação web foi projetada para ser totalmente responsiva, sendo acessível através de qualquer navegador em computadores, tablets ou celulares conectados à rede interna do IFSC.

4.5 RESULTADOS E ANÁLISES

Após a instalação do dispositivo no laboratório de piscicultura e a configuração do servidor central, realizou-se um ensaio de campo entre os dias 19 e 27 de fevereiro de 2026. O sistema foi parametrizado para realizar leituras em intervalos de uma hora. Durante a janela de testes, foram registrados 170 envios de pacotes de dados de um total esperado de aproximadamente 184, resultando em uma taxa de recepção de 92,4%.

A mediana do intervalo entre transmissões foi de 60,07 minutos, valor praticamente idêntico à frequência configurada, o que evidencia que, quando a comunicação se estabelecia, o sistema operava com regularidade e precisão temporal. Contudo, foram identificados 9 eventos de lacuna com intervalo superior a 70 minutos, sendo o mais expressivo uma interrupção de aproximadamente 7 horas ocorrida no dia 24 de fevereiro, entre as 12h33 e as 19h35. Os demais eventos corresponderam

a falhas pontuais de cerca de 2 horas cada, equivalentes a 1 ou 2 leituras perdidas consecutivamente. Essas intermitências são atribuídas à instabilidade da rede Wi-Fi do IFSC no ambiente do laboratório, cujo sinal é reconhecidamente fraco nesse espaço. Em ambientes com conectividade limitada, dispositivos *IoT* baseados em Wi-Fi tendem a apresentar exatamente esse padrão: longos períodos de operação estável intercalados com janelas de desconexão pontual. Uma solução para mitigar esse problema seria a instalação de um ponto de acesso dedicado próximo ao laboratório ou a adoção de uma interface de comunicação com maior tolerância a ambientes com sinal fraco. Um recorte dos dados gerado pelo sistema pode ser observado na imagem 81.

Imagem 81 - Arquivo gerado após o período de teste

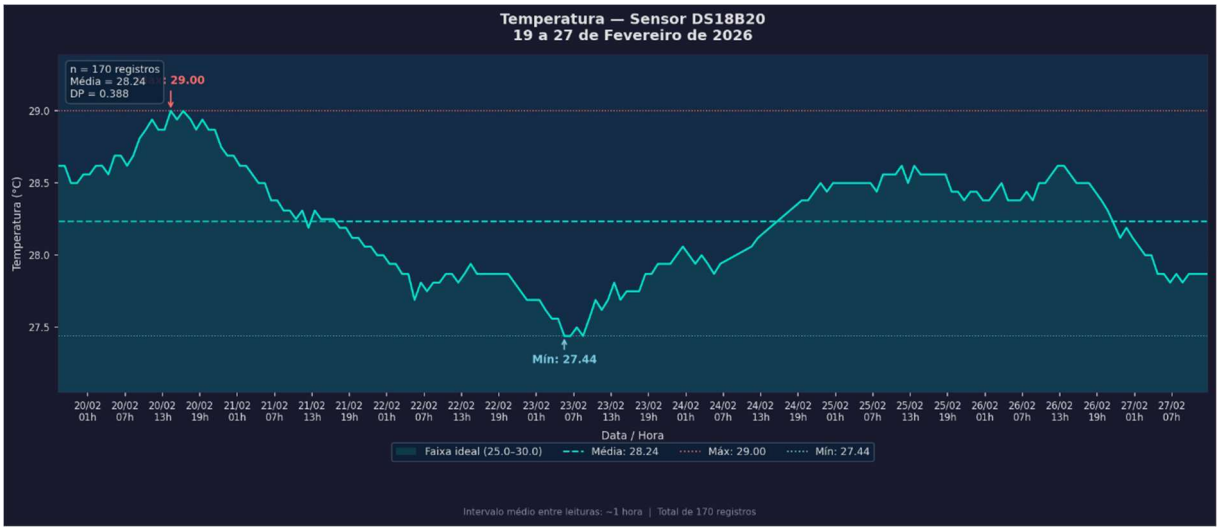
Nome do dispositivo	Aquário de Teste	Espécie	Tilápia				
Amônia	pH	Temperatu	Nitrito	Alcalinidade	Transparência	Oxigênio	Data do Registro
	7,34	27,87			284,2		27/02/2026 12:42:44
	7,34	27,87			315,54		27/02/2026 11:42:40
	7,26	27,87			582,6		27/02/2026 10:42:36
	7,3	27,87			516,05		27/02/2026 09:42:33
	7,22	27,81			590,62		27/02/2026 08:42:27
	7,26	27,87			517,93		27/02/2026 07:42:20
	7,33	27,81			577,03		27/02/2026 06:42:17
	7,2	27,87			597,4		27/02/2026 05:42:13
	7,19	27,87			638,49		27/02/2026 04:42:10
	7,23	28			679,82		27/02/2026 03:42:00
	7,3	28			836,86		27/02/2026 02:41:56
	7,25	28,06			654,94		27/02/2026 01:41:53
	7,32	28,12			1321,17		27/02/2026 00:41:46
	7,28	28,19			1117,6		26/02/2026 23:41:39
	7,35	28,12			949,33		26/02/2026 22:41:32
	7,37	28,31			693,12		26/02/2026 20:41:10
	7,38	28,38			679,22		26/02/2026 19:41:06
	7,38	28,44			703,37		26/02/2026 18:40:56
	7,23	28,5			633,6		26/02/2026 17:40:53
	7,36	28,5			729,79		26/02/2026 16:40:49
	7,4	28,5			592,47		26/02/2026 15:40:44
	7,36	28,56			679,22		26/02/2026 14:40:34
	7,38	28,62			907,57		26/02/2026 13:40:30
	7,27	28,62			511,66		26/02/2026 12:40:19
	7,3	28,56			941,35		26/02/2026 11:40:10
	7,24	28,5			904,13		26/02/2026 10:40:03
	7,26	28,5			578,89		26/02/2026 09:39:59
	7,37	28,38			661,02		26/02/2026 08:39:56
	7,38	28,44			856,68		26/02/2026 07:39:52
	7,23	28,38			776,25		26/02/2026 06:39:48
	7,38	28,38			787,49		26/02/2026 05:39:45
	7,26	28,38			798,71		26/02/2026 04:39:41
	7,33	28,5			678,01		26/02/2026 03:39:37
	7,39	28,44			770,91		26/02/2026 02:39:33
	7,23	28,38			1023,2		26/02/2026 01:39:29
	7,25	28,38			653,12		26/02/2026 00:39:25
	7,26	28,44			714,19		25/02/2026 23:39:21

Fonte: Autor (2026).

4.5.1 Análise térmica (DS18B20)

O sensor DS18B20 registrou temperaturas entre 27,44 °C e 29,00 °C, apresentando uma média aritmética de 28,24 °C com desvio padrão de 0,39 °C. Essa pequena dispersão dos dados confirma a alta estabilidade do sensor digital e sua fidelidade na captura das oscilações térmicas naturais do reservatório. Cinquenta por cento dos registros concentraram-se entre 27,87 °C e 28,56 °C, evidenciando que o sensor operou de forma extremamente consistente ao longo de todo o período monitorado. O gráfico detalhado das leituras do sensor de temperatura pode ser observado na imagem 82.

Imagem 82 - Gráfico da temperatura



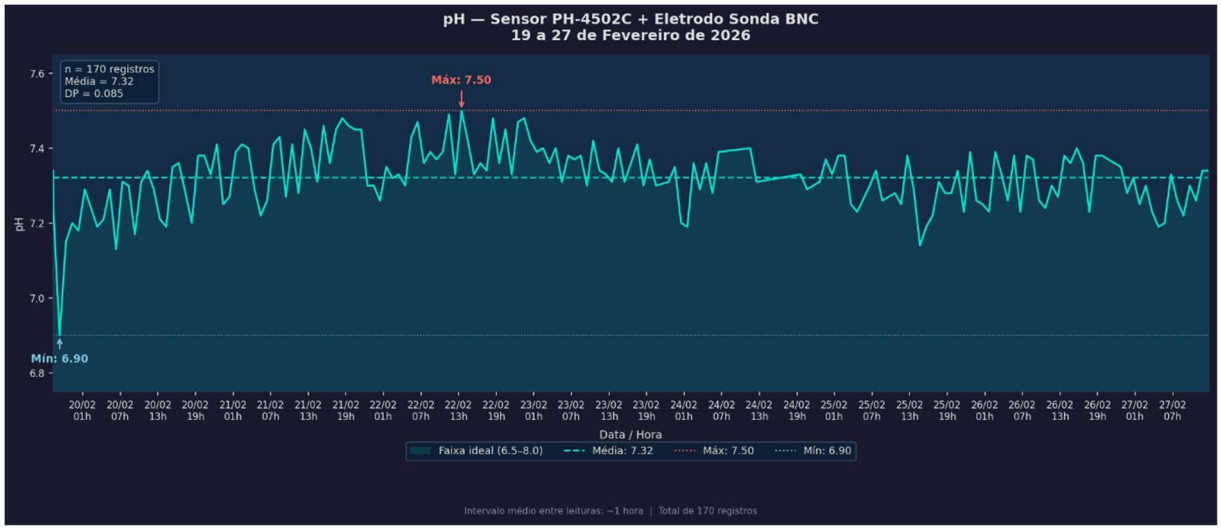
Fonte: Autor (2026).

A faixa registrada encontra-se em conformidade com o controle de climatização do laboratório, cujo sistema de ar-condicionado opera em 28,00 °C, contudo, sem a comparação simultânea com um termômetro de referência ou equipamento devidamente certificado, não há como assegurar que os valores registrados correspondam com precisão à temperatura real da água. Portanto, na ausência de verificação por outros sensores externos de precisão conhecida, os resultados obtidos devem ser interpretados prioritariamente como indicadores de tendência e estabilidade térmica do ambiente monitorado.

4.5.2 Análise do potencial hidrogeniônico (pH)

As medições de pH mantiveram-se em um intervalo entre 6,90 e 7,50, com média de 7,32 e desvio padrão de 0,085. A análise estatística indica que 50% dos registros concentraram-se na faixa entre 7,27 e 7,38, caracterizando uma leitura numericamente consistente ao longo de todo o período. O gráfico detalhado das leituras do sensor de pH pode ser observado na imagem 83.

Imagem 83 - Gráfico de pH



Fonte: Autor (2026).

Essa estabilidade é atribuída à utilização do conversor ADS1115, que, aliado à calibração linear por *Software*, permitiu filtrar variações comuns em sensores de pH de alta impedância. Os valores registrados situam-se dentro dos limites ideais para aquicultura de água doce (6 a 9), o que é compatível com o ambiente monitorado.

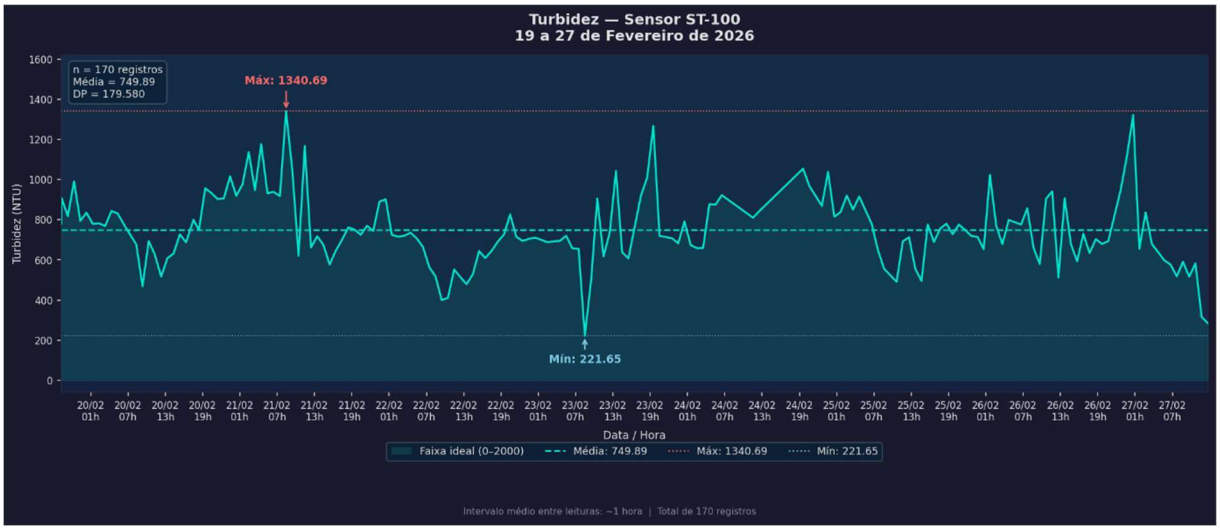
No entanto, é importante ressaltar que estabilidade numérica não equivale, necessariamente, à confiabilidade metrológica do sensor. O módulo PH-4502C é um sensor analógico sujeito a fenômenos de deriva eletroquímica ao longo do tempo. Durante o período de coleta não foi realizado nenhum procedimento de verificação de deriva, como a imersão do eletrodo em soluções tampão de referência, por exemplo, pH 4,0 e pH 7,0, antes e após a aquisição dos dados, a fim de confirmar se o sensor manteve sua calibração original e se, ao ser reinserido em uma amostra tampão, retorna ao valor calibrado. Sem essa verificação, não é possível garantir que os valores registrados correspondam ao pH real da água do aquário, pois o sensor pode apresentar um deslocamento sistemático em relação ao valor verdadeiro, sem que isso seja detectável pela análise estatística interna dos próprios dados. Portanto,

embora os dados de pH sejam úteis para indicar tendência e variação relativa ao longo do tempo, sua acurácia absoluta não pode ser assegurada sem uma rotina de verificação e recalibração periódica documentada.

4.5.3 Comportamento do sensor de turbidez

O sensor de turbidez apresentou variação significativa ao longo do período, com leituras entre 221,65 NTU e 1.340,69 NTU, média de 749,89 NTU e elevado desvio padrão de 179,58 NTU. O gráfico detalhado das leituras do sensor de Turbidez pode ser observado na imagem 84.

Imagem 84 - Gráfico de turbidez



Fonte: Autor (2026).

Diferente dos demais sensores, o modelo ST100 demonstrou instabilidade operacional no ambiente de teste. Identificou-se que o fototransistor é altamente suscetível à luz externa, o que, somado à utilização de uma fórmula polinomial genérica, resultou em flutuações acentuadas que não necessariamente refletem mudanças reais na qualidade da água. Qualquer variação nas condições de iluminação do ambiente, é capaz de introduzir leituras errôneas.

Essa vulnerabilidade fica evidente na distribuição dos valores coletados: apenas 4 leituras ficaram abaixo de 400 NTU (cenário de água muito limpa), 37 NTU leituras superaram 900 NTU (cenário de água turva), enquanto a maior parte, 129 leituras, concentrou-se entre 400 NTU e 900 NTU sem apresentar transições graduais e coerentes. Concluiu-se que, para sensores de baixo custo dessa categoria, a confiabilidade é garantida apenas em estados extremos, limpeza severa ou turbidez

acentuada, apresentando imprecisões nas faixas intermediárias de medição. O sensor é adequado para a detecção de situações críticas, como um aporte súbito de partículas em suspensão, mas não para um monitoramento quantitativo fino e contínuo da qualidade da água.

4.5.5 Gestão de dados e usabilidade

A geração automatizada do arquivo .xlsx (Imagem 81) validou a integração entre o dispositivo físico e o servidor. O sistema obteve êxito ao transformar dados brutos em informações estruturadas e legíveis, permitindo que a equipe técnica do laboratório realize análises em ferramentas externas, como o *Microsoft Excel.*

5. CONCLUSÃO

O ponto de partida deste trabalho foi um diagnóstico simples, mas com consequências práticas significativas: o laboratório de piscicultura do IFSC – Câmpus Itajaí dependia exclusivamente da presença humana para monitorar a qualidade da água de 56 aquários. Uma vez ao dia, quando havia estudantes disponíveis, os parâmetros eram aferidos manualmente e transcritos em formulários físicos. Nos fins de semana e períodos de recesso, os registros simplesmente cessavam. O ecossistema aquático, no entanto, não obedece ao calendário acadêmico.

Esse diagnóstico não é apenas operacional, ele aponta para uma contradição entre o rigor que a aquicultura exige, parâmetros como pH, temperatura, turbidez, amônia, nitrito, oxigênio dissolvido e alcalinidade devem ser monitorados de forma contínua para garantir a sobrevivência dos organismos e a natureza fragmentada do processo adotado. Erros de transcrição, leituras únicas diárias e interrupções previsíveis comprometiam não apenas a saúde dos aquários, mas a própria validade dos dados históricos como base para análise científica.

O sistema desenvolvido neste trabalho endereça esse problema a partir de uma lógica de inversão: em vez de depender da disponibilidade humana para produzir dados, a infraestrutura construída produz dados de forma autônoma e armazena-os independentemente de quem esteja presente. Isso representa uma mudança qualitativa no modelo de operação do laboratório, não apenas uma melhoria incremental de processo.

Do ponto de vista técnico, a solução integrou três camadas interdependentes dispositivo físico baseado em ESP32, backend em Node.js com persistência em PostgreSQL e interface web em Next.js, em uma arquitetura que demonstrou funcionamento coerente durante o período de testes. A taxa de recepção de 92,4% ao longo de oito dias contínuos de operação indica que o sistema é capaz de operar sem supervisão por períodos estendidos, incluindo janelas que antes eram simplesmente descartadas. Mesmo com as 14 leituras perdidas, atribuídas à instabilidade da rede Wi-Fi no ambiente do laboratório, e não a falhas de projeto, o volume de dados gerado supera em várias ordens de grandeza o que o processo manual produzia em igual período.

A análise dos dados coletados, no entanto, revela uma distinção importante

entre disponibilidade do sistema e confiabilidade das medições. Os três sensores operaram em regimes de confiabilidade distintos, e essa diferenciação merece ser compreendida como resultado científico, não apenas como limitação a superar.

O sensor de temperatura DS18B20 apresentou desvio padrão de 0,39 °C ao longo de oito dias, com 50% das leituras concentradas numa faixa de 27,87 °C e 28,56 °C, comportamento coerente com as características de um dispositivo digital pré-calibrado por laser, que elimina as principais fontes de erro analógico. Contudo, na ausência de comparação simultânea com um termômetro de referência certificado, não é possível afirmar com rigor que os valores registrados correspondem com precisão à temperatura real da água.

O sensor de pH PH-4502C apresentou consistência numérica, desvio padrão de 0,085 em torno de uma média de 7,32, mas com uma ressalva fundamental: consistência interna não prova acurácia absoluta. Um sensor analógico de alta impedância está sujeito à deriva eletroquímica ao longo do tempo, e nenhum procedimento formal de verificação com soluções tampão foi realizado antes ou após a coleta. Os dados de pH obtidos são clinicamente úteis para observar tendências e variações relativas, mas não podem ser utilizados como referência metrológica sem a implementação de uma rotina de verificação periódica documentada.

O sensor de turbidez ST100, por sua vez, evidenciou uma limitação de natureza diferente: não metrológica, mas física. A suscetibilidade do fototransistor à luz ambiente faz com que o sinal gerado reflita tanto a turbidez da água quanto as condições de iluminação do laboratório, tornando as leituras ambíguas em faixas intermediárias. Com desvio padrão de 179,47 e 76% das leituras concentradas numa zona intermediária sem transições graduais coerentes, o sensor demonstrou ser adequado apenas para detecção de eventos extremos. Esse resultado não invalida o módulo de turbidez do sistema, invalida a escolha do sensor para esse ambiente específico e orienta diretamente a recomendação de substituição por um modelo com encapsulamento óptico isolado.

Um aspecto transversal às três análises sensoriais é a ausência de medições paralelas por métodos tradicionais durante o período de testes. Não foi possível realizar registros simultâneos com termômetros manuais de precisão, eletrodos de referência ou kits colorimétricos, seja pela indisponibilidade de técnicos no laboratório durante o intervalo do ensaio. Essa lacuna impediu a execução de uma análise de

correlação entre os dados gerados pelo firmware e valores de referência externos, o que teria permitido quantificar desvios sistemáticos e validar as flutuações registradas com maior rigor. A consequência direta é que o sistema, neste estágio, cumpre plenamente seu papel como ferramenta de monitoramento contínuo, identificação de tendências, detecção de eventos anômalos, geração de histórico auditável, mas sua certificação como instrumento de medição absoluta permanece condicionada a rotinas futuras de validação cruzada. Trata-se de uma distinção importante: monitorar e medir com precisão absoluta são objetivos diferentes, e o presente trabalho entrega o primeiro com solidez.

Além do desempenho operacional e metrológico, este trabalho contribui com uma decisão de projeto que tem valor independente do contexto do IFSC: o módulo de calibração com validação estatística por coeficiente de determinação R^2. Em vez de confiar na habilidade do operador para julgar se uma calibração foi bem-sucedida, o sistema impõe uma trava objetiva, nenhuma calibração com R^2 inferior a 0,70 pode ser ativada no dispositivo. Essa abordagem transfere para o software uma responsabilidade que nos trabalhos correlatos identificados na literatura é deixada ao critério humano ou simplesmente ignorada. É uma contribuição metodológica com potencial de aplicação além deste trabalho específico.

Da mesma forma, a arquitetura *multidevice*, onde um único servidor gerencia múltiplos dispositivos independentes, cada um com seu ciclo de calibração, seus sensores habilitados e seu histórico de leituras, é uma decisão de projeto que responde diretamente à escala do laboratório e que diferencia esta solução das abordagens encontradas na literatura, majoritariamente concebidas para um único ponto de monitoramento.

Em síntese, o trabalho entregou o que se propôs: uma plataforma funcional, de baixo custo, capaz de operar de forma autônoma e contínua, com calibração validada estatisticamente, gestão centralizada de múltiplos dispositivos e exportação estruturada de dados históricos. As limitações identificadas qualidade da rede Wi-Fi, ausência de verificação formal de deriva do pH e inadequação do sensor de turbidez ao ambiente com variação de luz, não comprometem a validade da solução; elas delimitam com precisão o estágio atual de maturidade do protótipo e apontam, de forma concreta, para as melhorias necessárias antes de uma implantação em escala de produção.

O que este trabalho estabelece, acima de tudo, é uma infraestrutura sobre a qual o laboratório pode crescer: novos sensores podem ser integrados sem alteração arquitetural, novos dispositivos podem ser adicionados por provisionamento automático via interface web, e o histórico de dados, antes dependente da presença das leituras manuais, passa a ser um registro contínuo, auditável e exportável.

5.1 TRABALHOS FUTUROS

Como desdobramentos naturais deste trabalho, identificam-se as seguintes melhorias e extensões para implementação futura:

- A integração dos sensores previstos no projeto, mas não implementados nesta etapa, como os de amônia, nitrito, alcalinidade e oxigênio dissolvido. Paralelamente, recomenda-se a avaliação de sensores de turbidez alternativos ao ST100, de modo a obter leituras com maior linearidade e menor variação entre unidades.
- Todos os sensores do sistema deverão ser submetidos a uma validação estatística formal, incluindo testes de deriva, repetibilidade e verificação com amostras de referência, a fim de atestar sua confiabilidade metrológica para uso em ambiente de produção.
- No aspecto do hardware, a confecção de uma placa de circuito impresso (PCB) dedicada é indicada para substituir a atual, conferindo maior robustez mecânica, redução de ruído elétrico e facilidade de replicação do dispositivo.
- A incorporação de uma bateria ao sistema, combinada com um cartão SD para armazenamento local temporário, garantiria a continuidade da coleta de dados em situações de falta de energia elétrica. Nesse cenário, os registros acumulados localmente seriam sincronizados automaticamente com o servidor assim que a conectividade fosse restabelecida, eliminando lacunas de leituras.
- A implementação de um sistema de alertas automáticos que notifique o responsável sempre que algum parâmetro monitorado ultrapassar os limites pré-estabelecidos para a espécie em cultivo.
- Integrar o sistema com o banco de dados interno do IFSC — campus Itajaí, que está em fase final de desenvolvimento.
- Por fim, recomenda-se a melhoria da infraestrutura de rede Wi-Fi no laboratório de piscicultura. Embora o sistema tenha operado de forma funcional durante os

testes, verificou-se que a intensidade do sinal é reduzida em determinados pontos do ambiente, fator possivelmente atribuído à presença dos próprios aquários e à densidade dos materiais utilizados na estrutura do laboratório. A instalação de pontos de acesso adicionais contribuiria para a estabilidade da transmissão em todas as posições do laboratório, reduzindo as lacunas de leitura observadas.

REFERÊNCIAS

ABREU, G. S. **Monitoramento e controle de sistema de aquicultura**. 2023. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Universidade Federal de Uberlândia, Uberlândia, 2023. Disponível em: https://repositorio.ufu.br/bitstream/123456789/46914/1/MonitoramentoControleSistema.pdf. Acesso em: 16 fev. 2026.

AISLAN. **Monitorando o pH de líquidos com ESP32 e Sensor pH-4502C**. Blog Eletrogate, 2025. Disponível em: https://blog.eletrogate.com/monitorando-o-ph-de-liquidos-com-esp32-e-sensor-ph-4502c/. Acesso em: 21 fev. 2025.

ANDRADE, Rildo José Vasconcelos de et al. Aplicação da internet das coisas (IoT) no monitoramento automatizado de tanques de piscicultura no estado de Pernambuco. **Observatorio de la Economía Latinoamericana**, Curitiba, v. 23, n. 2, p. 1-20, 2025. Disponível em: https://ojs.observatoriolatinoamericano.com/ojs/index.php/olel/article/view/9877. Acesso em: 24 fev. 2026.

ASSOCIAÇÃO BRASILEIRA DA PISCICULTURA (PEIXE BR). **Anuário Peixe BR da Piscicultura 2025**. São Paulo: Peixe BR, 2025. Disponível em: https://www.peixebr.com.br/anuario-2025/. Acesso em: 22 fev. 2026.

CRISTIANO, Fausto. **Sistema autossuficiente de monitoramento remoto de qualidade da água em açudes para piscicultura**: projeto de TCC. São José-SC, 2024. 32 f. Disponível em: https://wiki.sj.ifsc.edu.br/images/c/cc/Projeto_de_TCC_FAUSTO_CRISTIANO.pdf. Acesso em: 22 fev. 2026.

DEVORE, J. L. **Probabilidade e Estatística para Engenharia e Ciências**. 9. ed. São Paulo: Cengage Learning, 2018.

DFROBOT. **Turbidity sensor SKU: SEN0189**. Wiki DFRobot, 2024. Disponível em: https://wiki.dfrobot.com/Turbidity_sensor_SKU__SEN0189. Acesso em: 22 fev. 2026.

DINIZ, Paulo S. R.; SILVA, Eduardo A. B. da; NETTO, Sergio L. **Processamento digital de sinais**: projeto e análise de sistemas. 2. ed. Porto Alegre: Bookman, 2014.

EMBRAPA. **CIM**: Centro de Inteligência e Mercado em Aquicultura. 2025. Disponível em: https://www.embrapa.br/cim-centro-de-inteligencia-e-mercado-em-aquicultura. Acesso em: 22 fev. 2026.

EXPRESS. **Express**: Fast, unopinionated, minimalist web framework for Node.js. 2024. Disponível em: https://expressjs.com/. Acesso em: 21 fev. 2026.

GARCÍA, J. et al. Sistemas de monitoramento inteligente aplicados à aquicultura: uma perspectiva tecnológica. **Cuadernos de Educación y Desarrollo**, v. 15, n. 6, 2023. Disponível em: https://ojs.cuadernoseducacion.com/ojs/index.php/ced/article/view/10307/6794. Acesso em: 16 fev. 2026.

GRZESZCZAK, K. **Desenvolvimento de um sistema de monitoramento de parâmetros físico-químicos**. 2025. Trabalho de Conclusão de Curso (Graduação) – Universidade Estadual do Rio Grande do Sul, [S. l.], 2025. Disponível em: https://repositorio.uergs.edu.br/xmlui/bitstream/handle/123456789/3546/_tcc_kenny.pdf. Acesso em: 22 fev. 2026.

JACOMINE, L. et al. **Monitoramento e controle de variáveis em sistemas de aquicultura**. 2025. Trabalho de Conclusão de Curso – Universidade Federal da Integração Latino-Americana, Foz do Iguaçu, 2025. Disponível em: https://dspace.unila.edu.br/items/3ff788f9-a571-4912-b687-dcd4ca4bc2e2. Acesso em: 22 fev. 2026.

LOPES, A. S. **Manual de Piscicultura**. Florianópolis: UDESC, 2018. Disponível em: https://www.udesc.br/arquivos/ceo/id_cpmenu/1043/222_15427278707761_1043.pdf. Acesso em: 22 fev. 2026.

MAKERHERO. **Como utilizar o Sensor de Turbidez Arduino**. 2026. Disponível em: https://www.makerhero.com/blog/sensor-de-turbidez-arduino/. Acesso em: 21 fev. 2026.

MAKERHERO. **ESP32**. 2026. Disponível em: https://www.makerhero.com/categoria/wireless-e-iot/esp32/. Acesso em: 21 fev. 2026.

MAKERHERO. **Sensor de Temperatura DS18B20 à prova d'água**. 2023. Disponível em: https://www.makerhero.com/produto/sensor-de-temperatura-ds18b20-a-prova-dagua/. Acesso em: 21 fev. 2026.

MATOS, J. G.; SENA, Y. de. **Monitoramento de variáveis físico-químicas em sistemas de aquicultura**. 2025. Trabalho de Conclusão de Curso (Graduação em Engenharia) – Universidade de Brasília, Brasília, 2025. Disponível em: https://bdm.unb.br/bitstream/10483/42588/1/2025_JoaoGabrieMatos_YanDeSena_tcc.pdf. Acesso em: 22 fev. 2026.

MAXIM INTEGRATED. **DS18B20**: Programmable Resolution 1-Wire Digital Thermostat. Rev. 6. San Jose, CA: Maxim Integrated Products, 2018. Disponível em: https://www.analog.com/media/en/technical-documentation/data-sheets/DS18B20.pdf. Acesso em: 22 fev. 2026.

MICROSOFT. **TypeScript**: JavaScript with syntax for types. 2026. Disponível em: https://www.typescriptlang.org/. Acesso em: 21 fev. 2026.

MORO, G. V. et al. **Qualidade da água na piscicultura**. Brasília, DF: Embrapa, 2021. Disponível em: https://www.infoteca.cnptia.embrapa.br/infoteca/bitstream/doc/1083545/1/cap.5.pdf. Acesso em: 22 fev. 2026.

MUI. **MUI Core**: Ready-to-use foundational React components. 2024. Disponível em: https://mui.com/. Acesso em: 21 fev. 2026.

NODE.JS FOUNDATION. **Node.js**: JavaScript runtime built on Chrome's V8 JavaScript engine. 2024. Disponível em: https://nodejs.org/. Acesso em: 21 fev. 2026.

OLIVEIRA, G. A. de. **Sistema de monitoramento e controle automatizado para aquicultura**. 2025. Trabalho de Conclusão de Curso – Instituto Federal da Paraíba, [S. l.], 2025. Disponível em: https://repositorio.ifpb.edu.br/handle/177683/4386. Acesso em: 22 fev. 2026.

OLIVEIRA, Samuel Santos de. **Monitoramento e controle de um sistema de aquicultura doméstico**. 2025. 33 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Eletrônica e de Telecomunicações) – Universidade Federal de Uberlândia, Patos de Minas, 01 set. 2025. Disponível em: https://repositorio.ufu.br/bitstream/123456789/46914/1/MonitoramentoControleSistema.pdf. Acesso em: 22 fev. 2026.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL**: The World's Most Advanced Open Source Relational Database. 2026. Disponível em: https://www.postgresql.org/. Acesso em: 21 fev. 2026.

PRISMA DATA. **Prisma**: Next-generation ORM for Node.js and TypeScript. 2026. Disponível em: https://www.prisma.io/. Acesso em: 21 fev. 2026.

REGRESSION.JS. **regression-js**: Curve Fitting in JavaScript. Versão 2.0.1. Disponível em: https://github.com/Tom-Alexander/regression-js. Acesso em: 27 fev. 2026.

SANTOS, F. C. **Desenvolvimento de um sistema de monitoramento de variáveis ambientais utilizando IoT**. 2018. Trabalho de Conclusão de Curso (Técnico em Telecomunicações) – Instituto Federal de Santa Catarina, São José, 2018. Disponível em: https://wiki.sj.ifsc.edu.br/images/c/cc/Projeto_de_TCC_FAUSTO_CRISTIANO.pdf. Acesso em: 16 fev. 2026.

SCHNEIDER, Yan. TCC. Repositório TCC Yan Schneider. Disponível em: https://github.com/yanschneider98/TCC. Acesso em: 16 mar. 2026.

SERVIÇO NACIONAL DE APRENDIZAGEM RURAL (SENAR). **Piscicultura**: manejo e produção. Brasília, DF: SENAR, 2017. Disponível em: https://www.cnabrasil.org.br/assets/arquivos/195-PISCICULTURA.pdf. Acesso em: 22 fev. 2026.

SKOOG, Douglas A. et al. **Fundamentos de Química Analítica**. 9. ed. São Paulo: Cengage Learning, 2014.

STRAUB. **Sensor de Turbidez ST100 para Arduino e ESP32.** Blog UsinaInfo, 2026. Disponível em: https://www.usinainfo.com.br/blog/sensor-de-turbidez-projeto-de-leitura-da-qualidade-da-agua/. Acesso em: 21 fev. 2026.

VON SPERLING, Marcos. **Introdução à qualidade das águas e ao tratamento de esgotos.** 4. ed. Belo Horizonte: Editora UFMG, 2014. 452 p.