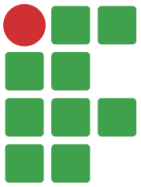


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS CAMPINA GRANDE
CURSO SUPERIOR BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

GABRIEL ALBINO DE OLIVEIRA

SISTEMA IoT PARA AUTOMAÇÃO E MONITORAMENTO INTELIGENTE PARA AQUÁRIOS

Campina Grande
2025

Gabriel Albino de Oliveira

SISTEMA IoT PARA AUTOMAÇÃO E MONITORAMENTO INTELIGENTE PARA AQUÁRIOS

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia de Computação, do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba Campus Campina Grande, em cumprimento às exigências parciais para a obtenção do título de Bacharel em Engenharia de Computação.

Instituto Federal da Paraíba

Orientador(a): George Sobral Silveira

Campina Grande
2025

Catalogação na fonte:
Ficha catalográfica elaborada por Gustavo César Nogueira da Costa - CRB 15/479

O48s Oliveira, Gabriel Albino de
Sistema IoT para automação e monitoramento inteligente para aquários / Gabriel Albino de Oliveira. - Campina Grande, 2025.
45 f. : il.

Trabalho de Conclusão de Curso (Curso Superior de Engenharia de Computação) - Instituto Federal da Paraíba, 2025.
Orientador: Prof. Me. George Sobral Silveira.

1. Engenharia de computação 2. Internet das Coisas - IoT 3. Automação de aquários 4. Aquário - Monitoramento remoto I. Silveira, George Sobral II.Título.

CDU 004.3

Gabriel Albino de Oliveira

SISTEMA IoT PARA AUTOMAÇÃO E MONITORAMENTO INTELIGENTE PARA AQUÁRIOS

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia de Computação, do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba Campus Campina Grande, em cumprimento às exigências parciais para a obtenção do título de Bacharel em Engenharia de Computação.

Trabalho aprovado. Campina Grande, 25 de fevereiro de 2025:

George Sobral Silveira
Orientador

Fagner de Araújo Pereira
Membro da Banca

Igor Barbosa da Costa
Membro da Banca

Campina Grande
25 de fevereiro de 2025

RESUMO

O aquarismo é uma prática que combina hobby, decoração e estudo de ecossistemas aquáticos, proporcionando benefícios como relaxamento e aprendizado sobre a natureza. No entanto, manter um aquário saudável exige atenção constante a fatores como temperatura, iluminação e alimentação, que são essenciais para o bem-estar dos organismos aquáticos. Essas tarefas, quando realizadas manualmente, demandam tempo, conhecimento técnico e dedicação, além de estarem sujeitas a falhas humanas, como esquecimento ou desregulação dos parâmetros. Esses desafios tornam a automação uma solução valiosa para simplificar a manutenção do aquário.

Este trabalho apresenta o desenvolvimento de um sistema automatizado para aquários, projetado para otimizar o controle de alimentação, iluminação e temperatura, reduzindo a dependência de intervenções manuais. Utilizando o microcontrolador ESP32, sensores como o DS18B20 para medição térmica, atuadores como relés e servo motores SG90, e protocolos de comunicação MQTT e HTTP, o sistema integra hardware e software para operar de forma autônoma e remota. A solução inclui uma interface gráfica desenvolvida em Next.js, que permite aos usuários configurar agendamentos e monitorar parâmetros em tempo real. Testes em aquários de 15 e 80 litros demonstraram a funcionalidade do sistema, no monitoramento térmico e capacidade de operação offline, garantindo continuidade nas tarefas essenciais mesmo sem conexão à internet.

ABSTRACT

Aquarium keeping is a practice that combines hobby, decoration, and the study of aquatic ecosystems, offering benefits such as relaxation and learning about nature. However, maintaining a healthy aquarium requires constant attention to factors like temperature, lighting, and feeding, which are essential for the well-being of aquatic organisms. These tasks, when performed manually, demand time, technical knowledge, and dedication, and are prone to human errors, such as forgetfulness or parameter misregulation. These challenges make automation a valuable solution to simplify aquarium maintenance.

This work presents the development of an automated system for aquariums, designed to optimize the control of feeding, lighting, and temperature, reducing the reliance on manual interventions. Using the ESP32 microcontroller, sensors such as the DS18B20 for thermal measurement, actuators like relays and SG90 servo motors, and communication protocols such as MQTT and HTTP, the system integrates hardware and software to operate autonomously and remotely. The solution includes a graphical interface developed in Next.js, allowing users to configure schedules and monitor parameters in real-time. Tests conducted on 15 and 80-liter aquariums demonstrated the system's functionality in thermal monitoring and offline operation capability, ensuring continuity in essential tasks even without an internet connection.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão a todas as pessoas que me apoiaram durante o desenvolvimento deste TCC.

Primeiramente, agradeço imensamente à minha namorada Ana Luiza, que esteve ao meu lado em todo o período, me apoiando, incentivando e sendo uma fonte constante de motivação.

Aos meus pais, Ansélio e Amarilis, e aos meus irmãos, Miguel e Anna, agradeço pelo apoio e companheirismo ao longo de todo esse processo. Sempre estiveram ao meu lado, oferecendo a base necessária para seguir em frente.

Também quero agradecer ao meu primo Felipe Almeida Albino, "Caçador de tetras-negros", pela valiosa ajuda na busca de referências. Sua parceria nas madrugadas foi essencial para o sucesso dessa parte do projeto.

Agradeço também ao meu orientador, George Silveira, pela orientação e paciência. Seus conselhos foram fundamentais para o desenvolvimento deste trabalho.

Aos meus amigos da faculdade, Alisson, Ana Carolina, Amanda e Sara, agradeço pela amizade e pelos momentos de colaboração que tornaram essa fase mais leve e divertida.

As pessoas que não tiveram o nome citado aqui, mas que de alguma forma também contribuíram para esse trabalho, meu muito obrigado.

SUMÁRIO

LISTA DE ABREVIATURA E SIGLAS

IoT - *Internet of Things*.

ESP32 - Microcontrolador.

MQTT - *Message Queuing Telemetry Transport*.

GPIO - *General Purpose Input/Output*.

PWM - *Pulse Width Modulation*.

NPM - *Node Package Manager*.

API - *Application programming interface*.

HTTP - *Hypertext Transfer Protocol*.

PLA - *Polylactic acid*.

DS18B20 - Sensor de temperatura digital.

E/S - Entrada/Saída.

KB - *Kilobyte*.

RAM - *Random access memory*.

V - Volts.

mA - Miliampere.

A - Ampere.

DC - *Direct Current*.

AC - *Alternating Current*.

I2C - *Inter-Integrated Circuit*.

NTP - *Network Time Protocol*.

ACID - Atomicidade, Consistência, Isolamento e Durabilidade

RTC - *real-time clock*

LED - *Light Emitting Diodes*

TLS - *Transport Layer Security*

AP - *Access point*

JWT - *JSON Web Token*

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

O aquarismo é uma prática amplamente difundida em ambientes domésticos e comerciais, oferecendo um habitat controlado para diversas espécies aquáticas. Contudo, a manutenção de aquários exige atenção constante a fatores como alimentação, iluminação e controle de temperatura, que são fundamentais para garantir um ambiente saudável e adequado aos organismos aquáticos (LENGER, 1993). Esses cuidados, embora necessários, podem ser desafiadores para muitos aquaristas, especialmente devido à rotina moderna, que muitas vezes dificulta o acompanhamento regular dessas tarefas.

A falta de tempo para realizar manutenções adequadas pode levar a desequilíbrios no ambiente aquático, comprometendo a saúde e o bem-estar dos peixes e de outras espécies. Variações térmicas podem induzir estresse fisiológico, reduzir a imunidade e aumentar a susceptibilidade a doenças, por isso é muito importante o monitoramento (LENGER, 1993).

Para minimizar esses problemas, tecnologias têm sido exploradas como uma forma de otimizar o manejo de aquários. Entre essas tecnologias, os dispositivos baseados em IoT (*Internet of Things*) se destacam como ferramentas que permitem monitorar e controlar remotamente as condições do aquário. Os microcontroladores, como o ESP32, têm ganhado espaço devido ao seu custo acessível e à capacidade de integrar sensores e atuadores de maneira eficiente (KHAN, 2019). Eles oferecem a possibilidade de automatizar diversas funções essenciais, tornando o manejo mais prático e reduzindo a dependência de ações manuais.

Este trabalho propõe o desenvolvimento de um sistema automatizado para aquários, capaz de integrar sensores e atuadores controlados por um microcontrolador ESP32. O sistema tem como objetivo principal automatizar tarefas essenciais, como a alimentação, iluminação e temperatura.

1.1 SOLUÇÕES EXISTENTES E DIFERENÇAS

Nos últimos anos, o avanço da Internet das Coisas (IoT) e da automação residencial trouxe soluções inovadoras para diversas áreas, incluindo o aquarismo. No mercado, empresas como a Quili Aquários (Quili Aquários, 2025) oferecem

aquários automatizados que integram controle de temperatura, iluminação e alimentação. Entretanto, essas soluções são vendidas como aquários completos, exigindo que o usuário compre um novo aquário para usufruir da automação.

O diferencial do sistema proposto neste trabalho está na sua versatilidade. Em vez de exigir a substituição do aquário, ele é um dispositivo modular e independente, que pode ser instalado em qualquer aquário já existente. Isso permite que aquaristas modernizem seus sistemas sem precisar trocar todo o equipamento, tornando a automação mais acessível.

1.2 OBJETIVO GERAL

Desenvolver um sistema automatizado para aquários, integrando hardware e software, capaz de controlar alimentação, iluminação e temperatura de forma autônoma e remota, reduzindo a necessidade de intervenção manual e garantindo estabilidade ambiental para organismos aquáticos.

1.3 OBJETIVOS ESPECÍFICOS

- Implementar o controle automatizado de alimentação utilizando servo motor, com programação de horários e quantidades precisas de ração.
- Projetar um sistema de iluminação inteligente baseado em relé, ajustável conforme ciclos definidos pelo usuário.
- Integrar um sensor de temperatura para monitoramento térmico contínuo.
- Desenvolver uma interface web responsiva para configuração remota de parâmetros e visualização de dados em tempo real.
- Avaliar o desempenho e a precisão do sistema em cenários reais, analisando sua eficiência na manutenção dos parâmetros do aquário.

2 FUNDAMENTAÇÃO TEÓRICA

O aquarismo, prática que proporciona um habitat controlado para espécies aquáticas, exige cuidados constantes com alimentação, iluminação e temperatura. Contudo, a rotina pode dificultar o acompanhamento necessário, comprometendo a qualidade do ambiente e a saúde dos organismos.

Com o avanço da tecnologia, dispositivos de IoT (*Internet of Things*) oferecem uma nova dimensão de controle e monitoramento remoto. Soluções baseadas em IoT possibilitam integrar sensores, atuadores e interfaces de controle em tempo real, transformando a maneira como interagimos com nossos ambientes.

Os microcontroladores têm desempenhado um papel fundamental no avanço da automação. Estes dispositivos possibilitam a conexão e o processamento de dados em tempo real, permitindo controle preciso sobre atuadores e sensores. Microcontroladores são vantajosos devido à sua capacidade de operar de forma independente e ao baixo consumo de energia, possibilitando seu uso em uma ampla gama de aplicações, especialmente na automação (UNESP, 2025).

A aplicação de microcontroladores na automação de aquários permite a integração de sensores e atuadores que podem ser controlados remotamente. Com essa tecnologia, o usuário pode definir horários para alimentação e iluminação além do monitoramento da temperatura, proporcionando um ambiente mais estável e seguro. O ESP32, especificamente, destaca-se por ser de baixo custo e oferece conectividade Wi-Fi, características essenciais para sistemas de automação e monitoramento que requerem comunicação com uma interface remota.

Para a construção de um sistema automatizado de monitoramento de aquário, são empregadas diversas tecnologias. Para viabilizar essa automação, microcontroladores como o ESP32 são utilizados, pois permitem a integração eficiente de sensores e atuadores, garantindo um controle do aquário remotamente.

2.1 ESP32 e Arduino

O ESP32 e o Arduino são amplamente utilizados para projetos de automação e IoT. No entanto, o ESP32 se destaca por oferecer conectividade Wi-Fi embutida, maior capacidade de processamento e eficiência energética, tornando-o mais adequado para sistemas que requerem comunicação remota e processamento simultâneo de múltiplas tarefas.

Tabela 1 - Comparação entre ESP32 e Arduino

Característica	**ESP32**	**Arduino (Uno)**
Arquitetura	Dual-core, 32 bits	Single-core, 8 bits
Clock	Até 240 MHz	16 MHz
Memória RAM	512 KB	2 KB
Armazenamento	4 MB Flash	32 KB Flash
Conectividade	Wi-Fi e Bluetooth embutidos	Nenhuma conectividade nativa
Pinos de I/O	25 GPIOs	14 GPIOs
Tensão de operação	3,3V	5V

Como mostrado na tabela 1, o ESP32 possui mais funcionalidades, como conectividade sem fio nativa, tornando-o uma escolha ideal para aplicações que necessitam de acesso remoto e processamento paralelo.

2.2 Microcontrolador ESP32

O ESP32 é um microcontrolador projetado para aplicações de IoT, que opera com uma tensão de funcionamento de 3,3V e suporta comunicação Wi-Fi e Bluetooth, facilitando sua integração em projetos que exigem conectividade com a Internet, além do baixo consumo de energia, graças aos modos de economia que permitem sua operação contínua em sistemas de baixa manutenção, sendo ideal para monitoramento constante (ESPRESSIF SYSTEMS, 2023). Além disso, o ESP32 possui 25 pinos de entrada e saída, que facilitam a interface com sensores e atuadores diversos, possibilitando a construção de sistemas complexos e flexíveis. Sua conectividade Wi-Fi e Bluetooth permite comunicação com redes locais e dispositivos móveis, favorecendo o controle remoto e o monitoramento de dados em tempo real.

Outro ponto relevante é a presença de dois núcleos de processamento com uma frequência de operação de até 240MHz (SMARTKITS, 2024). Esse clock elevado, em combinação com os dois núcleos, possibilita ao ESP32 executar múltiplas tarefas simultaneamente, como a leitura de sensores, o controle de atuadores e a comunicação Wi-Fi, eliminando a necessidade de módulos adicionais e possibilitando a criação de sistemas inteligentes e conectados. Além disso, a capacidade de executar tarefas paralelas com seus dois núcleos permitem a implementação de soluções mais responsivas, essenciais para o monitoramento e controle em tempo real.

A sua memória extra é um diferencial do arduino, já que permite armazenar informações importantes para o sistema, como páginas da interface web e configurações do aquário, garantindo uma melhor experiência de uso e maior autonomia do dispositivo.

A temperatura é um dos fatores cruciais para o equilíbrio de aquários, pois pequenas variações podem afetar significativamente a saúde dos animais (LENGER, 1993). Para garantir um ambiente adequado, é essencial monitorar

continuamente, utilizando sensores que podem ser facilmente integrados ao ESP32.

2.3 Sensor DS18B20

O sensor DS18B20 é um sensor de temperatura digital amplamente utilizado em projetos de monitoramento devido às suas características técnicas e praticidade. Com uma interface de comunicação de um fio (*1-Wire*), ele permite a conexão de múltiplos sensores em um único barramento, simplificando o projeto de sistemas mais complexos. Sua ampla faixa de medição, que varia de -55 °C a 125 °C, o torna aplicável em uma variedade de ambientes. Além disso, o sensor oferece precisão, com uma margem de erro de ±0,5 °C na faixa de -10 °C a +85 °C (ANALOG Devices, 2019), sendo ideal para aplicações que requerem leituras precisas, como no contexto dos aquários, em que a faixa mais relevante é aquela que atende às necessidades específicas das diferentes espécies de peixes. Cada espécie de peixe possui uma faixa de temperatura ideal para seu desenvolvimento e bem-estar. Por exemplo, peixes tropicais, como os tetras-negro (Gymnocorymbus ternetzi), geralmente preferem temperaturas entre 22 °C e 26 °C (CITESEERX, 2008).

Manter a temperatura dentro desses intervalos é crucial para a saúde dos peixes, pois variações fora desses limites podem causar estresse, doenças ou até mesmo a morte (LENGER, 1993). Outra vantagem do DS18B20 é sua comunicação digital simplificada, que elimina a necessidade de conversores analógicos, facilitando a integração direta com microcontroladores (ANALOG Devices, 2019).

Para aplicações que envolvem líquidos, como no monitoramento da temperatura em aquários, existe uma versão encapsulada à prova d’água do DS18B20. Nesse modelo, o sensor é protegido por uma capa de aço inoxidável e selado com resina, garantindo resistência à umidade e permitindo que ele seja submerso sem comprometer seu desempenho ou integridade. Essa proteção é essencial para ambientes aquáticos, onde o sensor pode estar em contato direto com a água, assegurando durabilidade e confiabilidade nas medições.

Figura 1 - Versão encapsulada do sensor DS18B20.

Fonte: Baseada em (USINAINFO, 2025).

2.4 Relé

Além do controle da temperatura, outro aspecto fundamental no aquarismo é o controle da iluminação. A iluminação desempenha um papel essencial no ecossistema do aquário, influenciando não apenas a estética, mas também o comportamento dos peixes e o crescimento de plantas aquáticas. Muitas espécies de peixes e plantas dependem de ciclos de luz e escuridão bem definidos para manter seus ritmos biológicos naturais (LENGER, 1993). Assim, para controlar uma carga com consumo elevado pelo microcontrolador, é necessário acoplar um dispositivo que possa oferecer ganho de corrente e tensão, como por exemplo relé.

O relé é um componente eletromecânico essencial em sistemas de controle, permitindo o acionamento de dispositivos de alta potência a partir de sinais de baixa potência, tornando-se fundamental em automação (HANDSONTEC, 2024). Ele oferece isolamento elétrico, garantindo segurança ao controlar dispositivos de alta potência e protegendo os circuitos de controle contra sobrecargas.

Neste projeto o relé é utilizado para o controle da iluminação do aquário, pois as lâmpadas utilizadas operam em 220V AC (*Alternating current*) e consomem aproximadamente 30W. O ESP32 opera com níveis lógicos de 3.3V DC (*Direct current*) e correntes máximas de 40mA por pino, o que é insuficiente para acionar

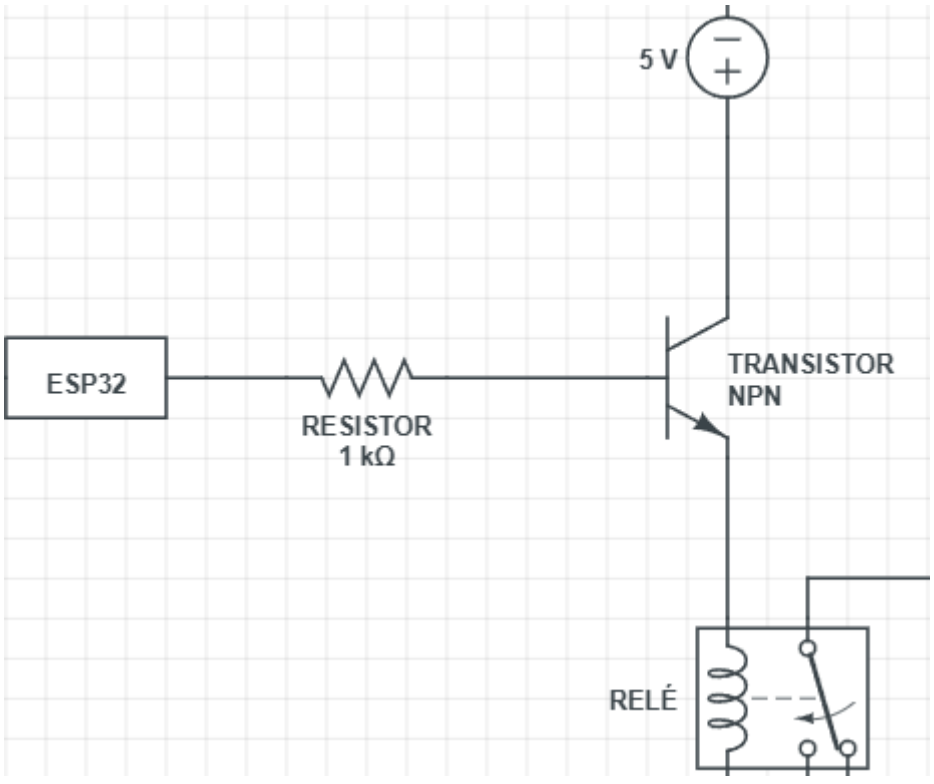
diretamente as lâmpadas.

Tabela 2 - Características do relé

Tensão de acionamento	5V DC
Corrente de acionamento	10mA a 15mA
Tensão suportada nos contatos	250V AC ou 30V DC
Corrente máxima suportada	10A

Através desse relé, o ESP32 pode ligar e desligar automaticamente as lâmpadas do aquário, garantindo que o ciclo de luz e escuridão seja mantido. No entanto, para proteger o microcontrolador de sobrecargas elétricas e evitar danos ao circuito, foi utilizado um transistor como intermediário entre o ESP32 e o relé, garantindo um acionamento seguro e eficiente.

Figura 2 - Circuito elétrico do relé



Fonte: Autoria própria

2.5 Servo motor

A automação em aquários não se limita ao controle de iluminação ou monitoramento da temperatura, mas também abrange aspectos críticos como a alimentação dos peixes, que demanda precisão e regularidade para garantir a saúde dos organismos aquáticos, neste caso, o uso atuadores como os servo motores auxiliam nessa tarefa

O servo motor SG90 é um atuador rotativo que permite o controle preciso da posição do seu eixo. através de PWM (*Pulse Width Modulation*) e é amplamente utilizado em eletrônica e robótica devido ao seu baixo custo e versatilidade (FRIENDLYWIRE, 2024). Ele possui uma faixa de rotação de 180 graus, sendo ideal para aplicações que exigem controle angular limitado, no contexto do projeto, o controle da pá de um alimentador automático. Sua operação ocorre em baixa tensão (geralmente 4,8V a 6V DC) e consome correntes moderadas (≈10mA), tornando-o compatível com microcontroladores como o ESP32, que opera em 3.3V e possui limites seguros de corrente por pino (até 40mA).

O servo motor SG90 é integrado ao mecanismo de liberação de ração. Ele é responsável por movimentar uma pá acoplada ao seu eixo, que abre ou fecha o compartimento de armazenamento de alimento.

Figura 3 - Servo motor SG90

Fonte: Baseada em (Instituto Digital, 2020).

A automação de processos como a alimentação programada, controlada pelo servo motor SG90, exige precisão temporal para garantir a execução dos agendamentos conforme definido pelo usuário. Assim, dispositivos que oferecem informações sobre tempo, como por exemplo RTC (*real-time clock*), se enquadram no processo de temporizar as ações do sistema.

2.6 RTC DS3231

O módulo RTC DS3231 destaca-se por sua alta precisão e pela integração de uma bateria de backup (geralmente CR2032), que mantém o relógio em funcionamento mesmo durante falhas na alimentação principal (MAXIM INTEGRATED, 2023). Sua comunicação via protocolo I2C (*Inter-Integrated Circuit*) simplifica a integração com microcontroladores como o ESP32, permitindo a leitura e escrita de dados temporais com baixo consumo. Além disso, o DS3231 inclui outras funcionalidades, como compensação automática de temperatura, que minimiza variações causadas por ambientes térmicos instáveis, garantindo precisão de longo prazo (MAXIM INTEGRATED, 2023).

O RTC desempenha um papel fundamental no sistema. Ao depender do clock interno do ESP32, perdia-se a referência temporal após reinicializações, exigindo sincronização constante com servidores NTP (*Network Time Protocol*). Essa dependência tornava o sistema vulnerável a falhas de conexão WIFI. A implementação do DS3231 elimina esse problema, armazenando localmente os horários programados e garantindo a execução das tarefas mesmo em cenários offline ou com quedas de energia prolongadas.

2.7 Node.js

O uso de microcontroladores e sensores possibilita a automação do sistema, mas exige um software intermediário para processar dados e coordenar comandos. Sem essa camada, a comunicação com o microcontrolador seria complexa e manual. Nesse sentido, o Node.js se destaca como uma solução robusta e assíncrona para gerenciar requisições e integrar o projeto.

Node.js é uma plataforma JavaScript construída sobre o motor V8 do Chrome, voltada para o desenvolvimento de aplicações no lado do servidor (NODE.JS

FOUNDATION, 2024). Sua arquitetura assíncrona permite a criação de sistemas escaláveis e responsivos, lidando de maneira eficiente com múltiplas requisições simultâneas. A programação assíncrona e não bloqueante, sendo uma característica fundamental do Node.js, garantindo que o processamento de uma requisição não bloqueie outras operações.

O Node.js foi escolhido para o desenvolvimento do backend, responsável por gerenciar a comunicação entre o ESP32 e o *frontend*. Sua capacidade de lidar com dados em tempo real e operações assíncronas foi essencial para integrar o protocolo MQTT e o HTTP, formatando os dados vindos do *frontend* e enviando para o ESP32. A escalabilidade do Node.js também permitiu suportar múltiplos aquários conectados simultaneamente, garantindo que cada dispositivo operasse de forma independente, sem comprometer a performance do sistema (CASCIARO et al., 2020). O software desenvolvido está hospedado na plataforma Railways, que oferece infraestrutura escalável e confiável para aplicações em nuvem.

Embora o Node.js gerencie requisições e integre a comunicação entre os dispositivos, é essencial um banco de dados para armazenar e organizar as informações do sistema. Para garantir a persistência e integridade dos dados, é necessário um banco de dados capaz de armazenar informações de forma eficiente, como por exemplo o PostgreSQL.

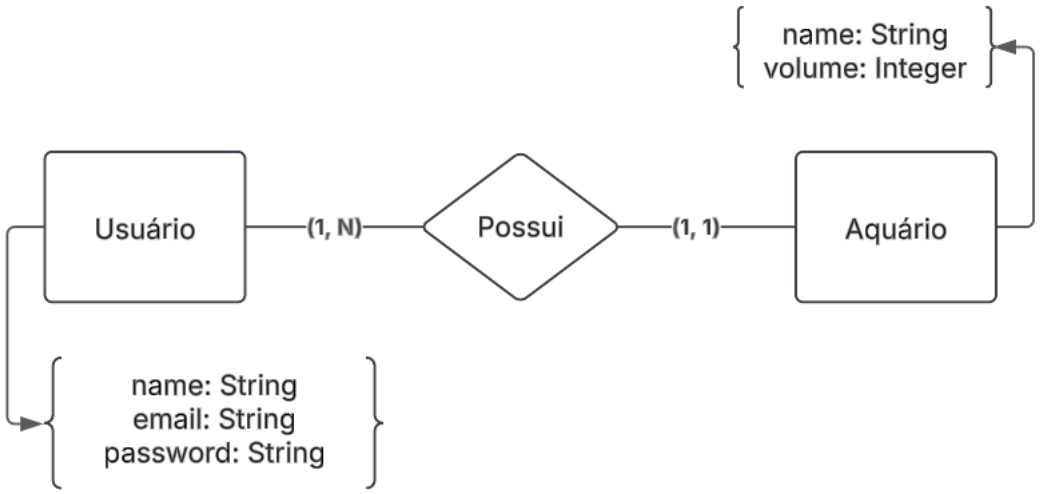
2.8 PostgreSQL

O PostgreSQL é um sistema de gerenciamento de banco de dados relacional de código aberto, reconhecido por sua robustez, flexibilidade e conformidade com padrões SQL (*Structured query language*). Sua arquitetura permite o armazenamento e recuperação eficiente de dado, suportando transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade), gatilhos, funções personalizadas e tipos de dados avançados (POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2024). Essas características o tornam ideal para aplicações como sistemas de automação que demandam armazenamento persistente de informações.

Neste projeto, o PostgreSQL foi escolhido para gerenciar o armazenamento das informações do sistema, incluindo perfis de usuários, parâmetros de aquários

(nomes e litragem), e vínculos entre usuários e aquários, como demonstrado na figura 4.

Figura 4 - Modelo do banco de dados



Fonte: Autoria própria

A integração do banco de dados foi realizada por meio do Supabase, uma plataforma gratuita *backend-as-a-service* que combina PostgreSQL com funcionalidades adicionais, como autenticação, APIs REST e *WebSocket*, simplificando o desenvolvimento de aplicações (SUPABASE INC., 2024). O Supabase oferece uma interface intuitiva para gerenciamento do banco de dados, além de ferramentas para consultas em tempo real. A combinação de PostgreSQL e Supabase permitiu a criação de uma arquitetura escalável para o backend.

Mesmo com o *backend* enviando os comandos, é necessário uma interface que facilite a interação do usuário com o sistema. Para isso, o Next.js é uma excelente escolha, pois permite a construção de uma aplicação otimizada, integrando-se ao backend para exibir informações.

2.9 Next.js

Next.js é um framework para o desenvolvimento de aplicações *frontend* com React, que oferece renderização híbrida, permitindo tanto a renderização *server-side* quanto a estática, o que resulta em melhorias significativas de SEO e performance. Além disso, o Next.js permite a criação de APIs internas e o roteamento dinâmico,

facilitando a construção de interfaces de usuário (NEXT.JS FOUNDATION, 2024). Essas funcionalidades tornam o framework ideal para a criação de aplicações web modernas e altamente escaláveis.

O Next.js foi utilizado para desenvolver a interface web do sistema, proporcionando uma experiência de usuário fluida e responsiva. Por meio do *frontend*, os usuários podem configurar horários de alimentação, ajustar ciclos de iluminação e monitorar a temperatura do aquário em tempo real. A integração simplificada com o backend permitiu que dados como agendamentos e status dos sensores fossem exibidos dinamicamente na interface, atualizando-se automaticamente sem necessidade de recarregar a página. O software desenvolvido está hospedado na Vercel, uma plataforma de hospedagem em nuvem que oferece *deploy* rápido e escalabilidade para aplicações web.

A figura 5 exibe a versão desktop, com as seções para o agendamento e monitoramento, organizadas em seções de agendamentos e um card principal com as informações. A figura 6 apresenta a versão para celulares, com as informações principais entregues ao usuário, e com botões maiores para facilitar o uso em telas menores.

Figura 5 - Tela de configuração no computador

Fonte: Autoria própria

Figura 6 - Tela de configuração no celular

Fonte: Autoria própria

Uma interface intuitiva permite que os usuários interajam com o sistema, mas para que as configurações sejam aplicadas, é necessário um meio de comunicação ágil e confiável. Nesse cenário, o protocolo MQTT se destaca, garantindo a transmissão rápida e eficiente de comandos e dados do sistema (OASIS, 2019).

2.10 MQTT

O MQTT (*Message Queuing Telemetry Transport*) é um protocolo de comunicação leve, projetado especificamente para dispositivos IoT que exigem baixa latência e comunicação em tempo real (OASIS, 2019). Ele funciona de forma eficiente em redes com baixa largura de banda, sendo uma escolha popular para sistemas que necessitam de trocas de mensagens rápidas e confiáveis.

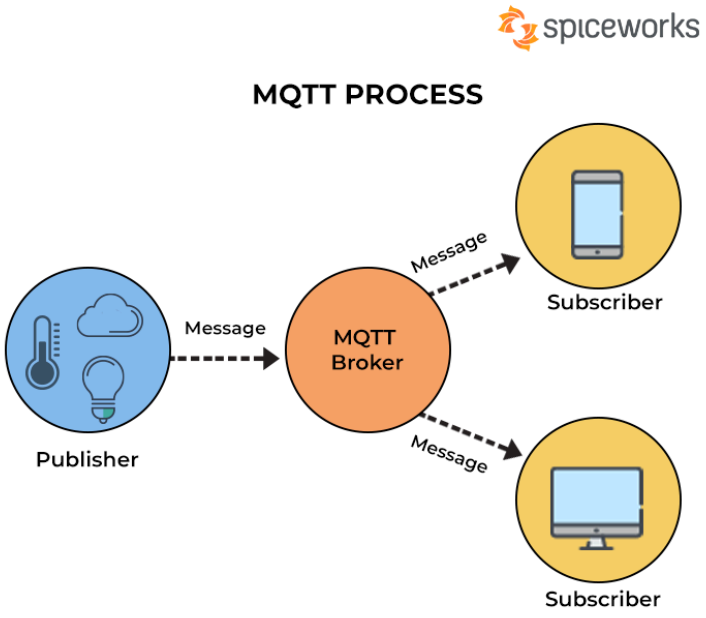
Uma das principais características do MQTT é o modelo de publicação e assinatura de tópicos, que organiza a troca de mensagens com base em tópicos específicos. Essa abordagem permite a comunicação de forma estruturada entre dispositivos, com a possibilidade de um dispositivo publicar mensagens em um

tópico e outros dispositivos se inscreverem para recebê-las.

Além disso, o MQTT oferece eficiência de banda, uma vez que utiliza uma estrutura de mensagens compacta, otimizando a comunicação em redes com recursos limitados. A sua capacidade de garantir comunicação em tempo real o torna adequado para aplicações que envolvem monitoramento e controle de dispositivos em tempo real, como sistemas de automação e IoT. No projeto, o broker MQTT utilizado está rodando no HiveMQ, uma plataforma para gerenciamento de mensagens MQTT.

A figura 7 ilustra o processo do protocolo MQTT. Quem publica (*Publisher*), envia mensagens para tópicos no Broker MQTT, que age como intermediário na comunicação. O Broker envia as mensagens para os assinantes (*Subscribers*). Esse modelo permite a troca de dados, garantindo que os dispositivos recebam apenas as mensagens dos tópicos nos quais estão inscritos

Figura 7 - Estrutura de comunicação MQTT



Fonte: Baseada em (spiceworks, 2022)

O protocolo MQTT permite a troca de mensagens em tempo real entre os dispositivos, garantindo uma comunicação ágil e eficiente. No entanto, para a integração com sistemas web e armazenamento de dados, é necessário um

protocolo amplamente utilizado para requisições estruturadas e seguras, como o HTTP, que facilita a comunicação entre o *frontend* e o *backend*.

2.11 HTTP

O HTTP (*Hypertext Transfer Protocol*), por outro lado, é um protocolo amplamente utilizado para a transferência de dados entre sistemas cliente-servidor, sendo a base para a comunicação na web (FIELDING et al., 1999). Sua estrutura de requisição e resposta organiza a comunicação entre dispositivos, permitindo que informações sejam solicitadas e enviadas de forma clara e sequencial.

Ele é amplamente utilizado em sistemas de visualização e controle via interfaces gráficas, como páginas web e APIs RESTful, facilitando a interação de usuários com dispositivos e sistemas de backend. Embora o HTTP seja mais pesado e menos eficiente em ambientes com restrições de banda, ele continua sendo fundamental para muitas aplicações, especialmente quando a compatibilidade com a web é necessária.

HTTP foi utilizado para a comunicação entre o *frontend* e o *backend*. Por meio de requisições HTTP, a interface web envia comandos de configuração para o backend, que processa e encaminha essas informações para o ESP32.

Neste capítulo foi abordado a integração de tecnologias para automação de aquários, com o uso do ESP32 como núcleo do sistema, o Sensor DS18B20 para medição de temperatura, o relé para controle de iluminação e o servo motor SG90 para o controle da alimentação.Os protocolos MQTT e HTTP garantem a troca de dados, enquanto o Node.js (*backend*), PostgreSQL e Next.js (*frontend*) compõem a arquitetura de software. O RTC DS3231 mantém salvo o tempo, independente de fatores externos, consolidando um sistema para gestão remota de aquários.

3 METODOLOGIA

Este trabalho adota uma abordagem aplicada, focada na solução de desafios enfrentados por aquaristas por meio da automação de atividades rotineiras. Para atingir esse objetivo, utilizou-se uma combinação de métodos descritivos e experimentais. O desenvolvimento do sistema começou com a identificação das necessidades do aquário, visando automatizar funções essenciais como alimentação, iluminação e controle de temperatura, para garantir a operação sem intervenção manual e manter as condições ideais do ambiente aquático.

Após definir os problemas a serem resolvidos, foi realizada uma pesquisa para identificar as tecnologias mais adequadas para a solução proposta. A seleção de componentes foi baseada em fatores como custo, facilidade de integração e disponibilidade de recursos de desenvolvimento. As tecnologias escolhidas foram:

Hardwares:

- **ESP32**: Microcontrolador escolhido devido à sua conectividade Wi-Fi, baixo consumo de energia e capacidade de controlar múltiplos dispositivos simultaneamente.
- **Sensor DS18B20**: Utilizado para monitoramento de temperatura, devido à sua alta precisão e resistência à água.
- **Relé**: Utilizado para controlar dispositivos de alta potência, como a iluminação e o sistema de alimentação automática.
- **Servo Motor SG90**: Atuador utilizado para acionar o mecanismo de alimentação automática.
- **RTC DS3231**: Relógio de tempo real usado para manter a precisão dos horários programados.

Softwares:

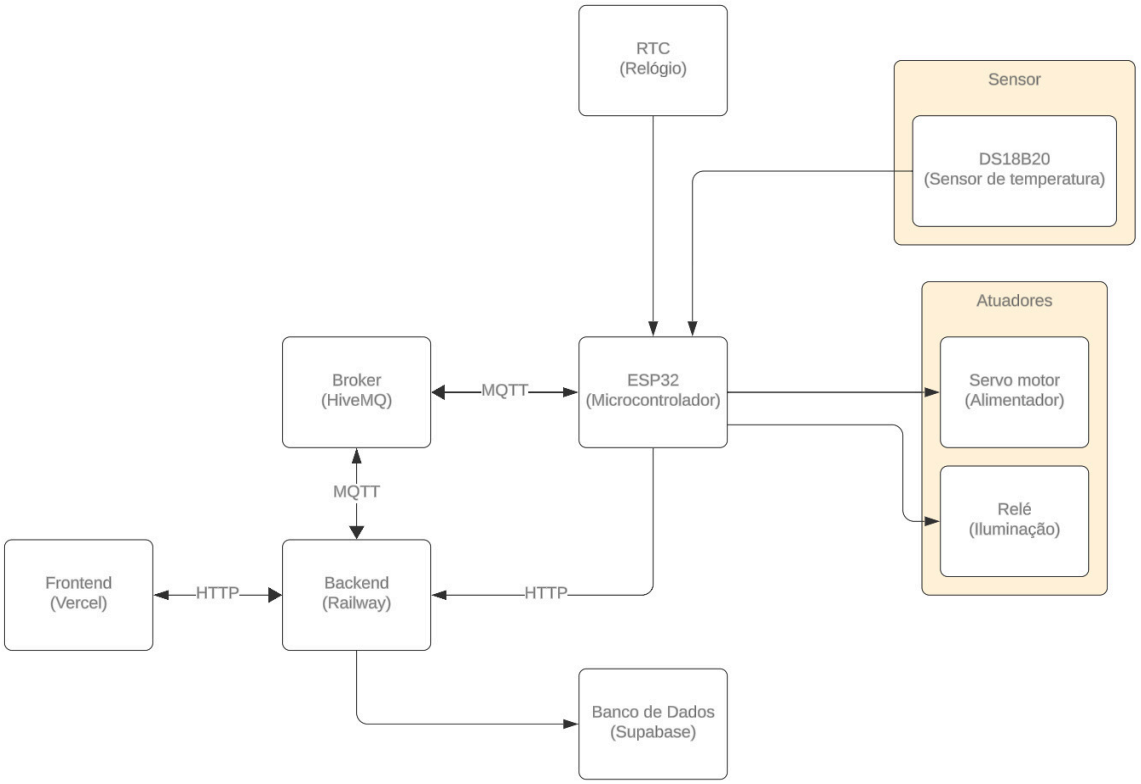
- **Backend**: Desenvolvido utilizando Node.js, permitindo uma arquitetura eficiente, com suporte à comunicação assíncrona e escalabilidade para lidar com múltiplos dispositivos.
- **Frontend**: Criado com o framework Next.js, permitindo o desenvolvimento de uma interface web dinâmica e responsiva.

- **ESP32**: Foi desenvolvido utilizando a linguagem arduino por já possuir previamente o conhecimento sobre a ferramenta de software.

Protocolos de comunicação:

- **MQTT**: Escolhido para a comunicação entre o ESP32 e o *backend*, devido à sua baixa latência e eficiência em redes com largura de banda limitada.
- **HTTP**: Utilizado para a comunicação entre o *frontend* e o *backend*, permitindo a configuração de agendamentos e o monitoramento em tempo real.

Figura 8 - Conexões do projeto



Fonte: Autoria própria

A figura 8 representa como é feita a comunicação de todo o projeto. O *frontend* hospedado na Vercel, serviço gratuito de hospedagem na nuvem (Vercel, 2025) e um *backend* rodando na Railway, outro serviço gratuito de hospedagem na nuvem, com foco em *backend*, (Railway, 2025) se comunicam via protocolo HTTP, com o *frontend* passando os comandos do usuário (agendamentos, receber informações do sensor) para o *backend*. o *backend* então formata esses dados para

poder enviar para O ESP32, fazendo uma publicação ao tópico em que o ESP32 está inscrita, que recebe o comando, processa ele e faz um retorno ao *backend* informando o status (e possiveis dados).

O ESP32 é responsável por controlar sensores e atuadores, incluindo um sensor de temperatura (DS18B20), um servo motor para alimentação e um relé para iluminação.

Para troca de dados em tempo real, o ESP32 utiliza o protocolo MQTT, conectando-se ao broker gratuito HiveMQ (HiveMQ, 2024), que intermedia a comunicação com o *backend*. Além disso, ao ser ligada pela primeira vez, o ESP32 faz uma requisição HTTP ao *backend* para receber suas configurações iniciais. Com a comunicação definida deu-se início ao desenvolvimento do *firmware* do ESP32

3.3 Desenvolvimento do Firmware do ESP32

O desenvolvimento do Firmware do ESP32 envolveu a programação do microcontrolador para desempenhar funções específicas dentro do sistema, como a leitura de dados, onde o ESP32 foi configurada para coletar informações de temperatura utilizando o sensor DS18B20 e monitorar o ambiente do aquário. Além disso, foi programada para controlar o relé, responsável pelo acionamento da iluminação, e o servo motor SG90, que atua no mecanismo de alimentação. A conectividade também foi uma parte essencial do projeto, com o ESP32 sendo conectado ao *backend* através do protocolo MQTT, permitindo o envio de dados de sensores e o recebimento de comandos em tempo real.

Para que o ESP32 possa se conectar à internet, foi implementado um método de configuração de rede via interface web. Quando o dispositivo é ligado pela primeira vez ou perde a conexão com a rede Wi-Fi, ele entra no modo AP (*Access Point*) e disponibiliza uma página de configuração acessível pelo navegador. Nessa página, o usuário pode inserir o SSID e a senha da rede Wi-Fi desejada. Após o envio dessas informações, o ESP32 tenta se conectar automaticamente e, caso tenha sucesso, salva os dados nas preferências do dispositivo para conexões futuras. A Figura 9 apresenta a tela de configuração utilizada para essa funcionalidade.

Figura 9 - Tela de configuração de rede do ESP32

Fonte: Autoria própria

Foi implementada uma solução em que os agendamentos são armazenados localmente no ESP32, garantindo que ela possa executar as tarefas programadas mesmo sem conexão WIFI. Essa abordagem permitiu a operação local independente.

O sistema de agendamento opera com base em um ciclo de verificação a cada minuto, utilizando um contador de minutos em escala de 24 horas (0 a 1440 minutos). A cada intervalo, o ESP32 verifica se há ações programadas para o minuto atual ou pendentes e caso exista uma tarefa, ela é executada imediatamente. Após a verificação, o ESP32 entra em um estado de espera de 1 minuto, dedicado apenas à recepção de atualizações do *backend*, retomando o ciclo sequencialmente. Essa lógica evita conflitos de temporização e garante funcionamento mesmo em cenários

offline. Feito todo o *firmware* do projeto, iniciou-se o desenvolvimento do *backend* com Node.js para a comunicação de todas as partes do projeto.

3.4 Desenvolvimento do Backend

O *backend* foi desenvolvido para atender às principais demandas do sistema, adotando uma API (*application programming interface*) que facilita a comunicação entre o *frontend* e o dispositivo. Essa arquitetura possibilita o recebimento de dados e comandos vindos do *frontend*, formatando-os adequadamente para encaminhar para o ESP32.

Para garantir a segurança na comunicação e a proteção dos dados dos usuários, foi implementado um sistema de autenticação e criptografia baseado em JWT (*JSON Web Token*). Quando o usuário realiza o login, um token JWT é gerado e retorna ao cliente contendo as informações criptografadas do usuário. Esse token deve ser enviado em todas as requisições subsequentes e é validado antes da execução de qualquer ação no sistema. Apenas requisições autenticadas com um token válido podem acessar e modificar os dados dos aquários, garantindo que apenas usuários autorizados possam interagir com seus dispositivos.

Além disso, a comunicação entre o backend e o ESP32 via MQTT é criptografada para evitar ataques de interceptação de dados (*Man-in-the-Middle*). O sistema utiliza TLS (*Transport Layer Security*) para proteger as mensagens enviadas entre os dispositivos e o broker MQTT, garantindo a integridade e confidencialidade dos dados trocados.

Outro ponto importante é o registro automático do equipamento no sistema. Assim que o dispositivo se conecta à internet pela primeira vez, ele envia as informações de ID (identificação) do usuário por uma requisição HTTP (*Hypertext Transfer Protocol*), permitindo a criação de um novo aquário no banco de dados e a associação deste ao usuário correspondente. Com o *firmware* e o *backend* concluídos, iniciou-se a implementação da comunicação entre ambos com o protocolo MQTT.

3.5 Comunicação MQTT e Controle de Dispositivos

A comunicação entre o ESP32, o *backend* e o *frontend* é baseada no

protocolo MQTT, permitindo a troca eficiente de informações em tempo real. Assim que o ESP32 é iniciado pela primeira vez, ele realiza uma requisição HTTP para o *backend* solicitando seu registro no banco de dados.

Como resposta, o *backend* retorna um ID (identificação), que é armazenado na memória do dispositivo. Com esse ID, o ESP32 se inscreve nos tópicos MQTT específicos para sua comunicação: "aquarium/{id}", utilizado para receber comandos do *backend* , e "aquarium/{id}/response", onde publica respostas e confirmações de execução. O *backend* , por sua vez, também se inscreve nesses tópicos após registrar o ESP32 no banco de dados, garantindo que possa enviar comandos e receber feedback do dispositivo.

A troca de mensagens ocorre de maneira assíncrona. Sempre que o *backend* pública comandos eles são enviados para o tópico correspondente e recebidos pelo ESP32, caso o ESP32 esteja desligado ou sem conexão *wireless* o *backend* retorna para o usuário que não foi possível realizar a ação. Com a conclusão da comunicação com o MQTT, iniciou-se o desenvolvimento do modelo do compartimento que fica o *hardware* do projeto.

3.6 Desenvolvimento do Modelo 3D

3.6.1 Compartimento

Após o desenvolvimento do código do ESP32, foi projetado o modelo 3D do equipamento, considerando a acomodação de todos os componentes eletrônicos como pode ser visto na Figura 10 e o posicionamento no aquário apresentado na Figura 11. O design foi elaborado para organizar os sensores, atuadores e o ESP32 dentro de um compartimento, facilitando a manutenção e o acesso aos componentes

Figura 10 - Compartimento

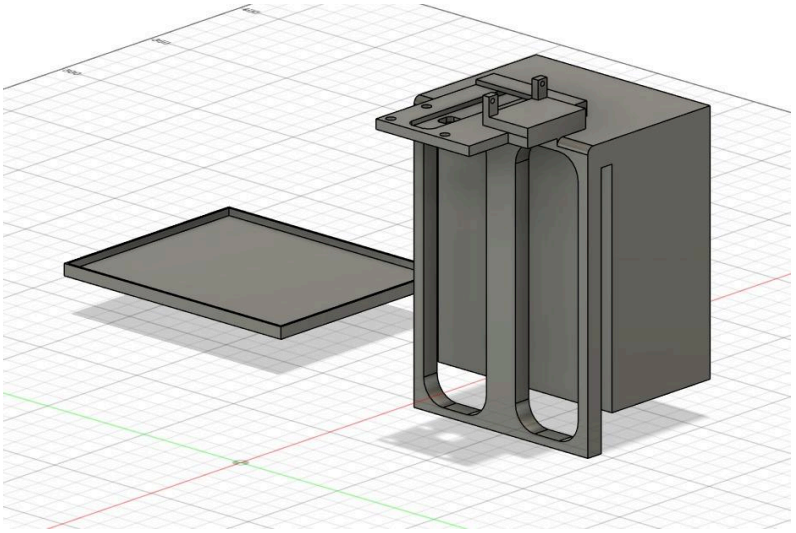
Fonte: Autoria própria

Figura 11 - Projeto posicionado no aquário

Fonte: Autoria própria

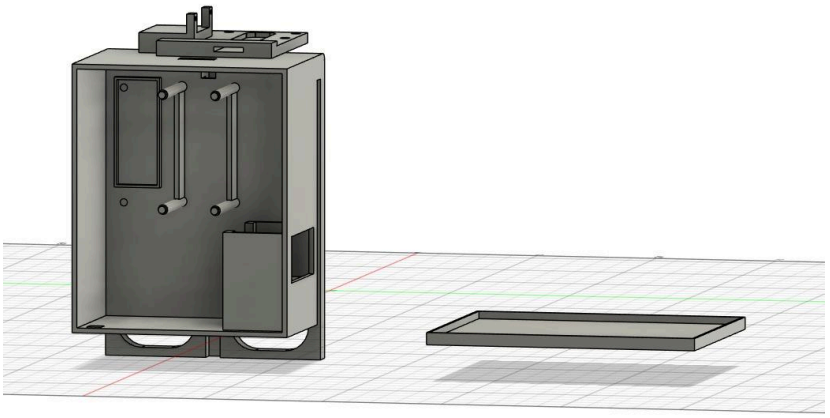
Na Figura 12, é apresentado o projeto do compartimento do alimentador, que foi otimizado com a integração do alimentador na parte superior do compartimento. Além disso, estruturas internas foram adicionadas para proporcionar suporte adequado aos componentes, estas podem ser visualizadas na Figura 13, garantindo maior estabilidade e organização.

Figura 12 - Exterior e tampa do compartimento



Fonte: Autoria própria

Figura 13 - Interior do compartimento

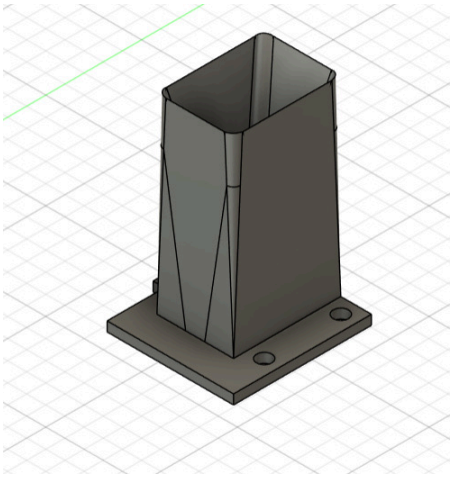


Fonte: Autoria própria

3.6.2 Alimentador

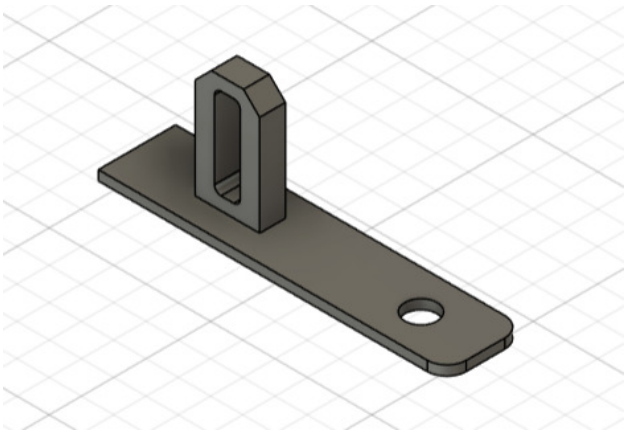
A principal função do alimentador é armazenar e distribuir a ração para os peixes, para isso, foi projetada uma peça que armazena a ração em um funil (Figura 14) e utiliza uma pá com um orifício para medir e liberar a quantidade correta (Figura 15).

Figura 14 - Funil para ração



Fonte: Autoria própria

Figura 15 - Pá do alimentador



Fonte: Autoria própria

O design deste alimentador foi inspirado no projeto disponível em (Thingiverse, 2018), adaptado para atender às necessidades deste trabalho, incluindo ajustes no tamanho do funil e no mecanismo de dosagem para garantir

uma alimentação precisa e estando contida direto no compartimento.

Com o modelo 3D concluído deu-se início ao processo de impressão do mesmo.

3.7 Impressão 3D e Montagem

A impressão 3D do modelo foi realizada com plástico PLA (*polylactic acid*), adequado para suportar as condições do ambiente de um aquário. Após a impressão, os componentes eletrônicos foram montados na estrutura, com todos os sensores e atuadores devidamente posicionados para garantir proteção e fácil acesso para eventuais manutenções.

3.9 Desenvolvimento do Frontend

O *frontend* foi desenvolvido utilizando o framework Next.js, com o objetivo de criar uma interface dinâmica e responsiva, permitindo que o usuário configure horários para alimentação, iluminação e controle de temperatura, além de exibir o status atual dos dispositivos e sensores, como temperatura, alimentação e iluminação.

A metodologia adotada, combinando abordagens descritivas e experimentais, foi fundamental para estruturar o desenvolvimento do sistema. A seleção de componentes de baixo custo e alta integração, junto com a divisão do desenvolvimento do projeto em etapas garantiu eficiência no processo.

A implementação de soluções como armazenamento local de agendamentos no ESP32 e a integração de um software intermediário via MQTT superou desafios práticos, como operação offline e escalabilidade. Testes iterativos em diferentes fases permitiram identificar e corrigir falhas de forma mais rápida. Essa abordagem metodológica assegurou que o projeto avançasse de forma organizada, alinhando-se aos objetivos propostos e ficando pronto para a validação em ambientes reais.

4 RESULTADOS

Este capítulo apresenta os resultados obtidos durante a implementação e os testes do sistema automatizado para aquários. Os resultados são organizados para detalhar cada aspecto funcional, o desempenho em cenários específicos, e a análise de métricas coletadas.

4.1 Backend

Os testes realizados no *backend* incluíram a validação da comunicação MQTT, verificando se o ESP32 estava publicando os dados corretamente nos tópicos e recebendo comandos adequados. Também foram realizados testes de agendamento, para garantir que as configurações feitas pelo *frontend* fossem corretamente processadas e publicadas no MQTT.

4.2 Frontend

Os testes realizados no *frontend* incluíram a validação da interface de agendamento, garantindo que os horários configurados pelo usuário fossem corretamente enviados ao *backend* , e o monitoramento em tempo real, testando a exibição dos dados dos sensores e atuadores na interface. Após os testes individuais, o sistema foi integrado e testado em conjunto para identificar possíveis erros. Concluída essa etapa, foram realizados testes práticos em cenários reais.

4.3 Aquário de 15 Litros

Este teste foi realizado para avaliar o funcionamento básico do sistema em um ambiente controlado. O aquário de 15 litros foi utilizado para testar as funcionalidades principais, como alimentação automática, controle de iluminação e monitoramento de temperatura, em um período de 14 dias.

4.3.1 Alimentação Automática

A alimentação foi programada para ocorrer duas vezes ao dia, às 8:00 e 18:00. O servo motor foi acionado com precisão em todos os horários, fornecendo a quantidade predeterminada de ração, e o sistema se comportou como o esperado. A variação do horário de acionamento foi de no máximo 1 minuto, o que está dentro do

intervalo de tolerância estabelecido. Durante os testes de resiliência, foram simuladas falhas, como a desconexão temporária da rede Wi-Fi, mas o sistema conseguiu retomar automaticamente as alimentações, sem perder nenhuma execução.

4.3.2 Controle de Iluminação

A iluminação foi programada para se ajustar automaticamente conforme os horários definidos no sistema. Durante os testes, o sistema conseguiu ajustar o status da iluminação, respeitando os intervalos e horários configurados pelo usuário. Não foram observadas falhas na execução, mesmo quando o sistema foi operado remotamente. Além disso, durante variações de temperatura, o controle da iluminação manteve sua eficiência, ajustando-se automaticamente conforme a configuração de horário, independentemente de picos ou quedas térmicas.

4.3.3 Comunicação e Interface

A interface de controle foi constantemente monitorada para garantir que os dados fossem atualizados em tempo real e que as interações entre o usuário e o sistema fossem processadas corretamente. A interatividade foi uma das principais qualidades, com os dados de temperatura, alimentação e status da iluminação sendo atualizados automaticamente na interface web, sem atrasos significativos.

Os comandos enviados pelo usuário, como alterar o horário de alimentação ou ligar/desligar a iluminação, foram rapidamente refletidos no *frontend* e executados no dispositivo. Além disso, a interface permitiu que o usuário visualizasse relatórios em tempo real, como o estado do próximo horário de alimentação e o estado atual da temperatura e da iluminação, oferecendo uma experiência de controle eficiente.

4.3.4 Testes no aquário de 80 litros

O teste no aquário de 80 litros foi realizado para avaliar a escalabilidade do sistema em um ambiente maior. Este ambiente apresentou novos desafios, como maior inércia térmica e maior volume de água, o que foi importante para testar o sistema em novas condições.

O aquário de 80 litros foi testado ao longo de 4 semanas, durante as quais o sistema gerenciou o controle de alimentação, iluminação e monitoramento de temperatura de forma contínua e sem interrupções. Mesmo com múltiplos aquários conectados e operando simultaneamente, o sistema manteve seu desempenho. A comunicação MQTT entre o *backend* e os dispositivos ESP32 se mostrou funcional, permitindo o controle de vários aquários simultaneamente sem perda de performance. A alimentação e iluminação ocorreram nos horários programados, sem falhas, e o sistema mostrou sinais que pode ser utilizado em aquários de maior litragem, capaz de atender aquários de até 80 litros e gerenciar as demandas de múltiplos dispositivos.

4.3.5 Monitoramento de Temperatura

Durante todo o período de testes, a temperatura foi comparada com um termômetro próprio para aquários e mantida com variações inferiores a 0,5°C, mesmo em condições de grandes variações térmicas no ambiente. Além disso, o sistema demonstrou capacidade de lidar com a maior inércia térmica do aquário de 80 litros, mantendo a precisão mesmo em mudanças mais rápidas de temperatura.

4.3 Testes Offline

Durante os testes offline, o sistema foi desconectado da internet por períodos de até 72 horas. O comportamento foi monitorado, e o sistema continuou funcionando normalmente, exceto pela falta de atualização de dados de temperatura e agendamentos. A alimentação e iluminação continuaram funcionando sem interrupções. No entanto, a falta de dados atualizados fez com que o sistema não fosse capaz de retornar a temperatura.

Tabela 3 - Funcionamento do sistema offline

Funcionamento	**Status**
Alimentação	Funcionando normalmente
Iluminação	Funcionando normalmente
Temperatura	Sem comunicação com o servidor
Agendamento	Sem comunicação com o servidor

4.4 Controle de Múltiplos Dispositivos pelo Frontend:

Durante a implementação do sistema, foi considerado o cenário de que cada usuário pode ter múltiplos aquários conectados em sua conta. Isso significa que o *backend* foi projetado para permitir que os usuários gerenciem vários dispositivos de aquário de forma totalmente independente, através de um painel único no site. Essa abordagem facilita a administração de diferentes aquários, cada um com suas próprias configurações e parâmetros, sem a necessidade de interfaces separadas para cada dispositivo.

Exemplo de Funcionalidade:

- **Aquário 1**: O usuário pode definir que a alimentação ocorrerá às 8h, 12h e 18h, enquanto a iluminação estará ativada das 7h às 19h.
- **Aquário 2**: Para outro aquário, o usuário pode programar horários diferentes, por exemplo, alimentação às 9h, 14h e 17h, com iluminação das 6h às 18h.

Os testes realizados em aquários de 15 e 80 litros demonstraram a funcionalidade do sistema, especialmente no monitoramento térmico e na capacidade de operação offline. Os resultados evidenciaram a viabilidade do sistema em ambientes reais, com operação estável por até quatro semanas em aquários de 80 litros. Com os resultados dos testes, o projeto demonstra sinais de que é possível ser utilizado em aquários de maior litragem e pela capacidade de gerenciar múltiplos dispositivos independentemente, enquanto a integração de tecnologias como o RTC DS3231 garantiu precisão temporal mesmo em quedas de energia.

5 CONCLUSÃO

O desenvolvimento do sistema automatizado para aquários proposto neste trabalho demonstrou ser uma solução funcional para a automação de tarefas essenciais, como alimentação, iluminação e controle de temperatura, proporcionando uma gestão mais prática para os usuários. Durante os testes realizados em aquários de diferentes volumes, o sistema se mostrou capaz de operar de forma contínua, atendendo às necessidades básicas de monitoramento e controle em tempo real. A integração de sensores e a comunicação via MQTT contribuíram significativamente para a estabilidade e reatividade do sistema, garantindo uma precisão nos ajustes automáticos.

Os testes realizados em ambientes controlados e em condições de uso real confirmaram a funcionalidade do sistema, com desempenho consistente ao longo de semanas. A implementação de um *backend* capaz de gerenciar múltiplos aquários por usuário proporcionou uma flexibilidade importante, permitindo que o sistema se adaptasse a diferentes cenários de uso, com a possibilidade de gerenciar vários dispositivos de aquário de forma independente, sem a necessidade de interfaces separadas. Esse aspecto se mostrou especialmente vantajoso, pois os usuários puderam controlar todos os seus aquários a partir de uma única plataforma, tornando a experiência mais intuitiva e prática.

A comunicação do sistema foi testada em diversas condições de rede, e os resultados mostraram que o sistema pode operar mesmo em cenários de conectividade limitada, garantindo que os processos de alimentação e iluminação continuem funcionando independentemente da conexão com a internet.

6 TRABALHOS FUTUROS

Com base nos resultados obtidos, o sistema tem potencial para evoluir com as seguintes melhorias:

- **Integração com Sensores Adicionais**: A adição de sensores de pH pode aumentar a capacidade do sistema, permitindo um controle mais preciso e diversificado.
- **Notificações Offline para o Usuário**: Para melhorar a experiência do usuário, seria interessante implementar notificações em caso de falhas de comunicação, usando LEDs (*Light Emitting Diodes*) ou efeitos sonoros para alertar sobre a necessidade de intervenção manual.
- **Notificações na Aplicação Web:** Implementação de alertas e notificações em tempo real na interface web para informar os usuários sobre eventos importantes.
- **Bateria para Funcionamento em Caso de Queda de Energia:** Inclusão de uma bateria de backup para garantir que o ESP32 continue operando o alimentador em caso de falha elétrica, evitando interrupções na alimentação dos peixes.
- **Integração com assistentes virtuais:** Inclusão de controle por assistentes virtuais (como Alexa, Google Assistant) para controle por comandos de voz

REFERÊNCIAS

ANALOG DEVICES. *Digital Thermometer DS18B20.* Disponível em: https://www.analog.com/media/en/technical-documentation/data-sheets/DS18B20.pdf. Acesso em: 1 nov. 2024.

FRIENDLYWIRE. *SG90 Servo Motor.* Disponível em: https://www.friendlywire.com/projects/ne555-servo-safe/SG90-datasheet.pdf. Acesso em: 1 nov. 2024.

HANDSONTEC. *1 Channel Relay Module.* Disponível em: https://handsontec.com/dataspecs/relay/1Ch-relay.pdf. Acesso em: 1 nov. 2024.

NEXT.JS FOUNDATION. *Next.js Documentation.* 2024. Disponível em: https://nextjs.org/docs. Acesso em: 5 fev. 2025.

NODE.JS FOUNDATION. *Node.js Documentation.* 2024. Disponível em: https://nodejs.org/en/docs/. Acesso em: 5 fev. 2025.

VERCEL. *Vercel Documentation*. 2025. Disponível em: https://vercel.com/docs. Acesso em: 6 fev. 2025.

RAILWAY. *Railway Documentation*. 2025. Disponível em: https://docs.railway.app. Acesso em: 6 fev. 2025.

HIVEMQ. *HiveMQ Documentation*. 2025. Disponível em: https://www.hivemq.com/docs/. Acesso em: 6 fev. 2025.

SMARTKITS. *ESP32 Pinout – Guia básico de GPIOs.* Disponível em: https://blog.smartkits.com.br/esp32-pinout-guia-basico-de-gpios/. Acesso em: 1 nov. 2024.

AQUÁRIOS & PEIXES. *Efeitos da luz no comportamento dos peixes.* Disponível em: https://aquarioepeixes.com.br/efeitos-da-luz-no-comportamento-dos-peixes/. Acesso em: 6 fev. 2025.

LENGER, R. *A importância da temperatura da água para a saúde dos peixes*. São Paulo: Editora XYZ, 2021. p. 1. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=5k8-gGDMJnwC&oi=fnd&pg=PA1&dq=importance+of+temperature+of+the+water+for+th+healf+of+fish&ots=fTqHhhKTL0&sig=-VoPbOucge3Efw0qRa9b3stcCWw#v=onepage&q=importance%20of%20temperature%20of%20the%20water%20for%20th%20healf%20of%20fish&f=false. Acesso em: 7 fev. 2025.

KHAN, M. I.; SHAH, S. F.; AL-HABIB, S. A. et al. *Temperature management for aquatic ecosystems: Modeling, estimation, and optimization.* IEEE Access, Piscataway, v. 7, p. 1-14, 2019. Disponível em: https://ieeexplore.ieee.org/abstract/document/8792852. Acesso em: 7 fev. 2025.

CITESEERX. Rep1. Documento online, State College, 2022. Disponível em: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=6134916602f7c8ec862f4cc036072331d744a6de. Acesso em: 6 fev. 2025.

CONSUL ADORAÇÃO. *A importância da limpeza e manutenção do seu aquário.* Disponível em: https://blog.consuladoracao.com.br/a-importancia-da-limpeza-e-manutencao-do-seu-aquario/#:~:text=Manter%20o%20ambiente%20do%20aqu%C3%A1rio,minimizam%20o%20estresse%20nos%20peixes. Acesso em: 6 fev. 2025.

INSTITUTO DIGITAL. *Micro Servo Motor SG90 Tower Pro 9g.* Disponível em: https://www.institutodigital.com.br/produto/micro-servo-motor-sg90-tower-pro-9g/. Acesso em: 6 fev. 2025.

SPICEWORKS. *O que é MQTT?* Disponível em: https://www.spiceworks.com/tech/iot/articles/what-is-mqtt/. Acesso em: 6 fev. 2025.

THINGIVERSE. *Automatic Fish Feeder.* Disponível em: https://www.thingiverse.com/thing:3008139. Acesso em: 6 fev. 2025.

USINAINFO. *Sensor de Temperatura DS18B20 à prova d'água 1m com adaptador.* Disponível em: https://www.usinainfo.com.br/sensor-de-temperatura/sensor-de-temperatura-ds18b20-a-prova-d-agua-1m-com-adaptador-8687.html. Acesso em: 6 fev. 2025.

QUILI AQUÁRIOS**.** *Loja Quili Aquários*. Disponível em: https://loja.quili.com.br/. Acesso em: 13 mar. 2025.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Campina Grande - Código INEP: 25137409
R. Tranqüilino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

Documento Digitalizado Restrito

TCC

Assunto:	TCC
Assinado por:	Gabriel Albino
Tipo do Documento:	Dissertação
Situação:	Finalizado
Nível de Acesso:	Restrito
Hipótese Legal:	Informação Pessoal (Art. 31 da Lei no 12.527/2011)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Gabriel Albino de Oliveira**, **DISCENTE (202011250005) DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO - CAMPINA GRANDE**, em 19/03/2025 12:13:40.

Este documento foi armazenado no SUAP em 19/03/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1425490
Código de Autenticação: 23cd435746