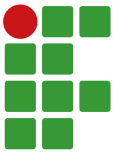


INSTITUTO FEDERAL
São Paulo
Câmpus Catanduva

Luiz Antônio Fernandes Gomes, Yago Oliveira Bomfim Dias

Sistema de Monitoramento de qualidade do ar para Ambientes Industriais

Catanduva, SP
Maio, 2024

Luiz Antônio Fernandes Gomes, Yago Oliveira Bomfim Dias

Sistema de Monitoramento de qualidade do ar para Ambientes Industriais

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, campus Catanduva como requisito parcial para conclusão do curso Especialização em Internet das Coisas.

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, campus Catanduva

Orientador: Prof. Dr. Márcio Andrey Teixeira

Catanduva, SP

Maio, 2024

Dados Internacionais de Catalogação na Publicação (CIP)
Bibliotecária: Luiza Correia Lima Felix – CRB-8/10837

Gomes, Luiz Antônio Fernandes.
G633s Sistema de monitoramento de qualidade do ar para ambientes industriais / Luiz Antônio Fernandes Gomes.
Catanduva, SP : IFSP, 2024.
40 f. : il.

Trabalho de Conclusão de Curso (Pós-Graduação – Internet das Coisas) – Instituto Federal de São Paulo, campus Catanduva, 2024.
Orientador: Prof. Dr. Márcio Andrey Teixeira.

1. IoT. 2. ESP. 3. MQTT monitoramento ambiental. 4. Automação industrial.
I. Título.

CDD (23 ed.): 004.6

ATA N.º 2/2024 - IOT-CTD/DRG/CTD/IFSP

Coordenação da Pós-Graduação em Internet das Coisas

Ata de Defesa de Trabalho de Conclusão de Curso - Pós-Graduação

Na presente data realizou-se a sessão pública de defesa do Trabalho de Conclusão de Curso intitulado **Sistema de Monitoramento de Qualidade do Ar para Ambientes Industriais** apresentado pelos alunos **Luiz Antônio Fernandes Gomes (CT3017079) e Yago Oliveira Bomfim Dias (CT3017087)** do Curso Lato Senso de **ESPECIALIZAÇÃO EM INTERNET DAS COISAS,** (Campus Catanduva). Os trabalhos foram iniciados às 18:00 pelo(a) Professor(a) presidente da banca examinadora, constituída pelos seguintes membros:

Membros	IES	Presença (Sim/Não)	Aprovação/Conceito (Quando Exigido)
Márcio Andrey Teixeira (Presidente/Orientador)	IFSP-CTD	Sim	APROVADO/9.00
Luis Fernando Castilho Maschi (Examinador 1)	IFSP-CTD	Sim	APROVADO/9.00
Murilo Secchieri de Carvalho (Examinador 2)	IFSP-CTD	Sim	APROVADO/9.00

Observações:

A banca examinadora, tendo terminado a apresentação do conteúdo da monografia, passou à arguição do(a) candidato(a). Em seguida, os examinadores reuniram-se para avaliação e deram o parecer final sobre o trabalho apresentado pelo(a) aluno(a), tendo sido atribuído o seguinte resultado:

[X] Aprovado(a) [ ] Reprovado(a) Nota (quando exigido): 9.00

Proclamados os resultados pelo presidente da banca examinadora, foram encerrados os trabalhos e, para constar, eu lavrei a presente ata que assino juntamente com os demais membros da banca examinadora.

Câmpus Catanduva, 03 de Dezembro de 2024

Avaliador externo: [ ]Sim [X]Não

Assinatura:

Documento assinado eletronicamente por:

- **Marcio Andrey Teixeira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 09/12/2024 08:36:42.
- **Luiz Antônio Fernandes Gomes**, **CT3017079 - Discente**, em 09/12/2024 09:25:02.
- **Murilo Secchieri de Carvalho**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 09/12/2024 11:01:08.
- **Yago Oliveira Bomfim Dias**, **CT3017087 - Discente**, em 09/12/2024 19:37:44.
- **Luis Fernando Castilho Maschi**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 11/12/2024 17:54:27.

Resumo

Este projeto apresenta o desenvolvimento de um sistema IoT para monitoramento de qualidade do ar em ambientes industriais, com foco na exibição de dados que possam auxiliar na questão de segurança e na automação de ações corretivas em tempo real. Utilizando sensores de temperatura, umidade e gases nocivos como monóxido de carbono e amônia, integrados ao protocolo MQTT para transmissão de dados, o sistema permite a coleta e o processamento contínuo das condições ambientais. A solução inclui uma interface web que possibilita a visualização em tempo real dos dados e a configuração de alarmes e limites de segurança. Os testes realizados confirmaram a viabilidade do sistema, que se mostrou escalável e adaptável a diferentes contextos industriais. Este projeto contribui para a melhoria das condições de trabalho em ambientes insalubres, facilitando o monitoramento e a resposta rápida a condições ambientais adversas.

Palavras-chave: IoT, ESP, MQTT monitoramento ambiental, qualidade do ar, automação industrial.

Abstract

This project presents the development of an IoT system for monitoring air quality in industrial environments, focusing on displaying data that can help with safety issues and automating corrective actions in real time. Using temperature, humidity and harmful gas sensors such as carbon monoxide and ammonia, integrated with the MQTT protocol for data transmission, the system allows the continuous collection and processing of environmental conditions. The solution includes a web interface that allows real-time viewing of data and the configuration of alarms and safety limits. The tests carried out confirmed the viability of the system, which proved to be scalable and adaptable to different industrial contexts. This project contributes to improving working conditions in unhealthy environments, facilitating monitoring and rapid response to adverse environmental conditions.

Keywords: ESP, MQTT, IoT, environmental monitoring, air quality, industrial automation.

Lista de figuras

Sumário

1 Introdução

A crescente aplicação da Internet das Coisas (IoT) em contextos industriais PAVÃO (2024) tem proporcionado novas soluções para o monitoramento ambiental, especialmente em ambientes onde a qualidade do ar e a segurança dos envolvidos no espaço são preocupações centrais. A IoT permite que dispositivos e sensores troquem dados em tempo real ZAMBELLI (2023), viabilizando uma visão abrangente das condições ambientais em plantas industriais. Este monitoramento contínuo pode ser fundamental em indústrias que lidam com substâncias químicas ou processos que podem liberar gases tóxicos e comprometer a saúde dos trabalhadores, como refinarias, fábricas de metais e instalações de tratamento de água.

Ambientes industriais frequentemente podem apresentar condições adversas, com alta concentração de substâncias nocivas no ar, que podem afetar tanto a integridade dos equipamentos quanto a segurança ocupacional (BRASIL, 2001). Nesse cenário, sistemas de monitoramento que capturam variáveis como temperatura, umidade e concentração de gases tóxicos tornam-se essenciais. A detecção rápida de alterações na qualidade do ar e a automação de respostas, como a ativação de dispositivos de mitigação, são medidas eficazes para prevenir acidentes e manter a segurança no local de trabalho.

Este trabalho propõe o desenvolvimento de um sistema de monitoramento de qualidade do ar em ambientes industriais utilizando tecnologias IoT, com o objetivo de monitorar as condições ambientais em tempo real. O sistema é composto por sensores estrategicamente distribuídos, capazes de detectar e informar alterações em parâmetros críticos, como gases tóxicos e condições atmosféricas desfavoráveis. Os dados coletados são transmitidos via protocolo MQTT para um servidor central, que permite o acompanhamento em uma interface web intuitiva. A *dashboard* possibilita aos usuários configurar parâmetros de alarme e limites de segurança, além de visualizar os dados em tempo real.

Ao propor uma solução IoT acessível e escalável, este projeto visa facilitar a modernização de processos industriais, reduzindo custos e oferecendo um sistema de monitoramento ambiental que não requer grandes alterações estruturais. A integração dos sensores com uma plataforma de controle e a utilização de tecnologias de código aberto garantem que o sistema proposto seja viável economicamente e fácil de implementar. Dessa forma, o sistema contribui para melhorar as condições de trabalho em ambientes industriais, garantindo que intervenções corretivas possam ser realizadas de forma ágil e automática.

2 Objetivos

2.1 Objetivo geral

Este projeto pretende especificar e construir um protótipo físico de um módulo de coleta de dados, e uma *dashboard*, que permita apresentar informações pertinentes ao ambiente em tempo real e que permita a customização de parâmetros e alarmes. O objetivo principal deste trabalho é ter um sistema integrado de monitoramento que seja capaz de medir e controlar a qualidade do ar em tempo real. O sistema deve coletar os dados das condições ambientais, como temperatura, umidade e concentração de gases e poderá permitir o acionamento de equipamentos externos específicos para a resolução do problema identificado.

2.2 Objetivos específicos

- Projetar e construir os módulos de *hardware* necessários para a coleta de dados ambientais.
- Especificar e implementar o *software* que será executado nos módulos, responsável pela coleta e transmissão dos dados.
- Utilização de um servidor para armazenamento e processamento dos dados coletados, utilizando *software* de controle.
- Implementar uma interface *web* (*dashboard*) para visualização dos dados em tempo real.
- Garantir que o sistema seja de fácil implementação, sem a necessidade de grandes alterações estruturais no local de instalação.

2.3 Estrutura do trabalho

A presente monografia está organizada da seguinte forma: no Capítulo 1, apresenta-se uma introdução ao tema e a proposta do projeto. O Capítulo 2 descreve os objetivos da pesquisa. No Capítulo 3, é explicada a metodologia utilizada para o desenvolvimento do projeto. O Capítulo 4 trata da fundamentação teórica, abordando os conceitos, como IoT, protocolos de comunicação e frameworks de desenvolvimento. O Capítulo 5 discute as tecnologias utilizadas. No Capítulo 6, realiza-se uma análise de trabalhos relacionados ao tema. O Capítulo 7 descreve o desenvolvimento do projeto, detalhando tanto o *hardware* quanto a parte do sistema. No Capítulo 8, são apresentados os resultados obtidos. Por fim, o Capítulo 9 traz as considerações finais e sugestões para trabalhos futuros.

3 Metodologia

O desenvolvimento do *software*, tanto dos módulos quanto da interface do usuário (*dashboard*), seguirá as melhores práticas e recomendações de desenvolvimento de *software*. Inicialmente, prevê-se o uso exclusivo de *softwares* livres e de código aberto, o que facilita o acesso e a adaptação às necessidades do projeto. A principal linguagem de programação utilizada será o TypeScript, que será aplicada no desenvolvimento da aplicação de controle dos módulos (*back-end*). Para os módulos de coleta de dados, será utilizada a linguagem MicroPython, escolhida por sua versatilidade e facilidade de desenvolvimento em sistemas embarcados SOUZA (2024).

Para garantir a comunicação eficiente entre os módulos e o servidor, será adotado o protocolo MQTT sobre uma rede TCP/IP, proporcionando uma comunicação leve e adequada para dispositivos com baixa capacidade de processamento CRAVO (2024). A comunicação entre a *dashboard* e o *back-end* ocorrerá por meio do protocolo HTTP, garantindo a troca de informações em tempo real de forma segura e eficiente.

O controle de versão do *software* será realizado utilizando a ferramenta Git, com o repositório hospedado no GitHub, permitindo o gerenciamento das modificações e a colaboração no desenvolvimento.

Os requisitos de *hardware* para a implementação da prova de conceito incluem uma placa de desenvolvimento compatível com a ESP8266, um sensor de temperatura e umidade DHT-22 (selecionado por sua precisão), um sensor de qualidade do ar MQ-135 (capaz de detectar gases como monóxido de carbono, benzeno, amônia e fumaça), um módulo de relé para controle de dispositivos externos e outros componentes acessórios. Esses elementos formarão a base para a coleta e análise dos dados ambientais, assegurando a precisão e a confiabilidade das medições.

4 Fundamentação teórica

Esta seção será referente a fundamentação teórica que irá abordar sobre os principais conceitos relacionados ao trabalho.

4.1 Internet das Coisas (IoT)

A Internet das Coisas (IoT) refere-se a uma rede de dispositivos que são incorporados com sensores, *software* e conectividade de rede, permitindo coletar e compartilhar dados. Conceitos fundamentais da IoT incluem a comunicação entre dispositivos, a coleta de dados sensoriais e a análise desses dados para a tomada de decisões. A IoT é crucial para o desenvolvimento do sistema de monitoramento, pois fornece a base para a integração dos sensores com a plataforma de controle IBM (2024).

4.2 Protocolos de comunicação em IoT

Os protocolos de comunicação em sistemas de Internet das Coisas (IoT) são essenciais para a transmissão de dados entre dispositivos conectados e servidores centrais, possibilitando a transmissão da informação entre as camadas e entre equipamentos, utilizando um conjunto de regras e padrões para possibilitar a comunicação entre dispositivos diferentes. Esses protocolos permitem que sensores, atuadores e outros dispositivos interajam de maneira eficiente e segura em uma rede distribuída, garantindo a troca de informações em tempo real. A escolha do protocolo de comunicação para um sistema IoT depende de vários fatores, incluindo a necessidade de economia de energia, a largura de banda disponível, a latência permitida e a complexidade do ambiente de implementação (BRITO et al., 2018). No contexto deste projeto, o protocolo MQTT é essencial, pois oferece uma solução eficiente para a transmissão de dados em tempo real entre os módulos de sensores e o sistema central, utilizando uma rede TCP/IP para garantir a estabilidade e a integridade das informações transmitidas.

4.3 Monitoramento ambiental

O monitoramento ambiental é a prática de medir parâmetros físicos e químicos de um ambiente, como temperatura, umidade, qualidade do ar e presença de gases nocivos. Em ambientes industriais, o controle dessas variáveis é crítico para garantir a segurança dos trabalhadores e a eficiência operacional TERA AMBIENTAL (2024). Sensores como o DHT-22, que mede temperatura e umidade, e o MQ-135, utilizado para detectar gases como monóxido de carbono, dióxido de carbono, amônia, fumaça e compostos orgânicos voláteis, são exemplos de sensores que podem ser utilizados em sistemas de monitoramento ambiental. Em locais, como plantas químicas, fábricas metalúrgicas e instalações de tratamento de resíduos, o controle dessas variáveis pode ser essencial para otimizar a eficiência

operacional e prevenir danos aos equipamentos. Os dados coletados pelos sensores podem ser enviados para sistemas de controle que, ao identificar condições adversas, podem acionar dispositivos de mitigação. Entre as ações corretivas mais comuns estão a ativação de exaustores para reduzir a concentração de gases tóxicos, o acionamento de sistemas de purificação de ar para eliminar partículas ou substâncias específicas, além de ajustes automáticos de climatização para manter a temperatura e a umidade nos níveis ideais. Essas medidas não apenas garantem condições adequadas para o funcionamento dos processos industriais, mas também atendem às regulamentações ambientais e reduzem o impacto das operações no entorno.

4.4 Arquiteturas distribuídas

Arquiteturas distribuídas referem-se a modelos de sistemas em que o processamento e o armazenamento de dados são realizados de maneira descentralizada, utilizando múltiplos nós (dispositivos, servidores ou sistemas) que colaboram para alcançar os objetivos da aplicação. Diferentemente das arquiteturas centralizadas, onde todo o processamento ocorre em um único local, em sistemas distribuídos as tarefas são divididas entre diversos dispositivos conectados por uma rede, como a Internet (SOUZA et al., 2016). Essa abordagem oferece benefícios como escalabilidade, resiliência a falhas e maior flexibilidade para lidar com cargas de trabalho distribuídas.

No contexto deste trabalho de conclusão de curso, que visa o monitoramento da qualidade do ar em ambientes industriais, foi adotada uma arquitetura distribuída para integrar sensores baseados em microcontroladores ESP32, um servidor central e uma interface *web*. Cada módulo de sensores opera de forma autônoma, coletando dados ambientais como temperatura, umidade e níveis de gases tóxicos, e transmite essas informações ao servidor utilizando o protocolo MQTT. Esse protocolo assegura uma comunicação leve e eficiente, essencial para redes com recursos limitados (ZOLETT; RAMIREZ, 2020).

O servidor central, por sua vez, processa e armazena os dados em um banco de dados MongoDB, enquanto a interface *web*, desenvolvida com Vue.js, exibe as informações em tempo real para os usuários. Essa arquitetura distribuída permite que os módulos sejam configurados e adicionados com facilidade, garantindo escalabilidade para aplicações industriais em larga escala. Além disso, a independência dos nós assegura a continuidade da coleta de dados mesmo em caso de falhas isoladas, tornando o sistema robusto e confiável.

4.5 Sensores e atuadores

Sensores e atuadores são componentes fundamentais em sistemas de monitoramento e automação. Sensores são dispositivos que capturam dados do ambiente, convertendo variáveis físicas ou químicas, como temperatura, umidade ou concentração de gases, em sinais elétricos que podem ser interpretados por sistemas computacionais. Atuadores, por outro lado, são dispositivos que executam ações físicas em resposta a comandos emitidos pelo sistema, como ligar exaustores, acionar alarmes ou abrir válvulas. Enquanto os

sensores monitoram o ambiente e fornecem os dados necessários para a tomada de decisão, os atuadores implementam essas decisões, completando o ciclo de controle.

Neste trabalho de conclusão de curso, foram utilizados os seguintes sensores:

- Sensor de Temperatura e Umidade (DHT22): Este sensor foi selecionado devido à sua alta precisão e confiabilidade. Ele é capaz de medir temperatura com precisão de 0,1 °C e umidade relativa com precisão de 0,1%, conforme especificado em sua folha de dados (*datasheet*). Esses parâmetros são críticos para o monitoramento de condições ambientais em ambientes industriais.
- Sensor de Gases (MQ-135): Este sensor foi escolhido pela sua capacidade de detectar uma ampla faixa de gases tóxicos, como amônia, monóxido de carbono, dióxido de carbono, óxidos de nitrogênio e fumaça. A saída analógica do sensor foi conectada a um módulo ADC (conversor analógico-digital) ADS1115 para melhorar a precisão das leituras, uma vez que o ADC nativo do ESP32 apresentou imprecisões durante os testes.
- Módulo ADS1115: Este componente foi integrado ao sistema para converter os sinais analógicos do MQ-135 em sinais digitais, oferecendo uma resolução de 16 *bits*. A comunicação I2C do ADS1115 facilitou sua integração com o microcontrolador ESP32, garantindo maior precisão nas leituras dos sensores.

A combinação desses sensores permitiu a coleta confiável de dados ambientais, enquanto os atuadores, como relés para controle de dispositivos externos, completam o ciclo de monitoramento e controle. A integração precisa entre sensores e atuadores é essencial para sistemas de IoT em tempo real, como o proposto neste trabalho, garantindo a execução automática de medidas corretivas quando limites críticos são excedidos (RAMOS, 2020; ANDRADE, 2016).

4.6 JavaScript e TypeScript

O JavaScript é uma linguagem de programação de computadores que é vastamente utilizada em aplicações para a *web*, permitindo que *websites* sejam mais dinâmicos e interativos. O JavaScript funciona no navegador *web*, hoje em dia a maioria dos sites a utilizam (LATEEF, 2018).

TypeScript é um superconjunto para o JavaScript que permite melhorar a forma que o código trabalha e deixá-lo mais conciso e legível. O TypeScript permite a adição de tipos de dados no JavaScript, tornando as aplicações que à utilizam muito mais estáveis e seguras, além de melhorar o suporte à programação orientada a objetos, garantindo uma melhor modularização da aplicação (NOLETO, 2020).

4.7 Tipos de bancos de dados

Bancos de dados são ferramentas essenciais para o armazenamento, organização e recuperação de dados em sistemas computacionais. Eles podem ser classificados em dois tipos principais: relacionais e não relacionais.

4.7.1 Bancos de Dados Relacionais

Bancos de dados relacionais armazenam informações em tabelas organizadas em linhas e colunas, estabelecendo relações bem definidas entre os dados. Esse modelo utiliza uma linguagem padrão, o SQL (Linguagem de Consulta Estruturada traduzido do inglês *Structured Query Language*), para realizar operações de consulta, inserção, atualização e exclusão de dados.

Entre as principais vantagens dos bancos relacionais estão:

- Estruturação de dados clara e bem definida, adequada para dados transacionais;
- Suporte robusto para integridade referencial e transações complexas;
- Ampla aceitação no mercado, com uma variedade de sistemas como MySQL, PostgreSQL e Oracle Database.

No entanto, esses bancos apresentam limitações quando se trata de escalabilidade horizontal e flexibilidade para lidar com dados sem estrutura fixa (CHAN, 2019).

4.7.2 Bancos de Dados Não Relacionais

Bancos de dados não relacionais, ou NoSQL, são projetados para lidar com grandes volumes de dados não estruturados ou semiestruturados. Diferentemente dos relacionais, esses bancos armazenam dados em formatos como documentos, pares chave-valor, grafos ou colunas largas, dependendo do modelo escolhido.

As vantagens dos bancos não relacionais incluem:

- Alta escalabilidade horizontal, permitindo distribuir dados entre vários servidores;
- Flexibilidade na estrutura de dados, eliminando a necessidade de um esquema fixo;
- Melhor desempenho para leituras e gravações em grandes volumes de dados.

Entre os bancos de dados não relacionais amplamente utilizados estão MongoDB, Cassandra e Redis. Esses sistemas são especialmente úteis em aplicações que requerem respostas rápidas e integração com dados de diversas fontes (GUEDES, 2020).

4.7.3 Escolha do Banco de Dados no Projeto

Para este trabalho, optou-se pelo uso do MongoDB, um banco de dados não relacional baseado no modelo de documentos. Essa escolha se deu devido às características do sistema proposto, que exige flexibilidade na estrutura dos dados e capacidade de lidar com um grande volume de informações provenientes de múltiplos sensores distribuídos.

O MongoDB permitiu armazenar os dados coletados pelos sensores em formato *JSON*, facilitando sua manipulação e integração com outros componentes do sistema, como a interface *web* desenvolvida com Vue.js. Além disso, sua escalabilidade horizontal garante que o sistema possa crescer conforme a adição de novos módulos de sensores e expansões na infraestrutura.

A escolha do MongoDB também levou em consideração sua capacidade de suportar atualizações rápidas e consultas eficientes, essenciais para o monitoramento em tempo real exigido pelo projeto (GUEDES, 2020).

4.8 MicroPython

MicroPython é uma implementação da linguagem de programação Python projetada para rodar em microcontroladores e dispositivos embarcados com recursos limitados de *hardware*, como memória e capacidade de processamento. Baseado no Python 3, ele mantém muitas de suas funcionalidades principais, mas é otimizado para operar em ambientes de baixa potência (ANDRADE, 2016).

4.8.1 Diferenças entre Python e MicroPython

Embora o MicroPython seja um subconjunto do Python, existem algumas diferenças importantes:

- **Tamanho reduzido**: MicroPython é muito mais compacto, projetado para caber em microcontroladores com poucos *kilobytes* de memória. Em contrapartida, o Python completo é projetado para sistemas mais robustos, como computadores pessoais e servidores.
- **Bibliotecas específicas**: MicroPython inclui bibliotecas dedicadas para lidar com hardware embarcado, como GPIO, I2C e SPI. Muitas das bibliotecas padrão do Python não estão disponíveis no MicroPython devido às limitações de memória e processamento.
- **Execução direta em hardware**: Enquanto o Python geralmente depende de um sistema operacional, o MicroPython pode ser executado diretamente em *hardware*, eliminando a necessidade de um sistema operacional subjacente.
- **Interface interativa**: Assim como o Python, o MicroPython possui um interpretador REPL (*Read-Eval-Print Loop*), mas adaptado para comandos diretamente no microcontrolador.

4.8.2 Por que Utilizar MicroPython?

A escolha do MicroPython para o projeto foi baseada em suas vantagens específicas para sistemas embarcados:

- **Facilidade de uso**: A linguagem Python é conhecida por sua sintaxe simples e acessível, e o MicroPython herda essa característica, permitindo um desenvolvimento ágil e de fácil compreensão.
- **Bibliotecas especializadas**: Com suporte nativo para sensores e comunicação, o MicroPython simplifica a integração de *hardware* como o ESP32 utilizado neste projeto.
- **Compatibilidade com sistemas embarcados**: O MicroPython é otimizado para rodar diretamente em microcontroladores, aproveitando ao máximo seus recursos limitados.
- **Código reutilizável**: Como é baseado no Python 3, o código escrito em MicroPython pode ser facilmente adaptado para outras aplicações Python, promovendo maior portabilidade.

4.8.3 Aplicação neste trabalho

No contexto do monitoramento da qualidade do ar, o MicroPython foi utilizado para programar os módulos baseados no ESP32. Sua capacidade de lidar com bibliotecas específicas para sensores, como DHT22 e MQ-135, permitiu o desenvolvimento de um sistema eficiente para coleta e transmissão de dados.

Além disso, a simplicidade do MicroPython facilitou a implementação de funcionalidades como a conexão à rede WiFi, a comunicação via protocolo MQTT e a integração com o módulo ADS1115 para leituras analógicas precisas. Essas características tornam o MicroPython uma escolha ideal para aplicações de IoT em tempo real, como a proposta neste projeto (ROVEDA, 2020).

5 Tecnologias envolvidas

Esta seção será referente as tecnologias fundamentais que foram necessários para a realização do trabalho, também com suas definições e exemplos de uso dentro do trabalho.

As tecnologias descritas foram integradas de maneira funcional para garantir o desempenho do funcionamento do sistema. O MQTT foi utilizado para a comunicação eficiente entre os sensores e o servidor, enquanto o WiFi, implementado com o módulo ESP32, permitiu a transmissão dos dados coletados para o servidor em tempo real. No *back-end*, o Nest.JS gerenciou a lógica do sistema, organizando os serviços em módulos e implementando as APIs RESTful para integração com a interface do usuário. Os dados coletados foram armazenados no MongoDB, aproveitando sua flexibilidade e escalabilidade para gerenciar grandes volumes de informações. A *dashboard* foi desenvolvida com Vue.JS e Nuxt.JS, garantindo uma interface simples e responsiva, enquanto o Socket.IO permitiu atualizações em tempo real dos dados sem necessidade de recarregar a página. Por fim, o uso do Git e GitHub assegurou o controle de versão e a organização do código durante o desenvolvimento. Essa integração resultou em um sistema direto, funcional e alinhado aos objetivos do projeto.

5.1 MQTT

O *Message Queuing Telemetry Transport* (MQTT) é um protocolo leve de comunicação, que opera sobre o protocolo TCP na camada de transporte. Foi desenvolvido para ser aplicado em ambientes com redes e dispositivos restritos, amplamente utilizado em aplicações IoT. Projetado para redes de baixa largura de banda e ambientes com recursos limitados, o MQTT é ideal para a troca de dados entre dispositivos como sensores e um servidor central. O protocolo opera em um modelo de publicação/assinatura, permitindo que diferentes dispositivos enviem e recebam mensagens de maneira assíncrona, garantindo eficiência e escalabilidade no sistema de monitoramento proposto (ZOLETT; RAMIREZ, 2020).

5.2 Bluetooth

O *Bluetooth* é uma tecnologia de comunicação sem fio de curto alcance amplamente utilizada para a troca de dados entre dispositivos próximos. Este projeto utiliza a tecnologia *Bluetooth Low Energy* (BLE), uma versão otimizada do *Bluetooth* projetada para dispositivos de IoT que precisam operar por longos períodos com baterias de capacidade limitada. O BLE é ideal para transferências de dados de baixa intensidade, como a configuração inicial do dispositivo de monitoramento ou o envio de pequenas quantidades de dados diretamente ao usuário final. Embora o foco principal deste projeto seja o uso do Wi-Fi para a comunicação, o *Bluetooth* pode ser considerado uma alternativa para cenários onde há restrições no acesso à internet, permitindo a comunicação direta entre o dispositivo de monitoramento e

o usuário final (ALVAREZ; ANTUNES, 2015).

5.3 WiFi

O WiFi é uma tecnologia de comunicação sem fio que utiliza as frequências de 2,4 GHz, 5 GHz ou 6 GHz para conectar dispositivos a uma rede local ou à internet Braga e Marques (2023). Essa tecnologia é amplamente empregada em dispositivos de IoT devido à sua alta taxa de transmissão de dados, confiabilidade e alcance satisfatório, que pode cobrir grandes áreas dependendo da infraestrutura de rede instalada. Dentro do projeto o WiFi é utilizado para transmitir os dados coletados pelos sensores ao servidor central, onde são processados e armazenados. Essa abordagem garante uma comunicação em tempo real entre os módulos e o sistema, permitindo que os dados sejam acessados e analisados instantaneamente na *dashboard* do usuário. Além disso, o WiFi é fundamental para a escalabilidade do sistema, pois possibilita a integração de múltiplos dispositivos em uma única rede, sem a necessidade de infraestrutura adicional (RAMOS, 2020).

5.4 Nest.JS

O Nest.JS é um *framework* de desenvolvimento web baseado em Node.js, que utiliza os princípios da programação orientada a objetos e programação modular. Sua arquitetura modular permite a construção de aplicativos robustos e escaláveis, sendo ideal para o desenvolvimento do *back-end* do sistema de monitoramento. No contexto deste projeto, o Nest.JS foi utilizado para desenvolver o *back-end* do sistema de monitoramento, sendo responsável por gerenciar a comunicação entre o servidor e os dispositivos de IoT. Além disso, o Nest.JS facilita a implementação de APIs RESTful, o que será essencial para a comunicação entre o servidor e a interface de usuário (*dashboard*) RANGEL (2023).

5.5 Arquitetura MVCS

A arquitetura MVCS (*Model-View-Controller-Service*) é uma variante do tradicional padrão MVC (*Model-View-Controller*). No contexto deste projeto, essa arquitetura será utilizada para organizar o código do sistema, separando as responsabilidades entre o modelo de dados (*Model*), a interface de usuário (*View*), o controle da lógica de aplicação (*Controller*) e os serviços de comunicação (*Service*). Essa abordagem modular facilita a manutenção e a escalabilidade do sistema, além de tornar o desenvolvimento mais eficiente e organizado SOUZA (2023).

5.6 API REST

As API's REST, é uma *Application Programming Interface* (interface de programação de aplicações) que segue os padrões de arquitetura REST, que permite a integração com serviços *web*. REST é uma sigla em inglês para transferência representacional de estado, que foi criada pelo cientista da computação Roy Fielding (REDHAT, 2018). Essa arquitetura

é geralmente utilizada para consulta de dados remotas de forma segura, como por exemplo, retornar o endereço a partir do CEP informado pelo usuário ou autenticar informações sensíveis por exemplo.

5.7 Git e GitHub

O Git é uma ferramenta de código aberto para controle de versão, sendo utilizado pela grande maioria dos programadores. Ela possibilita a criação de um histórico das alterações realizadas no código do projeto, garantindo também que seja possível restaurar essas alterações à qualquer momento (FERNANDES, 2018).

O GitHub é um serviço online que permite a hospedagem de repositórios Git, garantindo que seja possível o armazenamento do estado da aplicação, fora do ambiente de desenvolvimento (FERNANDES, 2018).

5.8 MongoDB

MongoDB é um tipo de banco de dados não relacional de alta velocidade que permite o armazenamento de dados utilizando documentos de dados. Ele é um banco de dados muito flexível, permitindo que distintos dados sejam armazenados no mesmo documento, além de permitir um escalonamento horizontal da aplicação sem muito esforço, garantindo velocidade e eficiência mesmo com uma quantidade massiva de dados (GUEDES, 2020).

5.9 Socket.IO

O Socket.IO é uma biblioteca JavaScript que permite a comunicação em tempo real entre cliente e servidor, utilizando *WebSockets*. No contexto deste projeto, ele será utilizado para garantir que as atualizações das medições dos sensores sejam refletidas instantaneamente na *dashboard*, sem a necessidade de recarregar a página ou realizar novas requisições HTTP a cada atualização.

5.10 Vue.JS e Nuxt.JS

O Vue.JS é um *framework* progressivo para a construção de interfaces de usuário, que será utilizado no desenvolvimento da *dashboard* do sistema. Ele oferece uma abordagem reativa e eficiente para a criação de interfaces dinâmicas e interativas. O Nuxt.JS, por sua vez, é um *framework* baseado no Vue.JS, que facilita a criação de aplicações *web* com renderização no lado do servidor, garantindo uma melhor performance e SEO.

6 Trabalhos Relacionados

Nesta seção, será apresentada uma análise de trabalhos acadêmicos e projetos que possuem relação com o tema abordado nesta monografia, especialmente no que diz respeito a parte de monitoramento ambiental e à utilização de tecnologias IoT para a coleta e tratamento de dados em tempo real.

6.1 Principais trabalhos encontrados

(CAVALCANTE, 2021) apresenta um sistema de monitoramento multiparamétrico da qualidade do ar em ambientes internos, utilizando sensores de baixo custo e tecnologias IoT. O sistema proposto é capaz de medir diversos parâmetros, como concentração de gases e dados meteorológicos, e enviar essas informações para um servidor central. O uso de sensores como o MQ-135 e módulos ESP8266 se destaca pela simplicidade e baixo custo, características importantes para soluções de monitoramento. O estudo também explora a importância da calibração dos sensores, apontando para a necessidade de garantir a precisão dos dados coletados, especialmente em ambientes fechados onde os parâmetros ambientais podem variar rapidamente. A metodologia utilizada por Cavalcante reforça a viabilidade de soluções de monitoramento baseadas em IoT com foco na acessibilidade e integração de dados.

(MUMBERGER, 2018) apresenta um sistema de monitoramento para ambientes industriais utilizando IoT, com ênfase em flexibilidade e escalabilidade. O trabalho se concentra no uso de LoRa e no protocolo MQTT, tecnologias utilizadas para a comunicação em ambientes onde a infraestrutura de rede pode ser limitada. O sistema proposto foi projetado para ser configurável, permitindo ajustes conforme as necessidades específicas de cada ambiente industrial. Testes em campo demonstraram a eficiência do sistema em distâncias variadas e com diferentes obstáculos, o que o torna aplicável para pequenos e médios ambientes industriais. A abordagem modular e configurável adotada é altamente relevante e visa oferecer uma solução flexível e de fácil implementação, com foco na escalabilidade para diferentes ambientes industriais.

(NICOLAS, 2024) apresenta um sistema IoT para monitoramento da qualidade do ar, especificamente para a startup IRIS. Este projeto possui uma certa escalabilidade combinando sensores de baixo custo com uma arquitetura distribuída que permite a integração de múltiplas fontes de dados. A solução proposta por foca na coleta, processamento e análise de dados em tempo real, além de oferecer recursos de calibração automática para garantir a precisão das medições. A integração de plataformas de referência e a capacidade de adaptar o sistema a diferentes condições ambientais fazem deste um estudo altamente aplicável a projetos de monitoramento em larga escala. A escolha de tecnologias, como o protocolo MQTT e sensores conectados ao microcontrolador ESP32, se alinha diretamente com os objetivos de buscar uma implementação de um sistema eficiente para o monitoramento da

qualidade do ar em ambientes industriais.

(OLIVEIRA et al., 2024) apresenta um sistema de monitoramento IoT voltado para plantas industriais. Utilizando módulos ESP8266 e o protocolo MQTT, o projeto propõe uma solução para o monitoramento em tempo real de sensores espalhados por uma planta industrial, com o objetivo de permitir a manutenção preditiva e aumentar a eficiência operacional. O sistema apresentado busca otimizar a coleta de dados de sensores de gás e temperatura, integrando essas informações em uma plataforma central de análise. A abordagem adotada foca em garantir a robustez da comunicação em uma rede industrial, o que é fundamental para o sucesso de sistemas de IoT em ambientes com interferências eletromagnéticas e grandes distâncias entre sensores e servidores. Este trabalho por mais que foque em outros tipos de problemas específicos, possui várias semelhanças que o traz para o mesmo caminho presente neste projeto e nos outros citados, a utilização de tecnologias IoT e sensores que permitem uma boa acessibilidade, integração de dados, flexibilidade e baixo custo.

6.2 Análise conclusiva

Os trabalhos analisados fornecem uma base sólida e se assemelham bastante em suas propostas em relação ao porjeto aqui envolvido. Evidencia-se a aplicabilidade de soluções IoT para o monitoramento de ambientes industriais e de qualidade do ar. Tecnologias como ESP8266, ESP32, LoRa, e MQTT são comumente utilizadas devido à sua eficiência em ambientes com restrições de infraestrutura de rede, alta demanda por escalabilidade e por ser economicamente viável. Além disso, é possível destacar que os trabalhos analisados demonstram a crescente aplicação das tecnologias da IoT em sistemas de monitoramento ambiental e industrial sendo possível observar uma convergência em termos de metodologias e tecnologias empregadas, como o uso de sensores de baixo custo, protocolos de comunicação leves e confiáveis, e a integração de dados em tempo real em plataformas centralizadas.

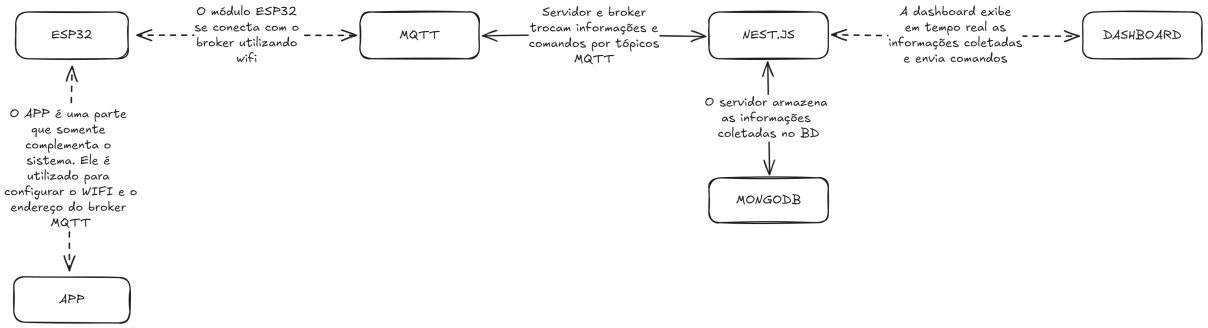
A partir das melhores práticas identificadas nos trabalhos revisados, o sistema proposto busca resolver os desafios de monitoramento em tempo real e integração de dados, garantindo que os parâmetros críticos sejam monitorados continuamente e de forma eficiente. Ao adotar essas tecnologias amplamente testadas e utilizadas, o projeto se posiciona como uma solução viável e robusta, pronta para ser adaptada a diferentes ambientes e escalas de operação.

7 Desenvolvimento

7.1 Arquitetura do projeto

Para permitir a escalabilidade e distribuição do projeto, foi elaborada a seguinte arquitetura:

Figura 1 – Diagrama de arquitetura do projeto



Fonte: Autoria própria

Essa arquitetura permite que os módulos ESP32 funcionem de forma independente, sem a necessidade de configurar endereços no lado do servidor, no cenário do projeto isso simplificaria a configuração e instalação de novos módulos em uma planta industrial grande (onde pelo menos 5 módulos seriam empregados), visto que seria necessário apenas uma configuração inicial (feita pelo aplicativo móvel nesse caso) que consiste apenas em informar o SSID da rede WiFi juntamente com a senha e o endereço IP do servidor (considerando que a aplicação do servidor estaria executando localmente). Todo o armazenamento, estruturação e exibição é de responsabilidade do *software* do servidor, que possui uma configuração simples e objetiva que será tratada no próximo capítulo.

7.2 Software do servidor

O *software* presente na aplicação do servidor é uma API REST desenvolvida no modelo Model-View-Controller-Service que integra o MongoDB com o MQTT e o Socket.IO para gerenciar todos os processos relacionados ao projeto no geral.

Para desenvolvimento desse *software* foi escolhido o Nest.JS, um *framework* focado em escalabilidade escrito em TypeScript com diversos módulos nativos que permitem o desenvolvimento mais rápido de aplicações de larga escala que requerem alta disponibilidade. Dentre esses módulos estão os módulos de conexão com o MongoDB, MQTT e Socket.IO.

Para capturar as informações dos módulos, o servidor utiliza um operador coringa asterísco (ou estrela) que permite separar os módulos em seus tópicos próprios, garantindo mais segurança e controle, visto que não seria um tópico só com diversos módulos enviando

ou recebendo diversos dados ao mesmo tempo. O tópico na qual os módulos publicarão os dados coletados possui o seguinte formato:

/sensors/*/**

O asterísco sozinho será substituído pelo identificador único do módulo, ficando algo parecido com:

/sensors/00000000-0000-0000-0000-000000000000/**

A partir disso o servidor sabe exatamente qual módulo foi responsável pela coleta da informação ou qual módulo executará a ação, visto que há tambem um tópico para comandos e um tópico para qual o módulo enviará pulsos indicando que ele está online.

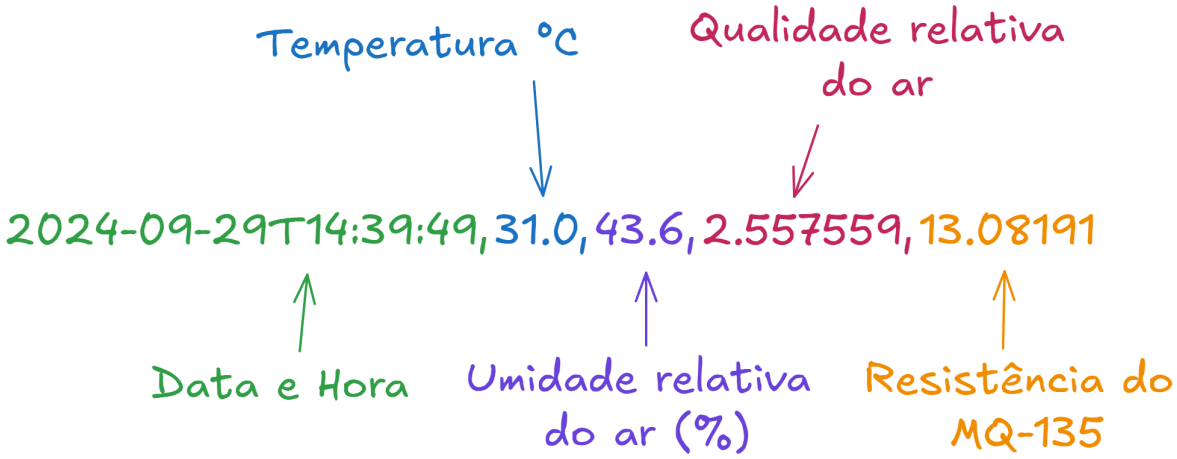
Os tópicos na qual o servidor se inscreverá são os seguintes:

- Recebimento das informações pelo módulo:

  /sensors/00000000-0000-0000-0000-000000000000/data

  As informações enviadas pelo módulo possuem o seguinte formato:

Figura 2 – Dados enviados pelo módulo para o tópico



Fonte: Autoria própria

- Recebimento dos pulsos pelo módulo:

  /sensors/00000000-0000-0000-0000-000000000000/state

- Recebimento do resultado da execução de algum comando:

  /sensors/00000000-0000-0000-0000-000000000000/commands/result

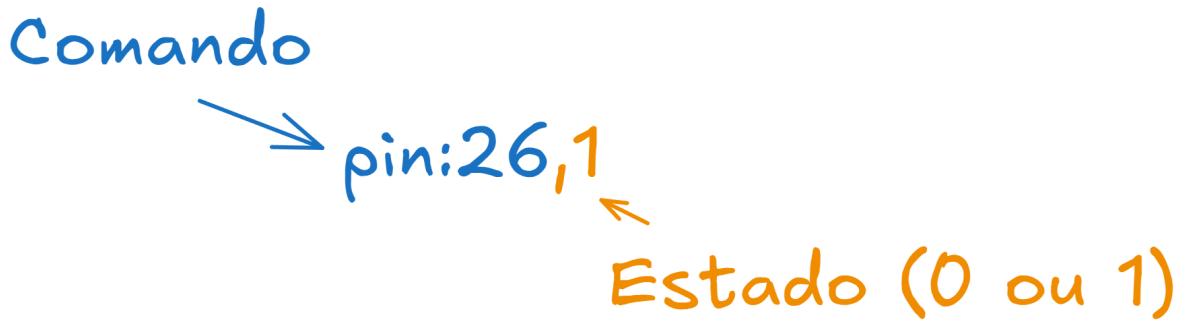
  Já os tópicos na qual o servidor publicará são os seguintes:

- Envio dos comandos e gatilhos:

  /sensors/00000000-0000-0000-0000-000000000000/commands

  Os comandos enviados pelo servidor possuem o seguinte formato:

  Figura 3 – Comando enviado pelo servidor para o tópico



  Fonte: Autoria própria

7.3 *Hardware* escolhido para os módulos

7.3.1 Microcontrolador

Para o *hardware* do módulo de detecção, foi escolhida a plataforma ESP32 devido à sua ampla gama de recursos, como conectividade Wi-Fi e Bluetooth de baixo consumo (*Bluetooth Low Energy*). Essas funcionalidades foram essenciais para o desenvolvimento do projeto. No entanto, durante o processo, foi identificado que o conversor ADC (*Analog-to-Digital Converter*) nativo do ESP32 apresentava imprecisões ao realizar leituras do sensor MQ-135. Para resolver esse problema, foi integrado o módulo ADS1115, que garantiu maior precisão nas leituras analógicas.

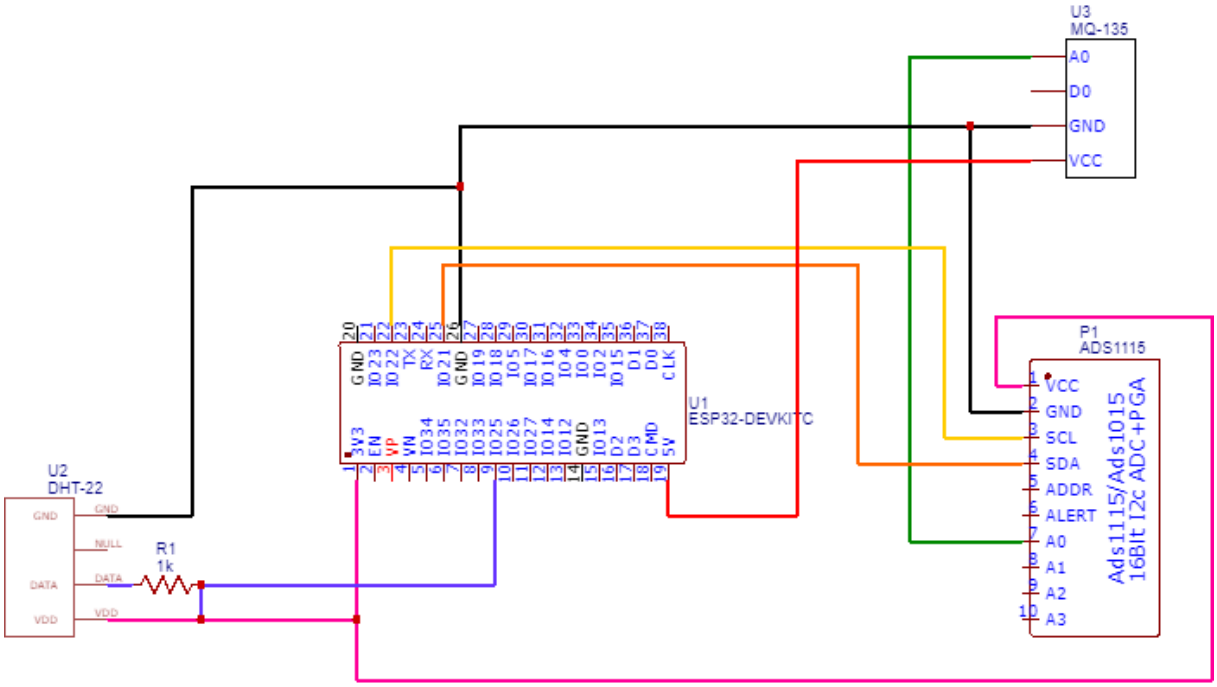
7.3.2 Sensor de temperatura e umidade

Para o sensor de temperatura e umidade foi escolhido o DHT22, por possuir uma boa precisão na medição de umidade e temperatura. Segundo a folha de especificações (do inglês, *datasheet*), a precisão na medição de temperatura é de 0.1 °C e na medição de umidade é de 0.1% na umidade relativa.

7.3.3 Sensor de gases

Para o sensor de gases foi escolhido inicialmente o MQ-135, isso por possuir uma boa faixa de detecção para gases tóxicos mais comuns em ambientes industriais como, Amoníaco, Álcool, Óxidos de Nitrogênio, fumaça, monóxido de carbono e dióxido de carbono. Esse sensor porém, requer calibração, essa calibração foi feita via *software*. Esse sensor possui uma saída analógica, que foi conectada à um ADC (conversor analógico para digital, do inglês, *analog to digital converter*) externo o ADS1115, visto que o ADC presente

Figura 4 – Circuito do Hardware



Fonte: Autoria própria

no ESP32 possui uma resolução muito baixa além de ser conhecido por não operar corretamente, o ADS1115 foi escolhido por possuir uma boa resolução por um preço mais acessível, permitindo a redução do custo final do projeto.

7.3.4 Módulo ADS1115

O ADS1115 é um conversor analógico-digital (ADC) externo de 16 *bits*, amplamente utilizado em aplicações que exigem alta precisão nas leituras de sinais analógicos. Ele foi integrado ao projeto para compensar as imprecisões observadas no ADC nativo do ESP32.

Esse módulo possui quatro canais de entrada e uma taxa de amostragem ajustável, o que permite maior flexibilidade e eficiência ao lidar com múltiplos sensores. Além disso, o ADS1115 utiliza comunicação I2C, o que facilita sua integração com microcontroladores como o ESP32. Sua resolução de 16 *bits* oferece uma melhoria significativa na qualidade das leituras, especialmente em sensores de gases, como o MQ-135, onde medições precisas são cruciais para o desempenho geral do sistema.

7.4 Software dos módulos

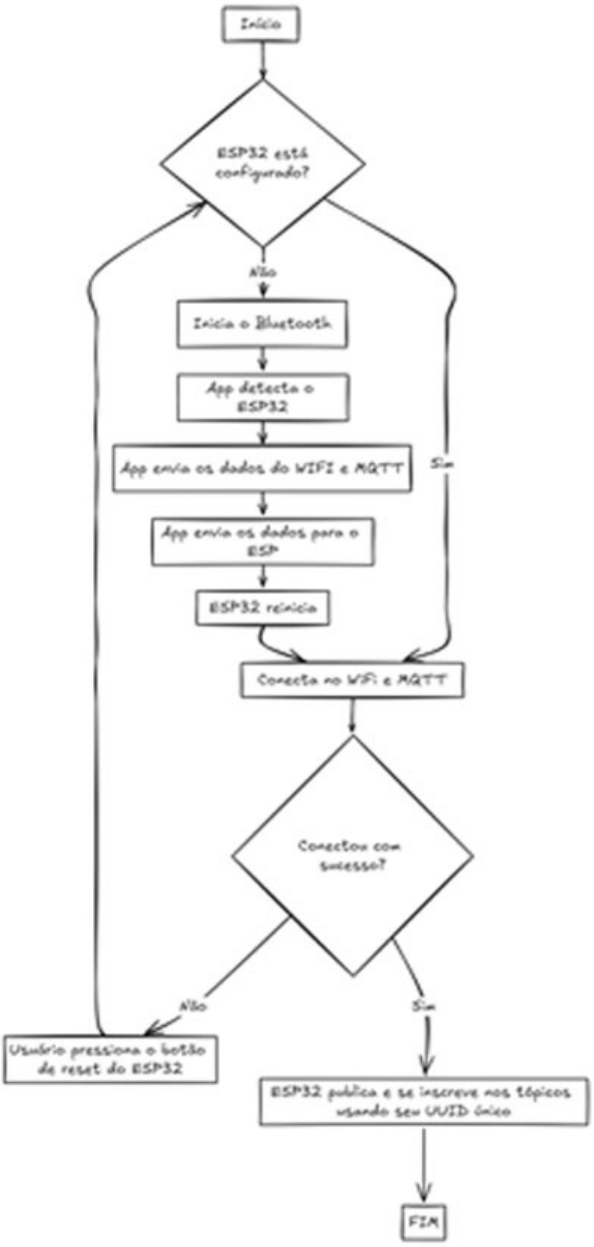
Para controlar o ESP32 presente nos módulos foi desenvolvido uma aplicação utilizando MicroPython, essa aplicação é responsável por gerenciar todas as funcionalidades do módulo, tais como:

- Configuração e conexão com a rede WiFi
- Conexão com o *broker* MQTT

- Coleta, processamento e envio dos dados
- Execução dos comandos enviados pelo servidor

A lógica do *software* presente no ESP32 segue o seguinte fluxograma:

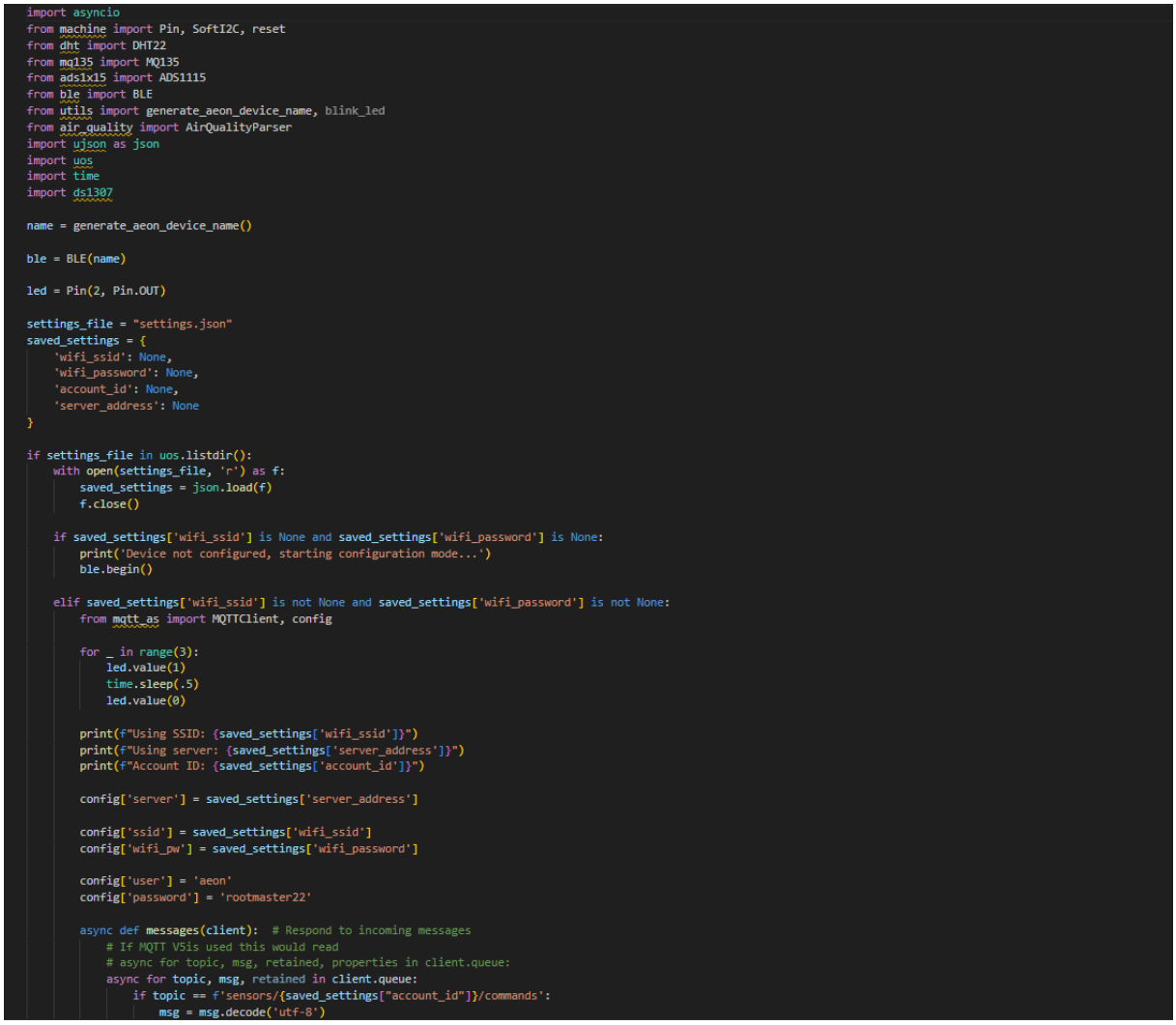
Figura 5 – Diagrama de estados do programa presente no módulo



Fonte: Autoria própria

O módulo se intercomunicará com o servidor utilizando o protocolo MQTT, isso significa que o módulo irá se inscrever e publicar nos mesmos tópicos na qual o servidor publicará e se inscreveu, ou seja, o módulo irá se inscrever em tópicos que o servidor publica e vice-versa.

Figura 6 – Exemplo de uma parte do código



```
import asyncio
from machine import Pin, SoftI2C, reset
from dht import DHT22
from mq135 import MQ135
from ads1x15 import ADS1115
from ble import BLE
from utils import generate_aeon_device_name, blink_led
from air_quality import AirQualityParser
import ujson as json
import uos
import time
import ds1307

name = generate_aeon_device_name()

ble = BLE(name)

led = Pin(2, Pin.OUT)

settings_file = "settings.json"
saved_settings = {
    'wifi_ssid': None,
    'wifi_password': None,
    'account_id': None,
    'server_address': None
}

if settings_file in uos.listdir():
    with open(settings_file, 'r') as f:
        saved_settings = json.load(f)
        f.close()

    if saved_settings['wifi_ssid'] is None and saved_settings['wifi_password'] is None:
        print('Device not configured, starting configuration mode...')
        ble.begin()

    elif saved_settings['wifi_ssid'] is not None and saved_settings['wifi_password'] is not None:
        from mqtt_as import MQTTClient, config

        for _ in range(3):
            led.value(1)
            time.sleep(.5)
            led.value(0)

        print(f"Using SSID: {saved_settings['wifi_ssid']}")
        print(f"Using server: {saved_settings['server_address']}")
        print(f"Account ID: {saved_settings['account_id']}")

        config['server'] = saved_settings['server_address']

        config['ssid'] = saved_settings['wifi_ssid']
        config['wifi_pw'] = saved_settings['wifi_password']

        config['user'] = 'aeon'
        config['password'] = 'rootmaster22'

        async def messages(client):  # Respond to incoming messages
            # If MQTT V5is used this would read
            # async for topic, msg, retained, properties in client.queue:
            async for topic, msg, retained in client.queue:
                if topic == f'sensors/{saved_settings["account_id"]}/commands':
                    msg = msg.decode('utf-8')
```

Fonte: Autoria própria

O código completo do exemplo acima foi desenvolvido para a parte da exibição dos dados em tempo real, realizando a coleta e análise de dados de múltiplos sensores e a comunicação desses dados com o servidor central. Abaixo, detalham-se os componentes principais do código e suas funções.

7.4.1 Importação de Bibliotecas e Configuração Inicial

O código utiliza diversas bibliotecas de sensores e comunicação para integrar os dispositivos:

- Bibliotecas de sensores: DHT22, MQ135, e ADS1115, entre outras, para medir temperatura, umidade e qualidade do ar.
- Comunicação: BLE para *Bluetooth Low Energy* e MQTTClient para a comunicação via protocolo MQTT.

- Utilitários: A biblioteca *"ujson"* é usada para manipulação de JSON, facilitando o armazenamento e carregamento das configurações do dispositivo.

7.4.2 Configuração de Parâmetros Salvos

Ao iniciar, o código verifica se um arquivo de configuração *settings.json* está presente. Caso o arquivo exista, ele carrega as credenciais de rede e configurações de servidor. Caso contrário, entra em modo de configuração, utilizando BLE para definir os parâmetros.

7.4.3 Comunicação MQTT

Após carregar as configurações, o dispositivo configura e se conecta a um servidor MQTT, utilizando o protocolo para enviar os dados coletados:

- Assinatura de Tópicos: O dispositivo assina os tópicos de *"heartbeat"* e *"commands"*, que permitem verificar a conexão do dispositivo e enviar comandos remotamente, respectivamente.
- Publicação de Dados: A cada intervalo de coleta, o dispositivo publica os dados dos sensores no tópico configurado.

7.4.4 Função de Coleta de Dados (*Main Loop*)

No *loop* principal *(main(client)*), o dispositivo executa as seguintes etapas para obter e enviar os dados:

- Leitura dos Sensores:
- DHT22: Utilizado para obter temperatura e umidade.
- MQ135: Utilizado para medir a qualidade do ar. A resistência do sensor é medida, e concentrações de gases (como CO2, NH3, e CO) são calculadas.
- Correção de Leitura: As leituras de concentração de gases são ajustadas com base na temperatura e umidade para maior precisão.
- Publicação dos Dados: Os valores de temperatura, umidade, e concentrações ajustadas de gases são enviados ao servidor via MQTT. Esses dados são formatados e publicados no tópico de dados, possibilitando o monitoramento em tempo real.

7.4.5 Gestão de Conectividade e Comandos

O código inclui rotinas para gerenciar a conectividade:

- Função *up*(*client*): Esta função é chamada quando a conexão é restabelecida, reassinando os tópicos e atualizando o estado do dispositivo.

- Função messages(*client*): Processa mensagens recebidas no tópico de comandos. O dispositivo responde a comandos para controle de pinos e estado dos pinos, permitindo controle remoto de dispositivos conectados.

7.4.6 Estrutura Assíncrona com asyncio

O código faz uso de programação assíncrona com asyncio para gerenciar múltiplas tarefas de forma eficiente, como leitura dos sensores, envio de dados e processamento de comandos, permitindo que o dispositivo execute várias operações em paralelo sem atrasos perceptíveis.

7.4.7 Publicação e Controle de Eventos

A cada ciclo, os dados coletados são publicados em um tópico MQTT específico para o dispositivo, com dados formatados em uma estrutura JSON. Além disso, LEDs e outros pinos podem ser controlados remotamente, oferecendo uma interface de controle de baixo nível para testes e ajustes.

7.4.8 Conclusão

O código implementa uma solução para coleta e monitoramento de dados ambientais, utilizando sensores de temperatura, umidade e qualidade do ar, integrados a uma plataforma de comunicação MQTT. A estrutura assíncrona e a capacidade de comando remoto oferecem flexibilidade e eficiência, essenciais para um sistema de monitoramento industrial em tempo real.

8 Resultados

Os resultados alcançados durante o desenvolvimento demonstram a viabilidade dessa solução IoT proposta para monitoramento ambiental em tempo real. Com os módulos de coleta e comunicação devidamente configurados, os dados dos sensores foram transmitidos ao servidor de forma estável, permitindo que as informações sobre a qualidade do ar fossem visualizadas em tempo real na *Dashboard*. Esse sistema facilita o monitoramento contínuo traz a possibilidade para que ações possam ser tomadas visando a solução de algum possível problema gerado de condições adversas no ambiente que foram detectadas.

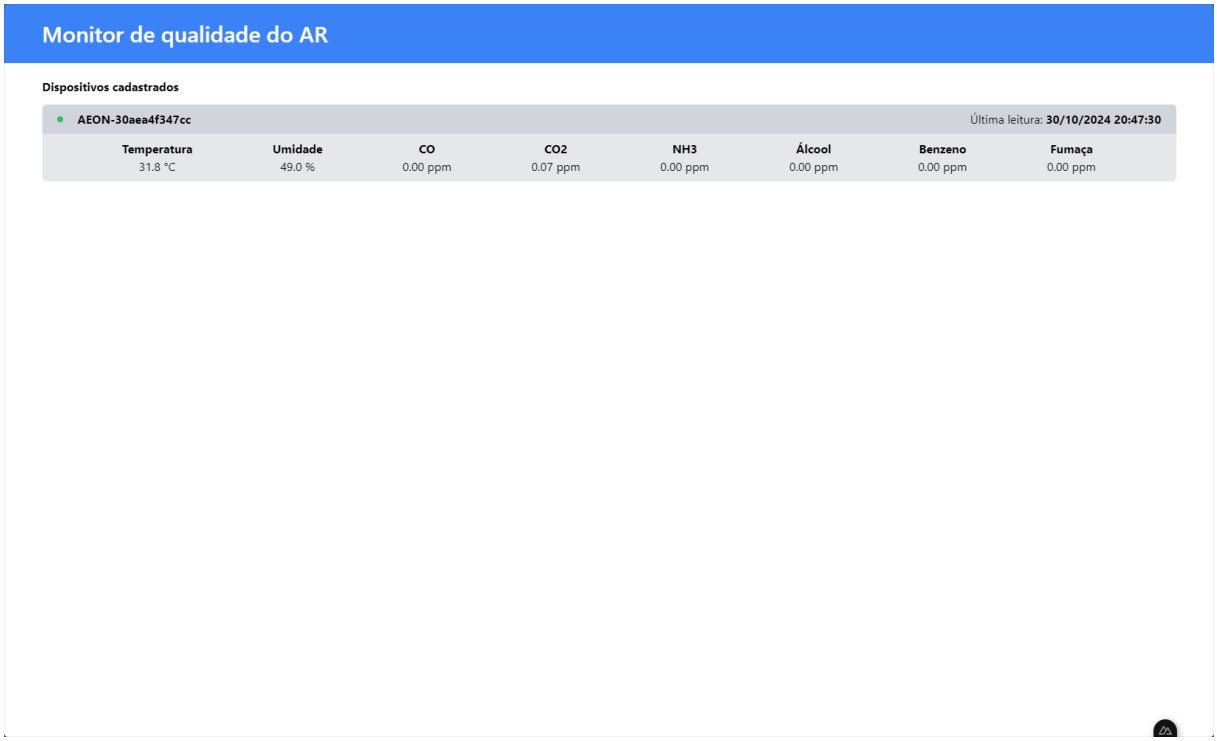
8.1 Testes de Coleta e Transmissão de Dados

Durante a fase de testes, os módulos foram configurados para realizar leituras contínuas dos sensores de temperatura, umidade e gases (como monóxido de carbono e amônia) e enviar os dados para o servidor por meio do protocolo MQTT. Os testes de transmissão confirmaram a eficiência do protocolo em redes de baixa largura de banda, demonstrando que o sistema é capaz de operar com estabilidade e precisão. A utilização do módulo ADS1115 contribuiu para melhorar a qualidade das leituras analógicas, principalmente para o sensor MQ-135, minimizando o ruído e proporcionando uma maior precisão dos dados transmitidos.

8.2 Visualização e Alerta em Tempo Real

A d*dashboard*, desenvolvida com o framework Vue.js, mostrou-se intuitiva e responsiva, permitindo a visualização de dados ambientais em tempo real. Em condições normais, a interface exibe os valores dentro dos limites configurados, mas, quando os níveis de poluentes excedem o valor seguro, alertas visuais são acionados na *dashboard* para sinalizar a necessidade de intervenção. Essa funcionalidade foi validada com a simulação de condições de alta concentração de monóxido de carbono, resultando em respostas de alerta instantâneas, o que confirma a eficácia da solução para situações críticas.

Figura 7 – *Dashboard* em um ambiente com níveis normais de CO2

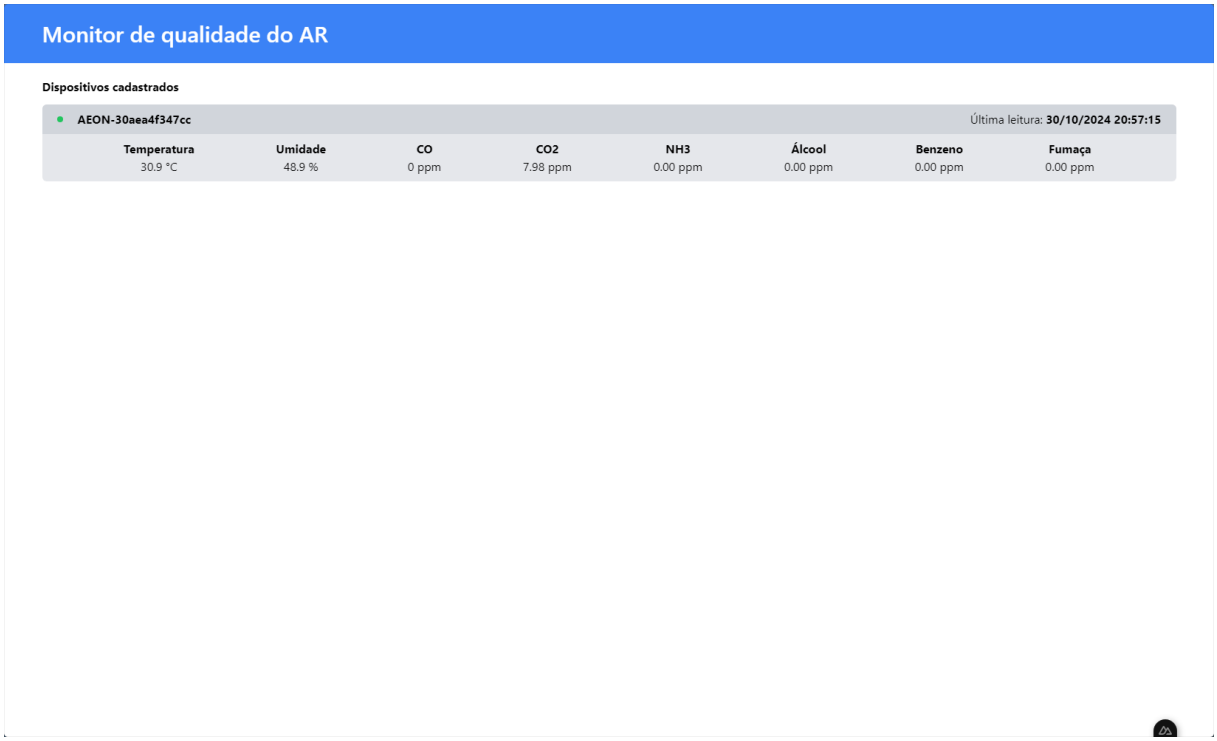


Fonte: Autoria própria

8.3 Desempenho e Escalabilidade

Os testes de desempenho indicaram que o sistema é escalável, podendo suportar a adição de novos módulos de monitoramento sem degradação significativa da comunicação ou da integridade dos dados transmitidos. A utilização do protocolo MQTT em uma arquitetura distribuída, conforme planejado, permitiu que vários módulos operassem de maneira autônoma, enquanto o servidor gerenciava centralizadamente as informações recebidas. Esse resultado é particularmente importante para aplicações industriais em larga escala, onde a cobertura de grandes áreas e a inclusão de vários pontos de monitoramento são requisitos essenciais.

Figura 8 – *Dashboard* em um caso onde os níveis de CO2 estão altos



Fonte: Autoria própria

8.4 Exibição de Dados em Tempo Real

O nível de monitoramento apresentado proporciona uma base sólida para a análise histórica de dados e potencial integração com outras ferramentas de análise e controle.

Figura 9 – Exibição de Dados em Tempo Real

readings

	device	device_date	temperature	humidity	ppm_alcohol	ppm_benzene	ppm_co	ppm_co2	ppm_nh3	ppm_smoke	update
1	AEON-30aea4f347cc	2024-10-30T23:50:00.000Z	27.6	78.2	0	0	0	0.02	0	0	2024-10-
2	AEON-30aea4f347cc	2024-10-30T23:51:20.000Z	29.8	79.9	0	0	0	0.02	0	0	2024-10-
3	AEON-30aea4f347cc	2024-10-30T23:51:40.000Z	27.6	78.1	0	0	0	0.02	0	0	2024-10-
4	AEON-30aea4f347cc	2024-10-30T23:52:00.000Z	27.6	78	0	0	0	1	0	0	2024-10-
5	AEON-30aea4f347cc	2024-10-30T23:52:20.000Z	27.6	78	0	0	0	1.15	0	0	2024-10-
6	AEON-30aea4f347cc	2024-10-30T23:52:40.000Z	28.6	79	0	0	0	1.78	0	0	2024-10-
7	AEON-30aea4f347cc	2024-10-30T23:53:00.000Z	27.6	78.3	0	0	0	2.24	0	0	2024-10-
8	AEON-30aea4f347cc	2024-10-30T23:53:20.000Z	27.6	78.4	0	0	0	2.65	0	0	2024-10-
9	AEON-30aea4f347cc	2024-10-30T23:53:40.000Z	27.6	78.6	0	0	0	3.01	0	0	2024-10-
10	AEON-30aea4f347cc	2024-10-30T23:54:00.000Z	27.6	78.6	0	0	0	3.52	0	0	2024-10-
11	AEON-30aea4f347cc	2024-10-30T23:54:20.000Z	27.6	78.3	0	0	0	3.78	0	0	2024-10-
12	AEON-30aea4f347cc	2024-10-30T23:54:40.000Z	27.6	77.9	0	0	0	4.22	0	0	2024-10-
13	AEON-30aea4f347cc	2024-10-30T23:55:00.000Z	27.6	77.8	0	0	0	4.5	0	0	2024-10-
14	AEON-30aea4f347cc	2024-10-30T23:55:20.000Z	27.6	78	0	0	0	5	0	0	2024-10-
15	AEON-30aea4f347cc	2024-10-30T23:55:40.000Z	27.6	77.8	0	0	0	5.34	0	0	2024-10-
16	AEON-30aea4f347cc	2024-10-30T23:56:00.000Z	27.7	78.1	0	0	0	5.79	0	0	2024-10-
17	AEON-30aea4f347cc	2024-10-30T23:56:20.000Z	27.7	78.1	0	0	0	6.11	0	0	2024-10-
18	AEON-30aea4f347cc	2024-10-30T23:56:40.000Z	27.7	77.9	0	0	0	6.56	0	0	2024-10-
19	AEON-30aea4f347cc	2024-10-30T23:57:00.000Z	27.7	77.8	0	0	0	6.89	0	0	2024-10-
20	AEON-30aea4f347cc	2024-10-30T23:57:20.000Z	27.7	77.6	0	0	0	7.12	0	0	2024-10-
21	AEON-30aea4f347cc	2024-10-30T23:57:40.000Z	27.7	77.7	0	0	0	7.57	0	0	2024-10-
22	AEON-30aea4f347cc	2024-10-30T23:58:00.000Z	27.7	77.7	0	0	0	7.89	0	0	2024-10-
23	AEON-30aea4f347cc	2024-10-30T23:58:20.000Z	27.7	77.5	0	0	0	7.98	0	0	2024-10-
24	AEON-30aea4f347cc	2024-10-30T23:58:40.000Z	27.7	77.5	0	0	0	7.98	0	0	2024-10-
25	AEON-30aea4f347cc	2024-10-30T23:59:00.000Z	27.7	77.6	0	0	0	7.98	0	0	2024-10-
26	AEON-30aea4f347cc	2024-10-30T23:50:16.000Z	27.7	77.5	0	0	0	7.98	0	0	2024-10-
27	AEON-30aea4f347cc	2024-10-30T23:50:16.000Z	27.7	77.7	0	0	0	7.98	0	0	2024-10-
28	AEON-30aea4f347cc	2024-10-30T23:50:16.000Z	27.7	77.9	0	0	0	7.54	0	0	2024-10-
29	AEON-30aea4f347cc	2024-10-30T23:50:16.000Z	27.7	77.7	0	0	0	7.32	0	0	2024-10-
30	AEON-30aea4f347cc	2024-10-30T23:50:16.000Z	27.7	77.6	0	0	0	6.98	0	0	2024-10-

Fonte: Autoria própria

A imagem acima mostra uma interface de monitoramento em tempo real, exibindo os dados coletados por um dispositivo de sensoriamento ambiental. Esta interface é organizada em uma tabela, com cada linha representando uma nova leitura capturada pelo dispositivo em intervalos de tempo específicos.

Os principais campos monitorados incluem:

- Dispositivo (*"device"*): identifica o código único do dispositivo responsável pela coleta, permitindo o monitoramento de múltiplos dispositivos em diferentes locais.
- Data da Coleta (*"device-date"*): indica a data e o horário exatos em que cada leitura foi realizada, sendo essencial para análise temporal e correlação dos dados.
- Temperatura (*"temperature"*): apresenta a temperatura ambiente em graus Celsius, monitorada em tempo real para detectar flutuações e padrões que podem ser importantes para o controle de processos industriais.
- Umidade (*"humidity"*): exibe a umidade relativa do ar, um parâmetro relevante em ambientes que exigem controle preciso de umidade para otimizar processos ou preservar materiais.

  Concentrações de Gases: incluem:

- Álcool (*"ppm-alcohol"*),
- Benzeno (*"ppm-benzene"*),
- Monóxido de Carbono (*"ppm-co"*),
- Dióxido de Carbono (*"ppm-co2"*),
- Amônia (*"ppm-nh3"*),
- Fumaça (*"ppm-smoke"*).

  Estes valores são medidos em partes por milhão (*"ppm"*), e refletem a presença de cada substância no ambiente, essencial para monitorar a qualidade do ar em termos de poluentes específicos. A monitoração de concentrações de gases é útil para entender como diferentes processos industriais impactam o ambiente local.

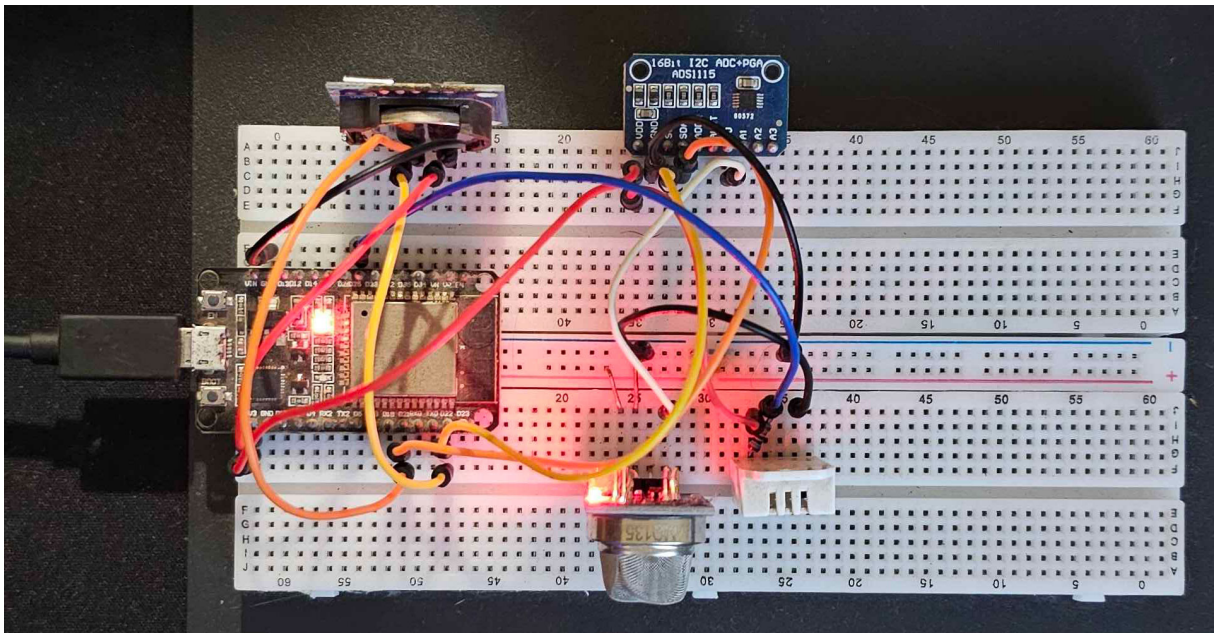
- Data de Atualização (*"update-date"*): mostra o horário em que os dados foram atualizados no sistema, permitindo o controle de integridade e sincronização das informações.

Esse sistema foi configurado para coletar e registrar os dados automaticamente, utilizando um banco de dados em tempo real que garante acesso instantâneo às informações. A interface permite a visualização e análise contínua dos dados, o que pode ajudar a gerar insights sobre os padrões e a dinâmica ambiental em ambientes industriais, favorecendo a otimização de processos e ajustes automáticos quando necessário.

8.5 Análise Geral dos Resultados

Os resultados gerais evidenciam que o sistema desenvolvido atende aos objetivos estabelecidos para a coleta e análise de dados de qualidade do ar em ambientes industriais, com precisão e capacidade de resposta. A integração entre sensores, servidor e dashboard foi eficiente, e a solução mostrou-se funcional para diferentes tipos de indústrias, especialmente aquelas que requerem monitoramento contínuo de variáveis ambientais. A implementação de tecnologias como MQTT, ADS1115 e a estruturação modular garantiram que o sistema fosse viável, escalável e adaptável, concluindo assim a prova de conceito com sucesso.

Figura 10 – Vista superior do protótipo montado em uma *Protoboard*



Fonte: Autoria própria

9 Conclusão

9.1 Considerações Finais

O presente trabalho teve como objetivo desenvolver um sistema de monitoramento de qualidade do ar em ambientes industriais, utilizando tecnologias de Internet das Coisas (IoT) para garantir a coleta e o controle eficiente de dados ambientais em tempo real. Desde a fase inicial de pesquisa e concepção do sistema, até a implementação e os testes finais, foi possível validar a eficácia da solução proposta em fornecer dados precisos e permitir a automação de respostas em situações de risco.

A implementação de sensores de temperatura, umidade e qualidade do ar, integrados a uma arquitetura distribuída e a uma plataforma de monitoramento baseada em MQTT, comprovou-se eficaz para capturar e transmitir informações de maneira confiável. O protocolo MQTT, escolhido por sua leveza e estabilidade, foi essencial para assegurar a comunicação contínua entre os módulos de sensores e o servidor central, mesmo em redes de baixa largura de banda, comuns em ambientes industriais. Além disso, a utilização de uma interface intuitiva e responsiva desenvolvida em Vue.js permitiu que os dados fossem visualizados e gerenciados em tempo real, tornando o sistema acessível e facilitando a operação por parte dos usuários.

Os testes realizados indicaram que o sistema desenvolvido é viável, escalável e adaptável para diferentes ambientes industriais. A capacidade de configurar limites e personalizar alarmes e respostas automáticas na *dashboard* trouxe flexibilidade e versatilidade ao projeto, permitindo que ele fosse ajustado para atender a diferentes necessidades e protocolos de segurança. A funcionalidade de acionamento de dispositivos externos, como exaustores e ventiladores, demonstrou o potencial do sistema para atuar de forma proativa e automatizada, reduzindo o tempo de resposta e mitigando riscos à saúde e à segurança dos trabalhadores.

No entanto, alguns desafios foram identificados durante o desenvolvimento e a validação do sistema. A calibração precisa dos sensores foi uma questão crucial para garantir a qualidade dos dados, e futuras melhorias podem considerar o uso de algoritmos de calibração automática para aprimorar ainda mais a precisão. Além disso, a expansão do sistema para plantas industriais de grande porte pode exigir adaptações para suportar um número maior de módulos de monitoramento sem comprometer a qualidade da comunicação e o processamento dos dados.

Em conclusão, o projeto alcançou seus objetivos ao oferecer uma solução eficaz e acessível para o monitoramento de qualidade do ar em ambientes industriais. A utilização de tecnologias de IoT e uma arquitetura modular permitiu a criação de um sistema robusto, adaptável e com potencial de aplicação em diferentes setores da indústria. Para trabalhos futuros, recomenda-se a implementação de novos sensores para a detecção de outros po-

luentes, além do desenvolvimento de algoritmos de aprendizado de máquina para prever condições críticas com base em padrões históricos de dados, promovendo um sistema ainda mais proativo e inteligente.

9.2 Trabalhos futuros

O desenvolvimento deste sistema abre diversas possibilidades para aprimoramentos e pesquisas futuras que podem expandir sua funcionalidade e aplicabilidade. Uma das melhorias potenciais está na integração de novos sensores que permitam a detecção de outros poluentes e partículas em suspensão, como dióxido de enxofre, partículas finas e compostos orgânicos voláteis. Essa ampliação da abrangência dos sensores tornaria o sistema aplicável a setores industriais específicos, como indústrias químicas, petroquímicas e de processamento de metais, que emitem poluentes variados.

Outro avanço importante seria a implementação de algoritmos de calibração automática, considerando que a calibração dos sensores é essencial para a confiabilidade dos dados coletados. Algoritmos de aprendizado de máquina poderiam ser utilizados para identificar desvios de calibração ao longo do tempo, garantindo precisão nos dados, mesmo em condições adversas ou com o uso prolongado dos sensores.

A aplicação de inteligência artificial também abre caminho para que o sistema seja capaz de detectar anomalias e prever situações de risco com base em padrões de dados históricos. Um sistema preditivo permitiria que o monitoramento ambiental identificasse potenciais situações críticas antes mesmo que elas ocorram, tornando a solução mais proativa e confiável.

Para grandes plantas industriais, a expansão para redes IoT de longo alcance, como o LoRaWAN, poderia otimizar a cobertura de áreas amplas, especialmente em locais onde a infraestrutura de rede é limitada. Isso permitiria que o sistema operasse com estabilidade em longas distâncias sem a necessidade de uma rede local extensa, viabilizando a comunicação em tempo real e a coleta de dados em áreas remotas.

O desenvolvimento de um aplicativo móvel também poderia facilitar o monitoramento, permitindo que os gestores e responsáveis acompanhem as condições ambientais de qualquer local. Um aplicativo proporcionaria mobilidade e agilidade na resposta a emergências, além de melhorar o acesso às informações críticas de segurança e operação.

Uma melhoria funcional importante para o futuro é a implementação de gatilhos automáticos para o acionamento de dispositivos externos, como exaustores e ventiladores, sempre que parâmetros ambientais ultrapassarem limites seguros. Esse recurso acrescentaria uma camada de automação ao sistema, permitindo que medidas preventivas e até preditivas sejam tomadas sem a necessidade de intervenção humana, aumentando a eficiência e a segurança em ambientes industriais.

Referências bibliográficas

ALVAREZ, D.; ANTUNES, F. *AUTOMAÇÃO RESIDENCIAL UTILIZANDO BLUETOOTH, ETHERNET E SMARTPHONE*. UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ, 2015. Disponível em: <https://riut.utfpr.edu.br/jspui/bitstream/1/9483/3/CT_COMET_2014_2_05.pdf>. Acesso em: 02/11/2024.

ANDRADE, E. *MicroPython: Python para microcontrolador - Embarcados*. 2016. Disponível em: <https://www.embarcados.com.br/micropython/>. Acesso em: 15/10/2024.

BRAGA, L.; MARQUES, A. *O que é Wi-Fi? Saiba como funciona a tecnologia de rede sem fio • Tecnoblog*. 2023. Disponível em: <https://tecnoblog.net/responde/o-que-e-wi-fi-como-funciona/>. Acesso em: 12/10/2024.

BRASIL, M. d. S. *DOENÇAS RELACIONADAS AO TRABALHO*. Ministério da Saúde do Brasil, 2001. Disponível em: <https://bvsms.saude.gov.br/bvs/publicacoes/doencas_relacionadas_trabalho1.pdf>. Acesso em: 25/10/2024.

BRITO, L. et al. PROTOCOLOS DE COMUNICAÇÃO PARA INTERNET OF THINGS (IOT). *Intercursos Revista Científica*, v. 17, n. UEMG, p. 73, 2018. ISSN 2179-9059. Disponível em: <https://revista.uemg.br/index.php/intercursosrevistacientifica/article/download/3712/2089/11948>.

CAVALCANTE, T. *Desenvolvimento de sistema de monitoramento multiparamétrico da qualidade do ar em ambientes internos*. PUC-Rio, 2021. Disponível em: <https://www.maxwell.vrac.puc-rio.br/53185/53185.PDF>. Acesso em: 09/11/2024.

CHAN, M. *SQL vs. NoSQL - what's the best option for your database needs?* 2019. Disponível em: <https://www.thorntech.com/sql-vs-nosql/>. Acesso em: 11/10/2024.

CRAVO, E. *Protocolo MQTT: como funciona, dicas e informações importantes*. 2024. Disponível em: <https://blog.kalatec.com.br/protocolo-mqtt/>. Acesso em: 09/11/2024.

FERNANDES, D. Journal, *Git & Github: O que é? Por que? Como iniciar?* 2018. Publisher: Rocketseat. Disponível em: <https://blog.rocketseat.com.br/iniciando-com-git-github/>. Acesso em: 11/10/2024.

GUEDES, M. *O que é MongoDB?* 2020. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-mongodb>. Acesso em: 11/10/2024.

IBM, R. blog. *O que é a Internet das Coisas (IoT)? | IBM*. 2024. Disponível em: <https://www.ibm.com/br-pt/topics/internet-of-things>. Acesso em: 23/10/2024.

LATEEF, Z. *What is JavaScript? | JavaScript Programming*. 2018. Disponível em: <https://www.edureka.co/blog/what-is-javascript/>. Acesso em: 11/10/2024.

MUMBERGER, L. *SISTEMA DE MONITORAMENTO DE AMBIENTES INDUSTRIAIS CONFIGURÁVEL BASEADO EM INTERNET DAS COISAS*. UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS UNIDADE ACADÊMICA DE GRADUAÇÃO, 2018. Disponível em: <https://repositorio.jesuita.org.br/handle/UNISINOS/11646>. Acesso em: 10/11/2024.

NICOLAS, R. *DESENVOLVIMENTO DE UM SISTEMA IOT DE MONITORAMENTO DA QUALIDADE DO AR PARA A STARTUP IRIS*. UNIVERSIDADE FEDERAL DE SANTA CATARINA, 2024. Disponível em: <https://repositorio.ufsc.br/handle/123456789/256020>. Acesso em: 01/11/2024.

NOLETO, C. *Typescript: o que é, principais conceitos e porquê usar!* 2020. Disponível em: <https://blog.betrybe.com/desenvolvimento-web/typescript/>.

OLIVEIRA, E. et al. *MONITORAMENTO INDUSTRIAL UTILIZANDO IoT*. FACULDADE DE TECNOLOGIA DE SÃO BERNARDO DO CAMPO "ADIB MOISÉS DIB", 2024. Disponível em: <https://ric.cps.sp.gov.br/bitstream/123456789/23065/1/Monografia-CORRIGIDA-FINAL%20-%20PDF.pdf>. Acesso em: 30/10/2024.

PAVÃO, G. *A Internet das Coisas (IoT) como pilar fundamental para impulsionar a Indústria 4.0 no Brasil*. 2024. Disponível em: <https://cryptoid.com.br/iot-internet-das-coisas/a-internet-das-coisas-iot-como-pilar-fundamental-para-impulsionar-a-industria-4-0-no-brasil/>. Acesso em: 09/11/2024.

RAMOS, D. *A IMPORTÂNCIA DA SEGURANÇA DA INFORMAÇÃO EM REDES WIFI*. Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, 2020. Disponível em: <https://repositorio.pucgoias.edu.br/jspui/bitstream/123456789/452/1/TCC%202%20-%20Diego%20Mendes%20Ramos-%20Final.pdf>. Acesso em: 14/11/2024.

RANGEL, J. *O que é NestJS? E para que serve NestJS?* 2023. Publisher: Hcode. Disponível em: <https://hcode.com.br/blog/o-que-e-nestjs-e-para-que-serve-nestjs>. Acesso em: 24/10/2024.

REDHAT. Blog, *O que é API REST?* 2018. Publisher: RedHat. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>. Acesso em: 11/09/2024.

ROVEDA, U. *O que é Python, para que serve e por que aprender?* 2020. Disponível em: <https://kenzie.com.br/blog/o-que-e-python/>.

SOUZA, J. *MICROPYTHON: LINGUAGEM PYTHON APLICADA EM MICROCONTROLADORES - InQ.IFBA*. 2024. Disponível em: <https://inq.conquista.ifba.edu.br/v1/micropython-linguaem-python-aplicada-em-microcontroladores/>. Acesso em: 15/11/2024.

SOUZA, R. et al. Uma arquitetura para iot direcionada à ciência do contexto baseada em eventos distribuídos. In: *Anais do VIII Simposio Brasileiro de Computacao Ubiqua e Pervasiva*. Porto Alegre, RS, Brasil: SBC, 2016. p. 1206–1215. ISSN 2595-6183. Disponível em: <https://sol.sbc.org.br/index.php/sbcup/article/view/9469>. Acesso em: 14/10/2024.

SOUZA, *Arquitetura MVC: Entendendo o Modelo-Visão-Controlador*. 2023. Publisher: DIO. Disponível em: <https://www.dio.me/articles/arquitetura-mvc-entendendo-o-modelo-visao-controlador>. Acesso em: 12/11/2024.

TERA AMBIENTAL. *Monitoramento ambiental: qual a importância?* 2024. Publisher: Tera Ambiental. Disponível em: <https://www.teraambiental.com.br/blog-da-tera-ambiental/monitoramento-ambiental>. Acesso em: 14/10/2024.

ZAMBELLI, R. *O que são sensores IoT e quais são seus benefícios?* 2023. Disponível em: <https://checklistfacil.com/blog/sensores-iot/>. Acesso em: 10/11/2024.

ZOLETT, D.; RAMIREZ, A. Desenvolvimento de uma Interface de Monitoração Remota para o Sistema Robótico ROBIX, Integrando o Protocolo MQTT e oROS. *XI Computer on the Beach*, v. 11, n. UNIVALI-SC, p. 8, abr. 2020. Disponível em:

<https://periodicos.univali.br/index.php/acotb/article/view/16799/9523>. Acesso em: 10/10/2024.