INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO MARANHÃO CAMPUS MONTE CASTELO

RESIDÊNCIA TECNOLÓGICA EM SISTEMAS EMBARCADOS

MANOEL FELIPE COSTA FURTADO

SISTEMA DE AQUISIÇÃO, CONTROLE E ANÁLISE DE DADOS IoT

São Luís – MA

2025

MANOEL FELIPE COSTA FURTADO

SISTEMA DE AQUISIÇÃO, CONTROLE E ANÁLISE DE DADOS IoT

Trabalho de Conclusão de Curso da Residência Tecnológica em Sistemas Embarcados no Instituto Federal de Educação, Ciência e Tecnologia do Maranhão, Campus Monte Castelo como requisito para obtenção do certificado de conclusão do curso.

Orientado: Prof. Doutor Francisco Borges Carreiro.
Coorientador:

São Luís – MA

2025

MANOEL FELIPE COSTA FURTADO

SISTEMA DE AQUISIÇÃO, CONTROLE E ANÁLISE DE DADOS IoT

Trabalho de Conclusão de Curso da Residência Tecnológica em Sistemas Embarcados do Instituto Federal de Educação, Ciência e Tecnologia do Maranhão, como pré-requisito para obtenção do certificado de conclusão.

Aprovada em: _____ de _____ de _____.

BANCA EXAMINADORA

_____

Prof. Dr. Almir Souza e Silva Neto

Doutor em ANTENAS DE MICROFITA PARA 4G, 5G E ARRANJO DE ANTENAS CILÍNDRICAS.

Instituto Federal de Educação, Ciência e Tecnologia (IFMA)

_____

Prof. Matheus Silva Pestana (Examinador 1)

Mestre em Engenharia Elétrica

Instituto Federal de Educação, Ciência e Tecnologia (IFMA)

_____

João Victor Costa Silva (Examinador 2)

Graduando em Engenharia Elétrica

Instituto Federal de Educação, Ciência e Tecnologia (IFMA)

Resumo

Este trabalho propõe o desenvolvimento de uma arquitetura modular e distribuída, baseada na integração entre sistemas embarcados e a Internet das Coisas (IoT), com suporte à computação em borda (edge computing), voltada para aplicações de monitoramento, controle e análise de dados em tempo quase real. A motivação parte da crescente demanda por soluções tecnológicas que conciliem baixo custo, autonomia, flexibilidade e escalabilidade, especialmente em contextos com infraestrutura de rede limitada, intermitente ou inexistente. A dependência de serviços em nuvem, comum em arquiteturas IoT convencionais, impõe desafios relevantes relacionados à latência, segurança, confiabilidade e custo operacional.

A proposta busca superar essas limitações por meio da construção de um ecossistema local, capaz de operar de forma autônoma. O núcleo da arquitetura é composto por uma rede de microcontroladores (ex: Raspberry Pi Pico W), com sensores e atuadores programados em linguagem C, conectados a um servidor local que abriga um broker MQTT (Mosquitto), um banco de dados relacional (PostgreSQL) ou não relacional (NoSQL), e um dashboard analítico desenvolvido em Python com o uso de aprendizado de máquina (machine learning). A comunicação entre os dispositivos ocorre via Wi-Fi e protocolo MQTT, utilizando mensagens no formato JSON, o que garante leveza, interoperabilidade e eficiência. O sistema opera em um ciclo contínuo de coleta de dados, análise local e ação automatizada sobre o ambiente físico, com possibilidade de decisões baseadas em regras ou modelos preditivos.

O projeto foi implementado em etapas iterativas, desde a configuração dos dispositivos até a visualização dos dados em tempo real. Testes realizados em ambiente controlado demonstraram a viabilidade técnica da solução, destacando sua capacidade de operar com baixa latência, elevada confiabilidade e boa adaptabilidade. Os resultados indicam o potencial da arquitetura para ser aplicada em ambientes residenciais, industriais, urbanos e agrícolas, contribuindo para a democratização do acesso à automação inteligente com controle local e resposta autônoma, mesmo em cenários de baixa complexidade e infraestrutura restrita.

Palavras-chave:

IoT; Sistemas Embarcados; Computação em Borda (Edge Computing); MQTT; Automação Inteligente.

Abstract

This work proposes the development of a modular and distributed architecture based on the integration of embedded systems and the Internet of Things (IoT), with support for edge computing, aimed at applications involving monitoring, control, and data analysis in near real time. The motivation arises from the growing demand for technological solutions that combine low cost, autonomy, flexibility, and scalability—especially in scenarios with limited, intermittent, or nonexistent network infrastructure. The exclusive dependence on cloud-based services, typical of many conventional IoT architectures, imposes critical challenges related to latency, security, reliability, and operational cost.

The proposed solution seeks to overcome these limitations through the construction of a local ecosystem capable of operating autonomously. The core of the architecture consists of a network of microcontrollers (e.g., Raspberry Pi Pico W), equipped with sensors and actuators programmed in the C language, connected to a local server hosting an MQTT broker (Mosquitto), a relational (PostgreSQL) or non-relational (NoSQL) database, and an analytical dashboard developed in Python with machine learning capabilities. Communication between devices occurs over Wi-Fi using the MQTT protocol, with messages transmitted in JSON format, ensuring lightweight, interoperable, and efficient data exchange. The system operates in a continuous cycle of data collection, local analysis, and automated physical action, supporting decision-making based on rules or predictive models.

The project was implemented through iterative phases, from device configuration to real-time data visualization. Tests conducted in a controlled environment demonstrated the technical feasibility of the solution, highlighting its ability to operate with low latency, high reliability, and adaptability. The results indicate the architecture’s potential for use in residential, industrial, agricultural, and urban environments, contributing to the democratization of intelligent automation technologies with local control and autonomous response—even in low-complexity scenarios and constrained infrastructures.

Keywords: IoT; Embedded Systems; Edge Computing; MQTT; Intelligent Automation.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

1.1 FASE 1 – INICIAÇÃO: TERMO DE ABERTURA DO PROJETO (TAP)

Este trabalho foi estruturado em cinco fases, conforme metodologias consagradas de gerenciamento de projetos, adaptadas ao contexto prático da Residência Tecnológica. A primeira fase, de Iniciação, compreende a definição dos objetivos gerais e específicos, o levantamento dos requisitos iniciais, o escopo preliminar do projeto, os recursos necessários, os principais stakeholders e os critérios de aceitação.

O principal produto desta etapa é o Termo de Abertura do Projeto (TAP), documento que formaliza o início do projeto e assegura o alinhamento com os objetivos educacionais e tecnológicos da Residência. Esse termo delimita os compromissos do projeto, viabiliza sua aprovação e serve como referência para as fases subsequentes.

1.2 MOTIVAÇÃO

A crescente demanda por soluções capazes de realizar o monitoramento e o controle em tempo real tem impulsionado o avanço de tecnologias baseadas em Internet das Coisas (IoT) e sistemas embarcados. Essas tecnologias são amplamente empregadas em ambientes diversos, como residências, comércios, indústrias, contextos de mobilidade urbana e dispositivos vestíveis (*wearables*), viabilizando a automação inteligente de processos e a geração de dados relevantes para tomada de decisão.

Entre as tecnologias que viabilizam soluções inteligentes de automação e monitoramento em tempo quase real, destacam-se os sistemas embarcados e a Internet das Coisas (IoT). Enquanto os sistemas embarcados consistem em dispositivos computacionais dedicados, projetados para executar funções específicas com eficiência, confiabilidade e baixo consumo de energia, a IoT expande seu potencial ao conectar esses dispositivos a redes de comunicação. Essa integração permite a coleta, transmissão e análise de dados de forma distribuída e automatizada. Em aplicações modernas, é justamente a combinação entre a capacidade de processamento local dos sistemas embarcados e a conectividade da IoT que

possibilita o desenvolvimento de arquiteturas robustas, acessíveis e adaptáveis às mais diversas demandas.

No entanto, muitos sistemas atuais ainda dependem fortemente da conectividade com a nuvem, o que impõe desafios significativos relacionados à latência, segurança, disponibilidade e custo. Este projeto surge com a proposta de superar essas limitações por meio da implementação de uma arquitetura modular com ciclo de feedback inteligente, capaz de operar com controle local e tomada de decisão autônoma, mesmo sem conexão constante com servidores externos.

1.3 CONTEXTUALIZAÇÃO

A evolução da IoT e dos sistemas embarcados tem democratizado o acesso a soluções de automação e monitoramento que antes eram restritas a grandes infraestruturas. Atualmente, é possível desenvolver sistemas complexos com baixo custo e alta escalabilidade, permitindo a coleta de dados de sensores, o controle de atuadores e a análise de informações em tempo real, mesmo em locais com conectividade limitada. Essa capacidade é particularmente relevante em cenários onde a latência é crítica e a dependência da nuvem pode ser um gargalo. A adoção da computação em borda (edge computing) surge como uma solução estratégica, permitindo que o processamento e a análise de dados ocorram mais próximos da fonte, reduzindo o tráfego de rede e aumentando a resiliência do sistema. Neste contexto, este projeto visa explorar a sinergia entre IoT, sistemas embarcados e edge computing para desenvolver uma arquitetura que ofereça flexibilidade e autonomia em diversos domínios de aplicação.

Tecnologias como o Raspberry Pi Pico W, brokers MQTT, bancos de dados relacionais e bibliotecas de análise como scikit-learn permitem que soluções complexas sejam implementadas com recursos limitados, mantendo a robustez e a escalabilidade.

Neste contexto, este projeto propõe o desenvolvimento de uma arquitetura IoT distribuída que realiza todo o ciclo de: coleta de dados → armazenamento local → análise (com possibilidade de machine learning) → ação física automatizada, conferindo inteligência embarcada e reatividade ao sistema, com forte aderência às demandas do mercado atual.

1.4 PROBLEMA DE PESQUISA

Em muitos cenários de aplicação, especialmente em ambientes com infraestrutura de rede limitada, intermitente ou inexistente, a dependência exclusiva de soluções baseadas em nuvem para monitoramento e controle em tempo real apresenta desafios significativos. A alta latência na transmissão de dados, a vulnerabilidade a falhas de conectividade e os custos operacionais associados ao tráfego de dados para a nuvem podem comprometer a eficácia e a viabilidade dessas soluções. Além disso, a segurança e a privacidade dos dados podem ser comprometidas quando todas as informações são processadas e armazenadas remotamente.

Diante disso, o problema de pesquisa que este trabalho busca abordar é: Como projetar e implementar uma arquitetura modular e distribuída, baseada na integração entre sistemas embarcados e IoT, com suporte à computação em borda, capaz de prover monitoramento e controle em tempo quase real, armazenamento persistente local e análise de dados, em ambientes com restrições de infraestrutura, priorizando baixo custo, escalabilidade, flexibilidade operacional e redução de latência?

1.5 JUSTIFICATIVA

A crescente digitalização dos processos em diferentes setores da sociedade tem ampliado a necessidade por soluções de monitoramento e controle que sejam eficientes, responsivas e acessíveis. Ambientes residenciais, comerciais, industriais, de mobilidade urbana e até mesmo vestíveis demandam sistemas capazes de operar de forma contínua, com baixo custo, flexibilidade e autonomia. A proposta se justifica por sua relevância prática e educacional. Do ponto de vista tecnológico, ela responde a um desafio real enfrentado em múltiplos setores: como garantir eficiência, confiabilidade e autonomia em sistemas de automação sem depender de serviços externos? Ao propor uma arquitetura que integra sensores, atuadores, conectividade local e inteligência embarcada com suporte a machine learning, o projeto demonstra uma abordagem viável, replicável e de alto impacto.

Nesse cenário, destaca-se a importância de desenvolver arquiteturas que permitam a coleta, o processamento e a análise de dados em tempo quase real,

mesmo em locais com infraestrutura limitada ou com baixa disponibilidade de conexão à internet. A capacidade de operar de forma local e autônoma não apenas reduz a dependência de serviços externos, como também contribui para maior segurança, menor latência e melhor adaptabilidade às necessidades específicas de cada aplicação.

Assim, este projeto se justifica pela necessidade de propor uma solução prática e escalável que atenda a tais demandas, contribuindo para democratizar o acesso a tecnologias de automação e monitoramento em contextos diversos e dinâmicos. Além disso, ao integrar sensores, atuadores e mecanismos de análise de dados em um mesmo ecossistema, esta arquitetura contribui para o fortalecimento do conceito de IoT distribuída, promovendo maior autonomia operacional, capacidade de adaptação e inteligência embarcada. A capacidade de operar de forma autônoma em borda, mesmo sem conectividade com a nuvem, confere robustez e confiabilidade, tornando-o adequado para aplicações críticas. Também podemos usar banco de dados não relacionais ajudando nas questões de Big Data caso seja necessária essa abordagem.

1.6 OBJETIVOS

1.6.1 Objetivo Geral

Projetar e implementar uma arquitetura modular e distribuída, baseada na integração entre sistemas embarcados e aplicações de Internet das Coisas (IoT), com suporte à computação em borda (*edge computing*), voltada à aquisição, transmissão por protocolos de comunicação leve, armazenamento persistente local e análise de dados em tempo quase real. A proposta visa promover a automação de processos e a tomada de decisões inteligente em ambientes com restrições de infraestrutura, priorizando o baixo custo, a escalabilidade, a flexibilidade operacional e a redução de latência. Adicionalmente, a análise dos dados poderá ser enriquecida por modelos de aprendizado de máquina (*machine learning*), ampliando a capacidade preditiva e adaptativa da solução.

1.6.2 Objetivos Específicos

A motivação inicial é usar a arquitetura proposta a seguir. Utilizando um Raspberry Pi 5 como hub central. Ele funcionará como um ponto de acesso Wi-Fi e abrigará um broker Mosquitto MQTT. Microcontroladores com seus sensores e atuadores se conectarão a essa rede e se comunicarão via MQTT. Um servidor local também se conectará para coletar e armazenar os dados em um banco de dados. Para análise, um dashboard se conectará a esse banco de dados. Podemos desenvolver esse dashboard em Python, aproveitando bibliotecas como o scikit-learn para análises de dados mais robustas e tomadas de decisão estratégicas. Alternativamente, o Node-RED pode ser empregado para simplificar esse processo.

Devido às limitações orçamentárias, este trabalho prático adotará uma abordagem mais simplificada. Utilizaremos um computador básico com recursos suficientes para hospedar o broker MQTT, o banco de dados, o dashboard rodando técnicas em ML em tempo real. A rede de microcontroladores, composta por placas como o Raspberry Pi Pico W e outros microcontroladores configurados, irá se conectar a essa rede Wi-Fi e enviar os dados diretamente para o computador, onde serão armazenados no banco de dados. A rede WiFi será fornecida por um roteador simples.

Para alcançar o objetivo geral, este projeto se propõe a projetar e implementar uma rede de microcontroladores com sensores e atuadores, programados em linguagem C, para coletar dados e executar comandos. Isso inclui a seleção de componentes, a programação para conectividade Wi-Fi e MQTT, o tratamento dos dados dos sensores e o controle dos atuadores. Configurar um servidor local (backend) para hospedar um broker MQTT (ex: Mosquitto) para a comunicação entre os dispositivos, e um banco de dados (ex: MongoDB ou PostgreSQL) para o armazenamento persistente dos dados coletados. Desenvolver um dashboard (frontend) utilizando ferramentas como Node-RED ou bibliotecas Python (ex: Dash, Streamlit) para a visualização, análise e apresentação dos dados em tempo real, permitindo o apoio à decisão.

Investigar a aplicação de algoritmos de aprendizado de máquina nos dados coletados para enriquecer as análises e possibilitar a implementação de funcionalidades preditivas ou adaptativas no sistema.

Realizar testes de desempenho e validação da arquitetura proposta em um ambiente controlado, avaliando a latência, a confiabilidade da comunicação e a capacidade de processamento em borda.

1.7 FASE 2 – PLANEJAMENTO: ESTRUTURA ANALÍTICA DO PROJETO (EAP)

A fase de planejamento consiste na decomposição do trabalho em entregas modulares e organizadas, através da Estrutura Analítica do Projeto (EAP). Essa estrutura orienta na criação do cronograma, a definição dos marcos de entrega, recursos, responsabilidades e ferramentas utilizadas. As principais atividades foram agrupadas por etapas técnicas: comunicação MQTT, integração com servidor, armazenamento de dados e desenvolvimento do dashboard. Os custos, riscos e o escopo do projeto

1.7.1 Escopo do projeto

As principais entregas técnicas foram estruturadas da seguinte forma:

Módulo 1 – Dispositivos de Campo: desenvolvimento do firmware em linguagem C para microcontroladores (ex: Raspberry Pi Pico W), incluindo leitura de sensores e controle de atuadores;

Módulo 2 – Comunicação e Persistência: configuração do WiFi e do broker MQTT (Mosquitto), desenvolvimento do serviço de ingestão e estruturação do banco de dados PostgreSQL;

Módulo 3 – Análise e Visualização: criação de dashboard com gráficos e KPIs em tempo real, utilizando bibliotecas como Dash;

Módulo 4 – Controle Inteligente: implementação de lógica de controle baseada em regras e machine learning (ex: scikit-learn), com retorno automático de comandos ao ambiente físico.

Essa estrutura permitiu organizar o desenvolvimento em fases iterativas, com entregas parciais testáveis e mensuráveis.

1.8 METODOLOGIA

A metodologia proposta para o desenvolvimento deste projeto será dividida em fases sequenciais, abrangendo desde o levantamento de requisitos até a validação do sistema, utilizando uma abordagem de pesquisa aplicada e desenvolvimento experimental.

1.8.1 Levantamento de Requisitos e Definição da Arquitetura

Nesta fase inicial, serão definidos os requisitos funcionais e não-funcionais do sistema, considerando as restrições de infraestrutura e o objetivo de baixo custo.

Revisão Bibliográfica: Pesquisa aprofundada sobre arquiteturas de IoT distribuídas, computação em borda, protocolos de comunicação leves (MQTT, CoAP), sistemas embarcados (Raspberry Pi Pico W, ESP32), bancos de dados embarcados/leves, e frameworks de visualização de dados (Node-RED, Dash, Streamlit).

Definição da Arquitetura de Hardware: Seleção dos microcontroladores (Raspberry Pi Pico W, ESP32, etc.) e dos sensores e atuadores específicos que serão utilizados para demonstrar a viabilidade da solução em um cenário prático (ex: monitoramento de temperatura, umidade, presença, controle de iluminação).

Definição da Arquitetura de Software: Detalhamento dos componentes de software, incluindo: Firmware dos Microcontroladores: Linguagem de programação (C/C++), bibliotecas para Wi-Fi e MQTT, tratamento de dados de sensores e lógica de controle de atuadores.

Backend (Servidor Local): Sistema operacional (Linux embarcado como Raspberry Pi OS, ou distribuição leve de Linux). Para simplificar foi usado o Windows 11 com um roteador Wi-Fi externo. Instalação e configuração do broker MQTT (Mosquitto), e escolha/instalação do banco de dados (ex: SQLite para simplicidade inicial, PostgreSQL para mais recursos).

Frontend (Dashboard): Definição da plataforma de desenvolvimento (Node-RED para agilidade, ou Python com Dash/Streamlit para maior flexibilidade e integração com ML).

Definição dos Protocolos de Comunicação: Detalhamento do uso do MQTT para a comunicação entre os microcontroladores e o servidor local, e entre o servidor e o dashboard.

1.8.2 Implementação e Desenvolvimento

Esta fase envolve a implementação prática dos componentes de hardware e software definidos na fase anterior.

1.8.2.1 Desenvolvimento do Firmware dos Microcontroladores

- Programação dos microcontroladores em C usando o SDK do Raspberry Pico W na usando o VSCode.
- Implementação da conectividade Wi-Fi na rede criada pelo servidor local.
- Implementação do cliente MQTT para publicação de dados dos sensores e inscrição em tópicos para receber comandos dos atuadores.
- Leitura e pré-processamento dos dados dos sensores.
- Controle dos atuadores com base nos comandos recebidos.

1.8.2.2 Configuração e Implementação do Servidor Local (Backend)

- Instalação e configuração do sistema operacional no computador/placa embarcada (ex: Raspberry Pi).
- Configuração do Wi-Fi no roteador externo.
- Instalação e configuração do broker MQTT (Mosquitto).
- Implementação do componente de persistência de dados: desenvolvimento de scripts ou aplicações que recebam os dados do broker MQTT e os armazenem no banco de dados local.

1.8.2.3 Desenvolvimento do Dashboard (Frontend)

Com Node-RED: Criação de fluxos para se conectar ao broker MQTT, consumir os dados do banco de dados, processá-los e exibir visualizações interativas.

Com Python (Dash): Desenvolvimento de scripts Python para criar o dashboard, conectando-se ao banco de dados para a recuperação e visualização dos dados. E foi usado ele porque, se mostrou mais versátil, se integra melhor as bibliotecas usadas em aprendizado de máquina e com o banco de dados.

1.8.2.4 Integração dos Componentes

Realização dos testes de integração entre os microcontroladores, o broker MQTT, o banco de dados e o dashboard para garantir a comunicação e o fluxo de dados adequado.

1.8.3 Cronograma de Desenvolvimento e Testes

O projeto será implementado em etapas modulares:

1) Comunicação entre microcontroladores e Pico W via MQTT.
2) Integração do Pico W com o servidor para envio de dados.
3) Armazenamento e consulta no banco de dados.
4) Visualização por meio de dashboard.

Serão realizados testes em ambiente controlado para validar o tempo de resposta, confiabilidade da comunicação e integridade dos dados armazenados.

Período	Etapa	Atividades principais
15/06 – 21/06	Etapa 1: Comunicação via MQTT	Estudo do protocolo MQTT/WiFi Configuração dos microcontroladores Envio de mensagens de teste ao Pico W
22/06 – 28/06	Etapa 2: Integração Pico W – Servidor	Teste de envio de dados via Wi-Fi/MQTT Validação da comunicação bidirecional
29/06 – 05/07	Etapa 3: Armazenamento em banco de dados	Escolha e configuração do banco (ex.: PostgreSQL) Criação da estrutura de tabelas Implementação de inserção e leitura de dados
06/07 – 15/07	Etapa 4: Dashboard analítico em Python	Desenvolvimento de interface básica Conexão com o banco de dados Exibição dos dados recebidos em tempo real

Tabela 5-Cronograma

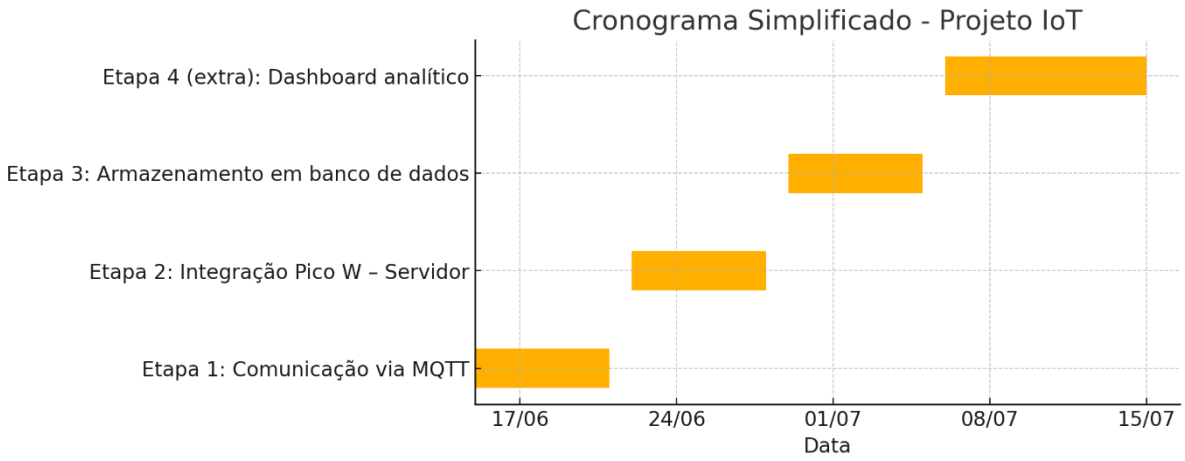

Figura 2- Cronograma

2 DESENVOLVIMENTO

2.1 FASE 3 E 4 – EXECUÇÃO E MONITORAMENTE: ENTREGA DO PROJETO E ACOMPANHAMENTO

Nesta fase 3 a ser descrita, o projeto foi implementado conforme o planejamento, com foco na montagem do protótipo funcional.

Todo código comentado do projeto está no link abaixo.

Link: https://github.com/ManoelFelipe/Embarcatech_37_IoT

Link: Projeto Completo: Embarcatech_37_Projeto_Fase_01_IoT.zip

Vídeo da apresentação do projeto:

Teórico: https://youtu.be/XMjRIqREULI

Prático: https://youtu.be/Qb1Wmr4_hvI

A fase 3 de execução consistiu na implementação prática da arquitetura proposta, com a construção do protótipo funcional e a integração dos módulos previamente planejados: dispositivos de campo, backend (servidor e persistência), análise de dados e controle automatizado. Todas as etapas foram conduzidas de forma iterativa, permitindo o teste, a validação e o ajuste incremental dos componentes do sistema.

A fase 04, foi feita em conjunto, monitorando, acompanhando o projeto. Verificando os status das atividades e os prazos.

2.1.1 Dispositivos IoT e Firmware (Módulo 1)

A base do sistema embarcado foi composta por microcontroladores Raspberry Pi Pico W, programados em linguagem C, utilizando o Pico SDK em ambiente de desenvolvimento Visual Studio Code. Cada nó IoT foi equipado com sensores (ex: temperatura (AHT10), umidade (AHT10), iluminação (BH1750)) e atuadores (ex: ventiladores, LEDs), implementando um ciclo local de coleta e resposta.

As principais funcionalidades embarcadas incluíram:

- Estabelecimento de conexão Wi-Fi com a rede local fornecida pelo servidor;
- Publicação dos dados dos sensores em tópicos MQTT específicos;
- Subscrição a tópicos para receber comandos de controle;
- Ações físicas imediatas com base nas mensagens recebidas, como o acionamento de um ventilador mediante leitura de temperatura crítica.

Esse módulo caracteriza-se pela baixa latência de resposta, baixa demanda energética e alta replicabilidade, sendo facilmente escalável em topologias horizontais.

Um ou mais microcontroladores (Raspberry Pico W) com sensores (e.g., temperatura e umidade com AHT10, iluminação com BH1750) e um atuador (e.g., um LED ou um relé que aciona uma lâmpada).

A parte dos atuadores não foi implementada, para trabalhos futuros.

Em resumo, o projeto é bem-organizado:

- main.c é o chefe.
- app_tasks.c é o gerente que executa as ordens.
- Os drivers (bh1750.c, aht10.c), o módulo mqtt_lwip.c e lwipopts.h são as bibliotecas especializadas.
- Os arquivos .h são os contratos que definem como eles se comunicam.
- configura_geral.h é onde você ajusta as senhas e configurações sem precisar mexer na lógica do programa.

A tabela 02 mostra um resumo para entender guiar no entendimento do trabalho.

Arquivo	Tipo de Arquivo	Propósito Principal (O que ele faz?)	Funções / Definições Chave	Relação com Outros Arquivos
main.c	Principal (Programa)	É o cérebro do projeto. Ele orquestra a inicialização e executa o loop infinito que mantém o dispositivo funcionando.	main()	Chama funções de app_tasks.h para executar as tarefas.
app_tasks.h	Cabeçalho (Interface)	É o "menu de serviços" da aplicação. Lista todas as tarefas que o main.c pode solicitar, como "conectar ao Wi-Fi" ou "ler sensores".	conectar_wifi(), processar_ciclo_operacional(), INTERVALO_LOOP_MS	É incluído pelo main.c (para usar os serviços) e app_tasks.c (para implementar os serviços).
app_tasks.c	Implementação (Lógica)	É a "cozinha" do projeto. Contém o passo a passo (a lógica) de como cada tarefa do "menu" (app_tasks.h) é executada.	Implementa as funções de app_tasks.h.	Inclui e usa os drivers dos sensores (bh1750.h, aht10.h) e o módulo MQTT (mqtt_lwip.h).
configura_geral.h	Configuração	É o "painel de controle". Centraliza todas as informações que podem mudar, como nome e senha do Wi-Fi, IP do servidor e pinos usados.	WIFI_SSID, WIFI_PASSWORD, MQTT_BROKER_IP, DEVICE_ID	É incluído por app_tasks.c para obter as credenciais e configurações.

bh1750.h e bh1750.c	Driver de Sensor	É o "manual de instruções" para o sensor de luz (BH1750). Sabe exatamente como se comunicar com este sensor para obter o valor de luminosidade.	bh1750_iniciar(), bh1750_ler_lux()	É usado por app_tasks.c para ler os dados de luz.
aht10.h e aht10.c	Driver de Sensor	É o "manual de instruções" para o sensor de temperatura e umidade (AHT10). Sabe como pedir e traduzir os dados deste sensor.	aht10_init(), aht10_read_data()	É usado por app_tasks.c para ler os dados de temperatura e umidade.
mqtt_lwip.h e mqtt_lwip.c	Módulo de Comunicação	É o "carteiro" do projeto. Sua única função é pegar uma mensagem e enviá-la pela internet para o servidor (Broker MQTT).	iniciar_mqtt_cliente(), publicar_mensagem_mqtt()	É usado por app_tasks.c para publicar os dados dos sensores.
lwipopts.h	Configuração de Rede	É a "configuração do motor da internet" (lwIP). É um arquivo técnico que ajusta o funcionamento da comunicação de rede em baixo nível.	NO_SYS, LWIP_TCP, MEM_SIZE	É usado internamente pela biblioteca de rede do Pico, não diretamente pelo nosso código.

Tabela 6 - Resumo do Código do Módulo 01

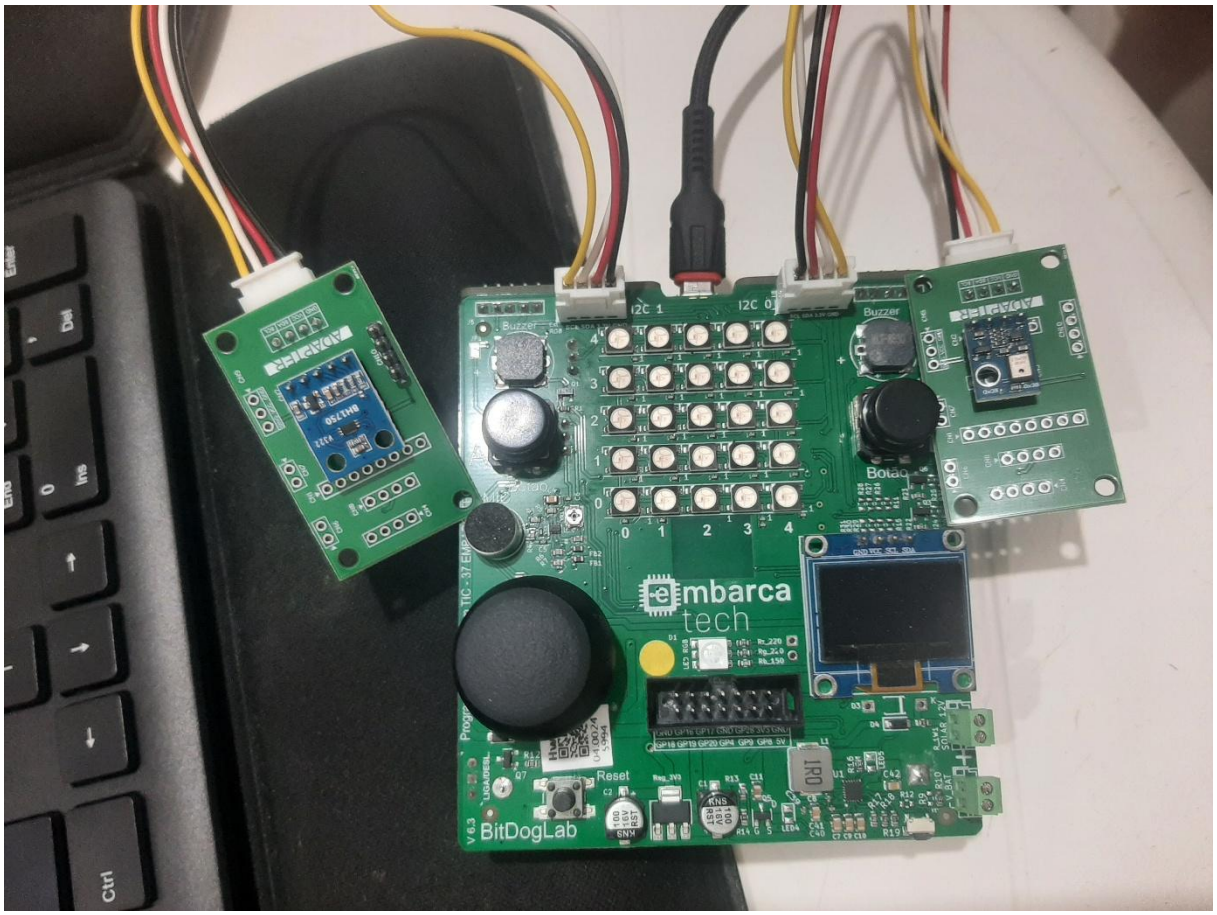

Figura 2 - Placa BitDogLab - Sensores AHT10/BH1750

2.1.2 Comunicação, Broker MQTT e Banco de Dados (Módulo 2)

A comunicação entre os dispositivos IoT e o backend foi realizada por meio do protocolo MQTT, utilizando o broker Mosquitto instalado em um computador pessoal que atuou como servidor local e ponto de acesso Wi-Fi. A leveza do protocolo foi essencial para garantir a estabilidade da troca de mensagens em tempo quase real, mesmo com limitações de hardware.

Um script Python denominado data_ingestor.py, desenvolvido com a biblioteca paho-mqtt e psycopg2, foi responsável por escutar os tópicos publicados pelos microcontroladores e realizar a persistência dos dados recebidos.

Para armazenamento, foi adotado um banco de dados relacional PostgreSQL, embora a arquitetura também seja compatível com soluções NoSQL, como MongoDB, dependendo da necessidade de flexibilidade ou performance.

A tabela 03 mostra um resumo para entender guiar no entendimento do trabalho.

Característica	data_ingestor.py (O Recebe e Arquiva)
Propósito Principal	Esperar pelos dados dos sensores que chegam pela "internet" (MQTT) e guardá-los de forma organizada no banco de dados.
O que ele faz?	Escuta o servidor de mensagens (Broker MQTT) sem parar. Recebe os dados de temperatura, umidade e luz em formato JSON. Conecta-se ao banco de dados PostgreSQL. Insere cada leitura recebida em uma nova linha na tabela do banco.
Tecnologias Usadas	Paho-MQTT Psycopg2 (PostgreSQL) JSON
Como ele se encaixa?	É um serviço que roda em segundo plano. Atua como a ponte entre os sensores (que enviam dados) e o banco de dados (que armazena).

Tabela 7 - Resumo do Código do Módulo 02

2.1.3 Dashboard Analítico e Visualização (Módulo 3)

A camada de visualização foi construída com foco na análise em tempo real dos dados coletados. Duas abordagens foram exploradas:

- Node-RED: utilizado para rápida prototipação, possibilitando a criação de fluxos de dados, painéis visuais e alertas sem necessidade de programação extensiva;
- Dash (Python): empregado para a construção de um dashboard mais robusto, com gráficos interativos, tabelas, KPIs e integração com bibliotecas de machine learning, como o scikit-learn.

Os dados trafegaram em formato JSON, facilitando a ingestão tanto pelo banco de dados quanto pela camada de frontend. A atualização em tempo real dos painéis permitiu o acompanhamento direto do comportamento dos sensores e das decisões automatizadas do sistema.

A tabela 04 mostra um resumo para entender guiar no entendimento do trabalho.

Característica	dashboard.py (O Analista e Apresentador)
Propósito Principal	Ler os dados que foram guardados, analisá-los, criar gráficos e apresentá-los de forma visual e interativa em uma página web.
O que ele faz?	Conecta-se ao mesmo banco de dados PostgreSQL. Busca os dados mais recentes da tabela. Cria gráficos de linha para cada sensor. Usa Machine Learning para detectar dados estranhos ("anomalias") e prever tendências. Exibe tudo em uma página web que se atualiza sozinha.
Tecnologias Usadas	Dash e Plotly (para a página web e gráficos) Pandas e SQLAlchemy (para manipular os dados) Scikit-learn (para Machine Learning)
Como ele se encaixa?	É a interface com o usuário. Atua como a janela para os dados que o data_ingestor.py armazenou. O usuário interage com este script através do navegador.

Tabela 8 - Resumo do Código do Módulo 03

2.1.4 Controle Inteligente e Feedback Automatizado (Módulo 4)

O módulo de controle foi projetado para simular cenários reais de automação, como acionamento de ventilação ou iluminação com base em condições ambientais.

O sistema pode contar com:

- Regras locais embarcadas nos microcontroladores (ex: temperatura > 30°C → ativar ventilador);
- Decisão baseada em ML: modelos simples de classificação ou regressão foram testados utilizando o scikit-learn, treinados com dados históricos e integrados ao backend para acionar os atuadores de forma preditiva.

Essa abordagem confere ao sistema um ciclo de feedback inteligente, permitindo:

1. Coleta do dado → 2
2. Armazenamento → 3

3. Análise (estatística ou preditiva) → 4
4. Ação automatizada → 5
5. Novo dado → 1 repetição do ciclo.

Contudo não foi implementado até a data de entrega do trabalho. Fica como uma implementação futura. Talvez integrando a outros trabalhos. Esse trabalho tem como objetivo auxiliar outros trabalhos.

2.1.5 Integração Final e Testes

A integração dos módulos permitiu a validação do sistema em um ambiente controlado. Foram realizados testes de:

- Latência entre a publicação do dado e a resposta do atuador;
- Estabilidade da conexão MQTT;
- Persistência correta no banco de dados;
- Atualização em tempo real dos dashboards;
- Desempenho dos modelos de ML.

Esses resultados não foram concluídos de forma satisfatória para apresentar de forma consistente até a data de entrega do projeto.

Os resultados obtidos indicaram que a arquitetura é tecnicamente viável, mesmo com hardware de baixo custo. A modularidade do sistema permite ampliações futuras, como maior número de nós IoT, integração com outros protocolos (ex: LoRaWAN), ou substituição de componentes conforme necessidade.

2.2 FLUXOGRAMA DOS DADOS

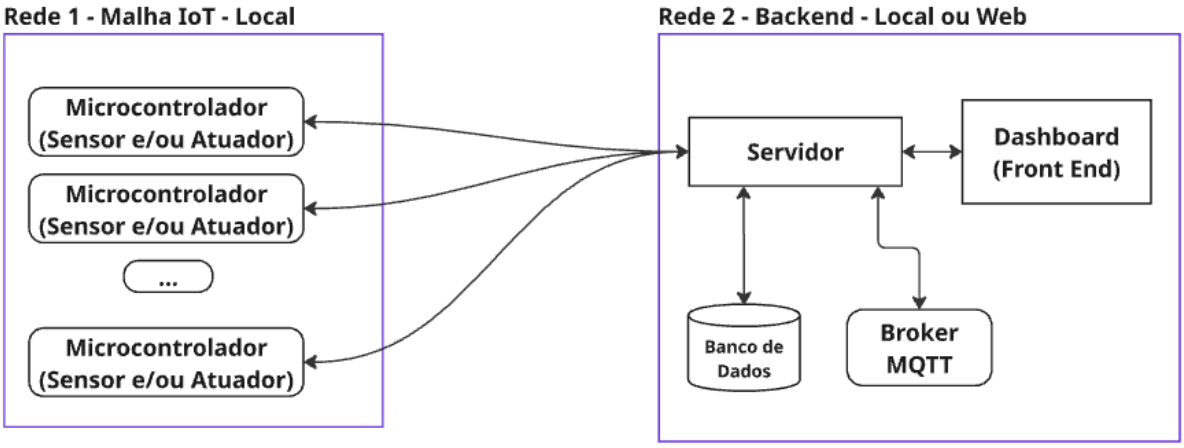

Figura 3 - Fluxo dos dados

3 CONCLUSÃO

3.1 FASE 5 – ENCERRAMENTO: TERMO DE ENCERRAMENTO DO PROJETO (TEP)

Ao final do ciclo de desenvolvimento, foi elaborado o Termo de Encerramento do Projeto (TEP), consolidando os resultados alcançados, as entregas realizadas, as contribuições técnicas e científicas, bem como os desafios enfrentados. A documentação do projeto foi organizada e submetida à banca avaliadora, incluindo os códigos-fonte, a arquitetura proposta, as evidências experimentais e os materiais de apoio como o vídeo demonstrativo e os dashboards desenvolvidos. Essa etapa também proporcionou uma reflexão crítica sobre o processo de execução, permitindo identificar oportunidades de melhoria e caminhos para trabalhos futuros.

3.2 RESULTADOS ESPERADOS

O projeto atendeu aos objetivos propostos ao demonstrar a viabilidade de uma arquitetura IoT modular, distribuída e de baixo custo, capaz de operar em ambientes com infraestrutura limitada. Os testes realizados comprovaram que o sistema é capaz de:

- Coletar dados de sensores de forma confiável;
- Transmitir os dados via protocolo MQTT com baixa latência;
- Armazenar as informações em banco de dados local (PostgreSQL);
- Visualizar os dados em dashboards interativos em tempo real;
- Integrar modelos de machine learning como suporte analítico (análise preditiva e detecção de anomalias). Para executar decisões automatizadas nos microcontroladores (Essas execuções não foi feito, ficou para trabalho futuro)

Esses resultados confirmam que a arquitetura é replicável, funcional e adaptável a diferentes cenários, como residências inteligentes, aplicações industriais e ambientes urbanos ou agrícolas.

3.3 CONTRIBUIÇÕES CIENTÍFICAS E TÉCNICAS

As principais contribuições deste trabalho estão divididas entre o campo prático e acadêmico:

- Contribuição técnica: desenvolvimento de um protótipo funcional com componentes de mercado acessíveis, integrando sensores, atuadores, protocolo MQTT, banco de dados local e análise por machine learning;
- Contribuição científica: proposição e validação de uma arquitetura edge-IoT descentralizada, com potencial de aplicação em áreas com baixa conectividade;
- Contribuição educacional: aplicação prática de conhecimentos interdisciplinares em programação embarcada (C), redes, banco de dados, inteligência artificial e visualização de dados;
- Código aberto: disponibilização pública do projeto em repositório GitHub, permitindo que outros pesquisadores e desenvolvedores utilizem, expandam ou repliquem a solução.

3.4 RESULTADOS VISUAIS

Os resultados visuais obtidos a partir do dashboard analítico incluem:

- Gráficos de séries temporais para cada sensor (temperatura, umidade e luminosidade);

- Indicadores em tempo real (KPIs) com atualização automática;
- Alertas visuais para valores fora do padrão;
- Detecção de anomalias com base em modelos simples de machine learning (quando ativado);
- Representação gráfica do ciclo de coleta, análise e ação.

Esses elementos tornaram a interface do sistema intuitiva, acessível e alinhada com os princípios de engenharia de dados para IoT.

3.5 ANÁLISE PRELIMINAR DOS DADOS (SE POSSÍVEL)

Durante os testes em ambiente controlado, os dados coletados permitiram observar padrões de variação ambiental ao longo do tempo. A análise estatística preliminar indicou:

- Oscilações cíclicas na luminosidade e na temperatura durante o dia;
- Variações moderadas na umidade do ambiente;
- Ausência de falhas críticas na coleta e transmissão de dados;
- Detecção de valores extremos simulados, tratados como anomalias pelo modelo preditivo.

Essas análises demonstram o potencial da arquitetura para funcionar como base para sistemas de previsão, alertas e automação inteligente em tempo real.

3.6 DESAFIOS E COMO ABORDÁ-LOS

Ao longo da execução do projeto, alguns desafios foram enfrentados:

- Limitações orçamentárias: o uso de um computador pessoal no lugar de um servidor dedicado foi uma adaptação necessária, mas eficaz;
- Integração entre componentes heterogêneos: a comunicação entre dispositivos embarcados e sistemas Python exigiu testes e ajustes constantes, especialmente no tratamento dos pacotes MQTT e formato JSON;
- Desempenho com ML embarcado: a aplicação de modelos de machine learning foi restrita a testes básicos por limitação de hardware e tempo. Contudo, a arquitetura permite expansão futura com modelos mais robustos e sensores adicionais;

- Tempo de desenvolvimento: a organização modular em fases (TAP, EAP, execução, testes e TEP) ajudou a manter o projeto dentro dos prazos.

Esses desafios foram mitigados com adaptações técnicas, testes iterativos e uso de ferramentas de fácil integração, como Dash, Mosquitto e PostgreSQL.

4 REFERÊNCIAS BIBLIOGRÁFICAS

ABADI, D. J. et al. Column-stores vs. row-stores: how different are they really? Proceedings of the ACM SIGMOD International Conference on Management of Data, 2008.

AL-FUQAHA, A. et al. Internet of Things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials, v. 17, n. 4, p. 2347–2376, 2015.

BANZI, M.; SHILOH, M. Arduino: a hands-on introduction. O'Reilly Media, 2014.

BONOMI, F. et al. Fog computing and its role in the Internet of Things. Proceedings of the first edition of the MCC workshop on Mobile cloud computing, 2012.

CIRANI, S. et al. A scalable and self-configuring architecture for service discovery in the Internet of Things. IEEE Internet of Things Journal, 2014.

HONG, I. et al. An IoT-based smart home automation system using Raspberry Pi. International Journal of Smart Home, v. 9, n. 10, p. 161–168, 2015.

KARIMI, K.; ATTIYA, G. Internet of Things (IoT): A Technical Overview. Cisco Systems, 2013.

LEE, I.; LEE, K. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Business Horizons, v. 58, n. 4, p. 431–440, 2015.

MORAES, F. G. et al. Sistemas Embarcados: Fundamentos, projeto e aplicações. 2. ed. São Paulo: Elsevier, 2019.

MUNIR, K. et al. A survey on role of MQTT in IoT. Journal of Network and Computer Applications, v. 174, p. 102886, 2021.

NUNES, L. D. R.; XEXÉO, G. Introdução ao Machine Learning com Python. Rio de Janeiro: Brasport, 2019.

PEREIRA, A. S.; et al. Protocolos de comunicação para a Internet das Coisas: características, aplicações e desafios. Revista de Engenharia e Pesquisa Aplicada, v. 6, n. 1, 2021.

RASPBERRY PI FOUNDATION. Raspberry Pi Pico and Pico W documentation. Disponível em: https://www.raspberrypi.com/documentation/

SCIKIT-LEARN DEVELOPERS. Scikit-learn: Machine Learning in Python. Disponível em: https://scikit-learn.org/stable/

SILVA, R. C. da; SANTOS, M. S. dos. Edge Computing: Conceitos, Arquiteturas e Aplicações. SBC – Sociedade Brasileira de Computação, 2021.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. PostgreSQL Documentation. Disponível em: https://www.postgresql.org/docs/

MOSQUITTO MQTT. Eclipse Foundation. Disponível em: https://mosquitto.org/

ESPRESSIF SYSTEMS. ESP32 Technical Reference Manual. 2023. Disponível em: https://www.espressif.com/en/products/socs/esp32/resources

ANEXO A – RESULTADOS VISUAIS

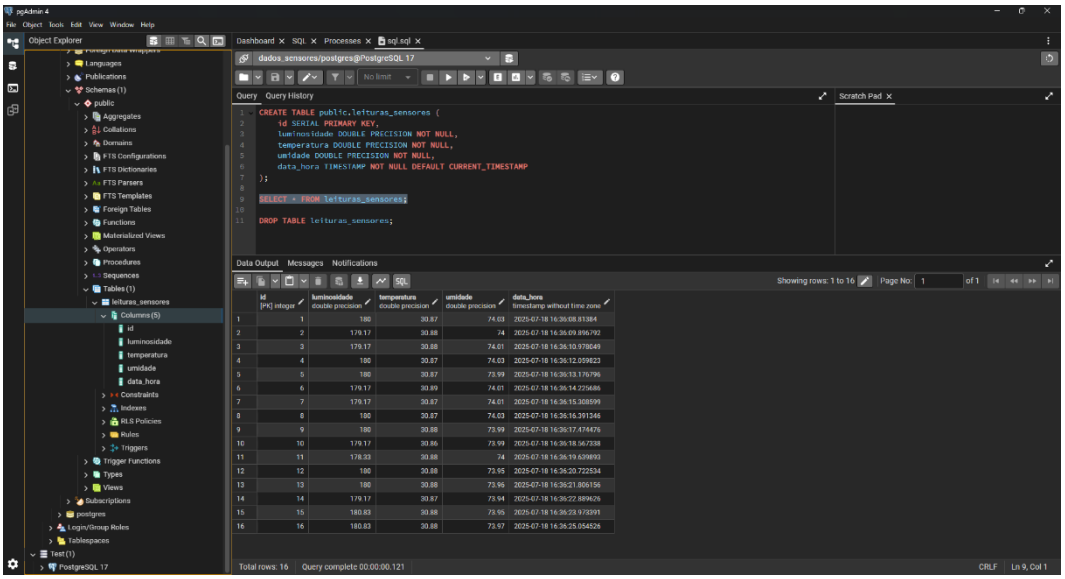

Figura 4 – Ilustração do sistema em funcionamento

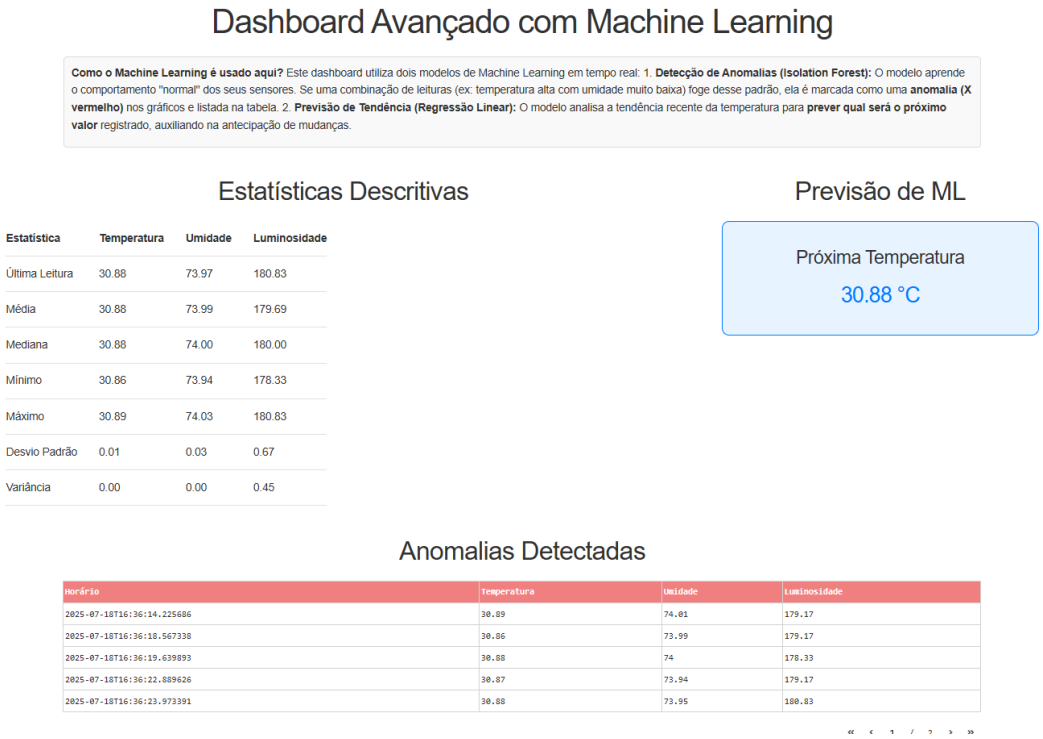

Figura 5 – Ilustração do sistema em funcionamento

Figura 6 – Ilustração do sistema em funcionamento

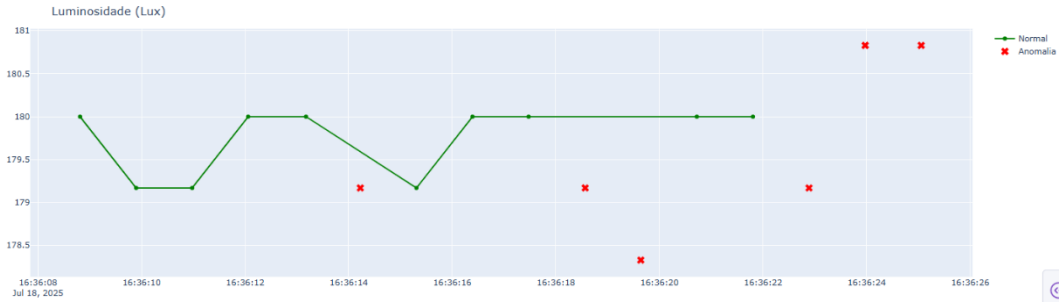

Figura 7 – Ilustração do sistema em funcionamento

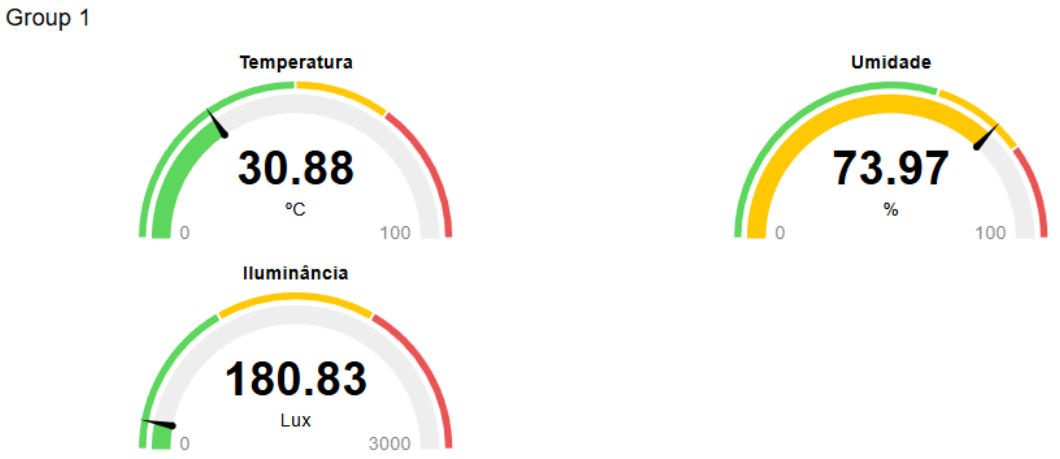

Figura 8 – Ilustração do sistema em funcionamento

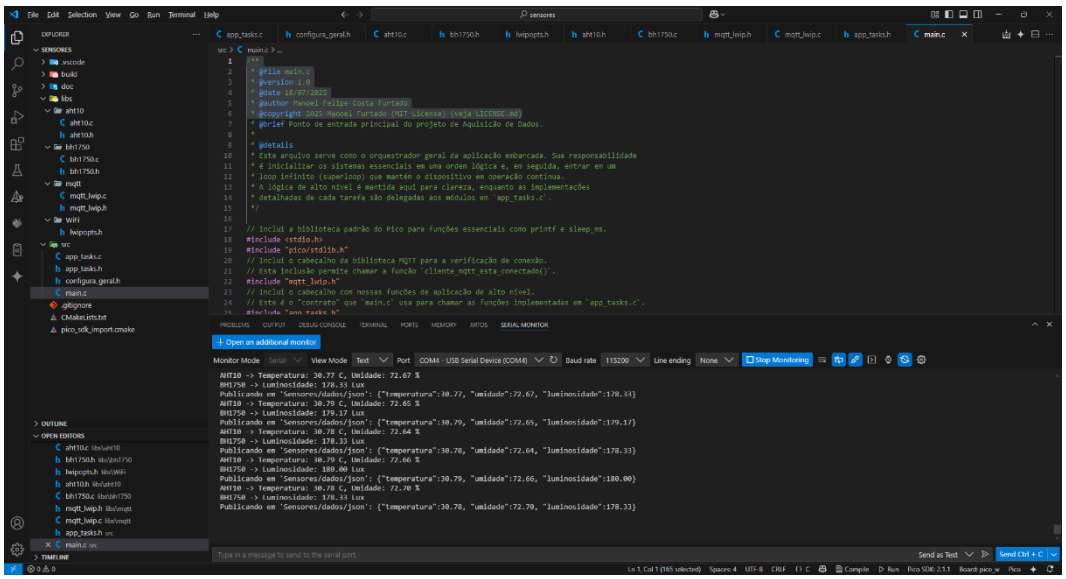

Figura 9 – Ilustração do sistema em funcionamento

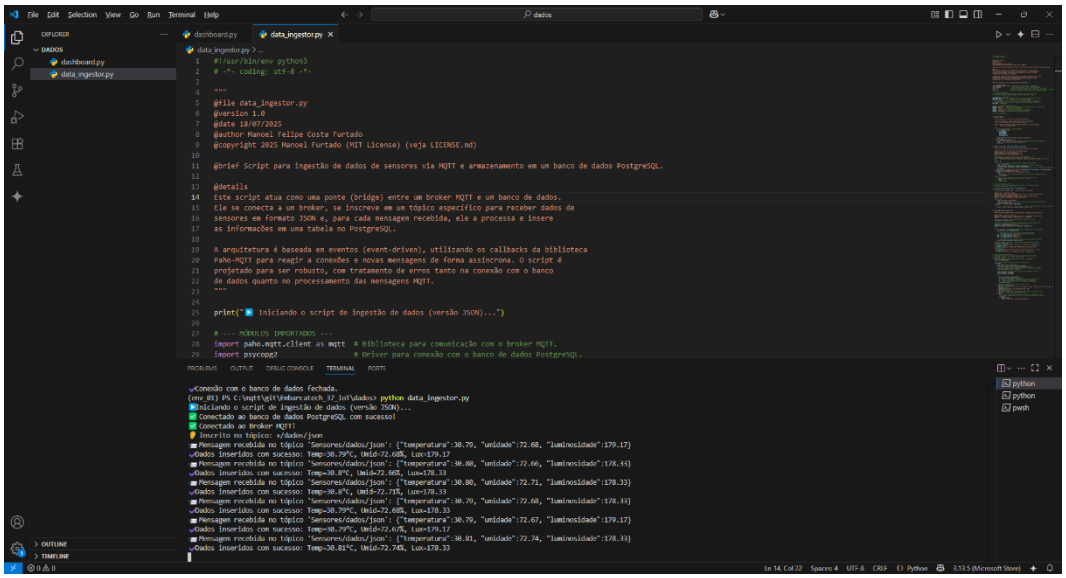

Figura 10 – Ilustração do sistema em funcionamento