Janela Inteligente

Caio Davi Rabelo Fiorini[1], **Caio Gomes Alcântara Glória**[1],
Daniel Salgado Magalhães[1], **Júlio César Gonzaga Ferreira Silva**[1]
Lucas Araujo Barduino Rodrigues[1]

[1]Instituto de Informática – Pontifícia Universidade Católica de Minas Gerais
– Coração Eucarístico
Caixa Postal 30535-901 Belo Horizonte – MG – Brazil

Abstract. *With the growing integration of technology into everyday objects in the context of the Internet of Things (IoT), this article describes the development of a smart window prototype. The project aims to solve the problem of indoor environments being affected by sudden climatic changes, such as rain and excessive humidity. The main objective is to create an automated system that, through temperature and humidity sensors, is capable of autonomously opening or closing a window, aiming for thermal comfort and environmental protection. The methodology involved the use of the ESP32 microcontroller, DHT22 sensors, and servomotor actuators, programmed in Arduino Language, Python, and Flutter.*

Resumo. *Com a crescente integração de tecnologia em objetos cotidianos no contexto da Internet das Coisas (IoT), este artigo descreve o desenvolvimento de um protótipo de janela inteligente. O projeto visa solucionar o problema de ambientes internos serem afetados por mudanças climáticas repentinas, como chuva e umidade excessiva. O objetivo principal é criar um sistema automatizado que, por meio de sensores de temperatura e umidade, seja capaz de abrir ou fechar uma janela de forma autônoma, visando o conforto térmico e a proteção do ambiente. A metodologia envolveu o uso do microcontrolador ESP32, sensores DHT22 e atuadores servomotores, programados em Arduino Language, Python e Flutter.*

1. Introdução

Em momentos que pessoas estão atrasadas, acabam saindo de casa apressadas e, quase sempre, esquecem alguma janela aberta. No Brasil, principalmente nas estações de Primavera e Verão, chuvas repentinas são comuns e podem causar situações de grande desconforto e prejuízos. Outro cenário recorrente ocorre no outono, quando um dia de clima agradável é substituído por um entardecer de ventos frios e queda brusca de temperatura. Neste contexto, os dispositivos conectados representam uma nova filosofia de vida, onde a tecnologia trabalha silenciosamente em segundo plano para criar um ambiente mais harmonioso, eficiente e, fundamentalmente, mais confortável para seus habitantes, conforme apontado por Stojkoska e Trivodaliev (2017) [Stojkoska and Trivodaliev 2017].

Para solucionar os problemas apresentados, este trabalho propõe o desenvolvimento de um protótipo de baixo custo que transforma uma janela convencional em um dispositivo inteligente. O objetivo geral é criar um sistema autônomo que monitore as condições climáticas e opere a janela sem a necessidade de intervenção humana. Para

alcançar este propósito, foram definidos os seguintes objetivos específicos: a) Integrar o microcontrolador ESP32 com um sensor de temperatura e umidade (DHT22) e um servomotor. b) Desenvolver um algoritmo capaz de processar os dados do sensor e tomar a decisão de abrir ou fechar a janela com base em parâmetros pré-definidos. c) Implementar o protocolo de comunicação MQTT para assegurar a transmissão de dados e o controle do dispositivo, mesmo em condições de conectividade intermitente.

Uma das dificuldades previstas no desenvolvimento é a dependência de uma conexão constante com a internet. A utilização do protocolo MQTT visa mitigar essa limitação, pois seu modelo de publicação/assinatura e a capacidade de reter mensagens permitem que o dispositivo guarde os dados gerados offline para enviá-los ao servidor assim que a conexão for restabelecida.

Este artigo está organizado da seguinte forma: a Seção 2 apresenta a revisão da literatura, explorando o estado da arte em automação residencial e soluções similares. A Seção 3 detalha a metodologia, os materiais utilizados e a arquitetura do protótipo. A Seção 4 apresenta os experimentos realizados e discute os resultados obtidos. Por fim, a Seção 5 traz a conclusão do trabalho, analisando os resultados alcançados e propondo melhorias para trabalhos futuros.

2. Revisão Bibliográfica

A seleção do microcontrolador é uma etapa crucial em projetos de Internet das Coisas (IoT). Enquanto plataformas como o Arduino Uno exigem módulos externos para conectividade, o ESP32 se destaca como uma solução integrada e versátil, que já inclui Wi-Fi e Bluetooth nativos. Conforme apresentado por Silva et al. (2018) [Silva et al. 2018], a arquitetura do ESP32, com sua CPU dual-core, oferece um poder de processamento robusto. Essa característica é fundamental para o projeto da Janela Inteligente, pois permite gerenciar simultaneamente as tarefas de monitoramento dos sensores, o controle do motor e a comunicação em tempo real pela rede, garantindo uma operação fluida e responsiva.

A comunicação eficiente entre o dispositivo e o servidor é um pilar fundamental em sistemas IoT, especialmente quando os dispositivos possuem recursos limitados de processamento e energia. Para estes cenários, protocolos de comunicação leves são essenciais. Um estudo comparativo de [Thangam and Rajan 2018] analisa os protocolos MQTT e CoAP, destacando a eficiência do MQTT e seu modelo publish/subscribe para ambientes de rede instáveis. A escolha do MQTT para a Janela Inteligente é, portanto, justificada por sua resiliência. O modelo de Qualidade de Serviço (QoS) do protocolo visa garantir a entrega de mensagens críticas, como um alerta de chuva ou uma mudança brusca de temperatura, mesmo que o dispositivo perca a conexão momentaneamente, alinhando-se perfeitamente com os desafios de conectividade do projeto.

A automação residencial eficaz depende da integração coesa de múltiplos dispositivos em uma arquitetura de sistema bem definida, e a viabilidade de projetos com componentes de baixo custo é um tema recorrente na literatura. O trabalho de [Farhan et al. 2021] apresenta o projeto e a implementação de um sistema completo de casa inteligente de baixo custo, detalhando a arquitetura desde os nós sensores até a interface com o usuário. Este artigo serve como um referencial prático para a Janela Inteligente, demonstrando um modelo de arquitetura de sistema no qual um dispositivo individual, como a janela, se insere. Ele valida a abordagem de utilizar componentes

acessíveis para criar soluções de automação robustas, contextualizando o presente trabalho no cenário mais amplo das casas inteligentes.

Um projeto IoT robusto deve ser compreendido através de uma arquitetura de múltiplas camadas, que abrange desde o hardware físico até a aplicação final que interage com o usuário. Uma revisão abrangente da literatura, conduzida por Al-Garadi et al. (2020) [Al-Garadi et al. 2020], sistematiza as tecnologias e os desafios da IoT em uma arquitetura de camadas bem definidas (física, de rede, de middleware e de aplicação). Essa visão arquitetônica é vital para o projeto da Janela Inteligente, pois permite posicionar cada componente e decisão de projeto dentro da camada apropriada. A escolha do ESP32 e do sensor DHT22 pertence à camada física, a implementação do protocolo MQTT à camada de rede, e a lógica de decisão autônoma à camada de aplicação, estruturando o desenvolvimento e a análise do trabalho de forma clara e organizada.

A base da Internet das Coisas reside nos sistemas embarcados, que são computadores dedicados a funções específicas, operando sob restrições de recursos. Conforme discutido por Giusto et al. (2010) [Giusto et al. 2010], o design de aplicações IoT é diretamente influenciado pelas limitações de processamento, memória e energia que são inerentes à arquitetura de computadores desses dispositivos. Esta perspectiva fundamenta o desenvolvimento da Janela Inteligente, tratando-a não como um simples dispositivo conectado, mas como um sistema embarcado completo. A necessidade de um código otimizado e de protocolos de comunicação leves, como o MQTT, não é apenas uma escolha de design, mas uma consequência direta das restrições de sua arquitetura de hardware, visando garantir a autonomia e a estabilidade da solução.

3. Metodologia

Este trabalho possui um cunho experimental, focado na construção de um protótipo funcional. O dispositivo inteligente desenvolvido será acoplado a uma janela convencional para permitir a operação autônoma do sistema.

Para detalhar a construção do protótipo, esta seção foi dividida em três camadas fundamentais, que representam a arquitetura do sistema IoT: Hardware Embarcado, Protocolos de Comunicação e Firmware, conforme a Figura 1. As subseções a seguir apresentarão cada uma dessas camadas.

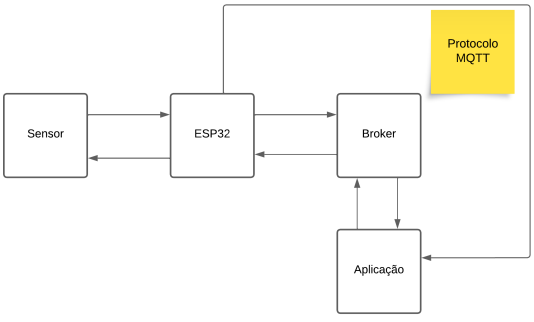

Figure 1. Fluxograma do Projeto

3.1. Hardware Embarcado

A seleção dos componentes de hardware para o protótipo foi guiada por três critérios principais: baixo custo, alta disponibilidade no mercado nacional e integração facilitada. A arquitetura de hardware do dispositivo, cujo diagrama esquemático é apresentado na Figura abaixo, é composta por um microcontrolador, um sensor de condições climáticas e um atuador para o movimento da janela.

O cérebro do protótipo é o microcontrolador ESP32. Esta plataforma foi escolhida por sua excelente relação custo-benefício e por ser um System on a Chip (SoC) que já integra conectividade Wi-Fi e Bluetooth. Sua arquitetura com CPU dual-core, conforme discutido por [Silva et al. 2018], é um diferencial importante, permitindo que o sistema gerencie as tarefas de rede e a lógica de controle de forma paralela e eficiente, sem comprometer o desempenho.

Para a coleta de dados ambientais, foi selecionado o sensor de temperatura e umidade DHT22. Este componente é capaz de medir temperaturas na faixa de -40 a 80°C e umidade de 0 a 100%, com boa precisão para a aplicação proposta. Sua interface de comunicação digital de um fio simplifica a conexão com o microcontrolador, e a vasta disponibilidade de bibliotecas para a plataforma Arduino facilita a implementação do firmware.

Finalmente, o mecanismo de abertura e fechamento da janela é controlado por um Servo-Motor de alto torque (modelo MG996R). A escolha de um servo-motor se justifica por seu controle preciso de posição angular, permitindo que a janela seja movida para estados definidos (ex: 0° para fechada, 90° para aberta). O torque elevado garante que o atuador seja capaz de movimentar o mecanismo da janela sem dificuldade, sendo controlado por um sinal de modulação por largura de pulso (PWM) gerado diretamente pelo ESP32.

3.2. Protocolos de Comunicação

A conectividade é um pilar central do projeto, permitindo que a Janela Inteligente opere de forma autônoma e, ao mesmo tempo, possibilite o monitoramento remoto. A arquitetura de comunicação do dispositivo opera sobre a rede local sem fio (WLAN), utilizando o módulo Wi-Fi integrado ao ESP32.

Sobre a camada de rede Wi-Fi, será implementado o protocolo MQTT (Message Queuing Telemetry Transport) para a troca de mensagens. A escolha do MQTT é fundamentada em sua natureza leve e no modelo publish/subscribe, ideal para dispositivos IoT com recursos limitados e que podem operar em redes com conectividade intermitente.

3.3. Firmware e Lógica de Controle

O software, responsável por integrar o hardware e os protocolos de comunicação, será desenvolvido em linguagem C++ sobre o framework Arduino, utilizando a IDE do Visual Studio Code com a extensão PlatformIO. Seram empregadas bibliotecas consolidadas pela comunidade para a interação com os componentes, com destaque para a DHT.h para o sensor e a PubSubClient.h para a comunicação MQTT.

3.4. Método de Avaliação

Como é proposto uma janela inteligente o formato de avaliação para esse trabalho, será por meio de experimentos e coleta de dados em diferente cenários como de alta humidade

e outros fatores externos, para garantir que o protótipo funcione. Ao decorrer do período de desenvolvimento do trabalho o formato de avaliação pode ser alterado, por opção dos avaliadores.

3.5. Organização da Equipe

Para o gerenciamento e a execução deste projeto, adotou-se uma metodologia ágil com o trabalho organizado em Sprints cíclicos de 15 dias de duração. A equipe, composta por cinco integrantes, teve suas responsabilidades distribuídas no início de cada ciclo para assegurar uma cobertura completa das diferentes frentes de desenvolvimento. As tarefas foram planejadas para abranger desde a pesquisa e documentação em formato de artigo, passando pelo desenvolvimento do software incluindo prototipagem, front-end e back-end com a tecnologia Flutter, até a montagem do hardware com Arduino e a implementação de protocolos de comunicação como o MQTT. Essa abordagem estruturada permitiu um desenvolvimento incremental e iterativo, garantindo o progresso contínuo em todas as áreas do projeto e facilitando a integração entre o software e o hardware a cada entrega quinzenal, conforme a Figura 2.

Sprint do Grupo a cada 15 dias					
Apresentações	Caio Davi	Caio Gomes	Daniel	Lucas	Julio
Sprint 1	Artigo	Protótipo Flutter	Componentes Hardware	Slides	Slide
Sprint 2	MQTT	Flutter - Front	Arduino	Dashboard	Slide - Artigo
Sprint 3	Teste	Flutter - Back	Teste	Flutter - Back	Slide - Artigo

Figure 2. Cronograma

3.6. Desenvolvimento do Sistema

Nesta sub-seção irá discorrer sobre a forma que os componentes foram configurados para que no final tudo seja juntado e o conjunto funcionar como um todo.

3.6.1. Arduino e Raindrops Module

O Arduino escolhido foi o ESP32, pois além de ser um dos modelos de baixo custo e consumo de energia, impactando diretamente no servo motor que será utilizado em etapas futuras, ele é um sistema-em-um-chip, ou seja, um circuito integrado com Wi-Fi e Bluetooth, que permite uma melhor conexão com os componentes virtuais, como os dashboards para análise e predição de tempo. Por fim, o ESP32 tem a funcionalidade de rodar tarefas paralelas, sendo capaz de ler os dados em tempo real do sensor de chuva e enviar dados via Wi-Fi ao mesmo tempo, ajudando na análise para fechar a janela sem esperar muito tempo após o começo de uma chuva mais forte.

Raindrops Module foi o sensor de chuva escolhido, pois é um sensor simples e eficaz para detectar chuva ou umidade a partir da coleta de gotas de água. O sensor é um conjunto de dois componentes: a placa sensora, responsável pela coleta de gotas; e o módulo comparador, que converte o sinal recebido pela placa em digital e analógico.

A conversão desses sinais permite transformar os valores recebidos em resultados. A saída analógica consegue demonstrar o nível de umidade da superfície e a saída digital

indica chuva. No programa desenvolvido, o valor 1 indica que não tem chuva, e o valor 0 indica que tem chuva.

Os três componentes se conectam a partir de *jumpers*. O ESP32 realiza a ligação com o módulo comparador LM393 com 4 *jumpers* principais. O primeiro deles é o VCC para alimentação de energia dos componentes. O segundo é o GND que age como terra, ou seja, age como um ponto de referência para a tensão do circuito, proprocionando o caminho de retorno de energia e evitando sobrecarga. O terceiro é conectado à entrada A0, que funciona como saída analógica para medição do nível de umidade. Por fim, o quarto *jumper* é conectado a entrada D0, que age como saída digital para indicar se há ou não há chuva.

Do outro lado, o sensor de chuva e o módulo comparador se conectam a partir de dois *jumpers*, realizando o processo de passagem de informações de um para o outro, e terminando o processo inicial entre estes 3 componentes.

3.6.2. HiveMQ e Telegraf

O Broker escolhido foi o HiveMQ Cloud, pois ele amplia o alcance de funcionamento do projeto além do wifi, permitindo que os dispositivos se comuniquem a distância. E um dos grandes benefícios dele é a conta ser gratuita e ser alocado em servidor da AWS, que trás uma camada de segurança a mais para o projeto.

Como os dados coletados pelos sensores que estão unidos no arduino são categorizados como séries temporais, é recomendado que o database seja dedicado ao tipo de dado tratado. Dentre os que existem online, no ano de 2025, temoso Influx DB, onde sua arquitetura otimizada para gravação contínua e consultas temporais rápidas o torna ideal para aplicações de IoT e monitoramento em tempo real. Além disso, integra-se nativamente com ferramentas como Telegraf e Grafana, facilitando a coleta e visualização dos dados.

O Telegraf é um agente de coleta de métricas e eventos desenvolvido pela InfluxData, amplamente utilizado para integrar dispositivos, aplicações e bancos de dados em arquiteturas de monitoramento. Ele possui uma estrutura modular composta por inputs, processors e outputs, o que permite receber dados de diversas fontes (como sensores, APIs ou brokers MQTT), processá-los e enviá-los para destinos como o InfluxDB. Essa flexibilidade torna o Telegraf ideal para sistemas de IoT, pois consegue captar mensagens de telemetria publicadas por dispositivos conectados e armazená-las de forma eficiente para análise posterior.

Ao utilizar o Docker, o Telegraf é executado dentro de um contêiner isolado, garantindo portabilidade, facilidade de configuração e reprodutibilidade do ambiente. A imagem oficial do Telegraf contém todos os binários e dependências necessários, bastando montar o arquivo de configuração (telegraf.conf) como volume no contêiner. Assim, o Docker simplifica o gerenciamento do agente, permitindo que ele seja iniciado, parado ou atualizado com um único comando, além de facilitar a integração com outros serviços como o HiveMQ e o InfluxDB, que também podem rodar em contêineres na mesma rede virtual.

3.6.3. Uso do Docker na Integração entre Interface, HiveMQ e InfluxDB

Dentro dessa arquitetura, o Docker desempenha um papel central na orquestração e comunicação entre os serviços. Cada componente a interface web, o Telegraf, o HiveMQ e o InfluxDB roda em seu próprio contêiner isolado, mas todos compartilham uma rede virtual interna criada pelo Docker Compose. Isso significa que os contêineres conseguem se comunicar entre si usando nomes de serviço (por exemplo, hivemq, telegraf e influxdb) em vez de endereços IP, o que elimina a necessidade de configurações manuais complexas de rede.

A interface web (ou o gateway FastAPI, dependendo da camada utilizada) é responsável por enviar comandos MQTT, como "abrir janela" ou "consultar umidade". Esses comandos são publicados em tópicos específicos no HiveMQ, que atua como o broker responsável por intermediar a comunicação entre o backend e o dispositivo físico (Arduino/ESP32). Quando o dispositivo publica suas medições por exemplo, temperatura ou posição essas mensagens também passam pelo HiveMQ, garantindo que os dados fluam de maneira padronizada e segura entre todos os elementos do sistema.

O Telegraf, por sua vez, funciona como um assinante (subscriber) dentro dessa mesma rede Docker. Ele está configurado para ouvir os tópicos MQTT do HiveMQ (como sensors/#) e, ao receber uma nova mensagem, automaticamente a processa e a converte para o formato aceito pelo InfluxDB. Esse processo é realizado internamente dentro do contêiner do Telegraf, sem necessidade de scripts adicionais, pois o agente já possui inputs e outputs nativos para MQTT e InfluxDB.

Finalmente, o InfluxDB armazena todas as medições enviadas pelo Telegraf, permitindo que a interface web consulte os dados em tempo real ou históricos. Assim, o Docker coordena toda essa cadeia: a interface publica comandos no HiveMQ; o dispositivo responde com telemetria; o Telegraf coleta essas respostas e as envia ao InfluxDB. Todo o ciclo de comunicação acontece dentro do ambiente orquestrado pelo Docker, garantindo isolamento, escalabilidade e facilidade de manutenção, além de permitir que qualquer parte do sistema possa ser atualizada ou reiniciada sem afetar os demais serviços.

3.6.4. Interface do Projeto

A interface do projeto *Smart Window* foi desenvolvida em HTML, CSS e JavaScript, com o objetivo de simular de forma interativa o funcionamento de uma janela inteligente. O sistema permite ao usuário visualizar notificações em tempo real, simular condições ambientais e interagir diretamente com o estado da janela, além de oferecer recursos de registro e exportação de dados.

O conceito principal da aplicação gira em torno da alternância entre dois estados: **janela aberta** e **janela fechada**. O botão principal da interface reflete dinamicamente o estado atual, mudando de cor, ícone e texto conforme a ação do usuário. Ao fechar ou abrir a janela, uma nova notificação é gerada e exibida em um painel de histórico, permitindo o acompanhamento das ações realizadas. Além disso, há a possibilidade de simular a detecção de umidade, que também aciona uma notificação e pode sugerir o fechamento automático da janela, representando um comportamento inteligente do sis-

tema.

Cada notificação inclui informações complementares de sensores, como **temperatura (ºc)**, **umidade (º/º)** e **detecção de gotas**, geradas aleatoriamente para simular medições ambientais reais. Esses dados são armazenados localmente em um histórico exibido na interface, possibilitando uma visualização temporal das condições detectadas e das ações executadas.

A aplicação também permite ao usuário **fechar a janela manualmente**, caso deseje simular o controle direto do sistema, bem como limpar o histórico de eventos ou exportar os dados registrados. Para a exportação, o sistema oferece duas opções: o formato **.txt**, que apresenta uma tabela alinhada e de fácil leitura em qualquer editor de texto, e o formato **.csv**, compatível com planilhas como Microsoft Excel e Google Sheets. O formato CSV é vantajoso quando se deseja realizar análises quantitativas ou gerar gráficos a partir dos dados, enquanto o formato TXT é ideal para armazenamento simples e consulta rápida.

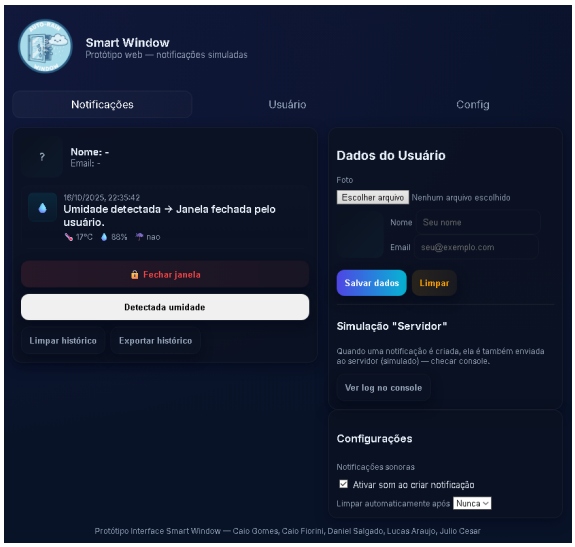

Figure 3. Teste inicial com sensor molhado. Resultado: Chuva detectada

Outro aspecto importante da interface é a **personalização do perfil do usuário**. O protótipo permite inserir nome, e-mail e foto, que são exibidos tanto na área de configurações quanto no resumo principal da interface. Essa funcionalidade adiciona praticidade e uma sensação de identidade ao sistema, tornando a experiência mais personalizada e próxima de uma aplicação real.

Por fim, o projeto conta com opções adicionais, como a **ativação de sons** para alertas e a **limpeza automática** do histórico após determinado tempo, simulando comportamentos típicos de sistemas embarcados inteligentes. O resultado é uma interface moderna, responsiva e intuitiva, que demonstra de forma clara o funcionamento conceitual da Smart Window e suas interações entre hardware, software e usuário.

A interface web do projeto pode ser acessada em: https://kyogomes.github.io/PrototipoAplica-owebTI5/ .

3.6.5. ThingSpeak

O ThingSpeak é uma plataforma online de Internet das Coisas (IoT) utilizada para armazenar, visualizar e analisar dados coletados por sensores. No projeto, ele foi essencial para registrar e monitorar as informações de presença de gotas de chuva, temperatura e umidade do ar, enviadas pelos sensores em tempo real. Com o uso do ThingSpeak, foi possível acompanhar o comportamento climático de forma prática e acessível pela internet, permitindo visualizar gráficos, identificar variações ambientais e facilitar a interpretação dos dados para futuras tomadas de decisão, como prever se há possibilidade de chuva, além de deixar os dados salvos em rede para futura consultas.

4. Resultados Preliminares

Nesta seção são apresentados os resultados obtidos até o momento da *sprint* 2. Ao longo das próximas semanas, esta seção será atualizada e complementada com novos testes e análises.

4.1. Arduino com Raindrops Module

Tal como proposto no desenvolvimento do sistema, a equipe realizou os testes iniciais com os componentes Arduino ESP32 e o sensor de chuva. De forma geral, o código define os valores dos pinos utilizados no ESP32, auxiliando na identifição dos pinos conectados nos *jumpers*. A partir disso, ele inicia a comunicação com o computador e define o estado do pino digital como entrada. Na função do loop principal, ocorre a leitura dos valores dos pinos, e a partir desses resultados, entra uma comparação para verificar se o estado do pino é alto ou baixo. Caso seja baixo, existe a detecção de chuva, caso seja alto, não há chuva.

Figure 4. Teste inicial com o sensor seco. Resultado: Sem chuva

O vídeo postado no Youtube [Salgado 2025] demonstra o protótipo inicial sendo testado, realizando os testes demonstrados nas imagens de testes com sensor.

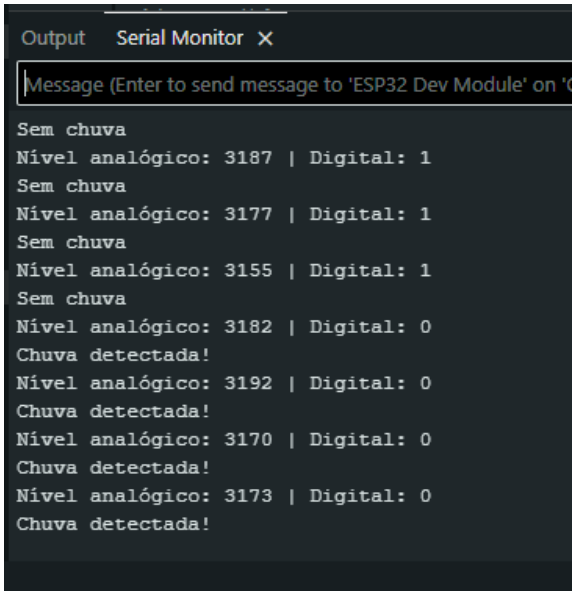

Figure 5. Teste inicial com sensor molhado. Resultado: Chuva detectada

4.2. Conexão Docker e envio de mensagens

Conforme proposto na metodologia, a equipe estruturou uma arquitetura baseada em *containers* Docker, responsável por gerenciar todos os processos de comunicação entre os componentes do sistema. A Figura 6 ilustra o fluxo geral dessa arquitetura.

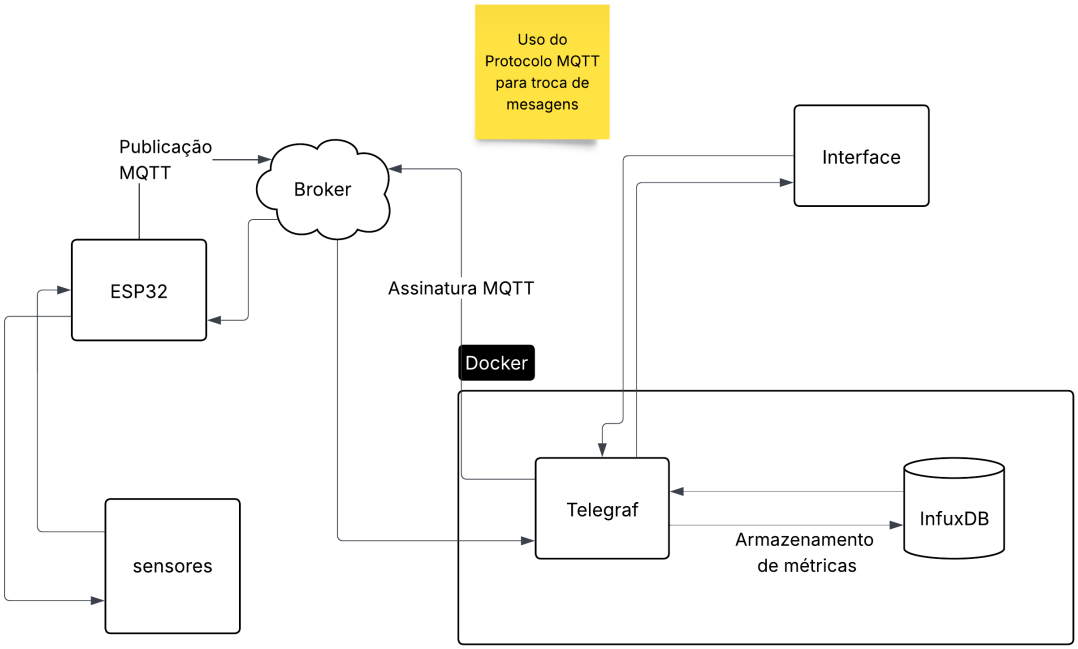

Figure 6. Diagrama de fluxo do sistema.

Foram realizados testes preliminares para validar a comunicação entre os módulos. A partir do terminal, executou-se uma aplicação Python responsável por publicar dados

via MQTT, enquanto outra aplicação consumia os dados diretamente do *gateway*, por meio de requisições à API. As Figuras 7 e 8 apresentam exemplos desses testes.

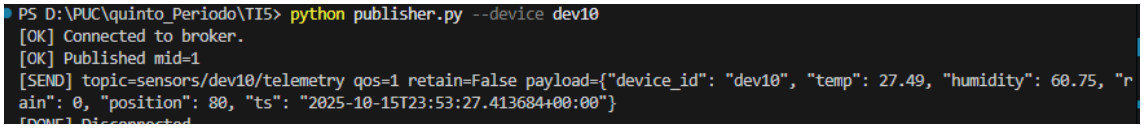

Figure 7. Dado publicado via MQTT.

Figure 8. Dado recuperado pela API.

Os resultados demonstram que os dados estão sendo corretamente gravados no banco de dados e posteriormente recuperados pela API, utilizando o Docker e o Telegraf como elementos de integração.

A Figura 9 exibe o log gerado pelo Telegraf durante a publicação, confirmando a recepção das mensagens e o correto encaminhamento dos dados ao InfluxDB.

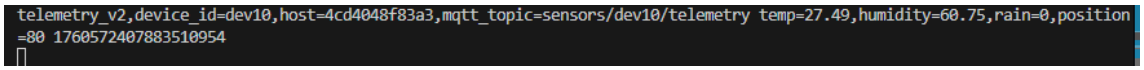

Figure 9. Log do Telegraf confirmando a publicação da mensagem.

Por fim, a Figura 10 apresenta os registros de execução do container Docker, evidenciando as requisições processadas com sucesso e o funcionamento estável do ambiente de comunicação.

Figure 10. Log do Docker com requisições bem-sucedidas.

References

Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., and Guizani, M. (2020). A review of architecture, technologies, and challenges in iot-based smart system. *IEEE Access*, 8:156321–156343.

Farhan, M. S., Mondal, M. N. I., Rahi, M. R. H., Rakib, M. R., and Deb, S. (2021). Design and implementation of a low-cost iot-based smart home system. In *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*, pages 1–6. IEEE.

Giusto, D., Iera, A., Morabito, G., and Atzori, L. (2010). *Embedded Systems for the Internet of Things (IoT)*. Springer Science & Business Media.

Salgado, D. (2025). Protótipo inicial com esp32 e sensor de chuva. `https://youtu.be/-Ad7hn2gZbc`. Acesso em: 16 out. 2025.

Silva, P. R. P. d., Silva, G. D. P. e., Almeida, L. F. d. A. T. B. d., Oliveira, R. A. B. d., and Perkusich, A. (2018). Performance evaluation of the esp32 microcontroller for iot applications. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 556–561. IEEE.

Stojkoska, B. L. R. and Trivodaliev, K. V. (2017). A review of internet of things for smart home: Challenges and solutions. *Journal of Cleaner Production*, 140:1454–1464.

Thangam, A. S. and Rajan, M. M. (2018). A comparison of mqtt and coap for iot. In *2018 International Conference on Communication and Signal Processing (ICCSP)*, pages 0693–0697. IEEE.