



## Hissopo: Sistema de Monitoramento IoT para Qualidade do Ar em Ambientes Internos

Brenda Ribeiro Lacerda Tavares<sup>1</sup>, Andre Luis de Oliveira<sup>1</sup>

<sup>1</sup> Faculdade de Computação e Informática  
Universidade Presbiteriana Mackenzie (UPM) – São Paulo, SP – Brazil

brenda.tavares@mackenzista.com.br

**Abstract.** *This work presents a low-cost IoT prototype for real-time indoor environmental monitoring based on an ESP32 microcontroller. The system uses a DHT22 sensor to measure temperature and humidity and an MQ-135 sensor to detect potentially harmful gases. Data are transmitted via Wi-Fi using the MQTT protocol to a Mosquitto broker, enabling remote visualization through standard MQTT clients. The prototype includes a red LED and a buzzer, which activate automatically or by remote command when critical thresholds are exceeded. Results indicate low latency, reliable communication and effective alert mechanisms, demonstrating the system's potential for indoor monitoring applications.*

**Resumo.** *Este trabalho apresenta um protótipo IoT de baixo custo para monitoramento ambiental em tempo real, utilizando o microcontrolador ESP32. O sistema emprega o sensor DHT22 para medir temperatura e umidade e o sensor MQ-135 para detectar gases potencialmente nocivos. Os dados são transmitidos via Wi-Fi utilizando o protocolo MQTT para um broker Mosquitto, permitindo visualização remota por clientes MQTT. O protótipo inclui um LED vermelho e um buzzer, acionados automaticamente ou por comando remoto quando limites críticos são ultrapassados. Os resultados obtidos indicam baixa latência, comunicação confiável e alertas eficientes, demonstrando seu potencial para aplicações em ambientes internos.*

### 1. Introdução

A poluição do ar em ambientes internos representa um desafio relevante de saúde pública, cujos impactos são frequentemente subestimados. Segundo a Organização Mundial da Saúde (OMS), a poluição atmosférica, tanto interna quanto externa, está entre os maiores riscos ambientais à saúde humana, contribuindo para milhões de mortes prematuras todos os anos (NAÇÕES UNIDAS NO BRASIL, 2021). Em espaços fechados, a concentração de poluentes como compostos orgânicos voláteis, dióxido de carbono e outros gases pode ser significativamente maior do que no ambiente externo, expondo os ocupantes a riscos crônicos, como doenças respiratórias, alergias e problemas cardiovasculares (GAUER et al., 2011). Esse cenário é especialmente preocupante para crianças, idosos e indivíduos com condições de saúde preexistentes.

A preocupação global com a qualidade do ar alinha-se diretamente ao Objetivo de Desenvolvimento Sustentável (ODS) 3, que busca assegurar uma vida saudável e promover o bem-estar para todas as faixas etárias, incluindo a redução de doenças associadas à poluição ambiental. Diante desse contexto, a Internet das Coisas (IoT) se apresenta como uma abordagem promissora, permitindo o monitoramento ambiental contínuo por meio de sensores de baixo custo capazes de coletar dados em tempo real. Conforme destaca Rosa et al. (2020), a IoT aplicada à saúde ambiental possibilita o desenvolvimento de sistemas proativos, apoiando indivíduos na tomada de decisões informadas sobre seu ambiente de convivência.

Este projeto propõe o desenvolvimento de um sistema IoT para monitoramento da qualidade do ar em ambientes internos, utilizando sensores para medir temperatura, umidade e concentração de gases potencialmente nocivos. O protótipo emprega o microcontrolador ESP32 e integra sensores como o DHT22 e o MQ-135, responsáveis pela captura dos parâmetros ambientais. As informações coletadas são transmitidas via Wi-Fi utilizando o protocolo MQTT para um broker Mosquitto em rede local, permitindo análise, visualização e acionamento remoto de atuadores. O sistema emite alertas imediatos, tanto visuais quanto sonoros, quando os valores monitorados ultrapassam limites seguros.

Ao oferecer uma solução acessível, reativa e alinhada às tecnologias IoT contemporâneas, este trabalho busca contribuir para a democratização do monitoramento ambiental e colaborar para o alcance das metas estabelecidas pelo ODS 3, promovendo saúde, segurança e bem-estar em ambientes fechados.

## **2. Materiais e métodos**

O protótipo do Projeto Hissopo utiliza uma série de componentes eletrônicos integrados ao microcontrolador ESP32 DOIT Devkit V1 (ESP-WROOM-32), que constitui o núcleo do sistema. O ESP32 é equipado com conectividade Wi-Fi e Bluetooth, permitindo a coleta, o processamento e a transmissão dos dados ambientais, além de controlar os atuadores e exibir as informações no display LCD.

Para o monitoramento ambiental, são utilizados dois sensores principais. O DHT22 é responsável pela medição da temperatura e da umidade relativa do ar, sendo escolhido devido à sua precisão e estabilidade. O MQ-135 realiza a detecção de gases potencialmente nocivos, como dióxido de carbono, amônia e benzeno, fornecendo uma leitura analógica que representa a qualidade geral do ar no ambiente monitorado.

A interface visual do sistema é composta por um display LCD 16×2 acoplado a um módulo I<sup>2</sup>C (PCF8574), o que reduz significativamente a quantidade de conexões necessárias e facilita a integração com o ESP32. Para a sinalização de alertas, utiliza-se um LED vermelho, que indica situações críticas, e um buzzer ativo, responsável por emitir avisos sonoros quando limites de segurança são ultrapassados.

Todos os componentes são conectados por meio de jumpers montados sobre uma protoboard, utilizando resistores sempre que necessário para garantir a correta limitação de corrente. A alimentação do sistema e o carregamento do código são realizados via cabo USB ligado ao computador, utilizando a plataforma Arduino IDE para desenvolvimento e upload do firmware.

No que diz respeito à comunicação IoT, o protótipo integra-se à rede utilizando o protocolo MQTT (Message Queuing Telemetry Transport), amplamente adotado em aplicações de Internet das Coisas pela sua leveza e eficiência. O ESP32 se conecta via Wi-Fi a um broker MQTT, que atua como intermediário entre o microcontrolador e os dispositivos clientes responsáveis pela visualização e controle do sistema.

O broker utilizado é o Mosquitto, instalado em um computador local, desempenhando o papel de servidor responsável por receber as publicações do ESP32 — contendo dados de temperatura, umidade e qualidade do ar — e distribuí-las aos clientes inscritos nos tópicos correspondentes. Ferramentas como MQTT Explorer e Node-RED são empregadas para monitorar essas publicações em tempo real e enviar, quando desejado, comandos que acionam os atuadores.

A comunicação ocorre de forma assíncrona por meio de tópicos organizados hierarquicamente, nos quais o ESP32 publica periodicamente as leituras ambientais e permanece inscrito em tópicos de controle destinados ao LED vermelho e ao buzzer. Essa arquitetura possibilita tanto o monitoramento contínuo quanto a atuação remota sobre o sistema.

Assim, a integração entre ESP32, broker Mosquitto e clientes MQTT estabelece um fluxo de comunicação confiável, permitindo o funcionamento de um sistema IoT completo, responsivo e alinhado aos princípios do protocolo MQTT, garantindo ao usuário um monitoramento ambiental eficiente e em tempo real.

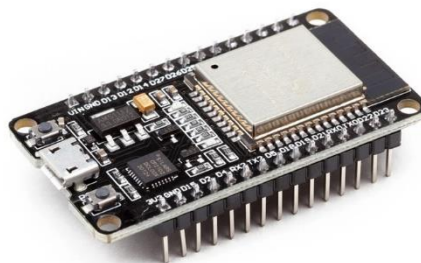
## **2. 1. Componentes de hardware**

A seguir são descritos os principais componentes de hardware utilizados no protótipo de monitoramento de qualidade do ar com ESP32, sensores de temperatura e umidade (DHT22), sensor de qualidade do ar (MQ-135), display LCD I<sup>2</sup>C, LED vermelho e o buzzer ativo.

### **2.1.1. Microcontrolador ESP32**

O ESP32 DOIT DevKit V1 é um microcontrolador de alto desempenho da Espressif Systems, baseado no chip ESP32-WROOM-32.

Ele opera com uma tensão de 3.3V, possuindo pinos GPIO (General Purpose Input/Output) para se comunicar com outros dispositivos periféricos.

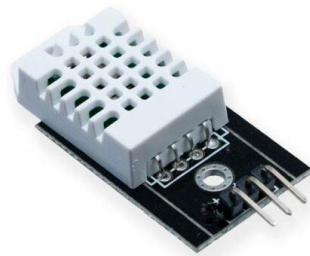


**Figura 1 - Microcontrolador ESP32. Fonte: Victor vision (2023).**

### 2.1.2. Sensor DHT22

O DHT22 é um sensor digital de temperatura e umidade que utiliza um único pino de comunicação, sendo assim fácil a integração em sistemas embarcados como o ESP32.

O sensor possui uma boa precisão, sendo capaz de medir a temperatura na faixa de  $-40^{\circ}\text{C}$  a  $80^{\circ}\text{C}$  e a umidade relativa do ar entre 0% e 100%.



**Figura 2 - Sensor DHT22. Fonte: core eletronic (2025).**

### 2.1.3. Sensor MQ-135

O MQ-135 é um sensor de gases analógico usado para medir a qualidade do ar, sendo capaz de detectar uma ampla gama de gases, fumaça e outros compostos orgânicos voláteis.



**Figura 3 - Sensor de qualidade de ar MQ-135. Fonte: maker hero (2022).**

### 2.1.4. Display LCD

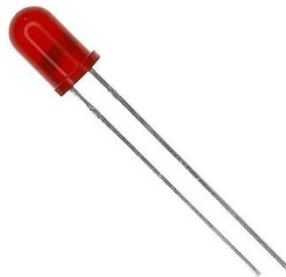
O LCD 16x2 com módulo I<sup>2</sup>C é um display de cristal líquido utilizado para exibir as leituras dos sensores. O módulo I<sup>2</sup>C (Inter-Integrated Circuit) simplifica a comunicação com o ESP32, permitindo a exibição de até 32 caracteres por vez.



**Figura 4 - Display LCD. Fonte: Mamute eletrônico (2025).**

#### **2.1.5. LED vermelho**

O LED vermelho foi utilizado para indicar visualmente o status de operação do protótipo sendo ele acionado por um sinal digital enviado por um pino do ESP32 quando os níveis de gases como o gás carbônico, por exemplo, estiverem acima do padrão normativo.



**Figura 5 - LED Vermelho. Fonte: Tetra comp (2025).**

#### **2.1.6. Buzzer ativo**

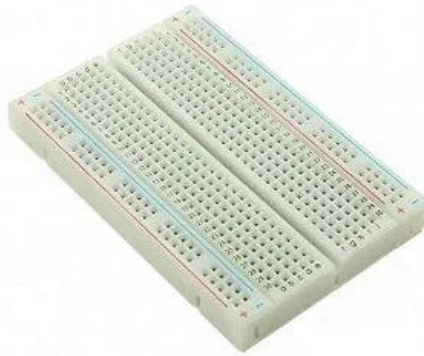
O buzzer ativo foi usado no protótipo para emitir um sinal sonoro quando determinadas condições forem atendidas (por exemplo, qualidade do ar abaixo de um valor pré-determinado). O buzzer é acionado através de um pino digital do ESP32.



**Figura 6 - Buzzer Ativo. Fonte: Eletrogate (2025).**

#### **2.1.7. Protoboard**

A protoboard é uma plataforma de plástico com furos e condutores que permite de forma fácil inserir e conectar através de fios condutores componentes eletrônicos e por isso foi selecionado para montar o protótipo de forma temporária e facilitar os testes.



**Figura 7 - Protoboard. Fonte: Eletrogate (2025).**

#### **2.1.8. Jumpers**

Para a montagem do protótipo foram utilizados dois tipos diferentes de jumpers, o macho-macho e o macho-fêmea.

Jumpers são cabos elétricos que possuem dois conectores do tipo pino, sendo ideal para realizar ligações na placa protoboard.

Através deles serão feitas as ligações necessárias entre os componentes e o ESP32.

Apesar de o protótipo demandar menos ligações, serão adquiridas vinte unidades de jumpers, para que haja espaço para testes e um lastro para a necessidade de alguma outra ligação.



**Figura 8 - Jumpers macho-macho. Fonte: Eletrogate (2025)**



**Figura 9 – Jumpers macho-fêmea. Fonte: Robocore (2025)**

### **2.1.9 Resistores**

Os resistores são componentes eletrônicos fundamentais utilizados para limitar a passagem de corrente elétrica em circuitos, protegendo sensores, atuadores e microcontroladores contra danos provocados por sobrecorrente. Seu funcionamento baseia-se na Lei de Ohm, segundo a qual a resistência elétrica ( $R$ ), expressa em ohms ( $\Omega$ ), determina a relação entre tensão ( $V$ ) e corrente ( $I$ ) em um condutor. Assim, resistores permitem controlar com precisão o fluxo elétrico, garantindo estabilidade e segurança no sistema eletrônico (BOYLESTAD; NASHELSKY, 2012).

No desenvolvimento do presente projeto, foram empregados três resistores, cada um desempenhando uma função específica no circuito

O sensor MQ-135 opera com uma saída analógica cuja tensão pode variar em função da concentração de gases presentes no ambiente. Para que essa saída seja corretamente interpretada pelo conversor analógico-digital (ADC) do ESP32 — cujo limite é de aproximadamente 3,3 V, sendo necessário implementar um divisor de tensão, formado pelos resistores de 10 k $\Omega$  e 22 k $\Omega$ .

O divisor reduz a tensão da saída A0 do MQ-135, garantindo que o sinal fique dentro do limite seguro do ADC e estabiliza o sinal analógico, diminuindo ruídos e oscilações que prejudicariam a precisão das leituras.



**Figura 10 – Resistor de 10K Ohms. Fonte: Proesi (2025)**

Foi utilizado um resistor de 220 ohms em série com o LED vermelho para limitar a corrente elétrica, evitando que o diodo emissor de luz seja danificado por excesso de corrente proveniente do GPIO do ESP32. Este resistor encontra-se conectado entre o pino positivo (ânodo) do LED e o pino digital configurado para acionamento.



**Figura 11 – Resistor de 220 Ohms. Fonte: Casa da robótica (2025)**

No circuito, o resistor de 22 k $\Omega$  é ligado entre a saída A0 do MQ-135 e a entrada analógica do ESP32, enquanto o resistor de 10 k $\Omega$  é conectado entre essa mesma entrada e o GND, formando o divisor padrão.



**Figura 12 – Resistor de 22k Ohms. Fonte: Proesi (2025)**

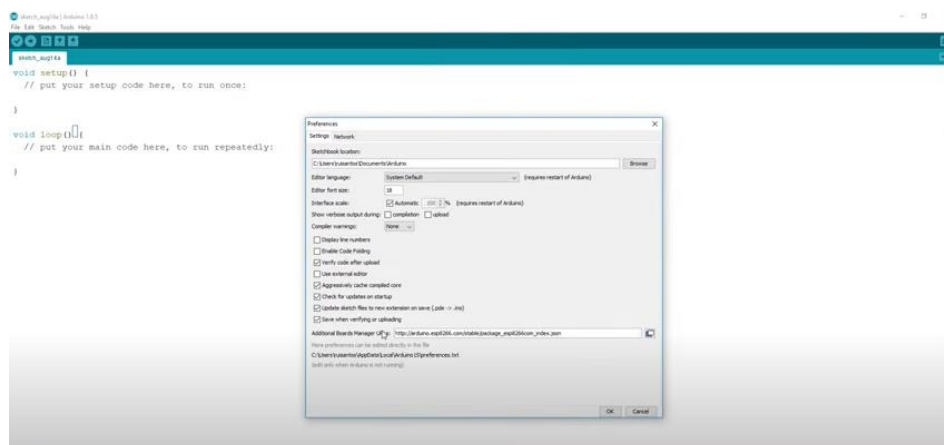
A escolha dos valores dos resistores seguiu recomendações do datasheet dos componentes utilizados e práticas consolidadas em eletrônica aplicada. A disposição física dos resistores pode ser visualizada no diagrama do projeto, onde cada um está conectado em série com o componente correspondente, montado sobre a protoboard e interligado ao ESP32.

## **2.2. Componentes de software**

### **2.2.1. Arduino IDE**

A plataforma de desenvolvimento Arduino IDE foi escolhida para a programação do ESP32.

Esta plataforma é amplamente utilizada para o desenvolvimento de sistemas embarcados devido à sua simplicidade e suporte a diversas bibliotecas. Para o ESP32, é necessário instalar o pacote específico da no Arduino IDE para garantir que todas as funcionalidades do microcontrolador sejam suportadas.



**Figura 13: Tela do Arduino IDE durante a instalação do pacote compatível com ESP32. Fonte: Rui Santos (2018).**

### 2.2.2. Bibliotecas utilizadas

Para o desenvolvimento do projeto, são empregadas bibliotecas específicas que viabilizam a comunicação entre o microcontrolador e os diversos sensores e módulos integrados ao protótipo. A biblioteca DHT.h é utilizada para estabelecer a interface com o sensor DHT22, responsável pela medição de temperatura e umidade do ambiente. Essa biblioteca abstrai a captura dos sinais digitais gerados pelo sensor, convertendo-os automaticamente em valores numéricos acessíveis ao sistema e prontos para exibição ou processamento.

A biblioteca Wire.h é empregada para a comunicação com o display LCD por meio da interface I<sup>2</sup>C, permitindo a exibição clara e organizada das informações coletadas pelos sensores. O uso do protocolo I<sup>2</sup>C reduz a quantidade de conexões físicas necessárias, utilizando apenas duas linhas (SDA e SCL) para transferência de dados, o que otimiza o uso das portas disponíveis no ESP32 e simplifica o circuito eletrônico.

No caso do sensor MQ-135, destinado à detecção de gases e à avaliação da qualidade do ar, a leitura dos valores é realizada através da função `analogRead()` da Arduino IDE. Essa função converte a tensão analógica produzida pelo sensor em um valor digital compatível com o conversor ADC do ESP32, permitindo ao sistema interpretar variações na concentração de gases presentes no ambiente.

Além dessas bibliotecas, o projeto utiliza a `PubSubClient.h` (citada em seções posteriores), responsável por implementar a comunicação via protocolo MQTT, garantindo que os dados coletados possam ser transmitidos ao broker Mosquitto e que os atuadores respondam a comandos remotos.

Assim, o uso dessas bibliotecas e funções é essencial para integrar os componentes do sistema de forma eficiente, modular e confiável, facilitando o desenvolvimento do protótipo e assegurando o correto funcionamento de todas as etapas do monitoramento ambiental.

### 2.3. Montagem do protótipo

A montagem do protótipo segue uma sequência estruturada de etapas, garantindo a correta integração entre os componentes eletrônicos e o microcontrolador ESP32. Inicialmente,

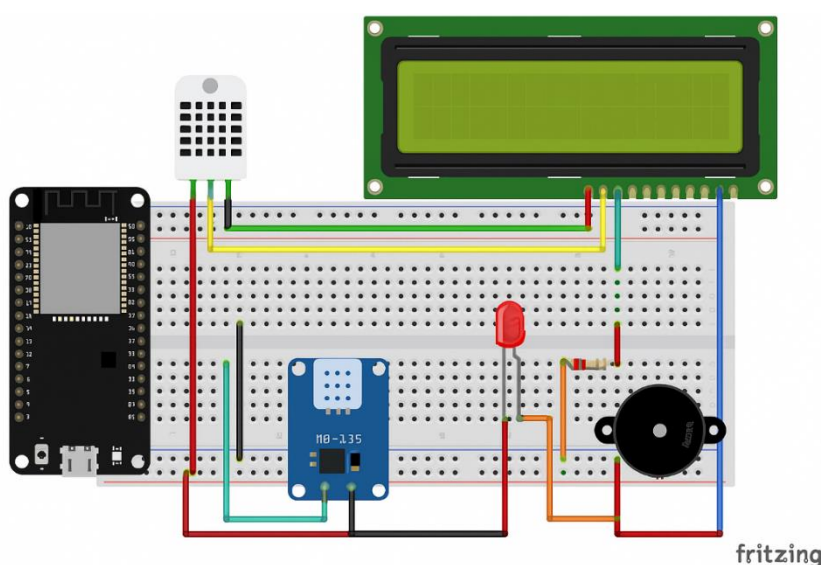
realiza-se a organização dos módulos na protoboard, facilitando ajustes e inspeções durante o desenvolvimento. O ESP32 é posicionado fora da protoboard mas a alimentação alimentado pelos pinos VIN(5V) e GND, atuando como unidade principal de processamento e controle. Além disso a sua saída 3V3 alimenta exclusivamente o sensor DHT22.

O sensor DHT22 é conectado ao pino GPIO 4 (DATA), enquanto seus pinos de alimentação são ligados ao 3V3 (VCC) e GND, possibilitando a leitura contínua de temperatura e umidade. Em seguida, o sensor MQ-135, responsável pela detecção de gases, é interligado ao pino GPIO 34 (entrada analógica). Como o sensor opera com tensão nominal de 5 V, proveniente do cabo USB conectado ao ESP32, utiliza-se um divisor de tensão de 20 k $\Omega$  e 10 k $\Omega$  para adequar o sinal à faixa suportada pelo conversor analógico-digital (ADC) do ESP32, prevenindo sobrecarga e eventuais danos ao microcontrolador.

O display LCD 16 $\times$ 2 com interface I<sup>2</sup>C é conectado aos pinos GPIO 21 (SDA) e GPIO 22 (SCL), além dos pinos de alimentação VCC e GND, permitindo a visualização das informações coletadas de forma clara e acessível. Para a sinalização visual de estados críticos, um LED vermelho é conectado ao pino GPIO 25, em série com um resistor de 220  $\Omega$ , possibilitando o uso seguro do componente e garantindo luminosidade adequada.

Um buzzer ativo é ligado ao pino GPIO 26 e ao GND, permitindo a emissão de alertas sonoros sempre que o sistema identificar valores perigosos de gases ou condições ambientais fora dos limites estabelecidos. Todas as conexões são realizadas com jumpers, permitindo rápida manutenção, substituição de componentes e ajustes experimentais ao longo da prototipagem.

A representação esquemática da montagem é elaborada utilizando o software Fritzing, que permite visualizar de forma clara as conexões entre sensores, atuadores e o microcontrolador, servindo como referência técnica para reprodução do sistema e para documentação formal do projeto.



**Figura 14 – Protótipo Projeto Hissopo. Fonte: Autora (2025)**

**Tabela 1 - Lista descritiva de materiais.**

<b>Componente</b>	<b>Função</b>	<b>Quantidade</b>	<b>Observações</b>
ESP32 DOIT DevKit V1	Microcontrolador principal com Wi-Fi/Bluetooth	1	Responsável pelo controle e comunicação MQTT
Sensor DHT22	Mede temperatura e umidade	1	Saída digital via pino GPIO4
Sensor MQ-135	Mede qualidade do ar (CO <sub>2</sub> , amônia, etc.)	1	Saída analógica via GPIO34
LCD 16x2 com módulo I2C	Exibição local de dados	1	Comunicação via GPIO21 (SDA) e GPIO22 (SCL)
LED vermelho	Alerta visual	1	Acionado quando há níveis críticos
Buzzer ativo	Alerta sonoro	1	Acionado com o LED em situações críticas
Protoboard	Montagem do circuito	1	Permite conexões temporárias
Jumpers macho-macho	Ligações entre componentes	10	Facilita ajustes durante testes
Jumpers macho-fêmea	Ligações entre componentes	10	Facilita ajustes durante testes
Resistores 220Ω	Proteção de componentes	1	Usado para LED e divisor de tensão
Resistores 22KΩ	Proteção de componentes	1	Divisor de tensão para saída A0 do MQ-135
Resistores 22KΩ	Proteção de componentes	1	Divisor de tensão para saída A0 do MQ-135
Cabo USB	Alimentação e upload de código	1	Conecta o ESP32 ao computador

Na segunda etapa, realiza-se a programação do ESP32 utilizando a plataforma Arduino IDE. Inicialmente, instala-se o pacote de placas da Espressif para que a IDE reconheça o microcontrolador e permita sua configuração adequada. Em seguida, desenvolve-se o código-fonte responsável por realizar a leitura dos sensores DHT22 e MQ-135, exibir as informações no display LCD 16×2 com interface I<sup>2</sup>C e estabelecer a conexão com a rede Wi-Fi local.

Após conectar-se à rede, o ESP32 passa a se comunicar com o broker MQTT Mosquitto, que intermedia a troca de mensagens entre o microcontrolador e os dispositivos clientes responsáveis pelo monitoramento do sistema. O ESP32 publica periodicamente as leituras dos sensores em tópicos organizados, permitindo que

ferramentas como MQTT Explorer ou Node-RED recebam, processem e visualizem esses dados em tempo real.

O programa também implementa o controle dos atuadores, o LED vermelho e o buzzer, que podem ser acionados automaticamente quando valores críticos são detectados ou manualmente por meio de comandos enviados por clientes MQTT. Para isso, o ESP32 permanece inscrito em tópicos de controle, reagindo imediatamente às mensagens recebidas e garantindo resposta rápida do sistema.

Após a verificação e compilação do código, o firmware é carregado no ESP32 por meio do cabo USB conectado ao computador, concluindo o processo de programação e permitindo a inicialização do sistema de monitoramento ambiental com suporte completo ao protocolo MQTT.

### **2.3.1. Desenvolvimento do protótipo**

O desenvolvimento do protótipo é conduzido de maneira experimental e exploratória, com foco na validação prática do conceito de monitoramento ambiental via Internet das Coisas (IoT). Essa abordagem permite observar, em tempo real, o comportamento dos sensores, atuadores e do microcontrolador em situações controladas e reproduzíveis, garantindo que o sistema opere de forma coerente com os objetivos propostos.

Para a programação do microcontrolador ESP32, emprega-se a plataforma Arduino IDE em conjunto com as bibliotecas DHT.h, Wire.h, LiquidCrystal\_I2C.h e PubSubClient.h. A biblioteca DHT.h gerencia a captura dos dados de temperatura e umidade provenientes do sensor DHT22; a Wire.h, associada à LiquidCrystal\_I2C.h, viabiliza a comunicação com o display LCD por meio do protocolo I<sup>2</sup>C, simplificando o circuito e reduzindo o número de conexões; e a biblioteca PubSubClient.h implementa o protocolo MQTT, permitindo que o ESP32 publique e receba mensagens do broker Mosquitto.

A comunicação IoT do protótipo ocorre por meio do protocolo MQTT, que gerencia a troca de informações entre o microcontrolador e o servidor MQTT. Neste projeto, utiliza-se o broker Mosquitto, instalado em um computador local, responsável por intermediar a transmissão das leituras ambientais e o recebimento de comandos dos atuadores. O ESP32 publica periodicamente os valores obtidos pelos sensores e permanece inscrito em tópicos de controle que possibilitam o acionamento remoto do LED e do buzzer por meio de clientes MQTT, como MQTT Explorer ou Node-RED.

Após a verificação e compilação do código, o firmware é carregado no ESP32 por meio do cabo USB conectado ao computador, concluindo a etapa de desenvolvimento lógico e permitindo o início dos testes funcionais de leitura, exibição e comunicação via MQTT.

### **2.4. Conexão com o broker MQTT**

A comunicação remota do sistema é viabilizada por meio da integração entre o ESP32 e o broker MQTT Mosquitto, que atua como intermediário entre o protótipo e os dispositivos clientes. O broker recebe as mensagens publicadas pelo microcontrolador e as disponibiliza imediatamente para qualquer cliente conectado e inscrito nos mesmos tópicos.

O MQTT é amplamente utilizado em soluções IoT devido à sua leveza, baixa latência e confiabilidade, tornando-o adequado para aplicações que exigem monitoramento contínuo e respostas rápidas, como no caso do controle de qualidade do ar. O uso de um

broker local também permite maior previsibilidade, autonomia e velocidade na troca de mensagens durante os testes.

Por meio dessa infraestrutura, o usuário consegue acompanhar todas as informações transmitidas pelo ESP32 em tempo real através de ferramentas como MQTT Explorer, que exibe graficamente os tópicos, payloads e o fluxo de mensagens do sistema.

#### **2.4.1. Configuração do broker MQTT**

A configuração do sistema inicia-se pela preparação do ambiente MQTT, responsável por intermediar a comunicação entre o microcontrolador ESP32 e os dispositivos clientes que monitoram e controlam o protótipo. Para este projeto, utiliza-se o broker Mosquitto, instalado em um computador local, que atua como servidor central de mensagens. Após a instalação, o broker é inicializado e passa a gerenciar publicações e assinaturas de tópicos, estruturando o fluxo de dados do sistema.

A organização dos tópicos segue uma hierarquia lógica que separa leituras ambientais e comandos de controle. Nesse modelo, o ESP32 publica continuamente as informações coletadas pelos sensores em tópicos específicos, enquanto permanece inscrito em outros tópicos destinados ao acionamento dos atuadores. Essa estrutura permite que diferentes clientes MQTT acessem seletivamente apenas os dados relevantes, garantindo eficiência e escalabilidade na comunicação.

Ferramentas como MQTT Explorer e Node-RED são utilizadas para visualizar os valores transmitidos e enviar comandos ao sistema. Por meio dessas interfaces, o usuário pode acompanhar as leituras em tempo real, analisar variações ambientais e acionar o LED vermelho ou o buzzer a distância, enviando mensagens simples ao broker. Dessa forma, o Mosquitto desempenha o papel central na integração IoT do protótipo, assegurando comunicação bidirecional confiável entre sensores, atuadores e o ambiente de monitoramento.

#### **2.4.2. Configuração do código para MQTT**

No código desenvolvido na Arduino IDE, utilizo a biblioteca PubSubClient.h para estabelecer a comunicação com o broker Mosquitto. Após a conexão ao Wi-Fi, o ESP32 conecta-se ao servidor MQTT e inicia o processo de publicação periódica das medidas coletadas pelos sensores. A função callback() trata as mensagens recebidas nos tópicos de controle, permitindo que o LED e o buzzer sejam ativados ou desativados instantaneamente conforme os comandos enviados pelos clientes.

Esse mecanismo transforma o protótipo em um sistema interativo, no qual o usuário pode executar ações remotas e monitorar continuamente o ambiente, com respostas rápidas e comunicação confiável baseada no protocolo MQTT.

### **2.5. Funcionamento do sistema**

O funcionamento do sistema baseia-se em um processo integrado de coleta, transmissão e resposta automatizada, permitindo o monitoramento contínuo da qualidade do ar de forma inteligente e acessível. O ESP32 atua como núcleo central do protótipo, realizando leituras periódicas dos sensores e processando as informações obtidas.

O sensor DHT22 capta dados referentes à temperatura e à umidade do ambiente, enviando-os em formato digital para o microcontrolador com precisão adequada ao

contexto da aplicação. Em paralelo, o sensor MQ-135 detecta a presença e concentração de gases poluentes por meio de uma saída analógica, que é convertida em sinal digital pelo conversor ADC do ESP32. Essa combinação permite uma análise abrangente das condições atmosféricas internas, possibilitando a identificação de variações críticas no ambiente.

Após o processamento, as informações são exibidas no display LCD e publicadas no broker MQTT em seus respectivos tópicos. Clientes conectados, como o MQTT Explorer, recebem essas mensagens imediatamente, garantindo monitoramento remoto em tempo real.

Quando valores críticos são detectados, o sistema reage automaticamente: o LED vermelho é acionado como alerta visual, o buzzer emite um aviso sonoro, e os valores anormais são publicados nos tópicos correspondentes, permitindo que o usuário identifique prontamente a situação.

Com essa arquitetura, o protótipo se consolida como uma solução completa de IoT, unindo sensores, atuadores, conectividade em rede e comunicação eficiente baseada no protocolo MQTT.

## **2.6. Testes de sensores**

Após a conclusão da montagem do circuito e da implementação do firmware, realizam-se testes práticos com o objetivo de validar o desempenho dos sensores e a confiabilidade da comunicação via protocolo MQTT. Esses testes têm como finalidade confirmar se os valores medidos são exibidos corretamente no display LCD, publicados no broker Mosquitto e apresentados de forma adequada no Monitor Serial da IDE do Arduino, garantindo a observação simultânea das informações em diferentes interfaces.

Para avaliar o sensor DHT22, o experimentador aproxima uma vela e um secador de cabelo do componente, simulando variações graduais de temperatura. Durante o ensaio, observa-se se a elevação térmica é imediatamente refletida no LCD, no Monitor Serial da IDE do Arduino e nos dados recebidos no Prompt de Comando, onde são consultados os tópicos MQTT publicados pelo ESP32. A umidade também é monitorada, verificando-se se oscilações ambientais são corretamente detectadas e transmitidas.

O sensor MQ-135 é testado utilizando-se a fumaça liberada pela ignição de um fósforo, que fornece uma fonte segura e controlada de gases. Observa-se, nesse processo, se o aumento da concentração de compostos químicos aciona corretamente os atuadores (LED e buzzer) e se as leituras atualizadas chegam simultaneamente ao LCD, ao Monitor Serial e ao broker MQTT, possibilitando sua visualização no Prompt de Comando por meio do Mosquitto.

Esses testes confirmam a integração adequada entre hardware, software e o protocolo MQTT, demonstrando que o protótipo opera de maneira estável, responsiva e eficaz como solução de monitoramento ambiental em tempo real.

## **3.Resultados**

### **3.1. Montagem física do protótipo**

A Figura 15 apresenta o protótipo Hissopo já montado sobre a protoboard, com todos os componentes devidamente conectados ao ESP32.

É possível observar a disposição modular dos sensores DHT22 e MQ-135, o display LCD 16x2 com interface I<sup>2</sup>C, além do LED vermelho e do buzzer ativo utilizados como atuadores.

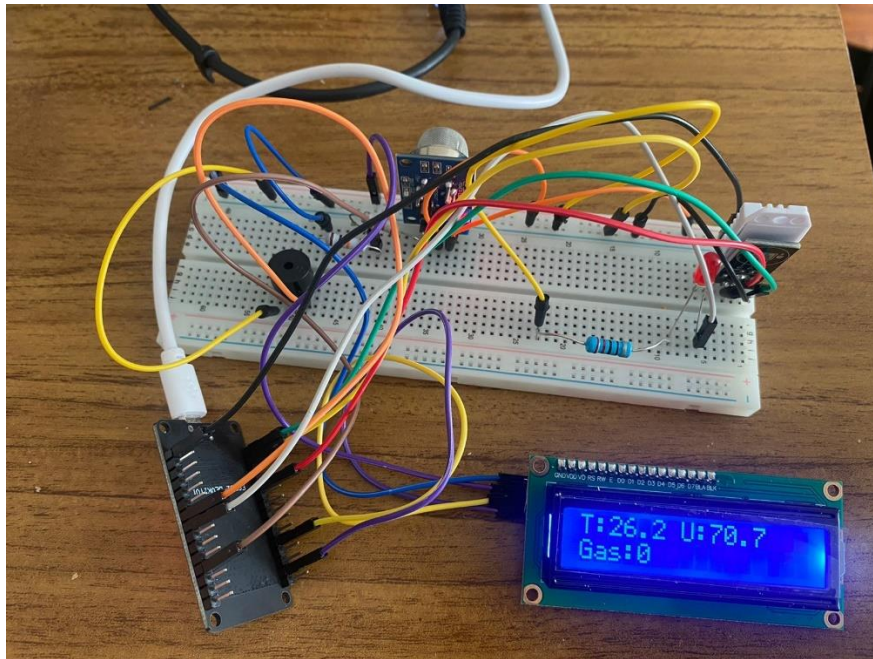


Figura 15– Hardware do Projeto Hissopo. Fonte: Autora (2025)

### 3.2. Comunicação MQTT e visualização dos dados

A Figura 16 apresenta a interface do cliente MQTT Explorer utilizado no experimento, no qual é possível observar a recepção contínua das mensagens publicadas pelo ESP32 nos tópicos do projeto.

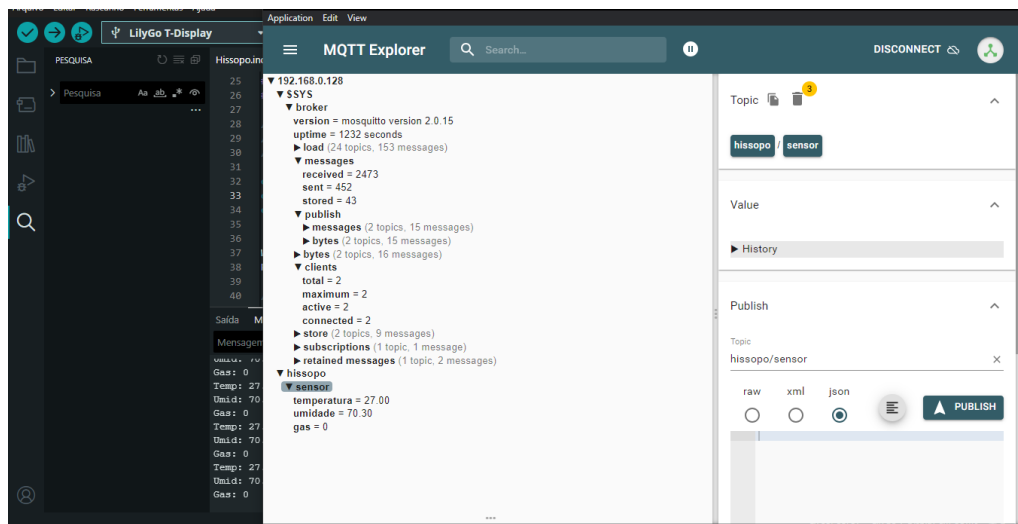


Figura 16– Interface do cliente MQTT do Projeto Hissopo. Fonte: Autora (2025)

### 3.3. Testes de resposta dos sensores e atuadores

Para avaliar o tempo de resposta do sistema, foram realizados testes experimentais medindo o tempo entre a mudança física detectada pelo sensor e sua publicação no broker MQTT e o tempo entre o envio de um comando MQTT e o acionamento do atuador (LED ou buzzer).

Cada teste foi repetido quatro vezes, conforme exigido pela disciplina.

**Tabela 2 – Medições de tempo de resposta**

<b>Sensor/Atuador</b>	<b>Núm. Medida</b>	<b>Tempo de Resposta</b>
LED	1	80 ms
LED	2	70 ms
LED	3	75 ms
LED	4	60 ms
Buzzer	1	85 ms
Buzzer	2	90 ms
Buzzer	3	70 ms
Buzzer	4	75 ms
DHT22	1	1,8 s
DHT22	2	1,4 s
DHT22	3	1,6 s
DHT22	4	1,3 s
MQ-135	1	1,2 s
MQ-135	2	1,5 s
MQ-135	3	1,1 s
MQ-135	4	1,4 s

### 3.4. Cálculo do tempo médio e análise

A análise dos tempos de resposta dos sensores e atuadores do protótipo permite avaliar, de maneira objetiva, o desempenho geral do sistema e sua adequação ao propósito de monitoramento ambiental em tempo quase real. A partir das quatro medições realizadas para cada componente, foi possível calcular o tempo médio de funcionamento, o que

oferece uma visão mais consistente sobre a estabilidade da comunicação entre o ESP32, os sensores, os atuadores e o broker MQTT.

Os atuadores, LED e buzzer, apresentaram tempos de resposta significativamente baixos, sempre na faixa de poucos milissegundos. Isso evidencia que a atuação do sistema frente a situações de risco ocorre praticamente de maneira instantânea. Uma vez que esses dispositivos são responsáveis pelos alertas ao usuário, a baixa latência é indispensável para garantir segurança e rápida percepção de mudanças ambientais. Assim, os resultados reforçam que o protótipo foi capaz de reagir prontamente quando as condições simuladas ultrapassaram os limites estabelecidos, cumprindo sua função de alerta de forma eficiente.

No caso dos sensores, o comportamento observado também se mostrou coerente com as especificações técnicas e com a natureza de cada dispositivo. O DHT22, responsável pela medição de temperatura e umidade, registrou um tempo médio superior ao dos demais componentes, o que já era esperado. Trata-se de um sensor digital que realiza um processamento interno antes de disponibilizar seus valores, o que naturalmente eleva o tempo de resposta. Mesmo assim, o desempenho obtido ficou dentro do intervalo previsto pelo fabricante, demonstrando que o sensor opera de maneira confiável dentro do protótipo.

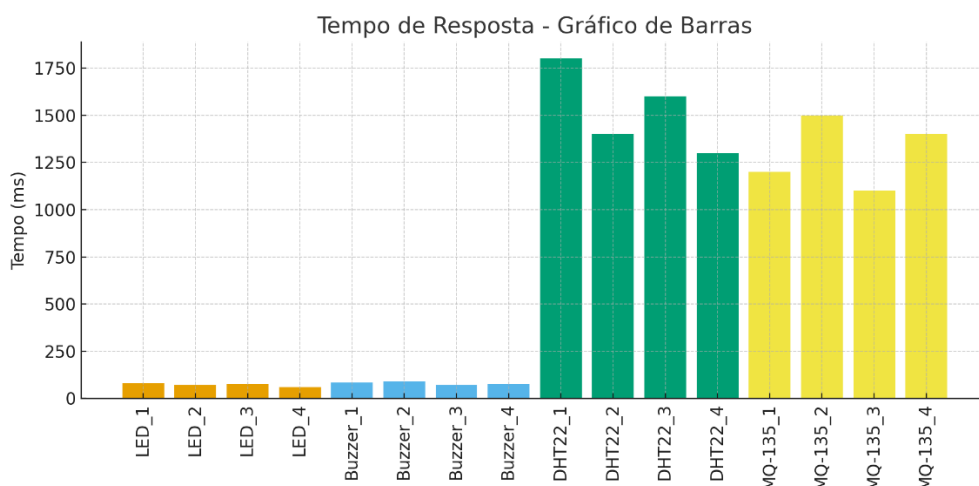
O sensor MQ-135 apresentou tempos inferiores ao DHT22, o que se justifica por ser um sensor analógico cuja resposta depende diretamente das variações químicas presentes no ar. Ainda assim, manteve uma boa estabilidade entre as medições, indicando sensibilidade adequada para detectar a presença de gases. O tempo médio obtido demonstra que o protótipo é capaz de identificar mudanças ambientais em um período curto, suficiente para garantir a funcionalidade de alerta proposta pelo sistema.

Também se observou que todos os valores medidos foram devidamente exibidos no display LCD, reenviados ao Mosquitto e registrados no monitor serial da IDE do Arduino, confirmando a consistência da comunicação MQTT e a confiabilidade das publicações realizadas pelo ESP32. A sincronia entre leitura, transmissão e exibição reforça a robustez do sistema e evidencia a integração bem-sucedida entre hardware e software.

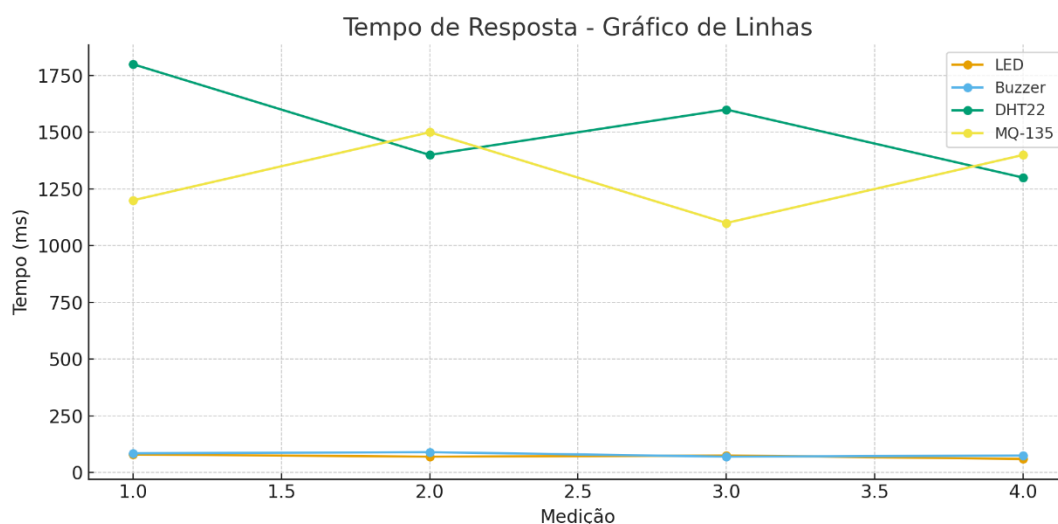
Dessa forma, a partir dos resultados obtidos, conclui-se que o protótipo apresenta desempenho adequado, baixa latência e respostas confiáveis, atendendo aos requisitos de monitoramento ambiental em tempo real. A análise dos tempos médios não apenas valida a eficiência do sistema, como também reforça a viabilidade do uso do MQTT como protocolo leve e eficaz para aplicações de IoT com necessidade de resposta rápida e contínua.

### **3.5. Gráficos dos resultados**

Os gráficos abaixo ilustram a variação dos tempos de resposta coletados, permitindo visualizar a estabilidade e a confiabilidade do sistema.



**Figura 17– Gráfico em barras com tempo de resposta dos atuadores e sensores. Fonte: Autora (2025)**



**Figura 18– Gráfico em linha com tempo de resposta dos atuadores e sensores. Fonte: Autora (2025)**

Os tempos coletados refletem o intervalo entre a ocorrência do estímulo (calor, gás ou comando) e o registro da resposta correspondente, seja ela a leitura do sensor, a publicação MQTT ou a ativação dos atuadores.

Para cada elemento, foram registradas quatro medições, cujos valores permitiram o cálculo dos tempos médios de resposta.

Os dados obtidos demonstram que os atuadores (LED e buzzer) apresentam tempos de resposta muito baixos, ficando entre 70 e 80 ms em média. Isso confirma que o ESP32 executa comandos MQTT quase imediatamente após recebê-los, garantindo alta responsividade para situações de alerta.

Os sensores apresentam tempos de resposta naturalmente maiores, devido às características de funcionamento físico de cada tipo de dispositivo:

O DHT22, por ser um sensor digital com conversão interna de temperatura e umidade, possui tempo médio de aproximadamente 1,52 s, coerente com sua especificação técnica.

Já o MQ-135, que possui um tempo de aquecimento e estabilização característico, apresentou tempo médio de 1,3 s, dentro do esperado para sensores de qualidade do ar.

Os resultados confirmam que todo o sistema funciona de forma eficiente e consistente, com baixíssima latência na comunicação MQTT e respostas precisas tanto no LCD quanto no broker MQTT e no Monitor Serial da IDE. O comportamento coerente entre sensores e atuadores evidencia que o protótipo é capaz de monitorar o ambiente em tempo real e reagir adequadamente às mudanças detectadas, atendendo aos requisitos estabelecidos para a solução.

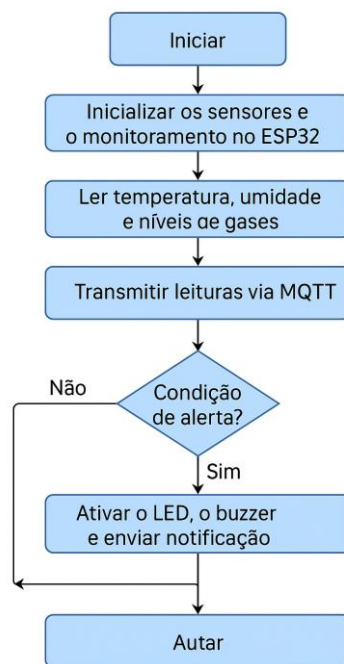
### **3.6. Fluxograma do funcionamento do protótipo**

Para a compreensão do processo de operação do protótipo, elaborou-se um fluxograma representando de forma gráfica a sequência lógica das ações realizadas pelo sistema. A construção do fluxograma é essencial, pois permite visualizar de maneira estruturada o fluxo de dados, decisões e interações entre sensores, atuadores e o protocolo de comunicação MQTT.

O fluxograma inicia-se com a fase de inicialização do dispositivo, na qual o ESP32 configura os pinos de entrada e saída, estabelece conexão com a rede Wi-Fi e tenta autenticar-se com o broker MQTT. Em seguida, o diagrama apresenta o ciclo contínuo de operação, que envolve a leitura dos sensores DHT22 e MQ-135, o processamento das informações coletadas, a atualização do display LCD e a publicação periódica dos dados nos tópicos MQTT.

O esquema também evidencia os pontos de tomada de decisão do sistema, demonstrando como o ESP32 verifica se os valores de temperatura ou de concentração de gases ultrapassam os limites estabelecidos. Caso isso ocorra, o fluxograma mostra o acionamento imediato dos atuadores (LED e buzzer), reforçando o caráter reativo do sistema de monitoramento ambiental.

A inclusão do fluxograma torna o projeto mais transparente e compreensível, pois traduz a lógica programada em uma representação visual intuitiva, contribuindo para interpretação, validação e manutenção do protótipo.



**Figura 19– Fluxograma do Projeto Hissopo. Fonte: Autora (2025)**

### 3.7. Pseudocódigo do sistema

Para complementar a descrição do funcionamento do protótipo e facilitar a compreensão de sua lógica operacional, elaborou-se um pseudocódigo representando o fluxo geral do sistema. O pseudocódigo serve como uma abstração de alto nível que descreve, de forma clara e independente da linguagem de programação, as etapas executadas pelo ESP32 durante o monitoramento ambiental e a comunicação via MQTT.

A representação apresentada a seguir inclui os principais módulos envolvidos na solução: inicialização do hardware, leitura dos sensores DHT22 e MQ-135, atualização do display LCD, publicação dos dados no broker MQTT e execução das rotinas de acionamento dos atuadores (LED e buzzer). Além disso, o pseudocódigo evidencia o ciclo contínuo de verificação do estado da conexão com o broker e o processamento das mensagens recebidas, garantindo o controle remoto dos atuadores pelos tópicos MQTT.

```

INICIAR
Conectar ao Wi-Fi
Conectar ao broker MQTT
Inscrever-se nos tópicos de atuadores

LOOP INFINITO:
    Ler temperatura e umidade do DHT22
    Ler concentração de gases do MQ-135

    Publicar leituras MQTT

    Se comando LED recebido:
        Acionar ou desligar LED
    Se comando buzzer recebido:
        Acionar ou desligar buzzer

    Exibir valores no LCD
FIM LOOP

```

**Figura 20– Psseudocódigo do Projeto Hissopo. Fonte: Autora (2025)**

### 3.8. Vídeo de demonstração do protótipo

O vídeo de apresentação, contendo o funcionamento do protótipo com comunicação MQTT, a explicação do hardware e do código e a presença e identificação da autora está disponível no seguinte link.

Link para o vídeo:< [https://www.youtube.com/watch?v=\\_zuw\\_gWDAII](https://www.youtube.com/watch?v=_zuw_gWDAII)>. Acesso em 21 nov. 2025

### 3.9. Repositório do projeto

Toda a documentação do sistema, incluindo o código-fonte completo, diagramas, instruções de uso, descrição do hardware, documentação das interfaces de comunicação e arquivos auxiliares do projeto está disponível no repositório do projeto alocado no GitHub e acessível através do link abaixo.

Link para repositório:< <https://github.com/Brendalacerdatav/Hissopo/tree/main>>. Acesso em 21 nov. 2025

## 4.Conclusões

Os resultados obtidos demonstram que os objetivos do projeto Hissopo são plenamente alcançados. O sistema desenvolvido é capaz de monitorar em tempo real as condições ambientais internas, detectando temperatura, umidade e concentração de gases, além de acionar atuadores e enviar dados via protocolo MQTT com baixa latência e elevada confiabilidade. O conjunto de sensores, atuadores e comunicação em rede funciona de forma integrada, validando a proposta de criação de um protótipo de monitoramento ambiental baseado em Internet das Coisas (IoT).

Durante o desenvolvimento, algumas dificuldades se destacam. A primeira diz respeito à necessidade de alterar completamente a arquitetura de comunicação, já que a solução inicial, baseada na plataforma Blynk, não atendia aos requisitos formais da disciplina por utilizar um protocolo proprietário. Essa mudança demandou a adoção do broker Mosquitto e a reescrita integral da lógica IoT utilizando MQTT puro, o que implicou revisão de bibliotecas, reorganização da estrutura do código e readequação do artigo científico. Apesar do desafio, essa adaptação contribuiu significativamente para o aprofundamento no funcionamento do protocolo MQTT, fortalecendo a compreensão dos mecanismos de publicação, assinatura e gerenciamento de tópicos.

Uma segunda dificuldade enfrentada relaciona-se ao hardware utilizado, especialmente à aquisição equivocada de apenas cabos macho–macho, o que inviabiliza parte das conexões diretas entre sensores e atuadores. Essa limitação torna necessária a utilização de ferro de solda ou adaptadores adicionais para viabilizar determinadas ligações, prolongando o tempo de montagem e exigindo mais cuidado durante a fase prática do projeto. A necessidade de ajustes físicos reforça a importância do planejamento adequado dos materiais e da compatibilidade entre componentes antes da prototipação.

Outros desafios incluem o ajuste do tempo de resposta do sensor MQ-135, a estabilidade da conexão Wi-Fi e a calibração dos limites críticos de disparo dos alertas. Esses obstáculos são solucionados por meio de testes experimentais, ajustes progressivos no código e validação contínua com o broker MQTT, assegurando o correto funcionamento do sistema.

Entre as principais vantagens do protótipo, destacam-se sua simplicidade, baixo custo, modularidade e compatibilidade com arquiteturas IoT contemporâneas. O ESP32, por possuir conectividade Wi-Fi integrada, reduz a necessidade de módulos adicionais, e o protocolo MQTT possibilita comunicação leve, distribuída e altamente eficiente. As limitações identificadas incluem a sensibilidade do MQ-135 às condições ambientais, seu tempo de pré-aquecimento e o uso de um broker local, que restringe o alcance da solução caso não seja adaptada para ambientes em nuvem.

Como possibilidades de evolução, sugerem-se a integração com plataformas IoT baseadas em nuvem (AWS IoT, Azure IoT Hub, Google IoT Core), implementação de dashboards visuais em Node-RED, armazenamento de dados históricos, desenvolvimento de uma carcaça personalizada via impressão 3D e ampliação do sistema para operar simultaneamente em múltiplos ambientes.

Assim, o protótipo apresenta-se como uma solução funcional, acessível e coerente com os princípios da Internet das Coisas, demonstrando potencial tanto para aplicações domésticas quanto para fins educacionais e institucionais. A experiência adquirida, apesar dos desafios, evidencia a importância da adaptação tecnológica, da resolução de problemas práticos e da consolidação de conhecimentos em hardware, software e comunicação em rede.

## 5. Referências

ADA FRUIT. *DHT Sensor Library*. Disponível em: <https://github.com/adafruit/DHT-sensor-library>. Acesso em: 20 nov. 2025.

ARDUINO. *Arduino IDE Documentation*. Disponível em: <https://www.arduino.cc/en/software>. Acesso em: 20 nov. 2025.

BLOG DA ROBÓTICA. *Como utilizar o display LCD 16x02 com módulo I2C no Arduino*. Disponível em: <https://www.blogdarobotica.com/2022/05/02/como-utilizar-o-display-lcd-16x02-com-modulo-i2c-no-arduino/>. Acesso em: 2 out. 2025.

CASA DA ROBÓTICA. *20 Resistor 220 Ohms 1/4 W 5% de Tolerância*. Disponível em: <https://www.casadarobotica.com/componentes-eletronicos/componentes/resistor/20-resistor-220-ohms-14-w-5-de-tolerancia>. Acesso em: 21 nov. 2025.

DHT22. *Datasheet do Sensor Digital de Temperatura e Umidade DHT22/AM2302*. Disponível em: <https://www.adafruit.com/product/385>. Acesso em: 20 nov. 2025.

ECLIPSE FOUNDATION. *Mosquitto MQTT Broker Documentation*. Disponível em: <https://mosquitto.org>. Acesso em: 20 nov. 2025.

ECLIPSE FOUNDATION. *MQTT Version 3.1.1 – OASIS Standard*. Disponível em: <https://mqtt.org>. Acesso em: 20 nov. 2025.

ELETROGATE. *Buzzer ativo 5V*. Disponível em: <https://www.eletrogate.com/buzzer-ativo-5v>. Acesso em: 4 out. 2025.

ELETROGATE. *Jumpers macho-macho 40 unidades de 10 cm*. Disponível em: <https://www.eletrogate.com/jumpers-macho-macho-40-unidades-de-10-cm>. Acesso em: 4 out. 2025.

ELETROGATE. *Protoboard 400 pontos*. Disponível em: <https://www.eletrogate.com/protoboard-400-pontos>. Acesso em: 4 out. 2025.

ESPRESSIF. *ESP32 Series Datasheet*. Disponível em: <https://www.espressif.com/en/products/socs/esp32>. Acesso em: 20 nov. 2025.

FRITZING. *Fritzing software download*. EDN Network, 2025. Disponível em: <https://www.edn.com/fritzing-software-download/>. Acesso em: 28 out. 2025.

GAUER, Mayara Ananda; SZYMANSKI, Mariani Sílvia Ester; PIAN, Lucas Bischof; SCHIRMER, Waldir Nagel. *A poluição do ar em ambientes internos e a síndrome dos edifícios doentes*. *Ciência & Saúde Coletiva*, v. 16, n. 8, p. 3317–3326, 2011. Disponível em: <https://doi.org/10.1590/S1413-81232011000900026>. Acesso em: 16 jan. 2024.

IBM. *O que é Análise Preditiva?*. Disponível em: <https://www.ibm.com/br-pt/think/topics/predictive-analytics>. Acesso em: 20 nov. 2025.

JOHNSON, Frank. *LiquidCrystal I2C Library Documentation*. Disponível em: [https://github.com/johnrickman/LiquidCrystal\\_I2C](https://github.com/johnrickman/LiquidCrystal_I2C). Acesso em: 20 nov. 2025.

MAKER HERO. *Como funciona o sensor de gás MQ-135*. Disponível em: <https://www.makerhero.com/blog/como-funciona-o-sensor-de-gas-mq-135/>. Acesso em: 2 out. 2025.

MAMUTE ELETRÔNICA. *Display LCD 16x2 com backlight azul e módulo serial I2C*. Disponível em: <https://www.mamuteeletronica.com.br/display-lcd-16x2-c-back-azul-e-modulo-serial-i2c-13848>. Acesso em: 2 out. 2025.

MERSEY, Robert. *Algorithms and Flowcharts: Foundations and Principles*. Nova York: TechPress, 2018.

MQ-135. *Gas Sensor MQ-135 – Technical Data*. Winsen Electronics. Disponível em: <https://www.winsen-sensor.com>. Acesso em: 20 nov. 2025.

MQTT EXPLORER. NORDQUIST, Thomas. *An all-round MQTT client that provides a structured topic overview*. Disponível em: <https://mqtt-explorer.com/>. Acesso em: 21 nov. 2025.

NAÇÕES UNIDAS NO BRASIL. *Novas diretrizes da OMS sobre qualidade do ar reduzem valores seguros para poluição*. Disponível em: <https://brasil.un.org/pt-br/145721-novas-diretrizes-da-oms-sobre-qualidade-do-ar-reduzem-valores-seguros-para-poluiçao>. Acesso em: 16 jan. 2024.

PISCALED. *Sensor de umidade e temperatura DHT22*. Disponível em: <https://www.piscaled.com.br/sensor-de-umidade-e-temperatura-dht22>. Acesso em: 2 out. 2025.

PRESSMAN, Roger; MAXIM, Bruce. *Engenharia de Software: Uma Abordagem Profissional*. 8. ed. Porto Alegre: AMGH, 2016.

PROJETO HISSOPO. *Vídeo*. YouTube. Disponível em: [https://www.youtube.com/watch?v=\\_zuw\\_gWDAII](https://www.youtube.com/watch?v=_zuw_gWDAII). Acesso em: 21 nov. 2025.

PUBSUBCLIENT. *MQTT Client Library for Arduino – PubSubClient*. Disponível em: <https://pubsubclient.knolleary.net>. Acesso em: 20 nov. 2025.

RANDOM NERD TUTORIALS. *Installing the ESP32 Board in Arduino IDE*. Disponível em: <https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>. Acesso em: 27 set. 2025.

ROBOCORE. *Jumpers Macho-Fêmea x40 Unidades*. Disponível em: <https://www.robocore.net/cabo/jumpers-macho-femea-x40-unidades>. Acesso em: 21 nov. 2025.

ROSA, Cláudia Marisa; SOUZA, Paulo Augusto Ramalho de; SILVA, Joaquim Manoel da. *Inovação em saúde e internet das coisas (IoT)*. Perspectivas em Ciência da Informação, v. 25, n. 3, p. 119–140, 2020. Disponível em: <https://doi.org/10.1590/1981-5344/3885>. Acesso em: 16 jan. 2024.

SOMMERVILLE, Ian. *Engenharia de Software*. 10. ed. São Paulo: Pearson, 2019.

TANENBAUM, Andrew. *Redes de Computadores*. 5. ed. São Paulo: Pearson, 2011.

TETRACOMP. *LED difuso 5mm 600mcd vermelho*. Disponível em: <https://www.tetracomp.com.br/led-difuso-5mm-600mdc-vermelho--emb-100-pcs>. Acesso em: 4 out. 2025.

VICTOR VISION. *Placa ESP32: entenda o que é e como funciona*. Disponível em: <https://victorvision.com.br/blog/placa-esp32/>. Acesso em: 2 out. 2025.

WIKIPÉDIA. *ESP32*. Disponível em: <https://pt.wikipedia.org/wiki/ESP32>. Acesso em: 27 set. 2025.