INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DE SANTA CATARINA

CAMPUS SÃO JOSÉ

DANIEL VALDELEY MARQUES

APLICATIVO MÓVEL INTEGRADO A SISTEMA DE MONITORAMENTO DA QUALIDADE DO AR EM AMBIENTES INTERNOS

SÃO JOSÉ

2026

Daniel Valdeley Marques

Aplicativo Móvel Integrado a Sistema de Monitoramento da Qualidade do Ar em Ambientes Internos

Monografia apresentada ao Curso de Engenharia de Telecomunicações do Instituto Federal de Santa Catarina, para a obtenção do título de bacharel em Engenharia de Telecomunicações.

Área de concentração: Telecomunicações

Orientador: Prof. Adilson Jair Cardoso, Dr. Eng.

São José

2026

Daniel Valdeley Marques

Aplicativo Móvel Integrado a Sistema de Monitoramento da Qualidade do Ar em Ambientes Internos

Monografia apresentada ao Curso de Engenharia de Telecomunicações do Instituto Federal de Santa Catarina, para a obtenção do título de bacharel em Engenharia de Telecomunicações.

São José, 13 de fevereiro de 2026.

gov.br
Documento assinado digitalmente
ADILSON JAIR CARDOSO
Data: 11/03/2026 15:18:18-0300
Verifique em https://validar.iti.gov.br

Prof. Adilson Jair Cardoso, Dr. Eng.
Instituto Federal de Santa Catarina

gov.br
Documento assinado digitalmente
CARLOS BOABAID NETO
Data: 11/03/2026 16:40:26-0300
Verifique em https://validar.iti.gov.br

Prof. Carlos Boabaid Neto, Dr. Eng.
Instituto Federal de Santa Catarina

gov.br
Documento assinado digitalmente
MARCELO LUIZ PEREIRA
Data: 12/03/2026 16:34:24-0300
Verifique em https://validar.iti.gov.br

Prof. Marcelo Luiz Pereira, Dr. Eng.
Instituto Federal de Santa Catarina

À minha esposa, aos meus filhos, aos meus pais e familiares, pelo amor, incentivo, oração e apoio incondicional ao longo de toda esta jornada.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me conceder saúde, força e sabedoria para enfrentar os desafios desta jornada acadêmica.

Agradeço especialmente a minha esposa, os meus filhos, os meus pais, os meus irmãos, minha avó e familiares que sempre estiveram ao meu lado, me apoiando e incentivando a seguir em frente. Sem o amor e a oração de vocês, eu não teria conseguido chegar até aqui.

Agradeço também ao meu colega e amigo de curso Suyan, que compartilhou comigo momentos de aprendizado e crescimento.

Agradeço ao meu orientador, Professor Adilson, pela orientação e apoio durante todo o processo de elaboração deste trabalho. Sua experiência e conhecimento foram fundamentais para o meu aprendizado e crescimento acadêmico.

Agradeço ao coordenador do curso, Professor Carlyle, pela gestão eficiente, pela compreensão e pelo suporte dedicado nessa fase final do curso.

Agradeço ao Instituto Federal de Santa Catarina, que me proporcionou uma formação sólida e de qualidade, e a todos os professores que contribuíram para o meu aprendizado.

“Adiquire a sabedoria, adquire a inteligência; e não te esqueças nem te apartes das palavras da minha boca. Não a abandones e ela te guardará; ama-a, e ela te protegerá.” (Bíblia Sagrada, Provérbios 4:5-6)

RESUMO

A grande maioria das pessoas passa a maior parte das horas do dia dentro de ambientes internos. Assim, a qualidade do ar respirado nestes ambientes é um fator crucial para a saúde e o bem-estar das pessoas. Porém, grande parte dos contaminantes físicos, químicos e biológicos presentes no ar são furtivos, ou seja, não são detectados pelo ser humano. Desta forma, monitorar a qualidade do ar nesses locais é essencial, para identificar e mitigar riscos, reduzindo ou eliminando possíveis efeitos negativos sobre a saúde dos ocupantes, garantindo assim um ambiente mais seguro e confortável. Isto é especialmente importante em ambientes onde a qualidade do ar pode flutuar rapidamente, como em salas de aula, escritórios e outros espaços internos. O presente trabalho apresenta o desenvolvimento de uma aplicação que forneceu informações em tempo real sobre a qualidade do ar em ambientes internos. Para isso, o aplicativo foi integrado a um sistema de medição, composto por microcontrolador, sensores e dispositivos de comunicação sem fio. O aplicativo foi capaz de exibir dados como temperatura, umidade relativa do ar, e concentrações de gás carbônico, de material particulado e de compostos orgânicos voláteis, monitorando continuamente a qualidade do ar, fornecendo dados precisos que ajudaram os usuários a compreender os riscos e adotar medidas para um ambiente mais saudável e seguro.

Palavras-chave: Qualidade do Ar; Ambiente interno; Contaminação do ar; Monitoramento; Internet das coisas.

ABSTRACT

Most people spend the majority of their day indoors. Therefore, the quality of the air in these environments is crucial for health and well-being. However, many physical, chemical, and biological contaminants present in the air are stealthy and cannot be detected by humans. Thus, monitoring air quality in these places is essential to identify and mitigate risks, reducing or eliminating possible negative effects on the health of occupants and ensuring a safer and more comfortable environment. This is especially important in environments where air quality can fluctuate rapidly, such as classrooms, offices, and other enclosed spaces. This work presents the development of an application that provided real-time information about indoor air quality. For this purpose, the application was integrated with a measurement system composed of a microcontroller, sensors, and wireless communication devices. The application was able to display data such as temperature, relative humidity, and concentrations of carbon dioxide, particulate matter, and volatile organic compounds, continuously monitoring air quality and providing accurate data to help users understand risks and take measures for a healthier and safer environment.

Keywords: Air Quality; Indoor Environment; Air Contamination; Monitoring; Internet of Things.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

ACL *Access Control List*.

ANVISA Agência Nacional de Vigilância Sanitária.

CO_2 Dióxido de Carbono.

COV Composto Orgânico Volátil.

ESP32 *Espressif Systems 32-bit*.

HTTP *Hypertext Transfer Protocol*.

I^2C *Inter-Integrated Circuit*.

IARC *International Agency for Research on Cancer*.

IFSC Instituto Federal de Santa Catarina.

IoT *Internet of Things*.

JSON *JavaScript Object Notation*.

LCD *Liquid Crystal Display*.

LED *Light-Emitting Diode*.

MQTT *Message Queuing Telemetry Transport*.

NDIR *Non-Dispersive Infrared*.

NO_x Óxidos de Nitrogênio.

O_2 Oxigênio.

PM Material Particulado.

QAI Qualidade do Ar Interno.

RTC *Real-Time Clock*.

SO Sistema Operacional.

SSL *Secure Sockets Layer*.

TCP *Transmission Control Protocol*.

TLS *Transport Layer Security*.

TVOC *Total Volatile Organic Compounds.*

UART *Universal Asynchronous Receiver-Transmitter.*

SUMÁRIO

1 INTRODUÇÃO

Pessoas passam a maior parte do tempo em ambientes internos, por uma combinação de fatores sociais, econômicos e tecnológicos. O uso de edificações foi motivado pela necessidade de controlar o ambiente visando conforto e proteção contra os rigores do clima natural, como temperaturas extremas, vento e chuvas. A imensa maioria das atividades diárias modernas acontece em ambientes internos, em escritórios, fábricas, estabelecimentos comerciais, e com o aumento do trabalho remoto, também em casa. Com o crescimento das cidades, os espaços ao ar livre são mais limitados, e muitas atividades diárias, como compras e lazer, acontecem dentro de estabelecimentos cobertos. A ascensão da tecnologia trouxe alternativas de lazer, como televisão, videogames, redes sociais e outras formas de entretenimento digital, que reduzem o tempo que as pessoas passam ao ar livre. Tudo isto significa que uma grande parte do dia útil é passado dentro de ambientes internos.

O ar que respiramos dentro de casa ou em outros recintos pode estar repleto de partículas invisíveis a olho nu, que afetam a saúde sem serem percebidos. Dentre os principais poluentes, pode-se citar: (I) gases tóxicos: monóxido de carbono e dióxido de nitrogênio liberados por fogões, aquecedores e sistemas de combustão; (II) partículas finas e poeira, originadas de carpetes, tecidos, fumaça de cigarro e poluentes externos; (III) Compostos Orgânicos Voláteis (COVs), provenientes de móveis, tintas, produtos de limpeza e até materiais de construção; (IV) mofo e fungos, que crescem em ambientes úmidos, podendo causar problemas respiratórios e alergias. Além disso, o próprio Dióxido de Carbono (CO_2), embora seja um produto natural resultado da própria respiração humana e de outros organismos, também pode afetar a saúde quando sua concentração se torna excessiva.

Estes poluentes podem ter um severo impacto na saúde e bem-estar dos seres humanos, a saber: (I) doenças respiratórias: vários poluentes podem agravar asma, bronquite e outras condições pulmonares; (II) fadiga e baixa produtividade: ambientes insalubres podem causar cansaço e dificuldades de concentração; (III) problemas na qualidade do sono: exposição prolongada a poluentes pode afetar padrões de sono e gerar desconforto. Estudos mostram que o ar interno pode ser até cinco vezes mais poluído do que o ar externo, o que torna ainda mais crucial a monitoração da qualidade do ar em ambientes internos (UNITED STATES ENVIRONMENTAL PROTECTION AGENCY, 2023).

Além dos impactos diretos na saúde, a má qualidade do ar pode afetar a produtividade e o desempenho cognitivo, levando a um aumento no absenteísmo devido às doenças, e uma redução na eficiência no trabalho e nos estudos.

Ambientes internos frequentemente apresentam ventilação (renovação de ar) insuficiente, resultando em acúmulo de poluentes como o CO_2, COVs e partículas em suspensão. Por sua vez, a ventilação deficiente pode contribuir para aumentar consideravelmente o risco de transmissão de doenças infecciosas (QIAN; ZHENG, 2018). A medição de poluentes em um ambiente interno é um indicador claro da deficiência de ventilação deste ambiente.

Desta forma, a falta de um sistema eficiente para monitorar a qualidade do ar agrava esses problemas, afetando negativamente a saúde e o bem-estar das pessoas. A Resolução RE nº 9, de 16 de janeiro de 2003 (ANVISA, 2003), determina que a concentração de CO_2 em ambientes climatizados de uso coletivo não deve ultrapassar 1.000 ppm, e os aerodispersóides também não devem ultrapassar 80 $\mu g/m^3$. Estes limites são estabelecidos para garantir a saúde e o conforto dos ocupantes, evitando problemas respiratórios e outras complicações associadas à má qualidade do ar.

Apesar disto, na imensa maioria dos ambientes climatizados, não há um monitoramento destas grandezas, em tempo real.

Considerando os desafios para monitorar os níveis de qualidade do ar em ambientes internos, bem como a diversidade de poluentes existentes, este trabalho propôs a implementação de um aplicativo integrado a um sistema de medição da qualidade do ar. O aplicativo foi capaz de fornecer dados relevantes sobre a qualidade do ar interno, possibilitando fácil acesso a informações. Dessa forma, permitiu a interoperabilidade e o acesso direto às informações por meio de dispositivos móveis pessoais.

Diante do exposto, o problema de pesquisa que norteia este trabalho é formulado da seguinte maneira: desenvolver um aplicativo móvel que se integre a um sistema de monitoramento de Qualidade do Ar Interno (QAI) para fornecer, de forma clara e em tempo real, informações sobre a qualidade do ar a fim de promover a conscientização e a segurança dos ocupantes de um ambiente interno. A busca por uma resposta a esta questão se desdobra nos objetivos apresentados a seguir.

1.1 OBJETIVOS

1.1.1 Objetivo geral

Desenvolver um aplicativo acessível para monitoramento da qualidade do ar em ambientes internos, integrando-o a um sistema de medição, com o propósito de fornecer dados em tempo real sobre temperatura, umidade, concentrações de CO_2, COVs e Material Particulado (PM) aos ocupantes de uma sala de aula do Instituto Federal de Santa Catarina (IFSC) - Campus São José.

Diante disso, a relevância e justificativa deste trabalho se dão na promoção de ambientes mais saudáveis por meio da conscientização dos indivíduos. A intenção é que

esta facilidade de acesso à informação contribua para distinguir os fatores que levam à exposição a poluentes, visando o bem-estar e o desempenho dos ocupantes. Portanto, a proposta geral do aplicativo é construir um canal direto de monitoramento, com o propósito de aproximar os indivíduos das condições do ambiente que frequentam.

O propósito é aplicar a tecnologia para fortalecer a conscientização sobre saúde ambiental, favorecendo a disseminação de dados cruciais sobre as condições do ar que são frequentemente ignorados. A relação que os indivíduos estabelecem com seu ambiente ganha importância quando se considera os impactos da QAI na saúde e na produtividade. Nesse sentido, a pesquisa buscou demonstrar como a apresentação clara e acessível desses parâmetros pode estimular uma interação mais consciente com o ambiente, incentivando decisões simples e imediatas, como a ventilação do espaço. Assim, a tecnologia proposta contribuiu para um entendimento mais intuitivo da dinâmica da qualidade do ar em ambientes internos, ampliando o potencial de uso do sistema em diferentes contextos educacionais.

Para atingir esse propósito, a interface do aplicativo priorizou clareza e simplicidade na apresentação dos dados, destacando apenas as informações essenciais para a tomada de decisão. Além dos valores numéricos, foram exibidos indicadores visuais, como cores ou ícones, que sinalizaram de forma rápida se os parâmetros estavam dentro dos limites recomendados por normas vigentes. Dessa forma, o usuário pôde interpretar facilmente a situação da qualidade do ar e agir de maneira informada para favorecer ambientes internos mais saudáveis.

As implicações decorrentes do uso da ferramenta incluem o incentivo à conscientização sobre a importância da ventilação adequada e da QAI. O aplicativo funcionou como uma interface de visualização objetiva, permitindo ao usuário consultar dados essenciais, como concentração de CO_2, níveis de PM, COVs, temperatura e umidade relativa do ar. A plataforma buscou ser um meio simples e intuitivo de acesso à informação, considerando que a compreensão sobre a qualidade do ar é fundamental para decisões que impactam diretamente a saúde.

Ressalta-se, contudo, que o aplicativo não é, por si só, a solução para a má qualidade do ar, mas sim uma ferramenta para diagnosticar e conscientizar. Sua efetividade dependeu da ação dos usuários e gestores do ambiente em resposta aos dados apresentados, e da manutenção contínua do sistema de *hardware* que coleta as informações.

1.1.2 Objetivos específicos

Os objetivos específicos deste trabalho foram:

- Identificar a melhor forma de traduzir dados técnicos complexos de sensores em informações visuais compreensíveis para ocupantes das salas de aula;

- Implementar uma interface que se adapte ao sistema de monitoramento já desenvolvido no IFSC – Campus São José, em parceria com a área técnica de Refrigeração e Climatização;
- Testar a viabilidade de uma aplicação com esta finalidade, servindo como uma prova de conceito para caracterizar sua aplicabilidade em outros ambientes monitorados.

2 REVISÃO BIBLIOGRÁFICA

2.1 QUALIDADE DO AR INTERNO E SEUS IMPACTOS

A preocupação com a QAI tem ganhado destaque à medida que se reconhece que os indivíduos passam a maior parte do seu tempo em ambientes internos. A qualidade do ar nesses locais é um fator crucial para a saúde, pois o ar interno pode ser significativamente mais poluído que o externo. A poluição do ar interno tradicionalmente recebeu menos atenção do que a poluição externa, apesar de os níveis de poluentes internos serem tipicamente o dobro, e de as pessoas passarem de 80 a 90% de suas vidas em edifícios cada vez mais herméticos (GONZALEZ-MARTÍN et al., 2021). A má qualidade do ar está associada a uma série de problemas de saúde, incluindo doenças respiratórias, alergias e até câncer, além de impactar negativamente a produtividade e o desempenho cognitivo.

Os poluentes de maior relevância incluem o PM e os COVs. O PM, especialmente a fração PM2,5, representa uma preocupação significativa devido à sua capacidade de penetrar profundamente no sistema respiratório, alcançando os alvéolos pulmonares e podendo desencadear efeitos adversos à saúde.

Entre os COVs, destaca-se o formaldeído (CH_2O), em razão de sua ampla utilização em materiais de construção e mobiliário, além de seu reconhecido potencial carcinogênico. Esse composto químico é empregado na fabricação de resinas, colas, tintas, produtos de papel, cosméticos, equipamentos eletrônicos, agentes de limpeza e tecidos. O formaldeído também está presente em materiais de construção, como espumas isolantes e produtos à base de madeira utilizados em pisos e móveis. No entanto, as emissões desses materiais (como compensado, aglomerado ou MDF) geralmente decaem ao longo de algumas semanas. O formaldeído também é produzido pela oxidação de outros COVs em presença de ozônio ou radiação, bem como pela reação incompleta de hidrocarbonetos (GONZALEZ-MARTÍN et al., 2021).

Neste complexo cenário de múltiplos poluentes, o CO_2 é frequentemente empregado como um indicador primário para a avaliação da taxa de renovação do ar, visto que sua concentração está diretamente ligada à ocupação e ao metabolismo humano. No Brasil, a Resolução RE n° 9 da Agência Nacional de Vigilância Sanitária (ANVISA) estabelece um valor de referência para o controle deste indicador em ambientes coletivos, fixando uma concentração máxima de 1.000 ppm (ANVISA, 2003).

2.2 POLUENTES DO AR INTERNO

Os principais poluentes presentes em ambientes internos podem ser classificados em três grandes grupos: dióxido de carbono (CO_2), material particulado (PM) e compostos orgânicos voláteis (COVs).

2.2.1 Dióxido de carbono (CO_2)

O CO_2 é um gás incolor e inodoro, produzido naturalmente durante o processo de respiração humana, no qual o Oxigênio (O_2) é inalado e o CO_2 é exalado como subproduto do metabolismo celular. Em ambientes internos ocupados, a concentração de CO_2 tende a aumentar proporcionalmente ao número de ocupantes e à taxa metabólica.

Embora o CO_2 em si não seja considerado um poluente tóxico em concentrações típicas de ambientes internos, ele é frequentemente empregado como um indicador primário para a avaliação da taxa de ventilação e renovação do ar. Concentrações elevadas de CO_2 indicam ventilação insuficiente, o que pode estar associado ao acúmulo de outros poluentes mais nocivos.

A exposição a concentrações elevadas de CO_2 pode produzir diversos efeitos adversos à saúde, incluindo dores de cabeça, tontura, inquietação, sensação de formigamento, dificuldade respiratória, sudorese, fadiga, aumento da frequência cardíaca e elevação da pressão arterial. Em casos de exposição extrema, podem ocorrer convulsões, coma e asfixia (WISCONSIN DEPARTMENT OF HEALTH SERVICES, 2025).

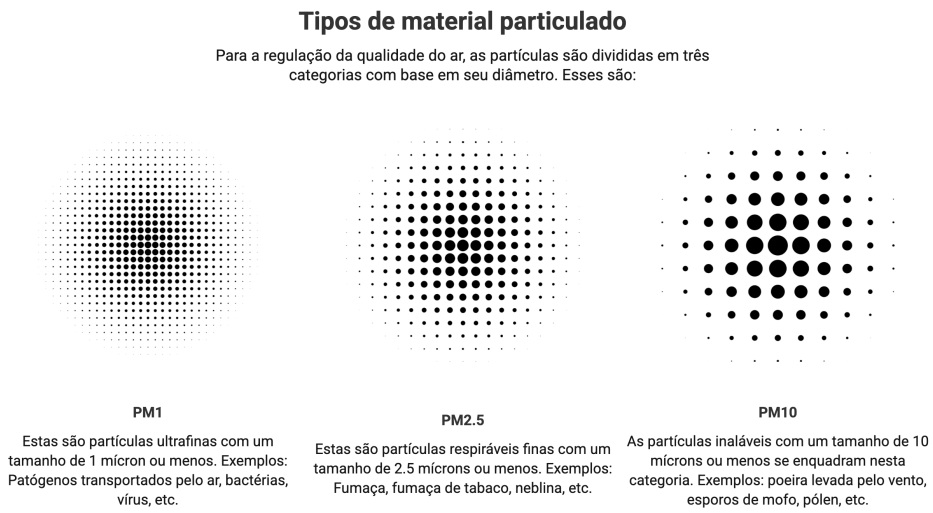
Quando as concentrações situam-se entre 1.000 e 2.000 ppm, recomenda-se aumentar a taxa de renovação do ar fresco; acima de 2.000 ppm, a situação pode ser considerada grave e demandar modificações no sistema de climatização. No Brasil, a Resolução RE n° 9 da ANVISA estabelece uma concentração máxima recomendável de 1.000 ppm para ambientes climatizados de uso coletivo (ANVISA, 2003).

2.2.2 Material particulado (PM)

O PM consiste numa mistura de partículas líquidas e sólidas que ficam suspensas no ar. Essas partículas podem variar de partículas microscópicas a partículas como fumaça, fuligem e poeira que podem ser vistas a olho nu. O PM é classificado em três categorias, com base em seu tamanho: PM10 (partículas grossas, com diâmetro inferior a 10 micrômetros), PM2,5 (partículas finas, com diâmetro inferior a 2,5 micrômetros) e PM1 (partículas ultrafinas, com diâmetro inferior a 1 micrômetro) (PRANAIR, 2025).

A Figura 1 ilustra os diferentes tipos de material particulado e suas respectivas dimensões em micrômetros (µm).

Figura 1 – Tipos de material particulado

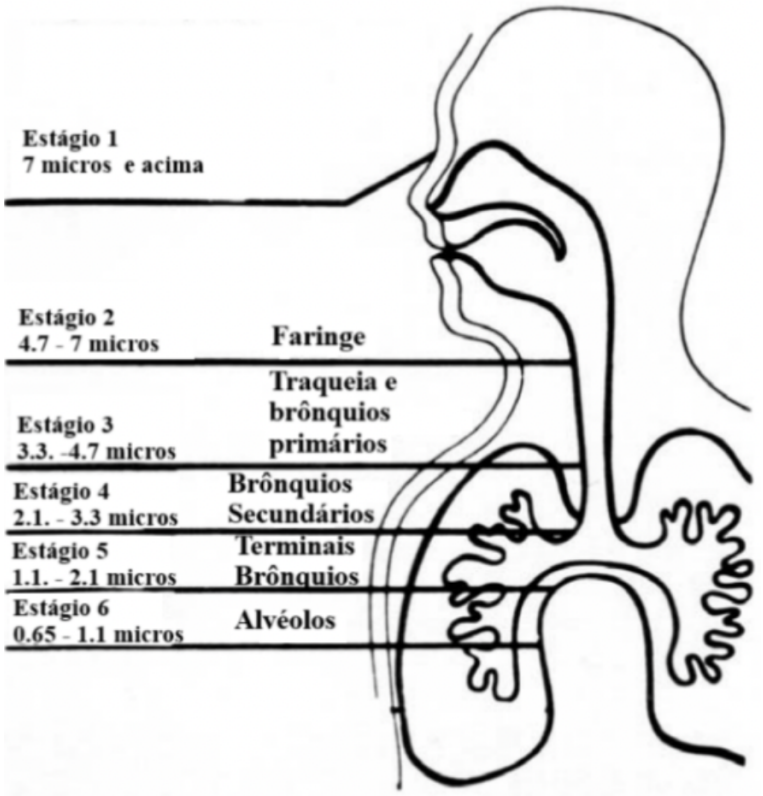


Fonte: PranAir (2025)

A deposição de partículas inaláveis no sistema respiratório humano está diretamente relacionada ao seu diâmetro aerodinâmico. Partículas maiores tendem a ser retidas nas vias aéreas superiores, enquanto partículas menores, especialmente o PM2,5, podem penetrar profundamente no sistema respiratório, alcançando os alvéolos pulmonares e podendo desencadear efeitos adversos à saúde, como doenças respiratórias e cardiovasculares.

Partículas com diâmetro superior a 7 µm tendem a ser retidas nas vias aéreas superiores, especialmente na cavidade nasal, devido aos mecanismos naturais de filtração. Na faixa de 4,7 a 7 µm, as partículas podem alcançar a faringe, enquanto partículas entre 3,3 e 4,7 µm atingem a traqueia e os brônquios primários. À medida que o tamanho das partículas diminui, aumenta sua capacidade de penetração no sistema respiratório. Partículas com diâmetro inferior a aproximadamente 1,1 µm são capazes de atingir os alvéolos pulmonares, conforme ilustrado na Figura 2.

Figura 2 – Penetração de partículas inaláveis no sistema respiratório humano



Fonte: **Energetica Ind. e Com. Ltda. https://encurtador.com.br/ysisn**

A deposição de partículas finas e ultrafinas nos alvéolos é especialmente preocupante, pois essa região é responsável pelas trocas gasosas e possui mecanismos limitados de remoção de contaminantes. Dessa forma, partículas como o PM2,5 e PM1,0 podem permanecer por períodos prolongados no organismo, estando associadas ao desenvolvimento ou agravamento de doenças respiratórias e cardiovasculares. A ANVISA recomenda que a soma de aerodispersóides totais no ar não ultrapasse 80 µg/m³ (ANVISA, 2003).

2.2.3 Compostos orgânicos voláteis (COVs)

Os COVs são uma ampla categoria de compostos orgânicos transportados pelo ar, que contêm carbono e hidrogênio, evaporam e se dispersam facilmente à temperatura ambiente. São emitidos por uma ampla gama de materiais de construção, tintas, móveis e produtos de consumo diário (PRANAIR, 2025).

A Figura 3 ilustra as principais fontes de COVs em ambientes internos, destacando materiais de construção, móveis, produtos de limpeza e atividades diárias como contribuintes significativos para a presença desses compostos no ar.

Figura 3 – Fontes primárias de COVs

Fonte: PranAir (2025)

O conjunto de compostos voláteis presentes em um ambiente caracteriza o indicador *Total Volatile Organic Compounds* (TVOC). Embora a medição do TVOC seja um bom parâmetro de triagem, não é adequado para avaliação de riscos à saúde, pois soma compostos de toxicidades muito diferentes (SALTHAMMER, 2022).

Estudo focado em campus universitário (JIN et al., 2023) demonstrou que, em ambientes educacionais, o risco não se distribui uniformemente entre os COVs. Foi constatado que a acroleína apresentou o maior risco não carcinogênico em todos os locais, incluindo salas de aula, sendo suas fontes associadas a móveis de madeira e atividades de cozinha. Além disso, benzeno e formaldeído (CH_2O) são consistentemente citados na literatura como poluentes de alto risco em ambientes internos.

O formaldeído, especificamente, é um dos compostos que exige atenção especial no monitoramento da qualidade do ar interno. A *International Agency for Research on*

Cancer (IARC) o classifica como um carcinógeno (INTERNATIONAL AGENCY FOR RESEARCH ON CANCER, 2006). Por ser utilizado em inúmeros processos de fabricação, é comum que sua concentração em ambientes internos seja substancialmente maior do que no ar externo. As principais fontes incluem produtos de madeira prensada (como pisos e painéis de partículas), carpetes, tintas, adesivos e agentes de limpeza, que podem levar de horas a meses para liberar o excesso de formaldeído retido (SAUERMANN, 2024).

2.3 MÉTODOS DE MEDIÇÃO DOS POLUENTES DO AR

2.3.1 Medição de Dióxido de Carbono (CO_2)

Para a medição de CO_2, a tecnologia mais confiável e precisa é a de *Non-Dispersive Infrared* (NDIR) (SENSIRION AG, 2020). O princípio de funcionamento do NDIR baseia-se na absorção de radiação infravermelha em comprimentos de onda específicos por diferentes moléculas de gás. Cada tipo de gás possui um espectro de absorção característico; no caso do CO_2, a absorção ocorre predominantemente na faixa de 4,26 µm.

Um sensor NDIR é composto por uma fonte de luz infravermelha, uma câmara de amostra por onde o ar passa, e um detector com filtros ópticos. Quando a radiação infravermelha atravessa a câmara contendo a amostra de ar, as moléculas de CO_2 absorvem parte dessa radiação no comprimento de onda específico. A quantidade de luz que chega ao detector é inversamente proporcional à concentração do gás-alvo, permitindo um cálculo preciso da sua presença com base na Lei de Beer-Lambert.

Em aplicações de monitoramento contínuo, sensores NDIR frequentemente incorporam mecanismos de autocalibração, ajustando periodicamente o ponto zero com base na menor concentração de CO_2 registrada em determinado intervalo de tempo. Essa estratégia compensa derivas térmicas e envelhecimento dos componentes ópticos, contribuindo para manter a confiabilidade das medições ao longo do tempo.

2.3.2 Medição de Material Particulado (PM)

Para a medição de material particulado, sensores ópticos baseados na técnica de espalhamento de luz (*light scattering*) são amplamente utilizados devido à sua capacidade de fornecer medições rápidas e em tempo real.

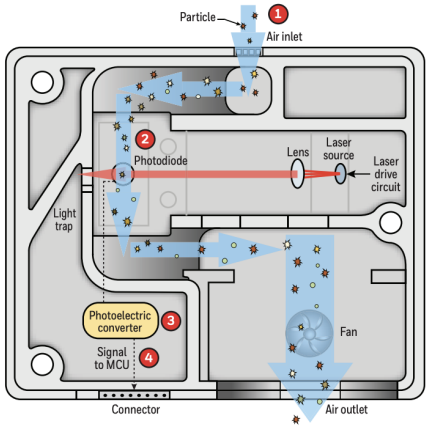
O princípio de funcionamento baseia-se na interação entre partículas em suspensão e um feixe de luz laser. Quando as partículas atravessam a câmara de medição do sensor, elas interceptam o feixe de luz e provocam seu espalhamento em múltiplas direções. Um fotodetector posicionado em ângulo específico capta a luz dispersada, e a intensidade do sinal captado é proporcional à quantidade e ao tamanho das partículas presentes.

A partir da análise do padrão de espalhamento, algoritmos de processamento calculam a concentração de partículas em diferentes faixas de tamanho (PM1, PM2,5, PM4 e PM10). É importante ressaltar que sensores baseados em espalhamento de luz fornecem estimativas de concentração de massa derivadas de contagem e distribuição de tamanho de partículas, utilizando aproximações baseadas em densidades típicas de aerossóis. Essa abordagem difere dos métodos gravimétricos de referência, que medem diretamente a massa depositada em filtros.

A Figura 4 ilustra o princípio de funcionamento de sensores ópticos de material

particulado.

Figura 4 – Funcionamento do sensor de material particulado



1. O ventilador puxa o ar através da entrada.
2. O ar passa pelo feixe de laser, onde a luz refletida pelas partículas é capturada pelo fotodiodo.
3. O fotodiodo envia as informações para o conversor fotoelétrico, que processa o sinal das partículas e o converte em densidade.
4. O sinal é transmitido ao microcontrolador, onde um algoritmo proprietário processa os dados e fornece a concentração de material particulado ($\mu g/m^3$).

Fonte: (HONEYWELL, 2021)

Sensores ópticos desse tipo podem apresentar derivas nas leituras ao longo do tempo devido ao acúmulo de contaminantes nos componentes ópticos e variações nas condições ambientais, especialmente umidade relativa elevada, que pode causar higroscopicidade das partículas e afetar as medições.

2.3.3 Medição de Composto Orgânico Volátils (COVs)

A medição de COVs em ambientes internos apresenta desafios particulares devido à grande diversidade de compostos existentes e suas concentrações variáveis. Diferentemente de poluentes como CO_2, que podem ser medidos diretamente em concentrações absolutas, a maioria dos sensores disponíveis no mercado para COVs fornece índices relativos que representam a qualidade geral do ar, sem identificar compostos específicos.

A tecnologia mais comum para detecção de COVs em sensores de baixo custo é a de óxido metálico semicondutor, conhecida como MOX (*Metal Oxide Semiconductor*). O princípio de funcionamento baseia-se na variação da resistência elétrica de uma camada de óxido metálico (tipicamente SnO_2, WO_3 ou In_2O_3) quando exposta a gases redutores ou oxidantes. Em condições de ar limpo, o oxigênio atmosférico adsorve na superfície do óxido, capturando elétrons livres e aumentando a resistência do material. Quando gases como COVs ou hidrogênio entram em contato com a superfície aquecida do sensor, reagem com o oxigênio adsorvido, liberando elétrons de volta para o material e reduzindo

sua resistência. Essa variação de resistência é então convertida em um sinal elétrico proporcional à concentração dos gases presentes.

2.3.3.1 Índices adimensionais de COV e TVOC

É importante compreender que a saída de sensores MOX para COVs geralmente não é expressa em unidades de concentração absoluta (como ppm ou µg/m³), mas sim em índices adimensionais. Isso ocorre porque o sensor responde de maneira não-seletiva a uma ampla gama de compostos voláteis, cada um com diferente sensibilidade e toxicidade. Assim, não é possível determinar diretamente a concentração de um composto específico sem equipamentos analíticos mais sofisticados, como cromatógrafos gasosos.

O índice de COV (VOC Index) e o índice de TVOC são calculados por algoritmos que processam as variações de resistência do elemento sensor ao longo do tempo (SENSIRION AG, 2022). Esses algoritmos tipicamente estabelecem uma **linha de base adaptativa**, que representa a condição média do ambiente ao longo de um período de referência (geralmente horas ou dias). O índice é então calculado com base no desvio instantâneo em relação a essa linha de base:

- Um índice igual a 100 representa a condição típica do ambiente (linha de base);
- Valores abaixo de 100 indicam ar mais limpo que o habitual;
- Valores acima de 100 indicam aumento na concentração de COVs em relação ao padrão histórico.

Essa abordagem relativa possui vantagens e limitações. A principal vantagem é permitir a detecção de **variações e eventos anômalos** na qualidade do ar, como a introdução de novos contaminantes ou o aumento súbito de emissões. A limitação é que um ambiente cronicamente contaminado pode apresentar índices aparentemente normais, uma vez que a linha de base se adapta às condições prevalentes.

Por essa razão, índices de COV são mais adequados para **monitoramento de tendências**, **detecção de eventos** e **sistemas de alerta**, mas não substituem análises laboratoriais quando se deseja identificar e quantificar compostos específicos, especialmente aqueles de maior risco toxicológico como formaldeído e benzeno.

2.4 DESENVOLVIMENTO DA INTERFACE E ARQUITETURA DE SOFTWARE

A implementação de uma solução de QAI eficaz não se limita ao hardware de sensoriamento; a forma como os dados são processados, transmitidos e apresentados ao usuário final é igualmente determinante para o sucesso do sistema. Esta seção detalha os conceitos e tecnologias que fundamentam o desenvolvimento da camada de software do projeto: o aplicativo móvel.

2.4.1 Abordagens para o desenvolvimento de aplicativos móveis

A escolha da plataforma de desenvolvimento é uma decisão estratégica que impacta diretamente o tempo, o custo e a manutenibilidade do projeto. Conforme a literatura da área, as abordagens se dividem em duas categorias principais: nativa e híbrida.

O desenvolvimento nativo envolve a criação de códigos-fonte separados para cada Sistema Operacional (SO), como iOS® e *Android*™, utilizando suas respectivas linguagens e ferramentas (*Java*/*Kotlin* para *Android*; *Swift*/*Objective-C* para iOS). Essa abordagem garante acesso irrestrito aos recursos do dispositivo, porém exige duplicação de esforços (MARQUES, 2024).

A abordagem híbrida permite o desenvolvimento de uma única base de código compilável para múltiplas plataformas. *Frameworks* como *React Native*™ e *Flutter*™ lideram este segmento, otimizando recursos e acelerando o ciclo de desenvolvimento.

Uma terceira alternativa são ferramentas de desenvolvimento orientadas a fluxo, como o *Node-RED*™, que permitem criar interfaces e integrações rapidamente através de programação visual. Considerando que o objetivo deste trabalho é a criação de uma interface de visualização de dados com agilidade e integração direta com fontes *Internet of Things* (IoT), o *Node-RED* foi adotado como plataforma de desenvolvimento.

2.4.2 Node-RED

e O *Node-RED* é uma ferramenta de programação visual de código aberto, originalmente desenvolvida pela IBM® e atualmente mantida pela *OpenJS Foundation*. Sua principal característica é o paradigma de programação baseada em fluxos (*flow-based programming*), onde cada nó (*node*) representa uma funcionalidade específica — como receber dados *Message Queuing Telemetry Transport* (MQTT), processar informações ou exibir gráficos. A ferramenta é construída sobre Node.js®, aproveitando seu modelo orientado a eventos para lidar com múltiplas entradas e saídas de forma eficiente (NODE-RED, 2025).

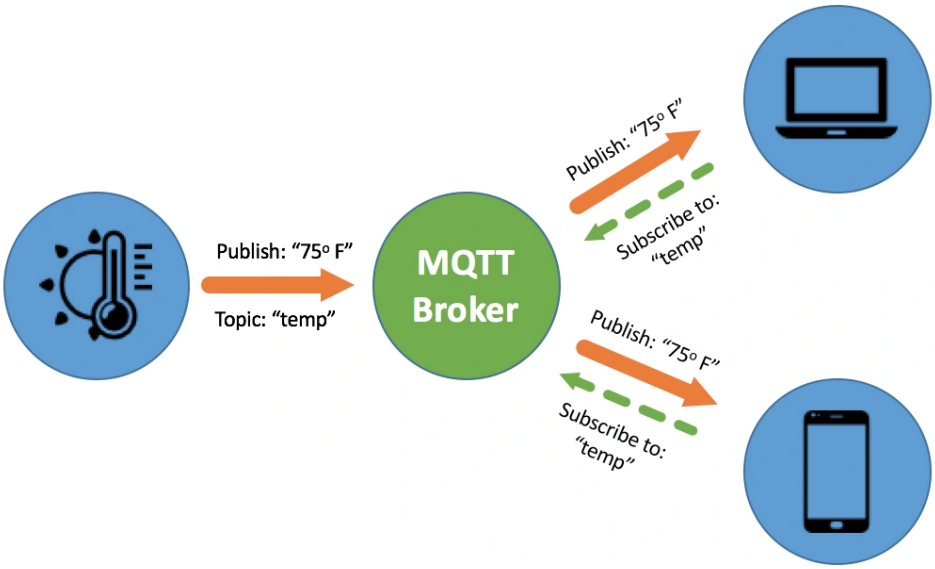
No contexto deste projeto, o *Node-RED* atua como cliente MQTT, inscrevendo-se nos tópicos do *broker* para receber os dados dos sensores em tempo real. Esses dados são processados e direcionados para nós de interface, gerando um painel de controle (*dashboard*) acessível por navegador *web*.

2.5 PROTOCOLO E ARQUITETURA PARA COMUNICAÇÃO COM IoT

No contexto de IoT, onde microcontroladores operam com restrições de energia, memória e largura de banda, protocolos tradicionais como o *Hypertext Transfer Protocol* (HTTP) tornam-se inadequados devido à sobrecarga de conexões *Transmission Control Protocol* (TCP) e cabeçalhos extensos. Como alternativa, o MQTT foi adotado neste trabalho por sua arquitetura leve baseada no modelo publicar/assinar (*publish/subscribe*), que permite o desacoplamento entre produtores e consumidores de dados.

A Figura 5 ilustra o funcionamento do MQTT: dispositivos publicam dados em tópicos específicos, enquanto aplicações consomem essas informações mediante inscrição prévia no *broker*.

Figura 5 – Exemplo de Broker MQTT - *Publish/Subscriber*



Fonte: **https://rightech.io/en/developers/faq/mqtt**

2.5.1 O protocolo MQTT

O MQTT é um protocolo de mensagens leve, executado sobre TCP/IP e estruturado no paradigma publicar/assinar. Criado no final da década de 1990 pela IBM para aplicações industriais e redes de baixa confiabilidade, consolidou-se como um dos principais padrões de comunicação para IoT (YUAN, 2021). Em 2014, sua especificação passou a ser mantida pela OASIS, estabelecendo-o como padrão aberto (HIVEMQ, 2025).

A principal vantagem do MQTT é a minimização da sobrecarga de comunicação: mensagens com cabeçalhos reduzidos, consumo otimizado de energia e tolerância a desconexões temporárias. A arquitetura envolve três elementos:

- *Publisher*: dispositivo que publica mensagens em tópicos específicos (neste projeto, o *Espressif Systems 32-bit* (ESP32));
- *Subscriber*: cliente que consome mensagens dos tópicos assinados (o *dashboard* desenvolvido);
- *Broker*: servidor que roteia mensagens entre publicadores e assinantes.

2.5.2 Eclipse Mosquitto

O *Eclipse Mosquitto* é um *broker* MQTT de código aberto desenvolvido pela *Eclipse Foundation*, selecionado para este projeto por sua leveza, robustez e ampla documentação (ECLIPSE MOSQUITTO, 2025). Suas principais características são:

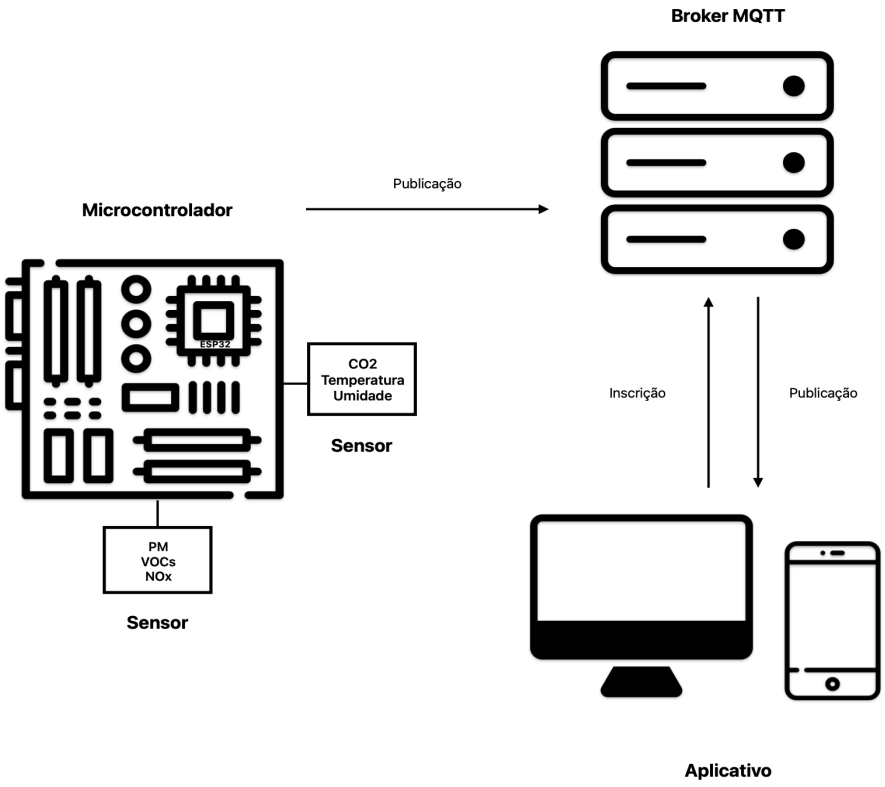
- Compatibilidade com MQTT 3.1, 3.1.1 e 5.0;
- Baixo consumo de CPU e memória;
- Suporte a autenticação, *Access Control List*s (ACLs) e criptografia *Transport Layer Security* (TLS)/*Secure Sockets Layer* (SSL);
- Disponibilidade multiplataforma (*Linux*, *Windows*, *macOS*);
- Ferramentas de linha de comando (`mosquitto_pub`, `mosquitto_sub`).

3 DESENVOLVIMENTO DO SISTEMA

Este capítulo descreve o desenvolvimento do sistema de monitoramento da qualidade do ar em ambientes internos. O projeto utilizou sensores de qualidade do ar, um microcontrolador para processamento dos dados e uma interface *web* para visualização. As seções a seguir apresentam a arquitetura do sistema desenvolvido, os componentes de *hardware* e *software* implementados, os atores da aplicação, as regras de negócio e a metodologia de apresentação das informações coletadas. O trabalho caracterizou-se como uma prova de conceito, com o objetivo de validar a viabilidade e a eficácia de uma interface *web* para um sistema de monitoramento de QAI.

3.1 ARQUITETURA DO SISTEMA

Figura 6 – Arquitetura do sistema



Fonte: Elaborado pelo autor

A eficácia de um sistema de QAI depende de uma arquitetura robusta, modular e escalável, capaz de integrar *hardware* e *software* de forma coesa e confiável. A solução desenvolvida neste trabalho estrutura-se a partir da infraestrutura de *hardware* previamente desenvolvida em colaboração com a área técnica de Refrigeração e Climatização do

IFSC-SJ, composta por dispositivos embarcados responsáveis pela medição de variáveis ambientais como concentração de gases, PM, temperatura e umidade relativa.

Os dispositivos possuem capacidade de pré-processamento local, permitindo a filtragem, validação e agregação de dados ainda no microcontrolador, antes da transmissão via protocolo MQTT. Essa estratégia reduziu ruídos, eliminou leituras inválidas e melhorou a estabilidade dos valores reportados.

Conforme ilustrado na Figura 6, a arquitetura foi organizada em quatro camadas principais:

3.1.1 Camada de sensoriamento e firmware embarcado

Essa camada corresponde ao módulo físico instalado no ambiente monitorado e é responsável pela obtenção das variáveis ambientais. O protótipo integra os sensores ambientais ao microcontrolador ESP32, que realiza a coleta periódica das grandezas de CO_2, temperatura, umidade, PM, COVs e Óxidos de Nitrogênio (NO_x).

O *firmware* embarcado no ESP32 é responsável por gerenciar a comunicação com os sensores, realizar a leitura dos dados e enviá-los para a camada de comunicação IoT.

3.1.2 Camada de comunicação IoT

Os dados coletados e tratados pelo microcontrolador são enviados a um servidor intermediário utilizando o protocolo MQTT. O ESP32 atua como *publisher*, publicando os valores em tópicos específicos definidos no *firmware*.

O uso do MQTT assegura:

- baixo consumo de banda;
- eficiência em redes instáveis;
- comunicação assíncrona orientada a mensagens;
- compatibilidade direta com ferramentas de *backend* e visualização.

Essa camada estabelece o elo entre o dispositivo sensor e os demais componentes do sistema.

3.1.3 Camada de backend

A camada de *backend* concentra o processamento central dos dados transmitidos pelo módulo sensor.

É composta por:

- **Eclipse Mosquitto**, operando como *broker* MQTT, responsável por receber e distribuir as mensagens;
- **Node-RED**, utilizado para construir fluxos que recebem, tratam, armazenam e repassam as informações para a camada de apresentação.

O *Node-RED* executa funções como:

- processamento e transformação das mensagens recebidas;
- organização dos dados;
- preparação das informações para exibição em *dashboards*.

3.1.4 Camada de apresentação

A camada de apresentação refere-se à interface visual disponibilizada ao usuário final. Ela foi construída utilizando os *dashboards* oferecidos pelo *Node-RED*, permitindo:

- visualização clara e contínua das variáveis ambientais;
- apresentação de gráficos e indicadores em tempo real;
- acesso multiplataforma, via navegador *web*, em dispositivos móveis ou computadores.

Essa solução eliminou a necessidade de desenvolvimento de um aplicativo dedicado, mantendo a simplicidade e a eficiência do sistema.

3.2 HARDWARE DO SISTEMA

Esta seção descreve os componentes de *hardware* que compõem o sistema de monitoramento, incluindo sensores, microcontrolador e configuração física.

3.2.1 Sensores

O sistema utiliza sensores especializados para a medição dos indicadores de qualidade do ar interno:

- **SCD30 (Sensirion®)**: responsável pela medição da concentração de CO_2, utilizando tecnologia de NDIR, além de temperatura e umidade relativa do ar;
- **SEN55 (Sensirion®)**: sensor destinado à medição de PM (PM1,0, PM2,5, PM4 e PM10), utilizando tecnologia de espalhamento de luz (*light scattering*), e à obtenção de índices de COVs e NO_x, por meio de sensor de óxido metálico semicondutor (MOX).

3.2.1.1 Sensor SCD30

A Figura 7 mostra o sensor SCD30, que oferece uma faixa de medição de 400 a 10.000 ppm, com uma incerteza de medição de ±(30 ppm + 3% do valor medido). Além disso, o sensor mede temperatura e umidade com incertezas de ±0,4 °C e ±3% UR, respectivamente (SENSIRION, 2020).

Figura 7 – Sensor SCD30

Fonte: (SENSIRION, 2020)

Tabela 1 – Especificações do sensor de CO_2 do SCD30

Parâmetro	**Condições**	**Valor**
Faixa de medição de CO_2	I2C, UART, PWM	0 – 40 000 ppm / 0 – 5 000 ppm
Incerteza de medição	400 ppm – 10 000 ppm	± (30 ppm + 3%MV)
Repetibilidade	400 ppm – 10 000 ppm	± 10 ppm
Estabilidade térmica	T = 0 ... 50°C	± 2.5 ppm / °C
Tempo de resposta	$\tau_{63\%}$	20 s
Deriva ao longo da vida útil	400 ppm – 10 000 ppm	± 50 ppm

3.2.1.2 Sensor SEN55

A Figura 8 mostra o sensor SEN55, que oferece medição de material particulado e índices de COV e NOx.

Figura 8 – Sensor SEN55

Fonte: (SENSIRION, 2022)

Tabela 2 – Especificações de material particulado (PM) e gases (VOC/NOx) do sensor SEN55 (Sensirion)

Parâmetro	**Condições**	**Valor**
Faixa de concentração de massa	—	0–1000 µg/m³
Tamanhos detectados	—	0.3–1.0 µm (PM1); 0.3–2.5 µm (PM2.5); 0.3–4.0 µm (PM4); 0.3–10.0 µm (PM10)
Incerteza de medição (PM1, PM2.5)	0–100 µg/m³	±5 µg/m³ e ±5% m.v.
	100–1000 µg/m³	±10% m.v.
Incerteza de medição (PM4, PM10)	50–100 µg/m³	±25 µg/m³
	100–1000 µg/m³	±25% m.v.
Saída VOC index	—	1–500 pontos (VOC Index)
Saída NOx index	—	1–500 pontos (NOx Index)
Variação entre dispositivos (VOC)	—	$< \pm 15$ pontos ou
Variação entre dispositivos (NOx)	—	$< \pm 50$ pontos ou
Tempo de resposta $\tau_{63\%}$ (gás)	Mudança 5→10 ppm etanol	< 10 s
Tempo de resposta $\tau_{90\%}$ (gás)	—	< 30 s

3.2.2 Microcontrolador ESP32

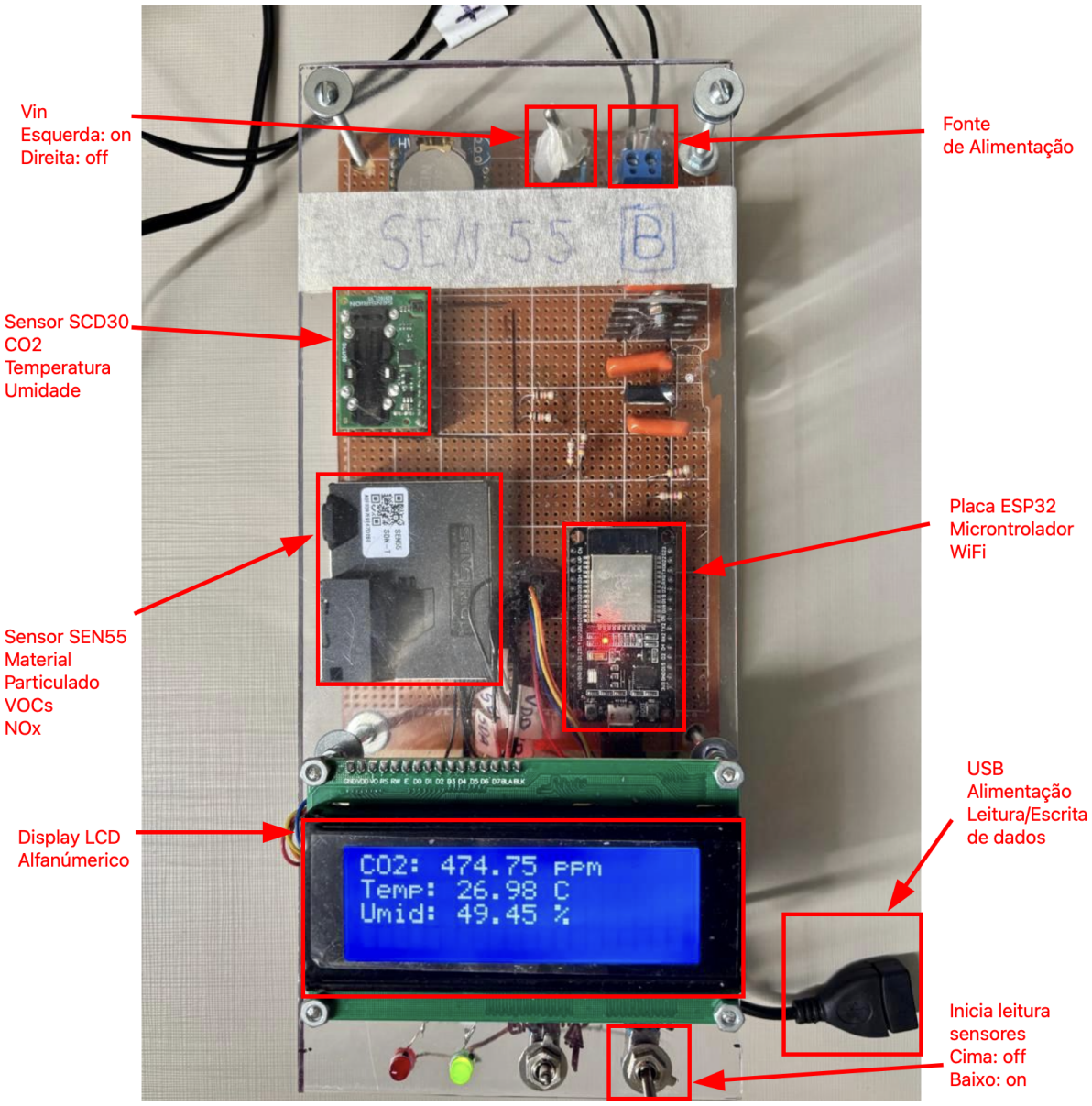
O ESP32 foi selecionado por sua conectividade *Wi-Fi*® integrada, custo reduzido e ampla disponibilidade de bibliotecas para sensores. Ele realiza:

- leitura dos sensores via *Inter-Integrated Circuit* (I^2C) e *Universal Asynchronous Receiver-Transmitter* (UART);
- execução do *firmware* embarcado;
- publicação dos dados no *broker* MQTT;
- gestão da comunicação *Wi-Fi*.

Sua eficiência energética e maturidade de plataforma tornam-no adequado para aplicações IoT estáveis.

3.2.3 Configuração física do sistema embarcado

Figura 9 – Placa do sistema embarcado com microcontrolador ESP32 e sensores



Fonte: Elaborado pelo autor

A Figura 9 apresenta a configuração física do sistema embarcado desenvolvido, montado sobre uma placa de circuito universal do tipo ilhada. Esta etapa do desenvolvimento foi realizada pela área técnica de Refrigeração e Climatização, responsável pela organização do conjunto de modo a permitir a integração dos módulos eletrônicos, garantindo estabilidade mecânica, facilidade de acesso aos componentes e adequada visualização do funcionamento do sistema.

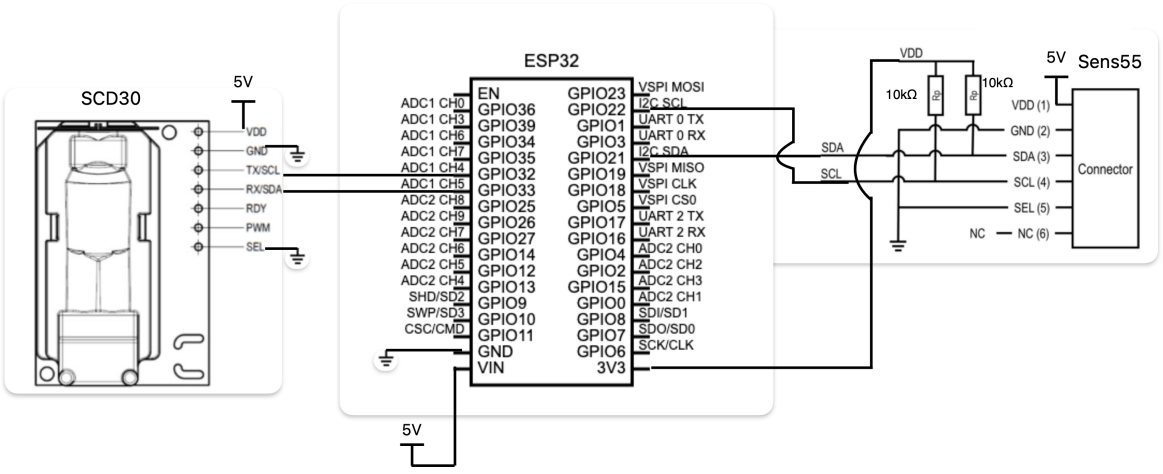
O núcleo do sistema é constituído por uma placa de desenvolvimento baseada no microcontrolador ESP32, posicionada na região central da placa. Observa-se a presença do

sensor SCD30 de CO_2, temperatura e umidade, e do sensor SEN55 destinado à medição de PM, COVs e NO_x, ambos conectados ao microcontrolador por meio de interfaces digitais.

Na porção inferior da placa, foi instalado um display *Liquid Crystal Display* (LCD) alfanumérico de 20 colunas por 4 linhas, utilizado para a apresentação local das medições. O sistema também conta com *Light-Emitting Diode*s (LEDs) indicadores e interruptores do tipo chave para controle manual. Toda a montagem encontra-se protegida por uma estrutura transparente.

O módulo de sensoriamento integra os sensores SCD30 e SEN55 ao microcontrolador ESP32 através de dois barramentos I^2C independentes: o sensor SEN55 utiliza os pinos GPIO21 (SDA) e GPIO22 (SCL), enquanto o sensor SCD30 possui barramento dedicado nos pinos GPIO33 (SDA) e GPIO32 (SCL). Os sensores são alimentados com 5V, nível exigido para seus componentes eletromecânicos e ópticos internos, enquanto as linhas de comunicação utilizam resistores de *pull-up* de 10 kΩ referenciados a 3,3V. A Figura 10 apresenta o diagrama esquemático das conexões dos sensores.

Figura 10 – Diagrama esquemático do módulo de sensoriamento



Fonte: Elaborado pelo autor

3.2.4 Desenvolvimento do firmware

O desenvolvimento do firmware constituiu etapa central do projeto, sendo responsável por coordenar a aquisição, o processamento, o armazenamento local e a transmissão remota dos dados ambientais. O firmware foi implementado na plataforma Arduino®, utilizando o microcontrolador ESP32.

A arquitetura do firmware foi organizada de forma modular, com separação lógica entre as rotinas de inicialização, leitura de sensores, apresentação de dados, arma-

zenamento em cartão SD e comunicação via rede. O sistema foi configurado com dois barramentos I^2C independentes: o primeiro dedicado ao display LCD, ao módulo *Real-Time Clock* (RTC) e ao sensor SEN55, e o segundo utilizado exclusivamente para o sensor SCD30.

Os sensores realizam leituras periódicas conforme intervalo configurável, garantindo consistência temporal dos dados. Para apresentação local, o display LCD exibe os valores medidos de maneira cíclica. Um LED indicador é acionado a cada ciclo de leitura bem-sucedido.

O controle do início e interrupção das leituras é realizado por chave física conectada ao ESP32. O módulo RTC garante a correta marcação temporal das medições. Os dados são armazenados localmente em cartão SD no formato CSV e transmitidos via MQTT em formato *JavaScript Object Notation* (JSON).

A Código 3.1 apresenta a configuração da conexão Wi-Fi e do cliente MQTT:

Código 3.1 – Setup da Rede Wi-Fi e MQTT no Firmware

```
// Rede wifi
const char* ssid = "Conforto-Termico";
const char* password = "master1466";

// Config MQTT
const char* mqtt_server = "192.168.50.20";
const int   mqtt_port = 1883;

WiFiClient espClient;
PubSubClient client(espClient);
```

O Código 3.2 apresenta o formato JSON utilizado para transmissão dos dados:

Código 3.2 – JSON resultado da leitura de Sensores

```
{
    "co2": 997.7599,
    "temperatura": 28.03616,
    "umidade": 48.61755,
    "pm1.0": 3.2,
    "pm2.5": 3.7,
    "pm4.0": 4,
    "pm10.0": 4.1,
    "vocIndex": 332,
    "noxIndex": 1
}
```

3.3 SOFTWARE DO SISTEMA

Esta seção descreve os componentes de *software* utilizados no sistema, incluindo o *broker* MQTT, a plataforma Node-RED, e os mecanismos de acesso do usuário às informações.

3.3.1 Broker MQTT

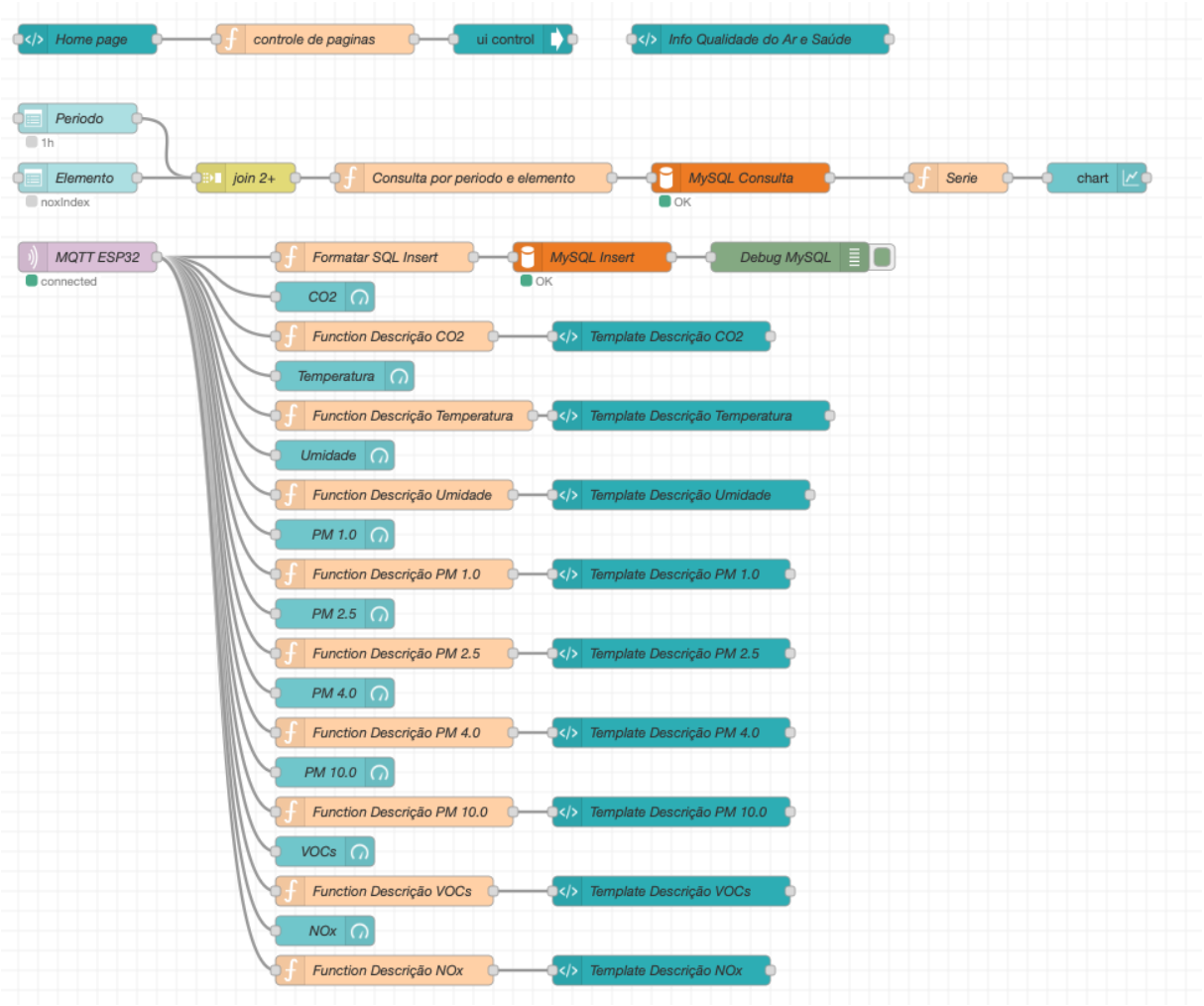
O *broker* utilizado foi o *Eclipse Mosquitto*™, por sua leveza, robustez e compatibilidade com plataformas de baixo custo. Ele gerenciou a comunicação entre o microcontrolador e os serviços de *backend* e apresentação.

3.3.2 Plataforma de supervisão Node-RED

Com o objetivo de viabilizar a supervisão do sistema embarcado, o armazenamento estruturado dos dados e sua visualização em tempo real e histórica, foi desenvolvida uma plataforma de monitoramento utilizando o ambiente Node-RED. Essa plataforma atua como camada de integração entre o sistema embarcado baseado em ESP32, o protocolo MQTT, o banco de dados relacional MySQL® e a interface gráfica apresentada ao usuário.

A Figura 11 apresenta o fluxo geral desenvolvido no Node-RED, evidenciando os blocos responsáveis pela recepção dos dados via MQTT, pela visualização em tempo real das variáveis ambientais, pelo armazenamento das medições no banco de dados e pelos mecanismos de consulta histórica.

Figura 11 – Flow Monitoramento Ambiental no Node-RED



Fonte: Elaborado pelo autor

3.3.2.1 Representação em tempo real a partir de dados MQTT

A representação em tempo real dos parâmetros ambientais é realizada a partir da recepção contínua das mensagens publicadas pelo sistema embarcado via protocolo MQTT. O ESP32 transmite periodicamente os dados dos sensores em formato JSON, contendo todas as variáveis ambientais monitoradas em uma única mensagem estruturada.

No Node-RED, um nó de subscrição MQTT é responsável por receber essas mensagens e disponibilizá-las para o restante do fluxo. Após a recepção, os dados são convertidos em objetos manipuláveis, permitindo o acesso direto a cada variável por meio de suas respectivas propriedades. Essa abordagem possibilita o processamento simultâneo e independente de cada grandeza ambiental, sem a necessidade de múltiplos tópicos ou mensagens fragmentadas.

Cada variável monitorada é encaminhada para um ramo específico do fluxo, no qual são implementados componentes de visualização do tipo *gauge*. Esses elementos permitem

a apresentação imediata dos valores medidos, possibilitando ao usuário acompanhar em tempo real as condições ambientais do ambiente monitorado.

Além da visualização numérica, foram implementadas rotinas de interpretação qualitativa por meio de nós do tipo `function`. Esses nós comparam os valores recebidos com faixas de referência estabelecidas na literatura técnica e associam cada medição a uma descrição textual relacionada à qualidade do ar e aos possíveis impactos à saúde. As descrições resultantes são exibidas no dashboard por meio de componentes `ui_template`, complementando a visualização gráfica e tornando a informação mais compreensível ao usuário final.

Essa arquitetura orientada a eventos permite que qualquer nova publicação MQTT seja imediatamente refletida no dashboard, garantindo baixa latência e atualização contínua das informações apresentadas. Os resultados obtidos com essa implementação são apresentados no Capítulo 4.

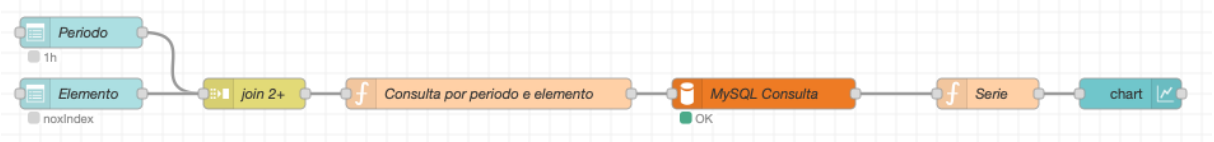
3.3.2.2 Armazenamento e consulta histórica dos dados ambientais

Paralelamente à visualização em tempo real, os dados recebidos via MQTT são armazenados de forma estruturada em um banco de dados relacional MySQL, possibilitando análises históricas e a rastreabilidade das medições realizadas. O armazenamento é realizado por meio da tabela `LeituraSensor`, projetada para contemplar todas as variáveis ambientais monitoradas e associá-las a um instante temporal específico.

No fluxo do Node-RED, um nó de função é responsável por formatar dinamicamente os comandos de inserção SQL, mapeando cada campo do objeto JSON recebido para os respectivos atributos da tabela do banco de dados. Em seguida, o nó de inserção MySQL executa o comando, efetivando o armazenamento da leitura. O campo temporal é automaticamente preenchido pelo banco de dados, garantindo consistência na marcação das medições.

A Figura 12 ilustra o fluxo específico desenvolvido para a realização de consultas históricas. Nesse fluxo, o usuário pode selecionar, por meio da interface gráfica, tanto o período de interesse quanto o parâmetro ambiental a ser analisado. Essas informações são combinadas e processadas por um nó de função, responsável pela construção dinâmica de consultas SQL do tipo `SELECT`, considerando intervalos de tempo definidos pelo usuário.

Figura 12 – Fluxo de Consulta Histórica no Node-RED



Fonte: Elaborado pelo autor

As consultas exploram um índice temporal associado ao campo de data e hora da tabela, o que contribui para a eficiência na recuperação dos dados, mesmo em bases com grande volume de registros. Os resultados obtidos são organizados em séries temporais e apresentados no dashboard por meio de gráficos, permitindo a análise da evolução dos parâmetros ambientais ao longo do tempo e a identificação de padrões ou eventos críticos.

A seguir, a Código 3.3 apresenta um trecho do código implementado no Node-RED para a consulta histórica, no qual a query SQL é construída dinamicamente conforme os valores selecionados pelo usuário para o período e o elemento de medição. O parâmetro consultado e o intervalo de datas são definidos de acordo com a entrada fornecida na interface.

Código 3.3 – Consulta Histórica no Node-RED

```
msg.topic = `
SELECT ${campos[variavel].campo} AS valor, data_hora
FROM LeituraSensor
WHERE data_hora >= NOW() - ${intervalo}
ORDER BY data_hora ASC;
`;
```

3.3.3 Acesso do usuário às informações

O acesso do usuário às informações de monitoramento é realizado por meio do *dashboard* disponibilizado pelo Node-RED, acessível via navegador *web*. Por tratar-se de uma aplicação *web* responsiva, o *dashboard* pode ser acessado a partir de qualquer dispositivo com conectividade de rede, incluindo computadores portáteis (*laptops*), *desktops* e dispositivos móveis como *smartphones* e *tablets*.

Para acessar o sistema, o usuário deve primeiro conectar seu dispositivo à rede *Wi-Fi* do laboratório, utilizando as seguintes credenciais:

- **Nome da rede (SSID):** Conforto-Termico
- **Senha:** master1466

Após conectar-se à rede, o usuário deve abrir um navegador *web* e acessar o endereço:

`http://192.168.50.20:1880/ui`

Nesse endereço, o usuário visualiza a interface de monitoramento com os indicadores em tempo real e as opções de consulta histórica.

A interface foi projetada para ser intuitiva e acessível a usuários sem formação técnica especializada. Os elementos visuais do tipo *gauge* apresentam os valores medidos de forma imediata, enquanto o sistema de cores (verde, amarelo, vermelho) comunica o estado qualitativo do ambiente de forma instantânea. Em dispositivos móveis, a interface adapta-se automaticamente ao tamanho da tela, mantendo a legibilidade e a funcionalidade dos componentes.

Dessa forma, o sistema permite que ocupantes de ambientes internos acompanhem as condições de qualidade do ar de qualquer localização dentro da rede, promovendo conscientização ambiental e possibilitando a tomada de decisões informadas sobre ventilação e uso do espaço.

3.4 ESPECIFICAÇÃO DOS ATORES E REGRAS DE NEGÓCIO DA APLICAÇÃO

Nesta seção, definem-se os atores envolvidos na aplicação e as regras de negócio que regem suas interações com o sistema de monitoramento da qualidade do ar interno.

3.4.1 Ator

- **Usuário Final**: Interage com a interface de apresentação para monitorar a qualidade do ar.

3.4.2 Regras de negócio

- O Usuário Final pode acessar o *dashboard* via navegador *web* em dispositivos móveis ou computadores.
- O sistema deve apresentar dados em tempo real, atualizados conforme as leituras dos sensores.
- O Usuário Final pode visualizar gráficos históricos das variáveis ambientais.
- O sistema deve informar visualmente o Usuário Final caso os níveis de poluentes ultrapassem limites predefinidos.
- O Usuário Final não possui permissões para alterar configurações do sistema.

3.5 METODOLOGIA DE APRESENTAÇÃO DAS INFORMAÇÕES

A apresentação das informações coletadas pelo sistema de monitoramento da qualidade do ar interno foi realizada por meio de *dashboards* interativos desenvolvidos utilizando o *Node-RED*. Essa abordagem ofereceu uma série de vantagens, como a facilidade de implementação, a flexibilidade na customização e a acessibilidade multiplataforma.

A eficácia do aplicativo dependeu de sua capacidade de traduzir dados técnicos complexos em informação compreensível e acionável para um público leigo. A literatura sobre monitoramento de QAI enfatiza a importância de não sobrecarregar o usuário com números brutos, cuja interpretação exige conhecimento especializado. Conforme Boerstra, (BOERSTRA; RAUE; CHENG, 2019), a melhor prática é transformar dados em informação que os usuários possam entender intuitivamente.

Reconhecendo as limitações inerentes a sensores de baixo custo sem uma calibração rigorosa em campo, conforme amplamente documentado na literatura (GIORDANO et al., 2021), a interface deste aplicativo foi projetada para focar na comunicação de tendências e na indicação de estados qualitativos, em vez de se apresentar como um instrumento de medição de referência. A utilização de um sistema de cores foi o pilar desta abordagem, comunicando o estado do ambiente de forma instantânea:

- **Verde**: Indica que os níveis de poluentes estão dentro dos limites aceitáveis, sugerindo um ambiente saudável.
- **Amarelo**: Sinaliza que os níveis estão se aproximando dos limites críticos, recomendando atenção e possíveis ações corretivas.
- **Vermelho**: Alerta para níveis elevados de poluentes, indicando a necessidade de intervenção imediata para melhorar a qualidade do ar.

Essa codificação por cores facilitou a rápida assimilação das condições ambientais, permitindo que usuários sem formação técnica pudessem tomar decisões informadas sobre o ambiente monitorado. Além disso, o *dashboard* incluiu gráficos simples e indicadores visuais que destacaram tendências ao longo do tempo, reforçando a compreensão do usuário sobre a dinâmica da qualidade do ar.

Além disso, o aplicativo contou com uma página dedicada à apresentação de informações sobre os elementos de medição monitorados, detalhando suas características, limites recomendados e possíveis efeitos à saúde associados à exposição a cada parâmetro. Essa seção teve como objetivo fornecer ao usuário final um contexto educativo, auxiliando na interpretação dos dados exibidos e promovendo maior conscientização sobre a importância do monitoramento da qualidade do ar interno.

4 RESULTADOS

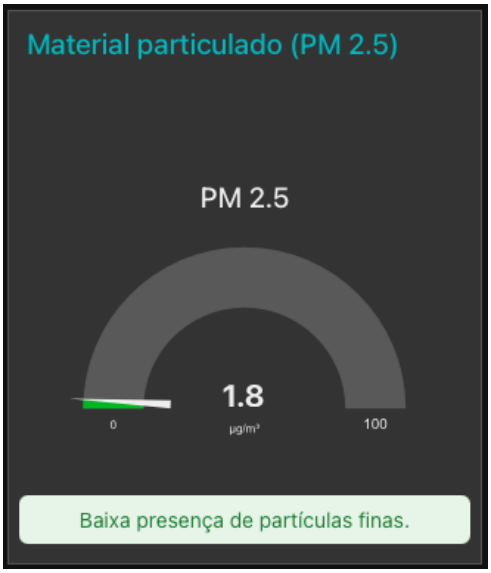
Este capítulo apresenta os resultados obtidos com o sistema de monitoramento da qualidade do ar interno desenvolvido, incluindo a análise das medições em tempo real e histórico, bem como a comparação com os padrões normativos brasileiros.

4.1 RESULTADOS DO MONITORAMENTO EM TEMPO REAL

A Figura 13 apresenta alguns dos medidores implementados no dashboard do Node-RED, ilustrando o resultado da integração entre o sistema embarcado e a plataforma de supervisão. Ressalta-se que, embora nem todas as variáveis de material particulado estejam representadas nas figuras, todas foram devidamente implementadas e podem ser visualizadas ao acessar a aplicação completa. Dessa forma, o dashboard permite o acompanhamento em tempo real de todas as variáveis ambientais monitoradas pelo sistema.

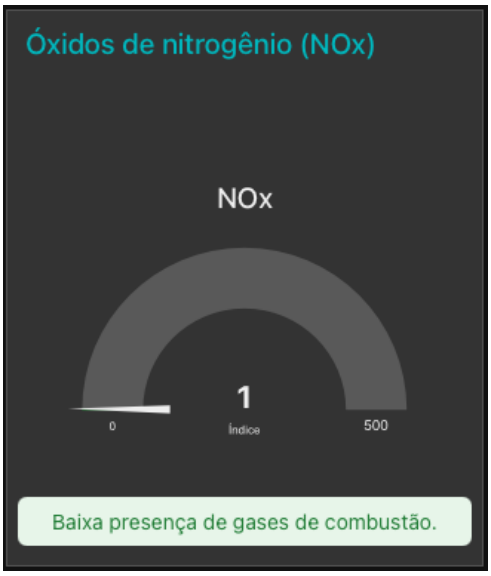
Figura 13 – Resultados da leitura dos sensores

(a) Concentração de CO_2



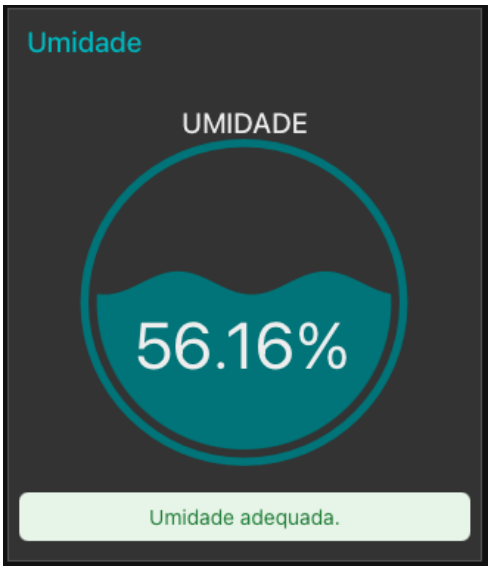
(b) Material Particulado PM2,5

(c) Concentração de COVs



(d) Concentração de NO_x

(e) Temperatura



(f) Umidade relativa

Fonte: Elaborado pelo autor

4.2 ANÁLISE COMPARATIVA COM PADRÕES NORMATIVOS

Os valores apresentados na Figura 13 foram comparados com os padrões de referência estabelecidos pela norma brasileira ABNT NBR 17037:2023 (ABNT, 2023), que define os parâmetros de qualidade do ar interior em ambientes não residenciais climatizados artificialmente. A concentração de CO_2 medida (532 ppm), conforme apresentado na Figura 13a, encontra-se significativamente abaixo do limite estabelecido, que define como valor máximo aceitável a elevação de 700 ppm acima dos níveis externos. Considerando uma concentração atmosférica externa típica de aproximadamente 400 ppm, o limite máximo interno é de 1100 ppm, conforme exemplificado pela própria norma.

Em relação ao PM, a mesma norma NBR 17037:2023 estabelece concentrações médias em 24 horas de 50 μg/m^3 para PM10 e 25 μg/m^3 para PM2,5 como indicadores adequados da qualidade do ar e limpeza do ambiente climatizado. As medições apresentadas na Figura 13b demonstram valores coerentes com essas referências, evidenciando condições satisfatórias em relação à presença de PM no ambiente monitorado.

Quanto aos COVs e NO_x, o sensor Sensirion SEN55 fornece indicadores adimensionais de qualidade do ar, conforme apresentados na Figura 13c e Figura 13d, respectivamente. Esses índices são calculados internamente a partir das respostas do sensor e de algoritmos proprietários do fabricante, representando variações relativas na composição do ar em relação a um nível de referência. Quanto maior o valor do índice, maior é a concentração dos respectivos elementos no ambiente. Dessa forma, os valores obtidos não permitem a verificação direta de conformidade com limites legais ou normativos, mas possibilitam a identificação de tendências e eventos de degradação da qualidade do ar, sendo adequados para aplicações de monitoramento contínuo, conscientização ambiental e apoio à tomada de decisão, como a necessidade de ventilação do ambiente.

No que diz respeito aos parâmetros de temperatura e umidade relativa, embora a norma técnica brasileira (ABNT, 2008) estabeleça faixas recomendadas para conforto térmico em ambientes climatizados, ressalta-se que o objetivo deste trabalho não contemplou a avaliação de conforto térmico, nem a medição de outras variáveis associadas, como a velocidade do ar. No entanto, esses parâmetros são relevantes no contexto da qualidade do ar interno, pois exercem influência direta sobre o crescimento e a proliferação de microrganismos.

Nesse contexto, a Tabela 3 apresenta as faixas de temperatura e umidade relativa aceitáveis de acordo com a estação do ano, conforme estabelecido pela (ABNT, 2008). As medições apresentadas na Figura 13e e Figura 13f indicam temperatura de 27,7°C e umidade relativa de 56,16%, respectivamente. Observa-se que, embora a umidade relativa isoladamente encontre-se dentro do intervalo estabelecido para verão (35 – 65%), a temperatura está acima da faixa recomendada (22,5 – 26,0°C). Essa combinação de tem-

peratura elevada associada a níveis de umidade intermediários pode favorecer condições inadequadas do ponto de vista da proliferação de microrganismos, indicando a necessidade de ajustes no sistema de climatização.

Tabela 3 – Faixas de Temperatura e Umidade Relativa conforme NBR 16401-2

Estação	**Temperatura (°C)**	**Umidade Relativa (%)**
Verão	22,5 – 26,0	35 – 65
Inverno	21,0 – 24,0	30 – 60

Fonte: Adaptado de (ABNT, 2008).

4.3 RESULTADOS DA CONSULTA HISTÓRICA

A Figura 14 apresenta os resultados obtidos a partir da consulta histórica das medições ambientais armazenadas no banco de dados, organizadas em gráficos temporais gerados pela plataforma de supervisão desenvolvida no Node-RED. Os gráficos ilustram o comportamento de três variáveis ambientais (CO_2, PM1,0 e COVs) em dois intervalos de análise (última hora e últimas 24 horas), permitindo tanto a observação de variações instantâneas quanto a identificação de tendências ao longo do tempo. Ressalta-se que, embora os exemplos apresentados estejam concentrados nesses elementos e períodos, a aplicação implementada contempla a visualização histórica para todos os parâmetros monitorados (CO_2, PM1,0, PM2,5, PM4,0, PM10, temperatura e umidade relativa) e permite a seleção de diferentes intervalos, incluindo 1 hora, 24 horas, 7 dias e 30 dias, conforme a necessidade do usuário.

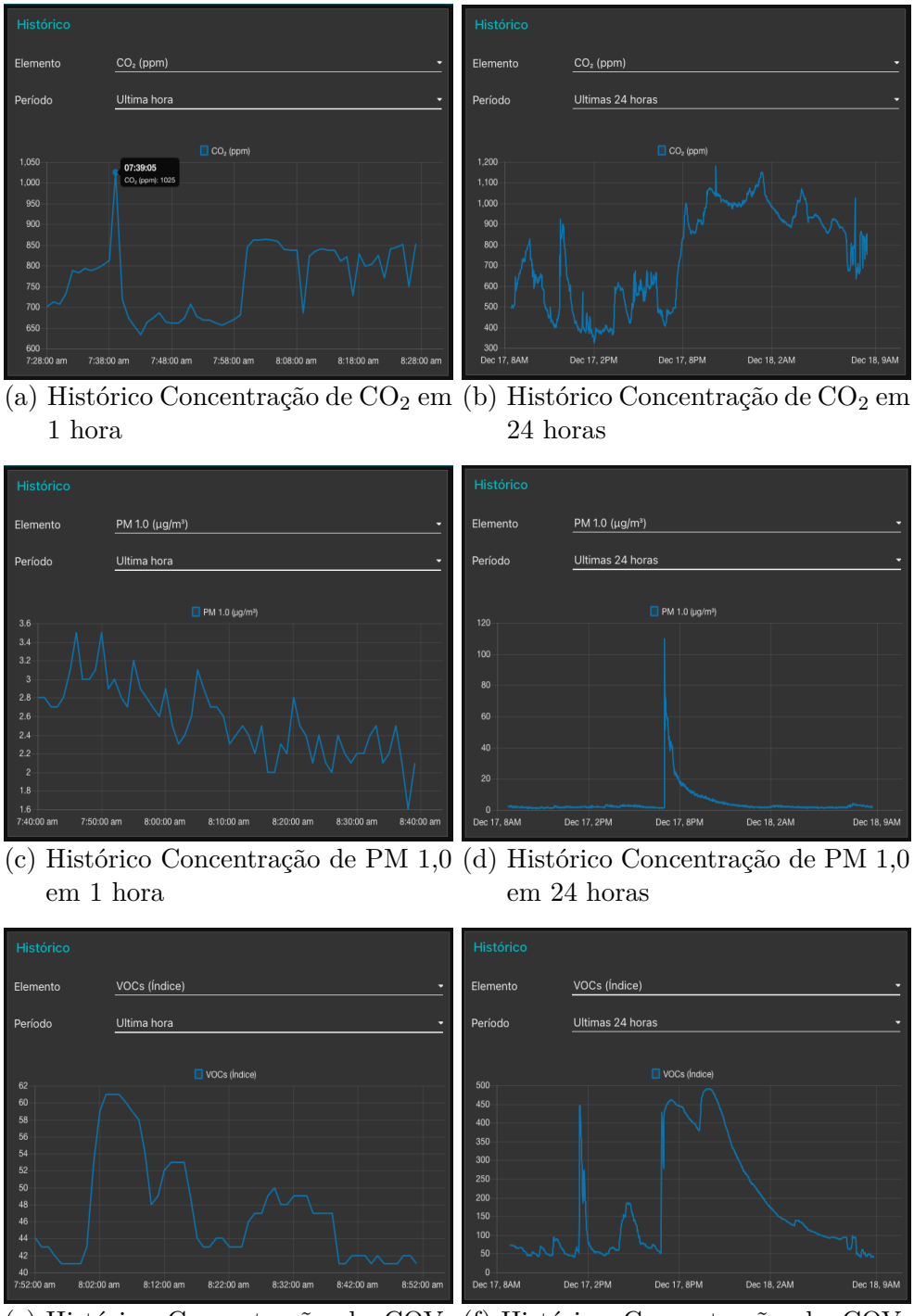
As Figuras Figura 14a e Figura 14b mostram o histórico da concentração de CO_2. No intervalo de uma hora, observa-se a presença de oscilações rápidas e picos pontuais, característicos de alterações momentâneas na ocupação do ambiente ou em sua ventilação. Já no período de 24 horas, o gráfico evidencia ciclos mais amplos, com elevação gradual dos níveis de CO_2 em determinados períodos e reduções subsequentes, indicando variações associadas ao uso do espaço ao longo do dia.

Os gráficos referentes à concentração de PM1,0, apresentados nas Figuras Figura 14c e Figura 14d, revelam um comportamento distinto. Na análise de curto prazo, nota-se uma tendência de redução gradual com pequenas flutuações, possivelmente associadas à deposição das partículas ou à ausência de fontes emissoras imediatas. No histórico de 24 horas, destaca-se a ocorrência de um pico acentuado, seguido de rápida diminuição, o que pode indicar um evento isolado de geração de partículas, como movimentação intensa no ambiente ou entrada de ar externo contaminado.

As Figuras Figura 14e e Figura 14f apresentam o histórico do índice de COVs. No período de uma hora, observa-se uma elevação abrupta do índice, seguida por oscilações e

posterior redução, sugerindo a presença temporária de fontes emissoras, como produtos de limpeza, materiais sintéticos ou atividades humanas específicas. No intervalo de 24 horas, o gráfico evidencia picos de maior magnitude e uma tendência de decaimento gradual ao longo do tempo, comportamento coerente com a dispersão e diluição dos compostos no ambiente.

Figura 14 – Resultados dos gráficos de leitura dos sensores



(a) Histórico Concentração de CO_2 em 1 hora

(b) Histórico Concentração de CO_2 em 24 horas

(c) Histórico Concentração de PM 1,0 em 1 hora

(d) Histórico Concentração de PM 1,0 em 24 horas

(e) Histórico Concentração de COVs em 1 hora

(f) Histórico Concentração de COVs em 24 horas

Fonte: Elaborado pelo autor

A plataforma desenvolvida no Node-RED demonstrou-se eficaz tanto para o monitoramento em tempo real quanto para a análise histórica dos dados ambientais, integrando de forma robusta o sistema embarcado, o banco de dados relacional e a interface gráfica de visualização. Os resultados apresentados evidenciam a capacidade do sistema em registrar, armazenar e exibir a evolução temporal das variáveis ambientais de maneira clara e acessível. A análise comparativa entre diferentes intervalos temporais possibilita uma compreensão mais aprofundada das condições do ambiente monitorado, contribuindo para a identificação de padrões, eventos críticos e potenciais riscos à qualidade do ar interno.

5 CONCLUSÕES

Este trabalho apresentou o desenvolvimento e a validação de uma solução integrada para o monitoramento da Qualidade do Ar Interno (QAI) em ambientes internos, utilizando sensores ambientais, um sistema embarcado baseado no microcontrolador ESP32, comunicação via protocolo MQTT e uma plataforma de supervisão e visualização desenvolvida no *Node-RED*. A proposta teve como foco principal a criação de uma arquitetura modular, escalável e de baixo custo, voltada à aplicação em ambientes educacionais, com ênfase na clareza da informação e na acessibilidade dos dados aos usuários.

A partir da fundamentação teórica apresentada, evidenciou-se que a qualidade do ar interno exerce influência direta sobre a saúde, o conforto e o desempenho cognitivo dos ocupantes. Parâmetros como concentração de dióxido de carbono, material particulado e compostos orgânicos voláteis são amplamente reconhecidos na literatura como indicadores relevantes das condições ambientais em ambientes internos. Nesse contexto, o sistema desenvolvido mostrou-se alinhado às necessidades atuais de monitoramento contínuo e à disseminação de informações ambientais de forma compreensível e acessível.

Do ponto de vista técnico, os resultados obtidos demonstram que o sistema embarcado implementado foi capaz de realizar a aquisição periódica e confiável das variáveis ambientais propostas. A integração dos sensores SCD30 e SEN55 permitiu a obtenção simultânea de múltiplos parâmetros de qualidade do ar, enquanto o uso do microcontrolador ESP32 possibilitou o processamento local, a apresentação das informações em *display* LCD e a transmissão local dos dados por meio de rede sem fio. A utilização do protocolo MQTT mostrou-se adequada para aplicações de IoT, oferecendo baixa latência, eficiência na comunicação e robustez frente a instabilidades temporárias da rede.

A plataforma desenvolvida no *Node-RED* desempenhou papel fundamental na consolidação do sistema, atuando como camada intermediária entre o *hardware* embarcado, o banco de dados relacional e a interface gráfica de visualização. Os fluxos implementados permitiram tanto a representação em tempo real das variáveis ambientais quanto o armazenamento estruturado das medições em banco de dados *MySQL*, viabilizando consultas históricas e análises temporais. A separação entre os fluxos de recepção via MQTT e os fluxos de consulta SQL contribuiu para a organização lógica do sistema, facilitando sua manutenção e possíveis expansões futuras.

Os *dashboards* desenvolvidos demonstraram-se eficazes na tradução de dados técnicos em informações visuais de fácil interpretação. A utilização de medidores instantâneos, gráficos temporais e descrições qualitativas baseadas em faixas de referência permitiu ao usuário identificar rapidamente alterações nas condições ambientais e compreender seus

possíveis impactos. A análise dos gráficos históricos evidenciou variações significativas nos níveis de CO_2, PM e COVs ao longo do tempo, reforçando a importância do monitoramento contínuo e da visualização histórica como ferramentas de apoio à análise da qualidade do ar.

Dessa forma, conclui-se que os objetivos propostos foram plenamente alcançados. O sistema desenvolvido demonstrou viabilidade técnica, integração eficiente entre *hardware* e *software* e potencial de aplicação prática em ambientes educacionais. A solução apresentada contribui para a disseminação do conceito de Qualidade do Ar Interno e evidencia o papel das tecnologias baseadas em IoT como ferramentas relevantes para a promoção de ambientes mais saudáveis, confortáveis e conscientes do ponto de vista ambiental.

O código-fonte desenvolvido para este trabalho está disponível em repositório público no *GitHub*[1]. Nele encontram-se o *firmware* do sistema embarcado, os fluxos do *Node-RED* e arquivos auxiliares, permitindo que outros possam reproduzir, adaptar e evoluir a solução proposta.

TRABALHOS FUTUROS

Como continuidade do trabalho desenvolvido, identificam-se diversas possibilidades de aprimoramento e expansão do sistema proposto, as quais são apresentadas a seguir:

- **Aplicação de técnicas de análise avançada de dados e aprendizado de máquina:** Empregar algoritmos de análise preditiva e aprendizado de máquina para a identificação de padrões de comportamento ambiental, previsão de eventos críticos e detecção automática de anomalias. Essas técnicas poderiam auxiliar na antecipação de condições de degradação da qualidade do ar e no suporte à tomada de decisão preventiva.

- **Desenvolvimento de aplicações móveis e sistemas de notificação:** Desenvolver aplicações móveis dedicadas e mecanismos de notificação em tempo real, permitindo alertar os usuários sobre condições ambientais inadequadas. Essa funcionalidade ampliaria o alcance do sistema e melhoraria a interação com os usuários finais.

- **Expansão da rede de monitoramento e estudos comparativos:** Replicar o sistema em múltiplos ambientes com diferentes características construtivas, padrões de ocupação e sistemas de ventilação, possibilitando análises comparativas e estudos mais abrangentes sobre a Qualidade do Ar Interno e sua relação com conforto, saúde e desempenho humano.

[1] https://github.com/DanielValdeley/projeto-monitoramento-QAI

- **Implantação de servidor em nuvem para acesso remoto:** Disponibilizar a plataforma de monitoramento em um servidor em nuvem, permitindo o acesso remoto aos dados e *dashboards* por meio da *internet*. Essa abordagem facilitaria o acompanhamento em tempo real de múltiplos ambientes, independentemente da localização física dos usuários, além de aumentar a escalabilidade e a segurança do sistema.

- **Implementação de procedimentos de calibração dos sensores:** Realizar a calibração dos sensores por meio de comparação com instrumentos de referência certificados, empregando a técnica de colocação (*collocation calibration*) conforme recomendado pela literatura especializada. Essa etapa é fundamental para reduzir a incerteza de medição e aumentar a confiabilidade das leituras, permitindo que o sistema atenda a padrões metrológicos mais rigorosos e possibilitando sua aplicação em contextos que exigem rastreabilidade e conformidade com normas técnicas de qualidade do ar. A calibração deveria contemplar diferentes condições ambientais e períodos sazonais, garantindo a robustez dos modelos de correção aplicados.

- **Aprimoramento da visualização com faixas de referência nos instrumentos:** Implementar indicações visuais de faixas de referência diretamente nos instrumentos analógicos (*gauges*) do *dashboard*, permitindo ao usuário identificar instantaneamente se os valores medidos estão dentro das faixas aceitáveis. As faixas sugeridas, baseadas nas normas técnicas e literatura especializada, são:

  - **Temperatura:** faixa ótima de 21 a 26°C, conforme NBR 16401-2;
  - **Umidade relativa:** faixa ótima de 30 a 65%, conforme NBR 16401-2;
  - **CO_2:** mínimo de 350 ppm (nível atmosférico), faixa aceitável abaixo de 1.000 ppm, faixa crítica acima de 2.000 ppm;
  - **PM2,5:** faixa aceitável abaixo de 25 μg/m^3 (conforme NBR 17037:2023);
  - **VOC e NOx:** escala de 0 a 500, onde o valor 100 representa a medição de referência (média móvel das últimas 24 horas), valores acima indicam degradação relativa da qualidade do ar.

  Essa melhoria tornaria a interface mais informativa e facilitaria a interpretação dos dados por usuários sem conhecimento técnico especializado.

REFERÊNCIAS

ABNT. **NBR 16401-2: Instalações de ar-condicionado – Sistemas centrais e unitários – Parte 2: Parâmetros de conforto térmico**. Rio de Janeiro: Associação Brasileira de Normas Técnicas, 2008. Disponível em: <http://ftp.demec.ufpr.br/disciplinas/EngMec_NOTURNO/TM374/NBR_16401-2_2008.pdf>.

_____. **NBR 17037: Qualidade do ar interior em ambientes não residenciais climatizados artificialmente**. Rio de Janeiro: Associação Brasileira de Normas Técnicas, abr. 2023. Disponível em: <https://baktron.com.br/wp-content/uploads/2025/07/ABNT-NBR-17037-Qualidade-do-ar-interior-em-ambientes-nao-residenciais-climatizados-artificialmente-042023.pdf>.

ANVISA. **Resolução RE nº 9, de 16 de janeiro de 2003**. 2003. Diário Oficial da União. Disponível em: <https://antigo.anvisa.gov.br/documents/10181/2718376/RE_09_2003_.pdf/8ccafc91-1437-4695-8e3a-2a97deca4e10>.

BOERSTRA, Atze; RAUE, Arjen; CHENG, Louie. Smart monitoring of building performance with IEQ sensor networks. **REHVA Journal**, v. 56, n. 4, p. 37–41, 2019. Disponível em: <https://www.rehva.eu/rehva-journal/chapter/smart-monitoring-of-building-performance-with-ieq-sensor-networks>.

ECLIPSE MOSQUITTO. **Eclipse Mosquitto An open source MQTT broker**. Disponível em: <https://mosquitto.org/>. Acesso em: 24 jul. 2025.

GIORDANO, Michael R. et al. From low-cost sensors to high-quality data: A summary of challenges and best practices for effectively calibrating low-cost particulate matter mass sensors. **Journal of Aerosol Science**, v. 158, p. 105833, 2021. Disponível em: <https://hal.science/hal-03442005v1>.

GONZALEZ-MARTÍN, Javier et al. A state-of-the-art review on indoor air pollution and strategies for indoor air pollution control. **Chemosphere**, Elsevier, v. 262, p. 128376, 2021. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0045653520321317>.

HIVEMQ. **Apresentando o Protocolo MQTT – Fundamentos do MQTT: Parte 1**. Disponível em: <https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/>. Acesso em: 23 jul. 2025.

HONEYWELL. **HPM Series Particulate Matter Sensors – Datasheet**. 2021. Disponível em: <https://br.mouser.com/datasheet/2/187/HWSC_S_A0012942921_1-3073234.pdf>.

INTERNATIONAL AGENCY FOR RESEARCH ON CANCER. **Formaldehyde, 2-Butoxyethanol and 1-tert-Butoxypropan-2-ol**. Lyon: IARC, 2006. v. 88. (IARC Monographs on the Evaluation of Carcinogenic Risks to Humans). Disponível em: <https://publications.iarc.who.int/Book-And-Report-Series/Iarc-Monographs-On-The-Identification-Of-Carcinogenic-Hazards-To-Humans/Formaldehyde-2-Butoxyethanol-And-1--Em-Tert-Em--Butoxypropan-2-ol-2006>.

JIN, Shengjia et al. Indoor Volatile Organic Compounds: Concentration Characteristics and Health Risk Analysis on a University Campus. **International Journal of Environmental Research and Public Health**, v. 20, n. 10, p. 5829, 2023. Disponível em: <https://www.mdpi.com/1660-4601/20/10/5829>.

MARQUES, Simone. **Aplicativos híbridos: tudo o que você precisa saber**. Dez. 2024. Disponível em: <https://uds.com.br/blog/aplicativos-hibridos/>. Acesso em: 7 jul. 2025.

NODE-RED. **Low-code programming for event-driven applications**. Disponível em: <https://nodered.org>. Acesso em: 24 jul. 2025.

PRANAIR. **O que é Material Particulado (PM)**. 2025. Disponível em: <https://www.pranaair.com/pt-pt/what-is-particulate-matter-pm/>. Acesso em: 3 jul. 2025.

QIAN, Hua; ZHENG, Xiaohong. Ventilation control for airborne transmission of human exhaled bio-aerosols in buildings. **Journal of Thoracic Disease**, v. 10, Suppl 19, 2018. ISSN 2077-6624. Disponível em: <https://doi.org/10.21037/jtd.2018.01.24>.

SALTHAMMER, Tunga. TVOC – Revisited. **Environment International**, Elsevier, v. 167, p. 107440, 2022. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0160412022003671>.

SAUERMANN. **The dangers of VOCS: Formaldehyde (CH2O)**. Abr. 2024. Disponível em: <https://sauermanngroup.com/es-US/node/38142>. Acesso em: 1 jul. 2025.

SENSIRION. **Datasheet SCD30 Sensor Module – CO_2, Humidity, and Temperature Sensor**. 2020. Disponível em: <https://www.mouser.com/datasheet/2/682/Sensirion_CO2_Sensors_SCD30_Datasheet-3539430.pdf>.

_____. **Datasheet SEN5x**. 2022. Disponível em: <https://br.mouser.com/datasheet/3/1278/1/F5BAF15E2B8C00B004F2B10D9E302F5D0FE2299CF4EE258A758E31959FB1362B.pdf>.

SENSIRION AG. **Datasheet SCD30: Sensor Module for HVAC and Indoor Air Quality Applications**. 2020. Disponível em: <https://sensirion.com/media/documents/4EAF6AF8/61652C3C/Sensirion_CO2_Sensors_SCD30_Datasheet.pdf>. Acesso em: 1 jul. 2025.

SENSIRION AG. **What is Sensirion's VOC Index?** 2022. Disponível em: <https://sensirion.com/media/documents/02232963/6294E043/Info_Note_VOC_Index.pdf>. Acesso em: 1 jul. 2025.

UNITED STATES ENVIRONMENTAL PROTECTION AGENCY. **Heating, Ventilation and Air-Conditioning Systems, Part of Indoor Air Quality Design Tools for Schools**. 2023. Disponível em: <https://www.epa.gov/iaq-schools/heating-ventilation-and-air-conditioning-systems-part-indoor-air-quality-design-tools>. Acesso em: 15 jan. 2024.

WISCONSIN DEPARTMENT OF HEALTH SERVICES. **Carbon Dioxide**. 2025. Disponível em: <https://www.dhs.wisconsin.gov/chemical/carbondioxide.htm>. Acesso em: 8 jul. 2025.

YUAN, Michael. **Conhecendo o MQTT**. Dez. 2021. Disponível em: <https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/>. Acesso em: 10 jul. 2025.

ANEXO A – CÓDIGO-FONTE COMPLETO DO FIRMWARE DO SISTEMA EMBARCADO

Este anexo apresenta o código-fonte completo do *firmware* desenvolvido para o microcontrolador ESP32, responsável pela aquisição de dados dos sensores ambientais, armazenamento local em cartão SD, apresentação das informações no display LCD e transmissão remota via protocolo MQTT.

O código foi implementado utilizando a plataforma Arduino e integra as bibliotecas necessárias para comunicação com os sensores SCD30 e SEN55, módulo RTC, display LCD, cartão SD e conectividade Wi-Fi. A estrutura modular do *firmware* permite fácil manutenção e expansão do sistema.

Código A.1 – Código-fonte completo do firmware ESP32

```
#include <Arduino.h>
#include <Wire.h>
#include <SensirionI2cScd30.h>
#include <SensirionI2CSen5x.h>
#include <LiquidCrystal_I2C.h>
#include <RTClib.h>
#include <SPI.h>
#include <FS.h>
#include <SD.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>

// Manter compatibilidade sensirion
#define NO_ERROR 0

// Definição dos pinos do PRIMEIRO barramento I2C (LCD)
#define SDA_1 21  // Verde
#define SCL_1 22  // Amarelo

// Definição dos pinos do SEGUNDO barramento I2C (Sensor SCD30)
#define SDA_2 33  // Verde
#define SCL_2 32  // Amarelo

// Definição do pino da chave que habilita gravação
#define CHAVE 14

// Definição do pino do led
#define LED 12

```

```
31 // Configuração do display LCD
32 int lcdColumns = 20;
33 int lcdRows = 4;
34 LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);
35 // Variável para controlar a limpeza do LCD em um loop (evita do LCD piscar)
36 int controleLCD = 0;
37
38 // Criando instância para o segundo barramento I2C
39 TwoWire I2Ctwo = TwoWire(1);
40
41 // Criando instância do sensores
42 SensirionI2cScd30 sensorSCD30;
43 SensirionI2CSen5x sensorSEN55;
44
45 // Criando instância do TinyRTC
46 RTC_DS1307 rtc;
47
48 // Struct para o SCD30
49 struct SCD30Data {
50   float co2;
51   float temperature;
52   float humidity;
53 };
54
55 SCD30Data dadosSCD30;
56
57 // Struct para o SEN55
58 struct SEN55Data {
59   float massConcentrationPm1p0;
60   float massConcentrationPm2p5;
61   float massConcentrationPm4p0;
62   float massConcentrationPm10p0;
63   float ambientHumidity;
64   float ambientTemperature;
65   float vocIndex;
66   float noxIndex;
67 };
68
69 SEN55Data dadosSEN55;
70
71 // Dias da semana
72 char daysOfTheWeek[7][14] = {
73   "Domingo", "Segunda-feira", "Terca-feira",
74   "Quarta-feira", "Quinta-feira", "Sexta-feira", "Sábado"
75 };
76
77 // Armazenar erros
```

```
78 static char errorMessage[256];
79 static int16_t error;
80
81 // Variável para armazenar o nome do arquivo
82 String arquivo = "";
83
84 // Determina o tempo entre as medidas (milisegundos)
85 const int intervalo = 30;  // Intervalo desejado em segundos
86 unsigned long ultimoTempoExecutado = 0;
87 unsigned int intervaloPorTela = 4000;
88
89 // ############################################################
90 // #                        Funções                           #
91 // ############################################################
92
93 void acendeLed() {
94   // Acende o LED
95   digitalWrite(LED, HIGH);
96   // Intervalo 500ms
97   delay(500);
98   // Apaga o LED
99   digitalWrite(LED, LOW);
100 }
101
102 // Limpar o display
103 void limparLinhaLCD(int linha) {
104   if (linha >= 0 && linha <= 3) {
105     lcd.setCursor(0, linha);
106     lcd.print("                ");
107   } else {
108     lcd.clear();
109   }
110 }
111
112 // Substitui o ponto por vírgula como separador de casas decimais
113 String numeroComVirgula(float valor, int casasDecimais = 2) {
114   String resultado = String(valor, casasDecimais);
115   resultado.replace('.', ',');
116   return resultado;
117 }
118
119 // Obter as leituras do sensor SCD30
120 bool lerSCD30(SCD30Data &dadosSCD30) {
121   dadosSCD30.co2 = dadosSCD30.temperature = dadosSCD30.humidity = -1.0;  // Define
      valores padrão em caso de erro
122   error = sensorSCD30.blockingReadMeasurementData(dadosSCD30.co2, dadosSCD30.
      temperature, dadosSCD30.humidity);
```

```
123   if (error != NO_ERROR) {
124     Serial.print("Erro ao ler SCD30: ");
125     errorToString(error, errorMessage, sizeof errorMessage);
126     Serial.println(errorMessage);
127     return false;
128   } else {
129     return true;
130   }
131 }
132 
133 // Obter as leituras do sensor SEN55
134 bool lerSEN55(SEN55Data &dadosSEN55) {
135   dadosSEN55.massConcentrationPm1p0 = -1;  // Define valores padrão em caso de erro
136   dadosSEN55.massConcentrationPm2p5 = -1;
137   dadosSEN55.massConcentrationPm4p0 = -1;
138   dadosSEN55.massConcentrationPm10p0 = -1;
139   dadosSEN55.ambientHumidity = -1;
140   dadosSEN55.ambientTemperature = -1;
141   dadosSEN55.vocIndex = -1;
142   dadosSEN55.noxIndex = -1;
143 
144   error = sensorSEN55.readMeasuredValues(
145     dadosSEN55.massConcentrationPm1p0, dadosSEN55.massConcentrationPm2p5,
      dadosSEN55.massConcentrationPm4p0, dadosSEN55.massConcentrationPm10p0,
146     dadosSEN55.ambientHumidity, dadosSEN55.ambientTemperature, dadosSEN55.vocIndex,
       dadosSEN55.noxIndex);
147 
148   if (error != NO_ERROR) {
149     Serial.print("Erro ao ler SEN55: ");
150     errorToString(error, errorMessage, sizeof errorMessage);
151     Serial.println(errorMessage);
152     return false;
153   } else {
154     return true;
155   }
156 }
157 
158 // Retorna a data e a hora do RTC
159 String obterDataHora(bool formatoGravacao) {
160   DateTime now = rtc.now();
161 
162   String yearStr = String(now.year());
163   String monthStr = (now.month() < 10 ? "0" : "") + String(now.month());
164   String dayStr = (now.day() < 10 ? "0" : "") + String(now.day());
165   String hourStr = (now.hour() < 10 ? "0" : "") + String(now.hour());
166   String minuteStr = (now.minute() < 10 ? "0" : "") + String(now.minute());
167   String secondStr = (now.second() < 10 ? "0" : "") + String(now.second());
```

```
168   // String dayOfWeek = daysOfTheWeek[now.dayOfTheWeek()];
169
170   String formattedTime = "";
171
172   if (formatoGravacao) {
173     formattedTime = yearStr + "/" + monthStr + "/" + dayStr + " " + hourStr + ":" +
       minuteStr + ":" + secondStr;
174   } else {
175     formattedTime = dayStr + "/" + monthStr + "/" + yearStr + " " + hourStr + ":" +
       minuteStr + ":" + secondStr;
176   }
177   return formattedTime;
178 }
179
180 // Atualizar os dados exibidos no LCD
181 void atualizaLCD(SCD30Data &dadosSCD30, SEN55Data &dadosSEN55) {
182   // Valores do SCD30
183   lcd.clear();
184   lcd.setCursor(0, 0);
185   lcd.print("CO2: " + String(dadosSCD30.co2) + " ppm  ");
186   lcd.setCursor(0, 1);
187   lcd.print("Temp: " + String(dadosSCD30.temperature) + " C ");
188   lcd.setCursor(0, 2);
189   lcd.print("Umid: " + String(dadosSCD30.humidity) + " % ");
190   delay(intervaloPorTela);
191
192   // Valores do SEN55
193   lcd.clear();
194   lcd.setCursor(0, 0);
195   lcd.print("PM 1.0: " + String(dadosSEN55.massConcentrationPm1p0));
196   lcd.setCursor(0, 1);
197   lcd.print("PM 2.5: " + String(dadosSEN55.massConcentrationPm2p5));
198   lcd.setCursor(0, 2);
199   lcd.print("PM 4.0: " + String(dadosSEN55.massConcentrationPm4p0));
200   lcd.setCursor(0, 3);
201   lcd.print("PM10.0: " + String(dadosSEN55.massConcentrationPm10p0));
202   delay(intervaloPorTela);
203   lcd.clear();
204   lcd.setCursor(0, 0);
205   lcd.print("Temp: " + String(dadosSEN55.ambientTemperature) + " C ");
206   lcd.setCursor(0, 1);
207   lcd.print("Umid: " + String(dadosSEN55.ambientHumidity) + " % ");
208   lcd.setCursor(0, 2);
209   lcd.print("VocIndex: " + String(dadosSEN55.vocIndex));
210   lcd.setCursor(0, 3);
211   lcd.print("NoxIndex: " + String(dadosSEN55.noxIndex));
212   delay(intervaloPorTela);
```

```
213 }
214
215 // Atualizar os dados exibidos no terminal
216 void atualizaTerminal(SCD30Data &dadosSCD30, SEN55Data &dadosSEN55) {
217   // Valores do SCD30
218   Serial.println("========== SCD30 ==========");
219   Serial.print("CO2: ");
220   Serial.print(dadosSCD30.co2);
221   Serial.print(" ppm\tTemp: ");
222   Serial.print(dadosSCD30.temperature);
223   Serial.print(" C\tUmidade: ");
224   Serial.println(dadosSCD30.humidity);
225
226   // Valores do SEN55
227   Serial.println("========== SEN55 ==========");
228   Serial.print("Massa PM1.0: ");  // Material Particulado com diametro inferior a
        1,0 um
229   Serial.print(dadosSEN55.massConcentrationPm1p0);
230   Serial.print("\t");
231   Serial.print("Massa PM2.5: ");  // Material Particulado com diametro inferior a
        2,5 um
232   Serial.print(dadosSEN55.massConcentrationPm2p5);
233   Serial.print("\t");
234   Serial.print("Massa PM4.0: ");  // Material Particulado com diametro inferior a
        4,0 um
235   Serial.print(dadosSEN55.massConcentrationPm4p0);
236   Serial.print("\t");
237   Serial.print("Massa PM10.0: ");  // Material Particulado com diametro inferior a
        10,0 um
238   Serial.print(dadosSEN55.massConcentrationPm10p0);
239   Serial.print("\t");
240   Serial.print("Umidade:");
241   if (isnan(dadosSEN55.ambientHumidity)) {
242     Serial.print("n/a");
243   } else {
244     Serial.print(dadosSEN55.ambientHumidity);
245   }
246   Serial.print("\t");
247   Serial.print("Temperatura:");
248   if (isnan(dadosSEN55.ambientTemperature)) {
249     Serial.print("n/a");
250   } else {
251     Serial.print(dadosSEN55.ambientTemperature);
252   }
253   Serial.print("\t");
254   Serial.print("VocIndex:");
255   if (isnan(dadosSEN55.vocIndex)) {
```

```
256     Serial.print("n/a");
257   } else {
258     Serial.print(dadosSEN55.vocIndex);
259   }
260   Serial.print("\t");
261   Serial.print("NoxIndex:");
262   if (isnan(dadosSEN55.noxIndex)) {
263     Serial.println("n/a");
264   } else {
265     Serial.println(dadosSEN55.noxIndex);
266   }
267 }
268 
269 // Atualiza os dados no arquivo atual
270 void atualizaArquivo(String arquivo, SCD30Data &dadosSCD30, SEN55Data &dadosSEN55,
      String dataAtual) {
271   // Ordem de valores da string: data;co2;scd30_temp;scd30_umid;pm1.0;pm2.5;pm4.0;
      pm10.0;sen55_umid;sen55_temp;vocindex;noxindex
272   String valores = "";
273   valores += dataAtual + ";";
274   valores += String(numeroComVirgula(dadosSCD30.co2)) + ";";
275   valores += String(numeroComVirgula(dadosSCD30.temperature)) + ";";
276   valores += String(numeroComVirgula(dadosSCD30.humidity)) + ";";
277   valores += String(numeroComVirgula(dadosSEN55.massConcentrationPm1p0)) + ";";
278   valores += String(numeroComVirgula(dadosSEN55.massConcentrationPm2p5)) + ";";
279   valores += String(numeroComVirgula(dadosSEN55.massConcentrationPm4p0)) + ";";
280   valores += String(numeroComVirgula(dadosSEN55.massConcentrationPm10p0)) + ";";
281   valores += String(numeroComVirgula(dadosSEN55.ambientHumidity)) + ";";
282   valores += String(numeroComVirgula(dadosSEN55.ambientTemperature)) + ";";
283   valores += String(numeroComVirgula(dadosSEN55.vocIndex)) + ";";
284   valores += String(numeroComVirgula(dadosSEN55.noxIndex)) + ";";
285   valores += "\n";
286   appendFile(SD, arquivo.c_str(), valores.c_str());
287 }
288 
289 // Criar um novo arquivo no cartão SD
290 void writeFile(fs::FS &fs, const char *path, const char *message) {
291   Serial.printf("Writing file: %s\n", path);
292 
293   File file = fs.open(path, FILE_WRITE);
294   if (!file) {
295     Serial.println("Failed to open file for writing");
296     return;
297   }
298   if (file.print(message)) {
299     Serial.println("File written");
300   } else {
```

```
301     Serial.println("Write failed");
302   }
303   file.close();
304 }
305 
306 // Agregar dados em um arquivo
307 void appendFile(fs::FS &fs, const char *path, const char *message) {
308   Serial.printf("Appending to file: %s\n", path);
309 
310   File file = fs.open(path, FILE_APPEND);
311   if (!file) {
312     Serial.println("Failed to open file for appending");
313     return;
314   }
315   if (file.print(message)) {
316     Serial.println("Message appended");
317   } else {
318     Serial.println("Append failed");
319   }
320   file.close();
321 }
322 
323 // Obtém o nome do próximo arquivo
324 String getNextFileName(fs::FS &fs, const char *prefix, const char *extension) {
325   File root = fs.open("/");
326   if (!root || !root.isDirectory()) {
327     Serial.println("Failed to open root directory");
328     return "";
329   }
330 
331   int maxIndex = 0;
332 
333   File file = root.openNextFile();
334   while (file) {
335     String name = file.name();  // exemplo: /medidas-3.csv
336 
337     // Remove o '/' inicial se existir
338     if (name.startsWith("/")) {
339       name = name.substring(1);
340     }
341 
342     if (name.startsWith(prefix) && name.endsWith(extension)) {
343       int dashIndex = name.indexOf('-');
344       int dotIndex = name.lastIndexOf('.');
345 
346       if (dashIndex != -1 && dotIndex != -1 && dotIndex > dashIndex) {
347         String numberStr = name.substring(dashIndex + 1, dotIndex);
```

```
348         int number = numberStr.toInt();
349         if (number > maxIndex) {
350           maxIndex = number;
351         }
352       }
353     }
354
355     file = root.openNextFile();
356   }
357
358   int nextIndex = maxIndex + 1;
359   return String("/") + prefix + "-" + nextIndex + extension;
360 }
361
362 // ###########################################################
363 // #                          Setup                          #
364 // ###########################################################
365
366 // Rede wifi
367 const char* ssid = "Conforto-Termico";
368 const char* password = "master1466";
369
370 // Config MQTT
371 const char* mqtt_server = "192.168.50.20";
372 const int   mqtt_port = 1883;
373
374 WiFiClient espClient;
375 PubSubClient client(espClient);
376
377 void publicarMQTT(SCD30Data &dadosSCD30, SEN55Data &dadosSEN55) {
378   StaticJsonDocument<900> doc;
379
380   doc["co2"]         = dadosSCD30.co2;
381   doc["temperatura"] = dadosSCD30.temperature;
382   doc["umidade"]     = dadosSCD30.humidity;
383
384   doc["pm1.0"]  = dadosSEN55.massConcentrationPm1p0;
385   doc["pm2.5"]  = dadosSEN55.massConcentrationPm2p5;
386   doc["pm4.0"]  = dadosSEN55.massConcentrationPm4p0;
387   doc["pm10.0"] = dadosSEN55.massConcentrationPm10p0;
388
389   if (!isnan(dadosSEN55.vocIndex)) {
390     doc["vocIndex"] = dadosSEN55.vocIndex;
391   }
392
393   if (!isnan(dadosSEN55.noxIndex)) {
394     doc["noxIndex"] = dadosSEN55.noxIndex;
```

```
395   }
396
397   char payload[900];
398   serializeJson(doc, payload);
399
400   client.publish("sensores/esp32", payload);
401 }
402
403 void mqtt_reconnect() {
404   while (!client.connected()) {
405     lcd.clear();
406     lcd.setCursor(0, 0);
407     lcd.print("Reconectando MQTT...");
408     Serial.print("Reconectando MQTT...");
409     delay(1000);
410
411     if (client.connect("ESP32Client", "admin", "paulvandyk11")) {
412       Serial.println("Conectado ao MQTT!");
413       lcd.clear();
414       lcd.setCursor(0,0);
415       lcd.print("Conectado ao MQTT!");
416       client.subscribe("comandos/esp32");
417     } else {
418       lcd.clear();
419       lcd.print("Falhou, rc=");
420       Serial.print("Falhou, rc=");
421       lcd.setCursor(0,2);
422       lcd.print(client.state());
423       Serial.print(client.state());
424       lcd.setCursor(0,3);
425       lcd.print("Retry in 5 sec...");
426       delay(5000);
427     }
428   }
429 }
430
431 void setup() {
432   Serial.begin(115200);
433   while (!Serial) {
434     delay(100);
435   }
436
437   client.setServer(mqtt_server, mqtt_port);
438
439   // Inicializa a chave para leitura de dados
440   pinMode(CHAVE, INPUT_PULLUP);
441
```

```
442   // Inicializa o LED indicador de leitura
443   pinMode(LED, OUTPUT);
444
445   // Inicializa o primeiro barramento I2C manualmente
446   Wire.begin(SDA_1, SCL_1);
447   delay(500);
448
449   // Inicializa o segundo barramento I2C
450   I2Ctwo.begin(SDA_2, SCL_2);
451   delay(500);
452
453   // Inicializa o display LCD no primeiro barramento I2C
454   lcd.init();
455   lcd.backlight();
456   lcd.setCursor(0, 0);
457   lcd.print("Iniciando...");
458   delay(500);
459
460   // Inicializa o TinyRTC no primeiro barramento I2C
461   rtc.begin(&Wire);
462   delay(600);
463
464   if (!rtc.begin()) {
465     Serial.println("Não foi possível encontrar o RTC.");
466     lcd.setCursor(0, 1);
467     lcd.print("TinyRTC nao encontrado");
468     Serial.flush();
469     while (1) delay(10);
470   } else {
471     lcd.setCursor(0, 1);
472     lcd.print("                    ");  // 20 espaços
473     lcd.setCursor(0, 1);
474     lcd.print("TinyRTC OK");
475   }
476
477   delay(500);
478
479   if (!rtc.isrunning()) {
480     Serial.println("O RTC parado, vamos definir o horário!");
481     limparLinhaLCD(1);  // 20 espaços
482     lcd.setCursor(0, 1);
483     lcd.print("Definindo o horario...");
484     delay(500);
485     // When time needs to be set on a new device, or after a power loss, the
486     // following line sets the RTC to the date & time this sketch was compiled
487     rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
488     // This line sets the RTC with an explicit date & time, for example to set
```

```
489     // January 21, 2014 at 3am you would call:
490     //rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
491   }
492
493   // Inicializa o sensor SCD30 no segundo barramento I2C
        ---------------------------
494   sensorSCD30.begin(I2Ctwo, SCD30_I2C_ADDR_61);
495   sensorSCD30.stopPeriodicMeasurement();
496   sensorSCD30.softReset();
497   delay(2000);
498
499   error = sensorSCD30.startPeriodicMeasurement(0);
500   if (error != NO_ERROR) {
501     Serial.print("Erro ao iniciar medição periódica: ");
502     errorToString(error, errorMessage, sizeof errorMessage);
503     Serial.println(errorMessage);
504     return;
505   } else {
506     lcd.setCursor(0, 2);
507     lcd.print("SCD30 OK");
508   }
509
510   // Inicializa o sensor SEN55 no primeiro barramento I2C
        ---------------------------
511   sensorSEN55.begin(Wire);
512
513   error = sensorSEN55.deviceReset();
514   if (error) {
515     Serial.print("Error trying to execute deviceReset(): ");
516     errorToString(error, errorMessage, sizeof errorMessage);
517     Serial.println(errorMessage);
518   }
519
520   float tempOffset = 0.0;
521   error = sensorSEN55.setTemperatureOffsetSimple(tempOffset);
522   if (error) {
523     Serial.print("Error trying to execute setTemperatureOffsetSimple(): ");
524     errorToString(error, errorMessage, 256);
525     Serial.println(errorMessage);
526   } else {
527     Serial.print("Temperature Offset set to ");
528     Serial.print(tempOffset);
529     Serial.println(" deg. Celsius (SEN54/SEN55 only");
530     lcd.setCursor(0, 3);
531     lcd.print("SEN55 OK");
532   }
533
```

```
534   // Start Measurement
535   error = sensorSEN55.startMeasurement();
536   if (error) {
537     Serial.print("Error trying to execute startMeasurement(): ");
538     errorToString(error, errorMessage, 256);
539     Serial.println(errorMessage);
540   }
541
542   delay(500);
543   lcd.clear();
544
545   // Inicializa o módulo SC
546   if (!SD.begin(5)) {
547     Serial.println("Falha ao montar cartão SD");
548     lcd.setCursor(0, 0);
549     lcd.print("SD erro");
550     return;
551   }
552
553   // Verifica se o cartão SD está inserido
554   uint8_t cardType = SD.cardType();
555
556   if (cardType == CARD_NONE) {
557     Serial.println("No SD card attached");
558     lcd.setCursor(0, 0);
559     lcd.print("SD inexistente");
560     return;
561   }
562
563   // Verifica o nome do último arquivo e cria o próximo
564   arquivo = getNextFileName(SD, "medidas", ".csv");
565   writeFile(SD, arquivo.c_str(), "data;co2;scd30_temp;scd30_umid;pm1.0;pm2.5;pm4.0;
        pm10.0;sen55_umid;sen55_temp;vocindex;noxindex\n");
566   Serial.print("Próximo nome de arquivo: ");
567   Serial.println(arquivo);
568   lcd.clear();
569   lcd.setCursor(0, 0);
570   lcd.print("Nome do arquivo");
571   lcd.setCursor(0, 1);
572   lcd.print(String(arquivo));
573
574   Serial.begin(115200);
575   delay(1000);
576
577   Serial.println("Conectando wiFi...");
578   lcd.clear();
579   lcd.setCursor(0, 0);
```

```
580   lcd.print("Conectando wiFi...");
581   WiFi.begin(ssid, password);
582
583   while (WiFi.status() != WL_CONNECTED) {
584     delay(500);
585     Serial.print(".");
586   }
587
588   Serial.println("\nWiFi conectado!");
589   lcd.clear();
590   lcd.setCursor(0, 0);
591   lcd.print("WiFi conectado!");
592   Serial.print("Endereço IP: ");
593   lcd.setCursor(0, 1);
594   lcd.print("Endereco IP: ");
595   lcd.setCursor(0, 2);
596   lcd.print(WiFi.localIP());
597   Serial.println(WiFi.localIP());
598
599   delay(5000);
600 }
601
602 void loop() {
603   // Lê o estado da chave
604   int estadoChave = digitalRead(CHAVE);
605
606   if (estadoChave == LOW) {
607     DateTime agora = rtc.now();
608     unsigned long tempoAtual = agora.unixtime();  // Tempo atual em segundos
609     if (tempoAtual - ultimoTempoExecutado >= intervalo) {
610       ultimoTempoExecutado = tempoAtual;
611
612       if (!client.connected()) {
613         mqtt_reconnect();
614       }
615       client.loop();
616
617       // Obtém as leituras e atualiza as variáveis
618       if (lerSCD30(dadosSCD30) && lerSEN55(dadosSEN55)) {
619         // pisca o led indicando que entrou no loop
620         acendeLed();
621         // atualiza os dados no terminal
622         atualizaTerminal(dadosSCD30, dadosSEN55);
623
624         // publica dados no broker mqtt
625         publicarMQTT(dadosSCD30, dadosSEN55);
626
```

```
627         // atualiza os dados no arquivo atual
628         atualizaArquivo(arquivo, dadosSCD30, dadosSEN55, obterDataHora(true));
629         // atualiza os dados no LCD
630         atualizaLCD(dadosSCD30, dadosSEN55);
631         // limpa o LCD e posiciona o cursos
632         lcd.clear();
633         lcd.setCursor(0, 0);
634         // exibe informações do arquivo gravado
635         lcd.print("Dados gravados:");
636         lcd.setCursor(0, 1);
637         lcd.print(String(arquivo));
638         delay(intervaloPorTela);
639       } else {
640         lcd.clear();
641         Serial.println("Falha na leitura dos sensores.");
642         lcd.setCursor(0, 0);
643         lcd.print("Erro na leitura!");
644         delay(intervaloPorTela);
645       }
646       controleLCD = 0;
647     } else {
648       // Limpa o LCD e evita de piscar ao limpar somente uma vez
649       if (controleLCD == 0) {
650         lcd.clear();
651         controleLCD = 1;
652       }
653       lcd.setCursor(0, 0);
654       lcd.print("Proxima leitura:");
655       float tempoProximaLeitura = intervalo - (tempoAtual - ultimoTempoExecutado);
656       lcd.setCursor(0, 1);
657       lcd.print(String(tempoProximaLeitura, 0) + " segundos");
658     }
659     delay(500);
660   } else {
661     // Limpa o LCD e evita de piscar ao limpar somente uma vez
662     if (controleLCD == 0) {
663       lcd.clear();
664       controleLCD = 1;
665     }
666     String horaAtual = obterDataHora(false);
667     Serial.println("Leitura desligada");
668     lcd.setCursor(0, 0);
669     lcd.print("IFSC Campus Sao Jose");
670     lcd.setCursor(0, 1);
671     lcd.print("Leitura desligada");
672     lcd.setCursor(0, 2);
673     lcd.print(String(horaAtual));
```

```
674     delay(500);
675   }
676 }
```

ANEXO B – ARQUIVO DE CONFIGURAÇÃO DO BROKER MQTT MOSQUITTO

Este anexo apresenta o arquivo de configuração do *broker* MQTT *Eclipse Mosquitto* utilizado no sistema de monitoramento. O arquivo `mosquitto.conf` define os parâmetros essenciais de funcionamento do *broker*, incluindo as configurações de segurança, autenticação, persistência de dados e registro de eventos.

As principais configurações implementadas incluem:

- **Autenticação obrigatória:** desabilitação de conexões anônimas (`allow_anonymous false`) para garantir segurança no acesso ao *broker*;
- **Arquivo de senhas:** definição do arquivo contendo as credenciais de usuários autorizados (`password_file`);
- **Persistência de mensagens:** armazenamento de mensagens e assinaturas no diretório especificado, permitindo recuperação após reinicializações;
- **Registro de eventos:** configuração do arquivo de *log* para monitoramento e diagnóstico do funcionamento do *broker*;
- **Porta de escuta:** definição da porta padrão 1883 para comunicação MQTT.

Código B.1 – Arquivo de configuração do Mosquitto (mosquitto.conf)

```
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

per_listener_settings true
pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d
allow_anonymous false
listener 1883
password_file /etc/mosquitto/pwfile
```

O *broker* também requer um arquivo de senhas para autenticação dos usuários. O arquivo `pwfile` contém as credenciais dos usuários autorizados a publicar e assinar mensagens, armazenadas de forma criptografada utilizando o algoritmo PBKDF2 com *hash* SHA-512. O formato segue o padrão `usuário:hash`, onde o *hash* é gerado pela ferramenta `mosquitto_passwd`, que aplica técnicas de *salting* e múltiplas iterações para garantir segurança contra ataques de força bruta. Para autenticação: Usuário `admin` com a senha `paulvandyk11` para autenticação no *broker*.

Código B.2 – Arquivo de senhas do Mosquitto (pwfile)

```
1 admin:$7$101$pcVMBqxsZ7PXxnfJ$Dfol8rN4F6Mp6AL3+
      d5lGYvJ7IDUa4SqvMHBxRoRr85DidSDzHEwbVEFTee92rc2jiHSTCrPOa4O4JXlQSP9AA==
```

ANEXO C – ESQUEMA DO BANCO DE DADOS MYSQL

Este anexo apresenta o esquema completo do banco de dados MariaDB utilizado para armazenamento dos dados de monitoramento da qualidade do ar. O banco de dados `sens` contém a tabela `LeituraSensor`, que registra todas as leituras dos sensores ambientais coletadas pelo sistema.

A tabela foi projetada para armazenar os dados dos sensores SCD30 e SEN55, incluindo concentração de dióxido de carbono (CO_2), temperatura, umidade relativa, material particulado em diferentes frações ($PM_{1.0}$, $PM_{2.5}$, $PM_{4.0}$ e $PM_{10.0}$), índice de compostos orgânicos voláteis (VOC) e índice de óxidos de nitrogênio (NOx). Cada registro é identificado por uma chave primária auto-incremental e inclui o *timestamp* de aquisição dos dados.

O banco de dados está configurado no Raspberry Pi do sistema e pode ser acessado através do seguinte comando:

```
mysql -u root -ppaulvandyk11
```

Após a conexão, selecionar o banco de dados `sens` com o comando:

```
USE sens;
```

A tabela `LeituraSensor` possui indexação na coluna `data_hora` para otimização de consultas temporais, permitindo análises históricas eficientes dos dados de qualidade do ar.

Código C.1 – Esquema de criação do banco de dados e tabela LeituraSensor

```
1  -- Criação do banco de dados
2  CREATE DATABASE IF NOT EXISTS sens;
3
4  -- Selecionar o banco de dados
5  USE sens;
6
7  -- Criação da tabela LeituraSensor
8  CREATE TABLE `LeituraSensor` (
9    `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
10   `co2` int(11) NOT NULL DEFAULT 0,
11   `temperatura` decimal(5,2) NOT NULL DEFAULT 0.00,
12   `umidade` decimal(5,2) NOT NULL DEFAULT 0.00,
13   `pm1_0` decimal(6,2) NOT NULL DEFAULT 0.00,
14   `pm2_5` decimal(6,2) NOT NULL DEFAULT 0.00,
```

```
15   `pm4_0` decimal(6,2) NOT NULL DEFAULT 0.00,
16   `pm10_0` decimal(6,2) NOT NULL DEFAULT 0.00,
17   `vocIndex` int(11) DEFAULT 0,
18   `noxIndex` int(11) DEFAULT 0,
19   `data_hora` datetime NOT NULL DEFAULT current_timestamp(),
20   PRIMARY KEY (`id`),
21   KEY `idx_data_hora` (`data_hora`)
22 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```