INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

# Sistema Inteligente de Monitorização de Multiparâmetros em Espaços Interiores

*Elizio Cardoso Mendes (46814)*

Dissertação apresentada à Escola Superior de Tecnologia e Gestão de Bragança para obtenção do Grau de Mestre em Engenharia Industrial

Orientado por:

Prof. Dr. José Barbosa

Bragança, Portugal

2023

# Sistema Inteligente de Monitorização de Multiparâmetros em Espaços Interiores

*Elizio Cardoso Mendes (46814)*

Prof. Dr. José Barbosa

Bragança, Portugal

2023

# DEDICATÓRIA

Dedico este projeto/dissertação em especial aos meus pais, pela constante inspiração e apoio ao longo desta jornada académica. Ao meu orientador, pela orientação valiosa, paciência e incentivo. A todos os amigos e familiares que, de uma forma ou de outra, ajudaram-me e que sempre acreditaram em mim durante estes anos de formação, este trabalho é também de vocês. Obrigado por tornarem este sonho possível.

# AGRADECIMENTOS

Por intermédio das seguintes palavras pretendo demonstrar a minha profunda gratidão a todos aqueles que tornaram possível a realização desta dissertação.

Começo por agradecer aos meus familiares, em especial aos meus pais pelo apoio incondicional que sempre me deram e continuam a dar, pois sem eles não tinha como chegar onde estou, e aos meus irmãos por estarem sempre presentes.

Ao meu orientador, Prof. Dr. José Barbosa, pela sua orientação excecional, sabedoria e dedicação incansável ao longo deste percurso académico, pois foram determinantes para conclusão desta tese.

Aos meus colegas de mestrado, que partilharam ideias e experiências valiosas, aos amigos feitos no Instituto Politécnico de Bragança, desde o primeiro ano até o último pela colaboração e pelos grandes momentos de convivência, pois foram essenciais para o encerramento deste ciclo.

Por fim, quero agradecer à Escola Superior de Tecnologia e Gestão de Bragança por proporcionar as condições necessárias para a realização deste mestrado.

Este trabalho é resultado de um esforço coletivo e é com humildade que dedico esta dissertação a todos os que acreditaram em mim. Obrigado por fazerem parte desta jornada.

*“Não creio que haja uma emoção mais intensa para um inventor do que ver suas criações funcionando. Essa emoção faz você esquecer de comer, de dormir, de tudo.”*

**Nikola Tesla**

*"Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível."*

**Charles Chaplin**

# RESUMO

O conforto e o bem-estar em espaços fechados têm um impacto direto na saúde e, atualmente, as pessoas passam cerca de 90% dos seus tempos em espaço fechados, onde por vezes a poluição atinge níveis mais elevados do que os registados ao ar livre.

A utilização de tecnologias baseadas em IoT permite automatizar e agilizar a troca de informação entre dispositivos constituintes de um sistema. Uma das aplicações de IoT consiste na sensorização, com o objetivo de capturar e transmitir dados para um conjunto de aplicações e utilizadores conetados em rede.

Neste contexto, pretende-se desenvolver um sistema composto por hardware e software, que permita ao utilizador monitorizar multiparâmetros num espaço interior e ser alertado para potenciais situações de degradação ou de má qualidade.

Esse sistema conta com uma placa de desenvolvimento ESP32 da Wemos D1 R32 e um *single board computer* Raspberry Pi modelo 3B (que será o servidor) juntamente com sensores que medem grandezas como temperatura, pressão, humidade, VOC, luminosidade, ruído, PM2.5 e PM10. Os valores medidos pelos sensores são enviados, através do protocolo MQTT, para o broker Mosquitto em determinados tópicos. Um aplicativo em Node-RED recebe as leituras dos sensores utilizando o MQTT e, esses valores são enviados e guardados numa base de dados (em InfluxDB) e, posteriormente são mostrados num *dashboard* em tempo real.

**Palavras-chave:** Qualidade do Ar, IoT, Sensorização, ESP32, Raspberry Pi, Node-RED.

## ABSTRACT

Indoor comfort and well-being have a direct impact on health. Currently people spend about 90% of their time indoors, where air pollution can reach higher levels than outdoors.

The use of IoT technologies enables the automation and streamlining of information exchange between the individual devices of a system. The IoT application consists of sensor objects with the aim of collecting and transmitting data to a set of applications and users connected in a network.

In this context, a system of hardware and software will be developed that will allow the user to monitor several parameters in an indoor space and be warned of potential deterioration or poor quality.

This system consists of an ESP32 microcontroller from Wemos D1 R32 and a Raspberry Pi model 3B microcomputer (which will serve as a server), as well as sensors that measure variables such as temperature, pressure, humidity, VOC, light, noise, PM2.5 and PM10. The values measured by the sensors are sent by the ESP32 via the MQTT protocol to the Mosquitto broker in specific topics. An application in Node- RED controls the ESP32 outputs and receives the sensor readings via MQTT. When received, these values are sent and stored in a database (in InfluxDB) and later displayed on a dashboard in real time.



**Keywords:** Air Quality, IoT, Sensing, ESP32, Raspberry Pi, Node-RED.

# CONTEÚDO

# INDICE DE FIGURAS

# INDICE DE TABELAS

# ABREVIATURAS E SÍMBOLOS

| | |
|---|---|
| ANSI | *American National Standard Institute* |
| ASCII | *American Standard Code for Information Interchange* |
| AVAC | Aquecimento, Ventilação e Ar-Condicionado |
| CI | Circuito Integrado |
| CNC | *Computer Numerical Control* |
| COM | *Communication Port* |
| CPU | *Central Processing Unit* |
| dB | Decibel |
| DHCP | *Dynamic Host Configuration Protocol* |
| EEPROM | *Electrically-Erasable Programmable Read-Only Memory* |
| EMC | E*lectromagnetic Compatibility* |
| EPA | *Environmental Protection Agency* |
| GPIO | *General Purpose Input/Output* |
| HDMI | *High-Definition Multimedia Interface* |
| HTTP | *Hypertext Transfer Protocol* |
| HTTPS | *Hyper Text Transfer Protocol Secure* |
| I2C | *Inter-Integrated Circuit* |
| IDAD | Instituto do Ambiente e Desenvolvimento |
| IDE | *Integrated Development Environment* |
| IoT | *Internet of Things* |
| LDR | *Light Dependent Resistor* |
| LED | *Light Emitting Diode* |
| MOX | Metal Oxide |
| MQTT | *Message Queuing Telemetry Transport* |
| OMS | Organização Mundial de Saúde |
| PC | *Personal Computer* |
| PWM | *Pulse-Width Modulation* |
| QAI | Qualidade de Ar Interior |
| RAM | *Random Access Memory* |

| | |
|---|---|
| RPI | Raspberry Pi |
| RPI 3B | Raspberry Pi 3 modelo B |
| RRAE | Regulamento dos Requisitos Acústicos dos Edifícios |
| SBC | *Single-Board Computer* |
| SCL | *Serial Clock* |
| SD | *Secure Digital* |
| SDA | *Serial Data* |
| SED | Síndrome do Edifício Doente |
| SGBD | Sistema de Gerenciamento de Base de Dados |
| SPI | *Serial Peripheral Interface* |
| SQL | *Structured Query Language* |
| SSH | *Secure Shell* |
| TCP | *Transmission Control Protocol* |
| TIC | Tecnologias da Informação e Comunicação |
| TSMC | *Taiwan Semiconductor Manufacturing Company* |
| UART | *Universal Asynchronous Receiver/Transmitter* |
| UI | *User Interface* |
| URL | *Uniform Resource Locator* |
| USB | *Universal Serial Bus* |
| VGA | *Video Graphics Array* |
| VOC | *Volatile organic compounds* |
| WEP | *Wired Equivalent Privacy* |
| Wi-Fi | *Wireless Fidelity* |
| WLAN | *Wireless Local Area Network* |
| WPA2 | *Wi-Fi Protected Access 2* |

# I. INTRODUÇÃO

## 1.1 Contextualização

O ar que respiramos é um recurso que para além de essencial é inesgotável, mas imensamente vulnerável face a elementos que podem contaminá-lo.

Para os seres vivos o ar é considerado um elemento crítico. Pessoas, por exemplo, podem sobreviver por duas semanas sem comida e por dois dias sem água, mas sem ar somente podem sobreviver por dois minutos. Nos países mais desenvolvidos, as pessoas passam em média 88-90% das suas vidas em espaço fechados (em casa, no trabalho ou na escola). A qualidade do ar que se respira em espaços fechados também tem um impacto direto na saúde. Deste modo, destaca-se a importância da qualidade de ar que é consumida [1].

Segundo a Organização Mundial de Saúde (OMS), a poluição do ar é considerada como um dos principais problemas ambientais e de saúde pública. Estima-se que nove em cada dez pessoas em todo o mundo respiram ar poluído. Dados mostram que aproximadamente metade da população mundial, possam ter problemas de saúde devido à má qualidade do ar [7].

Muitos dos poluentes são de difícil identificação pelos ocupantes por serem inodoros e incolores. Apesar da existência de legislações que exigem o controle da qualidade do ar nos ambientes internos, em muitos locais há indícios de que essa prática não ocorra.

A utilização de tecnologias baseadas em *Internet of Things* (IoT) permite automatizar e agilizar a troca de informação entre dispositivos constituintes de um sistema. No caso da presente proposta, é pretendido o desenvolvimento de um sistema, composto por hardware e software, que permita ao utilizador monitorizar a qualidade do ar interior e ser alertado para potenciais situações de degradação ou de má qualidade.

Uma vez definido, o sistema deverá permitir a troca de informação entre um elemento de sensorização (capaz de medir os parâmetros que permitirão aferir a qualidade

do ar onde está inserido) e o elemento de monitorização (i.e., a componente informática que permita a visualização e monitorização) em tempo real.

## 1.2 Objetivos

O objetivo deste projeto consiste no estudo e desenvolvimento de uma solução que visa a monitorização de multiparâmetros em espaços fechados, nomeadamente salas de aulas ou escritórios. Desta forma, o sistema a desenvolver deverá contemplar o desenvolvimento de uma aplicação móvel, um elemento sensorizador e de um sistema de apoio à decisão através da geração de alertas ao utilizador.

### 1.2.1 Objetivos Específicos

Neste contexto o projeto desenvolvido pretende:

- Familiarizar com o problema de sensorização e dos parâmetros de aferição da qualidade do ar;
- Desenvolver um sistema de sensorização;
- Desenvolver um sistema de monitorização;
- Desenvolver um sistema de alertas;
- Desenvolver um sistema de base de dados;
- Implementar protocolos de envio e troca de informação entre os diversos elementos;
- Testar e validar o protótipo.

## 1.3 Metodologia

Buscando analisar a temática proposta, como em qualquer projeto, começou-se por fazer um levantamento do estado da arte a fim de identificar as tecnologias e abordagens existentes, ou seja, uma pesquisa baseada em livros, artigos científicos, algumas dissertações e teses obtidas na internet, tentando buscar sempre em fontes seguras, para dar sustentabilidade à pesquisa e para ajudar a compreender os principais conceitos.

Inicialmente pretende-se realizar um breve estudo de todos os componentes eletrónicos do projeto, desde sensores até a placa de desenvolvimento ESP32 e o

Raspberry Pi e, de seguida, falar resumidamente das ferramentas e das tecnologias da informação e comunicação utilizadas.

Obtendo as informações necessárias, passa-se pela parte da implementação do projeto em si. Consiste no desenvolvimento do código para converter dados dos sensores em informações, criar uma base de dados onde essas informações serão guardadas e, por último, criar um *dashboard* para monitorização.

Para finalizar pretende-se elaborar o protótipo em PCI utilizando o software KiCad, implementando o sistema e elaborar testes para uma melhor avaliação das funcionalidades do mesmo, fazer os reajustes necessários e, no fim, realizar a conclusão do relatório final.

## 1.4 Estrutura da Dissertação

O documento está organizado em seis capítulos:

No primeiro capítulo fez-se o enquadramento geral do referente tema do projeto. No segundo, busca introduzir o leitor sobre a temática da qualidade do ar em espaços fechados mais conhecido por QAI (Qualidade do Ar Interior) apresentando alguns problemas à escala global e, parâmetros relevantes na determinação da QAI. Destina-se também a informar sobre os termos legais e normativos, procurando caracterizar a nível nacional (Portugal), as principais diretrizes e limiares de proteção para os parâmetros avaliados neste estudo.

No capítulo três, fez-se um estudo de forma resumida dos hardwares (placa de desenvolvimento, microcomputador e sensores) e softwares utilizadas ao longo do projeto.

Os capítulos quatro e cinco destinam-se à parte prática do projeto. Relatam todo o processo da implementação do projeto em si e, os resultados dos testes realizados com o protótipo.

Por fim, e não menos importante, o capítulo seis apresenta as considerações finais e propostas para trabalhos futuros.

# II. QUALIDADE DO AR EM ESPAÇOS INTERIORES

Acredita-se que uma boa parte das pessoas estão cientes de que a poluição do ar externo, ou seja, na zona urbana, pode prejudicar o meio ambiente, e a sua saúde e, em geral, toda a fauna e a flora. Mas podem não saber que a poluição do ar num espaço interior também pode ter efeitos significativos.

Mais da metade da ingestão do corpo durante a vida é o ar inalado em espaços fechados. Assim, a maioria das doenças relacionadas a exposições ambientais decorre da exposição ao ar interno e há evidências crescentes de que a exposição do ar em espaços fechados é a causa de morbidade e mortalidade excessivas [2] [3].

Até final dos anos 1960, a atenção à qualidade do ar concentrava-se principalmente no exterior, visto que a poluição do ar externo era considerada responsável por muitos efeitos adversos à saúde. No início dos anos 1970, cientistas começaram a investigar as causas das reclamações dentro dos ambientes de trabalho [1] [4]. Nessa altura, questões que envolvem efeitos na saúde relacionadas com a qualidade de ar em edifícios começaram a ser debatidas, como a Síndrome do Edifício Doente (SED), em que determinadas doenças alergénicas e crónicas, foram associadas a condições de má qualidade do ar [2].

A Agência de Proteção Ambiental dos EUA (EPA – U.S. *Environmental Protection Agency*) estuda sobre a exposição humana a poluentes atmosféricos e, indicaram que os níveis de muitos poluentes no ar interno podem ser de duas a cinco vezes e, ocasionalmente, mais de 100 vezes mais altos do que os níveis externos, e foram classificados entre os 5 maiores riscos ambientais para o público [1] [4].

Segundo EPA, 2008, a SED pode ser definida como sendo “uma situação na qual os ocupantes apresentam sintomas agudos sem uma explicação óbvia e sem a possibilidade de constatação de uma determinada etiologia, sendo, portanto, desconhecida”. Esse termo também é usado para descrever casos em que pelo menos 20% dos ocupantes de um edifício apresentam sintomas agudos e desconforto, e que geralmente estão relacionados com o tempo que estes ocupantes passam dentro dos edifícios [8].

As principais causas que influenciam de forma negativa a QAI estão relacionados com ventilação inadequada, presença de compostos orgânicos voláteis (VOC), alto índice de humidade, temperaturas extremas, deficiência na filtração do ar, falta de manutenção e limpeza das instalações, contaminação dos sistemas de AVAC e contaminantes gerados pelo metabolismo humano [9].

## 2.1 Índice de Qualidade do Ar

O Índice de Qualidade do Ar (IQA) é uma medida que avalia a qualidade do ar com base nos níveis de poluentes presentes, em que varia de 0 a 500, sendo que quanto maior o valor, pior é a qualidade do ar. O cálculo do IQA é baseado em um conjunto de fórmulas matemáticas que levam em consideração as concentrações de cada um desses poluentes na atmosfera. O IQA é comumente utilizado para fornecer informações sobre a qualidade do ar em uma determinada região, ajudando a alertar as pessoas sobre os riscos à saúde associados à exposição a níveis elevados de poluentes [37].

O IQA pode ser utilizado como uma ferramenta para avaliar a QAI, embora tenha sido originalmente desenvolvido para avaliar a qualidade do ar externo. Os mesmos poluentes que são avaliados pelo IQA podem estar presentes no ar interior, tornando-o uma medida útil para avaliar a qualidade do ar em ambientes fechados [38].

No entanto, o cálculo do IQA pode não levar em consideração alguns poluentes que são específicos do ar interior, como VOC emitidos por produtos químicos domésticos e de construção, além de poeira, fumo de tabaco, ácaros e mofo, entre outros [38].

Por isso, é importante que outras medidas de QAI também sejam consideradas, como a medição de níveis de VOC específicos e outras, como a humidade relativa, a temperatura e a taxa de renovação de ar [38].

Em resumo, o IQA pode ser uma ferramenta útil para avaliar a qualidade do ar interior, mas outras medidas específicas de qualidade do ar interior também devem ser consideradas para uma avaliação completa e precisa da qualidade do ar em ambientes fechados.

## 2.2 Parâmetros Relevantes na Determinação da QAI

A escolha dos parâmetros adequados depende do ambiente em questão e das preocupações específicas com a saúde e o conforto dos ocupantes. A QAI é geralmente determinada através da identificação de parâmetros físico-químicos, operativos e microbiológicos [39].

Neste tópico encontra-se uma breve descrição de alguns dos muitos parâmetros que afetam a qualidade do ar interior, das principais fontes de emissão e seus efeitos no conforto e na saúde humana.

### 2.2.1 Parâmetros físico-químicos

Existem vários parâmetros físico-químicos que podem ser medidos para determinar a qualidade do ar interior, incluindo:

**A. Concentração de gases**

A concentração de gases no ar interior pode ser medida para determinar a qualidade do ar. Alguns dos gases mais comuns que são medidos incluem dióxido de carbono ($CO_2$), monóxido de carbono (CO), ozono ($O_3$), dióxido de nitrogênio ($NO_2$), dióxido de enxofre ($SO_2$) e gases radão. A concentração desses gases pode afetar a saúde e o conforto dos ocupantes [39].

**B. Compostos orgânicos voláteis (VOC)**

Os VOC (sigla em inglês, *volatile organic compounds*) são compostos químicos orgânicos que podem ser liberados por produtos de limpeza, materiais de construção e mobiliário, entre outros. A exposição a níveis elevados de VOC pode levar a problemas de saúde, como irritação nos olhos, nariz e garganta, além de dores de cabeça, náuseas e tonturas [39].

**C. Material particulado (PM10, PM2.5)**

O material particulado (PM) é composto de partículas sólidas ou líquidas suspensas no ar. O PM é classificado em PM2,5 e PM10, que se referem às partículas com um diâmetro menor que 2,5 e 10 micrômetros, respetivamente. A exposição a níveis elevados de PM pode levar a problemas respiratórios, cardiovasculares e até cancro [39].

**D. Nível de ruído**

O nível de ruído em um ambiente interior pode afetar o conforto dos ocupantes e sua capacidade de se comunicar e realizar determinadas tarefas. O nível de ruído pode ser medido em decibéis (dB) para determinar se ele está dentro de níveis seguros.

### 2.2.2 Parâmetros operativos

Além dos parâmetros físico-químicos, também existem parâmetros operativos que podem ser medidos para determinar a qualidade do ar interior. Esses parâmetros incluem:

**A. Humidade relativa**

A humidade do ar é importante porque pode afetar a saúde e o conforto dos ocupantes, além de promover o crescimento de mofo e bactérias. A norma ISO 7730 é uma norma internacional que aborda a questão da humidade relativa no contexto da QAI. Segundo a norma ISO 7730, em termos de conforto térmico, a humidade relativa ideal geralmente fica na faixa de 30% a 60% [44].

**B. Temperatura**

A temperatura do ar é um parâmetro operativo importante porque pode afetar o conforto térmico dos ocupantes, além de afetar a eficiência energética propriamente dita. A norma ISO 7730 também aborda a temperatura no contexto de conforto térmico em ambientes internos. Segundo essa norma, uma temperatura do ar entre 20°C e 26°C é frequentemente considerada uma faixa de conforto térmico para a maioria das pessoas [44].

**C. Pressão do ar**

A pressão do ar é a diferença de pressão entre o interior e o exterior de um edifício ou sala. A pressão do ar é importante porque pode afetar a entrada de ar externo e a saída de ar interior.

**D. Iluminação**

A iluminação ou luminosidade é um parâmetro operativo de grande relevância pois, pode afetar o conforto visual e a produtividade dos ocupantes. A norma EN 12464-1, da série EN 12464, é uma norma europeia que borda os requisitos de iluminação em

locais de trabalho internos. Segundo a norma EN 12464-1, em espaços de escritório, é comum ter níveis de iluminância entre 300 a 500 lux e, para salas de conferência e reuniões, é recomendado ter iluminância entre 300 a 750 lux [45].

### 2.2.3 Parâmetros microbiológicos

Além dos parâmetros físico-químicos e operativos, os parâmetros microbiológicos também são de grande importância na determinação da QAI. Isso ocorre porque o ar pode conter microrganismos como bactérias, fungos, vírus e outros patógenos que podem afetar a saúde dos ocupantes [39].

É importante monitorar regularmente os parâmetros microbiológicos da QAI para garantir que o ar interior seja seguro e saudável para os ocupantes. Alguns exemplos de parâmetros microbiológicos da QAI incluem:

**A. Bactérias**

Bactérias são organismos unicelulares procariontes que podem ser encontradas no ar, água, solo e inclusive nos nossos corpos. Muitas podem ser prejudiciais à saúde do homem, sendo agentes causadores de diversas doenças (patógenas) [39].

As bactérias patogênicas são aquelas que podem causar doenças em seres humanos. Exemplos incluem *Legionella*, *Salmonella* e *Escherichia coli* [39].

**B. Fungos**

Os fungos são microrganismos eucariontes heterotróficos e apresentam ampla distribuição de tamanho, com diâmetros que variam conforme o género [39]. Os fungos patogênicos são aqueles que podem causar doenças em seres humanos. Exemplos incluem *Aspergillus fumigatus* e *Cryptococcus neoformans* [39].

**C. Vírus**

Os vírus podem ser transmitidos pelo ar e podem causar doenças respiratórias em seres humanos. Exemplos incluem o vírus da gripe e o coronavírus (COVID-19).

## 2.3 Legislação/Regulamentação Portuguesa

Ao longo do tempo, foram estabelecidas normas e legislações aplicadas na QAI a nível da regulamentação portuguesa. Essas, estabelecem os limiares de proteção para os principais poluentes do ar interior em ambientes ocupacionais de modo geral.

No ano de 2006, a temática qualidade do ar interior passou a ser abordada na legislação nacional com a publicação do **Decreto-Lei nº 78/2006 de 4 de abril**, da qual visa que o Estado assegura a melhoria do desempenho energético e da qualidade do ar interior dos edifícios através do Sistema Nacional de Certificação Energética e da Qualidade do Ar Interior nos Edifícios. Mais tarde, surgiu um novo decreto conhecido como Sistema Nacional de Certificação Energética e do Ar Interior (**Decreto-Lei nº 79/2006 de 4 de abril**), revogando o anterior [11].

Para fazer face ao elevado peso que o setor dos edifícios representa no consumo energético, têm vindo a ser adotadas regulamentações que visam promover a melhoria do desempenho energético e das condições de conforto dos edifícios [12].

É neste contexto que surge a Diretiva (EU) n.º 2018/844 de 30 de maio de 2018 (que altera a Diretiva 2010/31/UE relativa ao desempenho energético dos edifícios e a Diretiva 2012/27/UE sobre a eficiência energética), transposta para a legislação nacional através do **Decreto-Lei n.º 101-D/2020 de 7 de dezembro** [12] [13].

O **Decreto-Lei n.º 101-D/2020 de 7 de dezembro**, determina que todos os edifícios de comércio e serviços em funcionamento estão sujeitos a requisitos relacionados com a qualidade do ar interior, mediante o cumprimento de limiares de proteção e condições de referência [12] [13].

A **Portaria n.º 138-G/2021 de 1 de julho**, estabelece os requisitos para a avaliação da qualidade do ar interior nos edifícios de comércio e serviços, incluindo os limiares de proteção, condições de referência e critérios de conformidade, e a respetiva metodologia para a medição dos poluentes e para a fiscalização do cumprimento das normas aprovadas [12] [13].

O **Despacho n.º 1618/2022 de 9 de fevereiro**, determina os procedimentos de registo das obrigações previstas no Decreto-Lei n.º 101-D/2020, e o regime de avaliação simplificada anual de requisitos relacionados com a qualidade do ar interior [12] [13].

O IDAD (Instituto do Ambiente e Desenvolvimento), é uma associação científica e técnica, sem fins lucrativos, que atua ao nível do apoio integrado às necessidades ambientais do mundo das empresas e das organizações [14]. O IDAD realiza a caracterização da qualidade do ar em espaços interiores, permitindo a identificação de fontes poluidoras e definição de estratégias de controlo [15].

O regime de avaliação da QAI é enquadrado na legislação nacional através do Decreto-Lei n.º 101-D/2020, Portaria n.º 138-G/2021 e Despacho n.º 1618/2022 [15].

Como exemplo de parâmetros avaliados têm-se: partículas em suspensão (PM10, PM2.5), dióxido de carbono ($CO_2$), monóxido de carbono (CO), compostos orgânicos voláteis (VOC), formaldeído ($CH_2O$), bactérias, fungos, radão, temperatura, humidade relativa, velocidade do ar, caudais de ar, taxa de renovação [15].

A Portaria n.º 138-G/2021 de 1 de julho, estabelece os requisitos para a avaliação da qualidade do ar interior nos edifícios de comércio e serviços, incluindo os limiares de proteção e margem de tolerância para os poluentes físico-químicos (tabela 1), condições de referência e critérios de conformidade, e a respetiva metodologia para a medição dos poluentes e para a fiscalização do cumprimento das normas aprovadas [16].

***Tabela 1:*** *Limiar de proteção e margem de tolerância para os poluentes físico-químicos [16].*

| Poluentes | Unidade | Condições de referência | Margem de tolerância [%] |
|---|---|---|---|
| Partículas em suspensão (PM10) | $\mu g/m^3$ | 50 | 100 |
| Partículas em suspensão (PM2,5) | $\mu g/m^3$ | 25 | 100 |
| Compostos Orgânicos Voláteis (VOC) | $\mu g/m^3$ | 600 | 100 |
| Monóxido de Carbono (CO) | $\mu g/m^3$ | 10 | -- |
| Formaldeído (CH2O) | $\mu g/m^3$ | 100 | -- |
| Dióxido de Carbono ($CO_2$) | $\mu g/m^3$ | 2250 | 30 |

As concentrações em $\mu g/m^3$ e $mg/m^3$ referem -se à temperatura de 20ºC e à pressão de 1 atm (101,325 kPa). Os limiares de proteção indicados dizem respeito a uma média de 8 horas, por correspondência ao cenário de maior ocupação possível [16].

O **Decreto-Lei n.º 243/86 de 20 de agosto**, ainda em vigor, aprova o Regulamento Geral de Higiene e Segurança do Trabalho nos Estabelecimentos Comerciais, de Escritório e Serviços. O Artigo 11.º do referente Decreto-Lei visa que, a temperatura dos locais de trabalho deve, na medida do possível, oscilar entre 18ºC e 22ºC, salvo em determinadas condições climatéricas, em que poderá atingir os 25ºC. A humidade da atmosfera de trabalho deve oscilar entre 50% e 70% [18].

De acordo com o Artigo 14.º do Decreto-Lei 243/86 de 20 de agosto, a iluminação nos locais de trabalho deve ser adequada aos requisitos de iluminação das tarefas a executar e obedecer aos valores insertos no Regulamento Tipo de Segurança nos Estabelecimentos Industriais da Organização Internacional do Trabalho, com necessárias adaptações, enquanto não forem publicadas normas portuguesas [18].

De acordo com a norma e legislação portuguesa, em ambientes de trabalho geral, a iluminação deve fornecer um nível mínimo de 300 lux no plano de trabalho ou superfície onde as tarefas são realizadas [18].

O nível de ruído ideal para a QAI em Portugal depende do tipo de ambiente e do uso previsto do espaço. Em geral, é recomendado que os níveis de ruído sejam mantidos abaixo de certos limites para garantir o conforto e a saúde dos ocupantes.

O **Decreto-Lei n.º 9/2007 de 17 de janeiro** estabelece o Regulamento dos Requisitos Acústicos dos Edifícios (RRAE) em Portugal. Este Decreto-Lei estabelece os critérios e procedimentos para avaliação do desempenho acústico dos edifícios, incluindo os limites de ruído permitidos em diferentes tipos de ambientes internos. O RRAE define os limites de ruído em relação ao ambiente exterior e interior dos edifícios, assim como os procedimentos para a realização de medições de ruído e as condições de ensaio [25].

No caso do ambiente interior, o RRAE define que o ruído não deve exceder 30 dB durante o dia e 25 dB durante a noite para áreas sensíveis, como hospitais, escolas, creches e habitações. Para outros tipos de ambientes internos, como escritórios, salas de aula, salas de conferência, entre outros, o nível de ruído recomendado é de até 50 dB durante o dia [25].

# III. MATERIAL E MÉTODOS

Este capítulo é destinado ao estudo dos *hardwares* e *softwares* utilizados ao longo do projeto e, serão apresentados de forma separada, para uma análise e uma descrição técnica dos mesmos.

Os dois primeiros componentes estudados podem ser considerados o "cérebro do projeto". Apresenta-se o microcontrolador e o microcomputador, responsáveis por receber as informações dos sensores e processar essas informações. Para finalizar este capítulo, são estudados todos os sensores utilizados no sistema.

## 3.1 Estudo do Microcontrolador

A evolução da microeletrónica levou o homem à criação de dispositivos cada vez menores e mais repletos de funções programáveis, ou seja, capazes de entender códigos carregados de conjunto de instruções. Os dispositivos que descodificam esses códigos são denominados de microprocessadores.

Um microprocessador incorpora num único circuito integrado (CI) as funções de uma Unidade Central de Processamento (sigla em inglês CPU). São dispositivos programáveis que recebem dados digitais, capazes de descodificar instruções, processá-las e fornecer resultados com saída, ativando registos ou outros mecanismos. Além disso, pode-se ler e escrever em memórias e realizar operações lógicas e aritméticas.

Portanto, pode-se definir os microcontroladores como sendo equipamentos que usam um microprocessador para aplicações diversas na engenharia. Por outras palavras, microcontroladores são circuitos digitais programáveis, compostos por um microprocessador, memórias e periféricos de entrada e saída, acoplados em um único CI.

Os microcontroladores podem ser usados para controlar os mais diversos equipamentos, desde controlo de LED até motores elétricos pois, contam com equipamentos periféricos capazes de gerar conversões analógicos digitais (A/D) e modulação de largura de pulso (PWM).

### 3.1.1 ESP32 D1 R32

O ESP32, lançado a 6 de setembro de 2016, é uma evolução de uma série de placas do fabricante *Espressif Systems*, que se tornou um termo abrangente para uma variedade de placas e *chips* de desenvolvimento compatíveis com *Wi-Fi*. As placas ESP32 são uma das plataformas mais utilizadas para projetos de IoT [22] [23].

O ESP32 é um *chip* que combina os protocolos de comunicação de *Wi-Fi* e *Bluetooth*, projetado com a tecnologia TSMC de baixa potência. Foi desenvolvido para obter o melhor desempenho de potência, mostrando robustez, versatilidade e confiabilidade em uma ampla variedade de aplicações e cenários de energia [22] [23]. Dentre os diversos modelos do ESP32 existentes, para o projeto escolheu-se o modelo *ESP32 D1 R32*, por cumprir com todos os requisitos e pela sua disponibilidade.

O modelo *Wemos D1 R32*, assim como o Arduíno Uno, é uma placa de desenvolvimento que contém um microcontrolador baseada no *chip ESP32 WROOM-32*, contendo regulador de tensão que permite alimentar diretamente da porta USB ou da entrada para um conector DC [22].


***Figura 1:*** *ESP32 Wemos D1 R32 [32]*

O *chip ESP32 WROOM-32* integra um microcontrolador *Tensilica* de 32 bits, interfaces de periféricos digitais padrão, interruptores de antena, amplificador de potência, amplificador de recetor de baixo ruído, filtros e módulos de gerenciamento de energia em um pacote pequeno. Fornece também *Wi-Fi* de 2,4 GHz (suportando velocidade de até 150 MB/s), comunicação sem fio BLE (*Bluetooth Low Energy*) e *Bluetooth* clássica [22].

*Tabela 2: Especificações Técnicas do Wemos D1 R32 [22].*

| | |
|---|---|
| Tensão de alimentação (micro USB) | 5VDC |
| Tensão de entrada DC | 7-12V |
| Tensão de entrada/saída | 3.3V |
| Corrente operacional mínima | 250mA |
| Frequência de *Clock* | 240MHz |
| RAM | 512kB |
| Memória flash externa | 4MB |
| Interfaces de comunicação | SPI, I2C, I2S, IR, UART, PWM |
| Protocolos *Wi-Fi* | 802.11 b/g/n/i (802.11n até 150Mbps) |
| Frequência de Wi-Fi | 2.4 - 2.5 GHz |
| Antena sem fio | PCB |
| *USB to serial chip* | CH340 |

## A. Descrição dos pinos

Os pinos gerais fornecem conectividade aos pinos GPIO (*general-purpose input/output*) que suportam PWM, pinos GPI (somente entrada), sensores *capacitive touch*, interfaces I2C e I2S, ADC (conversão analógica para digital), DAC (conversão digital para analógica), interface SPI ou UART em pinos dedicados [22].

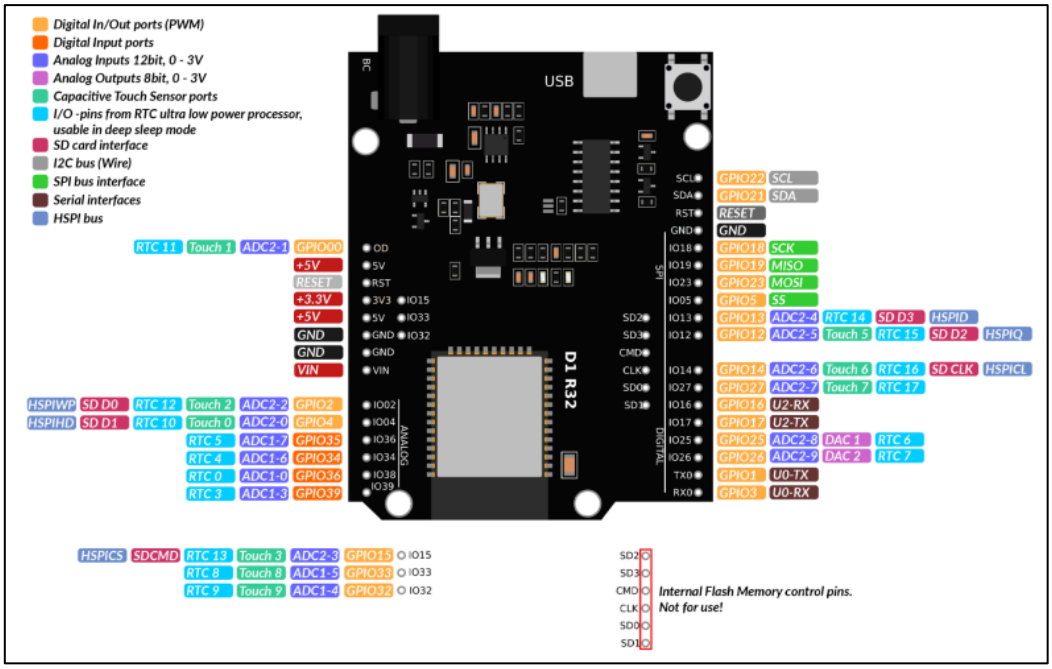


*Figura 2: Pinagem ESP32 D1 R32 [22].*

Tal como uma placa de Arduíno Uno, o *Wemos D1 R32* também possui pinos de entrada/saída, designados de pinos GPIO e, vale advertir que só operam com 3.3V, ou seja, aplicar mais de 3,3V em qualquer pino danificará o *chip* ESP32.

Os pinos GPIO34, 35, 36 e 39 são GPI e, esses não possuem resistências *pull-up* ou *pull-down* internos. Os pinos GPIO6, 7, 8, 9, 10 e 11 estão conectados ao *SPI flash* integrado no *chip ESP-WROOM-32* e não são recomendados para outros usos. O ESP32 possui sensores *capacitive touch* interno, que podem ser usados para ativar o ESP32 do *deep sleep* (suspensão profunda). Os pinos do *Wemos D1 R32* destinados ao *capacitive touch* são: GPIO4, 00, 2, 12, 14 e 27.

O controlador ESP32 PWM possui 16 canais independentes que podem ser configurados para gerar sinais PWM com diferentes propriedades. Todos os pinos que atuam como saídas podem ser usados como pinos PWM, exceto os GPIO 34 a 39. Para gerar um sinal PWM, devem ser definidos no código os parâmetros: frequência do sinal, ciclo de trabalho, canal PWM, GPIO onde enviar o sinal.

O ESP32 contém dois canais I2C e qualquer pino pode ser configurado como SDA ou SCL. Ao usar o ESP32 com o Arduíno IDE, por padrão os pinos I2C são: GPIO21(SDA), GPIO22(SCL).

### B. Comunicação *USB to serial*

A porta de conexão *micro USB* feito em torno do *chip* CH340, permite a comunicação em série de USB para UART. O *chip* possui o recurso de porta COM virtual (VCP) que aparece como porta COM em aplicativos de PC.

A interface CH340 UART implementa todos os sinais RS-232, incluindo sinais de controle e *handshaking*, portanto, o *firmware* do sistema existente não precisa ser modificado.

### C. Comunicação *Wi-Fi*

A interface de comunicação *Wi-Fi* integrada pode operar em três modos diferentes: estação *Wi-Fi*, ponto de acesso e ambos ao mesmo tempo. Ele suporta os seguintes recursos:

- taxas de dados 802.11b e 802.11g;
- 802.11n MCS0-7 em largura de banda de 20MHz e 40MHz;
- 802.11n MCS32;
- 802.11n 0.4μS intervalo de guarda;
- taxa de dados de até 150Mbps;
- até 21 dBm (decibel miliwatt) de potência de transmissão;
- potência de transmissão ajustável;
- diversidade e seleção de antenas (*hardware* gerenciado por *software*).

**D. Comunicação *bluetooth***

A *Wemos D1 R32* é um microcontrolador de baixo consumo de energia e alta conectividade que contém um rádio *Bluetooth* integrado. A sua comunicação *Bluetooth* é altamente flexível e pode ser configurada tanto como um dispositivo periférico quanto como um dispositivo central. Como um dispositivo periférico, o ESP32 pode transmitir dados para outros dispositivos e, como um dispositivo central, pode ser conectada a outros dispositivos e receber dados deles [22].

## 3.2 Estudo do Microcomputador

Além dos microcontroladores, existem outros sistemas que funcionam através do uso de microprocessadores. Desses sistemas, os mais encontrados são os microcomputadores.

Um microcomputador é um computador, caracterizado pela presença de um único microprocessador e possui dimensões e capacidade computacional limitado. Foi projetado para satisfazer as necessidades de um único utilizador cujo custo é bastante baixo. O que difere o microcomputador do microcontrolador, é que no microcomputador a CPU, as memorias (voláteis e não voláteis) e as interfaces dos seus periféricos, não estão presentes num único CI. Os microcomputadores realizam diversas funções ao mesmo tempo, muitas das quais provém da interação com os usuários, como, editar textos, executar jogos, gerar interfaces gráficas, além de outras funções, que se diferenciam daquelas normalmente realizadas por microcontroladores.

Para o projeto será utilizada o microcomputador Raspberry Pi 3 modelo B como um servidor. O Raspberry Pi propõe-se a ser um computador completo, de baixo custo.

### 3.2.1 Raspberry Pi 3 modelo B

Conhecido como um dos computadores mais baratos, Raspberry Pi ou simplesmente RPI, foi lançado para o mercado do consumidor no fim de fevereiro de 2012 pela *Raspberry Pi Foundation*, com a proposta de ser um equipamento barato e fazer parte do currículo escolar, a fim de participar do processo educacional não só das crianças como também de estudantes ou de outras pessoas quaisquer [34].

O RPI é um computador completo, com considerável poder de processamento, numa placa de circuito impresso. É classificada como SBC (*Single-Board Computer*), ou seja, um computador montado em uma única placa com processador, memória, entrada/saída e outros componentes necessários para o seu funcionamento [34].

O RPI é projetado para executar sistemas operativos baseados em GNU/Linux. Diferente do Windows ou do OS X, o Linux é aberto, ou seja, é possível baixar o código fonte de todo o sistema operacional e fazer quaisquer alterações necessárias. Os softwares desenvolvidos para Windows ou OS X não funcionam nativamente no Linux. É de referir também que o Linux não é um exclusivo do Raspberry Pi [34].

***Figura 3:*** *Raspberry Pi 3 Modelo B [32].*

O Raspberry Pi 3 Modelo B, lançado em 29 de fevereiro de 2016, contém um processador *Quad Core 1.2 GHz Broadcom BCM2837* de 64bit, capaz de desenvolver diversas aplicações tais como um computador convencional, por exemplo reprodução de vídeos em alta definição, folhas de cálculos, reprodução de jogos e entre outros [32].

No seu hardware, encontra-se integrados um adaptador *Wi-Fi* e uma interface *bluetooth 4.1* que, dessa forma beneficia o utilizador a não efetuar a compra de adaptadores externos, possibilitando assim a liberação de portas USB para interface com outros dispositivos [32].

***Tabela 3:*** *Especificações Técnicas do RPI 3B [32].*

| |
|---|
| Processador: Quad-Core 1.2GHz Broadcom BCM2837 64bit CPU |
| RAM: 1GB |
| Adaptador Wi-Fi 802.11n |
| Bluetooth 4.1 BLE (*Bluetooth Low Energy*) |
| Conector de vídeo HDMI |
| 4 Portas USB 2.0 |
| Conector Ethernet |
| Interface para câmara (CSI) |
| Interface para display (DSI) |
| Entrada para cartão microSD |
| Conector de áudio e vídeo |
| GPIO de 40 pinos |
| Fonte de alimentação micro USB 5V 2.5A |

Fonte: Autoria própria com dados do *datasheet*.

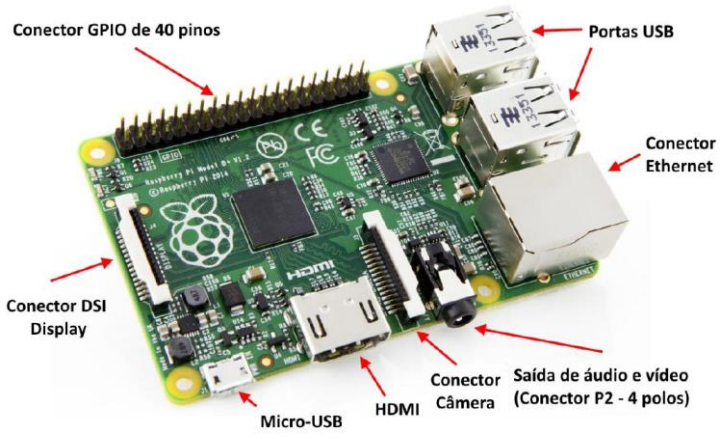


***Figura 4:*** *Legenda de periféricos de entrada/saída RPI 3B [32].*

## A. Pinos de entrada e saída de uso geral

O GPIO é basicamente um conjunto de pinos, responsável por fazer a comunicação de entrada e saída de sinais digitais. No caso do RPI 3B, é composto por 40 pinos. Esses pinos podem ser acessos para controlar *hardware* tais como LED, motores,

relés, etc., sendo esses como exemplos para saídas. Como entrada podem efetuar leitura de botões, interruptores, ou também leitura de sensores como, de temperatura, de luminosidade, de movimento e entre muitos outros.

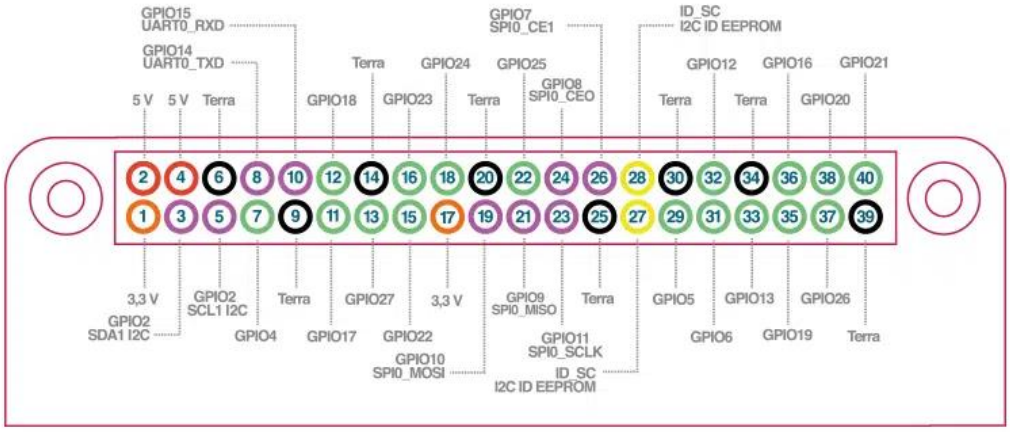


***Figura 5:*** *Diagrama de pinos GPIO do RPI 3B [33]*

Para entender o diagrama de uma forma mais clara, segue-se as descrições de cada pino:

- **5V:** saída para alimentação de tensão de 5V.
- **3.3V:** saída para alimentação, porém com uma tensão de 3.3V.
- **GND:** pino de terra (*ground*).
- **Entrada e Saída:** enviar e receber dados digitais.
- **I2C:** programadas para interface I2C (fazem conexões entre periféricos de baixa velocidade).
- **UART:** porta *serial* que utiliza o protocolo RS-232 para enviar e receber sinal digital.
- **SPI:** entrada e saída de dados digitais, porém, com esses pinos é possível fazer uma comunicação em série *full-duplex* síncrono.
- **ID EEPROM:** tipo de memória que pode ser programado e apagado várias vezes, através de uma tensão elétrica interna ou externa.

## B. Alimentação

O RPI 3B é alimentado pelo pequeno conector *micro USB* encontrado na lateral esquerda inferior da placa. Normalmente o RPI 3B consome mais energia do que a maioria dos dispositivos *micro USB* e requer uma fonte de tensão de 5V 2500mA para

funcionar. Contudo, deve-se ter o cuidado com alguns carregadores, alguns podem fornecer energia insuficiente, causando problemas intermitentes na operação do mesmo.

Conectar o RPI à uma porta USB de um computador é possível, mas não é recomendado, uma vez que, assim como os carregadores de menor porte, as portas USB de um computador não conseguem fornecer a potência requerida para que o RPI trabalhe corretamente.

Por não possuir um botão de "ligar" e "desligar", deve-se conectar a fonte de energia *micro USB* apenas quando estiver pronto para começar a usar. Sendo assim começará a funcionar no instante em que for conectado à energia e apenas poderá ser desligado se removermos, fisicamente, o cabo de energia. Todavia deve-se encerrar a execução do sistema operativo antes de se desligar o RPI da alimentação.

### C. Sistemas Operacionais Suportados

O RPI 3B é compatível com sistemas operacionais baseados em Linux, por exemplo *Debian*, *Arch Linux*, *Ubuntu* (*Ubuntu Mate* e *Snappy Ubuntu Core*), *Raspbian* (baseado no *Desbian*, é a variante Linux oficial do RPI) Windows 10 IoT Core, OSMC entre outros.

### D. Entrada de cartão micro SD

Por não possuir um disco rígido real como um computador convencional, um cartão micro SD é usado como SSD (*solid state drive*), que é destinado à instalação do SO e todos os outros softwares, bem como na armazenagem de arquivos.

Para o RPI 3B será necessário um cartão micro SD de pelo menos 8GB, e deve ser um cartão de classe 10 (o número de classe de um cartão SD determina a velocidade mínima de gravação sequencial), uma vez que estes cartões são capazes de transferir pelo menos 10MB/seg.

### E. Saída HDMI

Esta porta de saída é destinada para conectar o RPI com um monitor via HDMI (*High Definition Multimedia Interface*). Assim, qualquer monitor ou TV que possui

entrada HDMI pode ser conectada ao RPI. No entanto, se o monitor possuir entrada VGA (*Video Graphics Array*), deve-se utilizar um conversor HDMI-VGA.

### F. Entrada Ethernet e USB

Tanto a porta Ethernet como a porta USB do RPI 3B são fornecidos através do chip LAN9514 integrado. Contém quatro portas USB 2.0 de alta velocidade com um controlador Ethernet 10/100. As portas USB são usadas para conectar periféricos de entradas (teclado, rato, etc.). Quase tudo que pode ser conectado ao computador convencional via USB também pode ser conectado ao RPI.

### G. Saída de vídeo RCA e saída de áudio

O conetor de áudio e RCA está presente na placa e destina-se saída de áudio e vídeo. Mesmo sabendo que o RPI suporta o som em sua saída HDMI, há uma entrada de áudio padrão de 3,5 mm para conectar auriculares. Para o vídeo, a entrada RCA envia vídeo para qualquer dispositivo de vídeo RCA conectado.

## 3.3 Sensores

Sensores são dispositivos que convertem uma grandeza física (como temperatura, luz, pressão, humidade, movimento, etc.) em um sinal elétrico ou digital que pode ser processado e interpretado por um sistema eletrônico. Eles são utilizados em uma ampla variedade de aplicações, desde controle de processos industriais até monitoramento ambiental, sistemas de segurança, dispositivos médicos, entre outros.

Existem uma variedade de sensores, cada um projetado para medir uma grandeza física específica. Por exemplo, um termômetro é um sensor que mede a temperatura, um sensor de luminosidade é usado para detetar a quantidade de luz em um ambiente, um sensor de pressão pode ser usado para medir a pressão arterial em dispositivos médicos, e assim por diante.

Os sensores são essenciais para muitas tecnologias modernas, como na robótica, smartphones, câmaras digitais, carros autônomos e dispositivos de IoT. Eles permitem que os dispositivos “sintam" e “reajam” ao ambiente ao seu redor, tornando possível a

automação de tarefas e a coleta de dados para análise e tomada de decisões. Os tipos de sensores utilizados neste projeto são apresentados nesta subseção.

### 3.3.1 BME680

O *BME680* é um sensor digital 4 em 1 que permite efetuar a medição de gás, humidade relativa, pressão barométrica e temperatura de alta linearidade e alta precisão. É especialmente desenvolvido para aplicações móveis onde o tamanho e o baixo consumo de energia são requisitos críticos. Dependendo do modo de operação específico, garante consumo otimizado, estabilidade a longo prazo e alta robustez EMC. Para medir a qualidade do ar para o bem-estar pessoal, o sensor de gás do BME680 pode detetar uma ampla gama de gases, como VOC [19] [20].


***Figura 6:*** *Sensor BME680 [20].*

De acordo com o *datasheet* do sensor, o mesmo é capaz medir humidade com precisão de ±3%, pressão barométrica com precisão absoluta de ±1hPa e temperatura com precisão de ±1,0°C. Como a pressão muda com a altitude e as medições de pressão são muito boas, também pode ser usada como um altímetro com ± 1 metro ou mais precisão [20].

O BME680 contém um pequeno sensor MOX (óxido de metal) em que, quando aquecido muda de resistência com base nos VOC no ar, para que possa ser usado para detetar gases e álcoois como etanol, álcool e monóxido de carbono e realizar medições da qualidade do ar. Vale salientar que dará um valor de resistência, com conteúdo VOC geral, não pode diferenciar gases ou álcoois [20].

Nota-se que este sensor, como todos os sensores de VOC/gás, tem variabilidade e, para obter medições mais precisas, é recomendado que ao fazer o primeiro acionamento do sensor, o mesmo seja mantido ligado por um período de 48 horas, conhecido como "tempo de esquenta", após este primeiro período é recomendado a espera por 30 minutos no modo de operação e ambiente desejado [19] [20].

A alimentação do sensor é feita através do pino $V_{IN}$, como mostra na figura 7, a tensão fornecida deve ser na faixa de 3.0 a 5.0V, pois para conveniência do utilizador o módulo possui um regulador de tensão embutido [19] [20]. As especificações básicas do sensor BME680 são apresentadas na tabela 4.

***Tabela 4:*** *Especificações técnicas do BME680*

| | |
|---|---|
| Dimensões | 3.0 x 3.0 x 0.93mm |
| Interface | I2C e SPI |
| Tensão de alimentação | 1,71 - 3,6V |
| Faixa de medição | -40 à 85 °C, 0 – 100 %, 300 - 1100 hPa |
| Resolução de humidade | 0,008% |
| Resolução de pressão | 0,18 Pa |
| Resolução de temperatura | 0,01 °C |

**Fonte:** Autoria própria com dados do *datasheet*.

### 3.3.2 TSL2561

O *TSL2561* é um sensor de luminosidade, ou seja, um conversor de luz para digital que transforma a intensidade da luz em uma saída de sinal digital que pode ser lido por qualquer microcontrolador que trabalha com comunicação do tipo I2C [21].

Este sensor combina um fotodiodo de banda larga (para luz visível) e um fotodiodo de resposta infravermelha (para luz não visível) em um único circuito integrado CMOS capaz de fornecer uma resposta quase fotótica em uma faixa dinâmica efetiva de 20 bits (resolução de 16 bits). Possuí dois conversores ADC integrados que convertem as correntes de fotodiodo em uma saída digital que representa a irradiância medida em cada canal [21].

O grande diferencial desse sensor é que ele permite medir separadamente a luz infravermelha e a luz visível ao olho humano, o que facilita nos processos que exigem análise de luminosidade [21].


***Figura 7:*** *Sensor TSL2561 [21].*

A saída digital pode ser inserida em um microprocessador onde a iluminância (nível de luz ambiente) em lux é derivada usando uma fórmula empírica para aproximar a resposta do olho humano [21].

A alimentação do sensor é feita através do pino $V_{DD}$ ou $V_{CC}$ e, a faixa de tensão fornecida recomendada é de 2.7 a 3.6V [21]. As características básicas do sensor TSL2561 são apresentadas na tabela 5.

***Tabela 5:*** *Características técnicas do TSL2561 [25].*

| | |
|---|---|
| Tensão de operação | 2,7 – 3,6V |
| Temperatura | -30º à 70ºC |
| Corrente utilizada | 0,24 - 0,6mA |
| Faixa de medição | 0,1 – 40000 Lux |
| Interface | I2C |
| Endereço I2C | 0x39, 0x29, 0x49 |
| Dimensões | 19 x 16mm |

Fonte: Autoria própria com dados do *datasheet*.

### 3.3.3 KY-037

O KY-037 é um módulo de sensor de som de eletreto que pode ser usado para detetar níveis de som em um determinado ambiente. Ele é composto por um microfone de eletreto, um amplificador operacional e um circuito comparador [24].

***Figura 8:*** *Sensor KY-037 [24].*

O sensor é alimentado com uma tensão de 3.3V-5V e possui uma saída analógica OUT que muda de estado quando o nível de som ultrapassa um determinado limite. Isso significa que o sensor pode ser usado como um interruptor que é acionado quando há um nível de som específico no ambiente [24].

A saída digital OUT nas placas pequenas pode ser conectada diretamente ao microcontrolador, detetando o som ambiente através do microcontrolador detetando alto

e baixo nível. A saída digital OUT nas placas pequenas pode acionar diretamente o módulo de relé, que funciona como um interruptor ativado por voz [24].

O nível de som que aciona o sensor pode ser ajustado usando o potenciômetro presente no módulo. O sensor é fácil de usar e pode ser integrado a projetos eletrônicos que envolvam deteção de som, como projetos de automação residencial, sistemas de alarme, entre outros [24].

No entanto, é importante notar que o sensor de som KY-037 é sensível a ruídos e interferências, portanto, é necessário cuidado na escolha da localização e no projeto do circuito para minimizar essas interferências e garantir uma leitura precisa [24].

### 3.3.4 SDS011

O sensor SDS011 é um sensor de partículas finas (PM2.5 e PM10) que mede a concentração de partículas suspensas no ar. Ele é amplamente utilizado em projetos de monitoramento da qualidade do ar em ambientes internos e externos [41].

***Figura 9:*** *Sensor SDS011 [24].*

Esse sensor utiliza um princípio de dispersão a laser para contar e medir o tamanho das partículas presentes no ar. Contem um laser interno que emite luz e um foto detetor que mede a quantidade de luz dispersada pelas partículas. Com base na quantidade de luz dispersada, o sensor calcula a concentração de partículas em microgramas por metro cúbico (µg/m³) para partículas com diâmetro de 2,5 micrômetros (PM2.5) e 10 micrômetros (PM10) [41].

É relativamente preciso e possui uma ampla faixa de medição, podendo ser conectado a um microcontrolador, como Arduíno ou Raspberry Pi, através de suas portas de comunicação UART ou usando um conversor de nível lógico, se necessário [41].

Vale observar que o sensor SDS011 requer uma fonte de alimentação de 5V e deve ser protegido da exposição direta à luz solar e à humidade para garantir medições precisas. Além disso, requer calibração periódica para manter sua precisão ao longo do tempo [41].

***Tabela 6:*** *Características técnicas do SDS011.*

| | |
|---|---|
| Parâmetros de medida | PM2.5, PM10 |
| Faixa de medição | 0 – 999,9 μg/m$^3$ |
| Tensão nominal | 5 V |
| Corrente nominal | 70mA ± 10mA |

Fonte: Autoria própria com dados do *datasheet*.

## 3.4 *Internet of Things* (IoT)

Um dos exemplos importantes de TIC é a IoT (em português Internet das Coisas). A IoT refere-se a um sistema de dispositivos conectados à internet que são capazes de coletar e trocar dados entre si, sem a necessidade de intervenção humana. Esses dispositivos podem incluir sensores, câmaras, monitores de saúde, veículos eletrodomésticos e muitos outros objetos cotidianos [28].

A IoT é uma tecnologia em rápido desenvolvimento que tem um enorme potencial para melhorar a eficiência, a produtividade e a qualidade de vida, além de oferecer novas oportunidades de negócios em diversos setores. A IoT é uma das muitas tecnologias que se enquadram na categoria de TIC e demonstra como a conectividade e a troca de dados em tempo real estão se tornando cada vez mais importantes em nossas vidas e negócios [28].

### 3.4.1 Protocolo MQTT

MQTT (*Message Queuing Telemetry Transport*) é um protocolo de comunicação de mensagens leve e de baixo consumo de energia, desenvolvido para conectar dispositivos de IoT a sistemas de processamento de dados em nuvem ou em centros de dados [26].

O MQTT é um protocolo de mensagens *publish/subscribe* que funciona em cima do protocolo TCP/IP e usa um modelo de troca de mensagens entre um cliente e um servidor. Foi projetado para ser leve e eficiente em termos de uso de recursos, permitindo

que dispositivos de IoT com capacidade limitada de processamento e memória se comuniquem de forma confiável e segura com sistemas remotos [26] [27].

Sensores inteligentes, dispositivos acessórios e outros dispositivos da IoT normalmente precisam transmitir e receber dados por meio de uma rede com limitação de recursos e largura de banda limitada. Esses dispositivos IoT usam o MQTT para transmissão de dados, pois é fácil de implementar e pode comunicar dados IoT com eficiência [26].

### A. Princípio de funcionamento

A arquitetura MQTT é baseada em um modelo de *pub/sub*, no qual os dispositivos de IoT (*publishers*) enviam mensagens para um servidor intermediário (*broker*), que distribui as mensagens para os dispositivos de destino (*subscribers*). O protocolo MQTT usa tópicos para identificar as mensagens e permitir que os *subscribers* assinem e recebam apenas as mensagens que são relevantes para eles [26] [27].

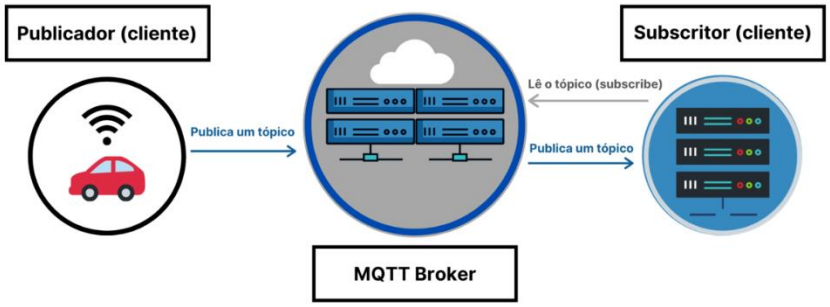


***Figura 10:*** *Arquitetura do protocolo MQTT.*

Fonte: https://pt.m.wikipedia.org/wiki/Ficheiro:Arquitetura_MQTT_exemplo.png

O funcionamento geral do MQTT pode ser descrito da seguinte forma [26] [27]:

1. Um dispositivo de IoT (*publisher*) envia uma mensagem para o *broker* MQTT em um tópico específico;
2. O *broker* recebe a mensagem e a armazena em um *buffer* (amortecedor);
3. O *broker* procura por todos os dispositivos de IoT que tenham assinado esse tópico;
4. O *broker* envia a mensagem para todos os dispositivos de IoT que assinaram o tópico relevante;

5. Os dispositivos de IoT (*subscribers*) recebem a mensagem e realizam as ações necessárias com base nas informações contidas nela.

O MQTT é amplamente utilizado em ambientes de IoT devido à sua eficiência, escalabilidade e segurança. Ele permite que os dispositivos de IoT se comuniquem de forma confiável e segura com sistemas de processamento de dados em nuvem ou centro de dados, facilitando a coleta e o processamento de dados de IoT em tempo real [26] [27].

## 3.5 Computação em Nuvem

Computação baseada em nuvem, também é um exemplo de grande importância de TIC. É um modelo de tecnologia de fornecimento de serviços de computação sob demanda pela internet, permitindo que os usuários tenham acesso a uma ampla gama de recursos de computação, incluindo servidores, armazenamento, base de dados, *software* e aplicativos, por meio de provedores de serviços em nuvem (CSPs – *Cloud Service Providers*) em qualquer parte do mundo [40].

A computação em nuvem é uma das tecnologias importantes e disruptivas na indústria de TI (Tecnologia da Informação) e, tem sido amplamente adotada por empresas de todos os tamanhos. Ela permite que as empresas reduzam os custos de infraestrutura, aumentem a eficiência operacional e melhorem a escalabilidade e a flexibilidade de seus sistemas de TI [40].

Existem três principais modelos de computação em nuvem: nuvem pública, nuvem privada e nuvem híbrida. Cada modelo tem suas próprias vantagens e desvantagens, e as empresas escolhem a opção que melhor atende às suas necessidades de negócios e orçamento. A computação em nuvem é uma das principais tecnologias que estão transformando a forma como as empresas gerenciam e acessão seus recursos de TI [40].


***Figura 11:*** *Visão geral de uma nuvem e seus periféricos.*

Fonte: https://www.researchgate.net/figure/Figura-7-Visao-geral-de-uma-nuvem-computacional-e-seus-perifericos_fig2_341062559

## 3.6 Protocolos de Comunicação em Série

A comunicação em série é um método de comunicação de dados que transmite bits de informação em uma única direção, um bit de cada vez, em uma sequência ordenada de bits. Esta difere da comunicação paralela, onde vários bits de informação são transmitidos simultaneamente por vários fios [35].

Na comunicação em série, os dados são transmitidos em uma série de pulsos elétricos ou sinais óticos, geralmente usando apenas dois fios de comunicação, um para transmitir dados (TX) e outro para receber dados (RX). Essa forma de comunicação é usada em uma ampla variedade de dispositivos eletrônicos, desde sistemas de controlo industrial até computadores pessoais e dispositivos móveis [35].

Existem vários tipos de protocolos de comunicação em série, cada um com suas próprias características e padrões de transmissão de dados como também, seus próprios conjuntos de requisitos de hardware e software para implementação. Neste projeto são utilizados três protocolos de comunicação em série para comunicação dos sensores com a placa do microcontrolador: I2C, UART e o USB.

### 3.6.1 I2C

O I2C (*Inter-Integrated Circuit*), desenvolvido pela *Philips* (atualmente *NXP Semiconductors*) em 1982, é um protocolo de comunicação em série utilizado para

conectar dispositivos eletrônicos, como sensores, memórias, displays, microcontroladores, entre outros [36].

É um protocolo síncrono, o que significa que há um sinal de *clock* compartilhado entre os dispositivos conectados para sincronizar a transmissão de dados. Ele utiliza apenas dois fios, um para transmitir dados (SDA) e outro para sincronizar o sinal de *clock* (SCL). Isso faz com que o protocolo seja simples e fácil de usar, e também economiza espaço em placas de circuito impresso [36].

Cada dispositivo I2C tem um endereço único, o que permite que vários dispositivos sejam conectados a um único barramento I2C. O endereço é utilizado para selecionar o dispositivo com o qual se deseja comunicar [36].

### 3.6.2 UART

Significa "*Universal Asynchronous Receiver/Transmitter*", é um protocolo de comunicação em série amplamente utilizado em dispositivos eletrônicos para a comunicação com outros dispositivos, como sensores, displays, microcontroladores, computadores, entre outros.

A UART é um componente de circuito que converte dados paralelos em série para transmissão e, em seguida, converte os dados recebidos em série em paralelo para processamento. A comunicação ocorre através de dois fios: um para transmitir dados (TX) e outro para receber dados (RX) [35].

A comunicação UART é assíncrona, ou seja, não requer um sinal de *clock* dedicado para sincronizar a transmissão de dados entre os dispositivos, como no caso do SPI ou I2C [35].

### 3.6.3 USB

O USB (*Universal Serial Bus*) é um protocolo bidirecional que usa um sistema de comunicação "mestre-escravo". O dispositivo *host*, normalmente um computador, é o mestre e controla a comunicação com os dispositivos escravos (os dispositivos USB conectados). O dispositivo *host* envia mensagens de controle para o dispositivo USB, solicitando informações ou instruindo-o a realizar uma ação específica [35].

Ele oferece recursos avançados de gerenciamento de energia, é capaz de fornecer energia elétrica para dispositivos conectados e suporta a conexão de vários dispositivos em um único barramento [35].

## 3.7 Eclipse Mosquitto

Eclipse Mosquitto, criado pela Eclipse Foundation em 2010, é um servidor de mensagens MQTT de código aberto, que implementa o protocolo MQTT de versões 5.0, 3.1.1 e 3.1. É amplamente utilizado como uma plataforma de mensagens para IoT e aplicações de comunicação de dados em tempo real [29].

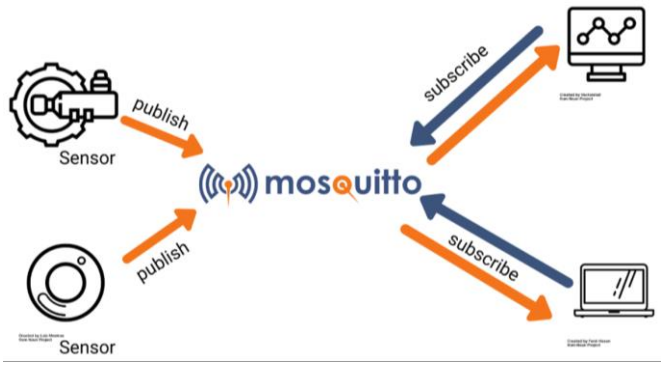


***Figura 12:*** *Visão geral do Eclipse Mosquitto.*

Fonte: https://developer.community.boschrexroth.com/t5/Store-and-How-to/Cedalo-Eclipse-Mosquitto-MQTT-Broker/ba-p/50927

O Mosquitto é projetado para ser leve, rápido e eficiente, permitindo que dispositivos de IoT com recursos limitados de processamento e memória se conectem a um servidor MQTT e troquem mensagens de forma confiável e segura. Ele suporta conexões SSL/TLS criptografadas para segurança adicional e oferece recursos como autenticação e autorização para garantir que somente os usuários autorizados possam ter acesso ao sistema [29].

Além disso, o Mosquitto é altamente configurável e pode ser executado em diferentes sistemas operacionais, incluindo Windows, Linux e MacOS. Ele também pode ser integrado a outras plataformas de IoT e ferramentas de desenvolvimento, como o Node-RED e o Eclipse Paho [29].

O Mosquitto é uma escolha popular para desenvolvedores e arquitetos de IoT que precisam de uma solução de mensagens de baixo custo, eficiente e escalável para seus projetos de IoT.

## 3.8 Node-RED

Node-RED, criado pela IBM em 2013 e mantido atualmente pela Fundação Node-RED, é uma plataforma de desenvolvimento visual de código aberto para IoT e outras aplicações de integração de sistemas [30].

O Node-RED permite que os desenvolvedores criem fluxos de aplicativos IoT usando uma interface visual, que é baseada em uma caixa de ferramentas de nós conectáveis. Os nós, são blocos de construção reutilizáveis que podem ser arrastados e soltos em um fluxo de trabalho para criar aplicativos de IoT complexos [30].

Esta plataforma suporta uma ampla variedade de dispositivos de IoT e protocolos de comunicação, incluindo MQTT, HTTP, *Modbus*, OPC UA, *Bluetooth*, entre outros. Ele também possui recursos avançados de automação e programação, permitindo que os desenvolvedores criem fluxos de trabalho complexos e regras de negócios personalizadas [30].

Além disso, o Node-RED pode ser integrado com outras ferramentas de desenvolvimento e plataformas de IoT, como o Raspberry Pi e o AWS IoT, permitindo que os desenvolvedores criem soluções de IoT completas e escaláveis [30].

O Node-RED é uma plataforma popular para desenvolvedores e arquitetos de IoT que desejam criar aplicativos de IoT de forma rápida e eficiente, sem ter que escrever código complexo.

## 3.9 InfluxDB

InfluxDB, desenvolvido pela empresa InfluxData em 2013, é um base de dados de séries temporais de código aberto, projetado especialmente para armazenar, consultar e visualizar dados de séries temporais. É amplamente utilizado em aplicações de monitoramento de infraestruturas, IoT, e análise de dados em tempo real, telemetria, entre outros casos de uso em que é necessário lidar com dados de séries temporais [42].

O InfluxDB foi projetado para lidar com dados que mudam com o tempo, como dados de sensores, métricas de desempenho de servidores, registos de eventos, entre outros. Ele oferece uma estrutura otimizada para armazenar, consultar e analisar grandes volumes de dados de séries temporais de forma eficiente [42].

Algumas características-chave do InfluxDB incluem:

- modelos de dados de séries temporais;
- consultas flexíveis;
- escalabilidade;
- integração com outras ferramentas.

## 3.10 KiCad

KiCad, criado em 1992 por Jean-Pierre Charras, é um software de código aberto utilizado para projetar circuitos eletrônicos e criar placas de circuito impresso (PCI) [31].

O KiCad oferece uma ampla variedade de ferramentas para design de circuitos, incluindo a criação de esquemas elétricos, roteamento de PCI, criação de bibliotecas de componentes e verificação de regras de design. Ele suporta uma ampla gama de formatos de arquivo, permitindo que os usuários importem e exportem projetos para outras ferramentas de design [31].

Para acrescentar, o KiCad é altamente personalizável e pode ser configurado para atender às necessidades específicas de cada projeto. Ele é multiplataforma, o que significa que pode ser executado em diferentes SO, como Windows, Linux e MacOS [31].

# IV. IMPLEMENTAÇÃO DO PROJETO

Este capítulo relata todo o trabalho prático que foi desenvolvido ao longo do projeto, começando por uma descrição de forma breve e detalhada em que consiste o mesmo. Mostra como foi feita o processo de montagem do circuito eletrónico, como também, o processo do desenvolvimento do *firmware* em si.

Para finalizar, depois de concluído todo o processo dos códigos, o circuito eletrónico do ESP32 é transposto para uma placa de circuito impresso, em que o seu esquemático e o seu *layout* foram desenvolvidos no *software* KiCad.

## 4.1 Descrição do Sistema

O sistema é composto por um microcontrolador ESP32 modelo *Wemos D1 R32* com *Wi-Fi* e sensores que medem temperatura, pressão, humidade, VOC, luminosidade, ruído, PM2.5 e PM10, todos em uma PCI de interface. Os valores medidos pelos sensores são enviados pelo ESP32, utilizando o protocolo MQTT, para o *broker/servidor* Mosquitto em determinados tópicos.

O *broker/servidor* Mosquitto, que é instalado num RPI 3B, é responsável por receber todas as mensagens, filtrar, decidir quem está interessado nelas e divulgar para todos os clientes inscritos.

Há um aplicativo em Node-RED que controla as saídas do ESP32 e recebe as leituras dos sensores utilizando o protocolo de comunicação MQTT. Esse aplicativo também é executado no mesmo RPI.

Após realizada a receção e a análise dos dados, os valores dos parâmetros já referidos, são enviados e salvaguardados numa base de dados (em InfluxDB) e, posteriormente são mostrados em um *dashboard* em tempo real.

O diagrama geral do sistema proposto pode ser acompanhado na figura a seguir.

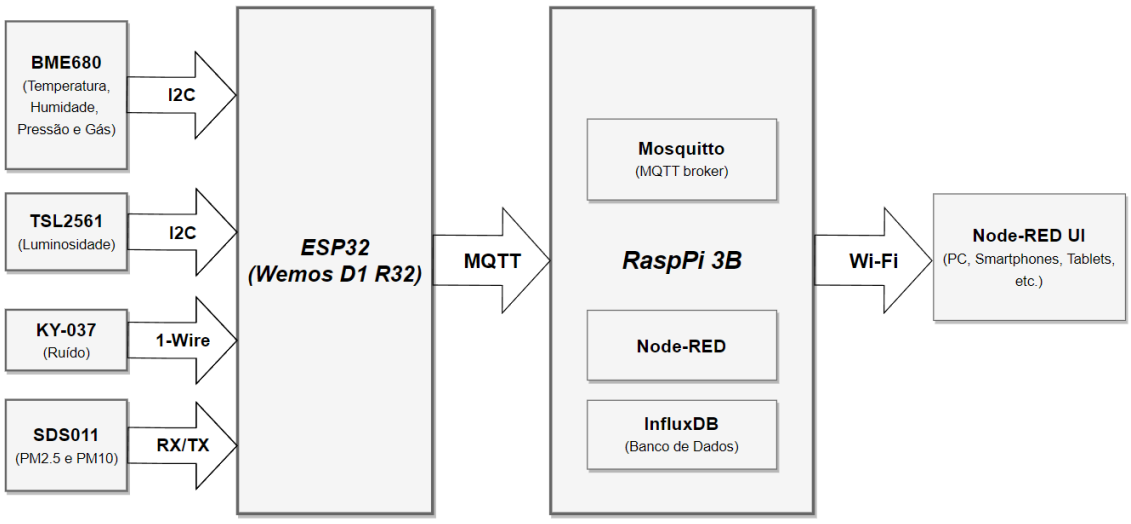


***Figura 13:*** *Diagrama geral do sistema.*

Fonte: Auditoria própria.

## 4.2 Configuração do Raspberry Pi

### 4.2.1 Instalação do SO

Assim como um qualquer outro computador, no RPI também deve ser instalado um sistema operativo (SO). Sabendo que o RPI não possui disco rígido, a instalação do SO, assim como todos os outros *softwares* e arquivos necessários de suporte são armazenados na memória externa (micro SD).

Entre dos diferentes tipos de SO existentes, optou-se pelo NOOBS (*New Out of the Box Software*) que contém o *Rasbian* incluído. O *download* da versão mais recente do NOOBS (arquivo zip) encontra-se disponível no *link* a seguir: https://www.raspberrypi.org/downloads/.

Depois de efetuar o *download* do arquivo, deve-se descompactar o mesmo, copiar para a memória externa (micro SD) e a instalação será feita no RPI.

### 4.2.2 Instalação do Mosquitto

Para instalar o Mosquitto Broker no Raspberry Pi, seguiu-se as seguintes etapas:

1. Abrir um terminal no seu Raspberry Pi;
2. Executar o comando *"sudo apt update && sudo apt upgrade"* para atualizar a lista de pacotes disponíveis;

3. Executar o comando *"sudo apt install -y mosquitto mosquitto-clients"* para instalar o Mosquitto Broker;
4. Executar o comando *"sudo systemctl status mosquitto"*, para verificar se o serviço está em execução.

### 4.2.3 Instalação do Node-RED

A versão mais recente do sistema operativo Rasbian instalado no Raspberry Pi, já vem com o Node-RED incluído, logo não foi preciso instalar.

### 4.2.4 Instalação do InfluxDB

Para instalar o base de dados InfluxDB no Raspberry Pi, seguiu-se as seguintes etapas:

1. Abrir um terminal no seu Raspberry Pi;
2. Executar o comando *"sudo apt update && sudo apt upgrade"* para atualizar a lista de pacotes disponíveis*;*
3. Executar o comando *"sudo apt install influxdb"* para instalar o InfluxDB;
4. Executar o comando *"sudo systemctl status influxdb"*, para verificar se o serviço está em execução;
5. Para ter acesso ao base de dados InfluxDB, deve-se abrir um novo terminal e executar a linha de comando *"influx"*.

## 4.3 Circuito Eletrónico

No capítulo 4, foi feito um breve estudo dos componentes que atenderam aos parâmetros necessários para a criação do sistema. Aqui, pretende-se mostrar a integração de todos os componentes que forma o circuito eletrónico final. Pode-se observar na figura a seguir como cada sensor é ligado ao microcontrolador ESP32, especificando quais são os terminais de aquisição de dados e os de alimentação, formando no final de todo o processo a interligação de todos os componentes.

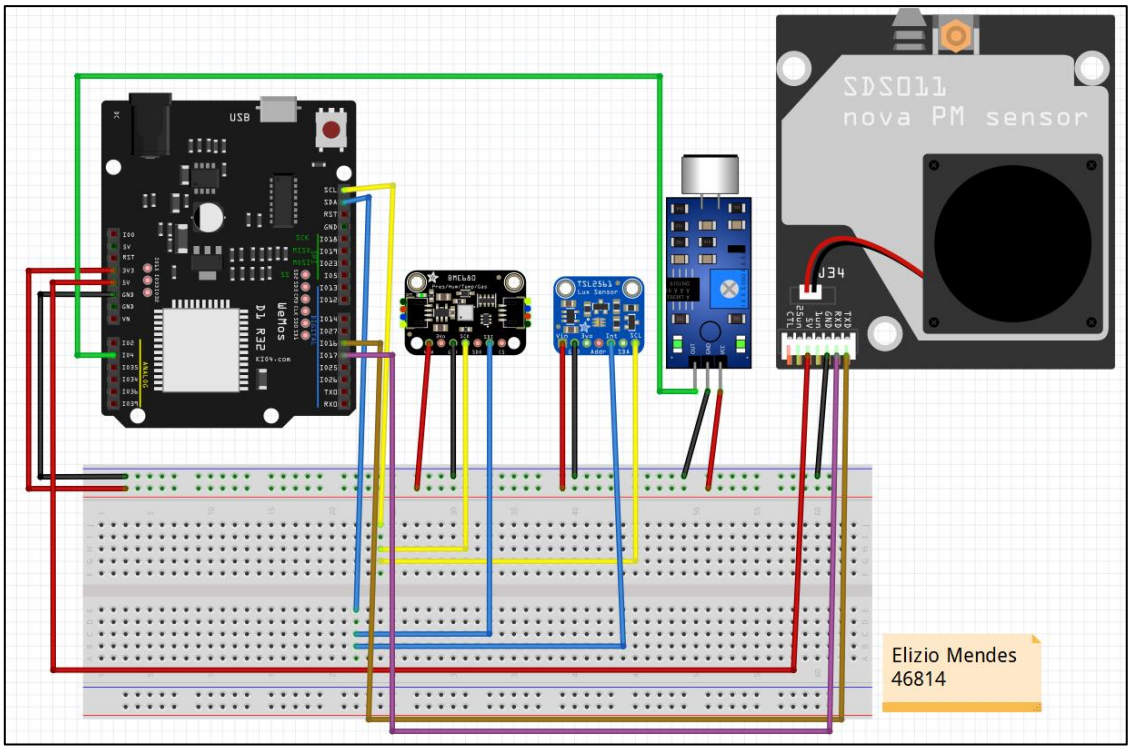


***Figura 14:*** *Circuito eletrónico do sistema*

Fonte: Auditoria própria.

Como visto anteriormente, os sensores BME680 e TSL2561 possuem interface I2C para aquisição de dados, através dos terminais SDL/SCK e SDA, ou seja, não necessitam de componentes externos para funcionar. Desse modo, as ligações desses sensores com o ESP32 são feitas de forma direta como mostrada na figura 14.

Relativamente ao sensor KY-037, a aquisição dos dados é feita através do protocolo *1-Wire* pelo terminal *out.* Por último, o SDS011 utiliza os pinos RX e TX para estabelecer a comunicação em série com o microcontrolador.

## 4.4 *Firmware* da ESP32

Toda a programação do *firmware* da ESP32 é feita no Arduíno IDE, onde a linguagem de programação é C/C++ com funções próprias para esta IDE. Foram utilizadas bibliotecas fornecidas pela própria plataforma e, essas bibliotecas foram analisadas, testadas e adaptadas para atender às necessidades do sistema.

Primeiramente, começou-se por programar o ESP32 com o propósito de ler os valores dos sensores BME680, TSL2561, KY-037 e SDS011, incorporados num só código. Para esse fim, foram utilizadas bibliotecas (*Adafruit_BME680*,

*Adafruit_TSL2561* e *Adafruit_Sensor*) capazes de configurar e ler valores desses sensores.

De seguida, para que o envio e a receção dos valores entre o microcontrolador e o Raspberry Pi fosse possível, foi necessário o uso do protocolo MQTT (utilizando a biblioteca *AsyncMqttClient*). Este permite enviar comandos para controlar saídas, ler e publicar dados de sensores. Entretanto, antes de conectar o MQTT é realizada a conexão do microcontrolador à rede *Wi-Fi*. Para tal, é utilizada a biblioteca *WiFi*.

Se a conexão entre o roteador e o *broker* MQTT for bem-sucedida, o código imprime o endereço IP do ESP32. Caso haja uma perda de ligação à rede *Wi-Fi* e/ou a conexão com o broker MQTT, o sistema inicia um *timer* e tenta reconectar.

O código completo do sistema está disponível no Apêndice A.

## 4.5 Fluxos do Node-RED

Uma das partes mais importantes do desenvolvimento do sistema consiste nos fluxos do Node-RED. É neste estágio que o usuário ou o administrador do sistema pode visualizar os dados, bem como efetuar a configuração do mesmo.

Inicialmente, optou-se pela integração de autenticação no acesso HTTPS visto que por padrão qualquer pessoa pode ter acesso ao editor do Node-RED pelo endereço de IP do sistema. Feito isto, a tela inicial é mostrada na figura 15.


***Figura 15:*** *Autenticação na página inicial do editor Node-RED.*

Fonte: Autoria própria.

### 4.5.1 Sistema de monitorização

Para desenvolvimento do fluxo do sistema de monitorização, ou seja, visualização de valores dos parâmetros em tempo real em um *dashboard*, foram utilizados os seguintes nós:

- ***MQTT-in*:** conecta-se em um *broker* e se inscreve em determinado tópico para receber as mensagens;
- ***Chart*:** mostra valores recebidos em um gráfico de linha no *dashboard*;
- ***Gauge*:** mostra valores recebidos em um medidor no *dashboard*;
- ***Text*:** mostra valores recebidos em forma de texto no *dashboard*.

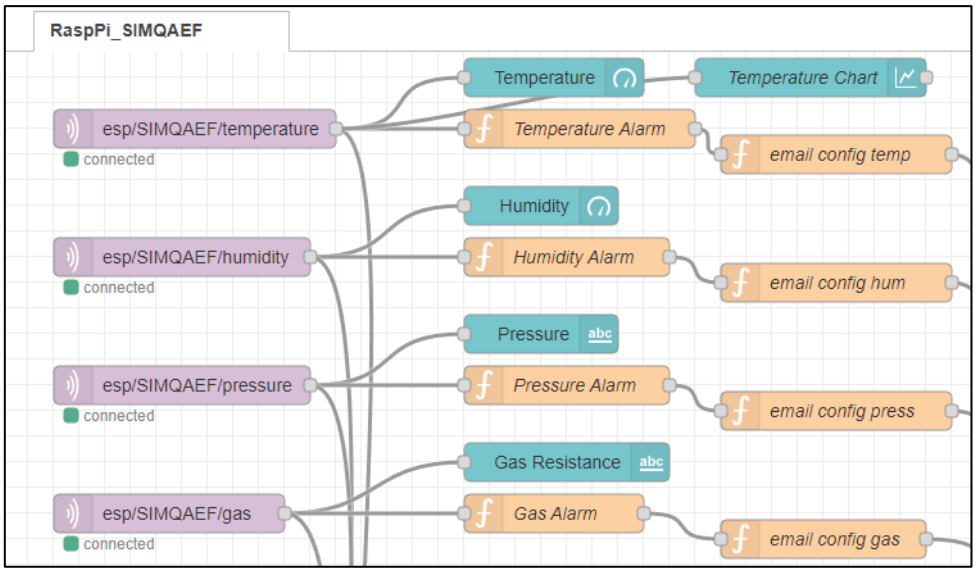


***Figura 16:*** *Fluxo Node-RED do sistema de monitorização.*

Fonte: Autoria própria.

### 4.5.2 Sistema de alarme e notificação

Para construção do fluxo do sistema de alarme e notificação via email, em tempo real, foram utilizados os seguintes nós:

- ***Function*:** permite executar um código em JavaScript que define os limites de valores dos parâmetros e enviar uma mensagem de notificação para email com o respetivo valor atual.
- ***Email*:** envia o *msg.payload* como um e-mail, com um assunto de *msg.topic*. Pertence à biblioteca *"node-red-node-email-variable"*.

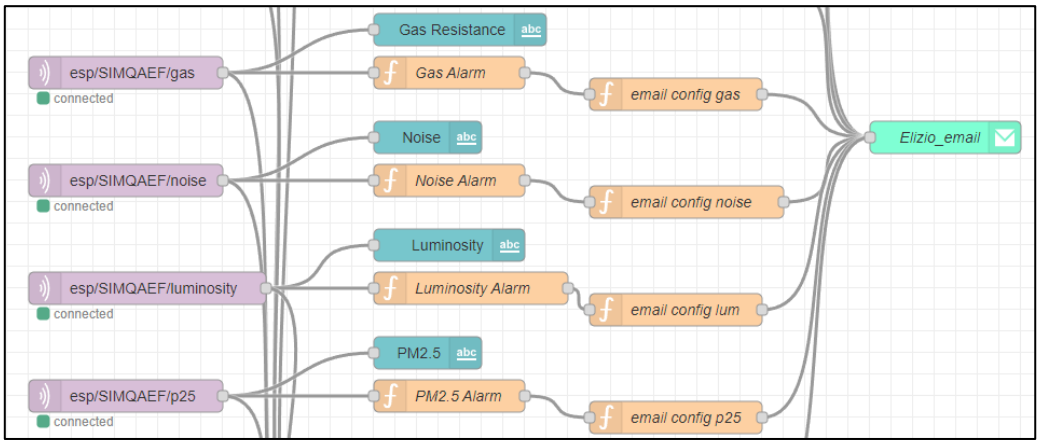


***Figura 17:*** *Fluxo Node-RED do sistema de alarme e notificação.*

Fonte: Autoria própria.

```
var payload = msg.payload;
var alarm_flag = context.get("alarm_flag");
if (typeof alarm_flag == "undefined")
    alarm_flag = false;

if (payload > 25 && !alarm_flag) {
    alarm_flag = true;
    msg.alarm = 1;
    context.set("alarm_flag", alarm_flag);
    return msg;
}

if (payload < 18 && !alarm_flag) {
    alarm_flag = false;
    msg.alarm = 0;
    context.set("alarm_flag", alarm_flag);
    return msg;
}
```

```
var temp=msg.payload;
msg.to ="eliziofg10@hotmail.com";

var d= new Date();
var message="";
if(msg.alarm)
{
    msg.topic="High Temperature Alarm";
    message ="\n Current temperature (ºC) is = ";
}
else
{
    message = "\nCurrent temperature (ºC) is = ";
    msg.topic = "\nLow Temperature Alarm";
}
msg.payload="Time: "+d+message+msg.payload;
return msg;
```

***Figura 18:*** *Código da função que define os limites da temperatura e enviar notificação para email.*

Fonte: Autoria própria.

### 4.5.3 Sistema de base de dados

Por último, para criação do fluxo do sistema de base de dados e para mostrar uma tabela, foram utilizados os seguintes nós:

- ***Change*:** usado para modificar os nomes dos tópicos em nome dos parâmetros a serem analisados;
- ***Join*:** permite unir sequências de mensagens em uma única mensagem;
- ***Influxdb out*:** permite guardar e consultar dados de uma base de dados de séries temporais InfluxDB. Pertence à biblioteca *"node-red-contrib-influxdb"*;
- ***Delay*:** permite enviar valores a cada 5 minutos para o base de dados;
- ***Debug*:** mostra as propriedades da mensagem selecionada na guia da barra lateral de depuração e, opcionalmente, o de tempo de execução;
- ***Table:*** exibe uma tabela de dados.

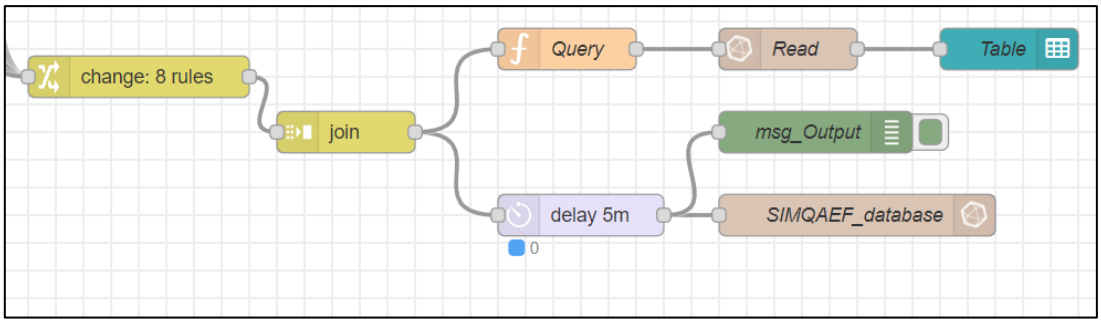


***Figura 19:*** *Fluxo Node-RED do sistema de base de dados.*

Fonte: Autoria própria.

## 4.6 Protótipo no KiCad

Para utilização do protótipo identificou-se a necessidade de desenvolvimento de uma PCI de interface para que se pudesse conectar os devidos pinos dos sensores no microcontrolador ESP32 e que ambos se mantivessem fixos. A projeção da PCI foi realizada no *software* KiCad que visa facilitar a confeção de *layouts* e suas conversões para PCI.

### 4.6.1 Esquemático

O esquemático consiste em uma das etapas da construção de uma PCI e, é nesta etapa que são estabelecidas as conexões entre os pinos dos dispositivos. Para estabelecer a conexão entre o microcontrolador ESP32 e os demais sensores, são necessárias 15 (quinze) ligações e estas estão definidas na tabela seguir. Vale realçar que na tabela, entre parênteses, corresponde à descrição do referente pino.

***Tabela 7:*** *Ligações entre os pinos dos sensores e da ESP32 D1 R32.*

| **BME680** | **ESP32 D1 R32** |
|---|---|
| 1 (VIN) | 3V3 |
| 3 (GND) | GND |
| 4 (SCK) | 22 (SCL) |
| 6 (SDI) | 21 (SDA) |
| **TSL2561** | **ESP32 D1 R32** |
| 1 (VCC) | 3V3 |
| 2 (GND) | GND |
| 3 (SCL) | 22 (SCL) |
| 4 (SDA) | 21 (SDA) |
| **KY-037** | **ESP32 D1 R32** |

| | |
|---|---|
| 1 (VCC) | 3V3 |
| 2 (GND) | GND |
| 3 (OUT) | GPIO35 |
| **SDS011** | **ESP32 D1 R32** |
| 3 (5V) | 5V |
| 5 (GND) | GND |
| 6 (RX) | 17 (U2-TX) |
| 7 (TX) | 16 (U2-RX) |

Fonte: Autoria própria.

Para a realização do esquemático, foi necessário criar os símbolos dos componentes, tanto dos sensores como do microcontrolador, visto que não faziam parte da biblioteca padrão de símbolos do KiCad. O esquemático finalizado é exposto na figura 20.

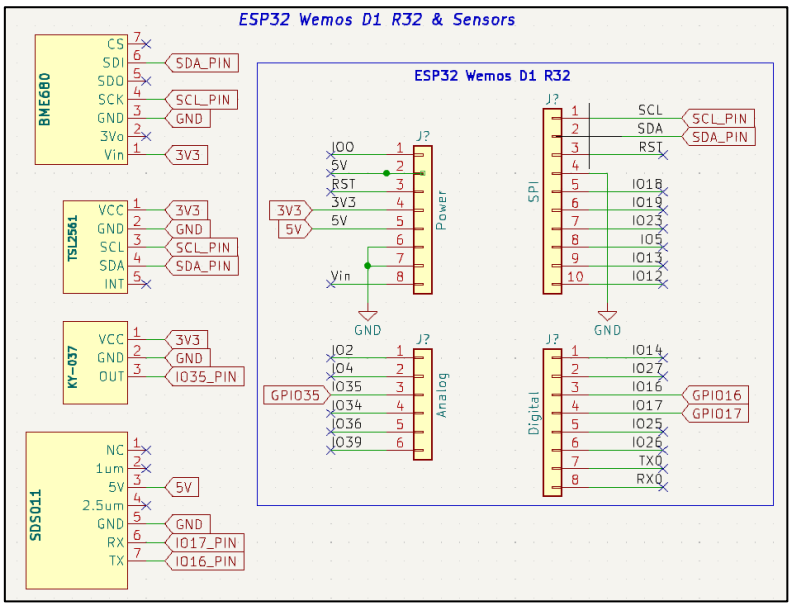


***Figura 20:*** *Esquemático do protótipo.*

Fonte: Autoria própria.

### 4.6.2 Layout

O *layout* corresponde a etapa em que a placa será desenhada e, uma fase preliminar desta etapa é a elaboração dos *footprints*, que são os desenhos das áreas onde os componentes eletrónicos ficam localizados e são soldados na PCI. Essas áreas devem ter exatamente as dimensões dos componentes físicos bem como as mesmas características, como furos e distância entre esses furos. Também é possível associar o

desenho 3D ao *footprint*, desta forma, após finalizar o *layout* pode-se ver e ter a noção de como ficará a placa após finalizada.

Após a elaboração dos *footprint* de todos componentes, o processo a seguir é a criação da placa em si que é feita no editor da PCI. Começa-se por posicionar os componentes como desejado e de seguida traça-se as pistas de acordo com as conexões previamente definidas, tendo atenção nas dimensões das larguras. Feito isto, delimita-se o tamanho da placa num todo.

A figura 21 e a figura 22 mostram o layout da placa finalizado na vista da camada de cobre superior e no ponto de vista a camada de cobre inferior, respetivamente.

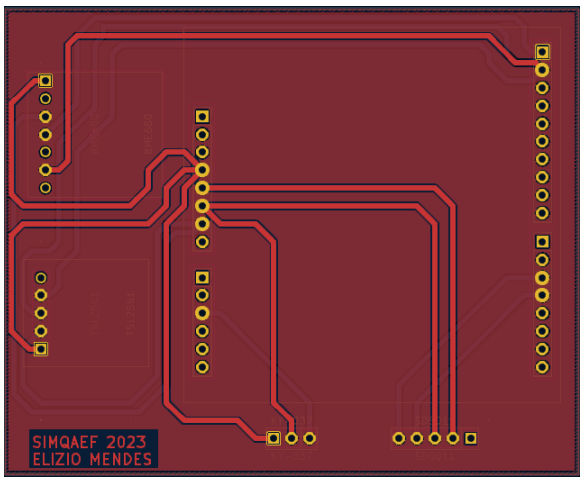


***Figura 21:*** *Layout da camada de cobre superior.*

Fonte: Autoria própria.

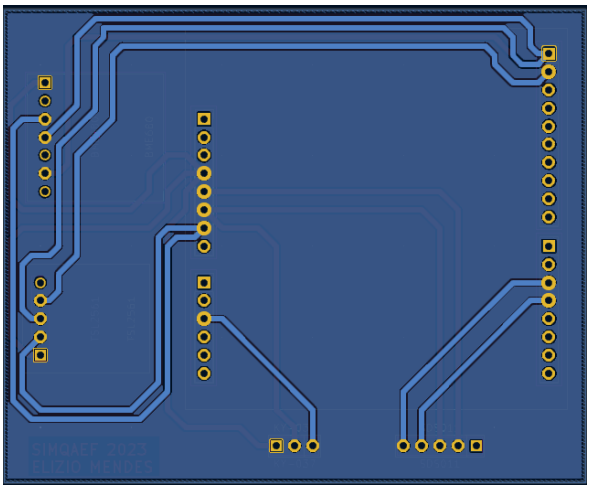

***Figura 22:*** *Layout da camada de cobre inferior.*

Fonte: Autoria própria.

Na figura 23 pode-se visualizar o *layout* para impressão da camada de cobre superior e inferior, onde somente as marcações dos pinos, pistas e letras ficam evidentes, e na figura 24, a visualização 3D da placa.

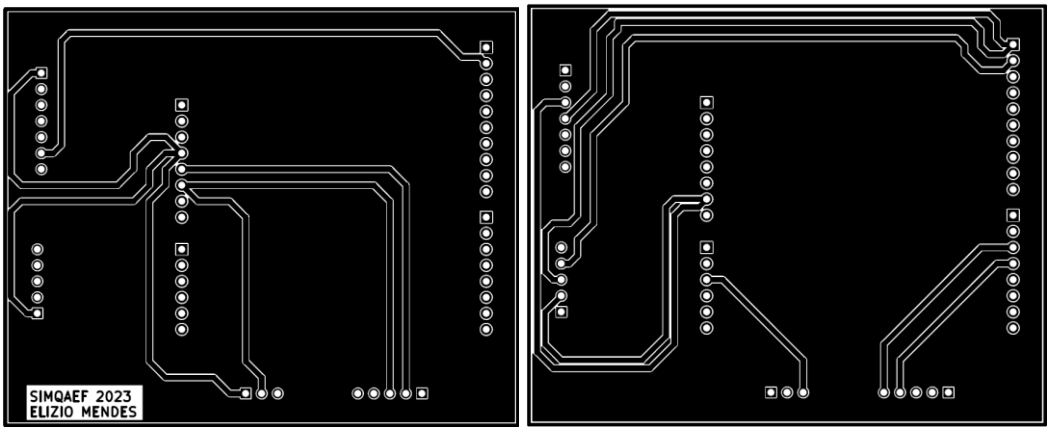


***Figura 23:*** *Layout para impressão da camada superior e inferior, respetivamente.*

Fonte: Autoria própria.

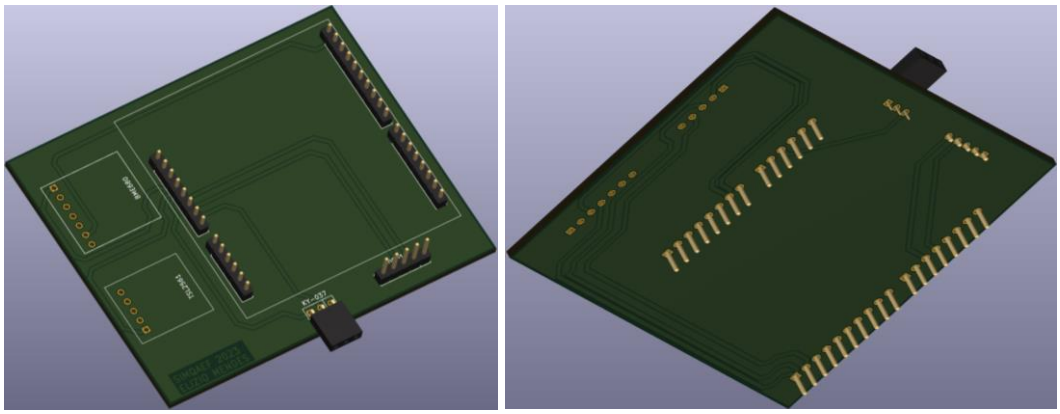

***Figura 24:*** *Visualização 3D da placa no KiCAD.*

Fonte: Autoria própria.

# V. TESTES E RESULTADOS

Neste capítulo são apresentados os testes e resultados de todo processo prático obtidos com esta dissertação. Seguidamente, serão analisados e discutidos de modo a esclarecer todos os procedimentos. Este capítulo está dividido em três secções: a primeira apresenta a funcionalidade do *firmware* desenvolvido na ESP32, a segunda as funcionalidades das aplicações desenvolvidas no Node-RED e a terceira apresenta o resultado do protótipo final.

Antes de mais, pretende-se mostrar através de uma fotografia feita pelo autor (figura 25), o funcionamento do todo o sistema montado durante a fase da realização de testes.

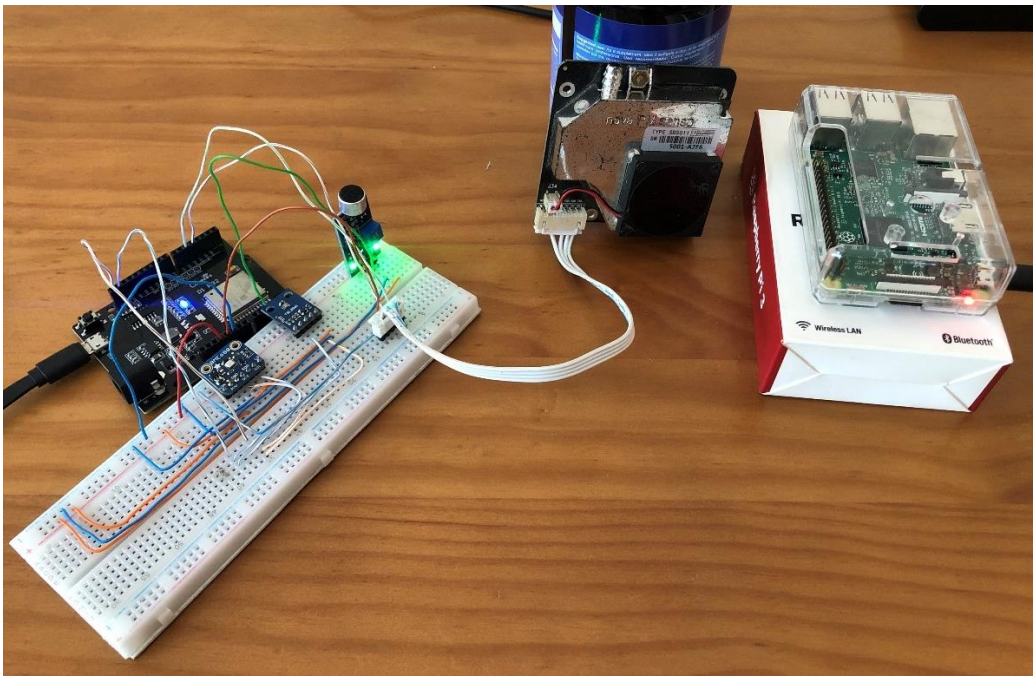


***Figura 25:*** *Funcionamento do sistema.*

Fonte: Autoria própria.

## 5.1 Funcionalidade do Firmware da ESP32

A figura 26 mostra, no *Serial Monitor* do Arduíno IDE, a impressão dos valores dos parâmetros medidos pelos demais sensores e, a publicação desses valores em forma de tópicos pelo protocolo de comunicação MQTT (*broker* Mosquitto).

```
Temperature = 27 °C
Humidity = 46 %
Pressure = 940.67 hPa
Gas Resistance = 143.74 KOhm
Noise Value = 0
Luminosity = 128 lux
PM2.5 = 7.50 µg/m³
PM10 =  8.30 µg/m³

Publishing on topic esp/SIMQAEF/temperature at QoS 1, packetId: 153Message: 27
Publishing on topic esp/SIMQAEF/humidity at QoS 1, packetId 154: Message: 46
Publishing on topic esp/SIMQAEF/pressure at QoS 1, packetId 155: Message: 940.67
Publishing on topic esp/SIMQAEF/gas at QoS 1, packetId 156: Message: 143.74
Publishing on topic esp/SIMQAEF/noise at QoS 1, packetId 157: Message: 0
Publishing on topic esp/SIMQAEF/luminosity at QoS 1, packetId 158: Message: 128
Publishing on topic esp/SIMQAEF/p25 at QoS 1, packetId 159: Message: 7.50
Publishing on topic esp/SIMQAEF/p10 at QoS 1, packetId 160: Message: 8.30

Publish acknowledged.  packetId: 153
Publish acknowledged.  packetId: 154
Publish acknowledged.  packetId: 155
Publish acknowledged.  packetId: 156
Publish acknowledged.  packetId: 157
Publish acknowledged.  packetId: 158
Publish acknowledged.  packetId: 159
Publish acknowledged.  packetId: 160
```

***Figura 26:*** *Impressão e publicação dos valores no* Serial Monitor *(Arduíno IDE).*

Fonte: Autoria própria.

## 5.2 Aplicação Node-RED

Como visto anteriormente, o intuito da utilização do Node-RED consistia na construção de aplicações capazes de monitorar os parâmetros medidos em tempo real, enviar uma mensagem de alerta por email em caso de situações que não favorecem uma boa qualidade de ar numa sala de acordo com as legislações portuguesas e, por fim, um base de dados onde esses valores seriam salvaguardas para que sejam analisados futuramente.

É de salientar que, relativamente as aplicações do Node-RED, todos os objetivos assumidos no início do projeto foram cumpridos.

### 5.2.1 Funcionalidade do sistema de monitorização

Nessa secção serão apresentados os resultados obtidos na UI (*User Interface*) fornecida pelo Node-RED. A UI pode ser acesso através de qualquer computador ou dispositivo móvel conectado à mesma rede *Wi-Fi* que o Raspberry Pi, no qual o node-RED está sendo executado, utilizando o endereço IP do mesmo, por exemplo, x.x.x.x:1880/ui.

Ao entrar na UI, o administrador ou utilizador deparar-se-á com uma página inicial onde se acompanha a visão geral, ou seja, de todos valores dos parâmetros do ambiente a ser medido em tempo real (mostrada na figura 27).

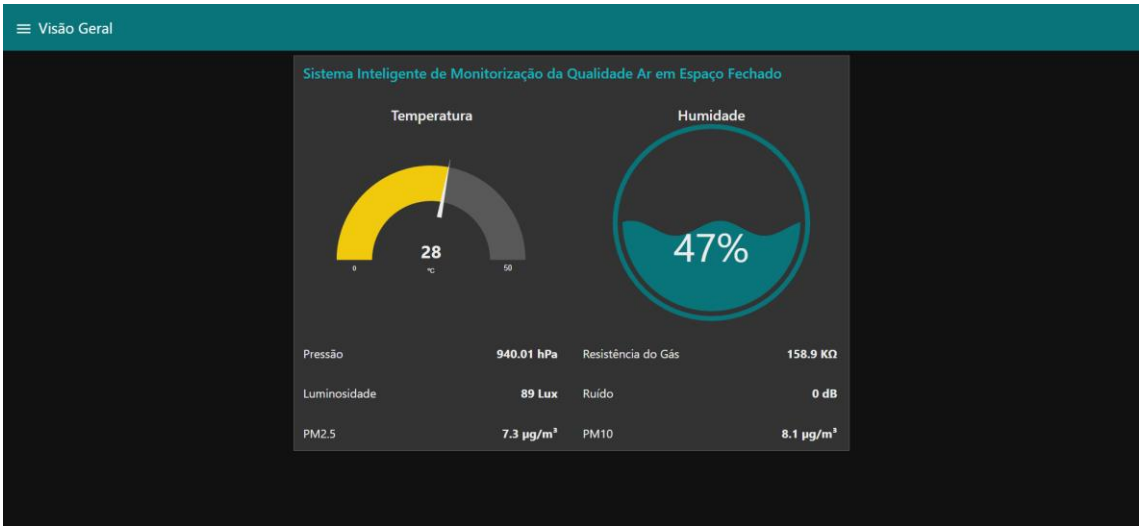


***Figura 27:*** *Página inicial da UI do Node-RED.*

Fonte: Autoria própria.

A partir dessa página inicial, o utilizador já pode ter acesso a mais duas páginas individuais ao clicar no ícone de menu no canto superior esquerdo, abrindo uma aba mostrada na figura 28.

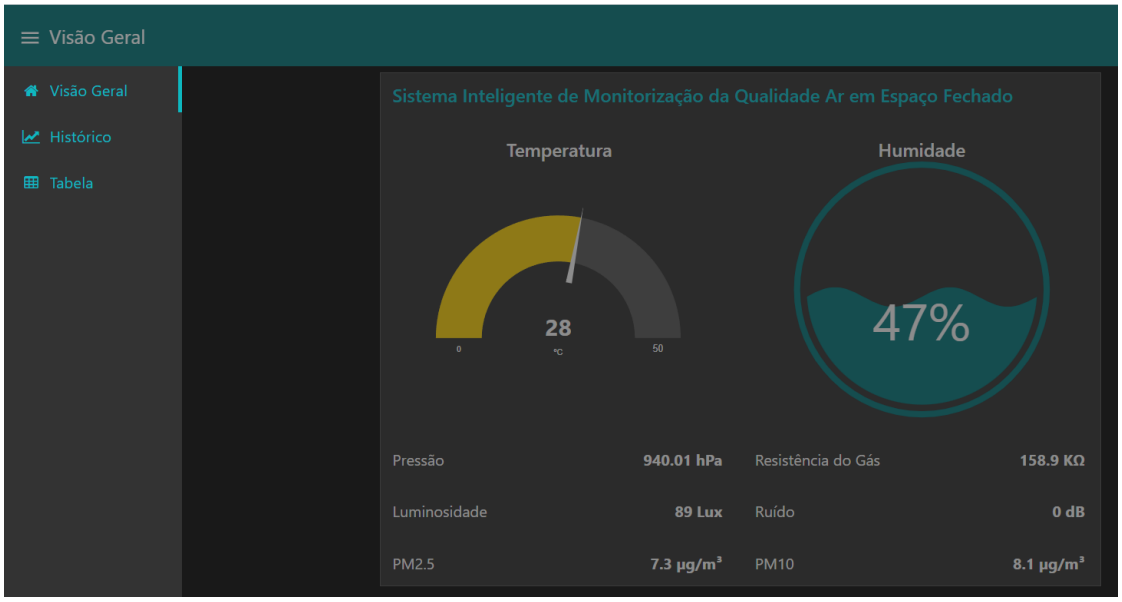


***Figura 28:*** *Aba de opções da UI do Node-RED.*

Fonte: Autoria própria.

Ao clicar no item Histórico, o utilizador será levado à uma página que mostra o histórico de alguns dos parâmetros em forma de gráficos de linha (figura 29) e, ao clicar no item Tabela, o utilizador será levado à uma página que mostra a tabela do base de dados criado anteriormente (figura 30).

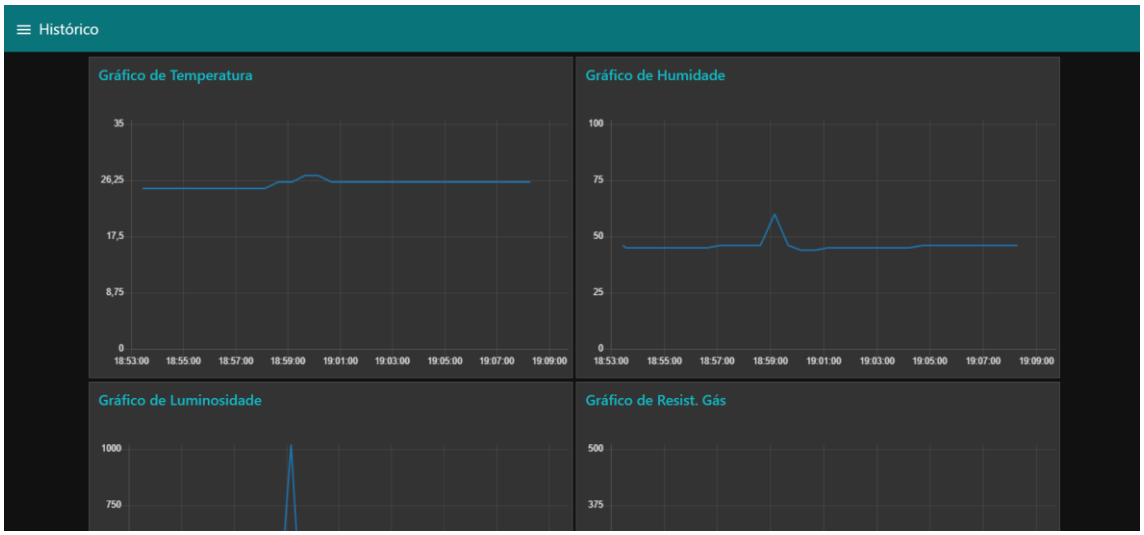


***Figura 29:*** *Página de Histórico da UI do Node-RED.*

Fonte: Autoria própria.

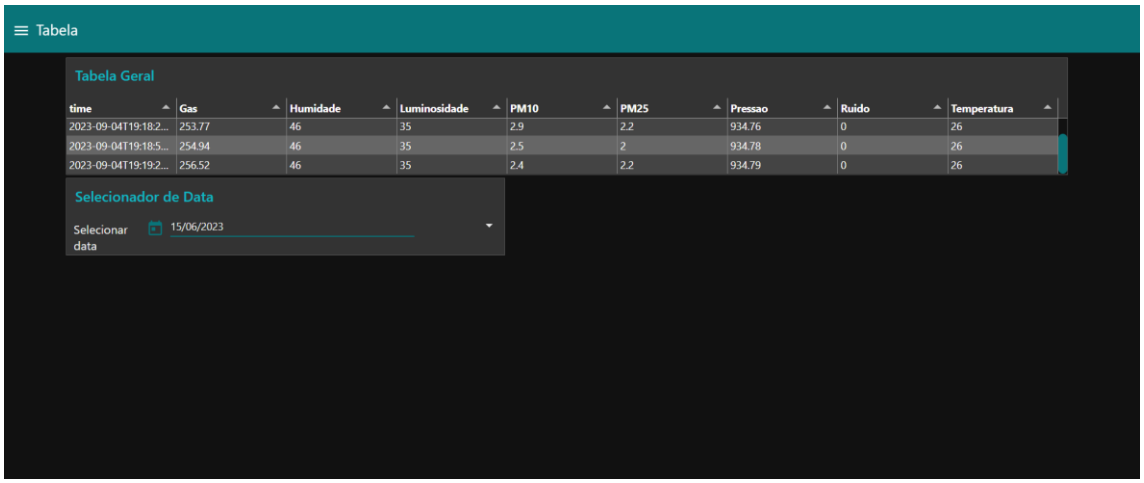


***Figura 30:*** *Página de Tabela da UI do Node-RED.*

Fonte: Autoria própria.

### 5.2.2 Funcionalidade do sistema de alarme e notificação

Analisando os gráficos da figura 31, pode-se observar que num determinado instante, a temperatura na sala onde foram feitos os testes ultrapassa os 25ºC, logo um alarme é enviado. O mesmo acontece com a humidade, nesse caso encontra-se abaixo do intervalo considerado ideal.

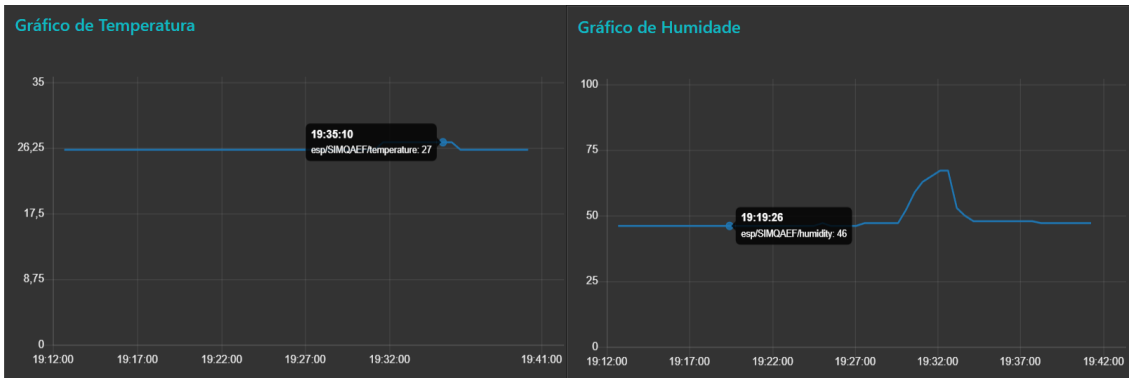


***Figura 31:*** *Gráficos de Temperatura e Humidade ao longo do tempo.*

Fonte: Autoria própria.

A figura 32 mostra uma sequência de diferentes notificações de alertas recebidas pelo correio eletrónico do administrador. Nota-se que o email recebido contem data e hora da ocorrência acompanhado do valor atual do parâmetro medido no local onde os testes foram realizados.

Analisando a imagem mais a esquerda, a temperatura atual é de 28ºC, o que excede os 25ºC que é o valor máximo para uma boa qualidade de ar numa sala segundo a legislação nacional (Artigo 11.º do Decreto-Lei n.º 243/86 de 20 de agosto). Visto isso, é enviado um email alertando que a temperatura no local está alta e, assim funciona com os demais parâmetros medidos, de acordo com os limites de cada um.

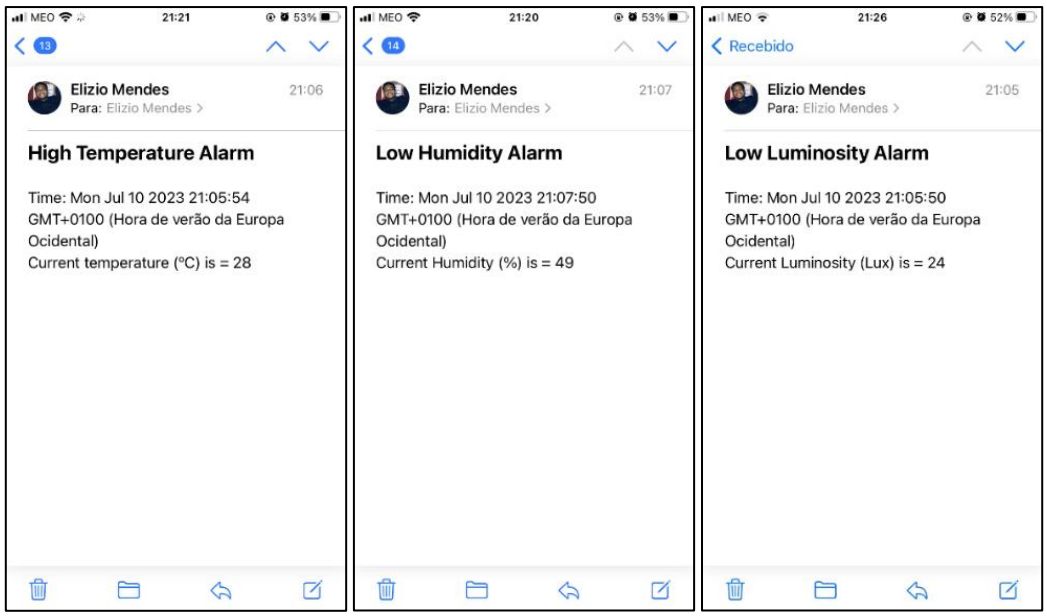


***Figura 32:*** *Sistema de alarme e notificação por email.*

Fonte: Autoria própria.

### 5.2.3 Funcionalidade do sistema de base de dados

Com os dados armazenados no base de dados, é possível realizar a consulta e a análise do mesmo afim de construir o painel de controlo deste trabalho. Pela aplicação desenvolvida no Node-RED, o base de dados é atualizado a cada 5 minutos, ou seja, a cada 5 minutos são armazenados valores dos parâmetros medidos.

A figura 33 mostra o base de dados em funcionamento no Raspberry Pi, a qual foi atribuído o nome de SIMQAEF, com a sua respetiva tabela SIMQAEF_tab.

```
raspuser@raspberrypi:~ $ influx
Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
> use SIMQAEF
Using database SIMQAEF
> select * from SIMQAEF_tab
name: SIMQAEF_tab
time                Gas    Humidade Luminosidade PM10 PM2.5 Pressao Ruido Temperatura
----                ---    -------- ------------ ---- ----- ------- ----- -----------
1689768585910960911 31.75  47       124          10   5.5   940.71  0     28
1689768616283997476 34.64  47       122          7.5  6.3   940.73  0     27
1689768646282704786 37.94  47       131          8.7  6.7   940.73  0     27
1689768720645433556 152.3  47       133          8.9  6.4   940.7   10    28
1689768750652257027 139.46 47       130          10.1 6.7   940.7   0     27
1689768781145884404 132.98 47       160          7.7  6.5   940.7   0     27
1689768811658299580 128.91 46       258          8.3  6.9   940.69  0     27
1689768840197697369 127.17 46       130          8.2  7     940.69  0     27
1689768872658053427 126.88 46       132          7.8  6.8   940.69  0     27
1689768903164397731 126.21 46       131          10.5 7     940.67  0     27
1689768933669842714 126.88 46       129          8.8  6.8   940.67  16    27
1689768962703246126 127.17 46       124          7.4  6.6   940.67  0     27
1689768993711542942 128.23 46       126          7.8  7     940.67  2     27
1689769024702278684 128.62 46       127          8.5  7.1   940.67  0     27
1689769055223682732 129.91 46       126          8.3  7.4   940.67  0     27
1689769083748412687 131.63 46       126          8.6  7     940.68  0     27
1689769114260050786 134.03 46       127          8.9  7     940.67  0     27
1689769146740545484 134.78 46       127          8.1  7     940.68  0     27
1689769177237623657 136.31 46       130          8.8  7.2   940.68  0     27
1689769206761107115 137.98 46       129          8.7  6.8   940.68  31    27
1689769235305003286 139.81 46       124          8.6  7.3   940.68  0     27
1689769268282024253 141.93 46       123          8.7  6.9   940.68  30    27
1689769300248249534 143.74 46       128          8.3  7.5   940.67  0     27
1689769328792749426 145.85 46       127          10.5 7.4   940.66  0     27
1689769359784395562 146.36 46       123          8.5  7.3   940.65  0     27
1689769390290714607 147.77 46       127          7.8  6.9   940.65  0     27
1689769420798754997 150.27 46       126          8.4  6.9   940.65  0     27
1689769450329628344 151.75 46       125          8.1  6.9   940.64  0     27
1689769478863825983 152.99 46       124          7.8  7     940.63  28    27
1689769511837949444 153.41 46       124          9.3  7.1   940.62  0     27
```

***Figura 33:*** *Base de dados no RPI.*

Fonte: Autoria própria.

## 5.3 Protótipo Final

Após o código implementado no microcontrolador ESP32 estar a funcionar corretamente, como referido anteriormente, foi desenvolvido no KiCad um esquema de todo o circuito elétrico, com a finalidade de obter um *layout* da PCI. Para que sua

produção seja possível, é necessário criar arquivos no formato Gerber[1] para ser impressa na máquina CNC.

Todavia não foi possível fabricar a placa pelo fato de não haver uma máquina CNC ou outro método de fabrico de PCI disponível na instituição. Como alternativa, projetou-se a sua montagem nos *softwares* SolidWorks e KiCAD para ter uma noção de como o protótipo ficaria no modo real.

Na figura 34, pode-se observar uma sequência de imagens da PCI já com os respetivos sensores soldados à placa, tendo atenção que o sensor SDS011 é conectada na placa através dos seus fios apropriados.

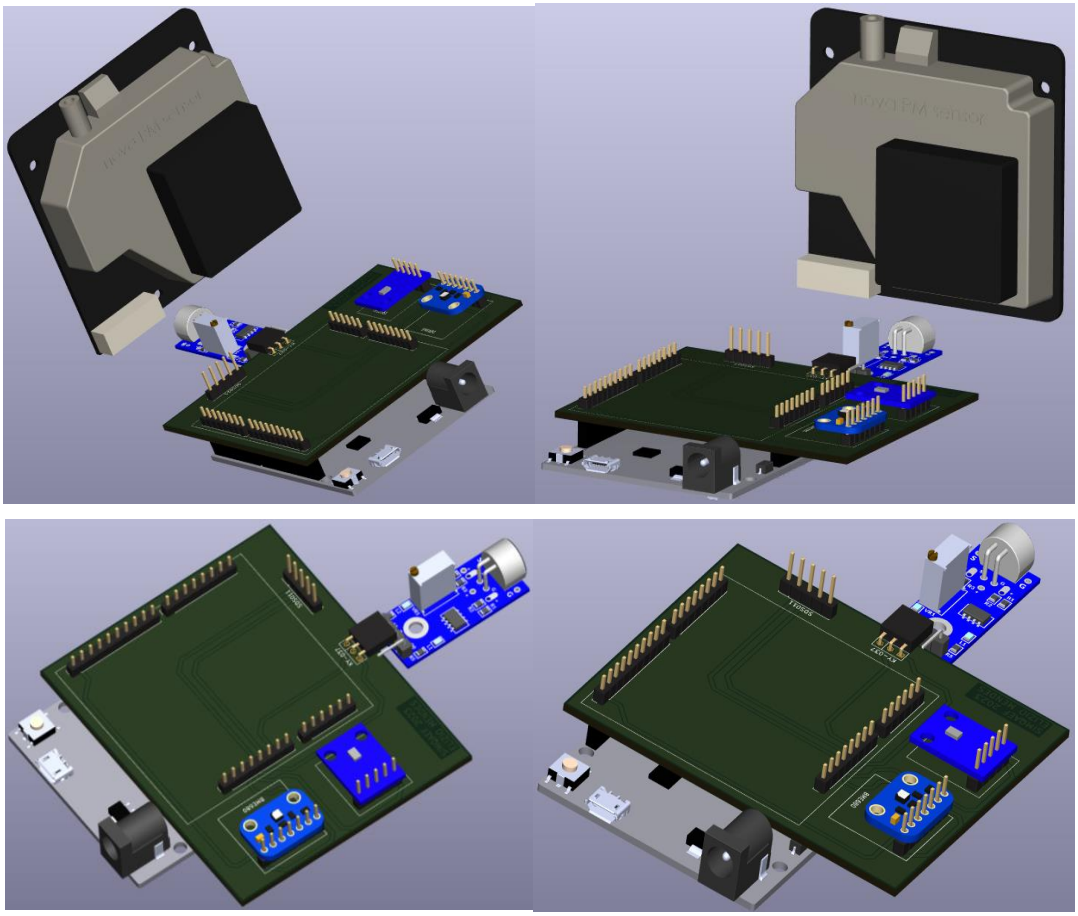

***Figura 34:*** *Protótipo final.*

Fonte: Autoria própria.

[1] O formato Gerber é um formato vetorial ASCII aberto para imagens binárias 2D. É o padrão usado pelo software da indústria de PCI [43].

# VI. CONSIDERAÇÕES FINAIS

O estudo da qualidade de ar em espaços interiores, principalmente em locais de trabalho, tem grande relevância uma vez que as concentrações de certos poluentes em ambientes internos podem ser maiores do que no ar externo devido à presença de diferentes fontes emissão e inadequada ventilação, entre outros fatores.

Partindo do pressuposto de garantir maior QAI, reduzindo assim os riscos e problemas à saúde de pessoas que frequentam um dado espaço fechado, o protótipo proposto mostrou-se ser confiável, precisa e de baixo custo, alcançando parcialmente o objetivo de monitorizar os parâmetros de forma inteligente e alertar o utilizador de situações extremas, através dos demais sensores, placa ESP32 Wemos D1 R32, Raspberry Pi 3B e Node-RED.

Durante a fase experimental, foram feitos alguns testes num curto período de tempo, não mais de duas horas, para avaliar funcionalidades do sistema. Pode-se afirmar que o sistema funciona de acordo com a metas traçadas inicialmente.

Todavia, deparou-se com um problema que afeta o funcionamento de um dos parâmetros do sistema. Pois, inicialmente o sensor KY-037 funcionava sem qualquer problema, mas ao adiciona-lo juntamente com os outros sensores na placa ESP32, o valor da leitura de ruído obtido, maior parte das vezes, é sempre zero. Fez-se teste do funcionamento do sensor com um osciloscópio, juntamente com os restantes sensores e, constou-se que funciona. Pensa-se que as bibliotecas instaladas no Arduíno IDE podem estar a interferir no resultado do mesmo.

Terminando, com base nas soluções IoT integradas, é possível desenvolver inúmeros projetos de boas precisões e de baixo custo, com elevada utilidade na vida quotidiana relacionadas com a área de automação, desde residência até industrial.

## 6.1 Trabalhos Futuros

Este projeto estende diversas aplicações para temas relacionadas com o monitoramento da qualidade de ar em espaços internos. Como trabalhos futuros, propõe-se o alargamento do estudo para diferentes pontos, tais como:

- Inclusão de novos sensores, tais como, sensores de dióxido de carbono ($CO_2$), monóxido de carbono (CO). Houve a ideia de utilizar estes sensores para este projeto, porém não foi possível devido a disponibilidade das mesmas;
- Resolução do problema de leitura do valor do sensor KY-037 no ESP32 juntamente com os demais sensores;
- Desenvolver um aplicativo em Node-RED para tornar o sistema de monitoramento do sistema mais atraente e mais interativo com o utilizador;
- Implementar uma aplicação de controle para que o sistema possa acionar atuadores, como sistemas AVAC, de forma automática e inteligente, buscando maior eficiência.

# REFERÊNCIAS


[1] Zhang Y., *"Indoor Air Quality Engineering"*, Boca Raton: CRC Press, 2004.

[2] Sundell J., *"INDOOR AIR, On the history of indoor air quality and health"*, p. 51–58, 2004.

[3] Walsh P. J., Dudney C. S., Copenhaver E. D., *"Indoor Air Quality"*, Boca Taton, Florida: CRC Press, 2000.

[4] Seguel J. M., Merrill R., Seguel D., Campagna A. C., *"Indoor Air Quality"*, American Journal of Lifestyle Medicine, vol. 11, nº 4, pp. 284-363, 2017.

[5] Jones A., *"Atmospheric Environment"*, Indoor air quality and health, vol. 33, nº 28, pp. 4535-4564, 18 May 1999.

[6] Wolkoff P., *"International Journal of Hygiene and"*, Indoor air humidity, air quality, and health – An overview, vol. 221, nº 3, pp. 376-390, April 2018.

[7] e. a. WHO, *"World health statistics 2018: monitoring health for the sdgs, sustainable"*, World Health Organization, Luxembourg, 2018.

[8] EPA, *"United States Environmental Protection Agency"*, http://www.epa.gov/iaq/largebldgs/i-beam/pdfs/text_modules_fundamentals.pdf. Acessado em março, 2023.

[9] Environmental Protection Agency, *"An introduction to indoor air quality—Biological Pollutants"*, Washington, DC, USA, November 2007.

[10] EPA, *"United States Environmental Protection Agency"*. http://www.epa.gov/airquality/particlepollution/. Acessado em março, 2023.

[11] Portugal, 2006, Decreto-Lei n.º 78/2006, de 4 de Abril, Diário da República, 1.ª série-A — N.º 67 — 4 de Abril de 2006.

[12] UA, *"Legislação QAI"*, https://www.ua.pt/pt/idad/page/15241. Acessado em março, 2023.

[13] Portugal, 2022, Despacho n.º 1618/2022, de 9 de Fevereiro, Diário da República, 2.ª série — N.º 28 — 9 de fevereiro de 2022.

[14] UA, *"O IDAD - Instituto do Ambiente e Desenvolvimento"*, https://www.ua.pt/pt/idad/page/9171. Acessado em março, 2023.

[15] UA, *"Qualidade do ar em espaços interiores"*, https://www.ua.pt/pt/idad/qualidade_do_ar_interior. Acessado em março, 2023.

[16] Portugal, 2021, Portaria n.º 138-G/2021, de 1 de Julho, Diário da República, 1.ª série — N.º 126 — 1 de julho de 2021.

[17] Portugal, 2022, Despacho n.º 1618/2022, de 9 de Fevereiro, Diário da República, 2.ª série — N.º 28 — 9 de fevereiro de 2022.

[18] Portugal, 1986, Decreto-Lei n.º 243/86, de 20 de Agosto, Diário da República, 1.ª série — N.º 190 — 20-8-1986.

[19] Bosch, 2017, *"BME680"*, Datasheet. Recuperado de: https://cdn-shop.adafruit.com/product-files/3660/BME680.pdf. Acessado em março, 2023.

[20] Adafruit, 2023, *"BME680"*, Datasheet. Recuperado de: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-bme680-humidity-temperature-barometic-pressure-voc-gas.pdf. Acessado em março, 2023.

[21] Adafruit, 2009, *"TSL2561"*, Datasheet. Recuperado de: https://cdn-shop.adafruit.com/datasheets/TSL2561.pdf. Acessado em março, 2023.

[22] AZ-Delivery, *"D1 R32"*, Datasheet. Recuperado de: https://www.halloweenfreak.de/arduino/pdfs/D1_R32_ENG.pdf. Acessado em março, 2023.

[23] Espressif Systems, 2023, "*ESP32-WROOM-32D & ESP32-WROOM-32U*", Datasheet. Recuperado de: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf. Acessado em março, 2023.

[24] JOY-IT, 2020, *"KY-037"*, Datasheet. Recuperado de: https://pdf1.alldatasheet.com/datasheet-pdf/view/1284506/JOY-IT/KY037.html. Acessado em março, 2023.

[25] Portugal, 2007, Decreto-Lei n.º 9/2007, de 17 de Janeiro, Diário da República, 1.ª série — N.º 12 — 17 de Janeiro de 2007.

[26] Amazon, *"what is MQTT?"*, https://aws.amazon.com/pt/what-is/mqtt/. Acessado em abril, 2023.

[27] Shah, Z., Noor, F., Khan, M. A., Nazir, S., Qamar, U., & Sher, M. (2017). *"MQTT Protocol for Internet of Things: A Review"*. International Journal of Advanced Computer Science and Applications (IJACSA), 8(10), 367-373. DOI: 10.14569/IJACSA.2017.081044.

[28] Amazon, *"what is IoT?"*, https://aws.amazon.com/pt/what-is/iot/. Acessado em abril, 2023.

[29] Mosquitto, *"Eclipse Mosquitto"*, https://mosquitto.org/. Acessado em abril, 2023.

[30] Node-RED, *"Nodered"*, https://nodered.org/. Acessado em abril, 2023.

[31] KiCad, *"About KiCad"*, https://www.kicad.org/about/kicad/. Acessado em julho, 2023.

[32] Raspberrypi, *"Raspberry Pi 3 Model B"*, https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Acessado em março, 2023.

[33] MakerHero, *"Primeiros passos com Raspberry Pi e Linux"*, https://www.makerhero.com/blog/primeiros-passos-raspberry-pi-e-linux/. Acessado em março, 2023.

[34] Raspberrypi, *"Raspberry Pi"*, https://www.raspberrypi.com/. Acessado em março, 2023.

[35] Dawoud, D. S., Dawoud, P., 2020, *"Serial Communication Protocols and Standards"*, River Publishers, New Yord, NY 10017, USA.

[36] Mankar J., Darode C., Trivedi K., Kanoje M., Shahare P., 2014, *"Review Of I2c Protocol"*, International Journal of Research in Advent Technology, E-ISSN: 2321–9637, Volume 2, Issue .

[37] Xu K., Cui K., Young L., Wang Y., Hsieh Y., Wan S., Zhang J., 2020, *"Air Quality Index, Indicatory Air Pollutants and Impact of COVID-19 Event on the Air Quality near Central China"*, Taiwan Association for Aerosol Research, Aerosol and Air Quality Research, 20: 1204–1221.

[38] Sun Y., Zuo J., Zou J., et al., *"Air Quality Index and Personal Exposure Assessment: A Review"*, Environment International, 2016;89-90:234-250, doi: 10.1016/j.envint.2016.02.013.

[39] Santos, A. R., 2019, *"Qualidade do Ar Interior em Academias De Ginásticas: Parâmetros Físicos, Químicos e Microbiológicos"*, teste de doutoramento, Universidade de Ribeirão Preto, Ribeirão Preto.

[40] Springer Link, *"Cloud Computing: An Overview"*, https://link.springer.com/chapter/10.1007/978-3-642-10665-1_63. Acessado em junho, 2023.

[41] Nova Fitness Co., 2015, *"Laser PM2.5 Sensor specification SDS011 V1.3"*, Datasheet. Recuperado de: https://cdn-reichelt.de/documents/datenblatt/X200/SDS011-DATASHEET.pdf. Acessado em junho, 2023.

[42] Influxdata, *"InfluxDB"*, https://www.influxdata.com/. Acessado em junho, 2023.

[43] UV LIGHT, *"O que são arquivos Gerber?"*, https://uvlight-solutions.com/home/o-que-sao-arquivos-gerber-.html. Acessado em julho, 2023.

[44] ISO, *"ISO 7730: 2005 (en)"*, https://www.iso.org/obp/ui/es/#iso:std:iso:7730:ed-3:v1:en. Acessado em abril, 2023.

[45] Valosto, *"Light and lighting - Lighting of work places - Part 1: Indoor work places"*, https://www.valosto.com/tiedostot/prEN%2012464-1.pdf. Acessado em abril, 2023.

# APÊNDICE A

## Firmware para ESP32

```
// Sistema Inteligente de Monitorização da Qualidade do Ar em Espaços Fechados (SIMQAEF)
// Elizio Cardoso Mendes (46814)
// Dissertação - Eng. Industrial Eletro - Ano Letivo 2022/2023

// ----------------- Inclusao das bibliotecas -------------------------------- //

#include <WiFi.h>
#include <AsyncMqttClient.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"
#include <Adafruit_TSL2561_U.h>
#include "SDS011.h"

// ------------------------------------------------------------------------- //

#define MIC_PIN A0  // Pino analógico para leitura do sinal do KY037

#define SEALEVELPRESSURE_HPA (1013.25)  // define valor da pressao a nivel do mar

#define RX_PIN 16  // Pino RX do ESP32 (conectado ao TX do SDS011)
#define TX_PIN 17  // Pino TX do ESP32 (conectado ao RX do SDS011)

// ------------------  Configuracao do Wi-Fi ------------------------------- //

#define WIFI_SSID "SIMQAEF_TEST_MODULE"
#define WIFI_PASSWORD "mendes94"

// ------------------  Configuracao do MQTT -------------------------------- //

#define MQTT_HOST IPAddress(192, 168, 137, 43)  // Raspberry Pi Mosquitto MQTT Broker
#define MQTT_PORT 1883

// ------------------- Definicao dos Topicos do MQTT ------------------------- //

#define MQTT_PUB_TEMP "esp/SIMQAEF/temperature"
#define MQTT_PUB_HUM "esp/SIMQAEF/humidity"
```

```
#define MQTT_PUB_PRES "esp/SIMQAEF/pressure"

#define MQTT_PUB_GAS "esp/SIMQAEF/gas"

#define MQTT_PUB_NOISE "esp/SIMQAEF/noise"

#define MQTT_PUB_LUM "esp/SIMQAEF/luminosity"

#define MQTT_PUB_P25 "esp/SIMQAEF/p25"

#define MQTT_PUB_P10 "esp/SIMQAEF/p10"


// ------------------- Criacao do objeto dos sensores BME680 e TSL2561 ----------------------
----------- //


Adafruit_BME680 bme;  // I2C

Adafruit_TSL2561_Unified tsl = Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 12345);


// ------------------- Variaveis para manter as leituras dos sensores ----------------------
---------- //


int temperature;

int humidity;

float pressure;

float gasResistance;

int noise;

int luminosity;

float p25;

float p10;


int error;

SDS011 my_sds;


AsyncMqttClient mqttClient;

TimerHandle_t mqttReconnectTimer;   // Temporizador para reconectar ao intermediário MQTT
quando desconectar

TimerHandle_t wifiReconnectTimer;  // Temporizador para reconectar ao Wi-Fi (roteador) quando
desconectar


unsigned long previousMillis = 0;

const long interval = 10000;  // Intervalo no qual publicar as leituras do sensor


void getBME680Readings() {

  // BME680 inicia a medicao.

  unsigned long endTime = bme.beginReading();

  if (endTime == 0) {

    Serial.println(F("Failed to begin reading :("));
```

```
    return;
  }
  if (!bme.endReading()) {
    Serial.println(F("Failed to complete reading :("));
    return;
  }
  temperature = bme.temperature;
  pressure = bme.pressure / 100.0;
  humidity = bme.humidity;
  gasResistance = bme.gas_resistance / 1000.0;
}

// -------------------- Conectar ESP32 ao seu Wi-Fi ------------------------------- //
void connectToWifi() {
  Serial.println("Connecting to Wi-Fi...");
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
}

// -------------------- Conectar ESP32 ao broker MQTT ------------------------------- //
void connectToMqtt() {
  Serial.println("Connecting to MQTT...");
  mqttClient.connect();
}

/* WiFiEvent() responsavel por lidar com os eventos Wi-Fi.
   Conexao bem-sucedida com o roteador e o broker MQTT, imprime o endereço IP do ESP32.
   Se a conexao for perdida, inicia um timer e tenta reconectar.
*/
void WiFiEvent(WiFiEvent_t event) {
  Serial.printf("[WiFi-event] event: %d\n", event);
  switch (event) {
    case SYSTEM_EVENT_STA_GOT_IP:
      Serial.println("WiFi connected");
      Serial.println("IP address: ");
      Serial.println(WiFi.localIP());
      connectToMqtt();
      break;
    case SYSTEM_EVENT_STA_DISCONNECTED:
      Serial.println("WiFi lost connection");
      xTimerStop(mqttReconnectTimer, 0);  // Nao reconectar ao MQTT enquanto se reconecta
ao Wi-Fi
```

```
      xTimerStart(wifiReconnectTimer, 0);
      break;
  }
}

void onMqttConnect(bool sessionPresent) {
  Serial.println("Connected to MQTT.");
  Serial.print("Session present: ");
  Serial.println(sessionPresent);
}

void onMqttDisconnect(AsyncMqttClientDisconnectReason reason) {
  Serial.println("Disconnected from MQTT.");
  if (WiFi.isConnected()) {
    xTimerStart(mqttReconnectTimer, 0);
  }
}

void onMqttPublish(uint16_t packetId) {
  Serial.print("Publish acknowledged.");
  Serial.print("  packetId: ");
  Serial.println(packetId);
}

void setup() {

  Serial.begin(115200);
  Serial.println();

  /********************** BME680 ********************/

  // --------------- Inicialização do sensor BME680 ----------------- //
  while (!Serial)
    ;
  Serial.println(F("Teste do sensor BME680"));
  if (!bme.begin()) {
    Serial.println(F("Nao foi encontrado o sensor BME680, verifique a ligacao!!"));
    while (1)
      ;
  }
```

```
  // -------------- Configuracao do sensor BME680 ----------------- //
  bme.setTemperatureOversampling(BME680_OS_8X);
  bme.setHumidityOversampling(BME680_OS_2X);
  bme.setPressureOversampling(BME680_OS_4X);
  bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
  bme.setGasHeater(320, 150);  // 320*C for 150 ms

  /********************** KY037 *******************/

  pinMode(MIC_PIN, INPUT);

  /********************** TSL2561 *******************/

  // -------------- Inicializacao do sensor TSL2561 ----------------- //
  Serial.println("Teste do Sensor TSL2561");
  if (!tsl.begin()) {
    Serial.print("Sensor TSL2561 nao detetado ... Verificar ligacao!");
    while (1)
      ;
  }
  tsl.enableAutoRange(true);
  tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_13MS);
  Serial.println("");

  /********************** SDS011 *******************/
  my_sds.begin(16, 17);

  // Temporizadores para reconectar MQTT e a Wi-Fi sejam, caso a conexao seja perdida
  mqttReconnectTimer = xTimerCreate("mqttTimer", pdMS_TO_TICKS(2000), pdFALSE, (void*)0,
reinterpret_cast<TimerCallbackFunction_t>(connectToMqtt));
  wifiReconnectTimer = xTimerCreate("wifiTimer", pdMS_TO_TICKS(2000), pdFALSE, (void*)0,
reinterpret_cast<TimerCallbackFunction_t>(connectToWifi));

  WiFi.onEvent(WiFiEvent);

  mqttClient.onConnect(onMqttConnect);
  mqttClient.onDisconnect(onMqttDisconnect);
  //mqttClient.onSubscribe(onMqttSubscribe);
  //mqttClient.onUnsubscribe(onMqttUnsubscribe);
  mqttClient.onPublish(onMqttPublish);
```

```
  mqttClient.setServer(MQTT_HOST, MQTT_PORT);
  // If your broker requires authentication (username and password), set them below
  //mqttClient.setCredentials("REPlACE_WITH_YOUR_USER", "REPLACE_WITH_YOUR_PASSWORD");

  connectToWifi();
}

void loop() {

  unsigned long currentMillis = millis();

  // A cada X segundos (intervalo = X segundos), publica uma nova mensagem MQTT
  if (currentMillis - previousMillis >= interval) {

    previousMillis = currentMillis;  // Guarda a ultima vez que uma nova leitura foi publicada

    /********************** BME680 ********************/

    getBME680Readings();
    Serial.println();
    Serial.printf("Temperature = %.2i °C \n", temperature);
    Serial.printf("Humidity = %.2i %% \n", humidity);
    Serial.printf("Pressure = %.2f hPa \n", pressure);
    Serial.printf("Gas Resistance = %.2f KOhm \n", gasResistance);

    /******************** KY037 ********************/

    noise = analogRead(MIC_PIN);
    Serial.printf("Noise Value = %d \n", noise);

    /********************** TSL2561 ********************/

    sensors_event_t event;
    tsl.getEvent(&event);
    if (event.light) {
      luminosity = event.light;
      Serial.printf("Luminosity = %.2i lux \n", luminosity);
    } else {
      Serial.println("Sensor overload");
    }
```

```
    /********************** SDS011 ********************/


    error = my_sds.read(&p25, &p10);

    if (!error) {

      Serial.print("PM2.5 = " + String(p25));

      Serial.println(" µg/m³");

      Serial.print("PM10 =  " + String(p10));

      Serial.println(" µg/m³");

      Serial.println();

    }


    //Publicar uma mensagem MQTT no topico esp/SIMQAEF/temperature

    uint16_t     packetIdPub1     =     mqttClient.publish(MQTT_PUB_TEMP,     1,     true,
String(temperature).c_str());

    Serial.printf("Publishing  on  topic  %s  at  QoS  1,  packetId:  %i",  MQTT_PUB_TEMP,
packetIdPub1);

    Serial.printf("Message: %.2i \n", temperature);


    // Publicar uma mensagem MQTT no topico esp/SIMQAEF/humidity

    uint16_t     packetIdPub2     =     mqttClient.publish(MQTT_PUB_HUM,     1,     true,
String(humidity).c_str());

    Serial.printf("Publishing  on  topic  %s  at  QoS  1,  packetId  %i:  ",  MQTT_PUB_HUM,
packetIdPub2);

    Serial.printf("Message: %.2i \n", humidity);


    // Publicar uma mensagem MQTT no topico esp/SIMQAEF/pressure

    uint16_t     packetIdPub3     =     mqttClient.publish(MQTT_PUB_PRES,     1,     true,
String(pressure).c_str());

    Serial.printf("Publishing  on  topic  %s  at  QoS  1,  packetId  %i:  ",  MQTT_PUB_PRES,
packetIdPub3);

    Serial.printf("Message: %.2f \n", pressure);


    // Publicar uma mensagem MQTT no topico esp/SIMQAEF/gas

    uint16_t     packetIdPub4     =     mqttClient.publish(MQTT_PUB_GAS,     1,     true,
String(gasResistance).c_str());

    Serial.printf("Publishing  on  topic  %s  at  QoS  1,  packetId  %i:  ",  MQTT_PUB_GAS,
packetIdPub4);

    Serial.printf("Message: %.2f \n", gasResistance);


    // Publicar uma mensagem MQTT no topico esp/SIMQAEF/noise

    uint16_t     packetIdPub5     =     mqttClient.publish(MQTT_PUB_NOISE,     1,     true,
String(noise).c_str());

    Serial.printf("Publishing  on  topic  %s  at  QoS  1,  packetId  %i:  ",  MQTT_PUB_NOISE,
packetIdPub5);
```

```
    Serial.printf("Message: %d \n", noise);


    // Publicar uma mensagem MQTT no topico esp/SIMQAEF/luminosity

    uint16_t     packetIdPub6     =     mqttClient.publish(MQTT_PUB_LUM,     1,     true,
String(luminosity).c_str());

    Serial.printf("Publishing  on  topic  %s  at  QoS  1,  packetId  %i:  ",  MQTT_PUB_LUM,
packetIdPub6);

    Serial.printf("Message: %.2i \n", luminosity);


    // Publicar uma mensagem MQTT no topico esp/SIMQAEF/p25

    uint16_t packetIdPub7 = mqttClient.publish(MQTT_PUB_P25, 1, true, String(p25).c_str());

    Serial.printf("Publishing  on  topic  %s  at  QoS  1,  packetId  %i:  ",  MQTT_PUB_P25,
packetIdPub7);

    Serial.printf("Message: %.2f \n", p25);


    // Publicar uma mensagem MQTT no topico esp/SIMQAEF/p10

    uint16_t packetIdPub8 = mqttClient.publish(MQTT_PUB_P10, 1, true, String(p10).c_str());

    Serial.printf("Publishing  on  topic  %s  at  QoS  1,  packetId  %i:  ",  MQTT_PUB_P10,
packetIdPub8);

    Serial.printf("Message: %.2f \n \n", p10);

  }

}
```