Universidade de Aveiro

Know yourself and your surroundings to live BETTER

Gonçalo Costa De Oliveira - 108405

Isaac David Delgado Santiago - 104327

Luís Barros Ferreira De Sousa - 108583

Nuno Matos Carvalho - 97783

Tomás Henriques Viana - 108445

Departamento de Eletrónica, Telecomunicações e Informática

Licenciatura em Engenharia de Computadores e Informática

Regente: Professor José Moreira

Orientador: Professor José Maria

8 de junho, 2025

Abstract

The aim of this project is to develop AirSense, a product designed to monitor indoor air quality. Using an ESP32 microcontroller connected to various environmental sensors, the system measures key parameters such as carbon dioxide levels, humidity, temperature, and particulate matter. These measurements are made accessible to users through the Home Assistant mobile app, which offers flexible visualization options such as graphs and real-time dashboards.

In the initial phase, we conducted research to determine the most suitable microcontroller and sensor combination, considering factors like size, power efficiency, cost, and measurement accuracy.

The finalized AirSense system features an ESP32-based sensor hub that communicates with a Raspberry Pi via Bluetooth Low Energy (BLE). The Raspberry Pi, running Docker containers, handles data processing and storage. Sensor data is forwarded to an InfluxDB time-series database and visualized through Home Assistant — both running as isolated services in Docker containers on the same Raspberry Pi. This architecture ensures local processing, ease of deployment, and seamless integration with smart home systems without relying on cloud infrastructure.

Key words: Microcontroller, Air quality of an indoor place, Home Assistant, Docker containers.

Table of Contents

1. Introduction

This report is part of the curricular unit Projeto em Engenharia de Computadores e Informática and describes the development of AirSense, a modular and automated air quality monitoring system aimed at improving environmental awareness in indoor spaces such as schools and homes. The system combines an ESP32 microcontroller with multiple environmental sensors to collect data on temperature, humidity, CO_2 levels and particulate matter (PM). The data is transmitted via Bluetooth Low Energy (BLE) to a Raspberry Pi, which processes and stores the information using InfluxDB and offers full integration with Home Assistant for visualization. The system setup is automated via a dedicated Android app, which allows users to scan a QR code and deploy the entire solution with minimal configuration.

The following sections present the context, motivation, goals, technical implementation, and future directions of the AirSense project.

1.1. Context – Importance of Indoor Air Quality

In recent years, increasing attention has been given to the quality of indoor air, as people spend a significant portion of their daily lives inside enclosed spaces such as homes, schools, and workplaces. According to the World Health Organization (WHO), poor indoor air quality can have serious health implications, contributing to respiratory issues, allergies, fatigue, and decreased cognitive performance. These effects are particularly concerning in environments where focus, comfort, and well-being are essential—such as classrooms and living areas.

Modern buildings, designed for energy efficiency, often suffer from limited natural ventilation. This leads to the accumulation of airborne pollutants such as carbon dioxide (CO_2), volatile organic compounds (VOCs), fine particulate matter (PM), and increased humidity levels. Over time, this buildup can create an unhealthy atmosphere that is not immediately noticeable but has long-term consequences for health and productivity.

While air quality monitoring systems exist for industrial or commercial buildings, many educational and domestic environments lack accessible and affordable solutions. There is a growing need for compact, cost-effective, and user-friendly systems that enable individuals to track and respond to indoor air conditions in real-time.

The AirSense project addresses this need by offering a flexible and automated system for continuous environmental monitoring. By making air quality data easily visible and actionable—through mobile apps and smart home platforms like Home Assistant—AirSense empowers users to improve their surroundings and make informed decisions about ventilation, occupancy, and air purification.

1.2. Motivation

The AirSense project emerges in response to a growing global awareness of the impact of air quality on human health, comfort, and productivity. The COVID-19 pandemic, increased urban pollution, and climate events such as wildfires have heightened public sensitivity to the invisible but critical role of clean indoor air. Scientific studies have consistently shown that prolonged exposure to poor air quality can lead to respiratory illnesses, cognitive fatigue, sleep disturbances, and other long-term health effects—especially in vulnerable populations such as children and the elderly.

Our motivation stems from both this global context and personal experience. As students living and studying in Aveiro, Portugal, we have observed firsthand the consequences of air pollution. During recent wildfire events in the region, local air quality dropped significantly due to increased smoke and particulate matter in the atmosphere. This led to the suspension of school activities and disrupted daily routines. These events made the issue tangible for us and revealed a critical gap: the lack of affordable, accessible, and real-time air quality monitoring solutions for classrooms and residential spaces.

Moreover, traditional monitoring systems tend to be expensive, complex, or limited to specific use cases. In contrast, our goal with AirSense is to develop a lightweight, modular, and cost-effective solution that not only gathers relevant environmental data (such as temperature, humidity, CO_2 levels, and particulate matter) but also makes this information actionable for end users.

Our project is also deeply motivated by the potential to integrate with smart home systems—like Home Assistant—and expand usability beyond academic settings. We believe that everyone should have the ability to monitor the air they breathe, whether at home, at school, or in shared community environments. The recent implementation of automated configuration through a mobile Android app further enhances this goal by reducing setup complexity and increasing accessibility for non-technical users.

In summary, AirSense is driven by a real-world need, strengthened by local relevance, and guided by a desire to empower individuals with the data they need to protect their health and well-being

1.3. Objectives

The **primary goal** of the AirSense project is to design and implement a flexible and modular environmental monitoring system capable of assessing and reporting indoor air quality in real time. While the initial focus was on university amphitheaters, the system has since evolved to support a broader range of applications, including homes, offices, and other shared indoor environments.

To achieve this, the following specific objectives were established:

- **Design and Develop a Sensor Hub:** Construct a compact and efficient hardware system based on an ESP32 microcontroller, capable of interfacing with multiple environmental sensors. These sensors are responsible for measuring key parameters such as temperature, humidity, carbon dioxide (CO_2) concentration, volatile organic compounds (VOCs), nitrogen oxides (NOx), and particulate matter (PM1.0, PM2.5, PM10).
- **Enable Wireless Data Transmission via BLE:** Implement a Bluetooth Low Energy (BLE) communication protocol to forward data from the ESP32 sensor hub to a Raspberry Pi. This ensures low-power, short-range data transfer without requiring internet access at the sensor node.
- **Deploy an Embedded Backend for Data Collection and Storage:** Configure a **Raspberry Pi** to receive BLE data and forward it to a local **InfluxDB** time-series database running inside a Docker container. This solution avoids the need for remote servers or cloud dependencies while ensuring flexibility and scalability.
- **Integrate with Home Automation Ecosystems:** Enable **Home Assistant integration** (also containerized within the Raspberry Pi) to allow real-time visualization and alerting based on air quality metrics. This makes the system accessible via user-friendly dashboards and compatible with existing smart home setups.
- **Automate Configuration Through an Android App:** Develop a mobile Android application that allows users to scan a QR code attached to the sensor hub. This simplifies configuration by automatically pairing the sensor hub with the correct Raspberry Pi and starting the data flow with minimal user intervention.
- **Ensure Data Reliability and Non-Redundancy:** Implement mechanisms to prevent redundant data entries and guarantee consistency, even during temporary interruptions in BLE communication or Raspberry Pi availability.
- **Provide Extensibility for Future Expansion:** Ensure that the architecture supports the integration of new sensors, additional Raspberry Pi nodes, or forwarding to cloud databases if needed in future deployments.
-

Ultimately, the project seeks to provide a **robust, low-cost, and user-friendly solution** for monitoring air quality, enhancing health and safety in indoor environments, and contributing to broader environmental awareness.

Air pollution is a significant global issue, with harmful effects on health, ecosystems, and daily life. The increasing frequency of events such as wildfires, industrial emissions, and urban congestion exacerbates the need for effective air quality monitoring. However, the availability of detailed and timely air quality data is still limited, especially in specific environments like classrooms and amphitheaters, where students and faculty spend significant amounts of time.

Current air monitoring systems often rely on centralized infrastructure, which may not always provide adequate coverage in real-time. This highlights the need for localized solutions that can measure and monitor the air quality specifically within educational spaces, helping to ensure an environment conducive to learning.

1.4. Scope

The AirSense project focuses on the design, implementation, and validation of a modular indoor air quality monitoring system tailored for environments such as schools, homes, and small offices. This system is intended to provide real-time data on key environmental metrics—temperature, humidity, CO_2 concentration, particulate matter (PM1.0, PM2.5, PM10), volatile organic compounds (VOCs), and nitrogen oxides (NOx)—collected through a custom ESP32-based sensor hub.

Covered in This Project:

- **Hardware Design**: Development of a compact sensor hub using the ESP32 microcontroller, integrating multiple environmental sensors.
- **Wireless Communication**: Implementation of Bluetooth Low Energy (BLE) for efficient, low-power data transmission to a local receiver (Raspberry Pi).
- **Backend Infrastructure**: Deployment of a lightweight backend on the Raspberry Pi using Docker, including InfluxDB for time-series data storage.
- **Smart Home Integration**: Real-time data visualization and alerts through integration with Home Assistant.
- **User Interface**: Development of an Android mobile application to simplify the setup process via QR code scanning and automated configuration.
- **Modular and Extensible Architecture**: System designed with the potential for future expansion (e.g., new sensors, cloud sync, or additional nodes).

Not Covered in This Project:

- **Cloud Infrastructure via VM**: Although the original plan included migrating the backend (InfluxDB and Home Assistant) to a virtual machine provided by the **Universidade de Aveiro (UA)** — thus using the Raspberry Pi solely as a gateway — this cloud deployment was not implemented in the current version due to **inaccessibility** to a Virtual Machine of IT. All services currently run locally on the Raspberry Pi but for the project to run on a public network like eduroam all files need to be installed in the UA's Virtual Machine and the gateway correctly configured to use the created **MQTT** file to transmit the data via wifi when Bluetooth Low Energy reach is not enough.
- **Scalability Across Distributed Locations**: With the backend hosted locally, multi-site support and centralized monitoring across different physical spaces (e.g., multiple classrooms or buildings) remains outside the current scope.
- **Security and Remote Access Management**: As cloud hosting was not integrated, features such as secure remote data access, centralized authentication, or VPN support are not part of this version.

This scope reflects a deliberate focus on developing a functional and autonomous prototype that can operate fully on local infrastructure with minimal user intervention. At the same time, it establishes a solid foundation for future migration to a centralized architecture hosted on institutional virtual machines, enabling enhanced scalability, maintainability, and remote accessibility.

2. Related Work

1. **Co-Designing an Air Quality Web App with School Pupils and Staff: The SAMHE Web App**

The SAMHE Web App is a UK-based initiative that applies a citizen science approach to improve classroom air quality awareness. Developed in collaboration with teachers and students, the app visualizes key metrics—such as CO_2 levels, temperature, and humidity—enabling users to make informed decisions about ventilation. The participatory design process emphasized usability and engagement, making it suitable for educational settings. The project underlines the importance of user-centered design in developing tools for air quality monitoring and behavioral change.

2. **Indoor Air Quality and Climate Change – U.S. Environmental Protection Agency (EPA)**

The EPA's work highlights the strong connection between indoor air quality and public health, especially in the context of global climate change. Their research identifies key pollutant sources—both internal (e.g., household activities) and external (e.g., wildfires, urban pollution)—and offers mitigation strategies such as improved ventilation and filtration. The report stresses the importance of localized monitoring to reduce exposure risks, particularly in sensitive environments like homes and schools.

3. **AirGradient ONE – Open-Source Indoor Air Quality Monitor**

AirGradient ONE is an open-source, modular monitoring system capable of measuring a wide range of environmental parameters including PM1.0, PM2.5, PM10, CO_2, TVOCs, NOx, temperature, and humidity. It is notable for its high-quality sensors and strong integration with platforms like Home Assistant. Its design philosophy—modularity, transparency, and user empowerment—closely aligns with AirSense's goals. Unlike many proprietary solutions, AirGradient ONE allows for customization, making it a strong point of comparison for DIY and academic projects.

3. State-of-the-art

In recent years, indoor air quality monitoring has gained significant attention due to its direct impact on human health, particularly in closed environments such as classrooms, offices, and homes. Advances in sensing technology, edge computing, and IoT integration have enabled the development of compact, accurate, and affordable systems capable of real-time monitoring.

1. **Sensor Technologies**:

Modern air quality monitoring solutions rely on compact, low-power sensors that can detect a variety of pollutants and environmental conditions. The **AirSense** project integrates three key sensors selected for their precision, reliability, and complementary measurements that together supply a complete state of environment :

- SEN55: A state-of-the-art sensor module capable of measuring particulate matter (PM1.0, PM2.5, PM10), temperature, and humidity with high precision. Its compact design and internal algorithms for air quality compensation make it ideal for high-quality indoor monitoring.
- SGP30: A digital gas sensor from Sensirion that provides real-time measurement of Total Volatile Organic Compounds (TVOCs) and CO_2-equivalent levels. It's optimized for low-power applications and is widely used in air purifiers and smart home devices.
- S88: A smoke and fire detection sensor that identifies burnt particles and fumes, offering an additional layer of safety in air quality systems. While less common in typical environmental monitors, its inclusion enhances the system's ability to detect hazardous combustion byproducts.

Together, these sensors provide a well-rounded, multi-dimensional view of indoor air quality, enabling the detection of both chronic environmental issues and acute health threats.

2. **Wireless Communication - Bluetooth**:

Reliable and efficient data transmission is a key requirement for indoor air quality monitoring systems, particularly when aiming for portability and low energy consumption. In the **AirSense** system, wireless communication is handled primarily through **Bluetooth Low Energy (BLE)**, which facilitates direct data transfer from the sensor hub (ESP32) to the central processing unit (Raspberry Pi).

BLE is chosen for its low power consumption, lightweight communication model, and adequate throughput for periodic transmission of sensor data. The ESP32 continuously reads environmental metrics such as particulate matter, temperature, humidity, and volatile organic compounds, and transmits this data via BLE in predefined intervals.

On the Raspberry Pi, a **custom Python script** establishes a BLE connection with the ESP32, handles data parsing, and immediately forwards the sensor readings to a local instance of **InfluxDB**, a time-series database optimized for high-throughput data ingestion and querying.

This streamlined communication model ensures a responsive, energy-efficient, and easily maintainable system architecture, suitable for real-time monitoring in educational and domestic environments.

3. **Remote Database Integration**:

In the AirSense system, storing and managing environmental data efficiently is critical to ensuring long-term monitoring, analysis, and visualization. For this purpose, the project integrates with **InfluxDB**, a high-performance, open-source time-series database optimized for storing sensor data.

Sensor readings collected by the ESP32 are transmitted via **Bluetooth Low Energy (BLE)** to a **Raspberry Pi**, which acts as the central processing hub. The Raspberry Pi is responsible for receiving, organizing, and forwarding the data to InfluxDB. This approach enables the system to operate without relying on cloud services, ensuring data privacy and reducing dependency on external infrastructure.

Once received by the Raspberry Pi, sensor data is continuously written to InfluxDB using a dedicated script. Each data point is tagged with metadata such as sensor type and location, and timestamped for precise temporal tracking. This structure supports efficient querying and enables advanced data analytics.

The benefits of using InfluxDB include:

- **Scalability**: Capable of handling high-frequency data streams without performance degradation.
- **Reliability**: Ensures consistent and lossless data storage, even under intermittent connectivity conditions.
- **Real-Time Access**: Enables immediate availability of data for visualization and alerting through platforms such as **Home Assistant**.
- **Local Deployment**: The entire pipeline runs locally on the Raspberry Pi, offering greater control, offline functionality, and enhanced data security.

This local, lightweight, and modular database integration makes AirSense highly suitable for scalable indoor air quality monitoring across classrooms, homes, and small institutions.

4. **Portable and Affordable Solutions**:

A core objective of the AirSense project is to deliver a low-cost and easily deployable air quality monitoring system without compromising on performance or flexibility. To achieve this, the system relies on two key components: the **ESP32 microcontroller** and the **Raspberry Pi**.

The **ESP32** is a compact and energy-efficient microcontroller equipped with integrated **Wi-Fi** and **Bluetooth Low Energy (BLE)**. In AirSense, it acts as the sensor hub, collecting environmental data from the connected air quality sensors and transmitting it wirelessly to the Raspberry Pi. Its low power consumption, small footprint, and dual-mode wireless communication capabilities make it ideal for embedded and portable IoT applications.

The **Raspberry Pi** serves as the processing and coordination unit. It receives data via BLE from the ESP32, manages local storage in **InfluxDB**, and enables visualization and automation through **Home Assistant**. Its small size, low cost, and extensive software support make it a practical choice for on-site data management and visualization.

By combining these two components, AirSense achieves a system that is:

- **Portable**: Small form factor and wireless communication enable easy deployment in classrooms, offices, or homes.
- **Affordable**: Utilizes readily available, low-cost hardware, significantly lowering the barrier to adoption in budget-constrained environments.
- **Scalable**: The modular architecture allows for the addition of more sensors or nodes without major reconfiguration.
- **Accessible**: No need for cloud infrastructure or expensive proprietary solutions, making it ideal for educational and experimental settings.

This combination ensures that AirSense remains accessible and practical for a wide range of use cases, from academic research to real-world indoor air quality monitoring.

5. Why Current Solutions Fall Short and our Improvements:

Despite significant progress in the field of indoor air quality monitoring, many existing solutions still present critical limitations in terms of **cost, accessibility, modularity**, and **data transparency**. These shortcomings often hinder widespread adoption, especially in educational, research, or low-resource environments.

Common Limitations in Existing Solutions:

- **High Cost and Proprietary Systems**: Many commercial IAQ devices rely on closed ecosystems with proprietary hardware and software, leading to high upfront costs and limited flexibility for customization or integration with other systems.
- **Cloud Dependency**: Most consumer-grade air quality monitors require constant internet access and cloud-based storage, which raises privacy concerns and makes them unsuitable for local-only deployments or offline environments.
- **Limited Modularity**: Commercial products often come as fixed bundles with little room to swap sensors, adjust sampling rates, or adapt the system to specific use cases (e.g., classrooms vs. industrial labs).
- **Opaque Data Handling**: Users typically have limited access to raw sensor data, making it difficult to perform custom analysis, validate readings, or integrate the data into broader automation systems.

AirSense was designed from the ground up to address these issues by prioritizing **modularity**, **transparency**, and **affordability** without sacrificing performance. Some key improvements include:

- **Open and Modular Architecture:** The system allows users to easily add, remove, or swap sensors (such as SEN55, SGP30, and S88), enabling customized configurations tailored to different indoor environments and monitoring needs.
- **Local Data Processing:** Data is collected and processed locally on a Raspberry Pi, giving users full control over data storage and privacy. This eliminates dependence on external cloud platforms, reducing risks related to data security and accessibility.
- **Full Data Transparency:** All raw sensor data is stored in an InfluxDB time-series database and visualized through Home Assistant. This provides users with complete access to detailed metrics, enabling thorough analysis and informed decision-making.
- **Low-Cost and Scalable Design:** The use of an ESP32 microcontroller for sensor data acquisition ensures the system remains affordable. Paired with the Raspberry Pi as a central hub, this setup can be easily replicated or scaled up for larger deployments or educational purposes.
- **Efficient BLE Communication:** Bluetooth Low Energy (BLE) is employed to transmit sensor data from the ESP32 to the Raspberry Pi, minimizing power consumption while maintaining reliable communication. This makes AirSense suitable for semi-mobile setups or battery-powered applications.

These enhancements make AirSense a functional, accurate, and transparent indoor air quality monitoring solution, filling the gaps left by many current systems by providing a customizable, scalable platform tailored for diverse environments.

4. Requirements

4.1. Functional Requirements

- **Environmental Measurements:** The system must measure a comprehensive range of indoor air quality parameters, including temperature, humidity, particulate matter (PM1.0, PM2.5, PM4.0, PM10), VOCs, NOx, and CO_2-equivalent levels. These measurements are performed using modular sensors such as the SEN55, SGP30, and S88.
- **Data Acquisition and Local Processing:** Sensor data is transmitted from the ESP32 to the Raspberry Pi via Bluetooth Low Energy (BLE). The Raspberry Pi is responsible for parsing, validating, and formatting the received data.
- **Data Storage:** All collected sensor data must be stored locally on the Raspberry Pi using InfluxDB, a time-series database optimized for environmental data. This allows for efficient storage, querying, and future analysis.
- **Data Visualization and Monitoring:** The system must provide users with access to real-time and historical data through a graphical interface built on Home Assistant. This enables continuous monitoring and supports informed decision-making about indoor air quality.
- **Scalability:** The system must support the integration of additional sensors or sensor types without major architectural changes, ensuring adaptability to new monitoring needs or different indoor environments.

4.2. Non-Functional Requirements

- **Usability:** The system must be easy to use and understand, even by non-technical users such as students and faculty. The interface should clearly present air quality metrics, trends, and alerts via Home Assistant, requiring no advanced configuration or technical knowledge.
- **Modularity and Portability:** The solution must be lightweight, compact, and easy to deploy across different rooms or buildings. The modular architecture, based on the ESP32 and Raspberry Pi, allows for quick setup, disassembly, and reconfiguration with minimal technical effort.
- **Ease of Assembly and Deployment:** The system must be straightforward to assemble using widely available components, with minimal soldering or specialized tools. This ensures ease of maintenance and encourages replication in educational or research settings.
- **Scalability:** The architecture must support expansion, allowing multiple sensor nodes to be deployed in larger spaces or across multiple rooms. New sensors should be integrable with minor configuration changes, supporting long-term system growth.
- **Energy Efficiency:** BLE-based communication and low-power microcontrollers ensure that the system can operate with minimal energy consumption, making it suitable for semi-mobile or battery-powered deployments when needed.

5. Methodology

5.1. Project Phases

The development of the AirSense system followed a structured, phased methodology inspired by the Rational Unified Process (RUP), which divides the project lifecycle into four key stages: **Inception**, **Elaboration**, **Construction**, and **Transition**. This approach enabled clear planning, incremental progress, and validation of each component before final integration.

5.1.1. Inception

The inception phase focused on identifying the problem space and defining the project's main goals. Research was conducted on the impact of indoor air quality (IAQ) in educational settings and the limitations of existing commercial solutions. Key system requirements were defined, such as modularity, affordability, and transparency. Initial component selection began during this stage, including consideration of various sensors (e.g., SEN55, SGP30, S88) and computing platforms (ESP32, Raspberry Pi). The high-level architecture and communication model were also outlined.

5.1.2. Elaboration

During elaboration, the project's architecture was refined, and feasibility was validated through prototypes. The sensors were individually tested with the ESP32 to verify signal quality, response time, and BLE communication capabilities. The communication between the ESP32 and Raspberry Pi via Bluetooth Low Energy (BLE) was prototyped using Python and the bleak library. A detailed plan for data acquisition, local processing, and storage was formulated, including the use of InfluxDB for time-series data storage and Home Assistant for visualization. This phase concluded with a functioning prototype for sensor data acquisition and local communication.

5.1.3. Construction

The construction phase focused on full system development and integration. Custom firmware was written for the ESP32 to periodically gather sensor readings and transmit them via BLE to the Raspberry Pi. On the Raspberry Pi, Python scripts were developed to receive BLE notifications, parse the data, and write it into InfluxDB using the official client API. Docker containers were configured to host both InfluxDB and Home Assistant, ensuring portability and ease of deployment. The modular architecture allowed for easy expansion and sensor substitution during this phase. The entire data pipeline — from sensor to visualization — was successfully implemented and validated.

5.1.4. Transition

In the final phase, the project was refined for real-world use and presentation. The system was deployed in an environment, where data was collected and analyzed. Usability and portability were evaluated, including the system's ability to be relocated across different rooms. Documentation was prepared to guide installation, usage, and future development. Feedback from early tests was used to make final adjustments, such as improving BLE reconnection handling and optimizing JSON parsing routines. The project was then finalized for demonstration and academic evaluation.

5.2. Hardware Development

This section explains the selection, configuration, and assembly of the hardware components used in AirSense. The goal was to create a modular, low-cost, and portable system capable of reliable indoor air quality monitoring.

Microcontroller – ESP32

- Chosen for its low cost, low power consumption, and built-in BLE and Wi-Fi.
- Acted as the data acquisition node, collecting data from connected environmental sensors.
- Handled sensor interfacing and periodic transmission of data to the Raspberry Pi via BLE.
- Configured to ensure minimal energy consumption for potential battery-powered operation.

Sensors

- **SEN55**: Measured particulate matter (PM1.0, PM2.5, PM4.0, PM10), temperature, and humidity.
- **SGP30**: Provided total VOC and equivalent CO_2 estimations.
- **S88 (CO_2 sensor)**: Offered more reliable CO_2 concentration readings for environments like classrooms.
- Sensors were connected to the ESP32 via I^2C and carefully powered to maintain accuracy and stability.

Host – Raspberry Pi

- Served as the central hub for receiving BLE data from the ESP32.
- Hosted the InfluxDB database and Home Assistant dashboard via Docker containers.
- Enabled local data storage and visualization without relying on third-party cloud platforms.
- Allowed users to interact with the system through a web interface.

Enclosure and Portability

- All components were assembled into a compact and lightweight enclosure.
- Designed to allow airflow for accurate sensing while protecting electronics.
- The modular wiring and layout allow quick sensor replacement or system relocation.
- System was tested for deployment in multiple rooms (e.g., university amphitheaters).

5.3. Software Architecture

The AirSense system is composed of a layered software architecture that emphasizes modularity, local processing, and seamless data visualization. It integrates firmware-level control on the ESP32 microcontroller with backend processing and user-facing tools running on the Raspberry Pi.

At the core of the sensing unit lies the ESP32, responsible for interfacing with environmental sensors such as the SEN55, SGP30, and S88. The firmware, developed in C++ using the Arduino framework, gathers sensor data at regular intervals (which can be changed if needed). Once collected, this data is serialized into a string and transmitted over Bluetooth Low Energy (BLE) to the Raspberry Pi. This approach allows the ESP32 to operate efficiently with minimal energy consumption, making it suitable for semi-portable deployments.

On the Raspberry Pi, a Python-based BLE client (using the **bleak** library) receives data packets from the ESP32, parses them into structured key-value pairs, and prepares them for storage. This data is then formatted using the **influxdb-client** library and written into InfluxDB—a time-series database running in a Docker container. Each data point is tagged with sensor metadata and stored with precision, enabling reliable long-term storage and future analysis.

Visualization is handled through Home Assistant, also running in a Docker container on the Raspberry Pi. Home Assistant is configured to connect to InfluxDB and generate dashboards that provide real-time and historical views of all environmental parameters. This platform not only enables easy monitoring but also allows for the implementation of alerts and automation rules based on air quality thresholds.

Overall, the architecture is designed for modularity and maintainability. Each component—from sensor firmware to data storage and visualization—operates independently yet cooperatively. Docker containers ensure reproducibility and simplify system deployment and updates. This layered and containerized approach provides a scalable foundation for future enhancements, whether that involves adding new sensors, integrating cloud services, or expanding to multiple sensing nodes.

5.4. Challenges and Adjustments

During the development of AirSense, several practical challenges arose that required adjustments to ensure system reliability and usability. One of the primary issues encountered was the intermittent reliability of Bluetooth Low Energy (BLE) connection and communication between the ESP32 and Raspberry Pi. This necessitated implementing robust reconnection and error-handling mechanisms within the data acquisition scripts to maintain continuous data flow.

Parsing sensor data also presented challenges due to the variety and format of measurements transmitted. Careful design of the data parsing logic was essential to correctly interpret and organize the sensor outputs for storage and visualization.

Configuration management on the Raspberry Pi, including environment variables and dependency handling, required attention to create a stable and reproducible setup.

Finally, power supply considerations, particularly when using portable power banks, highlighted the need for efficient energy consumption and effective power management to maximize system uptime during deployment in different environments. Since one of the project's goals was to enable mobility and ease of use in classrooms or other locations, relying on main power was not always practical. To address this, we deliberately designed the system to operate in intermittent sessions rather than maintaining a continuous monitoring cycle. By activating the sensor hub and data collection scripts only during defined time windows, we reduced both energy consumption and thermal strain on the components. This decision significantly improved battery efficiency, allowing the system to function for extended periods on a single charge. Additionally, it opened the possibility for scheduling or automation based on user needs or environmental triggers, making the setup more adaptable and sustainable in the long term.

These challenges shaped iterative improvements and contributed to a more resilient and user-friendly system.

6. Architecture

6.1. Logical Architecture

The AirSense system architecture is designed with modularity, decentralization, and energy efficiency as guiding principles, aiming to provide a reliable and flexible indoor air quality monitoring solution.

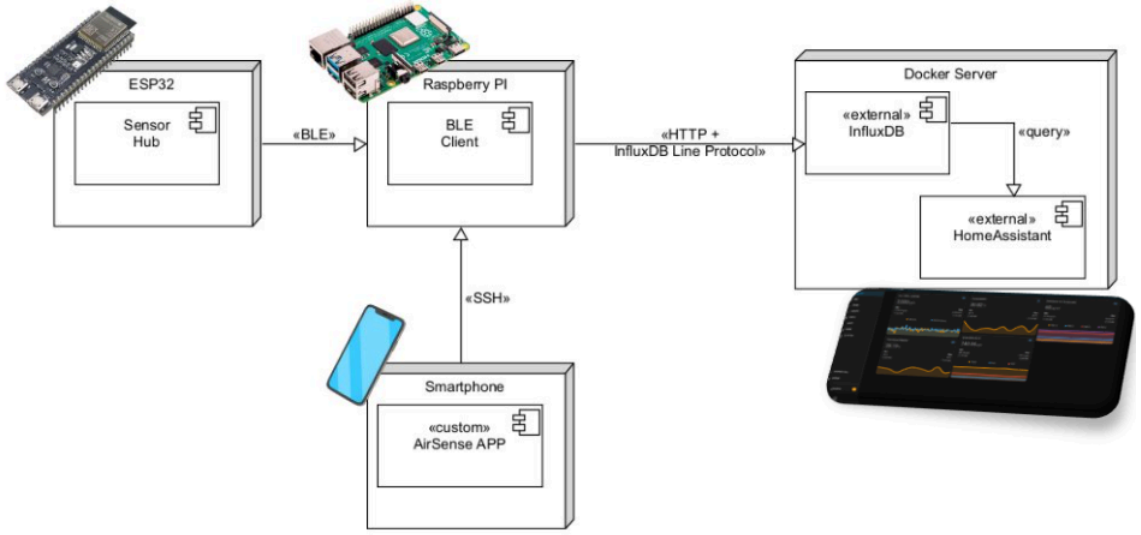


Figure 1 – AirSense Architecture Overview

The system consists of five key components, each fulfilling specific roles:

ESP32 Sensor Node:

This is the edge device responsible for data acquisition. It hosts an array of environmental sensors such as the SEN55 (measuring particulate matter and climatic parameters), SGP30 (for VOCs and CO_2-equivalent), and S88 (for NOx gases). The ESP32 aggregates sensor readings periodically, applies preliminary processing, and broadcasts the data via Bluetooth Low Energy (BLE). The choice of BLE is critical for minimizing power consumption, enabling the device to run on portable power banks with optimized uptime through deep sleep modes.

In the final system, three primary sensors, SEN55, SGP30, and S88 are integrated into the ESP32 sensor node, along with a Real-Time Clock (RTC) module (DS3231) to ensure accurate timestamping of measurements. These components collectively enable the monitoring of critical air quality indicators, including particulate matter, volatile organic compounds (VOCs), nitrogen oxides (NOx), CO_2 equivalents, as well as ambient temperature and humidity. The inclusion of the RTC ensures temporal precision in data logging, which is essential for time-series analysis and database consistency. The next table summarizes the selected components and their technical specifications.

The exact model used is ESP32-S3 WROOM 1U with the cost of 7€ approximately

Model	**SEN55**	**SGP30**	**S88**	**RTC Module**
Manufacturer	Sensirion	Sensirion	SPEC Sensors / Winsen	Maxim Integrated
Monitored Parameters	PM1.0, PM2.5, PM10, Temperature, Humidity	VOCs, CO_2 equivalent	Nitrogen oxides (NOx)	Date and Time
Communication Protocol	I²C / UART (via interface)	I²C	Analog / UART	I²C
Measurement Range	PM: 0–1000 µg/m³ Temperature: -10 to +60 °C Humidity: 0–100%	VOCs: 0–60,000 ppb CO_2eq: 400–60,000 ppm	NOx: 0–10 ppm (varies by model)	
Accuracy	PM: ±10 µg/m³ or ±10% Temperature: ±0.8 °C Humidity: ±5%	VOC: ±15% CO_2eq: ±15%	~±5% of measured value	
Sensing Principle	Optical laser scattering (PM) Capacitive (RH/Temp)	MOx (Metal-Oxide Semiconductor)	Electrochemical	
Estimated Cost	~19 €	~6.5 €	~15 €	~2 €

Tabela - Especificações técnicas dos sensores utilizados

Raspberry Pi Data Collector:

Acting as the central data hub, the Raspberry Pi serves multiple functions. It continuously acts as a BLE client, scanning and connecting to the ESP32 to receive sensor data streams. A Python-based BLE client application parses incoming BLE notifications into structured JSON entries, which are temporarily stored locally. This buffering allows the system to tolerate network or service interruptions without data loss. Subsequently, another Python script reads the JSON buffer and formats the data for insertion into a time-series database.

The model used is Raspberry Pi 4 Computer Model B 2GB RAM with the cost of 35€ approximately

InfluxDB Time-Series Database:

Deployed within a Docker container on the Raspberry Pi, InfluxDB is the chosen backend for persistent storage of sensor measurements. It provides efficient handling of high-frequency time-series data, tagging each data point with contextual metadata such as sensor and location IDs. Data is ingested via HTTP using the InfluxDB Line Protocol, maintaining data integrity and enabling scalable queries.

Home Assistant Visualization Platform:

Also containerized on the Raspberry Pi, Home Assistant provides a responsive and customizable dashboard interface. It queries InfluxDB to display real-time and historical air quality metrics through graphs, gauges, and alerts. This interface is designed to be accessible locally or remotely, making it easy for end-users like students or facility managers to monitor environmental conditions and react accordingly.

AirSense Mobile Application:

A custom smartphone app connects to the Raspberry Pi over SSH to facilitate remote control, configuration, and diagnostics. While currently focused on BLE service management and telemetry, this app can evolve into a comprehensive remote monitoring and alerting tool.

6.2. Deployment Architecture

The deployment architecture of AirSense emphasizes portability, modularity, and practicality, designed to seamlessly fit within typical indoor environments such as classrooms or office spaces.

At the heart of the deployment is a custom enclosure constructed entirely from LEGO bricks, measuring approximately 12 cm by 23 cm by 5cm. This unique enclosure not only protects the internal electronics but also reinforces the project's branding and modular design philosophy. The LEGO construction allows for easy access, modifications, and reconfiguration of components, making it ideal for educational settings where flexibility is key.

Inside this LEGO housing are the key components: the environmental sensors (SEN55 for particulate matter and basic environmental parameters, SGP30 for TVOC and CO_2 equivalent readings, and S88 for NOx detection), the ESP32 microcontroller responsible for aggregating sensor data and broadcasting it via Bluetooth Low Energy (BLE), and the Raspberry Pi, which collects and processes the data.

Power is supplied by a portable power bank, also housed within the LEGO enclosure. The power bank provides energy to both the ESP32 and the Raspberry Pi through USB connections, enabling the entire system to operate independently from fixed power sources. This compact all-in-one design enhances system portability and allows for flexible placement and easy relocation between rooms without cumbersome wiring.

Wireless communication via BLE between the ESP32 sensor node and the Raspberry Pi within the same enclosure reduces cable clutter and supports streamlined data flow. The Raspberry Pi performs data parsing, local buffering, and forwards information to the Docker-hosted InfluxDB and Home Assistant services for long-term storage and visualization.

This deployment setup effectively balances compactness, robustness, and ease of use. The LEGO enclosure supports user interaction and quick modifications, the integrated power solution enables true portability, and housing all electronics together simplifies maintenance while preserving a tidy, cohesive form factor.

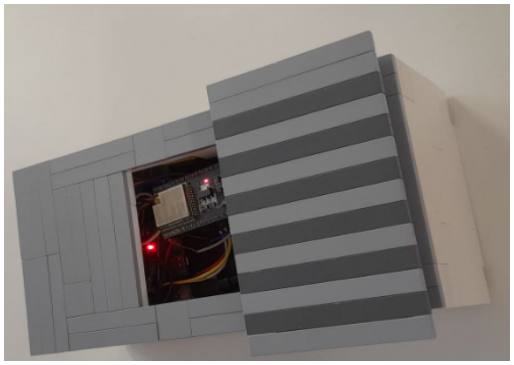


Figure 2 – AirSense LEGO Enclosure with Integrated Sensors, ESP32 and RaspberryPi

6.3. Data Flow Diagrams

The data flow within the AirSense system is structured to ensure efficient, low-latency, and fault-tolerant communication from the sensors to the end-user interface. The ESP32 microcontroller collects environmental data from its connected sensors—such as the SEN55, SGP30, and NOx modules—at fixed time intervals, typically every five seconds. After applying local processing such as averaging and formatting, the ESP32 broadcasts the data asynchronously over Bluetooth Low Energy (BLE). This method ensures fast, low-power transmissions without requiring acknowledgment, which is ideal for battery-optimized sensing environments.

The Raspberry Pi acts as the BLE client, continuously scanning for nearby sensor broadcasts. A Python-based BLE listener receives and parses the structured data into JSON format. Rather than sending this data immediately, it is first cached in a local buffer—a JSON file on disk—which decouples acquisition from storage and protects against transient failures in downstream services.

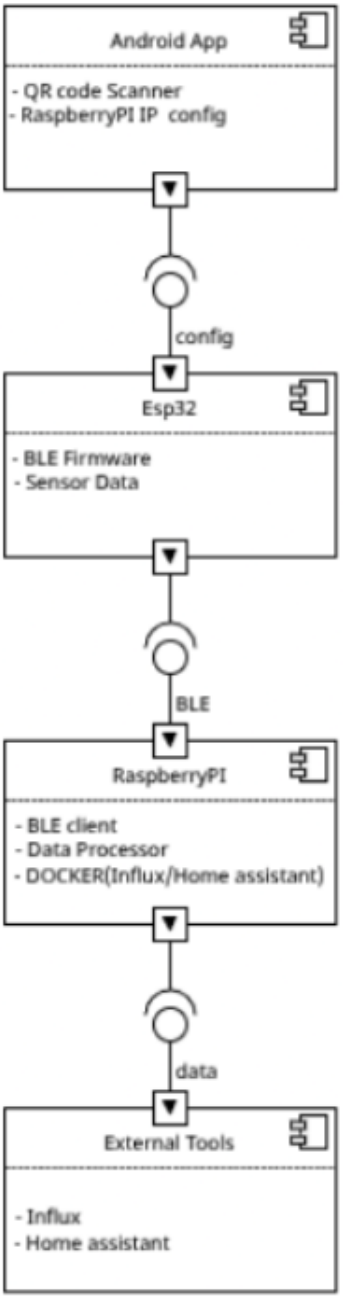
A second script on the Raspberry Pi periodically reads this buffer, formats the entries according to the InfluxDB Line Protocol, and transmits them via HTTP to a Docker-hosted InfluxDB instance. This time-series database ensures that each record is timestamped and tagged appropriately (e.g., sensor ID, room ID), enabling fine-grained historical queries.

On the visualization end, Home Assistant interfaces with InfluxDB to retrieve both live and historical sensor data. Its responsive dashboard displays dynamic charts, status gauges, and alert systems that reflect environmental changes in near real-time. This web-based UI provides both desktop and mobile access.

Complementing this setup is the custom Android application, which connects to the Raspberry Pi via secure SSH. This mobile interface allows users to remotely control BLE services, manage deployment settings, and—potentially in future updates—access diagnostics or live telemetry. Together, these layers form a robust and extensible pipeline that transforms raw environmental signals into actionable insights with minimal latency and high reliability.

6.4. UML Diagrams

The UML component diagram (Figure – AirSense UML Component Diagram), based on the corresponding .uxf model, provides a high-level visualization of the modular structure of the AirSense system and the interactions between its core components. Each component is represented as a distinct logical module, encapsulating specific responsibilities within the architecture.



Android App:

This mobile application acts as the primary deployment interface. Its main function is to configure the Raspberry Pi's network settings during installation. It achieves this by scanning a QR code and initiating a secure SSH session with the Raspberry Pi. Although it does not directly interact with the ESP32 sensor node, it plays a crucial role in initializing deployment parameters and controlling services remotely.

ESP32 Microcontroller:

Functioning as the edge node, the ESP32 hosts all the environmental sensors (SEN55, SGP30, NOx). It periodically polls these sensors, structures the data, and transmits it over Bluetooth Low Energy (BLE). Its responsibilities include sensor aggregation, BLE packet formatting, and energy-efficient data broadcasting.

Raspberry Pi:

The Raspberry Pi serves as both the BLE client and the local processing unit. It continuously scans for BLE advertisements from the ESP32 and parses the received data. This data is temporarily buffered locally before being forwarded to external services. The Raspberry Pi also hosts a Docker environment that runs the InfluxDB and Home Assistant containers, effectively integrating data storage and visualization into a compact and portable system.

External Tools (Dockerized Services):

- **InfluxDB** serves as the time-series database that persists all environmental data using timestamped entries.
- **Home Assistant** acts as the front-end dashboard for users, querying InfluxDB to generate real-time visualizations, historical trends, and environment-based alerts.

This UML component diagram provides a concise yet informative overview of AirSense's system architecture, emphasizing modularity, logical separation of concerns, and extensibility. Its simplicity allows both technical and non-technical stakeholders to grasp the system's structure and data interactions at a glance.

6.5. Communication Protocols

The AirSense system relies on a carefully selected set of communication protocols, each chosen to align with the project's core goals of energy efficiency, modular integration, local autonomy, and minimal complexity. These protocols enable seamless interactions between hardware and software components while ensuring robust and low-latency operation in a constrained environment.

BLE (Bluetooth Low Energy):

Used for transmitting sensor data from the ESP32 microcontroller to the Raspberry Pi, BLE provides an energy-efficient solution with sufficient bandwidth for periodic environmental readings. Its short-range capabilities are ideal for AirSense's compact, self-contained deployment, minimizing interference and power usage while ensuring reliable data delivery.

HTTP (InfluxDB Line Protocol):

The Raspberry Pi communicates with the InfluxDB service using HTTP requests formatted with the Line Protocol—InfluxDB's native format for data ingestion. This choice simplifies the software pipeline and ensures compatibility with the database, allowing for efficient, timestamped storage of structured sensor data.

SSH (Secure Shell):

The Android application establishes an SSH connection with the Raspberry Pi to manage configuration, service control, and diagnostics. SSH offers a secure, lightweight, and scriptable alternative to traditional web-based interfaces, enabling advanced control during setup and maintenance without increasing system overhead.

Wi-Fi / Local LAN:

Home Assistant, hosted on the Raspberry Pi, is accessed through the local network via a web browser or mobile device. This allows real-time and historical visualization of environmental data without requiring internet connectivity. The use of LAN ensures responsive interaction and offline operability within indoor spaces like homes, schools, or offices.

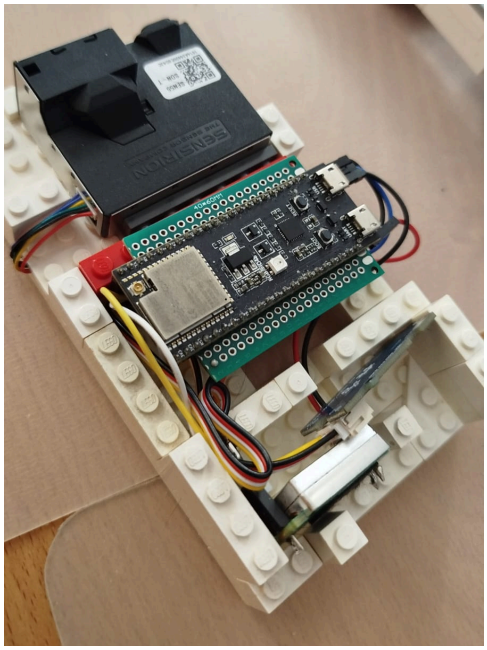
Advantages of the Selected Protocols:

The selected communication protocols offer several advantages that align with the goals of the AirSense system. BLE is extremely power-efficient and ideally suited for short-range, low-bandwidth communication between embedded devices, making it perfect for transmitting sensor data from the ESP32 to the Raspberry Pi. HTTP simplifies data ingestion into InfluxDB by leveraging a well-documented and widely supported interface through the Line Protocol, ensuring seamless integration with the time-series database. SSH provides a secure and flexible channel for configuration, diagnostics, and remote access during deployment, allowing the Android app to manage the Raspberry Pi without introducing unnecessary complexity. Lastly, LAN/Wi-Fi enables high-speed, local monitoring of sensor data via Home Assistant, ensuring real-time accessibility without relying on internet connectivity—supporting responsive user interaction within closed, offline environments.

Together, this protocol stack ensures that AirSense remains lightweight, self-sufficient, and easily maintainable. The design prioritizes local operability and secure, modular communication pathways—making it well-suited for scalable deployment in a variety of indoor environments.

7. Implementation

7.1. Approach



The implementation of AirSense followed a modular and iterative approach, combining embedded firmware development with containerized services on a lightweight backend. Each layer of the system was developed and tested independently before being integrated into a cohesive pipeline.

ESP32 Firmware:

The firmware was developed using the Arduino framework in C++, chosen for its extensive community support and rapid prototyping capabilities. The microcontroller reads data from connected sensors via I²C and serial interfaces. Sensor readings are collected periodically, serialized into a compact string format, and transmitted over BLE using custom services and characteristics. Power efficiency was a primary consideration, with features like dynamic sampling intervals and potential for deep sleep implemented to support battery-powered use cases.

BLE Communication and Parsing:

On the Raspberry Pi, a Python application built with the **bleak** library serves as the BLE client. It establishes and maintains a connection with the ESP32, listens for data notifications, and writes the raw output into structured JSON files. A second Python script, running as a scheduled task, parses this intermediate data, attaches metadata (e.g., timestamps, sensor IDs), and formats it according to the InfluxDB Line Protocol.

InfluxDB Integration:

Sensor data is inserted into InfluxDB using the official Python client. The time-series database runs inside a Docker container to simplify deployment and ensure consistency across different environments. Docker volumes are used for persistent data storage, enabling long-term retention even during system restarts or updates.

Visualization via Home Assistant:

Home Assistant is deployed alongside InfluxDB, also within a Docker container. It is configured to query the database and generate real-time dashboards. Custom cards, gauges, and charts are created to visualize metrics such as CO_2 concentration, particulate matter levels, and humidity. Automation scripts and alert thresholds are also configured through the Home Assistant UI.

Android App and Remote Access:

The mobile application connects to the Raspberry Pi over SSH, allowing users to remotely start or stop BLE services and perform diagnostics. While still in early development, the app lays the foundation for future interaction patterns, such as push notifications or system health monitoring.

Deployment Considerations:

All system components — including sensors, the ESP32, the Raspberry Pi, and a portable power bank — are housed in a compact enclosure. This setup supports true plug-and-play deployment in classrooms or shared indoor environments, requiring only local Wi-Fi access to begin data monitoring.

Overall, the implementation process prioritized modularity, maintainability, and local autonomy. Each component was designed to operate independently but integrates seamlessly to form a reliable and flexible indoor air quality monitoring solution.

7.2. Implementation Challenges Encountered

During the implementation of AirSense, several technical and practical challenges were encountered across both hardware integration and software orchestration layers.

Sensor Interference and Signal Noise: In early prototypes, simultaneous readings from multiple sensors led to inconsistent values and occasional communication loss. This was mitigated by implementing staggered polling intervals on the ESP32 and applying rolling averages to smooth out transient fluctuations.

BLE Stability and Packet Loss: Bluetooth Low Energy communication between the ESP32 and Raspberry Pi exhibited occasional packet drops during extended operation. Switching from a connection-based model to an event-driven BLE scanning strategy improved reliability, allowing the system to recover gracefully from transmission gaps.

Docker Networking: Initial Docker setups created isolated networks that prevented Home Assistant from accessing InfluxDB. This was solved by using a shared Docker network bridge and fine-tuning container environment variables to ensure compatibility across services.

Power Management: Keeping all components (sensors, ESP32, Raspberry Pi) within a battery-powered setup was challenging. The Raspberry Pi's power draw proved substantial. To extend uptime, the firmware was optimized for efficiency, and the box was designed with passive ventilation to avoid overheating without active cooling.

Mobile App SSH Handling: Establishing a stable SSH session from Android to the Raspberry Pi was non-trivial. The app was adapted to use embedded SSH libraries with pre-configured keys and minimal permission prompts.

Physical Space Constraints: Fitting all components, including cabling, into a 23x12x5 cm box required careful spatial planning and cable management to avoid signal crosstalk or power interference.

These challenges played a pivotal role in shaping the final architecture of AirSense. Each obstacle — whether it involved unreliable BLE connections, container stability, or limited power availability — prompted deeper reflection on system resilience and usability. As a result, several components were redesigned or reconfigured to prioritize fault tolerance, modularity, and recovery from failure states. This iterative problem-solving process did more than resolve technical issues; it helped reinforce a development mindset centered around adaptability and long-term maintainability. Ultimately, overcoming these real-world challenges contributed to a more robust, portable, and user-centric solution, capable of operating dependably in diverse indoor environments with minimal intervention.

7.3. System Integration

The core of the AirSense system integration revolves around the seamless data flow from embedded sensors through wireless communication to backend processing and storage. This section provides an in-depth overview of the key software components responsible for this integration: the ESP32 firmware and the Raspberry Pi BLE gateway Python script.

7.3.1. ESP32 Firmware

The embedded firmware, written in C++ using the Arduino framework, acts as the sensor hub and BLE peripheral. It interfaces with multiple air quality sensors, including the SEN55 and the SGP30. Periodically, the firmware reads sensor values such as particulate matter concentrations (PM1.0, PM2.5, PM4.0, PM10), humidity, temperature, VOC and NOx indexes, and CO_2 equivalents.

To maintain reliability, the firmware monitors BLE connection status with robust callbacks that handle client connections and disconnections. Upon connection loss or after multiple failed connection attempts, the ESP32 performs a soft restart to ensure continuous availability.

Sensor readings are serialized into a compact, comma-separated string format containing key-value pairs, including a monotonically increasing count for data traceability. This string is written to a BLE characteristic which supports notifications, allowing the connected Raspberry Pi client to receive updates asynchronously every 5 seconds. The firmware optimizes power usage by carefully timing sensor reads and BLE notifications, with potential support for deep sleep modes in future iterations.

7.3.2. Raspberry Pi BLE Gateway

On the receiving end, the Raspberry Pi runs a Python 3 asynchronous script that utilizes the bleak library to serve as a BLE central device. The script continuously scans for the ESP32 peripheral by its advertised device name and attempts to establish a connection.

Once connected, it subscribes to notifications on the custom BLE characteristic to receive sensor data packets. Each incoming packet is decoded from a UTF-8 string and parsed into a dictionary of sensor readings. Special handling is applied for the S88CO2 sensor value, which is scaled for visualization purposes.

The script attaches a timestamp in ISO 8601 format to each data set and atomically writes the JSON-encoded sensor data to a local file (sensor_data.json). This intermediate file acts as a data buffer for downstream processing, facilitating decoupled workflows and reducing data loss risks during transient system states.

To increase robustness, the script implements error handling and automatic Bluetooth interface resets via system commands if connection issues occur, ensuring high availability in real-world deployments.

7.3.3. Data Processing and Ingestion Module

The data_processor.py script implements the background service responsible for reading sensor data from the local JSON file and writing it to the InfluxDB time-series database. It operates on a continuous loop, ensuring the persistence and availability of environmental data collected by the ESP32.

The script begins by securely loading configuration parameters (such as database URL, token, and bucket) from environment variables. It then reads sensor data from the shared JSON file (sensor_data.json), applying file locking mechanisms to prevent concurrent access issues.

To avoid redundant writes, a counter (COUNT) included in the JSON file is used to verify whether new data has been received since the last transmission. If the data is new, it is parsed and structured into measurement points (Point objects), each tagged by sensor type (SEN55, SGP30, S88) and location, and containing relevant environmental metrics (e.g., PM values, humidity, VOCs, CO_2).

These points are then written synchronously to InfluxDB using the official client API. The process repeats every 5 seconds, providing a lightweight and reliable bridge between the local data acquisition layer and the centralized storage system.

This modular pipeline—from sensor acquisition on the ESP32, through wireless transmission to the Raspberry Pi, and finally to structured storage in InfluxDB—ensures end-to-end data integrity and scalability. By decoupling each stage via standardized interfaces (BLE characteristics and JSON files), the system remains resilient to faults and flexible for future extensions, such as cloud synchronization or real-time analytics dashboards.

8. Results and Evaluation

The AirSense system was tested extensively in a real indoor environment—specifically, in university amphitheaters and classrooms. Over several monitoring sessions, sensor data was successfully collected, transmitted, stored, and visualized in real-time, validating the effectiveness of the entire system architecture.

8.1. Data Collection and Visualization

Air quality metrics such as PM1.0, PM2.5, PM10, VOCs, NOx, temperature, humidity, and CO_2 equivalents were continuously gathered by the sensor hub and transmitted to the Raspberry Pi over BLE. This data flow was handled efficiently, ensuring minimal loss and maintaining a consistent update frequency. Once received, the data was parsed and structured, then stored in InfluxDB. Through integration with Home Assistant, this information was visualized in dynamic dashboards, offering users real-time insights alongside long-term historical trends.

These visualizations not only made the data more accessible but also enabled detailed analysis of environmental conditions across different times of day, occupancy levels, and activity types, supporting informed decision-making for indoor air quality management.

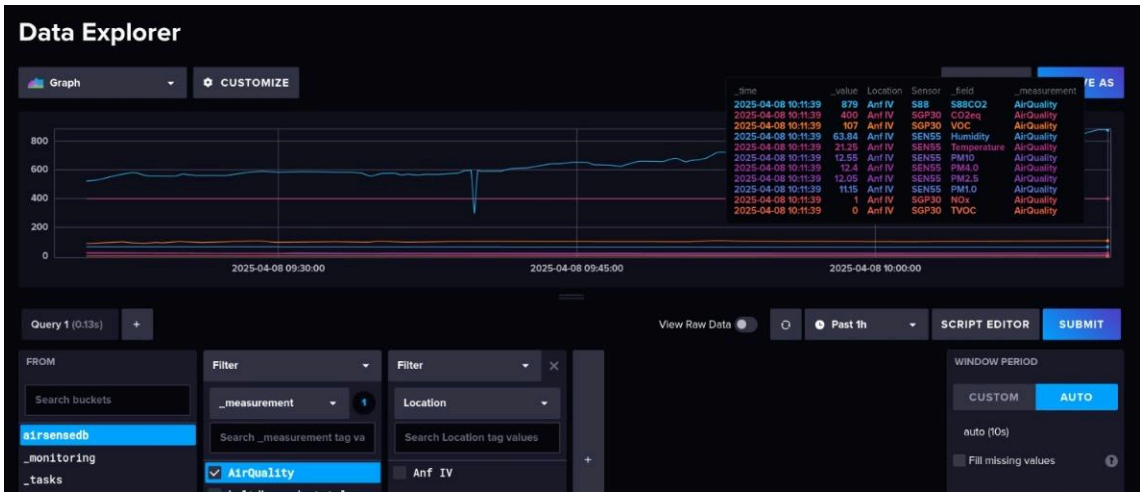


Figure 3 – CO_2 Concentration Over Time in an Occupied Indoor Environment

8.2. Challenges Encountered

While the system was robust overall, several challenges emerged:

- **Data Duplication**: Early versions of the script experienced occasional duplicate readings due to timing mismatches. This was resolved by tracking a COUNT value in the sensor data and discarding previously seen entries.
- **Parsing Issues**: Some malformed BLE messages caused parsing failures. Error handling was added to skip over corrupted entries and prevent crashes.
- **Power Management**: As discussed earlier, the decision to avoid continuous monitoring allowed the system to operate more efficiently on battery power, but required careful user activation and shutdown procedures.

8.3. Air Quality Observations

The graphs generated over time revealed insightful environmental patterns. For instance:

- **CO_2 equivalent levels** rose steadily during periods of human occupancy and dropped after ventilation.
- **Humidity and temperature** remained stable, but showed sharp changes depending on HVAC system usage.
- **PM concentrations** spiked during cleaning activities or open windows near traffic, highlighting the system's responsiveness to environmental changes.

These results confirm that AirSense can serve not only as a monitoring tool but also as a feedback mechanism for improving indoor air quality through ventilation or occupancy planning.

Figure 4 – Home Assistant Dashboards

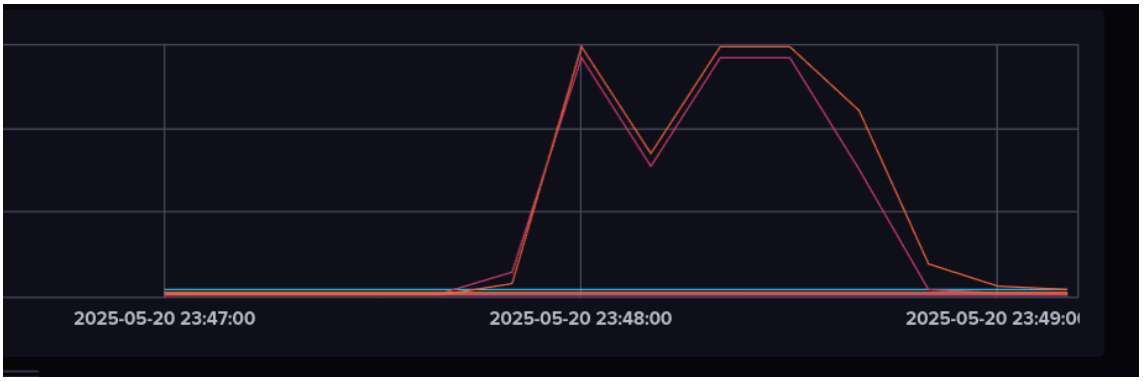


Figure 5 – Sudden CO_2 Spike Triggered by Alcohol Proximity

This figure illustrates a sharp and temporary spike in the reported CO_2 concentration when a source of alcohol (ethanol) was brought near the sensor during a controlled test. The observed spike is due to the way many low-cost CO_2 sensors (often referred to as "eCO_2" sensors) estimate CO_2 levels indirectly, based on volatile organic compounds (VOCs) like ethanol. Specifically, these sensors use internal algorithms that correlate VOC readings — particularly from gases like ethanol — to equivalent CO_2 levels, based on assumed indoor activity patterns. As a result, the presence of alcohol vapors near the sensor leads to a misleadingly high CO_2 reading, despite there being no actual increase in carbon dioxide concentration. This experiment highlights one of the known limitations of eCO_2 estimation methods and reinforces the importance of interpreting sensor data in context.

9. Discussion and Analysis

9.1. What Went Well and Why

Several aspects of the AirSense project were executed successfully, thanks to a combination of clear planning, modular design decisions, and appropriate technology choices. One of the most notable achievements was the seamless integration of hardware and software components. The ESP32 microcontroller paired with the Raspberry Pi offered a highly effective balance between data acquisition and local processing. The ESP32's ability to transmit sensor data via Bluetooth Low Energy (BLE) proved to be reliable in short-range deployments, while its low power consumption made it suitable for portable and semi-permanent setups.

The decision to use a modular architecture also paid off significantly. By designing the system to support multiple sensors with independent data streams—such as the SEN55 for particulate matter and temperature/humidity, the SGP30 for VOCs and NOx, and the S88 for CO_2—AirSense became adaptable to a range of environments and use cases. This flexibility allows users to tailor the hardware configuration based on the specific monitoring needs of a given room or institution.

The use of local data processing and storage was another strategic success. By running InfluxDB on the Raspberry Pi via Docker and visualizing the data through Home Assistant, the project ensured real-time accessibility and user-friendly dashboards without relying on external cloud infrastructure. This choice enhanced data privacy, reduced latency, and simplified network configurations, making the solution more deployable in environments with limited connectivity.

Moreover, the decision to separate BLE communication from the database logic enhanced the system’s resilience. It allowed the Raspberry Pi to buffer and parse data even in cases where there might be temporary issues with database connectivity, and also opened the door for easier debugging and future scalability.

Lastly, the user experience was kept in mind throughout development. Whether through clear sensor naming, location tagging, or intuitive Home Assistant dashboards, the system succeeded in providing meaningful, real-time insights that can be understood even by non-technical users. This aligns with the project’s educational and practical goals, especially in environments like classrooms or public institutions.

9.2. Limitations of the Solution

While AirSense successfully meets its core objectives of affordability, modularity, and transparency, the current implementation presents limitations that may affect its applicability in certain scenarios or long-term use.

A key constraint lies in the **local-only data storage** and processing model. Although running InfluxDB and Home Assistant on a Raspberry Pi ensures data privacy and independence from cloud services, it introduces drawbacks such as limited storage capacity, higher data loss risk in case of hardware failure, and lack of native remote access when outside the local network. Additionally, scaling the system to multiple nodes in different locations requires manual setup or complex networking, which complicates centralized monitoring.

Bluetooth Low Energy (BLE), despite being power-efficient and simple to implement, also introduces constraints. Its limited range and susceptibility to signal degradation in obstructed indoor environments restrict sensor placement flexibility and may reduce data transmission reliability. The current version lacks robust auto-reconnection logic, meaning temporary disconnections might cause delays or require manual intervention.

Power supply considerations further influence system performance. Although using portable power banks enhances mobility, it limits operational duration unless frequent recharging is performed. Choosing not to keep the system always-on reduces power consumption but also lowers temporal resolution, which may impact applications requiring continuous monitoring.

Sensor calibration is another relevant concern. AirSense sensors include built-in auto-calibration features, which help maintain accuracy over time without user intervention. However, for long-term or critical deployments, periodic cross-validation with reference-grade instruments would be advisable. While such comparisons were considered during development, formal external calibration was not implemented due to logistical constraints. Thus, while readings are reliable for indicative purposes and observing trends, they may not satisfy strict regulatory standards for precision.

Lastly, the lack of user management features limits deployment in multi-user or institutional contexts. Beyond a prototype Android setup via QR code, no authentication or access control mechanisms exist. Future versions would benefit from an advanced configuration portal, user authentication, or cloud-syncing capabilities to reach production-grade maturity.

Despite these limitations, most were anticipated and remain addressable in future iterations. Importantly, none compromise the system’s main goal: delivering a functional, modular, and transparent platform for indoor air quality monitoring.

9.3. Technical Challenges

Throughout the development of AirSense, several technical challenges emerged that required iterative problem-solving and adaptation. One of the earliest and most persistent challenges was the stability of Bluetooth Low Energy (BLE) communication between the ESP32 and the Raspberry Pi. Despite BLE being an ideal choice for low-power, short-range transmission, it is inherently sensitive to environmental interference, device pairing issues, and connectivity drops. Ensuring a reliable and persistent connection required careful tuning of retry mechanisms, connection intervals, and exception handling to avoid data loss during transmission.

Another challenge involved parsing and managing sensor data efficiently. The ESP32 transmits data in a compact format, which needed to be robustly parsed on the Raspberry Pi. This process had to be resilient to occasional malformed or partial packets, which could occur due to noise or delays in BLE transmission. Structuring the parsed data into a clean, accessible JSON format required clear design decisions to balance readability, performance, and fault tolerance.

Setting up the InfluxDB instance for time-series storage also presented its own complexities. While Docker simplified deployment, careful configuration was necessary to ensure persistent volumes, secure access credentials, and adequate performance for writing data at frequent intervals. Additionally, tuning the InfluxDB client to manage write precision and optimize batch uploads helped reduce overhead and improve the consistency of database entries.

Lastly, designing the system to be modular and maintainable without introducing tight coupling between components was a non-trivial task. By separating BLE communication from database logic via the use of intermediary JSON files, the system gained robustness and extensibility, but this also required added synchronization handling to avoid concurrency issues or race conditions when accessing shared files.

Despite these challenges, the solutions implemented proved effective and laid a solid foundation for future improvements. Each technical difficulty pushed the system architecture toward greater resilience, modularity, and reliability.

9.4. Scalability and extensibility

AirSense was designed with modularity at its core, enabling both vertical and horizontal scalability. On the hardware side, the system supports the addition of new sensors — such as particulate matter, VOCs, or even sound and light sensors — without requiring architectural changes. Multiple ESP32 nodes can also be deployed in parallel to monitor larger or multiple rooms, with all data aggregated by a central Raspberry Pi hub or synced across devices. Also particularly the ability to add or replace sensors ensures that the system can be adapted to diverse indoor environments with varying monitoring needs with minimal changes to its core infrastructure.

From a software standpoint, the loosely coupled design allows for easy integration of new data pipelines, notification systems, or external APIs. The use of Docker containers and InfluxDB ensures the solution can be scaled up to cloud-hosted environments or larger networks without restructuring the codebase. Visualization and control via Home Assistant can also be extended with more advanced dashboards or automations which enables flexible dashboard customization, and the Android application introduces an intuitive setup process through QR code scanning. Future updates could expand its role to include mobile notifications, remote monitoring, and user authentication, further enhancing usability and deployment flexibility.

This foundation allows AirSense to evolve into a more comprehensive environmental monitoring solution, making it suitable not only for academic settings but also for larger institutional or commercial applications.

9.5. Critical Questions for Further Analysis

How reliable is BLE communication in large-scale or crowded environments?

While BLE provided stable and low-latency data transmission in our controlled environment, real-world settings with physical obstructions, multiple devices, or radio interference could degrade performance. Future tests should quantify BLE's effective range and robustness in such scenarios to ensure consistent data flow.

How do sensors behave over long-term deployments?

The sensors used in AirSense include built-in auto-calibration features, which help maintain accuracy over time. However, environmental factors like dust accumulation or temperature fluctuations may still cause drift. Incorporating periodic cross-validation with reference instruments or advanced calibration routines would further enhance long-term reliability.

Can the system support multiple users and maintain data privacy?

Currently, data is processed locally without user differentiation. To expand to multi-user environments, implementing secure user authentication and role-based access control will be necessary to ensure data privacy and prevent unauthorized access.

What are the implications of integrating cloud synchronization?

Integrating cloud storage would enable remote monitoring and centralized data aggregation, but introduces concerns about data security, latency, and potential dependence on internet availability. Ensuring encrypted transmission, compliance with privacy regulations, and failover mechanisms will be key considerations.

How feasible is the addition of mobile app interfaces for user interaction?

A mobile app would improve accessibility and user engagement, providing real-time alerts and data visualization on personal devices. Designing this interface requires balancing usability, data synchronization efficiency, and secure authentication, especially for battery-powered sensor nodes.

10. Conclusion

The AirSense project set out to design and implement a low-cost, modular, and transparent air quality monitoring system suitable for educational and institutional indoor environments. Built around the ESP32 microcontroller and Raspberry Pi, the system leverages Bluetooth Low Energy for efficient wireless communication and stores sensor data locally in an InfluxDB instance, enabling powerful real-time visualization through Home Assistant. This architecture allowed the project to avoid reliance on cloud services, emphasizing privacy, transparency, and local control, while also offering flexibility for future extensions.

Throughout the project, several key achievements were realized. The modular hardware design supports various sensors such as the SEN55, SGP30, and S88, which collectively provide a comprehensive view of indoor air conditions—including particulate matter, volatile organic compounds, temperature, humidity, and CO_2 levels. The BLE-based wireless communication proved energy-efficient and reliable for short-range transmissions, and the integration of InfluxDB with Home Assistant enabled user-friendly, interactive dashboards that visualize real-time and historical air quality trends. These tools were particularly effective in demonstrating how human presence impacts indoor air conditions, as seen in logged data showing CO_2 accumulation in occupied rooms.

Despite these successes, the project encountered several challenges. BLE connection stability required careful error handling, sensor data formatting needed standardization, and power management was a recurring concern. Initial attempts at continuous system uptime using power banks revealed the need for more efficient sleep/wake cycles or periodic operation strategies. These limitations were addressed through iterative improvements, including decoupling data acquisition from processing and adopting periodic data transmission to conserve power.

Valuable lessons were learned in areas such as embedded system communication, time-series database optimization, and the design of user-centric interfaces. The importance of modularity—for both hardware and software—became increasingly clear as it enabled quick adaptation to new use cases, sensor changes, or deployment scenarios. Moreover, the emphasis on local data handling highlighted the benefits of retaining control over sensitive information while still enabling future cloud-based extensions if needed.

Overall, AirSense demonstrates that affordable, flexible, and scalable air monitoring systems can be built using open-source technologies and low-cost hardware. It bridges the gap between accessibility and functionality, and lays the groundwork for future enhancements such as cloud synchronization, mobile app integration, OTA updates, and broader deployment across learning environments or smart building infrastructures.

11. Future Work

While the AirSense system offers a robust, modular, and affordable platform for indoor air quality monitoring, several avenues for future improvements and extensions remain to enhance its functionality, usability, and scalability.

One key area for development is the enhancement of the mobile application. Currently, the system's primary user interface is through Home Assistant dashboards and web-based visualizations. Expanding the mobile app with richer features such as real-time alerts, personalized notifications, sensor configuration controls, and simplified onboarding will greatly improve user engagement and accessibility, especially for non-technical users.

The system's sensor array can also be expanded to include additional environmental parameters, such as formaldehyde (HCHO), ozone (O_3), or particulate matter of different size ranges. Integrating auto-calibration routines for existing sensors like the SGP30 and SEN55 would improve measurement accuracy over time without manual intervention, enhancing the reliability of long-term monitoring.

Cloud synchronization represents another promising future enhancement. While the current setup emphasizes local data processing and storage for transparency and privacy, optionally integrating secure cloud services would enable remote access, large-scale data aggregation, and collaborative data sharing. A cloud-based platform could also support a user authentication system, allowing different users to log in, view personalized dashboards, manage devices remotely, and access only the data relevant to their roles or locations. This would be especially beneficial for deployments across multiple classrooms, buildings, or institutional networks, where data access needs to be structured and scalable.

An MQTT module for data transmission over Wi-Fi was developed but was not tested on a public network. As a next step, the deployment of a university-hosted virtual machine is planned to further develop and validate this module, ensuring a stable connection for data transmission across the campus. With this implementation, any classroom equipped with a hub will be able to transmit sensor data, which can then be accessed in real time by authorized users connected to the university's Wi-Fi network (eduroam).

Advances in data visualization could further improve the user experience. Developing more sophisticated, customizable dashboards with predictive analytics, air quality indexing, and historical trend comparisons would empower users to make better-informed decisions and foster greater awareness of indoor air quality dynamics.

Lastly, improving system autonomy by developing a fully battery-operated version, optimized for low power consumption, is essential for flexible, mobile deployments. Incorporating remote Over-The-Air (OTA) update capabilities would simplify maintenance and feature updates, reducing the need for physical access to the devices.

Together, these future enhancements would build on AirSense's core strengths—modularity, transparency, and affordability—delivering a more powerful, user-friendly, and scalable air quality monitoring platform suited to diverse real-world applications.

12. Bibliography

EPA, "Indoor Air Quality," [Online]. Available: https://www.epa.gov/report-environment/indoor-air-quality#:~:text=Pollutants%20and%20Sources,-Typical%20pollutants%20of&text=Combustion%20byproducts%20such%20as%2

EPA, "Indoor Particulate Matter," [Online]. Available: https://www.epa.gov/indoor-air-quality-iaq/indoor-particulate-matter

Government of Canada, "Carbon dioxide in your home," [Online]. Available:

https://www.canada.ca/en/health-canada/services/publications/healthy-living/carbon-dioxide-home.html

Department of Health, "Carbon Dioxide (CO2)," [Online]. Available:

https://www.health.state.mn.us/communities/environment/air/toxins/co2.html

Airgradient, "Low Cost CO2 Sensors Comparison: Photo-Acoustic vs NDIR," [Online]

Available:https://www.airgradient.com/blog/co2-sensors-photo-acoustic-vs-ndir-updated/#:~:text=Photo-acoustic%20sensors%20use%20the,the%20absorption%20with%20a%20microph

"ESP32 DevKit ESP32-WROOM GPIO Pinout," [Online]. Available:

https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/

A. Ramos, V. B. Jesus, C. Gonçalves, F. Caetano and C. Silveira, "Monitoring Indoor Air Quality and Occupancy with an IoT System: Evaluation in a Classroom Environment," 2023 18th Iberian Conference on Information Systems and Technologies (CISTI), p. 6, 2023

G. Marques and R. Pitarma, "An Internet of Things-Based Environmental Quality," MDPI, p. 18, 2019

H. Fritz, S. Bastami, C. Lin, K. Nweye, T. To, L. Chen, D. Le, A. Ibarra, W. Zhang, J. Y. Part, W. Waites, M. Tang, P. Misztal, A. Novoselac, E. Thomaz, K. Kinney and Z. Nagy, "Design, fabrication, and calibration of the Building EnVironment and," Building and Environment, vol. 222, p. 23, 2022

S. Ward, M. Gittens, N. Rock and K. James, "CampusEMonitor: Intelligent Campus Environment Room," SIGUCCS '19: Proceedings of the 2019 ACM SIGUCCS Annual Conference, p. 8, 2019

P. K. Sharma, A. Mondal,, S. Jaiswal,, M. Saha,, S. Nandi,, T. De and S. Saha, "IndoAirSense: A framework for indoor air quality estimation," Atmospheric Pollution Research, vol. 12, no. 1, p. 13, 2021

B. W. Dionova, D. Hendrawati, M. N. Abdulrazaq, D. J. Vresdian, A. A. Hapsari, M. I. Abdullah and L. P. Pratama, "Design and Simulation of Environment Indoor," 2023 International Seminar on Intelligent Technology and Its Applications (ISITIA), p. 6, 2023

J. Rosenberger, Z. Guo,, A. Coffman, D. Agdas and P. Barooah, "An Open-Source Platform for Indoor Environment Monitoring," Sensors, p. 17, 2022

A. Maier, A. Sharp and Y. Vagapov, "Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things," 2017 Internet Technologies and Applications (ITA), 2017

sarah e. west; lucy way; rhys archer; victoria j. beale; sam bland; henry burridge; claudia castro-faccetti; lia chatzidiakou; prashant kumar; carolanne vouriot; natalie williams; samhe project consortium (2023) Co-Designing an Air Quality Web App with School Pupils and Staff: The SAMHE Web App

AirSense (2025). ***AirSense: Indoor Air Quality Monitoring System*** **[Source code]. GitHub.** https://github.com/luisbfsousa/Airsense

AirSense (2025). ***AirSense Project Website***. https://luisbfsousa.github.io/Airsense/