**DUNDALK INSTITUTE OF TECHNOLOGY**

**A logo with a bird and text

Description automatically generated**

**Alpha Release Technical Documentation On**

**AEROSENSE – SMART HUB FOR AIR QUALITY MONITORING**

Project Carried Out

**By**

**D00251785 PATRICK ORJIEH**

**D00244618 ROBERT NUGENT**

**D00261104 CONOR MC GINN**

**D00240060 HANNAH MC ELROY**

Under the Supervision of

**DR. JOHN LOANE**

DEPARTMENT OF COMPUTING AND MATHEMATICS

**SCHOOL OF INFORMATICS AND CREATIVE ARTS**

**DEPARTMENT OF VISUAL AND HUMAN-CENTRED COMPUTING**

**EXECUTIVE SUMMARY**

The Alpha release of the Aerosense project represents a key phase in the development of our Smart Indoor Air Quality Hub. This initial release is a comprehensive prototype that demonstrates the capability of our system to assist individuals with asthma in monitoring the air quality within indoor environments.

Our system's architecture is detailed through clear diagrams that illustrate the interactions between the IoT components, the cloud-based web server, and the user interface. These diagrams serve to clarify the flow of data and the security measures in place to protect it.

The prototype, hosted on an AWS cloud server, showcases the core functionalities that will be present in the final product. It includes a working model of the IoT hardware, the software on the web server, and the database that stores and processes the gathered data. The use of PubNub ensures secure and efficient communication between the IoT devices and the web server.

We have taken significant steps to secure the application, with particular attention to the protection of data while it is stored and as it moves through the system. The security protocols we have implemented are critical in maintaining the privacy and integrity of user data.

The documentation for the Alpha release reflects the steps to creating a user-friendly and secure system. It outlines our approach to addressing the needs of asthma patients by providing them with actionable insights into their indoor air quality.

In summary, the Alpha release documentation shows the need of the Aerosense project and sets a clear direction for future development. It shows the first step to delivering a product that is of value to users, particularly those managing asthma in their daily lives.

Table of Contents

[**GLOSSARY** 4](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932930)

[**LIST OF TABLES AND DIAGRAMS** 8](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932931)

[**List Of Tables** 8](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932932)

[**List Of Diagrams** 8](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932933)

[**1. INTRODUCTION** 9](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932934)

[**2. SYSTEM ARCHITECTURE** 10](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932935)

[**Overview** 10](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932936)

[**IoT Elements** 10](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932937)

[**Data Communication via PubNub** 11](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932938)

[**Cloud Server and Database** 11](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932939)

[**Firebase Authentication** 11](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932940)

[**Mobile Application** 12](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932941)

[**Community Data Sharing** 12](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932942)

[**Conclusion** 12](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932943)

[**Fritzing** 13](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932944)

[**3. Alpha Prototype Documentation** 15](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932945)

[**Introduction** 15](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932946)

[**Hardware** 15](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932947)

[**Webserver** 15](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932948)

[**Hosted on AWS** 16](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932949)

[**PubNub Communication** 16](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932950)

[**Database** 16](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932951)

[**Security** 16](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932952)

[**3. Cloud Server Description** 17](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932953)

[**Endpoints** 17](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932954)

[**4.PubNub** 18](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932955)

[**CONCLUSION** 18](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932956)

[**REFERENCES:** 18](file:///C:\Users\rober\Downloads\Alpha_Release_Documentation.docx#_Toc151932957)

## **GLOSSARY**

* **Aerosense**: A project aimed at developing a portable device (Smart Indoor Air Quality Hub) for monitoring indoor air quality, particularly beneficial for individuals with asthma.
* **Smart Indoor Air Quality Hub**: A portable device designed to provide real-time data and insights about indoor air quality, helping individuals, especially those with asthma, to understand and manage their environment better.
* **PubNub**: A cloud-based service that provides real-time data streaming and messaging solutions, used in Aerosense for real-time communication between the device and the server.
* **AWS (Amazon Web Services)**: A comprehensive and widely adopted cloud platform that offers various services such as computing power, database storage, and content delivery. In Aerosense, it's used for data processing and storage.
* **Alpha Prototype**: An early version of a product that is functional enough to demonstrate the concept and design but may not have all the final features and polish of the final product.
* **Cloud Server**: Remote servers accessed over the internet used to store, manage, and process data, as opposed to a local server or personal computer.
* **User Interaction**: The process and experience of a person engaging with the Aerosense system, particularly through its mobile application.
* **Sensor Data Acquisition**: The process of collecting data from various sensors (like those measuring air quality) used in the Aerosense system.
* **Raspberry Pi**: A small, affordable computer used for various programming and electronics projects. In Aerosense, it serves as the central unit for collecting sensor data.
* **PMS7003 Particle Sensor**: A sensor that measures particulate matter (PM) in the air. It's crucial for providing data on air quality, especially for asthma patients.
* **BME680 Sensor**: This sensor provides a range of environmental data, including temperature, humidity, and volatile organic compounds (VOCs), which are chemicals that can affect air quality.
* **Grove Air 530 GPS Sensor**: A sensor used for determining geographical location, enabling the Aerosense system to tag air quality readings with specific locations.
* **PubNub**: A cloud-based service that provides real-time data communication between devices and servers. It's used in Aerosense for transmitting sensor data from the Raspberry Pi to the cloud server.
* **PubNub Access Keys**: Unique identifiers used for secure access to PubNub's publish and subscribe channels, ensuring that only authorized devices and servers can communicate through the platform.
* **AWS Cloud**: Amazon Web Services Cloud, a cloud computing platform that hosts the server for the Aerosense system. It's responsible for data processing and storage.
* **AES-256 Encryption**: A method of encrypting data to protect it from unauthorized access. It's used in Aerosense to secure data at rest in the database.
* **TLS Protocol**: Transport Layer Security protocol, a method for encrypting data during transmission to prevent interception by unauthorized parties. Used in Aerosense for secure data transfer.
* **Firebase Authentication**: A Google service that provides user authentication for applications. In Aerosense, it manages user login processes and secures access to user data.
* **Google OAuth**: An authentication method provided by Google that allows users to log in to applications securely using their Google account.
* **Mobile Application**: The user interface of the Aerosense system, available on smartphones. It displays air quality data and allows users to interact with the system.
* **Community Data Sharing**: A feature in Aerosense where users can share anonymized environmental data to contribute to broader air quality studies.
* **GDPR (General Data Protection Regulation):** A regulation in EU law on data protection and privacy. In Aerosense, it's referenced to highlight the system's compliance with data privacy laws.
* **Power Bank**: A battery pack that provides electricity to the Raspberry Pi so it can work without being plugged into a wall.
* **Breadboard**: A board for making an electrical circuit without soldering, useful for testing parts of a circuit.
* **Vibrating Motor Disc**: A small disc that shakes to give a physical alert when the air quality changes.
* **LED Indicator**: A light that changes color to show different air quality levels.
* **Buzzer**: A device that makes a beeping sound when the air quality reaches a level that could be concerning.
* **USB to UART Converter**: A tool that lets the Raspberry Pi talk to the sensors using USB ports, which are very common on computers (TX & RX).
* **Wiring**: Cables that connect different parts of the electrical circuit in the diagram, each with a different color for easy identification.
* **Flask**: A lightweight web application framework written in Python, used for creating the web server in the Aerosense project.
* **Cron Job**: A scheduled task in Unix-like operating systems. In Aerosense, it's used to periodically trigger data collection from the sensors.
* **Encryption**: The process of converting information or data into a code to prevent unauthorized access. Used in Aerosense to secure data transmission.
* **HTTPS (Hypertext Transfer Protocol Secure):** An extension of HTTP used for secure communication over a computer network. In Aerosense, it's used for secure communication between the client application and the server.
* **Database Schema**: The structure of a database system, described in a formal language. In Aerosense, it defines the structure of the MySQL database.
* **Real-time Data Transfer**: The process of continuously transferring data as it's collected without delay. In Aerosense, this is facilitated by PubNub.
* **API (Application Programming Interface):** A set of protocols for building and interacting with software applications. Aerosense uses APIs to enable communication between different components of the system.
* **Volatile Organic Compounds (VOCs):** Organic chemicals that have high vapor pressure at room temperature and can affect air quality. Measured by the BME680 sensor in Aerosense.
* **MySQL:** An open-source relational database management system. In Aerosense, it's used to store and manage data.
* **AWS EC2 (Amazon Web Services Elastic Compute Cloud**): A web service that provides resizable compute capacity in the cloud, used to host the server for Aerosense.

## **LIST OF TABLES AND DIAGRAMS**

### **List Of Tables**

### **List Of Diagrams**

Diagram 1: System Architecture Diagram of Aerosense Application…................................... 8

Diagram 2: Updated Fritzing Diagram: Sensor Connections with Pi .....................................11

## **1. INTRODUCTION**

The Aerosense project is an initiative designed to offer an innovative solution to air quality monitoring, specifically addressing the needs of individuals with asthma. The project aims to develop a portable "Smart Indoor Air Quality Hub" that provides real-time insights into indoor air quality, which is crucial for people who suffer from respiratory conditions.

This documentation outlines the development process, architecture, and functionalities of the Aerosense system. It details the collaborative effort put forth to design a system that combines sensor data acquisition, cloud-based processing, and user interaction through a mobile application.

Our goal is to create a system that is user-friendly, aligning with our educational objectives and the practical needs of asthma patients. The project leverages the capabilities of the Raspberry Pi as a data collection point, PubNub for real-time communication, and AWS cloud services for data processing and storage.

The other sections will provide an overview of the system architecture, describe the alpha prototype, discuss security measures, demonstrate the deployment on a cloud server.

As the developers of Aerosense, we have committed ourselves to a practical and simplified approach, ensuring that the end product is not only functional but also accessible to our target users. We believe that the integration of real-time environmental data with health management tools can significantly improve the daily lives of asthma patients, and this belief has been the driving force behind our project as well as the academic part.

## **2. SYSTEM ARCHITECTURE**

A diagram of a cloud computing system

Description automatically generated

Diagram 1: System Architecture Diagram of Aerosense Application

### **Overview**

The Aerosense project's system architecture is a comprehensive framework designed to monitor and analyse air quality, tailored specifically for asthma patients. This architecture integrates various technologies including Internet of Things (IoT) devices, cloud computing, real-time data communication, and a mobile application interface.

### **IoT Elements**

The core of the Aerosense system is based around a Raspberry Pi, which acts as the central data collection unit. Attached to the Pi are three crucial sensors:

1. **PMS7003 Particle Sensor**: This sensor is responsible for measuring particulate matter in the air, providing data on air quality that is particularly relevant to asthma sufferers.
2. **BME680 Environmental Sensor**: Offers a broader range of environmental readings including temperature, humidity, and volatile organic compounds (VOCs).
3. **Grove Air 530 GPS Sensor**: Used to provide location data, enabling tagged air quality readings, which is essential for mapping and analysing environmental conditions in different areas.

### **Data Communication via PubNub**

Data from these sensors is transmitted using PubNub, a real-time communication platform. The Raspberry Pi publishes the sensor data to a dedicated PubNub channel, which is then subscribed to by the cloud server. This setup ensures the delivery of data from the sensors to the server for processing.

To secure this communication:

* **PubNub Access Keys**: Unique access keys are used for both publishing and subscribing to the data, ensuring that only authorized devices and servers can access the communication channel.
* **Bidirectional Communication**: Certain scenarios allow for two-way communication between the cloud server and the Raspberry Pi, facilitating real-time updates and commands.

### **Cloud Server and Database**

The server hosted on AWS Cloud forms the backbone of data processing and storage. It subscribes to the PubNub channel to receive sensor data, processes this data according to predefined processing algorithms that will be made by us, and stores it in a secure database. The database maintains records of air quality readings, user profiles, and other relevant data.

For securing data:

* **At Rest**: The database employs encryption mechanisms like AES-256 to secure data at rest.
* **In Transit**: Data transferred between the IoT devices, server, and the mobile application is encrypted using TLS protocols by PubNub, ensuring secure data transmission.

### **Firebase Authentication**

User authentication is managed by Firebase, which integrates Google OAuth for a secure and convenient login process. Firebase provides authentication tokens that are used to verify user identity and secure access to the mobile application and personal data.

### **Mobile Application**

The client-side of the Aerosense system is a mobile application that serves as the user interface. It allows users to view real-time air quality data, receive alerts, and manage their profiles. The app fetches data from the cloud server, user interactions with the app are also sent back to the server for processing and response.

### **Community Data Sharing**

One of the innovative features of Aerosense is its community data sharing aspect. Users can choose to share anonymized environmental data, contributing to a broader understanding of air quality trends. This data is coupled by the server and can be accessed for public awareness purposes.

### **Conclusion**

The Aerosense system architecture has been designed to ensure a flow of data from the sensors to the end-user. Security protocols are in place at every step to protect sensitive information, especially considering the health-related nature of the data (GDPR).

### **Fritzing**

A diagram of a circuit board

Description automatically generated

Diagram 2: Updated Fritzing Diagram: Sensor Connections with Pi

**Fritzing Diagram Description for Aerosense Project**

The Fritzing diagram provides a visual representation of our Aerosense hub's electronic setup. This setup is crucial for monitoring air quality and ensuring the system is interactive and responsive to the environment it monitors.

1. **Raspberry Pi 400**: The heart of our project, the Raspberry Pi 400, serves as the microcontroller that processes data from the attached sensors. It also acts as the communication bridge between the sensors and our cloud services.
2. **Power Bank**: A portable power bank supplies electricity to the Raspberry Pi, ensuring that our hub remains switched on even without a direct power outlet. This helps the portability of our device.
3. **Breadboard**: We've used a breadboard for mounting the electronic components. This allows for a modular and non-permanent setup, making it easy to test the system.
4. **PMS7003 Particle Sensor**: This sensor detects particulate matter (PM) levels in the air, crucial for assessing air quality. The sensor's data output is sent to the Raspberry Pi.
5. **BME680 Sensor**: The BME680 provides comprehensive environmental readings, including humidity, temperature, and VOCs. These factors are vital for a good understanding of indoor air quality, particularly for users with asthma.
6. **Grove Air 530 GPS Sensor (Air530)**: This GPS module provides geolocation data, enabling our system to tag air quality measurements with precise locations. This feature is key for tracking and analysing environmental conditions across different indoor settings.
7. **Vibrating Motor Disc**: Added for haptic feedback, the vibrating motor disc, offers an alert option alongside the buzzer and notification. This ensures all users receive prompt notifications regarding air quality changes.
8. **LED Indicator**: The multi-color LED indicator displays real-time air quality status. It's connected through a resistor to the GPIO pin on the Raspberry Pi to manage voltage and current flow, ensuring the LED operates safely.
9. **Buzzer**: The buzzer serves as an audible alert mechanism, making a sound when air quality thresholds are reached. It's also connected to a GPIO pin and programmed to trigger alongside the LED indicator.
10. **USB to UART Converter**: This converter is critical for serial communication between the Raspberry Pi and the sensors that require UART (TX & RX) communication. It facilitates the transfer of sensor data to the Raspberry Pi for processing.
11. **Wiring**: The diagram shows all necessary wiring connections, including power lines, ground connections, and data lines. Each wire is color-coded for easy identification of the circuit flow.

## **3. Alpha Prototype Documentation**

### **Introduction**

This section shows the development and features of our alpha prototype for the Aerosense project. Our prototype integrates Internet of Things (IoT) hardware, a web server hosted on AWS, and PubNub communication for real-time data transfer. We've focused on building a system that demonstrates the core parts of aerosense.

### **Hardware**

For the IoT hardware, we chose to use the BME680 sensor and the PMS7003 particulate matter sensor. However, during development, we encountered a challenge with the PMS7003 sensor, due to an incompatible IDC adapter. Now, our current prototype primarily relies on the BME680 sensor, which measures air quality, temperature, humidity, and gas resistance.

Our approach has been to keep the hardware setup as simple as possible while ensuring reliable data collection. The BME680 sensor connects to a Raspberry Pi, serving as our primary data collection point. The Raspberry Pi is programmed to periodically read sensor data, preparing it for transfer to our web server.

### **Webserver**

The core of our prototype is a Flask-based web server. This server is responsible for handling requests from both the IoT hardware and the client application. It processes incoming sensor data, interacts with the database, and manages user requests.

The web server's primary functions include:

* Receiving encrypted sensor data from the IoT devices.
* Decrypting and processing this data for storage.
* Serving processed data to the user interface upon request.
* Handling user authentication and session management.

### **Hosted on AWS**

Our web server is hosted on Amazon Web Services (AWS), The domain **https://www.aerosense.life** is custom-configured to point to our AWS-hosted server, offering a the needed endpoints for out project.

### **PubNub Communication**

For real-time data transfer between the IoT devices and the server, we employed PubNub. PubNub is efficient and reliable to offer real-time messaging system. Our implementation uses PubNub to:

* Securely transmit encrypted sensor data from the Raspberry Pi to the server.
* Enable real-time updates in the client application, particularly for live air quality readings.
* User can change the settings of the Hub as well.

The server runs a background task that constantly listens for incoming data on the PubNub channel, ensuring that new sensor readings are immediately captured and processed.

### **Database**

We are using a MySQL database hosted on the same AWS server. This database stores user data, sensor readings, asthma profile and settings. Our database schema is designed to efficiently store and retrieve data.

Key aspects of our database implementation include:

* Tables for user data, sensor readings, and device configurations.
* Efficient indexing to speed up queries, especially for retrieving the latest sensor data.
* Secure storage of user credentials and personal data.

### **Security**

Given the sensitivity of user data and the potential dangers in IoT systems. Our security measures include:

* Encryption of data transmitted between the IoT devices and the server.
* Secure handling and storage of user credentials using Firebase.
* Implementation of HTTPS for secure communication between the client application and the server.

## **3. Cloud Server Description**

Our Aerosense project uses a cloud server hosted on Amazon Web Services (AWS), which handles various aspects of the system. This server plays a role in processing, storing, and managing data received from IoT devices and user interactions. Below is a detailed overview of our cloud server's functionalities and the endpoints implemented:

### **Endpoints**

* **/ (Root Endpoint):** This endpoint serves a simple purpose: when accessed, it returns a "Hello, World!" message. This was primarily created for initial testing to ensure that our server setup was successful and responding to HTTP requests.
* **/api/register:** This endpoint is integral for user registration. It leverages Firebase for secure and efficient user authentication. When a new user registers through our mobile application, this endpoint processes the registration data, creating a new user record in Firebase and our MySQL database.
* **/api/login:** While we have implemented this endpoint, it's mainly redundant due to the Android SDK handling user logins. However, it exists as a part of our backend architecture for potential future use or alternative login methods.
* **/api/home:** One of the most important endpoints, **/api/home**, is responsible for delivering real-time air quality data to the user's dashboard. When accessed, it retrieves the latest sensor measurements from the database and provides insights into the current air quality, indicating whether it's good, moderate, or poor.
* **/api/settings:** This endpoint allows users to view and modify their settings. Additionally, it enables users to adjust their Smart Indoor Air Quality Hub settings, like toggling alerts or changing measurement intervals.

## **4.PubNub**

In our project, we have implemented PubNub for real-time data communication, which is essential for transmitting sensor data and user settings:

* **Aerosense Channel (aerosense\_channel):** This channel is the backbone of our real-time data transmission. The Raspberry Pi, equipped with the BME680 sensor, sends the collected air quality data to this channel. Our server, constantly listening to this channel, receives the data, decrypts it, and stores it in the database for further processing and access via the **/api/home** endpoint.
* **User Settings Channel (user\_settings\_channel):** This channel facilitates the real-time update of user preferences. When a user modifies settings in the app, the changes are sent to this channel. Our server listens to this channel, updates the user's settings in the database, and ensures that these preferences are reflected in the operation of the Smart Indoor Air Quality Hub.

## **Security Implementation in the Aerosense Project**

**IoT Device Security:**

Our IoT device, primarily the Raspberry Pi equipped with a BME680 sensor, we implemented a software-level security by regularly updating the Raspberry Pi's operating system and software packages, ensuring protection against vulnerabilities.

**Access to Communication Channels:**

We utilized PubNub for real-time data communication between the IoT device and the server. PubNub inherently employs TLS (Transport Layer Security) protocols, ensuring the encryption of data in transit.

The unique PubNub publish and subscribe keys ensure that only our authorized devices and server can access the designated communication channels.

**Database Security:**

Our MySQL database, hosted on AWS, is protected within a secured environment. We have configured AWS security groups to allow access only from specific IP addresses and ports, effectively guarding against unauthorized entry.

Database access credentials are managed securely, and limited access rights, ensuring that only essential services and personnel can access the database.

**Webserver Security:**

The webserver, also hosted on AWS, operates under HTTPS, which encrypts data between the client application and the server, safeguarding against data interception and tampering.

**Data in Transit Security:**

For data transmission, we use HTTPS and TLS protocols to encrypt data sent from the IoT device to the server and vice versa, ensuring secure data in transit.

Our implementation of cipher encryption for PubNub messages adds an additional layer of security, ensuring that the sensor data remains confidential during transmission.

**Firebase Security:**

We incorporated Firebase for user authentication and management. Firebase Authentication provides secure handling of user credentials and sessions.

Firebase's security rules ensure that only authenticated and authorized users can access their respective data, thereby enhancing the overall security of our system.

## **CONCLUSION**

## **REFERENCES:**