**DUNDALK INSTITUTE OF TECHNOLOGY**

**A logo with a bird and text

Description automatically generated**

**Beta Release Technical Documentation On**

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

Project Carried Out

**By**

**D00251785 PATRICK ORJIEH**

**D00244618 ROBERT NUGENT**

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 Beta release of the Aerosense project shows an advancement from our Alpha prototype, showing good progress in the Smart Indoor Air Quality Hub's development. This release includes the successful integration of an additional USB-to-UART converter, enhancing the PMS7003 sensor's functionality within the system. Also, the inclusion of the L76K GPS (new GPS sensor) module.

A significant difference between the alpha release and beta release in this phase is the transition to the PubNub Access Manager, making our system's security framework stronger. This upgrade ensures a more controlled communication environment between the IoT devices and our cloud infrastructure the AWS server.

With these enhancements, the Beta release strengthens the core attributes of Aerosense. It demonstrates our system's capability to support asthma patients effectively by providing vital, real-time insights into air quality dynamics. This Beta Release documentation will detail these improvements.

In conclusion, the Beta release signifies more about the final product, showing a position improvement from the Alpha release and the effect of the feedback received from the lecturers. It is a good step to our ongoing project to enhancing indoor air quality management, especially for individuals with severe asthma.

Table of Contents

[**GLOSSARY** 3](#_Toc156184160)

[**LIST OF TABLES AND DIAGRAMS** 6](#_Toc156184161)

[**List Of Tables** 6](#_Toc156184162)

[**List Of Diagrams** 6](#_Toc156184163)

[**1. INTRODUCTION** 7](#_Toc156184164)

[**2. SYSTEM ARCHITECTURE** 7](#_Toc156184165)

[**Overview** 8](#_Toc156184166)

[**IoT Elements** 8](#_Toc156184167)

[**Data Communication via PubNub** 9](#_Toc156184168)

[**Cloud Server and Database** 9](#_Toc156184169)

[**Firebase Authentication** 9](#_Toc156184170)

[**Mobile Application** 10](#_Toc156184171)

[**Community Data Sharing** 10](#_Toc156184172)

[**Conclusion** 10](#_Toc156184173)

[**3. Beta Prototype Documentation** 10](#_Toc156184174)

[**Hardware** 10](#_Toc156184175)

[**WebServer** 11](#_Toc156184176)

[**UI** 11](#_Toc156184177)

[**Security** 14](#_Toc156184178)

[**Database** 15](#_Toc156184179)

[**Data Processing** 16](#_Toc156184180)

[**4. AWS Custom Domain** 16](#_Toc156184181)

## **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 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.
* **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.
* **L76K GPS Module:** A new GPS sensor module implemented in the Aerosense system for enhanced geolocation capabilities.
* **PubNub Access Manager:** A security feature in PubNub that provides fine-grained access control to channels for improved security in real-time communication.
* **Beta Prototype:** The subsequent version of the Aerosense product after the alpha phase, which includes new features, improvements, and integrations based on
* **User Feedback Integration:** The process of incorporating suggestions and critiques from users into the development of the Aerosense system to enhance its functionality and user experience.
* **Enhanced User Interface:** Improvements made to the system's interface to make it more intuitive and user-friendly, often as a result of user feedback.
* **Firmware Updates:** Software updates for the embedded system within the Aerosense device, which can improve functionality or fix issues.

## **LIST OF TABLES AND DIAGRAMS**

### **List Of Tables**

### **List Of Diagrams**

Diagram 1: System Architecture Diagram of Aerosense Application (Revised for Beta Release) ……………………………………………………………………................................... 8

Diagram 2: Diagrams of the Mobile Application pages.........................................................12

Diagram 3: Diagram Of The Database Model…………………………………………………..15

Diagram 4: Diagram of the Landing page of the Aerosense Website………………………...16

Diagram 5: Diagrams of the Login Page for the Admin………………………………………...17

Diagram 6: Diagram Of Admin Dashboard………………………………………………………17

## **1. INTRODUCTION**

The Beta release of the Aerosense project marks a significant step forward from our initial Alpha prototype, showing the improved parts in monitoring indoor air quality, important for people with asthma. In this release, we've integrated new hardware components and made good improvements, and for the universal design principles caters to innovation and user-centric design. This document will go through the advanced features and architectural enhancements, telling on the integration of real-time data acquisition and user interaction. With a focus on accessibility and functionality.

## **2. SYSTEM ARCHITECTURE**

A diagram of a cloud computing system

Description automatically generated

Diagram 1: System Architecture Diagram of Aerosense Application (Revised for Beta Release)

### **Overview**

The Beta release of Aerosense maintains the foundational system architecture established in the Alpha release, with just little changes made to the PubNub communication protocol and GPS module integration for functionality and improved accuracy.

### **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. **L76K GPS Module**: 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. We have advanced our data communication setup by implementing PubNub's Access Manager. This makes the security protocols better, ensuring admin control over data streams:

To secure this communication:

* **PubNub Access Manager:** Replacing simple access keys, the Access Manager introduces a permission layer, safeguarding channel communication.
* **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).

## **3. Beta Prototype Documentation**

### **Hardware**

In the Beta release of Aerosense, we were to perfect our hardware components, crucial for air quality monitoring. Our PMS7003 sensor is now fully integrated, with the additional USB-to-UART converter, enhancing its capabilities to accurately measure particulate matter sizes PM1, PM2.5, and PM10. These particles are very important when looking at particles that affect asthma sufferers and can now be tracked with greater precision.

We've also introduced the L76K GPS Module, that ensures our device knows its exact location at all times. The Raspberry Pi is programmed to periodically read sensor data, preparing it for transfer to our web server.

### **WebServer**

The backbone of our Beta Prototype is a robust web server developed using the Flask framework. This server is a link between our IoT hardware and the client application, facilitating seamless interactions.

Webserver

The Flask Web Server Hosted on the AWS EC2 Instance does the following:

* It securely receives encrypted sensor data from our IoT devices.
* Decrypts the information for validation and storage.
* Efficiently serves this data to our user interface on-demand.
* Seamlessly manages user authentication and sessions.

Hosted Solution

The server on Amazon Web Services (AWS), offering a reliable and scalable cloud environment on ‘https://www.aerosense.life/’

### **UI**

There have been a lot of new aspects added to the UI since last time as well as some changes implemented which were suggested by users from the user tests. One of the main parts of the ui that has been updated is the capability to display and update all real data from the database. With the successful implementation of all 3 sensors for this CA, we have made sure that all data from these is being displayed on the front-end. We have also ensured that all other relevant data from the database is being shown in the UI such as asthma profile data and the users’ settings. Essentially, all of the hardcoded values that were present in the alpha prototype have been removed and replaced with actual values from the database. In terms of additions to the UI, new pages have been added including the history page, the educational page, and the notifications page.

**New Pages**

A graph showing the average air quality

Description automatically generated A screenshot of a cell phone

Description automatically generated A screenshot of a phone

Description automatically generated

Diagram 2: Diagrams of the Mobile Application pages

**History Page:**

The history page provides the user with insights into their weekly air quality measurements and how bad or good they are each day via a graph and some averages of the measurements. The graph is automatically generated from the readings coming in from the backend from each day. Originally this page only showed air quality statistics for the current week but after user testing, it was suggested that there should be a way for the user to view previous weeks so we added a drop down option for choosing the statistics you want to see (this week or last week).

**Educational Page:**This page was made as a result of the feedback gathered during user testing. A few of the people who participated in the user testing mentioned that we should add some way of adding more meaning to the measurements or making them easier to understand/interpret so it was decided that this page would be created. The educational page simply provides the user with in-depth information about the meaning of each air quality measurement if they are interested in learning more about them.

**Notifications Page:**

The notifications page reads all of the notification data from the database and displays all of them to the user. Each notification includes the time it was created as well as the title and description of the notification which can be reviewed by the user. This page will be handy for the user in case they miss a mobile notification or want to read more about the recommendations being provided by the notification.

**All user testing feedback**

**Session 1:**

**Age:** 21

**Personal experience with Asthma:** None

**Feedback:**

* The login and register pages are simple and clear.
* Could be clearer for the user if we showed the actual parameters for the overall air quality percentage somewhere in the app (e.g. 60% - 100% = good, 40 – 50% = Medium etc.)
* Provide baselines/averages for each measurement so the user can know how much worse or better the measurement is compared to how it should be.
* The extra measurements in the “View More” section of the home page have no comparisons/ways to tell if they are good or bad.
* Add a screen where you can see all notifications received about air quality so that you can look back on recommendations given by them.
* Find a way to show the time for each bad air quality area being marked on the map on the locations page so that the user of the app can know how long ago it was detected as it may be cleared a few hours after that.

**Session 2:**

**Age:** 23

**Personal experience with asthma:** Father has asthma, so they have a decent amount of experience with it.

**Feedback:**

* The name and logo suit the app well and the logo looks nice.
* The colours on the home page are a good way of showing air quality.
* The home page is laid out in a nice and clear way.
* We should add some sort of status signifier for numbers at the bottom of the home page to signify whether they are good or bad.
* For the user asthma page, we should add in a wider variety of options for main trigger or allow the user to enter one themselves since there are a lot more things that can trigger asthma other than what we have there. For example, their dad’s asthma can often grow worse when he is around pets, and we don’t have an option there for that.
* Would be nice if we added an option to the history page to view previous weeks data so that the person can look back on it.
* Educational page is very useful and informative.

### **Security**

For the Beta release, we've strengthened our security measures to safeguard the Aerosense ecosystem comprehensively.

Enhanced Security Protocols

We've implemented several additional security features:

• Adoption of the PubNub Access Manager to tightly control device-server communication.

• Scheduled software updates for the IoT device firmware through cron jobs, ensuring our system is resilient against vulnerabilities.

• Strengthened encryption during data transmission, done by the PubNub Access Manager, to maintain the confidentiality and integrity of sensitive data.

• A secure authentication process using Firebase, which incorporates the OAuth 2.0 framework for more reliable and secure user validation.

• Enforced HTTPS protocols across all communication channels to guarantee the integrity and confidentiality of data in transit.

To better understand the Update for the Device - Raspberry the Script to Update the Software and the Cron job script can be found in the \air-quality-monitor\backend\_hub\sensor\_pi directory

#!/usr/bin/env python3

import subprocess

def update\_system\_packages():

print("Updating system packages...")

subprocess.run(['sudo', 'apt-get', 'update'], check=True)

subprocess.run(['sudo', 'apt-get', 'upgrade', '-y'], check=True)

def update\_pip\_packages():

print("Updating pip packages...")

subprocess.run(['pip3', 'install', '--upgrade', 'pip'], check=True)

subprocess.run(['pip3', 'list', '--outdated'], check=True)

if \_\_name\_\_ == "\_\_main\_\_":

update\_system\_packages()

update\_pip\_packages()

To access the crontab

- crontab -e

then type in the python and the location of your script

- \* \* \* \* \* python3 /home/pi/IOT/Aerosense/sensors.py

for the update scripts this will run will run daily at midnight and will redirect both standard output and standard error to a log file for review.

0 0 \* \* \* /usr/bin/python3 /home/pi/IOT/Aerosense/update\_script.py >> /home/pi/cron.log 2>&1

### **Database**

A diagram of a computer program

Description automatically generated with medium confidence

Diagram 3: Diagram Of The Database Model

The database schema for the Beta release has been redesigned to create space to the storage needs of our enhanced feature set. We've introduced read\_token and write\_token fields in the User and Hub tables to integrate with the PubNub Access Manager, securing our data channels effectively. Relationships between tables are optimized for query efficiency with the implementation of indexes, enhancing the responsiveness of our system.  
  
CREATE INDEX idx\_user\_email ON User(email);

CREATE INDEX idx\_hub\_user ON Hub(userID);

CREATE INDEX idx\_location\_hub ON Location(hubID);

CREATE INDEX idx\_measurement\_hub ON AirQualityMeasurement(hubID);

CREATE INDEX idx\_notification\_user ON Notification(userID);

CREATE INDEX idx\_asthma\_user ON AsthmaProfile(userID);

CREATE INDEX idx\_setting\_user ON UserSetting(userID);

### **Data Processing**

The data processing algorithms have been chnaged to offer more precise air quality metrics. We calculate an overall air quality score based on key environmental parameters from the Irish Environmental Protection Agency, using weighted averages to ensure a measure of air quality. Our backend efficiently handles weekly data aggregation, providing users with a comprehensive view of air quality trends.

## **4. AWS Custom Domain**

<https://www.aerosense.life/>

This is a summary of each endpoint for the Aerosense AWS server, to be included in the Beta Release documentation:

1. **Homepage Endpoint** (**/**):

A black and white picture of a machine

Description automatically generated

Diagram 4: Diagram of the Landing page of the Aerosense Website

* + Serves as the landing page for Aerosense, providing the first point of interaction with users through a welcoming interface to get the application. The QR code doesn’t work to the play store just a design

1. **Admin Login Endpoint** (**/admin\_login**):

A screenshot of a computer screen

Description automatically generated

Diagram 5: Diagrams of the Login Page for the Admin

* + Facilitates administrative access to the Aerosense dashboard, allowing credential verification and redirecting to the main admin dashboard upon successful login.

1. **Admin Dashboard Endpoint** (**/admin\_dashboard**):

A screenshot of a computer

Description automatically generatedDiagram 6: Diagram Of Admin Dashboard

* + Central hub for administrators to manage user and hub access tokens, with a comprehensive view of all registered users and devices.

1. \*\*Token Request

Endpoint (**/api/request\_token**):

* Manages token issuing for Raspberry Pi devices. When a Pi needs to publish data to PubNub, it requests a token based on its model number. This endpoint verifies the hub existence and provides the appropriate token.

1. **User Access Update Endpoint** (**/update\_access\_user**):
   * Administers user access levels, allowing the admin to update individual user permissions based on their interaction with the admin dashboard.
2. **Hub Access Update Endpoint** (**/update\_access\_hub**):
   * Adjusts the access level for each hub, updating the permissions for read or write access, and handles token issuance or revocation accordingly.
3. **Admin Logout Endpoint** (**/admin\_logout**):
   * Safely logs out an administrator from the dashboard, ensuring that session data is cleared for security purposes.
4. **User Registration Endpoint** (**/api/register**):
   * Facilitates new user registration, integrating with Firebase for authentication and creating new entries for user and hub information in the database.
5. **Asthma Profile Fetch and Update Endpoints** (**/api/asthma-profile**):
   * Retrieves and updates individual asthma profile settings for users, personalizing their experience based on their specific triggers and conditions.
6. **Notifications Retrieval Endpoint** (**/api/get\_notifications**):
   * Collects and provides notifications for users, informing them of relevant air quality updates and alerts based on their settings.
7. **Hub Location Data Endpoint** (**/api/locations**):
   * Gathers and presents the location data for all hubs, enabling tracking and analysis of air quality across different geographical points.
8. **User Settings Fetch and Update Endpoints** (**/api/settings**):
   * Retrieves and updates user-specific settings, allowing customization of notification frequency and sensory alerts such as vibration and sound.
9. **Air Quality Historical Data Endpoint** (**/api/history**):
   * Offers a historical view of air quality data, giving users the ability to review past readings and observe trends over selectable time frames.
10. **Real-time Air Quality Data Endpoint** (**/api/home**):
    * Provides the latest air quality data from a user’s specific hub, ensuring they have up-to-date information for immediate decision-making.