Design and Implementation of an IoT-Based, Fuzzy Logic-Integrated Indoor Air Quality Monitoring Dashboard

Andreas Josef C. Diaz
Department of Electronics, Computer, and Communications Engineering
Ateneo de Manila University
Quezon City, Philippines
andreas.diaz@obf.ateneo.edu

Marc Jefferson B. Obeles
Department of Electronics, Computer, and Communications Engineering
Ateneo de Manila University
Quezon City, Philippines
marc.obeles@obf.ateneo.edu

Justin Gabriel M. Sy
Department of Electronics, Computer, and Communications Engineering
Ateneo de Manila University
Quezon City, Philippines
gabriel.sy@obf.ateneo.edu

***Abstract*—With the objective of monitoring and inhibiting the spread of COVID-19 within educational institutions, the study aims to design and implement an indoor air quality monitoring dashboard that utilizes different components that measure air quality indicators such as carbon dioxide, carbon monoxide, particulate matter, temperature, and humidity. Fuzzy inference systems are utilized to arrive at a finite set of fuzzy rules that guide the system in estimating a room's indoor air quality index and determining an appropriate polling interval for the sensors. These sensor data are uploaded to Google Sheets using Google Apps Script, sent to a cloud Python script for indoor air quality index fuzzy logic calculation, and visualized using Tableau.**

***Index Terms*—Indoor air quality (IAQ), environment indoor air quality index (EIAQI), Google Sheets, Google Apps Script, Hypertext Transfer Protocol (HTTP), web application, fuzzy logic, temperature-humidity index, thermal comfort index, Python, polling interval, Tableau, dashboard**

I. Introduction

A. Background of the Study

Ever since the onset of the COVID-19 pandemic, there has been a continuous struggle for educational institutions to conduct onsite activities, which have been proven necessary for students to gain technical skills related to their field of study. However, despite improvements in online learning delivery, there remain skills that students can only learn when using different modes of interaction, such as laboratory use, face-to-face interactions, and in-person assessments, which are complex and infeasible to replicate in an online environment. Thus, educational institutions such as Ateneo de Manila University have begun rolling out onsite pilot classes during the second half of the year 2022. With this in mind, health protocols must be closely monitored to ensure the safety of all individuals who frequent the campus.

Based on the published work of Sassi and Fourati, pollution particles found in the air, expressly referred to as particulate matter ($PM_{2.5}$ and PM_{10}), are "transport vectors" for the viral transmission of numerous diseases, including COVID-19 [6]. These pollution particles are microscopic solid and liquid particles that, when inhaled, may cause health complications. For example, the study showed that a rise of 1 g/m^3 in the concentration of $PM_{2.5}$ is linked to an 8% increase in COVID-19 mortalities from a general purview. Thus, high levels of $PM_{2.5}$ and PM_{10} are evidently hazardous and must be considered, especially in pollution-dense areas. An example cited by Sassi and Fourati is the case of individuals who are either symptomatic or asymptomatic and may proliferate the virus nasally and orally through the carbon dioxide (CO_2) particles expelled by the body within a given radius. Furthermore, the viability of aerosols in spreading viruses have shown to last approximately 3 hours under indoor conditions.

Another interesting phenomenon to consider is the reinforcing correlation between temperature and humidity with respect to the spread of the COVID-19 disease across different countries. For instance, in a study conducted by Olinto et al., COVID-19 cases and deaths were correlated with respect to the temperature and relative humidity in Brazil using climate indicator data from the first 59 days since the onset of the disease in the country [1]. From their analysis, higher temperatures and lower relative humidity amplitudes resulted in a lower COVID-19 mortality rate, showing an inverse relationship between temperature and disease incidence. More specifically, temperatures below 25.8°C resulted in higher COVID-19 cases.

Using a similar regression model, Ganegoda et al. conducted a study on the interrelationship between COVID-19 incidence and temperature and humidity in Germany [2]. From their findings, the virus survives best at temperatures of 20°C and relative humidity of 50%, lasting at least 28 days. Furthermore, they arrived at the conclusion that temperature may predict cases only during hot and cold seasons (summer and winter), while humidity can only do so during transitional seasons (autumn and spring). However, a correlation between these quantities does not imply causation and may need further study. Nonetheless, the simple random-effects model utilized in the analysis of Ganegoda et al. shows that there is a strong reason to use these meteorological indicators, including $PM_{2.5}$ and PM_{10}, as predictive scales in guiding educational

institutions in battling the virus.

B. Statement of the Problem

Given the microscopic particles that prove to be hazardous with regard to the spread of the COVID-19 disease, the implementation of an Internet of Things (IoT)-based indoor air quality monitoring system is one of the most effective and cost-efficient ways to address the safety concerns encountered by educational institutions when conducting onsite activities, as classes in Ateneo de Manila University are mostly held in closed, air-conditioned classrooms. This study is implemented in specific classrooms of the Faura Hall through the installation of indoor air quality monitoring (IAQM) devices that measure different air quality indicators such as CO_2, carbon monoxide (CO), $PM_{2.5}$, PM_{10}, temperature, and humidity, and consequently displaying these information in a dashboard.

From a hardware perspective, the implementation uses different components that measure air quality indicators such as the Adafruit SGP30 to measure CO_2, MQ-7 to measure CO, PMS7003 dust sensor to measure fine particulate matter ($PM_{2.5}$ and PM_{10}), DHT22 to measure temperature and humidity, photoresistor to measure room light intensity, KY-016 RGB light-emitting diode (LED) to display the environment indoor air quality index (EIAQI) status, and the ESP-32 being the central controller of the system to activate the sensors and directly process the data through its IoT capabilities.

On the other hand, from a software perspective, IAQ sensor data are uploaded from the ESP-32 to Google Sheets using an Hypertext Transfer Protocol (HTTP) request over Google Apps Script, processed using fuzzy logic programmed using MATLAB's Fuzzy Logic Toolbox and translated for use in Python's scikit-fuzzy package, then hosted in a virtual machine (VM), and visualized using a Tableau dashboard hosted in Tableau Public. The sensor data reflected in Tableau displays the analog readings from the sensors for direct access by building administrators, while people inside the room will be notified using the RGB LED. Additionally, the user interface of the dashboard will provide general and specific information pertaining to the classrooms' IAQ to ultimately aid in making data-driven decisions in containing the spread of the COVID-19 disease.

C. Objectives of the Study

In order to successfully implement the study, the researchers are to design and develop an IoT-based, fuzzy logic-integrated indoor air quality monitoring dashboard. Specifically, the objectives of the study are:

1) To build and synergize the microcontroller and sensors into a product designed for indoor air quality monitoring;
2) To create a fully functional and comprehensive monitoring dashboard that holds historical data and patterns to enable the administrators of Ateneo de Manila University to conduct further analyses;
3) To utilize fuzzy inference systems to estimate an environment indoor air quality index and allow a dynamic polling interval for the sensors; and
4) To successfully notify people within a room of the indoor air quality condition using an RGB light-emitting diode while informing building administrators through the dashboard.

II. Review of Related Literature

Our living situation has changed due to the onset of the COVID-19 pandemic, and we have been forced to monitor our surroundings to diminish the risk of contracting the virus. One such way is to monitor air quality, which studies intensively dived into during the pandemic. The following literature review shows the development of monitoring air quality systems during the period of COVID-19 pandemic and also looks at air quality monitoring systems that utilize fuzzy logic systems.

A. Development of Air Quality Monitoring Systems

During the COVID pandemic, some researchers saw the need to monitor air quality. De Medeiros and Girão observed that commercial solutions for air quality monitoring, which environmental agencies in Brazil usually utilize, have high implementation, maintenance, and operation costs [3]. The authors then proposed an IoT-based platform to measure air quality. The proposed system operated PMSA003, MICS-6814, MQ-131 sensors, which measure ammonia, carbon monoxide, nitrogen dioxide, ozone, and particulate matter ($PM_{2.5}$ and PM_{10}). The sensors were connected to an ESP-WROOM-32 microcontroller, sending the data using an MQTT protocol to a server. Moreover, the authors proposed using Zabbix and Grafana's tools for monitoring and visualizing the stored data, which would then be used in a web service for the public to access. The authors also plan to give users the option to receive notifications and alerts in case the air quality of the area is hazardous. The proposed system was estimated to cost $75, only accounting for the components.

The early stages of the COVID-19 pandemic focused on outdoor air quality. However, after a year, the focus on monitoring air quality gradually shifted to indoor. Loosová and Hernych observed that organizations have been focusing on improving the indoor air quality of rooms after the onset of the COVID-19 pandemic [4]. Organizations, such as the Centers for Disease Control and Prevention (CDC), and the World Health Organization (WHO), recommend inspecting ventilation systems and improving them to lessen the spread of the virus. The authors found that CO_2 levels provide relevant information on room ventilation, which can help further monitor indoor air quality. Specifically, a CO_2 concentration of 800 ppm or below indicates good ventilation and air quality. A problem that Loosová and Hernych encountered is that the concentration of CO_2 varies within a room, and thus recommend placing sensors in the central exhaust air outlet of the room.

Studies that create systems to monitor indoor air quality increased since then. Most of these studies incorporate the

IoT, as it is necessary to visualize indoor air quality to their constituents. Budiharto et al. developed a portable device that measures body temperature and air quality [5]. This device screens people for the COVID-19 disease and checks the ventilation of rooms and air filtration to prevent the spread of COVID-19 inside. The device used an MLX90614 infrared digital temperature sensor to measure body temperature, while SGP30 was used to monitor air quality. These sensors were connected to an Arduino Uno in an I2C communication bus. The measured data were then displayed in a liquid crystal display. Tests show that the device has an accuracy of 95%, failing to get the body temperature when the person is wearing a mask. Sassi and Fourati presented an IoT system to monitor the air quality of rooms through LoRaWAN and predict its value using Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) [6]. The system uses an Arduino Uno integrated with a LoRA shield and uses sensors capable of measuring CO, humidity, temperature, $PM_{2.5}$, and PM_{10}. Moreover, the data is sent to a MySQL database through a 4G LTE network. The system also uses AR visualization for users to analyze. Data visualization was done through the Vuforia engine and utilized a floor view of the building to visualize the indoor air quality in each room. In addition, Sassi and Fourati used a color scale that describes the safety level of rooms based on the air quality index shown in Fig. 1. Longo et al. observed that current implementations of air quality monitoring smart services do not calculate the risk of COVID-19 propagation [7]. On the other hand, existing tools that calculate and model the risk of propagation lack the smart service feature. In response to this gap, the researchers then created a smart service that monitors indoor air quality and alerts people when current conditions of the room are susceptible to proliferating the COVID-19 virus. The smart service monitored CO_2 levels, humidity, particulate matter ($PM_{2.5}$ and PM_{10}), temperature, and total volatile organic compounds (TVOC). The collected real-time data from these sensors then go through a decision tree algorithm to calculate the transmission risk according to the Salubrity Scale. This service benefits users, particularly building managers and individuals using rooms in the building, as the system would improve their decisions regarding room air quality. Vaheed et al. built an IoT-assisted indoor air quality monitoring system using a ZPHS01B multi-channel gas sensor that measures CO_2, CO, nitrogen dioxide (NO_2), ozone (O_3), particulate matter ($PM_{2.5}$ and PM_{10}), and volatile organic compounds (VOC) concentrations. [8]. This sensor is connected to a Raspberry Pi 4, which can send data to Google Firebase over the internet. Moreover, the researchers visualized the stored data by creating a Python web portal. The system calculates the indoor air quality index (IAQI) using the measured factors. However, the researchers recommend using different methodologies and utilizing better sensors to more accurately calculate the air quality index (AQI). Moreover, Vaheed et al. also recommend adjusting data update times when needed to benefit the public. Furthermore, they also seek to predict future values of AQI through machine learning.

Fig. 1: Air Quality Index Scale from Sassi and Fourati [6]

B. Utilization of Fuzzy Inference Systems

Fuzzy inference systems have been explored in air quality monitoring systems before the COVID-19 pandemic started, particularly in estimating the air quality index. Aggarwal et al. simulated a fuzzy inference system that calculated the air quality index value using particulate matter ($PM_{2.5}$ and PM_{10}) as input factors [9]. The fuzzy inference system utilized the two inputs with six linguistic variables to generate 36 rules. Tests and simulations were done in MATLAB for five days. The authors showed that the air quality index (AQI) determined by the fuzzy inference system is similar to the AQI value calculated using a traditional linear interpolation method. Teologo et al. simulated a system to classify the AQI using two inputs, namely CO and NO_2 [10]. A Mamdani fuzzy inference system was utilized to calculate and classify the resulting air quality index. The fuzzy inference system generated 36 rules using six categories for each input. Overall, the model was considered reliable. However, the authors explained that increasing the number of inputs in a fuzzy logic system would be complex in producing various rules but may overall improve the classification model. Therefore, they recommend using rule reduction and optimization techniques to allow another input, then pairing the resulting fuzzy logic model with an automatic warning device.

Moreover, the use of fuzzy inference systems is not only limited to estimating the air quality index as studies likewise used it for control systems that react to the indoor air quality parameters. Pradityo and Surantha implemented an indoor air quality monitoring and control system using an MQ-135 sensor and a Sharp GP2Y1010AU0F dust sensor connected to an Arduino Uno [11]. It was then interfaced with a Raspberry Pi 3, which has Wi-Fi connection capabilities. The system gets the CO_2 level and PM_{10} density and sends it to a Amazon Web Service cloud, which is then displayed in a dashboard. Moreover, the Raspberry Pi 3 is utilized to process the fuzzy logic rules with the CO_2 level and PM_{10} density as input. The rules then dictate the length of time the room exhaust fan would turn on to effectively improve indoor air quality. The implementation resulted in nine fuzzy rules and showed that the system could better control the room's air quality index to a safer degree than conventional systems. They also claimed to save energy, although Pradityo and Surantha did not perform tests to validate their claim. Bushnag created an indoor air quality monitoring (IAQM) system using an MQ-135 air quality sensor and a DHT11 sensor [12]. The output of these sensors, mainly air quality, humidity, and temperature, were used as inputs for a fuzzy logic controller. The fuzzy logic controller generated 50 rules that would dictate the behavior of the indoor ventilation system, specifically by adjusting fan speed. The created air quality monitoring and

control system reduced power consumption. For improvement, Bushnag recommends utilizing the same type of sensor in different locations to improve monitoring accuracy.

After the onset of the COVID-19 pandemic, a study looked into utilizing fuzzy inference systems to accurately estimate indoor air quality index (IAQI) using cheap sensors. Zareb et al. explored utilizing a fuzzy inference system to estimate indoor air quality [13]. The researchers created an IoT device that uses low-cost sensors, mainly two MQ-7 sensors that measure carbon monoxide (CO) levels, a Shenyi PPD42N particulate matter sensor, and a DHT22 temperature and humidity sensor. The researchers calibrated the MQ-7 sensors by obtaining their limits and utilizing the range in creating the membership functions for the fuzzy inference system. The fuzzy inference system uses the two MQ-7 sensors as input, and generates an AQI estimation. Including the fuzzy inference system proved helpful, making the AQI estimation accurate despite using cheap sensors. The researchers recommend estimating future air quality values using machine learning and improving the fuzzy inference system by upgrading to its type-2 counterpart.

In Schlatter's discussion, it was explored that the human body has a reaction when combining heat and humidity, making the temperature-humidity index (THI) a viable measurement that can observe climate within a particular area. However, heat and moisture are not the only factors that affect heat stress. At a THI of less than 70, inactive people should be comfortable. At approximately 75, half of the population would find the environment uncomfortable. At 79, most would be uncomfortable. Higher THI values are known to create acute discomfort for humans, which can result in drops in work efficiency and even raise health emergencies [14]. Furthermore, Chan et. al recognizes that most air quality indices have a direct correlation with $PM_{2.5}$ and PM_{10} alongside ozone levels, which are determined through data collection and weather forecasting information which are stored in data management centers [15].

Thus, Dionova et. al poses a unique solution that calculates an environment indoor air quality index (EIAQI) using a fuzzy inference system, utilizing multiple sensors such as MQ-135, MQ-7, MQ-9, and DHT11 to measure CO, CO_2, NO_2, O_3, $PM_{2.5}$, temperature, and humidity. Clustering techniques were utilized to develop four cluster indices, which compute for two indoor air quality indices (IAQI) and two thermal comfort indices (TCI) determined by the fuzzy logic controller, holding 64 fuzzy rules to account for eight components. Eight input values would defuzzify into four output variables that determine the final environment indoor air quality index (EIAQI) by intermediately adding similar indices together, then adding into a final index that ranges from 0 to 6, with 0 being the worst and 6 being the best. Given the researchers' implementation, there would be recommendations per the EIAQI category that involves the fan, inlet-outlet exhaust, and the LED buzzer to improve indoor air quality proactively. [16].

Moving forward from the COVID-19 pandemic, monitoring indoor air quality is a step toward alleviating airborne diseases in the future. The reviewed literature shows that monitoring indoor air quality is beneficial to people occupying rooms as they could be informed of the potential risks involved with poor air quality. Moreover, the literature show promise in improving the estimation of indoor air quality by utilizing fuzzy inference systems.

III. Methodology

The proposed system aims to capture, store, estimate, and visualize room air quality. The system consists of four main components shown in Fig. 2 to achieve this goal. The first component are the indoor air quality monitoring (IAQM) devices, which are deployed in target rooms, particularly the classrooms inside Faura Hall. These devices measure indoor air quality parameters such as temperature, humidity, carbon dioxide (CO_2), carbon monoxide (CO), and particulate matter ($PM_{2.5}$ and PM_{10}). In this implementation, three IAQM devices were constructed. The second component of the system is Google Sheets, which acts as a database for the sensor readings collected by the monitoring devices. The third component is the fuzzy logic controller, which estimates the environment indoor air quality index (EIAQI) based on current sensor readings and sends this information back to both Google Sheets and the IAQM devices. Finally, the Tableau dashboard component of the system provides a visual representation of the indoor air quality data stored in Google Sheets, which is accessible to both the public and administrators for viewing. The following sections provide a detailed discussion of each system component, including their functions, interactions, and implementation considerations.

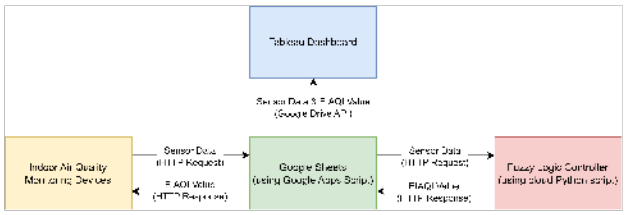

Fig. 2: IAQM System Block Diagram

A. Indoor Air Quality Monitoring Device Hardware

The IAQM devices, which form the first component of the proposed system, consist of a microcontroller, six sensors, and one output device. These devices are designed to capture various air quality parameters, including temperature, humidity, CO_2, CO, $PM_{2.5}$, and PM_{10}, as well as room information such as light level. The captured data is then sent to Google Sheets through an HTTP request. Fig. 3 presents the utilized components and wiring to outline their connection with the microcontroller. Moreover, these components are placed in an enclosure with a dimension of 97 mm by 91 mm by 65 mm. Fig. 4 show the placement of the hardware components, while Fig. 5 and 6 show alternative views of the box. A particular design choice when creating the box was that the DHT22 sensor was placed between the SGP30 and MQ-135 sensors, as the sensors require temperature and humidity to correct their readings.

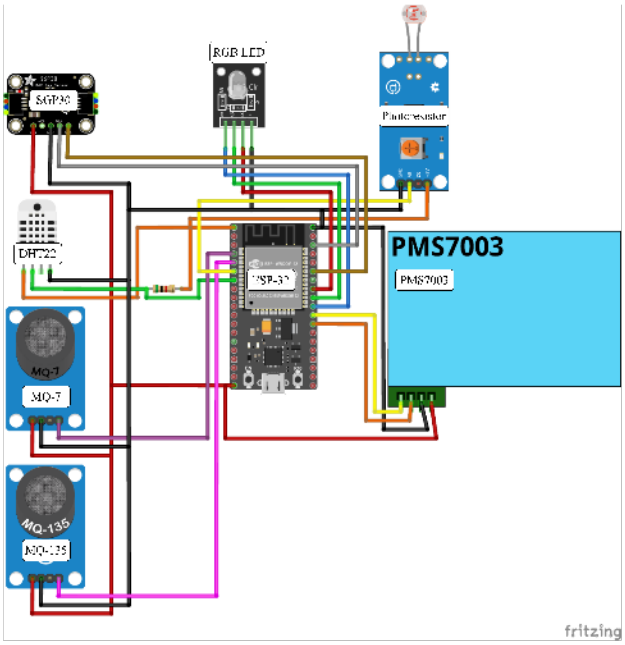

Fig. 3: IAQM Hardware Layout and Connections

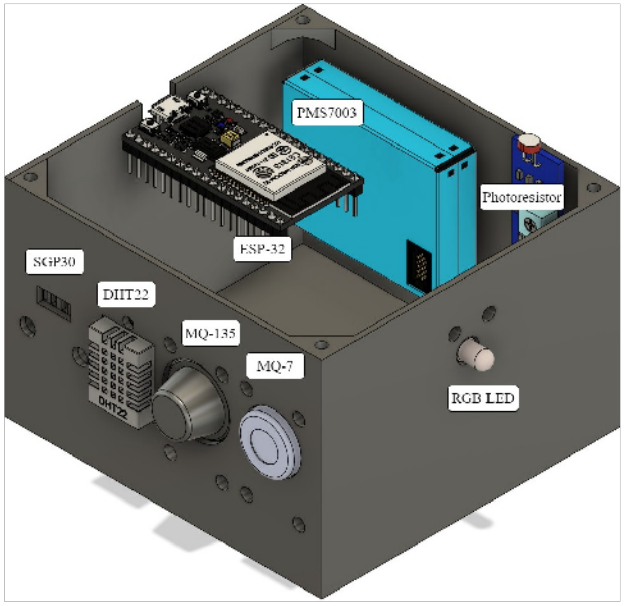

Fig. 4: IAQM Device Component Placements

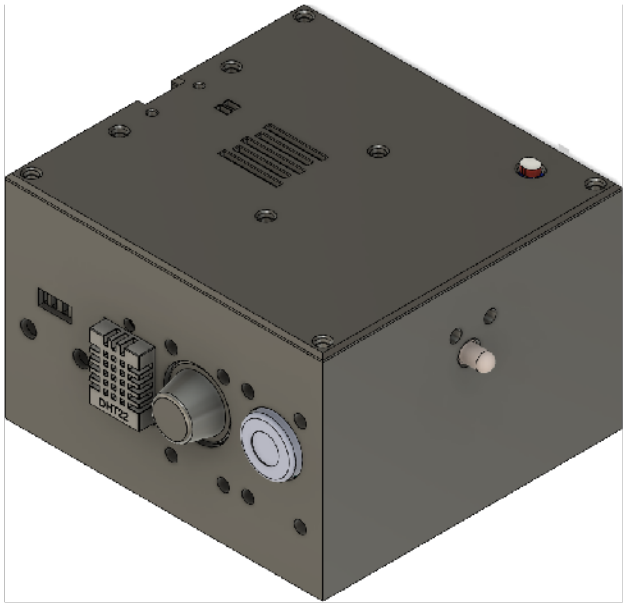

Fig. 5: IAQM Device Box View 1

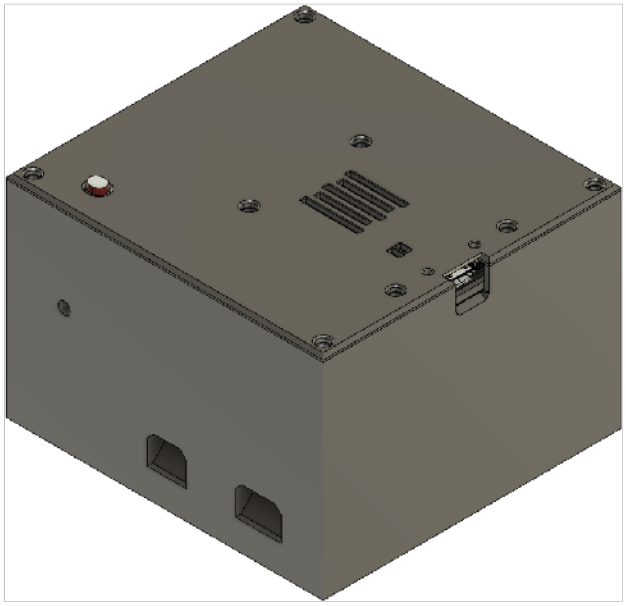

Fig. 6: IAQM Device Box View 2

The devices are powered by a 5 V, 1.5 A power adapter, as the expected current draw from the system is approximately 700 mA.

Additionally, three devices have been developed and deployed in specific rooms. These devices are named after the rooms in which they are deployed: F-323 A, F-323 B, and F-324. The first two devices are situated in the same room, as it is spacious, and comparing the data from these two devices in the same location would be beneficial in observing concentration distribution of IAQ indicators. The last device is located in a typical-sized room.

Hardware Components:

1) Microcontroller: The microcontroller used is the ESP-32 NodeMCU 32-S. The ESP-32 is a 2.4 GHz Wi-Fi-and-Bluetooth combo chip with 3 available UARTs, 34 programmable GPIOs, I2C channels, PWM pins, and 12-bit ADC pins, making it well-suited for capturing and transmitting indoor air quality data. The ESP-32's capabilities for IoT applications, specifically IoT data logging, make it an ideal choice for this system. Moreover, the ESP-32 chip can be powered using a 3.3 V supply. Still, other microcontroller implementations, such as the NodeMCU 32-S, can accept a 5 V DC supply, making it possible to power 5 V sensors and

modules while still powering the ESP-32 chip at 3.3 V. On the other hand, the ESP-32 can consume up to 240 mA of current while using Wi-Fi [17].

2) CO_2 Sensor: The SGP30 is a metal-oxide gas sensor that measures carbon dioxide (CO_2) levels. The SGP30 has a digital I2C interface, making it easy to integrate into the system. It can be powered with a 3.3 V or 5 V, allowing flexibility in the device's power source. The sensor can measure the carbon dioxide concentrations from 400 to 60,000 parts per million (ppm), with a typical accuracy of 15% within measured values. It is important to note that the measurement is only based on concentrations of hydrogen (H_2), providing a general idea of air quality trends [18].

3) Temperature and Humidity Sensor: The device also utilizes the DHT22, a low-cost sensor with accurate temperature and humidity readings. The DHT22 features a capacitive humidity sensor and a thermistor, capable of precise measurements. The sensor sends the temperature and humidity readings through a digital signal and can be powered using 3 V to 5 V. The DHT22 draws a maximum of 2.5 mA of current and can provide humidity readings from 0 to 100% relative humidity (RH) with an accuracy range of 2-5%. Additionally, the sensor can measure temperature readings from -50 to 80 degrees Celsius (°C) with an accuracy of ± 0.5°C [19].

4) CO Sensors: The MQ-7 sensor measures carbon monoxide (CO) levels in the air. It has a high sensitivity to CO and is particularly useful in detecting low concentrations. The sensor operates at a voltage of 5 V and uses a maximum current of 150 mA when connected to a breakout board [20].

On the other hand, the MQ-135 is another metal-oxide sensor that can detect multiple gasses, such as ammonia, carbon dioxide, and carbon monoxide. In this system, the sensor was configured to detect carbon monoxide. It draws roughly 135 mA of current when powered by a 5 V DC source. The MQ-135 has a carbon monoxide sensitivity range of 10-1000 ppm [21]. When obtaining the measure for CO, the average of MQ-7 and MQ-135 readings is used rather than individually accounting for the two sensors.

5) Dust Sensor: The PMS7003 is a dust sensor that measures $PM_{1.0}$, $PM_{2.5}$, and PM_{10} particulate matter. It utilizes a laser scattering principle to count the number of particles in the air and their concentration. The sensor operates on a 5 V direct current (DC) voltage, consuming approximately 100 mA of current. The PMS7003 has a measurement range of 0 to 500 $\mu g/m^3$ for $PM_{1.0}$, $PM_{2.5}$, and PM_{10}. The sensor communicates its readings through a UART digital interface [22].

6) Light Sensor: The light dependent resistor (LDR), or a photoresistor, is a passive component that adjusts its resistivity depending on the light it receives. It can operate with an input voltage of 3.3 V to 5 V DC, and its maximum output voltage is the same as its input voltage. This component was utilized in the IAQM system to detect if people are present in the rooms, which adjusts the polling interval of the sensors.

7) RGB LED: The KY-016 is an RGB light-emitting diode (LED) module that contains a 5 mm RGB LED. It has three inputs, mainly the red, green, and blue channels, accepting pulse-width modulated (PWM) signals and one pin that connects to the ground [23]. The module informs the current air quality situation inside the room by changing colors according to the specified range of EIAQI values.

B. Indoor Air Quality Monitoring Device Firmware

In utilizing the IAQM device hardware components, the firmware was created as an Arduino sketch and uploaded to the ESP-32 microcontroller. Fig. 7 shows the general flow of the firmware, with distinctions of the functions used in "void setup" and "void loop".

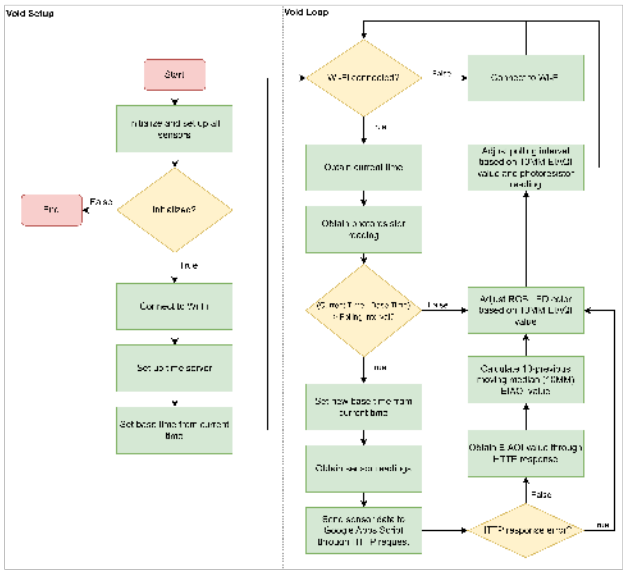

Fig. 7: IAQM Device Firmware Flowchart

Moreover, the firmware functions with the aid of Arduino libraries to interface with the sensors, including the ESP-32 itself. The following lists the libraries utilized for within firmware:

1) WiFi, a built-in ESP-32 Library
2) HTTPClient, another built-in ESP-32 Library
3) C Time, a built-in C Library
4) Adafruit DHT Sensor Library [24]
5) Adafruit SGP30 Sensor Library [25]
6) MQUnifiedSensor Library [26]
7) PMS Library [27]

C. Calibration Methods

Calibrations were performed to the six sensors to ensure that all sensors are reading correctly. Calibration methods for each sensor vary, and so are discussed in their respective subsections.

1) SGP30: The calibration for SGP30 is already available in its accompanying Arduino library. Adafruit states that calibration is possible by storing a baseline value using their algorithm [28]. However, the SGP30 sensor has to be running for over 12 hours to get a valid baseline value. Additionally, the sensor has to be exposed outdoors in order to obtain a

baseline from the global H_2 value. The obtained baseline value is only valid for seven days while the sensor is turned off.

Based on this, all SGP30 sensors were ensured to run continuously for 12 hours before running the baseline algorithm, which also ran continuously for approximately 4 hours. The values are sent to a Google Sheets file through an HTTP request and averaged to get the baseline value used in each sensor. Since the IAQM devices are expected to run continuously and the SGP30 sensors will not be turned off, there is no need to recalibrate the sensors further beyond the initialization process of the IAQM devices.

2) MQ-7 and MQ-135: The calibration process for both MQ-7 and MQ-135 sensors is similar to the process for the SGP30 as their respective libraries already have a function for calibration. Similarly, it outputs a value that can be used to set up each MQ sensor in the firmware. In this case, the library calculates and outputs the sensor's resistance value ($R0$) in the presence of clean air.

The library for MQ Sensors, MQUnifiedSensor library, also documented the ratio of the sensors in clean air. These values were derived from the data sheets of the MQ sensors. Specifically (1) and (2) show the ratio values used for calibration.

$$\text{Ratio in Clean Air}_{\text{MQ-7}} = 27.5 \text{ ppm} \quad (1)$$

$$\text{Ratio in Clean Air}_{\text{MQ-135}} = 3.6 \text{ ppm} \quad (2)$$

All MQ sensors ran continuously for 2 days before the calibration algorithm. The algorithm sent the $R0$ value to a Google Sheets file for storage. These ran for 4 hours with a polling interval of 30 seconds. The readings were averaged and results were used for the SetR0 function in MQ sensors.

The calibration algorithm for the SGP30, MQ-7, and MQ-135 sensors were done simultaneously. Fig. 8 shows the calibration setup, where all the IAQM devices were placed near an open window allowing the sensors to take in clean air from their surroundings.

3) DHT22: The DHT22 had two parts in its calibration: temperature and relative humidity. An additional temperature and humidity device was utilized as a reference to calibrate the DHT22. The calibration setup was designed so that changes in temperature and humidity is apparent. Particularly, the reference device and DHT22 sensors were placed in a room with an air conditioning unit to allow temperature changes. Low and high temperature readings were taken while the devices were running. Moreover, the reference device and the DHT22 sensors were placed inside an enclosed container with materials that could change the relative humidity. Specifically, a small dish containing salt water and packets of desiccants were utilized to increase and decrease humidity, respectively. Humidity readings were taken 12 hours after each humidity-changing material was added. Fig. 9 and 10 show the setup of each one respectively. Readings of the DHT22 sensors were sent to a Google Sheets file every 30 seconds, while readings for the reference device were manually noted.

The obtained temperature and humidity readings were compared against the reference readings. Moreover, a line of best fit was utilized to create correction equations to be used in the devices' firmware.

Fig. 8: Calibration Setup of SGP30, MQ-7, and MQ-135 Sensors

Fig. 9: Calibration Setup of DHT22 Using Salt Water

Fig. 10: Calibration Setup of DHT22 Using Desiccant

4) PMS7003: The process for calibrating the PMS7003 sensors was similar to the method of calibrating DHT22 sensors as it likewise utilizes a reference device, a commercially available air quality device that contains a PMS3003 sensor. The setup was made such that all sensors and reference device

was placed in the same area in the room, shown in Fig. 11. One significant difference in its calibration process from the DHT22 was the absence of the ability to control the concentration of $PM_{2.5}$ and PM_{10} in the room. Hence, the setup employed was to run all devices continuously for a day and compare their values. The PMS7003 sensors were connected to an ESP-32 microcontroller that sends $PM_{2.5}$ and PM_{10} sensor readings every 30 seconds to a Google Sheets file through an HTTP request.

A problem faced in this setup is the inability of the reference device to store readings. Hence, a solution employed was to capture the device's screen using a phone every 30 seconds. The automation of this process was made possible by the "Open Camera" Android application. Moreover, Tesseract was used for optical character recognition (OCR) to extract the data from the pictures, particularly the time, $PM_{2.5}$, and PM_{10} concentration values. Afterwards, Python was utilized to automate the OCR process and join the readings of the reference device and the PMS7003 sensors based on the time the measurement was observed. The values from the reference device were then placed on a .csv file for analysis.

Comparisons were then made between the PMS7003 sensors and the reference device. The best fit line method was utilized to find correction and calibration equations for $PM_{2.5}$ and PM_{10} in each sensor to ensure that the PMS7003 sensors were similar to the reference device.

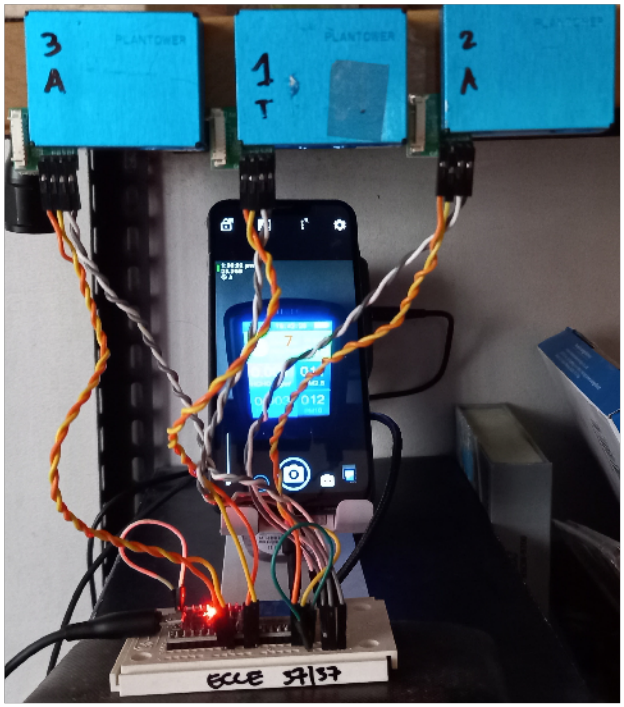

Fig. 11: Calibration Setup of PMS7003

5) Photoresistor Module: Since the photoresistor module was used only to see patterns in room lighting, the only calibration method utilized was to deploy the module in the rooms and observe their range of values. The average maximum value was then used to calculate the photoresistor value threshold for polling intervals.

D. Fuzzy Logic to Estimate Environment Indoor Air Quality Index

The fuzzy inference system takes the outputs of the CO_2, CO, $PM_{2.5}$, PM_{10}, temperature, and humidity sensors as an input to the logic to estimate the IAQ of a classroom and subsequently classify the environment indoor air quality index (EIAQI) according to Dionova's EIAQI four-category assessment: hazardous, bad, good, and excellent. The maximum and minimum values of the sensors' analog readings are first identified from the sensor datasheet, which are then utilized in the membership functions of the fuzzy inference system to create three output variables, namely IAQI, TCI1, and TCI2, after defuzzification based on 48 fuzzy rules. These were then processed by summing and averaging the three output variables to create an EIAQI value scaled from 0 to 100, forwarding this result into Google Sheets and back to the ESP-32 through an HTTP response. This process is outlined in Fig. 12.

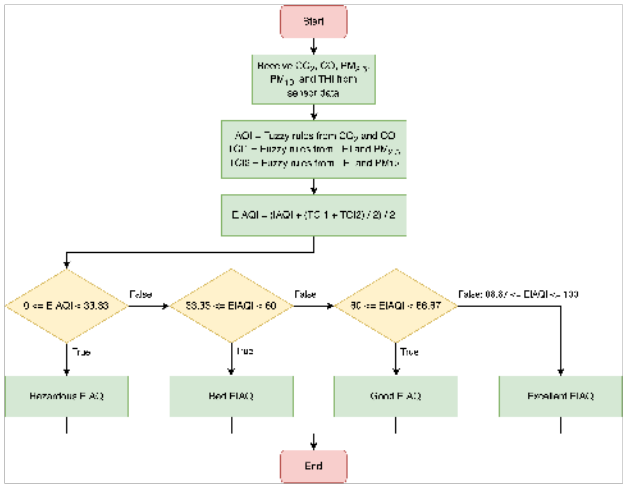

Fig. 12: Fuzzy Inference System Flowchart

The fuzzy inference system necessitates the use of the temperature-humidity index (THI) based on the equation used by Chan et al. The equation will be implemented through Google Apps Script since the raw data will be taken for intermediate calculation before rendering as a new dataset. This calculation will then be sent to the cloud Python script. (3) shows the equation for THI given T, temperature in Fahrenheit, and H, humidity in % RH.

$$\text{THI} = T - 0.55(1 - H)(T - 58) \tag{3}$$

Diving into the specifics of the fuzzy logic, the 48 fuzzy rules come from Dionova's recommended 4x4 matrix designed to classify four categories when comparing two components against each other. These rules are shown in Table I, II, and III.

CO \ CO_2	Good	Moderate	Unhealthy	Hazardous
Good	Good	Moderate	Moderate	Unhealthy
Moderate	Moderate	Moderate	Unhealthy	Unhealthy
Unhealthy	Moderate	Unhealthy	Unhealthy	Unhealthy
Hazardous	Unhealthy	Unhealthy	Unhealthy	Hazardous

TABLE I: Indoor Air Quality Index (IAQI) Rule Base

THI \ $PM_{2.5}$	Good	Moderate	Unhealthy	Hazardous
Comfortable	Very Comfortable	Comfortable	Comfortable	Neutral
Slight Discomfort	Comfortable	Comfortable	Neutral	Neutral
Moderate Discomfort	Comfortable	Neutral	Neutral	Neutral
Extreme Discomfort	Neutral	Neutral	Neutral	Uncomfortable

TABLE II: Thermal Comfort Index 1 (TCI1) Rule Base

THI \ PM_{10}	Good	Moderate	Unhealthy	Hazardous
Comfortable	Very Comfortable	Comfortable	Comfortable	Neutral
Slight Discomfort	Comfortable	Comfortable	Neutral	Neutral
Moderate Discomfort	Comfortable	Neutral	Neutral	Neutral
Extreme Discomfort	Neutral	Neutral	Neutral	Uncomfortable

TABLE III: Thermal Comfort Index 2 (TCI2) Rule Base

Accounting for threshold points, Dionova et. al's and Chan et. al's studies helped in establishing four separate categories to determine how good or bad component levels are. These result in three output variables, namely IAQI, TCI1, and TCI2, which are described in a similar manner. For most of the implementation except for CO and CO_2 levels, the trapezoidal function from MATLAB's Fuzzy Logic Toolbox was used to recreate the final output values given that there was a 46.8% difference between the EIAQI taken previously when utilizing triangular functions, and similar differences to all outputs as well. These threshold points are shown in Table IV, V, and VI.

CO_2 (ppm)	CO (ppm)	IAQI	IAQI Status
0 - 606	**0 - 2**	75 - 100	Good
580 - 1020	**1.8 - 8.5**	58.33 - 83.33	Moderate
800 - 1520	**7 - 10**	25 - 66. 66	Unhealthy
1480 - 5000	**9 - 50**	0 - 50	Hazardous

TABLE IV: IAQI Threshold Points

The following figures describe the utilization of trapezoidal and triangular functions on all input and output variables as seen from the threshold points from Table IV, V, and VI. The Fuzzy Logic Designer shows these as shown in Fig. 13 to 23.

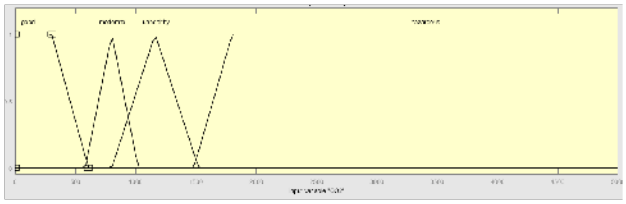

Fig. 13: Input Variable CO_2 Membership Function

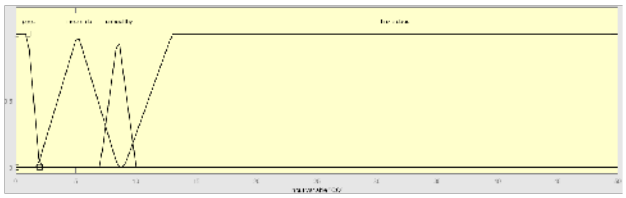

Fig. 14: Input Variable CO Membership Function

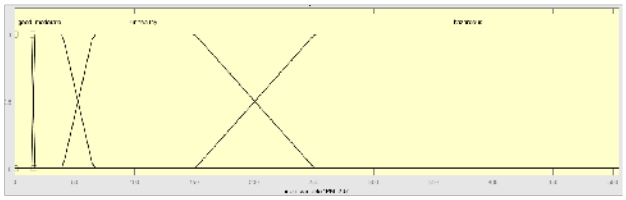

Fig. 15: Input Variable $PM_{2.5}$ Membership Function

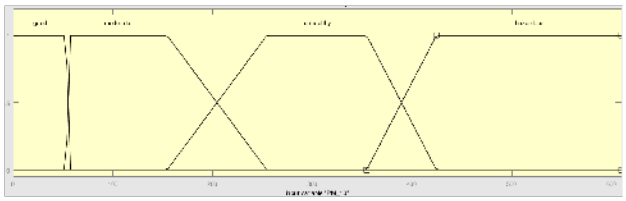

Fig. 16: Input Variable PM_{10} Membership Function

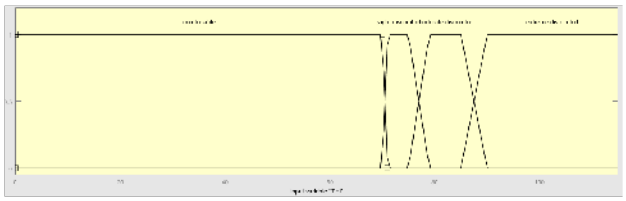

Fig. 17: Input Variable THI Membership Function

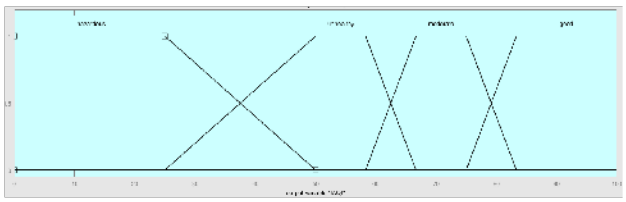

Fig. 18: Output Variable IAQI Membership Function

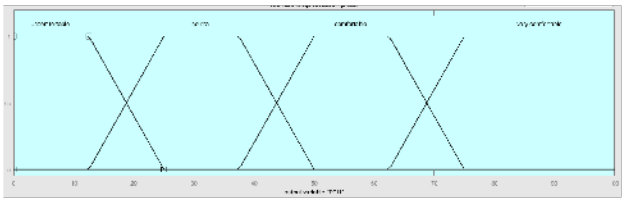

Fig. 19: Output Variable TCI1 Membership Function

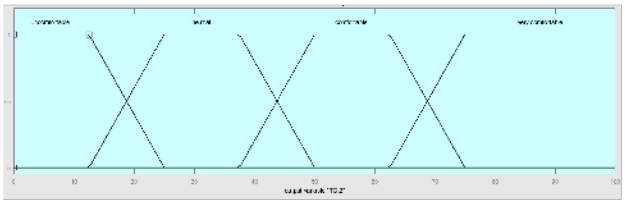

Fig. 20: Output Variable TCI2 Membership Function

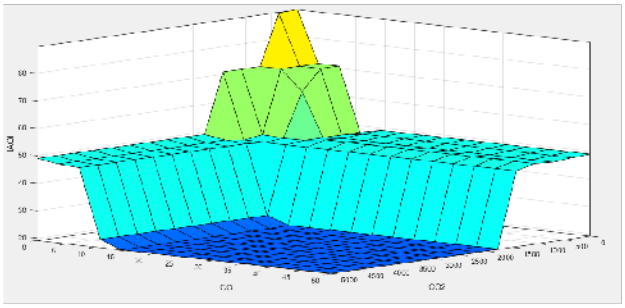

Fig. 21: MATLAB Fuzzy Logic IAQI Surface Graph

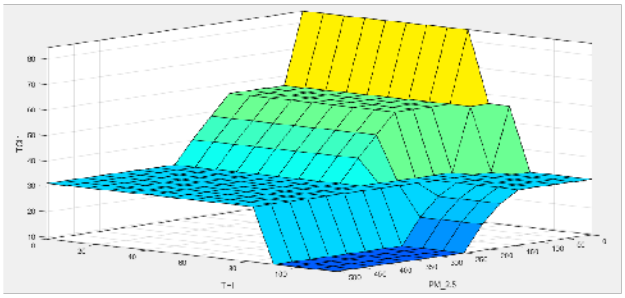

Fig. 22: MATLAB Fuzzy Logic TCI1 Surface Graph

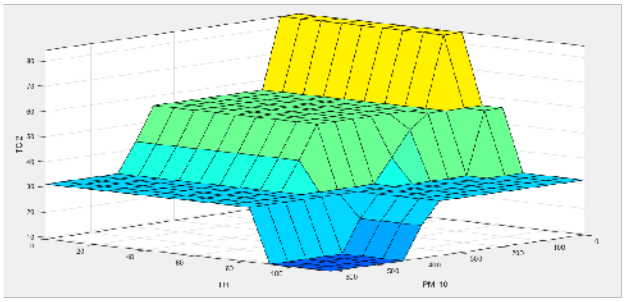

Fig. 23: MATLAB Fuzzy Logic TCI2 Surface Graph

$PM_{2.5}$ ($\mu g/m^3$)	THI	TCI1	TCI1 Status
0 - 16	**0 - 71**	62.5 - 100	Very Comfortable
15 - 65.5	**70 - 79**	37.5 - 75	Comfortable
40 - 250.5	**75 - 90**	12.5 - 50	Neutral
150.5 - 505	**85 - 115**	0 - 25	Uncomfortable

TABLE V: TCI1 Threshold Points

PM_{10} ($\mu g/m^3$)	THI	TCI2	TCI2 Status
0 - 55	**0 - 71**	62.5 - 100	Very Comfortable
54 - 254.5	**70 - 79**	37.5 - 75	Comfortable
154.5 - 424.5	**75 - 90**	12.5 - 50	Neutral
354.5 - 610	**85 - 115**	0 - 25	Uncomfortable

TABLE VI: TCI2 Threshold Points

Based on Fig. 13 to 23, it is seen how the MATLAB Fuzzy Logic Designer Toolbox showcases a visual representation of the ranges of where the fuzzy logic implementation was made. For Fig. 13 and 14, these follow Dionova et al.'s implementation, showcasing membership functions that utilize ranges integrating triangular and trapezoidal functions to depict an accurate representation of the IAQI. However, Fig. 15 and 16 utilize Chan et al.'s ranges based on their study, as Dionova et al. were not able to use the PM_{10} variable in their implementation. Fig. 17 then uses the THI ranges from Schlatter's discussion. All modified functions used trapezoidal functions to avoid the median phenomenon that occurs on extreme ends and gives a precise index in the final values after the output variables for Fig. 18 to 20 were defuzzified. Fig. 21 to 23 shows the surface graph for all output variables where an overarching conclusion could be made. Most values are skewed right, with more possible values for all datasets leaning towards the latter two categories showcasing general unpleasantness across all variables. This is caused by considering worst-case values for each component, highlighting that the number of values falling into that spectrum should be many. Similarly, very few value combinations for each correlation in each clustered index can appear as these fall under a very small and specific range. Thus, this should not affect the final EIAQI rendered given that better ranges could potentially be repeated over a period of time should it fall under this category, marking this as a good representation of what is to be expected, especially when taking these indices into consideration for the final EIAQI.

E. Google Sheets Data Ingestion

Data ingestion from the ESP-32 to Google Sheets is powered by the Google Apps Script web application. The process is shown in Fig. 24.

Upon receiving an HTTP request from the ESP-32 microcontroller detailing the sensor readings, the URL parameters are first checked for validity, detecting unknown parameters or non-numeric arguments. In such cases, the Google Apps Script program returns an error to the ESP-32. An additional password parameter is required when sending HTTP requests

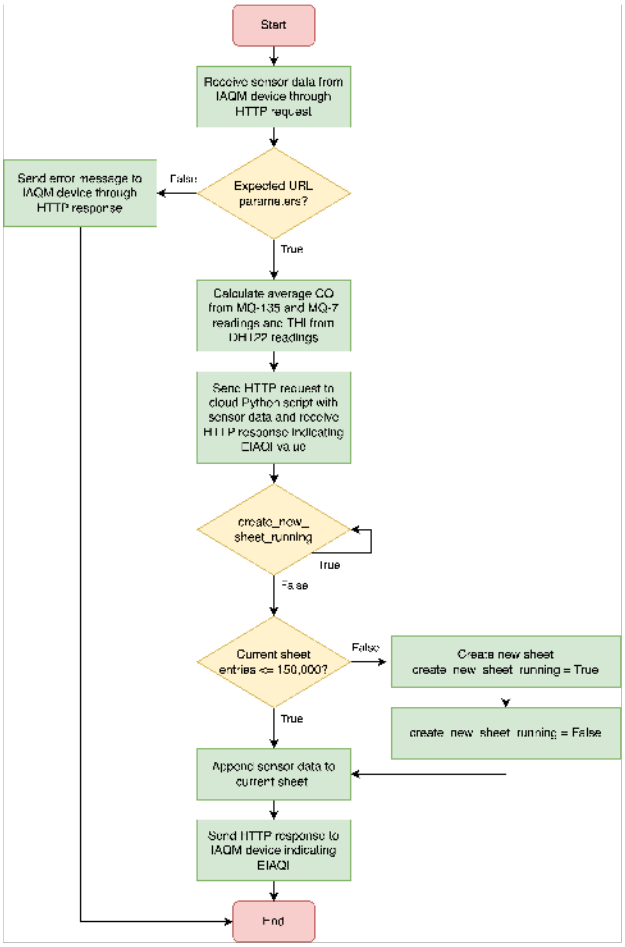

Fig. 24: Google Apps Script Web Application Flowchart

to Google Apps Script in order to prevent unauthorized access and preserve data integrity.

Once confirmed for validity, Google Apps Script sends an HTTP request to a cloud Python script, indicating only the required parameters to compute for the environment indoor air quality index (EIAQI), including another password parameter for the same purpose. Since the IAQM device uses two CO sensors, namely MQ-7 and MQ-135, the average of their readings is used instead as an input to the fuzzy logic controller. Upon retrieval of the EIAQI value, the data entry involving sensor data from the ESP-32 and EIAQI value from the cloud Python script is appended to the Google Sheets file in use.

However, it was observed that Tableau Desktop, the data visualization dashboard, can only connect to Google Sheets files with a maximum of 150,000 entries. In order to handle this exception, for each HTTP request from the ESP-32 to Google Apps Script, the program first checks for the total number of entries within the current Google Sheets file in use. If the count of current entries exceed 150,000, a new Google Sheets file will be automatically created. However, since the process for the creation of a new Google Sheets file takes time, subsequent HTTP requests from the ESP-32 while this process is still ongoing will be held until completed. Without exception handling, it is possible that the program enters an infinite loop that may result in the creation of an infinite number of Google Sheets files. The information for the current Google Sheets file ID, sheet number, and a Boolean parameter to determine the status of the creation of a new Google Sheets file is stored as a script property of the Google Apps Script project so that they remain accessible even when the program is not running.

Finally, an HTTP response is returned to the ESP-32 in order to compute the 10-previous moving median (10MM) EIAQI value and consequently update the color of the RGB LED.

An additional time-based trigger is installed on Google Apps Script that sorts all the data in the current Google Sheets file by timestamp and room name in ascending order every 5 minutes. This is to mitigate instances where delays in the ESP-32's Wi-Fi connection that may lead to an earlier sensor reading being sent to Google Apps Script at a later time as opposed to other IAQM devices.

F. Fuzzy Inference System Implementation in Python

Given the limited functionality of MATLAB within the context of the series of HTTP requests between the ESP-32 and Google Apps Script, it was determined that the implementation of the fuzzy inference system should be performed in Python for improved versatility.

In order to replicate the fuzzy inference system program from MATLAB's Fuzzy Logic Toolbox, the scikit-learn Python package was utilized, implementing antecedent and consequent functions that act as parameters with a chosen number of points for all input and output variables, respectively. Meanwhile, the "ctrl" function allows for fuzzy rule creation and simulations. This process is outlined in Fig. 25

The Flask Python package is likewise used to convert the fuzzy logic controller into a web application that can be used by Google Apps Script to compute for the EIAQI given a set of sensor data. This may is accomplished through an HTTP request that uses the route "/eiaqi" as defined in the Python program.

Similar to the Google Apps Script web application, a password parameter is required when sending HTTP requests in order to prevent unauthorized access, which may disrupt the VM's operation. In the event of unexpected URL parameters, such as unknown parameters or non-numeric arguments, the Python program returns an error to Google Apps Script.

The Python program is hosted on a Ubuntu 22.10 x64 virtual machine (VM) in Singapore with 1 GB memory, which costs approximately $6 per month in DigitalOcean. In order to get the Python program accessible through the VM's public IP address at the specified route, NGINX, a web server that can be used as a reverse proxy, is installed onto the VM. Essentially, it serves as an intermediary between Google Apps Script and the cloud Python script, enabling communication through HTTP requests. Furthermore, NGINX removes the need to specify the port of the cloud Python script and is immediately accessible at the VM's public IP address.

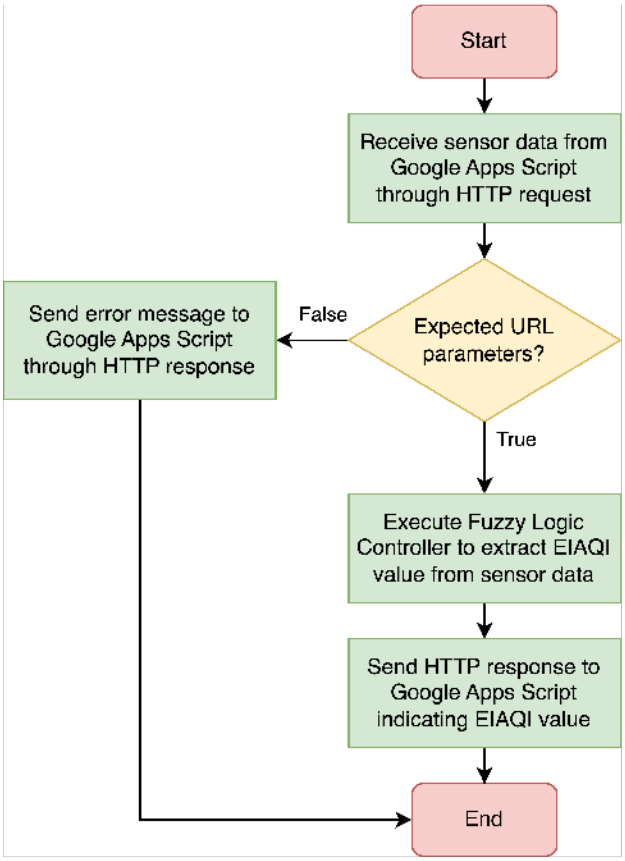

Fig. 25: Python Web Application Flowchart

Afterwards, Node.js is installed in order to access the PM2 package, a production process manager. Using PM2, the cloud Python script can be left running from the virtual machine even when access to the cloud terminal is terminated, as shown in Fig. 26.

```
root@compute-ciaqi:~# pm2 list
```

id	name	mode	↺	status	cpu	memory
0	compute_ciaqi	fork	0	online	0%	84.5mb

Fig. 26: Running Programs in the Virtual Machine as seen from the Terminal

G. Fuzzy Logic to Determine RGB LED Color and Sensor Polling Interval

The EIAQI value calculated by the fuzzy logic controller is sent to the IAQM device, as shown in Fig. 7. The retrieved EIAQI value from the HTTP response is then inserted into a 10-element array in the IAQM device. The 10MM EIAQI value, which is the median value calculated from the aforementioned array, is used to determine both the RGB LED color and the polling interval of the sensors. The process of changing them in the firmware is illustrated in Fig. 27 and 28, respectively. The polling interval will not adjust based on the 10MM EIAQI value if the photoresistor reading is greater than the threshold (90% of the maximum photoresistor reading). This is because it is assumed that the room will not be occupied by people at night, eliminating the need for quick information gathering.

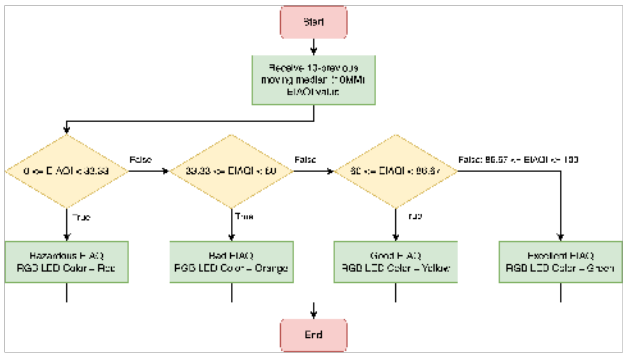

Fig. 27: IAQM Device Firmware Flowchart for Changing RGB LED Color

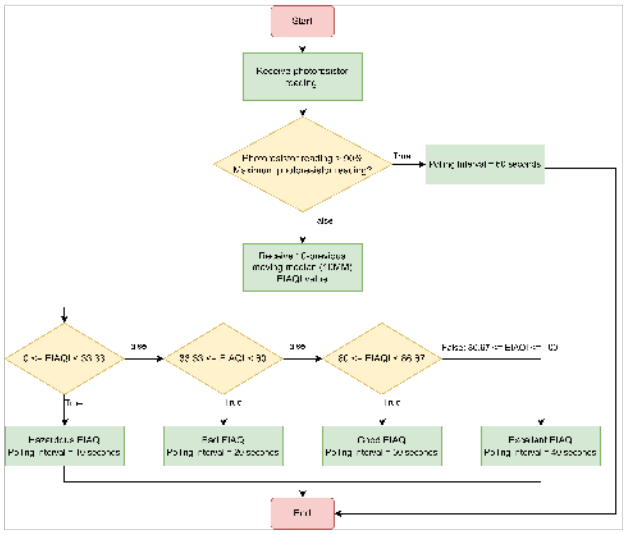

Fig. 28: IAQM Device Firmware Flowchart for Changing Polling Interval

H. Tableau Dashboard

In order to visualize the IAQ status of the rooms, live data from Google Sheets is imported to Tableau using the Google Drive API. However, because Tableau can only connect to Google Sheets files with a maximum of 150,000 entries, multiple sheets must be connected to Tableau, resulting in the need to perform a union operation to combine the data into a single logical table.

Considering the existence of hundred thousands of entries involving sensor data and EIAQI values, the dashboard is configured to ingest data in an incremental manner, identifying new rows based on the "Timestamp" column. The Tableau dashboard is shown in Fig. 29.

From Fig. 29, information pertaining to the latest readings are displayed in the top half of the dashboard, all expressed

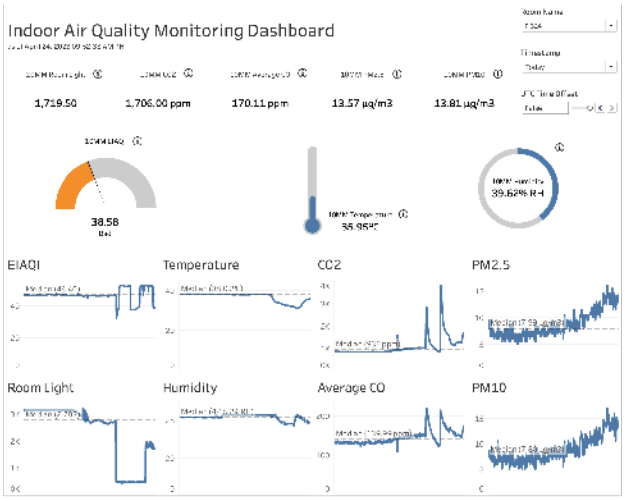

Fig. 29: Tableau Dashboard

as 10-previous moving median (10MM) values. Embedded within the latest readings are information buttons which, when hovered, displays a brief description of the air quality indicator being measured. On the other hand, historical readings are displayed in the bottom half of the dashboard in the form of line charts, whose time frame can be adjusted using the "Timestamp" control filter. An additional broken line guide describing the median of the sensor data over the specified time frame is also present to gain an overview of its usual value.

Moreover, three control filters are available, namely "Room Name," "Timestamp," and "UTC Time Offset." The "Room Name" filter controls the information displayed on the dashboard to reflect the specified room. On the other hand, the "Timestamp" filter controls the scope of the line charts at the bottom half of the dashboard view to increase or decrease the time frame. Finally, the "UTC Time Offset" filter is a Boolean parameter that ensures that relative times are accurate when hosted online, where server times are in Coordinated Universal Time (UTC). When enabled, the filter subtracts 8 hours from all time values in order to align timestamp values with Philippine time.

In order to easily access the dashboard even without the Tableau Desktop application, the dashboard is hosted on Tableau Public, which can be viewed using a dedicated URL link. Hosting the dashboard on Tableau Public allows the uploader to request data refresh upon a click, thus still able to present live data from the ESP-32 microcontroller.

IV. Results and Discussion

A. Calibration Results

The results of the calibration methods outlined in the methodology are presented in the following, and the baseline values and correction equations have been incorporated into the sensors' respective IAQM device firmwares.

1) SGP30, MQ-135, and MQ-7: The results of the calibration for the SGP30, MQ-135, and MQ-7 are compiled in Table VII. It is important to note that the values in the SGP30 column are in hexadecimal format, which is required by the function of the Adafruit SGP30 library. Additionally, the values in the MQ-135 and MQ-7 column correspond to the $R0$ values used in the MQUnifiedSensor library.

Device Name	CO_2	MQ135	MQ7
F-323 A	0x96AC	1.893319104	0.1675133404
F-323 B	0x9A68	22.03808893	1.054153264
F-324	0x9D87	47.47781824	12.41954391

TABLE VII: SGP30, MQ-135, and MQ-7 Calibration Values

2) DHT22: The temperature results are presented in Fig. 30, 31, and 32, while Fig. 33, 34, and 35 display the humidity results. The figures illustrate that the readings are linear, and the equations shown on the figures can be used to correct the DHT22 readings. However, to improve the accuracy of the readings, it would be advisable to gather more data points by applying the calibration methodology of the PM7003 device, involving OCR, rather than relying on a limited number of points.

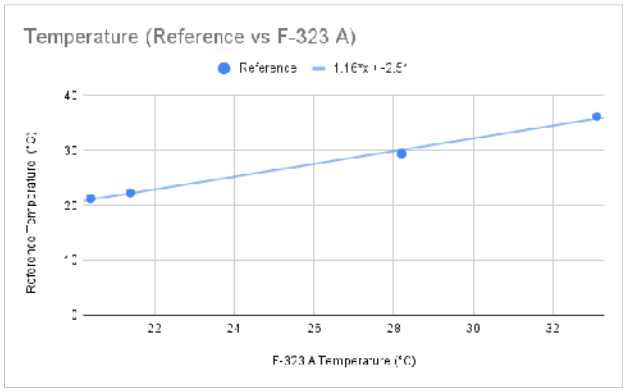

Fig. 30: Calibration Equation for F-323 A Temperature

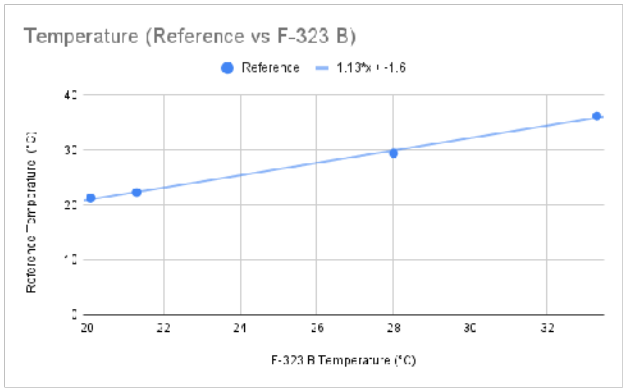

Fig. 31: Calibration Equation for F-323 B Temperature

3) PMS7003: Compared to the DHT22 results, the PMS7003 results have a larger number of collected data points. The $PM_{2.5}$ and PM_{10} results are presented in Fig. 36, 37, 38, 39, 40, and 41. All figures show some degree of linearity.

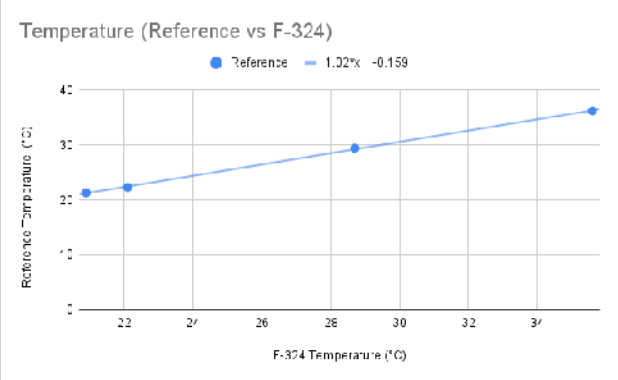

Fig. 32: Calibration Equation for F-324 Temperature

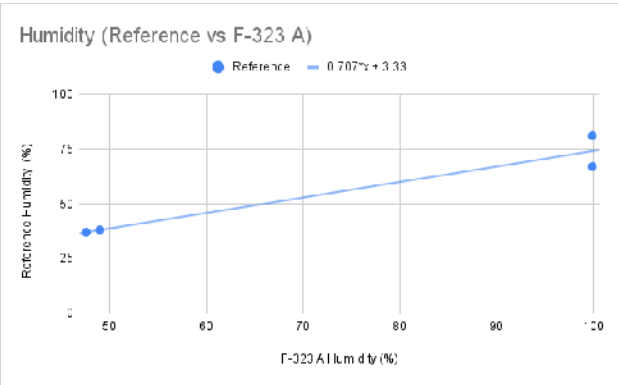

Fig. 33: Calibration Equation for F-323 A Humidity

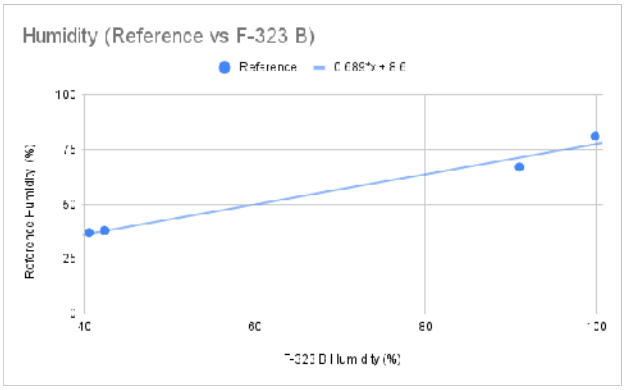

Fig. 34: Calibration Equation for F-323 B Humidity

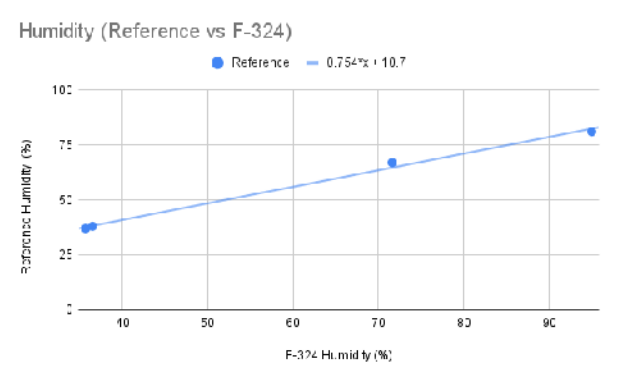

Fig. 35: Calibration Equation for F-324 Humidity

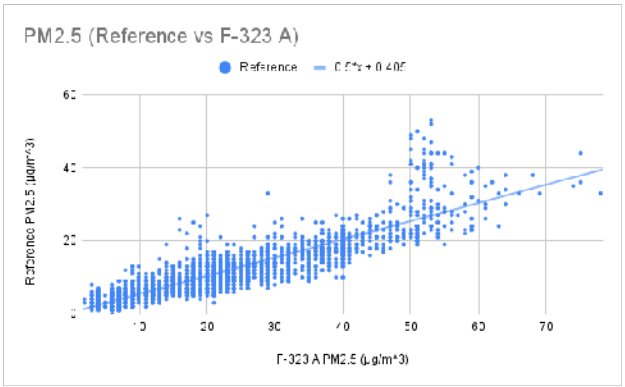

Fig. 36: Calibration Equation for F-323 A $PM_{2.5}$

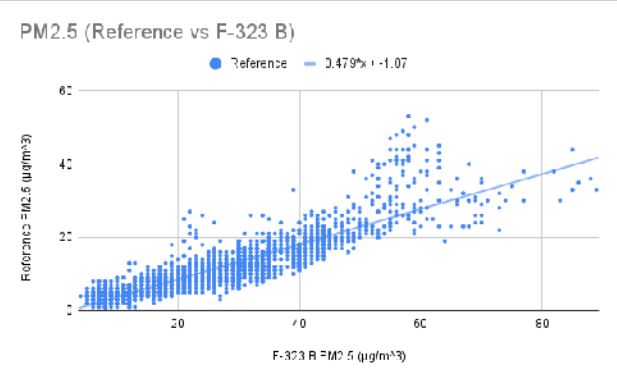

Fig. 37: Calibration Equation for F-323 B $PM_{2.5}$

However, for higher values of $PM_{2.5}$ and PM_{10} (particularly between 50 to 60 $\mu g/m^3$ of $PM_{2.5}$ and 60 to 70 $\mu g/m^3$ of PM_{10}), the points do not appear to follow the linear trend. Despite this observation, the correction equations based on the best fit lines were still used in the IAQM device firmware.

B. Comparison Analysis Between Dionova and Fuzzy Logic Implementation

Based on our comparison analysis, taking the test values from the study of Dionova et. al and our current implementation for the IAQM system, it was found that the use of trapezoidal functions posed smaller percentage differences when compared to the triangular function implementation of Dionova et. al's research. On a sample dataset, the IAQI from this newly innovated implementation shows a 3.47% percent difference, comparing the IAQI values of 52.83 to 51.03. On TCI averages based on two cluster indices, there is also a notable 1.64% percent difference, comparing the TCI values of 43.745 and 44.47, respectively. However, despite the EIAQI differences with Dionova being variable and scattered, the average difference still comes out to 9.26%, being within

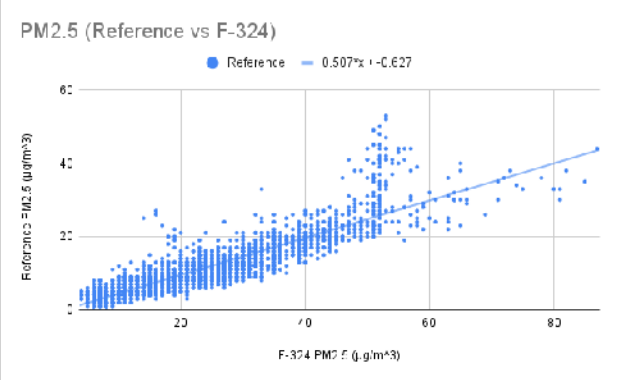

Fig. 38: Calibration Equation for F-324 $PM_{2.5}$

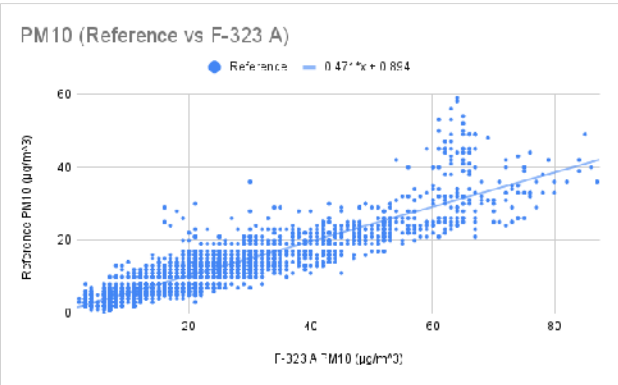

Fig. 39: Calibration Equation for F-323 A PM_{10}

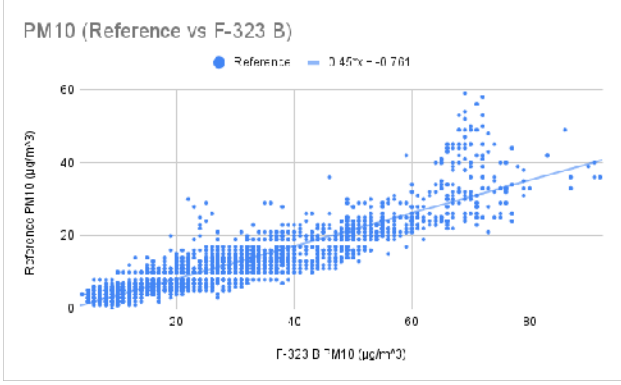

Fig. 40: Calibration Equation for F-323 B PM_{10}

acceptable bounds. Furthermore, this implementation outputs a final recommendation result that evidently matches other studies, with exceptions to extreme cases wherein Dionova et. al scales their index in a pessimistic lens and a manner that considers their sensors. Overall, the usage of the current fuzzy inference system presents good promise without compromising how data is construed for users to take advantage.

C. Collected Data

The present study involved the collection of air quality parameters within Faura Hall, specifically in rooms F-323 and F-324. As previously mentioned, F-323 was outfitted with two installed indoor air quality monitoring (IAQM) devices, owing to the substantial size of the room. Conversely, F-324 only had one IAQM device deployed. Data was collected over the course of a month, and the ensuing subsections aim to comprehensively analyze each of the aforementioned parameters.

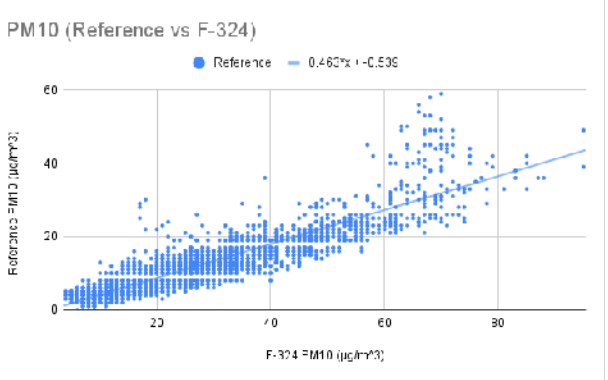

Fig. 41: Calibration Equation for F-324 PM_{10}

EIAQI (Dionova)	Proposed EIAQI	Percent Difference
65.17	52.83	20.91%
39.27	39.49	0.56%
90.27	84.36	6.77%
60.50	52.14	14.84%
82.67	70.11	16.44%
51.08	52.05	1.88%
25.00	20.01	22.17%
80.63	77.45	4.02%
52.78	51.04	3.35%
70.96	69.79	1.66%

TABLE VIII: EIAQI Comparison Analysis

1) Room Light: Fig. 42, 43, and 44 depict the visualized data gathered for room light in the aforementioned rooms. It should be noted that these values are analog in nature and are not accompanied by any units of measurement. Nonetheless, it is worth emphasizing that higher analog values correspond to a darker room and conversely, lower values imply greater luminosity. Evidently, the room light readings for F-323 A and F-323 B frequently peaked at a value of 4095, whereas the F-324 readings typically peaked at 3100, albeit higher values were also recorded on certain occasions.

Given that the calibration methodology stipulated that the maximum values be utilized in determining the room light threshold for the polling rate, the maximum value of 4095 was employed for the F-323 A and F-323 B IAQM devices. By contrast, the maximum value of 3100 was deemed suitable for the F-324 device, considering it was the average peak value recorded.

2) Temperature: The temperature data collected is shown in Fig. 45, 46, and 47. Upon close examination, it is apparent that all the figures exhibit similar peaks, but their minimum values vary significantly. The data for F-324 displays lower

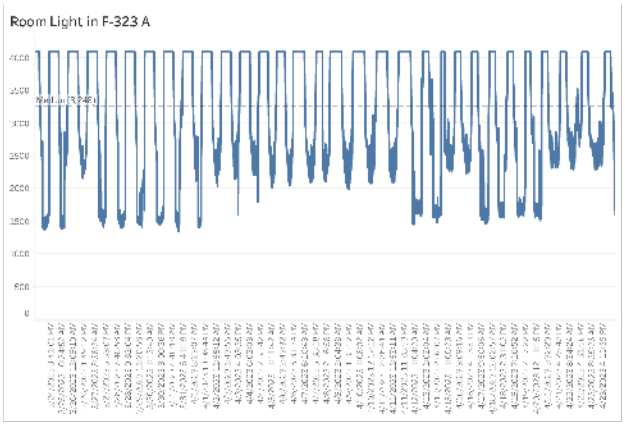

Fig. 42: Room Light in F-323 A Over a Month

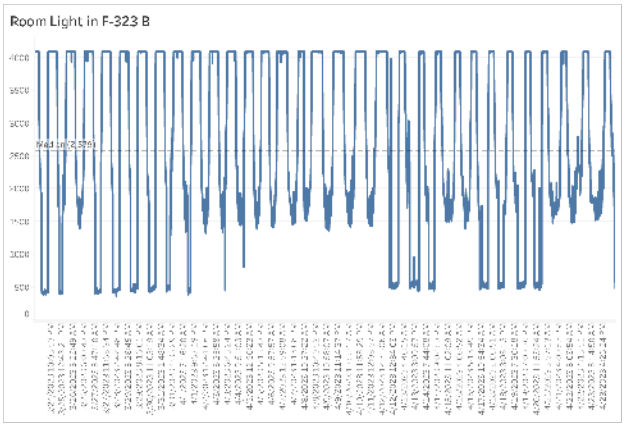

Fig. 43: Room Light in F-323 B Over a Month

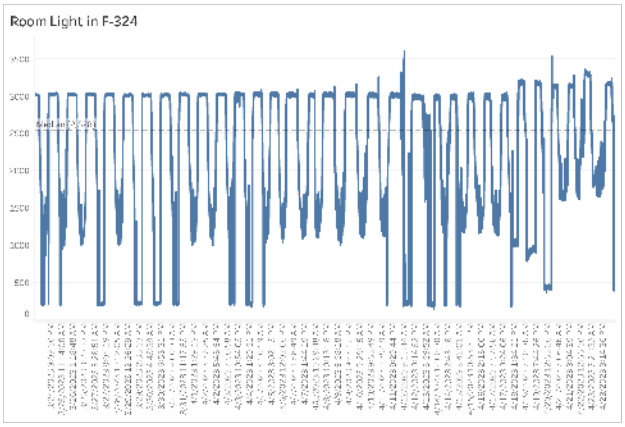

Fig. 44: Room Light in F-324 Over a Month

temperatures, which could be due to the frequent use of air conditioning. On the other hand, F-323 B has lower temperatures than F-323 A, although not as frequent as F-324. F-323 A, on the other hand, has fewer instances of lower temperatures. Despite some minor temperature fluctuations, these are not as significant as those seen in F-323 B, suggesting that the air conditioning unit near F-323 B may have caused the changes.

However, it is important to note that the temperature values are unusually high, with median values ranging from 36.5∘C to 39∘C. These values are well above the normal temperature range of 25∘C to 32∘C. Therefore, the reliability of these readings is questionable and requires further investigation in subsequent sections.

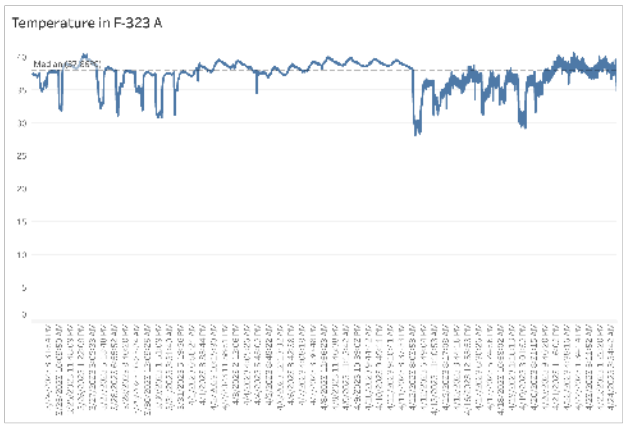

Fig. 45: Temperature in F-323 A Over a Month

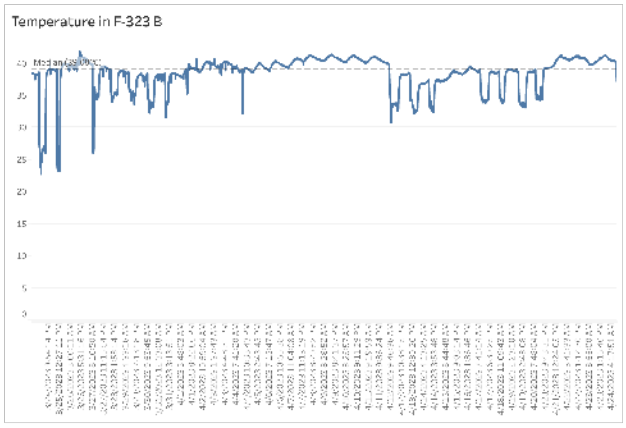

Fig. 46: Temperature in F-323 B Over a Month

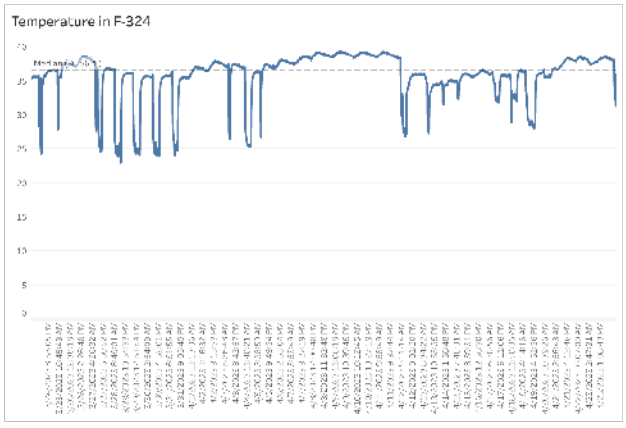

Fig. 47: Temperature in F-324 Over a Month

3) Humidity: The data collected for the humidity parameter is illustrated in Fig. 48, 49, and 50. From the figures, it is apparent that F-323 A and F-323 B exhibit analogous trends at the onset of the data collection. However, from April 14, 2023, onwards, F-323 A depicts higher humidity values than F-323 B. During this period, F-323 B humidity values appear to be consistent with F-324 data. Despite this, the median values for all the devices are relatively close to each other.

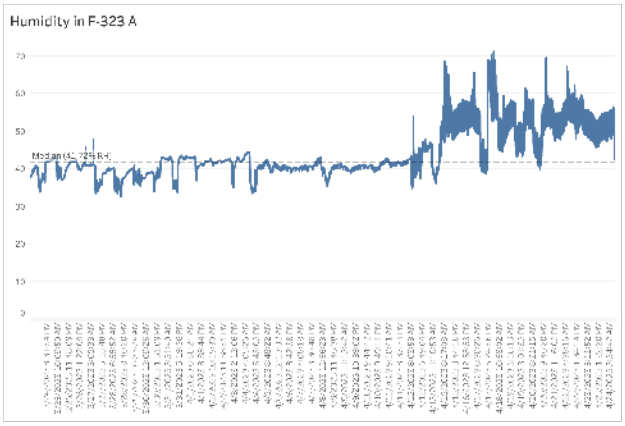

Fig. 48: Humidity in F-323 A Over a Month

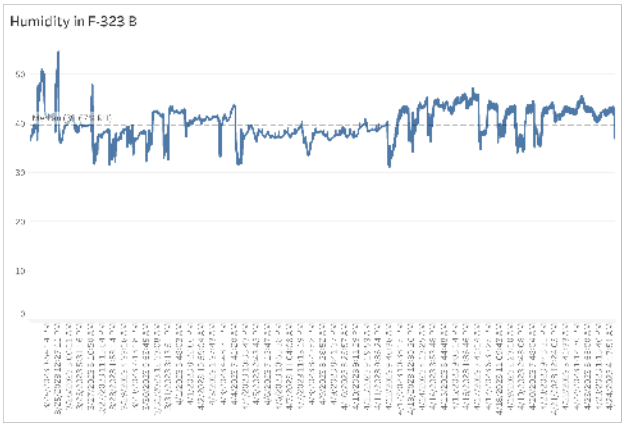

Fig. 49: Humidity in F-323 B Over a Month

4) CO_2: In Fig. 51, 52, and 53, spikes in the collected sensor data are noticeable at first glance. These spikes may most likely be errors or outliers as some of the values exceed the range of the SGP30 sensor. However, upon analysis of the trends, it appears that F-323 A and F-323 B exhibit similar trends, which suggests that a single SGP30 sensor is capable of detecting general trends even in a large room. This is supported by the median values, as F-323 A and F-323 B exhibit similar median values of 1,162 ppm and 1,172 ppm, respectively, over a one month period. F-324, on the other hand, has a median value of 838 ppm. The F-323 room may have issues with ventilation, as having a median of 1,100 ppm CO_2 is high.

5) Average CO: A notable finding in analyzing the average CO data through Fig. 54, 55, and 56 is its large values. Particularly, it could be seen that the median value spans from

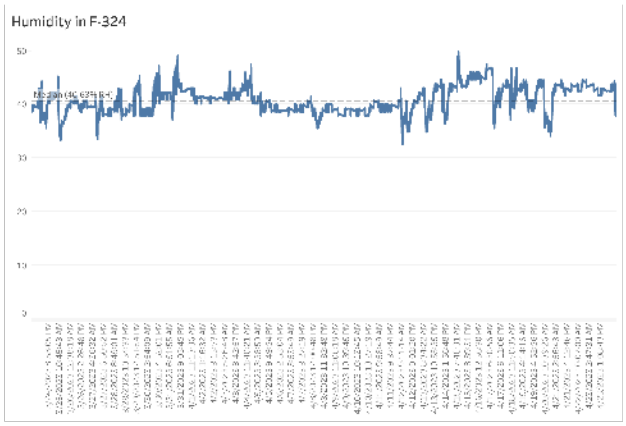

Fig. 50: Humidity in F-324 Over a Month

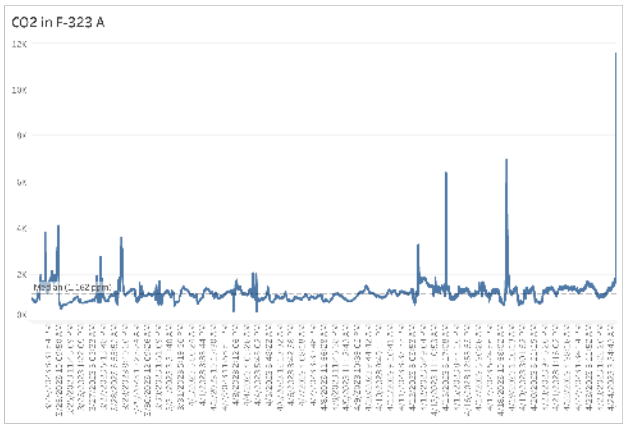

Fig. 51: CO_2 in F-323 A Over a Month

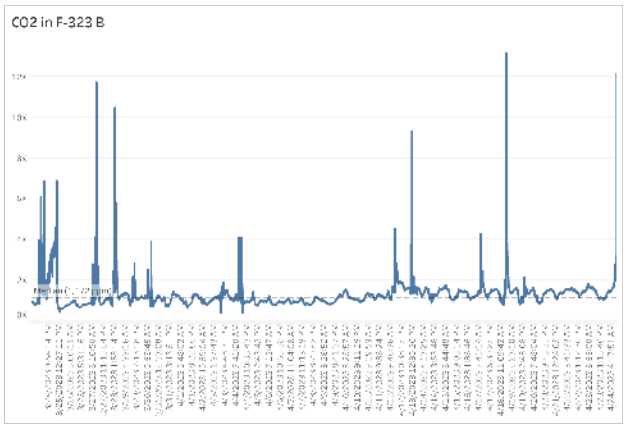

Fig. 52: CO_2 in F-323 B Over a Month

11.76 ppm to 75.41 ppm. This implies a problem within the sensors, either in the calibration or the setup. However, while looking at the values obtained from the MQ-7 and MQ-135 sensors, the MQ-7 readings are particularly low from 1 to 5 ppm while the MQ-135 readings range from 15 to 150 ppm. This means that the MQ-135 is skewing the average CO readings. An explanation for this skewing is that MQ-135 is a sensor that measures air quality through different gases, such

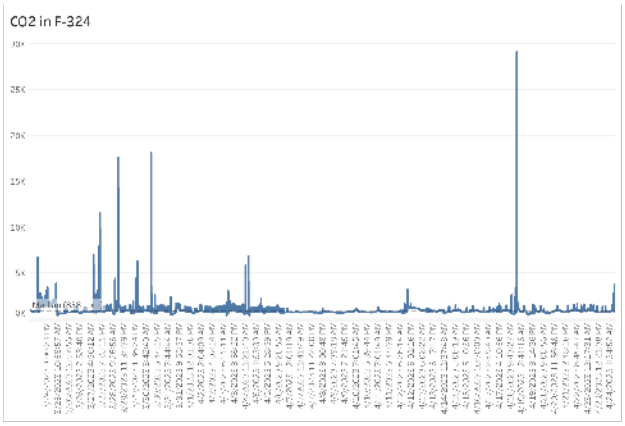

Fig. 53: CO_2 in F-324 Over a Month

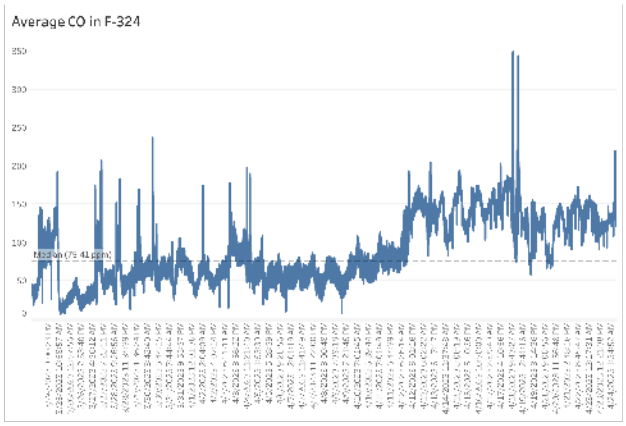

Fig. 56: Average CO in F-324 Over a Month

as CO_2, CO, and alcohol. Since this sensor has the ability to detect a multitude of gases, it is prone to be affected by other gases even if it is set to only read one gas.

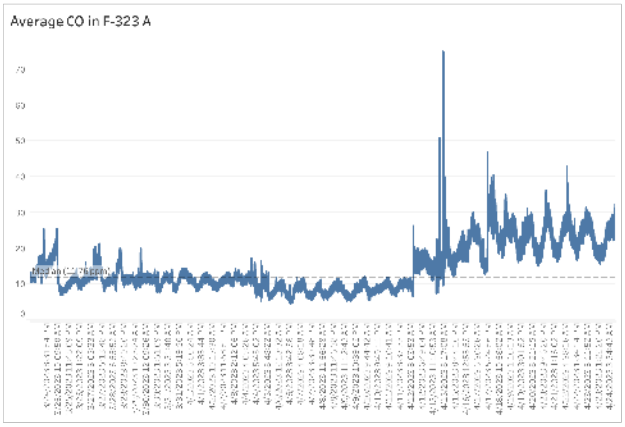

Fig. 54: Average CO in F-323 A Over a Month

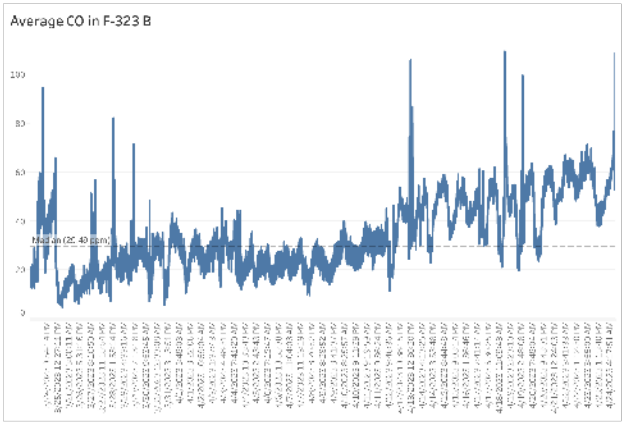

Fig. 55: Average CO in F-323 B Over a Month

6) $PM_{2.5}$: Upon examination of Fig. 57, 58, and 59, it is evident that all $PM_{2.5}$ readings exhibit a similar pattern. However, discrepancies in the data were observed on April 15, 2023, where F-323 A registered higher $PM_{2.5}$ levels compared to F-323 B, while F-324 indicated relatively lower values than F-323 B. Nonetheless, all devices showed a consistent trend again on April 18, 2023. Notably, the median $PM_{2.5}$ levels of F-323 A and F-323 B were comparable at 7.9 μg/m^3 and 8.03 μg/m^3, respectively. Conversely, F-324 recorded a median value of 6.98 μg/m^3, indicating lower values compared to F-323.

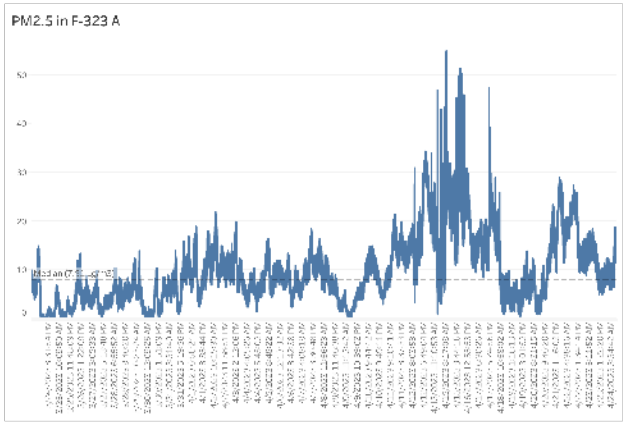

Fig. 57: $PM_{2.5}$ in F-323 A Over a Month

7) PM_{10}: The PM_{10} readings in Fig. 60, 61, and 62 exhibit similar trends to those observed in the $PM_{2.5}$ readings. Specifically, the readings from all sensors initially follow a similar pattern, with deviations from this trend observed on the same date, April 15, 2023, and returning back to its normal levels on April 18, 2023. Furthermore, the median PM_{10} values for F-323 are found to be close in magnitude, while the median value for F-324 is comparatively lower.

8) Environment Indoor Air Quality Index (EIAQI): Upon analysis of the computed EIAQI for each device as presented in Fig. 63, 64, and 65, a consistent trend was observed among the devices in F-323, albeit with occasional spikes in F-323 A. Meanwhile, F-324 exhibited a distinct trend. Notably, the median EIAQI value for F-323 B was lower than that of F-323 A by a margin of 2, while F-324 had the best median EIAQI

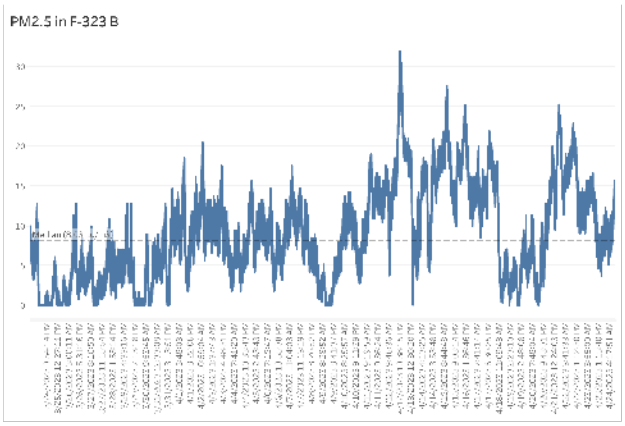

Fig. 58: $PM_{2.5}$ in F-323 B Over a Month

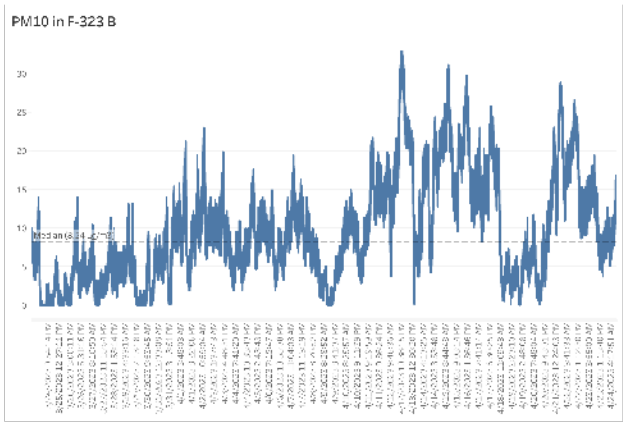

Fig. 61: PM_{10} in F-323 B Over a Month

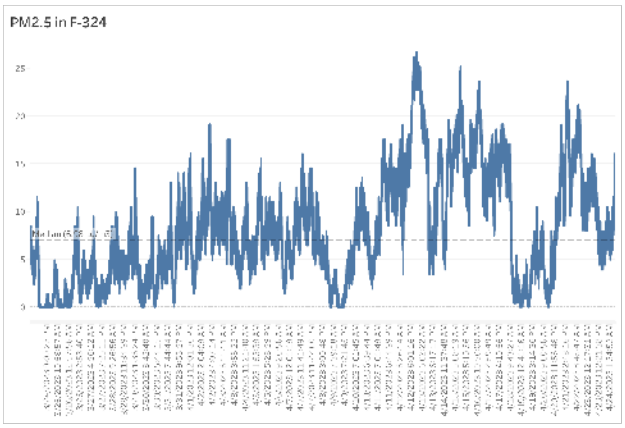

Fig. 59: $PM_{2.5}$ in F-324 Over a Month

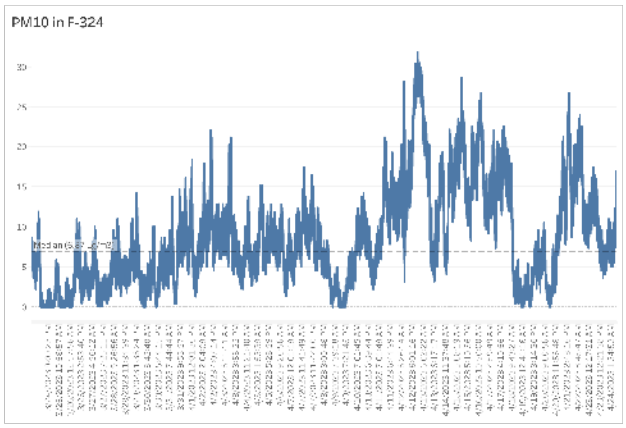

Fig. 62: PM_{10} in F-324 Over a Month

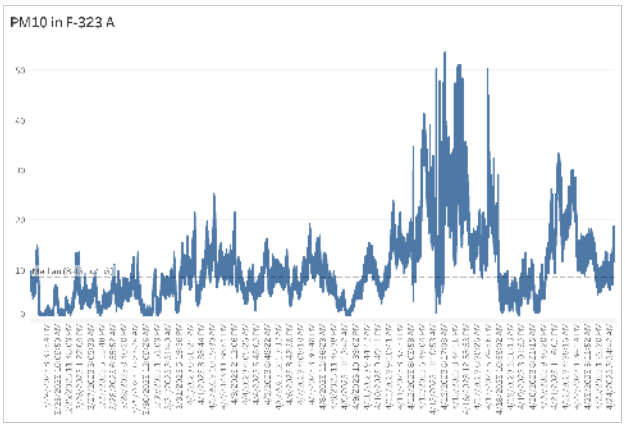

Fig. 60: PM_{10} in F-323 A Over a Month

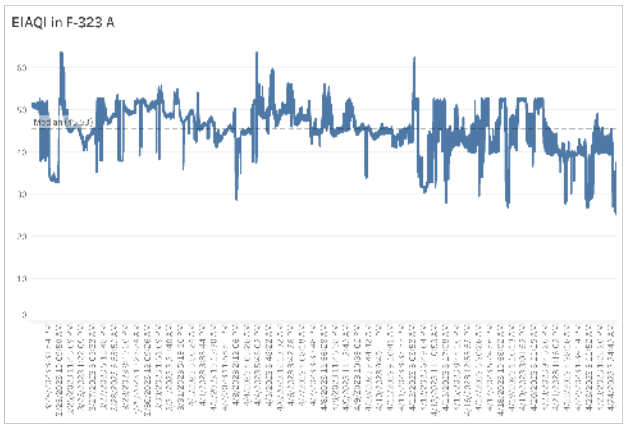

Fig. 63: EIAQI in F-323 A Over a Month

value of 50. Despite the values ranging from 40 to 60, which indicates bad indoor air quality, it is possible that the elevated average CO readings were influenced by the MQ-135's impact.

D. Other Findings

1) Power Consumption: The voltage and current consumption of the devices were measured using a USB meter, with a particular focus on the F-324 device. The results showed that each box uses approximately 5.21 V and a current of 450 mA. Furthermore, it was discovered that an IAQM device consumed 10,638 mAh over a 24-hour period, equating to an hourly consumption of approximately 443.25 mAh. Although this is a high consumption rate for a monitoring system, it falls short of the expected current of 700 mA. One reason for the elevated current draw is the MQ sensors, which consume

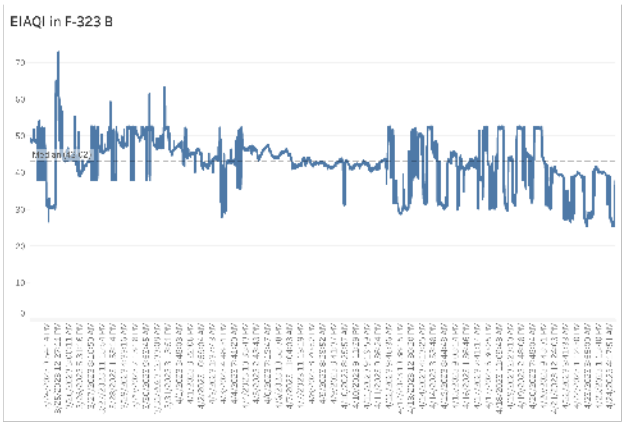

Fig. 64: EIAQI in F-323 B Over a Month

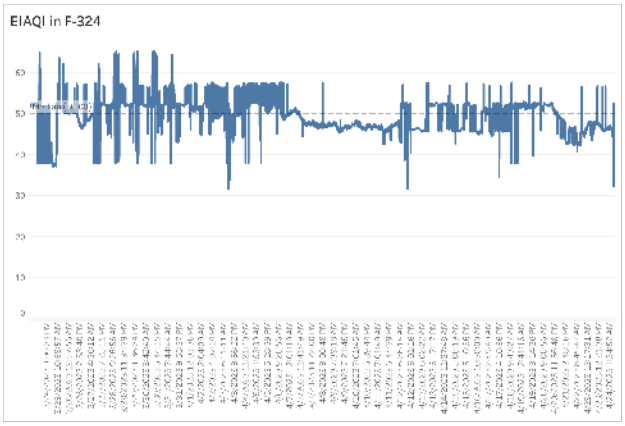

Fig. 65: EIAQI in F-324 Over a Month

100 mA or more. Another factor is the Wi-Fi functionality of the ESP-32, which uses 200 mA.

2) DHT22 Placement in IAQM Device Box: One notable finding from the collected temperature data in Fig. 45, 46, and 47 is that the recorded values are significantly higher than expected for room temperatures. Despite calibration, one possible reason for the elevated temperature values is the proximity of the DHT22 sensors to the MQ sensors, which have heating elements. To investigate this, a DHT11 sensor, calibrated similarly to the DHT22 sensors, was connected to the F-324 device and placed outside of the device's box, away from the MQ sensors. The setup for this is shown in Fig. 66. The F-324 device continued to function normally, monitoring the air quality of the room, but also had the added task of sending the DHT22 and DHT11 data to a separate Google Sheets file for analysis. Comparisons were made and a line of best fit was found to determine the correction equation.

The results presented in Fig. 67 and 68 provide confirmation that the DHT22 data are impacted by the nearby heating elements of the MQ sensors. Specifically, Fig. 67 illustrates that the temperature readings were elevated by 8.6°C due to the heating elements. In contrast, while the humidity readings of the DHT22 were also impacted, they were not affected in

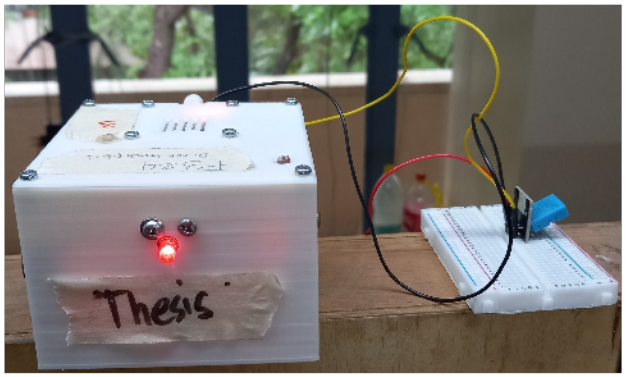

Fig. 66: Inclusion of DHT11 in F-324 Device

a similar as depicted in Fig. 68, unlike temperature.

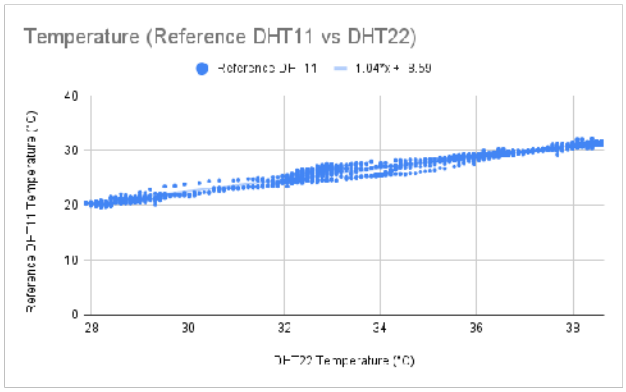

Fig. 67: Temperature Comparison Between DHT11 and DHT22 Readings in F-324

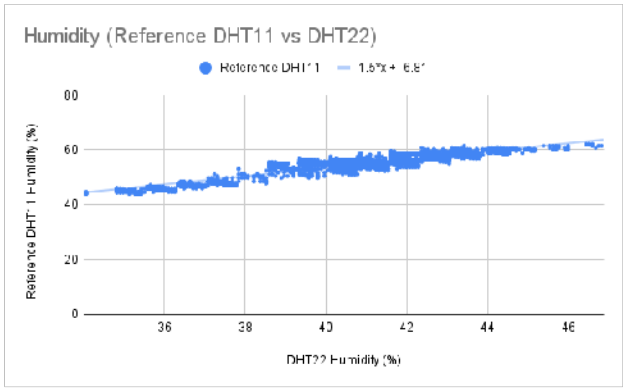

Fig. 68: Humidity Comparison Between DHT11 and DHT22 Readings in F-324

3) Use of RGB LED: Observations showed that the RGB LED module used in the system may cause confusion to individuals inside the room, as the color displayed changes depending on the person's viewing angle. As demonstrated in Fig. 69, the current color of the RGB LED appears orange when viewed from the front. However, when viewed from the sides, as depicted in Fig. 70 and 71, the color appears green.

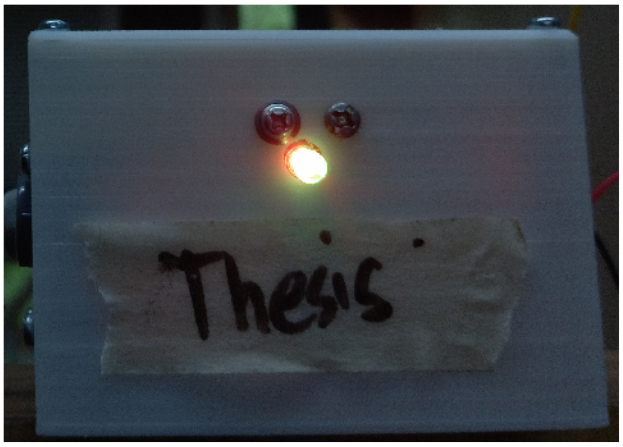

Fig. 69: Front View of RGB LED in F-324

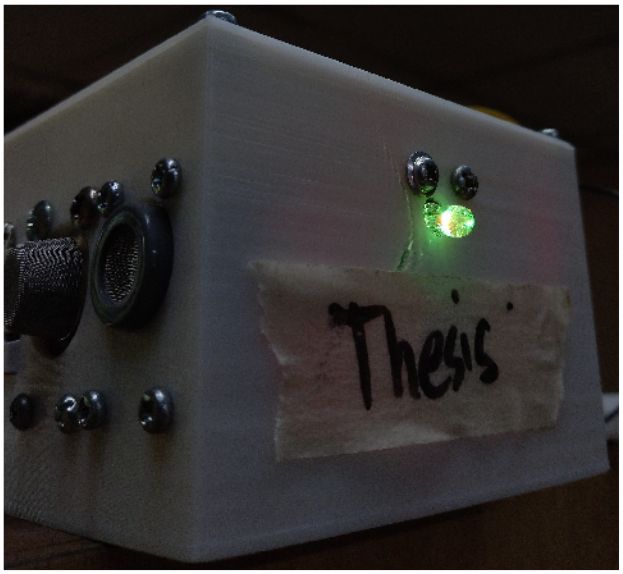

Fig. 70: Side View of RGB LED in F-324

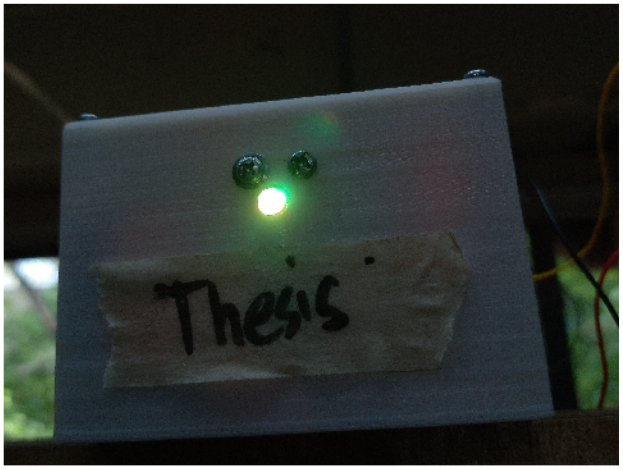

Fig. 71: Another Side View of RGB LED in F-324

V. Conclusion

Ultimately, from the design and implementation of the IoT-based indoor air quality monitoring (IAQM) devices, three products were successfully developed whose sensor readings are integrated into a single dashboard, ready for any interested party's viewing. Although the RGB LED used for the implementation cannot accurately convey the status of the environment indoor air quality index (EIAQI) due to the inconsistency of the color depending on a person's viewing angle, it is still a novel solution to inform room occupants of the indoor air quality in a non-invasive manner.

Moreover, the current fuzzy logic inference system shows an innovative approach that requires fewer sensors and fewer rules compared to other studies which aimed to accomplish similar objectives. Furthermore, this implies cost efficiency and flexibility for usage in other cloud-based developments due to the system's ability to run over Python instead of the traditional MATLAB. This not only fulfills the objective of creating a functional indoor air quality system from a software side but also approaches the problem with the goal of creating an alternative to commercial equipment, which tends to offer similar accuracy but with a higher barrier to entry.

In the pursuit of further developing this study, the following recommendations should be taken into account for future implementations:

A. Recommendations

1) Sensor Replacement: To improve the accuracy of the overall environment indoor air quality index (EIAQI) provided by the fuzzy inference system, it is recommended to replace the MQ-135 sensor with a sensor that measures other air quality parameters such as SO_x or NO_x. This will provide a more comprehensive analysis of the room's air quality, rather than relying on redundant data from the MQ-135 sensor.

2) DHT22 Readings: To address the issues with the DHT22 sensor readings, two recommendations are proposed. The first is to recalibrate the DHT22 sensor in the presence of sensors with heating elements, which can affect the sensor's accuracy. The second recommendation is to reposition the temperature humidity sensor within the monitoring device, ensuring that it is not located in close proximity to heating elements. These measures will improve the accuracy and reliability of the IAQM device's temperature and humidity readings.

3) Conveying EIAQI Value in Rooms: To enhance the usability of the IAQM device, it is recommended to explore better methods for conveying the EIAQI value to users. For instance, a user-friendly interface or display, such as an LCD or OLED screen, could be utilized to provide additional information and recommendations for improving indoor air quality. Such enhancements would improve the device's user experience and increase its usefulness for monitoring indoor air quality.

4) IAQM Device Power Consumption: The study found that the IAQM device consumes 0.42 A of current, which is lower than the expected 0.7 A. However, this current consumption remains relatively high, particularly when multiple devices are

deployed, making it a concern for scaling up the system. To reduce power consumption, it is recommended to use the deep sleep function of the ESP-32. However, this would require increasing the polling intervals, which could affect the device's real-time monitoring capabilities. Thus, a trade-off will be made between power consumption and real-time monitoring, and the optimal settings should be determined based on the specific monitoring needs and requirements of the IAQM system.

5) Temporary Storage of Sensors Data: To ensure continuous monitoring of indoor air quality, the use of an SD Card module for temporary data storage is recommended. When the IAQM device loses internet connectivity, data can be stored on the SD Card module until the connection is re-established. The stored data can then be sequentially pushed to the database, preventing any gaps in the database. This measure ensures that the IAQM device maintains accurate and continuous monitoring of indoor air quality despite Wi-Fi connection interruptions.

6) Calibration: Furthermore, the study recommends exploring better calibration methods for all sensors that measure gas concentrations. This can be achieved by using known gas concentrations to determine the sensors' actual readings, which can improve the IAQM device's overall performance and utility. Accurate calibration of gas sensors is crucial for ensuring that the IAQM device provides reliable and accurate measurements of indoor air quality.

7) Fuzzy Inference System Determining EIAQI: Although the current implementation of the fuzzy inference system showcases a reliable EIAQI to depict the status of the room, two recommendations can help improve this further. The first recommendation would be to create an improvised calibration on the categories and ranges used for the room, especially if it is a preferred choice to deploy these devices into a room permanently. This could be accomplished by installing the current system into a room and creating the most ideal environment possible for the room to achieve. By doing so, the ideal baseline could be zeroed in to shift the minimum requirements to fulfill a perfect EIAQI value of 100, then scaling it down with proper proportions. The second recommendation would be to incorporate more sensors should there be extra budget allocation to this system. This will allow better accuracy as other components can help determine a more accurate EIAQI value. Although, this will also mean adding more rules and indices to account for these changes. However, the fuzzy inference system integrated with the current hardware capabilities of the device still shows excellent results, especially if this implementation is executed in large institutions and in high volume.

8) IAQM System Data Flow: Currently, the IAQM devices send an HTTP request to Google Apps Script with the sensor readings, which then sends another HTTP request to the cloud Python script to compute the EIAQI using the scikit-fuzzy Python package. In order to streamline the data transfer processes further, the cloud Python script can serve as the sole HTTP gateway of the ESP-32 microcontroller and the Google Sheets file. Effectively, this eliminates the need to use Google Apps Script to handle HTTP requests to and from the ESP-32 and the cloud Python script. However, a drawback of this implementation is the added cost, as more computing power from the virtual machine will be needed. Thus, in this implementation, Google Apps Script was used to handle HTTP requests from the ESP-32 so that the cloud Python script can focus solely on the computation of the EIAQI from the fuzzy inference system.

9) Google Sheets as a Poor Database: As Google Sheets is designed for computing, other database management systems (DBMS) should be used, such as MySQL, that can be hosted on a virtual machine. In doing so, limitations concerning Tableau can be mitigated, improving the speed and performance of the dashboard. Moreover, using a DBMS may significantly reinforce data integrity due to their configurable access points.

10) HTTP Requests: The current implementation of the series of HTTP requests in the Google Apps Script and cloud Python script web applications implement HTTP GET methods, which require passing a fixed password as a URL parameter. Moving forward, HTTP POST methods should be used along with token encryption and time-expiry. Furthermore, its secure counterpart, HTTPS, should be used in order to prevent interference from unauthorized parties.

11) Tableau Dashboard Hosting: As Tableau Public is very limited, hosting the dashboard on Tableau Cloud should be considered in the future for improved speeds and more frequent data refreshes. Currently, the dashboard can only be refreshed by the owner of the dashboard on Tableau Public, which leaves the organization forced to grant authorization to any party interested in retrieving live data from the dashboard. Furthermore, hosting the dashboard on Tableau Cloud allows the dashboard to be accessible on Tableau's mobile applications.

REFERENCES

[1] N. C. Ganegoda, K. P. Wijaya, M. Amadi, K. K. W. H. Erandi, and D. Aldila, "Interrelationship between daily COVID-19 cases and average temperature as well as relative humidity in Germany," Sci Rep, vol. 11, no. 1, Art. no. 1, May 2021, doi: 10.1038/s41598-021-90873-5.

[2] M. T. A. Olinto, A. Garcez, G. Brunelli, F. A. Olinto, M. Fanton, and R. Canuto, "Relationship between temperature and relative humidity on initial spread of COVID-19 cases and related deaths in Brazil," J Infect Dev Ctries, vol. 16, no. 5, pp. 759–767, May 2022, doi: 10.3855/jidc.15324.

[3] H. P. L. de Medeiros and G. Girão, "An IoT-based Air Quality Monitoring Platform," in 2020 IEEE International Smart Cities Conference (ISC2), Sep. 2020, pp. 1–6. doi: 10.1109/ISC251055.2020.9239070.

[4] J. Loosová and M. Hernych, "Indoor environment monitoring as a measure to reduce epidemic spreading," in 2021 IEEE International Workshop of Electronics, Control, Measurement, Signals and their application to Mechatronics (ECMSM), Jun. 2021, pp. 1–7. doi: 10.1109/ECMSM51310.2021.9468838.

[5] W. Budiharto, E. Irwansyah, R. Dewanti, A. A. Santoso Gunawan, D. Widhyatmoko, and J. S. Sembodo, "Development of Portable Temperature and Air Quality Detector for Preventing Covid-19," in 2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI), Oct. 2021, vol. 1, pp. 47–50. doi: 10.1109/ICCSAI53272.2021.9609753.

[6] M. S. H. Sassi and L. C. Fourati, “Deep Learning and Augmented Reality for IoT-based Air Quality Monitoring and Prediction System,” in 2021 International Symposium on Networks, Computers and Communications (ISNCC), Oct. 2021, pp. 1–6. doi: 10.1109/ISNCC52172.2021.9615639.

[7] A. Longo, A. Aghazadeh Ardebili, M. Zappatore, and D. P. Mulla, “May SARS-CoV-2 be prevented by an indoor air monitoring smart data service?,” in 2021 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), Dec. 2021, pp. 365–371. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics53846.2021.00065.

[8] Sk. Vaheed, P. Nayak, P. S. Rajput, T. U. Snehit, Y. S. Kiran, and L. Kumar, “Building IoT-Assisted Indoor Air Quality Pollution Monitoring System,” in 2022 7th International Conference on Communication and Electronics Systems (ICCES), Jun. 2022, pp. 484–489. doi: 10.1109/ICCES54183.2022.9835822.

[9] A. Aggarwal, T. Choudhary, and P. Kumar, “A fuzzy interface system for determining Air Quality Index,” in 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS), Dec. 2017, pp. 786–790. doi: 10.1109/ICTUS.2017.8286113.

[10] A. T. Teologo, E. P. Dadios, R. G. Baldovino, R. Q. Neyra, and I. M. Javel, “Air Quality Index (AQI) Classification using CO and NO2 Pollutants: A Fuzzy-based Approach,” in TENCON 2018 - 2018 IEEE Region 10 Conference, Oct. 2018, pp. 0194–0198. doi: 10.1109/TENCON.2018.8650344.

[11] F. Pradityo and N. Surantha, “Indoor Air Quality Monitoring and Controlling System based on IoT and Fuzzy Logic,” in 2019 7th International Conference on Information and Communication Technology (ICoICT), Jul. 2019, pp. 1–6. doi: 10.1109/ICoICT.2019.8835246.

[12] A. Bushnag, “Air Quality and Climate Control Arduino Monitoring System using Fuzzy Logic for Indoor Environments,” in 2020 International Conference on Control, Automation and Diagnosis (ICCAD), Oct. 2020, pp. 1–6. doi: 10.1109/ICCAD49821.2020.9260514.

[13] M. Zareb, B. Bakhti, Y. Bouzid, C. E. Batista, I. Ternifi, and M. Abdenour, “An Intelligent IoT Fuzzy Based Approach for Automated Indoor Air Quality Monitoring,” in 2021 29th Mediterranean Conference on Control and Automation (MED), Jun. 2021, pp. 770–775. doi: 10.1109/MED51440.2021.9480313.

[14] T. W. Schlatter, ”Temperature-humidity index,” in Encyclopedia of Earth Science: Climatology, Springer, Boston, MA, 1987, pp. 176. doi: 10.1007/0-387-30749-4_176.

[15] J. White, A. Chan, T. Dye, and R. Wayland, ”AIRNow: Air Quality Notification and Forecasting System,” in IEEE 3rd International Conference on Environmental Science and Technology, Singapore, 2014, pp. 169-173. doi: 10.1109/ICEST.2014.7056432.

[16] B. W. Dionova, M. N. Mohammed, S. Al-Zubaidi, and E. Yusuf, ”Environment indoor air quality assessment using fuzzy inference system,” ICT Express, vol. 6, no. 3, pp. 185-194, 2020, ISSN: 2405-9595. doi: 10.1016/j.icte.2020.05.007.

[17] Espressif Systems, ”ESP32 Datasheet,” Aug. 2016, revised Jan. 2023, version 4.2. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[18] Adafruit Industries, ”Adafruit SGP30 Air Quality Sensor Breakout,” Adafruit Industries, [Online]. Available: https://www.adafruit.com/product/3709.

[19] Adafruit Industries, ”DHT11, DHT22 and AM2302 Sensors,” Adafruit Industries, [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/dht.pdf.

[20] Henan Hanwei Electronics Co., Ltd, ”MQ-7 Datasheet,” DatasheetHUB. [Online]. Available: https://datasheethub.com/wp-content/uploads/MQ7.pdf.

[21] Olimex Ltd., ”Use introduction of HS-129 type gas sensitive components,” Olimex Ltd., [Online]. Available: https://www.olimex.com/Products/Components/Sensors/Gas/SNS-MQ135/resources/SNS-MQ135.pdf.

[22] Zhou Yong, ”PMS7003 Datasheet,” Jun. 2016. [Online]. Available: https://www.espruino.com/datasheets/PMS7003.pdf.

[23] ”KY-016 RGB Full Color LED Module,” ArduinoModules.info, [Online]. Available: https://arduinomodules.info/ky-016-rgb-full-color-led-module/.

[24] Adafruit, ”DHT-sensor-library,” GitHub, 2011. [Online]. Available: https://github.com/adafruit/DHT-sensor-library.

[25] Adafruit, ”Adafruit SGP30,” GitHub, 2017. [Online]. Available: https://github.com/adafruit/Adafruit_SGP30.

[26] M.A. Califa Urquiza et al., ”miguel5612/MQSensorsLib: 3.0.0 (v3.0.0),” Zenodo, 2022. doi: 10.5281/zenodo.6374212.

[27] M. Kacki, ”fu-hsi/pms,” GitHub. [Online]. Available: https://github.com/fu-hsi/pms.

[28] Adafruit. ”Adafruit SGP30 Gas TVOC eCO2 MOX Sensor,” Adafruit Learning System. [Online]. Available: https://learn.adafruit.com/adafruit-sgp30-gas-tvoc-eco2-mox-sensor/arduino-code.