IoT Air Monitoring and Reporting System for Residential Environments

Farah Shaheen[#1]

M.Sc in Artificial Intelligence Systems, University of Bahrain
[1]20175966@stu.uob.edu.bh

***Abstract*—The impact of indoor air quality is a growing matter of concern due to its effects on the health, comfort and well-being of humans worldwide. People spend 80-90% of their routine time indoors [6], so IAP can directly impact the health and productivity of people worldwide. Sources of indoor air pollution include burning coal or charcoal, fireplaces, and inadequate ventilation in HVAC systems [1].**

This project aims to provide indoor air quality monitoring for residential environments using the Internet of things (IoT), with the main objective of raising awareness to the risks caused by poor indoor air quality. Furthermore, this paper provides a review of two microcontrollers used for system design and challenges faced in the development of real time monitoring systems. This paper also presents a time series forecasting algorithm inorder to provide residents of the indoor environment a forecasting of their air quality and take further action when necessary.

***Keywords*— IAQ (Indoor Air Quality), IAP (Indoor Air Pollution), Microcontrollers, IoT, Sensors**

I. Introduction

Indoor air pollution is a global concern, but often overshadowed by the emphasis placed on outdoor air pollution. People spend 80- 90% indoors, either in offices, homes or hospitals, making it necessary to take immediate action to improve air quality indoors [6].

Indoor air quality refers to the quality of air in residential environments such as homes, hospitals, and office spaces. Many studies confirm that indoor air pollution can be more deadly than outdoor air pollution [7].

Research has shown that 90% of rural households in developing countries rely on unprocessed biomass for open fires for heat and cooking. Burning coal and biomass can release harmful pollutants such as Particulate matter (PM), Nitrogen Dioxide (No2), Carbon Monoxide (CO), Sulfur Oxides, formaldehyde, Carbon Dioxide (CO2) and others [8].

These methods of generating heat are responsible for high levels in indoor air pollution, and causing health issues that are deadly, especially to vulnerable individuals such as children and elders [8]. High levels of IAP can cause chronic obstructive pulmonary disease, acute respiratory infections, asthma, lung cancer, low birth rate, perinatal conditions, severe eye diseases causing blindness, and many more. [9].

According to the World Energy Outlook in 2017, even after implementing improvements in cooking methods in developing countries by using solar and wind energy instead of fuel combustion, 1.3 billion people in Asia are still expected to rely on biomass for cooking by the year 2030 [10].

People residing in third-world countries are not the only ones affected, first world-countries are also affected by IAP. The use of indoor heating and cooling systems instead of natural ventilation systems without proper ventilation can lead to poor indoor quality. Nowadays buildings are airtight and rely on HVAC systems. Among the various health risks associated with poor indoor air quality, Sick Building Syndrome (SBS) stands out as a significant concern. It happens when people in a building start feeling unwell but their symptoms often disappear when leaving the affected building, proving the direct impact of indoor environments on health. SBS refers to a set of symptoms caused by poor air quality, including headaches, dizziness, nausea, and eye, nose, or throat irritation. SBS showcases the importance of monitoring and improving air quality in residential and commercial buildings [13].

Sources of these dangerous pollutants are from multiple sources , including but not limited to burning coal, fireplaces, and inadequate ventilation in HVAC systems has been identified as significant contributors to indoor air pollution, along with common household products such as insect repellent sprays, rubbing alcohol and butane.

The table below demonstrates the major pollutants affecting the quality indoors.

Table 1 Major pollutants affecting the quality indoors [6]

Pollutants	**Emissions**	**Health Effects**
CO	Stoves, kerosene, unvented kerosene, water heaters, boilers, incomplete combustion of carbon-containing fuels	Increase in perinatal death,
CO2	Burning butane, fumes from stovetop or oven, burning of oil, coal and gas	Headaches, sleepiness, poor concentration
Asbestos	Fire retardant materials, insulation	Cancer, pleural thickening,
NO2	Fuel burning	Asthma, wheezing, reduced lung function in children, infections in respiratory organs
Lead	Paint, soil, firearms	Memory loss, hearing loss, high blood pressure, kidney and heart disease
VOCs	Gas burning, cleaning products, hair spray, perfumes, tobacco smoke	Allergic reactions, memory impairments, damage to the central nervous system, kidney and liver failure.

By leveraging the capabilities of the Internet of Things (IoT) and contributing to monitoring of air quality, this article presents a solution based on IoT using low cost sensors to measure pollutants in the atmosphere. The solution also provides an integration through the cloud for real-time monitoring using Arduino Cloud . This article will also present an effective way to utilize the data collected from the monitoring system by employing predictive analytics to anticipate IAQ concerns.

The significance of this study is to raise awareness of the potential health hazards caused by poor indoor air quality and to provide users with the tools to take informed actions. The paper will be organized as follows: Section II will provide a literature review that will explain various existing IAQ systems. Section II will provide the System Architecture , Section IV is the implementation and Section V will present the machine learning algorithm.

II. Literature Review

The significance of Internet of things technologies has impacted the field of IAQ monitoring systems. This literature review will provide a high-level overview of various different implementations and studies on IAQ monitoring systems with a focus on the technologies and sensors used.

The work developed by the paper titled “Internet of Things Mobile- Air Pollution Monitoring System (Iot-Mobair)”, proposed a three-phased air pollution monitoring system, integrating gas sensors, Arduino IDE, and a Wi-Fi module. The system also includes an application called IoT-Mobair for accessing the air quality data for the cloud. One distinguishing fact of this paper compared to others is that they provided an algorithm to ensure all the sensors are working as expected and the values are being retrieved correctly. The application also provides predictions on the pollution levels and warning for high pollution levels according to the user routes [3].

Another research project titled “Building IoT-Assisted Indoor Air Quality Pollution Monitoring System” , developed an IoT air quality monitoring system using Raspberry Pi as a processor and communicator to the cloud, and a ZPHS01B multi-channel sensor developed by Winsen Electronics. According to the authors, it is a single module that has the following sensors: gas, CO2, Electrochemical formaldehyde to measure formaldehyde in the air, along with ozone and CO sensors [4]. This study highlighted the increased dangers of indoor air pollution compared to outdoor pollution, especially post-COVID-19, as people spend more time indoors. One of the drawbacks in this paper is the lack of explanation on the energy efficiency and power requirements needed to operate their proposed IAQ system.

A battery operated and portable IoT based IAQ system was implemented by Sivash Esfahani..et al

which is a low-cost monitoring system featuring a 30-hour battery life and used BME680 temperature and humidity sensor , CC81 sensor, SCD30 carbon dioxide sensor, SPS30 particulate matter and VEML7700 sensor for light.
The paper also states that the device can also be powered by USB cable, which charges the batteries simultaneously [5]. The authors tested the device effectiveness by conducting test experiments in different environments such as cooking in a kitchen and a full computer laboratory, which are ideal environments for testing IAP.

Lastly, a paper by Ladekar et al implemented an IoT IAQ monitoring device using Raspberry Pi , ESP8266 for monitoring O2 , CO2, and PM2.5 concentrations. The system is connected to the ESP8266, which is connected to the sensors via an electrical circuit board. [11]. The data is then collected and sent via the Raspberry pi to the cloud. The raspberry pi in this implementation is used as an MQTT broker. The system, along with the other analyzed paper has the goal of creating an indoor air quality monitoring system. However, there are some unjustified steps taken. The Raspberry pi is not justified as the data can be sent directly to the ESP 8266 to the cloud.

III. System Architecture

A. Step 1: IoT Architecture and System Design

The system is a multi-tier architecture which includes sensors for data collection, a microcontroller and a cloud-based platform for data processing and user interaction. This architecture based on the IoT architecture model is three-layered, with a perception, transport, processing and application layer.

Application Layer	Arduino Cloud real time monitoring Dashboard
Network Layer	Between the communication and sensing layer using Wi-Fi.
Perception Layer/ Sensing Layer	Sensing the physical parameters via MQ-2, MQ-9 , MQ-135 and DHT11 sensors

Fig. 1 Four-layered IoT Architecture for IAQ Monitoring System

B. Step 2: Hardware Implementation

The implementation model of the IAQ monitoring system first attempt began with an ESP8266 microcontroller. Due to its low cost and easy availability, this microcontroller was chosen to proceed with the implementation, however there were significant limitations encountered throughout the planning. Firstly, there was insufficient number of ports to accommodate multiple gas sensors simultaneously which was a crucial requirement for a comprehensive monitoring system as the objective of this research was to detect a wide range of chemicals found indoors.

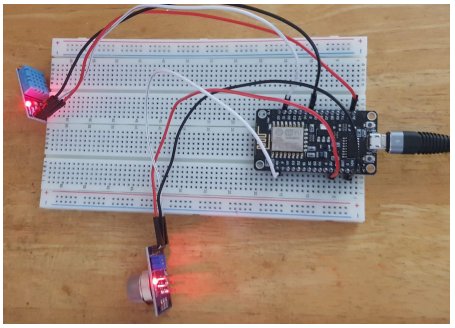

Fig. 2 Hardware Setup of ESP8266 with MQ-2 and DHT11 Sensors

To address this limitation, the implementation was switched to using an ESP32 microcontroller. The ESP32 GPIO (General Purpose Input/Output) ports allowed for the flexibility of connecting to multiple sensors simultaneously while also offering high processing power and dual-core functionality, providing better real-time analysis which is essential for an IAQ monitoring system.

In order to detect a broad range of pollutants found indoors, the array of sensors used are: MQ-2, MQ-9, MQ-135 and DHT11 sensors.
The MQ-2 sensor is selected for its high sensitivity to smoke, butane and other flammable gases and steam, and will be used in this project to detect smoke.
The MQ-9 gas sensor module is capable of detecting potential gas leakage, liquefied petroleum gas (LPG), carbon monoxide (CO), methane (CH4), as it has high sensitivity to LPG, this sensor will be used for detecting LPG indoors.
The MQ-135 gas sensor module is used to detect carbon monoxide (CO), carbon dioxide (CO2),

ammonia (NH3), sulfur dioxide (SO2) and benzene; this implementation will use MQ-135 for sensing alcohol and acetone.
The DHT11 is used for measuring temperature and humidity, which provides essential information to understand the environmental conditions influencing the IAQ.

Table 2 High sensitivity for each gas sensor

Sensor	High Sensitivity
MQ-135	Alcohol. Acetone
MQ-2	Smoke, liquified natural gas, propane
MQ-9	LPG, Gas leakage detection, CO

Table 2 showcases the sensors that will be used in the implementation and the chemicals they are highly sensitive to.

C. Step 3: Software Components

The Arduino IDE is used to write , compile and upload the code to the microcontroller, which dictates how the MC interacts with the various sensors. Within the code, the variables are defined for each sensor and with a pin number to which the gas and DHT11 sensors are connected to on the microcontroller. Then the connection is established from the IDE to allow communication to the Arduino Cloud.
To collect the readings of all the sensors, IF THIS THEN THAT (IFTTT) is used to automate the date collection process. IFTTT is a software platform that provides connections to applications via webhooks. By setting up a webhook from the Arduino Cloud things to the IFTTT Applet the data readings are efficiently stored into Google sheets.

D. Step 4: Calibration Sensors

Calibration of these sensors is a critical step in the methodology, which involves setting the baseline readings in a controlled environment. The sensors were left in a neutral environment for 12 hours to establish a baseline reading. After the 12 hours is complete, the sensors were exposed to different chemicals to trigger the sensors and show higher readings and ensure the sensors revert back to the baseline setting.

To establish the threshold, each sensor was exposed to substances that it was highly sensitive to and alert the sensors to the highest point of exposure. This included smoke for MQ-2, alcohol for MQ-135, and butane for MQ-9.
The purpose of establishing the threshold is to set the specific point in which the sensor will trigger an action or alert when exposed to gas concentrations above that level.

IV. Implementation

A. Hardware Setup

The core of the hardware is the ESP32 microcontroller, chosen for its high processing capability and multiple GPIO ports. Sensors include the MQ-2, MQ-9, and MQ-135 along with the DHT11 sensor. Each sensor's output is linked to specific pins on the microcontroller for data transmission.

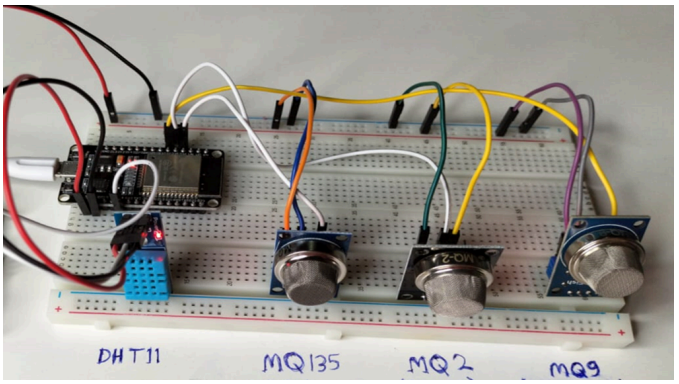

Fig. 3 Hardware Setup Illustrating the ESP32 with the four sensors

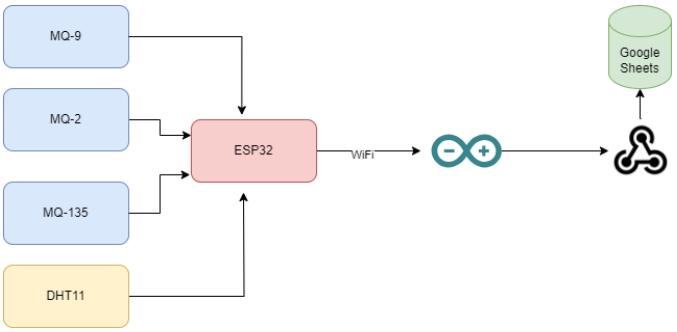

Fig. 4 Block diagram of the IoT system

B. Software Implementation

In the software implementation of the IAQ monitoring system, the Arduino IDE is used to program and configure the microcontrollers. The code begins with the necessary libraries such as “DHT11.h” and “DHTesp.h" for the DHT11

sensors, and “MQUnifiedSensor.h” for interfacing with the MQ series gas sensors. The ‘thingroperties.h’ library is required for managing the variables in the Arduino Cloud and allowing users to view the sensors via the dashboard from gauges or graphs.

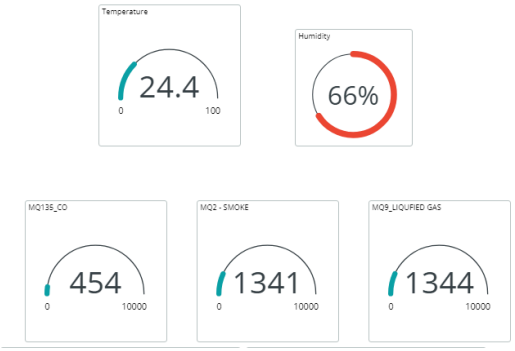

Fig.4 Arduino Cloud Real-time Monitoring Dashboard

C. *Data Collection and Real-time Monitoring*

A significant part of this paper is the data collection and analysis process. IFTTT applets were utilized to connect to the arduino cloud things via a webhook to collect data in a google sheets file. Meanwhile, users are also able to view real-time updates from the Arduino Cloud dashboard.

The dashboard provides immediate observation and indication to the changes when the sensors are exposed to chemicals they are sensitive to.

V. Testing and Validation

To ensure the data readings are accurate and the sensors are reacting to the environment properly, various testing cases were conducted.

The first testing case was exposing the MQ-135 , MQ-9, MQ-2 sensors to various different test subjects. Table 3 shows the sensitivity for each sensor against different testing subjects.

Table 3 Sensor Readings Against Different Testing Subjects

Pollutant	**Sensors**		
	MQ-2	**MQ-9**	**MQ-9**
Butane	4095	4095	1234
Vape Smoke	1383	1500	170
Incense Smoke	1776	2065	169
Insect Repellent Spray	2441	1473	4095
Alcohol	3319	4095	4095
Fire	2533	1917	300

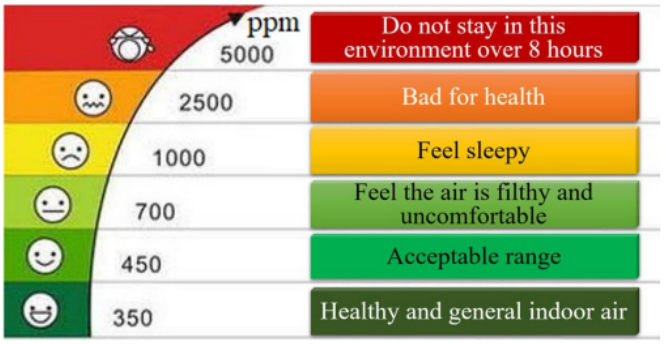

Fig. 5 IAQ Index for CO2 concentrations and the effects on the human body [12]

A. *Test Case #1: MQ-2 Smoke Sensor Testing*

Testing is done by exposing the sensor to large amounts of smoke to trigger the threshold.

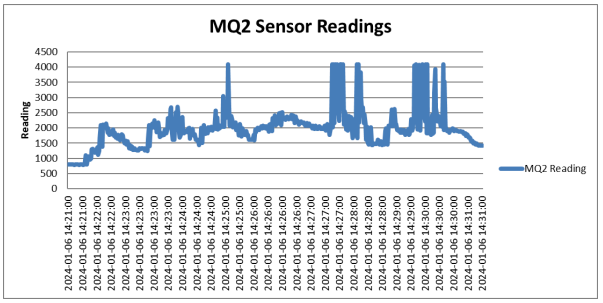

Fig. 6 MQ-2 sensor detecting smoke

B. *Test Case #2: MQ-135 Alcohol & Acetone*

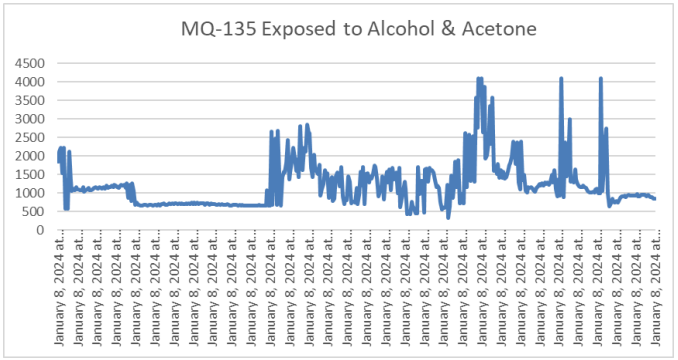

Fig. 7 MQ-135 sensor detecting alcohol & acetone

To prove that the MQ-135 sensor does not detect smoke, test case three was conducted, showing that the sensor readings are close to the baseline, indicating a healthy environment presented in Figure 8.

C. *Test Case #3: MQ-135 to Smoke*

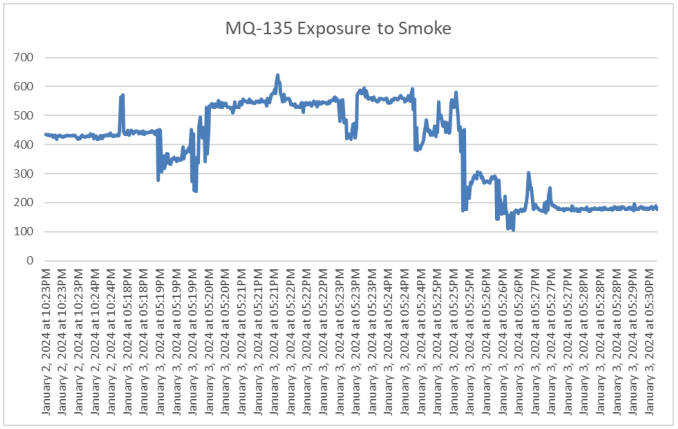

Fig. 8 MQ-135 sensor detecting smoke

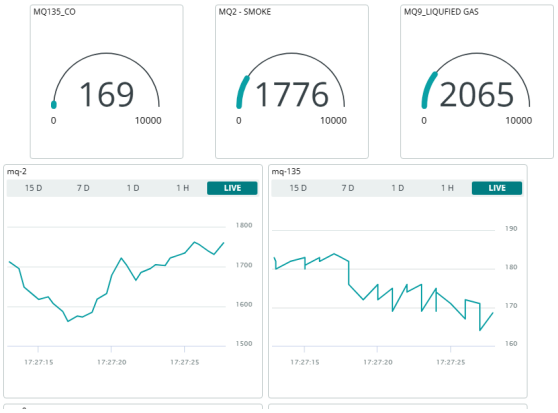

Fig. 9 MQ-135 sensor readings shown in Arduino Cloud

D. *Test Case #4: MQ-9 Liquefied Gas (butane) exposure*

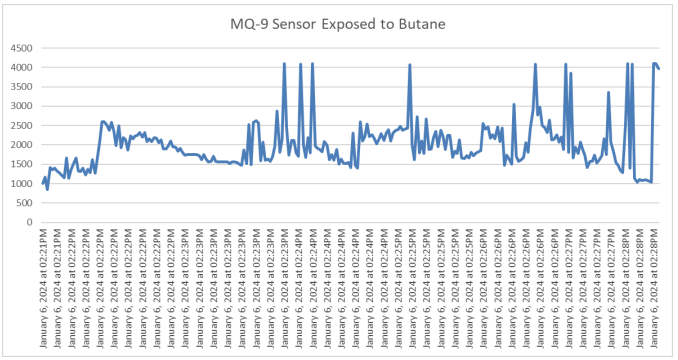

Fig. 10 MQ-9 sensor readings liquified gas

E. *Test Case #5: Temperature and humidity sensor exposed to high levels of smoke*

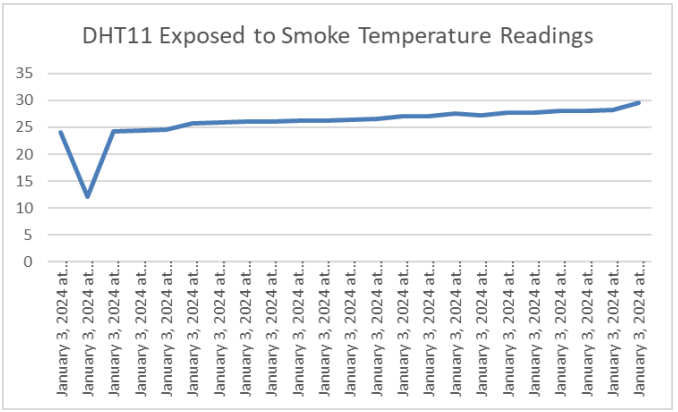

Fig. 11 DHT11 Temperature Readings

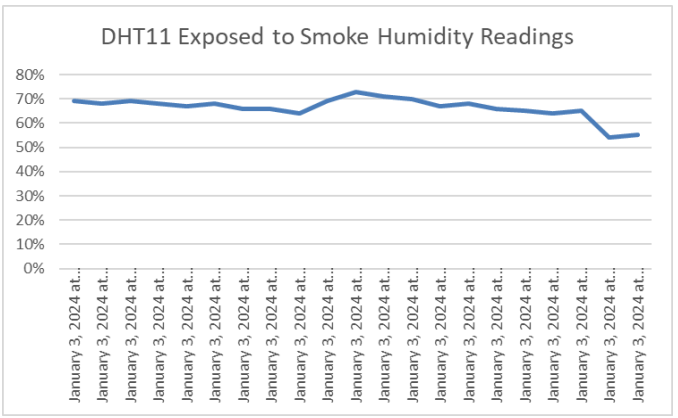

Fig. 12 DHT11 Humidity Readings

VI. Time Series Forecasting Algorithm

Long Short-Term Memory Model (LSTM)

To perform the time series forecasting algorithm, the LTSM model is used to learn from the sequences of the data. This model is a type of recurrent neural network that is best suited for time series forecasting as it has the ability of remembering previous data inputs for a longer period of time.

The dataset used for this algorithm is the MQ-135 sensor reading, collected over a period of four days.

Prior to using the LSTM model, the preprocessing stage was conducted which includes normalization of the sensor readings to a range of 0 to 1, splitting the dataset to 70/30 for training and testing.

Results of Time Series Forecasting

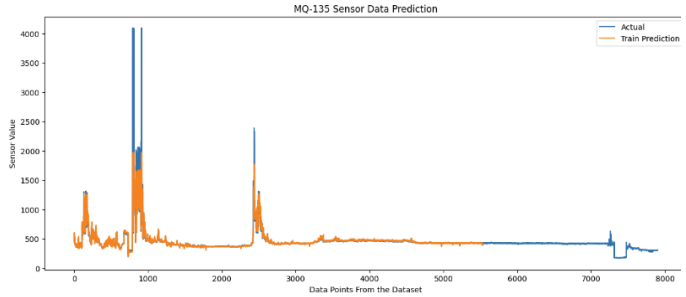

Fig. 13 Results of the forecasting and the actual data readings

The goal of this algorithm is to provide an efficient method of using the dataset collected from the implemented IoT system, along with providing a solution to residents in indoor environments with a forecasting of the IAQ and take further action whenever necessary. Providing a forecast of the air quality can help save lives along with avoiding impact on human health.

RMSE (root mean squared error) is used as an error measurement to provide the metric and evaluate the performance ensuring the accuracy of the predictions. The algorithm is flexible, and can be adapted to different types of sensor readings.

VII. Results and Discussion

The purpose of this implementation is to allow people to keep IAQ at a safe level and help reduce the effect of IAP on human health by allowing people to take the measures necessary prior to reaching critical levels of high pollutants in a residential environment. The implementation of the IoT system involves a full lifecycle from hardware to software to using cloud solutions to provide real time monitoring. The sensors used are MQ-2, MQ-9, MQ-135 and DHT11 to detect a wide range of indoor pollutants like CO, CO2, LPG. The sensors are connected to an ESP32 microcontroller and via WiFi, Arduino cloud is used to read the sensors and provide a real time monitoring dashboard.

Limitations

One of the limitations faced in the planning and implementation phase is the ability to ensure the COM ports were set up properly of the microcontrollers to the arduino agent. It took multiple tries to ensure the connection was stable.

Another limitation was with the microcontroller chosen at the initial implementation stage. The microcontroller ESP8266 has limited ports to connect to multiple MQ sensors. It limited the scope of identifying more pollutants in the air. To resolve the issue, an ESP32 was used instead.

VIII. Conclusions

In conclusion, IAP is significantly impacting human health. There are numerous pollutants found indoors that are caused from inadequate ventilation systems or simply by performing daily household activities such as cleaning or cooking. Although these pollutants are found in lower concentrations indoors, having long-term exposure to them can cause significant health issues and cause many diseases and at times lead to death [14].

This research was conducted to highlight the critical need for an IAQ monitoring system, and with the purpose of raising awareness about the risks associated with poor IAQ. The paper also provides a solution to allowing residents at home to have control over their environment prior to the IAQ reaching critical levels using a forecasting algorithm. Future work can explore integrating more advanced sensors to enhance data processing and expand on detecting different pollutants found indoors.

References

[1] J. Esquiagola, M. Manini, A. Aikawa, L. Yoshioka and M. Zuffo, "Monitoring Indoor Air Quality by using IoT Technology," 2018 IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON), Lima, Peru, 2018, pp. 1-4, doi: 10.1109/INTERCON.2018.8526380.

[2] Tran, V. V., Park, D., & Lee, Y. C. (2020). Indoor Air Pollution, Related Human Diseases, and Recent Trends in the Control and Improvement of Indoor Air Quality. International Journal of Environmental Research and Public Health, 17(8), 2927. https://doi.org/10.3390/ijerph17082927

[3] Dhingra, S., Madda, R. B., Gandomi, A. H., Patan, R., & Daneshmand, M. (2019). Internet of things mobile–air pollution monitoring system (IOT-mobair). IEEE Internet of Things Journal, 6(3), 5577–5584. https://doi.org/10.1109/jiot.2019.2903821

[4] Vaheed, Sk., Nayak, P., Rajput, P. S., Snehit, T. U., Kiran, Y. S., & Kumar, L. (2022). Building IOT-Assisted Indoor Air Quality Pollution Monitoring System. 2022 7th International Conference on Communication and Electronics Systems (ICCES). https://doi.org/10.1109/icces54183.2022.9835822

[5] Esfahani, S., Rollins, P., Specht, J. P., Cole, M., & Gardner, J. W. (2020). Smart City Battery operated IOT based Indoor Air Quality Monitoring System. 2020 IEEE SENSORS. https://doi.org/10.1109/sensors47125.2020.9278913

[6] Saini, J., Dutta, M. & Marques, G. A comprehensive review on indoor air quality monitoring systems for enhanced public health. Sustain Environ Res 30, 6 (2020). https://doi.org/10.1186/s42834-020-0047-y

[7] Cincinelli A, Martellini T. Indoor air quality and health. Int J Environ Res Pu. 2017;14:1286.

[8] Saini, J., Dutta, M. & Marques, G. A comprehensive review on indoor air quality monitoring systems for enhanced public health. Sustain Environ Res 30, 6 (2020). https://doi.org/10.1186/s42834-020-0047-y

[9] Ezzati M, Kammen DM. Quantifying the effects of exposure to indoor air pollution from biomass combustion on acute respiratory infections in developing countries. Environ Health Perspect. 2001;109:481–8.

[10] IEA. World Energy Outlook 2017. Paris: International Energy Agency; 2017.

[11] Ladekar and R. Daruwala, "Indoor Air Quality Monitoring on AWS Using MQTT Protocol", 2019 10th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1-6, 2019.

[12] Sung, W.-T., & Hsiao, S.-J. (2021). Building an indoor air quality monitoring system based on the architecture of the Internet of Things. EURASIP Journal on Wireless Communications and Networking, 2021(1). doi:10.1186/s13638-021-02030-1

[13] A. E. Igwe, A. A. Ezeobi, F. O. Okeke, E. O. Ibem, and E. C. Ezema, "Causes and remedies of sick building syndrome: a systematic review," in Proceedings of the ICECAE 2023, E3S Web of Conferences, vol. 434, p. 02007, 2023. https://doi.org/10.1051/e3sconf/202343402007

[14] Tran, V. V., Park, D., & Lee, Y.-C. (2020). Indoor air pollution, related human diseases, and recent trends in the control and improvement of Indoor Air Quality. International Journal of Environmental Research and Public Health, 17(8), 2927. https://doi.org/10.3390/ijerph17082927

Appendix A
Arduino IDE Code Snippets

```
#include "thingProperties.h"
int gas_sensor = 32;
#include <DHT.h>
#define DHTPIN D4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  // Initialize serial and wait for port to open:
  pinMode(gas_sensor, INPUT);
  Serial.begin(9600);
  dht.begin();
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is found
  delay(1500);
  // Defined in thingProperties.h
  initProperties();
  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
```

Fig. A.1 Packages used in Arduino IDE

Appendix B
Time Series Forecasting Algorithm Code Snippets

```
import numpy as np
import matplotlib.pyplot as plt
# Data Manipulation
import pandas as pd
from datetime import datetime
# Scaling the data
from sklearn.preprocessing import MinMaxScaler

#For building the neural network model
from keras.models import Sequential
from keras.layers import Dense, LSTM

#For evaluating model performance
from sklearn.metrics import mean_squared_error
```

Fig. B.1 Packages used for Time Series Algorithm

```
# Split into training and test sets
train_size = int(len(scaled_data) * 0.70)
test_size = len(scaled_data) - train_size
train, test = scaled_data[0:train_size, :], scaled_data[train_size:len(scaled_data), :]
```

Fig. B.2 Splitting the Data for Training and Testing

```
# Convert an array of values into a dataset for LSTM
def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset) - look_back - 1):
        a = dataset[i:(i + look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    return np.array(dataX), np.array(dataY)
```

Fig. B.3 Converting array values to a dataset

```python
# Training the Model
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(trainX, trainY, epochs=15, batch_size=1, verbose=2)
```

```
Epoch 1/15
5530/5530 - 11s - loss: 0.0025 - 11s/epoch - 2ms/step
Epoch 2/15
5530/5530 - 9s - loss: 0.0015 - 9s/epoch - 2ms/step
Epoch 3/15
5530/5530 - 10s - loss: 0.0014 - 10s/epoch - 2ms/step
Epoch 4/15
5530/5530 - 9s - loss: 0.0013 - 9s/epoch - 2ms/step
Epoch 5/15
5530/5530 - 9s - loss: 0.0013 - 9s/epoch - 2ms/step
Epoch 6/15
5530/5530 - 11s - loss: 0.0013 - 11s/epoch - 2ms/step
Epoch 7/15
5530/5530 - 10s - loss: 0.0012 - 10s/epoch - 2ms/step
Epoch 8/15
5530/5530 - 9s - loss: 0.0012 - 9s/epoch - 2ms/step
Epoch 9/15
5530/5530 - 10s - loss: 0.0012 - 10s/epoch - 2ms/step
Epoch 10/15
5530/5530 - 9s - loss: 0.0012 - 9s/epoch - 2ms/step
Epoch 11/15
5530/5530 - 9s - loss: 0.0012 - 9s/epoch - 2ms/step
Epoch 12/15
5530/5530 - 9s - loss: 0.0012 - 9s/epoch - 2ms/step
Epoch 13/15
5530/5530 - 10s - loss: 0.0012 - 10s/epoch - 2ms/step
Epoch 14/15
5530/5530 - 9s - loss: 0.0011 - 9s/epoch - 2ms/step
Epoch 15/15
5530/5530 - 9s - loss: 0.0011 - 9s/epoch - 2ms/step
<keras.src.callbacks.History at 0x7b73f51c20b0>
```

Fig. B.4 Perform data fitting using 15 Epoches

```python
# Calculate root mean squared error
trainScore = np.sqrt(mean_squared_error(trainY[0], trainPredict[:, 0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = np.sqrt(mean_squared_error(testY[0], testPredict[:, 0]))
print('Test Score: %.2f RMSE' % (testScore))
```

```
Train Score: 130.63 RMSE
Test Score: 27.55 RMSE
```

Fig B.5 Calculating root mean squared with results

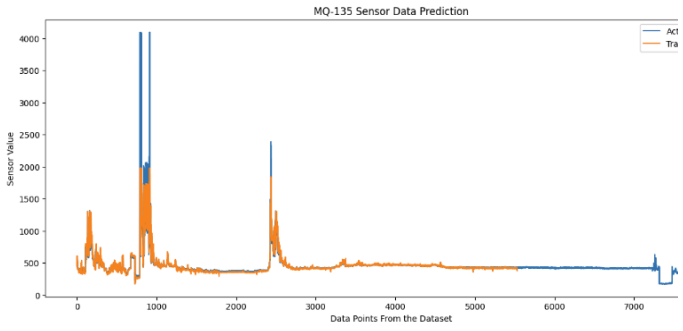

Fig B.6 Forecasting Result