Indoor Air Quality Monitoring using IoT

***Abstract*—Wildfires lead to the production of an enormous amount of smoke, consisting of a mixture of toxic gases and fine particles. Large wildfires can pollute numerous cities in its vicinity. While this makes outdoor air unhealthy to breathe, the smoke from outdoors can enter our home and make it harmful to breathe indoor air as well. The advent of COVID-19 has forced us to spend most of our time at home. This has fostered the need to improve indoor air quality. This paper proposes a solution for users to analyze the high concentration of Particulate Matter (PM2.5) and Carbon Monoxide (CO) indoors, using a real-time Internet of Things (IoT) system.**

***Index Terms*—Wildfires, PM2.5, CO, IoT**

I. Introduction

Smoke pollution created from uncontrolled wildfires, can have profound health impacts. [12] The major health threat of smoke is from fine particles, for instance, PM2.5. These microscopic particles can enter your eyes and respiratory system, causing health issues such as burning eyes, runny nose, and bronchitis. They can also intensify chronic heart and lung diseases. [15] Smoke constitutes a mixture of toxic gases, mainly CO, when breathed can cause headache, fatigue, dizziness, drowsiness, or nausea. [11] Large wildfires can spread cross numerous cities polluting both indoor and outdoor air quality. This suggests that cities in the vicinity of wildfires suffer from poor indoor air quality.

The arrival of COVID-19 has compelled us to be confined at home. [1] In this scenario, it is necessary to make sure that we are aware of the quality of air we breathe indoors. The Internet of Things (IoT) has proved to be a cost-effective and efficient solution to measure and study indoor air and its quality.

In this paper, we have proposed an Indoor Air Quality Monitoring system using IoT. We have created a Level 6 wireless sensor network as shown in Fig.1. It consists of two independent monitoring nodes. The first node constitutes a NodeMCU connected to a PPD42NJ dust sensor to measure the concentration of fine Particulate Matter, PM2.5. The second node constitutes a NodeMCU connected to a MQ9 gas sensor to measure the concentration of Carbon Monoxide, CO. The NodeMCU is a low-cost microcontroller which uses an open source IoT platform called Arduino IDE. This platform can mold the NodeMCU to perform different IoT applications. It consists of a built-in Wi-Fi module for networking. The readings from each of these nodes is relayed to a Cloud database using a 2.4GHz cellular network. ThingSpeak, the Cloud database used in our system is an open-source software. It facilitates logging, access, and analysis of data measured by the end nodes.

The fundamental goal of the system is to create awareness of the polluted air inhaled in indoor environments. The PM2.5 and CO readings are logged into ThingsSpeak, we have set a threshold level for PM2.5 and CO in ThingSpeak. If the concentration level of any of the two inputs goes beyond the set threshold, the user of this system will receive an email alert informing them about the obnoxious air they are breathing.

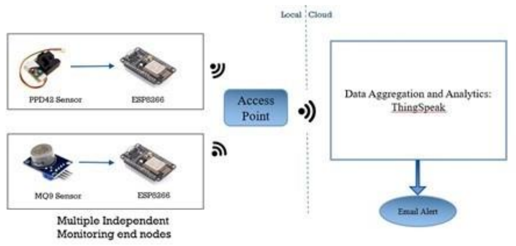

Local
Cloud
PPD42 Sensor
ESP8266
MQ9 Sensor
ESP8266
Multiple Independent Monitoring end nodes
Access Point
Data Aggregation and Analytics: ThingSpeak
Email Alert

Fig. 1. IoT level 6 System Architecture

II. Hardware Experiment Description

This section gives a brief description of the sensors used in our Indoor Air Quality Monitoring System.

A. PPD42NJ-Dust Sensor for PM2.5

Fig.2 shows a typical PPD42NJ Dust Sensor. It is used to measure the concentration of particulate matter (PM2.5). Heater (resistor) generates an updraft for the air to move into the sensor. Infrared light beam from the LED is focused with a lens to a sensing point at the center. Airborne particles have been taken into the sensor box with the updraft. Particle passing through sensing point scatters light, and receptor receive scattered light through the lens and transforms it into pulse signal. Pulse per unit time is proportional to the particle concentration. [13]

Fig.3 shows the node consisting of PPD42NJ connected to NodeMCU. The microcontroller reads the sensor data from pin 2 of the sensor and processes using pin D5 of the NodeMCU. This concentration can be mathematically calculated using Arduino IDE programmed in the NodeMCU to give us data. PM2.5 beyond a concentration level of 35 μg/m3 is dangerous. [2]

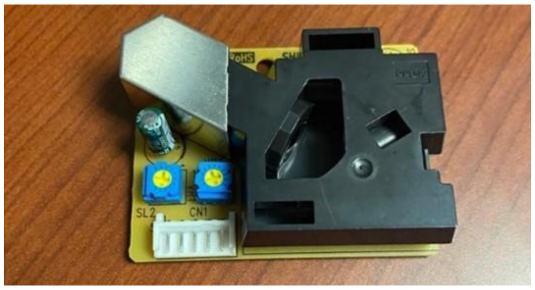


Fig. 2. PPD42NJ-Dust Sensor

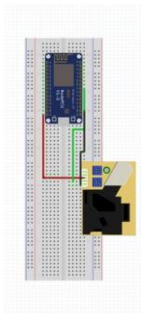


Fig. 3. Node 1-PPD42NJ connected to NodeMCU

B. MQ9-Gas Sensor for CO

Fig.4 shows a typical MQ9 sensor, introduced in 2015. In our system it is used to measure Carbon Monoxide (CO). It consists of a core sensing material namely, Stannic Oxide (SnO2) which has low conductivity in clean air. The voltage cycles from high voltage to low voltage. At a high voltage the sensor can measure combustible gases and at low voltage of 1.5V it can measure CO. [3]

Fig. 4. MQ9-Gas Sensor

Fig. 5 shows the node consisting of Mq9 sensor connected to NodeMCU using A0 pin. The microcontroller can read CO values from the sensor using AnalogRead function programmed onto NodeMCU using Arduino IDE. According to US Environment Protection Agency average level of CO at home should be no more than 30ppm [14]

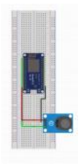


Fig. 5. Node 2-MQ9 connected to NodeMCU

The reason we have chosen these two sensors is that they are low-cost and can be easily programmed to obtain sensor readings. Users with little or no knowledge can install this system in their indoor environment. These sensors are easily accessible. As they are low cost these sensors can be easily replaced as well. Therefore, creating a cost-effective and reliable IoT system.

This system can be easily extended into a bigger picture. Let us say for example a house consists of a total of 4 rooms. A replica of each node needs to be placed in every room. The data collected by all 8 NodeMCUs can be stored in the cloud. Each room is now equipped with a sensor for detecting PM2.5 and CO. Through an email alert the user identifies which sensor has measured a concentration level beyond the optimum level. In turn informing the user that the room has polluted air.

III. Network Experiment Description

The network architecture shown in Fig.1 includes two parts: a Wired sensor to NodeMCU [4] interface and wireless node to cloud interface. As shown in Fig 1, each of the multiple end nodes consist of one ESP8266 or NodeMCU module and the sensors. The MQ9 [3] and the PM2.5 [13] sensors attach to an individual NodeMCU via the onboard GPIO pins. Specifically, the PM2.5 sensor communicates with the NodeMCU via one of the digital pins while the MQ9 sensor communicates via one of the analog pins. The NodeMCU comes with an inbuilt Wi-Fi communication module with an antenna [4], enabling it to connect to any nearby 2.4GHz Wi-Fi access point. We exploit this to connect it to a cloud and aggregate the sensor data that it collects in the cloud, making the need for a controller node obsolete. Each of the end nodes is connected to the ThingSpeak cloud database, using the ThingSpeak client in the Arduino IDE [8] via the REST API [6] and HTTP methods [10]. The cloud connection is established in two stages, connection to Wi-Fi AP, and connection to cloud through HTTP of the Wi-Fi Client. Specifically, in the first stage, the NodeMCU establishes a WIFI connection through a defined Wi-Fi client object. The SSID and authentication of the AP are used to initiate the connection through the Wi-Fi client object. On successful connection with the AP, the status is updated on the NodeMCU. The second step of wireless connections is to connect to ThingSpeak via the same Wi-Fi client. ThingSpeak provides a library for ESP8266 or the NodeMCU [10], which enables the node to connect to the ThingSpeak URL via an HTTP request. The HTTP request is done through the REST API, with which ThingSpeak provides users flexibility to write and read to the channels via various defined HTTP methods and functions[12]. The additional parameters necessary to make this connection are the channel id, API key, and fields of the channel to which the data is supposed to be uploaded, all of which are unique to every user on ThingSpeak. The channel specific id and API key act as identifier and encryption respectively to enable the connection to the specific channel. Once connected to the cloud, the sensor value read from the wired connection to the sensor is written to the channel using a writeField command which takes Channel ID, API key and the value to be written as its arguments. Again, this is done through a HTTP based request using the in-built REST API. On successful write to the channel a status code is returned via a HTTP code to the node from ThingSpeak. This way, the WiFi based cloud connection is established between the NodeMCU and ThingSpeak. The pictorial description of the sequence of steps involved in network establishment are shown in Fig.6 below. Finally, the last part of the network is the automation part where upon data insertion the cloud compares the reading with a set threshold and triggers an email to the registered email address whenever a data reading exceeds the threshold value, with time stamps. This is further covered in section V. The use of multiple sensors prompts the use of multiple channels to distinguish each sensor data. To this end, we use the unique channel ID and API keys to distinguish between the two channels, the PM2.5 levels and gas levels respectively. Thus in this way, ThingSpeak's REST API enables the seamless connection and distinguishment between these two channels and their corresponding sensors by providing channel specific identifiers.

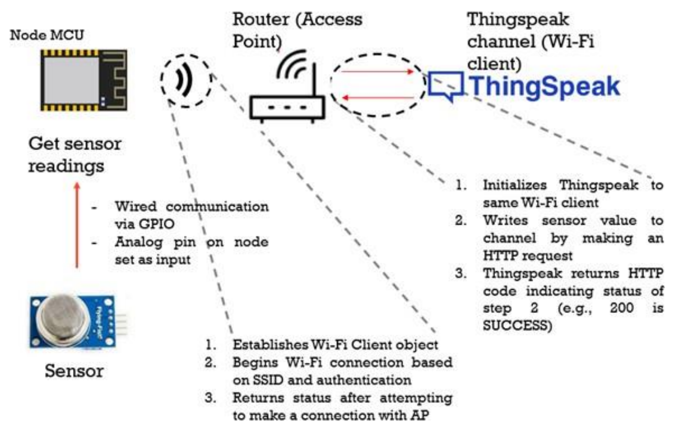

Node MCU
Get sensor readings
- Wired communication via GPIO
- Analog pin on node set as input
Sensor
Router (Access Point)
Thingspeak channel (Wi-Fi client)
ThingSpeak
1. Establishes Wi-Fi Client object
2. Begins Wi-Fi connection based on SSID and authentication
3. Returns status after attempting to make a connection with AP
1. Initializes Thingspeak to same Wi-Fi client
2. Writes sensor value to channel by making an HTTP request
3. Thingspeak returns HTTP code indicating status of step 2 (e.g., 200 is SUCCESS)

Fig. 6. Different connections and stages involved in the IoT network

IV. Cloud Experiment Description

Data aggregated at the NodeMCU is uploaded to the cloud using ThingSpeak. ThingSpeak is an IoT analytics service that allows users to aggregate, visualize, and analyze live data streams in the cloud. It is often used for prototyping and proof-of-concept IoT systems that require analytics. Here, the IoT technology is implemented for a periodic monitoring of

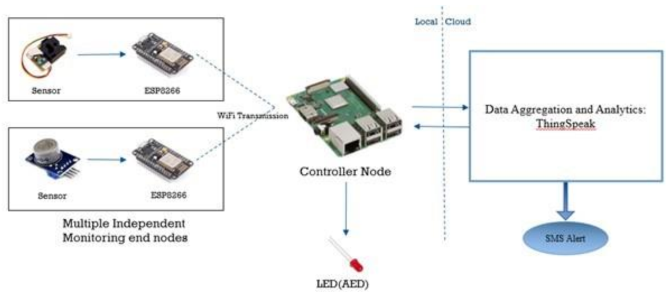


Fig. 7. IoT level 5 network architecture used in Project proposal

designed PPD42NJ and MQ9 sensor information to monitor the PM2.5 and CO levels respectively. The information is collected from sensors on different nodes and is saved on the local database using the ThingSpeak cloud database.

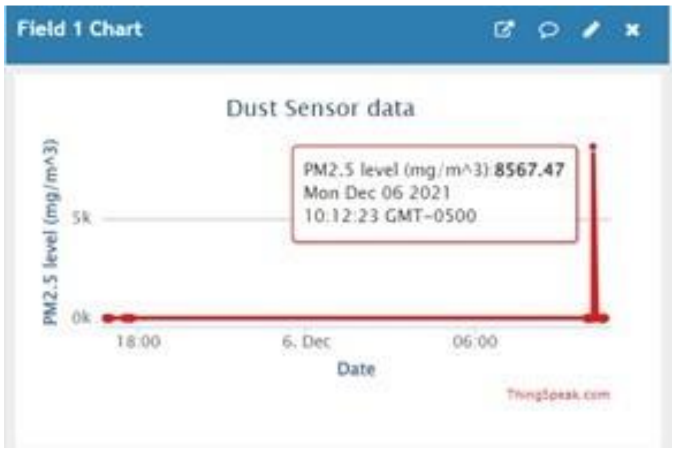


Fig. 8. Graph of real-time PM2.5 monitoring in cloud

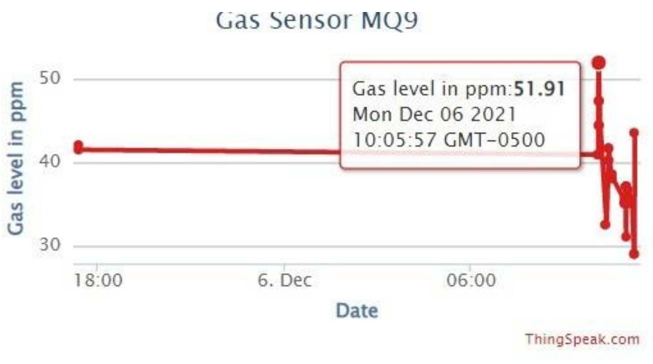


Fig. 9. Graph of real-time Carbon Monoxide monitoring in cloud

The network shown and described in section III serves as a proof of concept to demonstrate the working of an IoT based Air quality monitoring system. This can be easily extended to multiple end nodes in many ways. A possible way is to use a single NodeMCU and connect all the different additional air quality monitoring sensors to the same NodeMCU and duplicate this setup. Such a node configuration can be deployed in multiple rooms giving the air quality all across one’s house. This expansion is very simple because each sensor output can be separately mapped or routed to a unique field in one single ThingSpeak channel or can be routed to different unique ThingSpeak channels, depending on the requirement. Owing to the unique identifiers in each channel, and each channel in ThingSpeak providing up to 8 fields in parallel, multiple sensors can be used to monitor a room’s air quality as well. Thus, this network can easily be expanded beyond this proof of concept implementation.

As shown in Fig 8, channel 1 indicates the real-time monitoring of PM2.5 data in mg/m3 and channel 2 indicates the real-time monitoring of CO data, as shown in Fig.9. The complete running application on each NodeMCU automatically initiated data upload to the channels after the booting of the microcontroller and the collection of sensor data was completed [5]. Each data point on the channel graph indicates the value of the sensor reading referenced with a timestamp of record. ThingSpeak based cloud services are generally used for storing information in the online monitoring cloud database for organizational analytics technology services. This cloud service is measured as the data is stored in the database MySQL. The data is continuously updated to the related channels and is analyzed in a Real Time System (RTS), also called as React using ThingSpeak. React works with ThingHTTP [7] and ThingTweet [9] to perform actions when channel data meets a certain condition.

In the air quality monitoring IoT system, a React analysis is setup such that every new entry to the PM2.5 and CO ThingSpeak channels are compared to a threshold level. This threshold signifies that air pollutant levels below its value are qualified as breathable air and any instance where the received sensor value crosses the threshold indicates that the indoor air is hazardous to breathe. The threshold levels for CO and PM2.5 were set by referring to [16].

Fig.10 shows the React parameters set for the PPD42NJ sensor channel on ThingSpeak. Condition Type is set to ‘Numeric’ and Test Frequency set to ‘On Data Insertion’ indicates that a numeric value with a condition will be checked with every new data inserted to the cloud channel. The channel on which React analysis takes place is set based on the Channel ID. In Fig. 10, threshold level for PM2.5 levels is set to 100 mg/m3. Every time a new sensor value is uploaded to this channel and it crosses the threshold, the action field, that is set to ‘MATLAB Analysis’ will trigger a MATLAB code. Here, the code is set to trigger an Email indicating that the sensor has detected high pollutant levels indoors. ThingSpeak provides a RESTful web service that returns data formatted as an internet media type such as JSON, HTTP request, XML, image or text. The MATLAB code uses the ‘webwrite function provided by ThingSpeak to POST (similar to PUT, which is a REST operation) data to a web service. The function takes in a URL and data as arguments. Here, the URL is set as ‘https://api.thingspeak.com/alerts/send’ which is provided by ThingSpeak to trigger emails and data is the body and message of the Email. Fig. 11 shows an example email that is sent to the user when the PM2.5 level measured by the PPD42NJ sensor crosses the set threshold level of 100 mg/m3. The email subject and content of the body can be modified by changing the MATLAB code triggered by React. Every email instance is also logged with a timestamp to indicate when the event was recorded on the cloud channel.

As a scope of improvement, the current cloud implementation can be extended to trigger an actuation mechanism on the NodeMCU. For example, every time the sensor data crosses the threshold, instead of triggering an email, the cloud application can trigger the ventilation mechanism on site to recycle indoor air and decrease the pollutant levels.

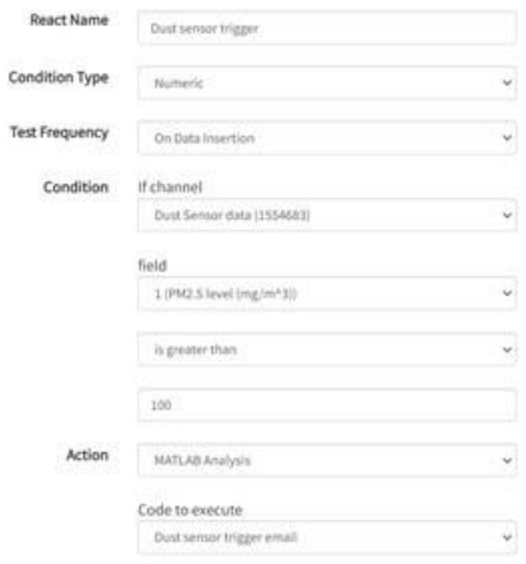
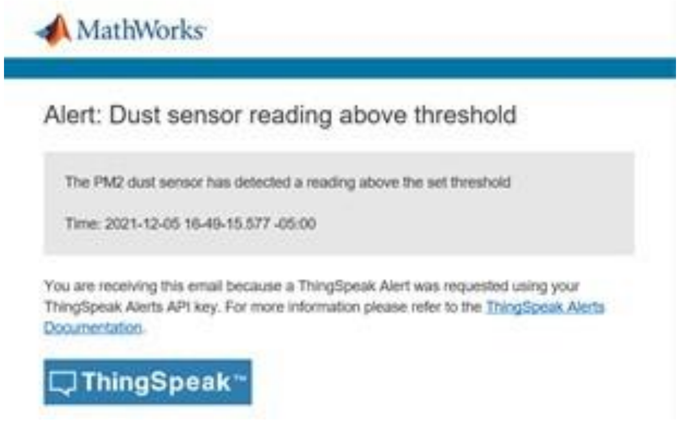

React Name	Dust sensor trigger
Condition Type	Numeric
Test Frequency	On Data Insertion
Condition	If channel: Dust Sensor data (1554683)
	field: 1 (PM2.5 level (mg/m^3))
	is greater than
	100
Action	MATLAB Analysis
	Code to execute: Dust sensor trigger email

Fig. 10. React parameters set for PPD42NJ (PM2.5) sensor


MathWorks

Alert: Dust sensor reading above threshold

The PM2 dust sensor has detected a reading above the set threshold

Time: 2021-12-05 16-49-15.577 -05:00

You are receiving this email because a ThingSpeak Alert was requested using your ThingSpeak Alerts API key. For more information please refer to the ThingSpeak Alerts Documentation.

ThingSpeak™

Fig. 11. Email triggered from ThingSpeak when PM2.5 value crossed threshold

V. Conclusion

In this project, air quality monitoring system in integrated with a wireless sensor network using NodeMCUs and the Internet of Things using ThingSpeak cloud. Here, the Wi-Fi protocol is implemented in connecting the sensor nodes with the cloud. The future development work is on building an actuation system integrated with the central ventilation system of a residential unit so that the indoor air quality is automatically controlled based on the sensor and cloud operations.

The PM2.5 level measuring PPD42NJ Dust sensor was of decent build quality. There was sufficient documentation available for usage of the sensor. It included pinout, operation modes, operating voltage levels. The sensors do not require much of soldering and is very well packaged. They are pre-calibrated and give accurate readings from the first few cycles itself. The CO level measuring MQ9 sensor was lightweight and operated with only 3.3V. Although the sensor was not pre-calibrated, this meant that the initial few readings uploaded were not completely accurate.

The NodeMCU used in the project served both as the communication and controller unit by itself. It had sufficient GPIO pins for interfacing the sensors and operated both at 3.3V and 5V making it possible to connect both MQ9 and PPD42NJ sensors without using additional voltage converter equipment. There was a strong community support and very well written documentation available on the internet for its usage. The NodeMCU operated in three different sleep modes, each consuming different levels of current but one common functionality missing from all sleep modes was that we could not trigger the MCU from sleep mode wirelessly because the Wi-Fi module on the controller was turned off. This made it difficult to configure the network as we had to rely on timing cycles between the nodes.

Additionally, the NodeMCU would often fail to auto-connect to the Wi-Fi Access Point after waking up from the sleep mode. To make this action consistent, we had to configure the NodeMCU in Wi-Fi STA mode though it was the default mode.

References

[1] Nehul Agarwal, Chandan Swaroop Meena, Binju P Raj, Lohit Saini, Ashok Kumar, N. Gopalakrishnan, Anuj Kumar, Nagesh Babu Balam, Tabish Alam, Nishant Raj Kapoor, and Vivek Aggarwal. Indoor air quality improvement in covid-19 pandemic: Review. *Sustainable Cities and Society*, 70:102942, 2021.

[2] Department of Health, New York State. *Fine Particles (PM 2.5) Questions and Answers*, December 2021. https://www.health.ny.gov/environmental/indoors/air/pmq_a.htm , Last accessed: December 15, 2021.

[3] DFRobot. *Gravity: Analog CO/Combustible Gas Sensor (MQ9) For Arduino*, December 2021. texttthttps://www.dfrobot.com/product-688.html , Last accessed: December 15, 2021.

[4] Espressif Systems. *ESP8266EX - Espressif Systems*, December 2021. https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf , Last accessed: December 15, 2021.

[5] Syed Faiazuddin, M.V. Lakshmaiah, K. Tanveer Alam, and M. Ravikiran. Iot based indoor air quality monitoring system using raspberry pi4. In *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 714–719, 2020.

[6] MathWorks. *REST API – MATLAB and Simulink*, December 2021. https://www.mathworks.com/help/thingspeak/rest-api.html, Last accessed: December 15, 2021.

[7] MathWorks. *ThingHTTP App – MATLAB and Simulink*, December 2021. https://www.mathworks.com/help/thingspeak/thinghttp-app.html, Last accessed: December 15, 2021.

[8] MathWorks. *ThingSpeak Communication Library for Arduino, ESP8266 and ESP32*, December 2021. https://github.com/mathworks/thingspeak-arduino , Last accessed: December 15, 2021.

[9] MathWorks. *ThingTweet App – MATLAB and Simulink*, December 2021. https://www.mathworks.com/help/thingspeak/thingtweet-app.html, Last accessed: December 15, 2021.

[10] MathWorks. *Write Data to channel – MATLAB and Simulink*, December 2021. https://www.mathworks.com/help/thingspeak/writedata.html , Last accessed: December 15, 2021.

[11] Occupational Safety and Health Administration. *Carbon Monoxide Poisoning*, December 2021. https://www.osha.gov/sites/default/files/publications/carbonmonoxide-factsheet.pdf , Last accessed: December 15, 2021.

[12] Sonya Sachdeva and Sarah McCaffrey. Using social media to predict air pollution during california wildfires. In *Proceedings of the 9th International Conference on Social Media and Society*, SMSociety '18, page 365–369, New York, NY, USA, 2018. Association for Computing Machinery.

[13] SHINYEI Technologies. *PPD42NJ Particle Sensor unit*, December 2021. https://www.shinyei.co.jp/stc/eng/products/optical/ppd42nj.html , Last accessed: December 15, 2021.

[14] United States Environmental Protection Agency. *Carbon Monoxide's Impact on Indoor Air Quality*, December 2021. https://www.epa.gov/indoor-air-quality-iaq/carbon-monoxides-impact-indoor-air-quality , Last accessed: December 15, 2021.

[15] United States Environmental Protection Agency. *Wildfires and Indoor Air Quality (IAQ)*, December 2021. https://www.epa.gov/indoor-air-quality-iaq/wildfires-and-indoor-air-quality-iaq, Last accessed: December 15, 2021.