Pawamaan
Air Quality Monitoring System

Abhishek Prasad, Parth Gupta, Soham Bhar, Mohit Mohandas

Abstract

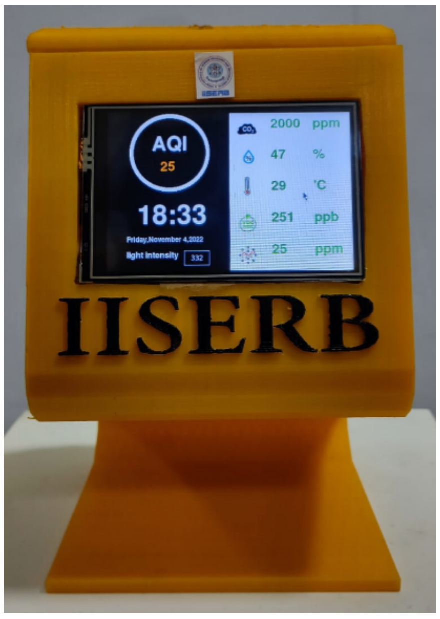
With a recent fall in air quality around us, individuals' health is at risk. The importance of a good atmosphere is evident now as people are experiencing symptoms of various illnesses. Some problems, such as fatigue, short breath, or restlessness, are caused by minute factors of the air, which are usually ignored.

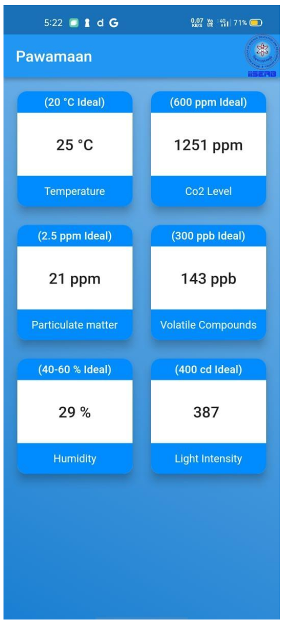
The solution presented is a Smart Air Quality Monitoring system which shall serve as a means to act upon the degrading air quality and proactively avoid its ill effects

1 Introduction

Most work is now done online while sitting comfortably at home and spending much of our time indoors in the post-covid era. Despite being more health conscious than ever, people are the least informed about the air quality in their surroundings. We frequently ignore how harmful it is to our health. In our proposed solution, the user can see quantitatively how clean or contaminated the indoor air is around them using this Smart Air Quality Monitoring System. The air quality around them will be improved by giving them a summary of their microclimate and supporting them in making better decisions.

We have provided the user an end-to-end solution where the user can see six parameters of air which are, CO_2 ,Particulate matter, Volatile Organic Compounds, Humidity, Temperature and light intensity. All these parameters are measured and displayed on the device and the mobile app. The user can also see the graphical data for the past seven days. “Pawamaan” is made from two words “Pawan” and “Maan” which, in Sanskrit, translates to “Air” and “Measurement” thus justifying the name.





Pawamaan

(20 °C Ideal)	(600 ppm Ideal)
25 °C	1251 ppm
Temperature	Co2 Level
(2.5 ppm Ideal)	(300 ppb Ideal)
21 ppm	143 ppb
Particulate matter	Volatile Compounds
(40-60 % Ideal)	(400 cd Ideal)
29 %	387
Humidity	Light Intensity

(PAWAMAAN The Smart Air Quality Monitoring System and the Mobile application)

2 Goals

The goal of designing this device is to give the user a quantitative view of how clean or contaminated the air inside is around them and help them to act accordingly.

3 System Architecture and Design

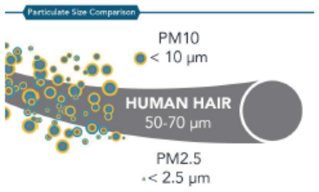
3.1 Hardware

Component /Device	**Use**	**Power consumption (in milliamps)**
MHZ-19 Sensor	Used to detect the concentration of Carbon di Oxide	20mA
PMS 7003 Sensor	Used to detect the number of particles of size 1.5 and 10 micrometers	100mA
AGS-02 MA Sensor	Used for detecting the total concentration of Total Volatile Organic Compounds	35mA
DHT-11 Sensor	To measure Humidity and Temperature	30mA
LDR	For measuring light intensity	<1mA
Raspberry pi	Used as a Microprocessor for running the Python code and connecting to the LCD screen and Firebase cloud	900-1100mA
ESP-8266	Used to integrate all the sensors and communicate with the Rpi	70mA
3.5-inch Rpi Display	Touchscreen Display as a user interface	140 mA
Power bank (10000 mAh)	To power the electronics inside	

CO_2
MHZ-19

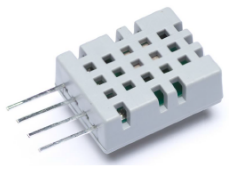
Normal outdoor value: 350-500 ppm
Higher levels of CO2 adversely impact day to day activities.
Users experiencing Dizziness, Restlessness and Asphyxia are exposed to unhealthy levels of CO2

Particulate matter (PM)
7003 Sensor



Normal values below 30 µg/m3
Higher levels of PM can aggravate lung disease, asthma attacks, acute bronchitis, etc.

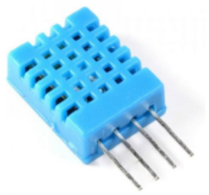
Volatile Organic Components (VOC)
AGS-02-ma Sensor



The acceptable range for TVOC is 300-500 PPB
Unusual levels of Organic compounds in the air lead to permanent damage to organs.

Raspberry Pi, ESP-8266
and LCD screen

Humidity and Temperature
DHT11 Sensor



Light Intensity
LDR

The sensors are chosen to take into account their low cost, low power consumption, longer life spans and availability.The sensors were calibrated beforehand by the manufacturer. Also, some sensors like the MHZ-19 CO2 sensor support built-in auto calibration function at initialisation which takes about 15-20 minutes to normalise to the actual value at every startup. We chose the DHT-11 for temperature and humidity due to its accessibility, but a more precise and expensive sensor can be used because our system is adaptable to the addition of other sensors.The ESP Node MCU used here is suitable for simultaneously communicating with all the sensors as it facilitates multiple Hardware and Software Serial. The ESP is also able to power all the sensors by itself.

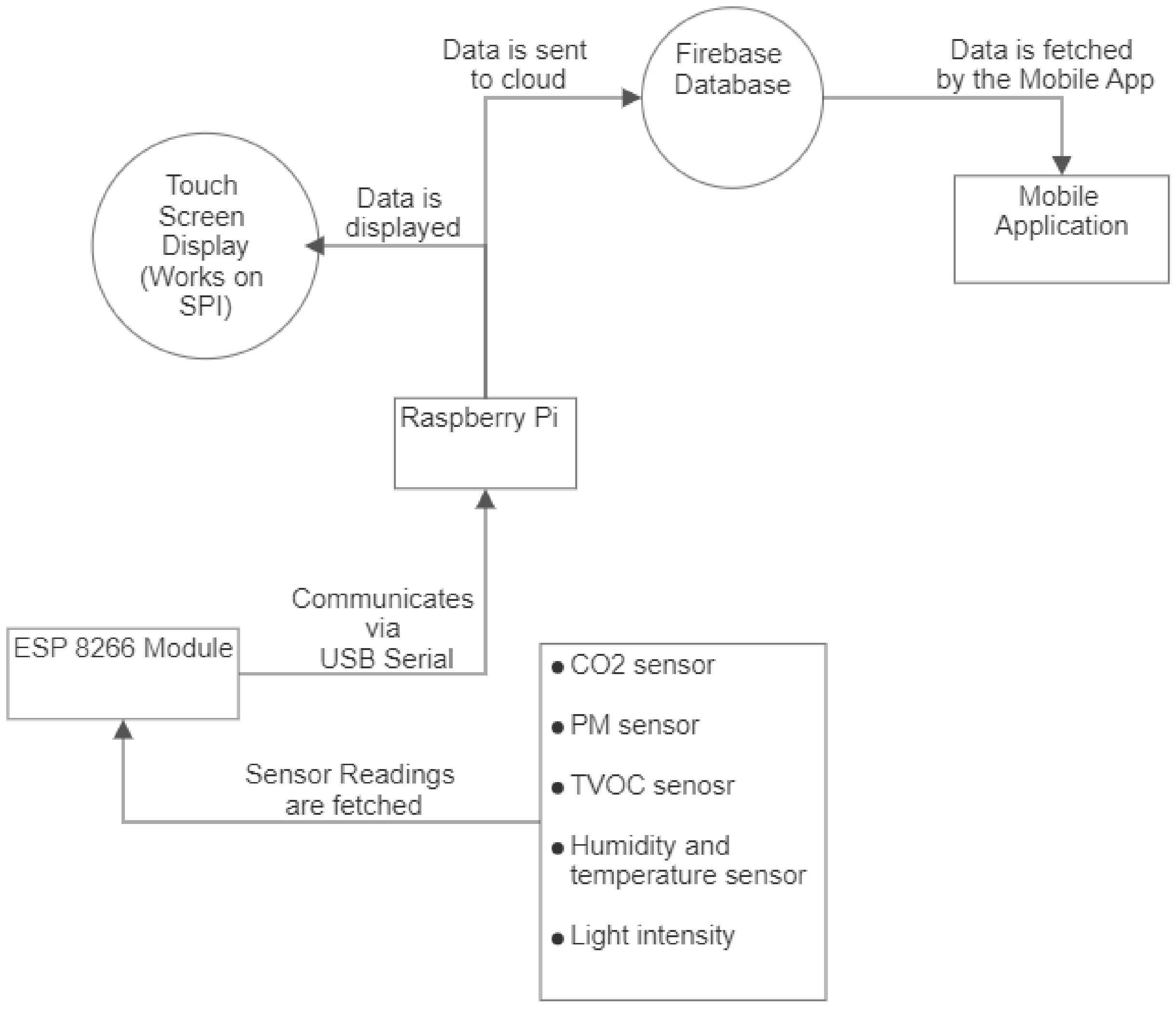
The ESP transfers the sensor values to the Rpi via Serial USB, and the Rpi takes care of the rest. The Rpi stores the data locally on the device for seven days and also transfers the value of the sensor to a Firebase account in real time. The device needs to be connected to a WiFi network to transfer the data to Firebase.

The whole system can work with ESP, but we choose RPI for incorporating touch screen and voice control for future prospects as "Offline" voice control features require high computational power.

We designed and 3D printed the enclosure in our lab at IISERB. It has one distinct feature that addresses the removal of waste heat from the raspberry pi board as well as proper airflow for the sensor. The warm air surrounding the raspberry pi rises and exits through the top vent, creating a low-pressure inside that sucks air from the bottom vents where the sensors are located, ensuring proper airflow inside the device.

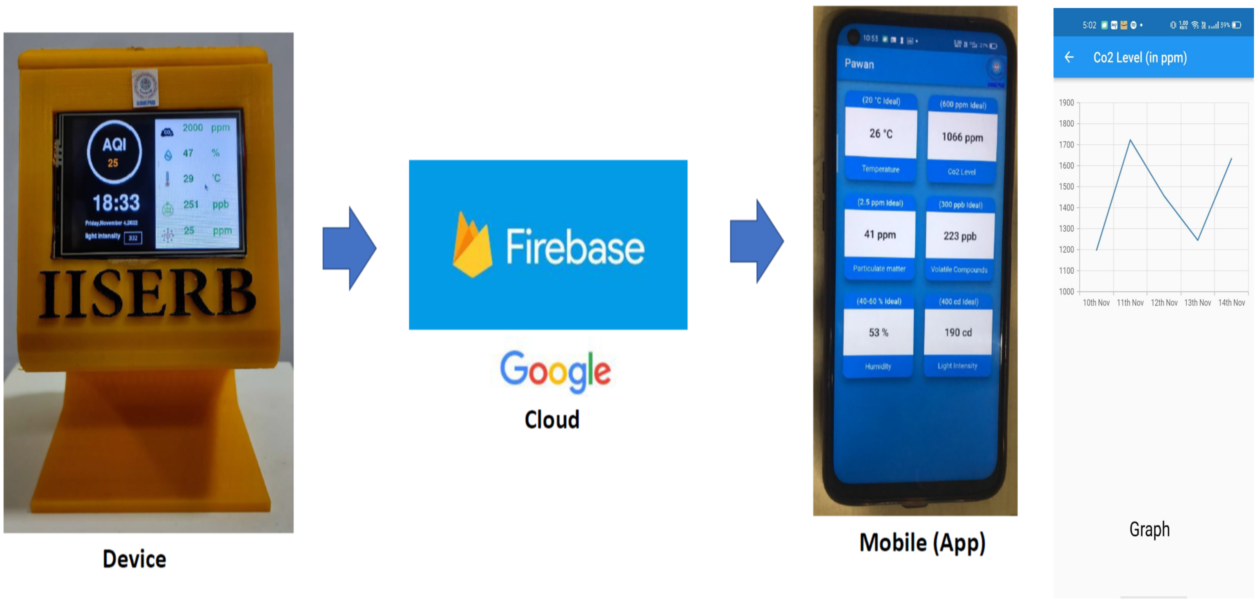
The device is powered with a Li-Po battery (10000 mAh), which can be recharged via the USB-C port on the back. The device can be used while it is charging by using a suitable power adaptor or a phone charger.

Hardware Block Diagram:


Data is sent to cloud
Firebase Database
Data is fetched by the Mobile App
Mobile Application
Touch Screen Display (Works on SPI)
Data is displayed
Raspberry Pi
Communicates via USB Serial
ESP 8266 Module
Sensor Readings are fetched
• CO2 sensor
• PM sensor
• TVOC senosr
• Humidity and temperature sensor
• Light intensity

3.2 Software

We have developed a mobile app using Flutter. The app can run on Android and iOS. The app displays the values of each sensor and a graph of the data from the last seven days. For the device GUI, we used the Tkinter library of Python and all the backend is written in Python. The GUI presents all of the sensor data to users in an easily understandable style. We have used Crontabs to facilitate the automatic launch of scripts at the device's startup. For data storage and the transfer, we have used Google firebase services.



Moreover, we have used Google's Firebase API to access our firebase account and exchange data from it. The Raspberry Pi inside the device uses standard Raspberry Pi OS, and the scripts are written in Python.

We have used Crontabs to facilitate the automatic launch of scripts at the startup of the device. Also, we have used the Tkinter library for GUI on the device.

Also, the ESP communicates through Serial via USB to the Rpi.

4 Addressing Challenges

The solution looks pretty straightforward, but we encountered various types of errors and conceptual faults while building this device.

1. The number one problem was to make all the sensors communicate at the same time.
    a. Initially, we could only use a maximum of three sensors simultaneously. Adding another sensor would cause unexpected errors. We read the official sensor documentation from the manufacturer to find the source of errors and worked our way, eventually mitigating it.
2. Insufficient Power
    a. The whole system comprises RPI, LCD screen, esp32 and sensors. We faced a major problem in power management. We tried Underclocking the CPU so that RPI consumed less energy and tried a few other solutions. Eventually, we discovered that the cables were not correctly transferring power. We then used good-quality cables and ripped off the excess plastic to make them fit inside the Enclosure.
3. Communication between different scripts
    a. Initially, we made two separate scripts, one for Gui and another for sensor data. This created a problem of periodically feeding the output from one script to another. After a few

iterations of code, we eventually wrote all the scripts as part of a single script. This aided us by decreasing unnecessary complexity.

5 Performance Evaluation and Testing Results

The device presented here was tested in three very different environments. Every five minutes, the device takes a recording. The reading from the devices was taken from

i. A large open Football ground on Campus
ii. A closed Canteen with poor Ventilation
iii. Institute library
iv. A closed room (Cabin/Office)

The results and concentrations match of what was expected. For reference, we observed the following:

I. The CO2 concentration at the football ground was ideal (~600-800 PPM)
II. The total VOC concentration was far higher inside the closed canteen.
III. The CO2 concentration was comparatively higher in the closed room(~1200-1400 PPM).

As far as the performance is concerned, we have measured the following:

I. When run continuously at 100% battery, the display on, and data being constantly fetched and forwarded, the device can run up to 6 Hours, after which charging is needed. Note that the device can run while charging itself.
II. The overall response time of sensors is about 60 seconds. This is because the sensors measure the average concentration over a specific time interval.

6 Can be used

- To keep an eye on the quality of the air inside of buildings, whether they be homes, businesses, or other indoor spaces, and to assist people in making decisions to improve the air quality
- To monitor different aspects of a food storage warehouse and assist in maintaining the inside environment (this is our future plan which can be done with better quality sensors and a new design of the device)

7 Concluding Remarks and Avenues for Future Work

Since it has an impact on everyone's daily life, this air quality issue should not be taken lightly. We have encountered startlingly high levels of CO2 indoors even while testing the device and have responded by increasing ventilation to the problem.

We've been working on a second iteration of the device with the capability to control air vents or exhaust inside the home based on the indoor air quality. The device will be able to regulate these directly, or the user can use an app. A WiFi-enabled IoT network of devices can be used for this (ESP 32 in this case). Its feature can be extended to control other appliances like fan, lights or AC making it a complete smart home ecosystem.

RPI and the LCD screen can be removed from this device to make it more portable and energy-efficient but at the cost of less functionality. Data can be recorded in homes and lab settings where the user can access the data via their phone.

The device's design prioritises versatility. Additional sensors can be added in response to consumer demand with minor software adjustments. This can be done due to the availability of pins in ESP and RPI.

Advanced voice control features are the additional significant functionality that can be added. We can currently demonstrate "Offline" Speech Recognition in real time. With the addition of this feature, it will be more appealing to customers as a real-time air-monitoring companion for smart homes.

This device serves as the framework for our future goals. The purpose of this device was to showcase a Proof of Concept where we integrated sensors, developed a GUI and an app, and enabled remote access to data and its storage. Additionally, we were able to create a functional enclosure. These can be combined in many ways to develop devices that are tailored to various sorts of problems.

8 Availability

GitHub Repository: https://github.com/IISERB-UG/Pawamaan---Smart-Air-Quality-Monitoring-System

Video Demonstration: https://drive.google.com/file/d/11L0pprsN07rqs1DkLnEMdsFnRsXau7B2/view

References:2

[1] https://youtu.be/OpaEW-Q4m1s (0:00-0:19 in demonstration video)
[2] https://www.youtube.com/watch?v=_K21bv638qc (0:19-0:54 in demonstration video)

(Note: All the photos and videos are used under "fair use" as we are not using them for any commercial purposes)