

AI-Based Indoor Air Quality Management Using TinyML

Vishal V. Panchbhai¹, Gopal Premkumar Deotare², Vansh Sharad Titarmare³, Harshal Harishchandra Chopkar⁴, Himanshu Rajesh Raut⁵

^{1,2,3,4,5}*Department Of Electronics and Telecommunication Engineering, Priyadarshini College of Engineering, Nagpur, Maharashtra, India*
doi.org/10.64643/IJIRT209857

Abstract—People spend most of their day indoors, yet the air in homes, classrooms and offices is rarely checked. Low-cost monitors usually show raw readings or push them to a cloud server, leaving the user to work out what the numbers mean. This paper presents a compact monitor that interprets the air on the device itself. An ESP32 reads CO₂, temperature and relative humidity from an SCD40 sensor, particulate matter (PM1.0, PM2.5 and PM10) from a PMS5003 sensor, and a combustible-gas signal from an MQ6 sensor. A small neural network with 12 hidden neurons runs on the ESP32 and rates the room as GOOD, MODERATE, POOR or VERY POOR. In parallel, a PM2.5-based AQI is calculated, and a separate threshold check on the MQ6 sounds a buzzer when a gas leak is suspected. The results appear on an OLED screen and on a web page that the ESP32 serves to a phone, together with simple advice on running a fan or purifier or opening a window. On a held-out synthetic test set, the classifier reached 87.3 % accuracy. The prototype shows that useful, offline interpretation of air quality is possible on inexpensive hardware.

Index Terms—Edge AI, CO₂ Sensing, Gas Leak Detection, Neural Network Classification.

I. INTRODUCTION

Indoor air carries a mix of pollutants: carbon dioxide from breathing, fine dust, smoke and cooking fumes, and gas escaping from faulty appliances. At levels that already matter for health, most of these cannot be seen, smelled or tasted, so a sensor is the only practical way to notice them. The effects go beyond health. In a controlled chamber study, Satish et al. found that decision-making performance fell as CO₂ rose to concentrations commonly found in crowded, poorly ventilated rooms [1]. Many monitors are now built as IoT devices, and a large body of literature describes them [2]. TinyML takes a different route: a trained machine-learning model is compressed until it fits in a

microcontroller's memory, so the decision is made where the data are collected [4], [3]. Bringing the two ideas together gives a monitor that can both sense and decide without relying on a server.

Most commercial and hobby monitors stop at showing numbers. A reading such as 'CO₂ = 1450 ppm' means little to an ordinary user, who still has to decide whether to open a window or switch on a purifier. Where an interpretation is offered, it is usually produced in the cloud, so the device is only as useful as its network connection and every result arrives with some delay. The TinyML studies reviewed in Section 2 mostly deal with a single pollutant or a single prediction target. Very few studies were found that combine particulate, CO₂, climate and gas sensing on one board and also turn the result into clear actions for non-expert users.

This paper presents a low-cost air quality monitor built around an ESP32 and three sensors. A small neural network (7 inputs, 12 hidden neurons, 4 classes) runs directly on the board and rates room air at one of four levels, while the device also calculates an AQI from PM2.5 readings and suggests simple actions. Gas leaks trigger a separate fixed-threshold alarm, so safety warnings never wait on the model. The OLED display, the web dashboard hosted on the ESP32 itself, and the buzzer all keep working without internet. A custom PCB, designed in EasyEDA and made in-house, replaces the loose jumper wiring common in similar prototypes.

Section 2 reviews related work and the gaps it leaves. Section 3 describes the proposed system: its hardware, data, model and decision logic. Section 4 presents the results, and Section 5 discusses them in the light of earlier work and sets out the limitations. Section 6 concludes the paper, and Section 7 outlines future work.

II. LITERATURE REVIEW

The Air Quality Index (AQI) shows people how clean the air is with one number from 0 to 500 and a simple label such as Good or Hazardous. Each pollutant is given its own score, and the highest score becomes the AQI. India has used its own National Air Quality Index since 2014. The AQI was made for outdoor air, so indoors it handles fine dust well but leaves out carbon dioxide and gas leaks. For this reason, researchers now combine AQI calculation with extra sensors and small machine-learning models on low-cost devices.

A number of recent studies show that machine learning can run on small, inexpensive hardware for air-quality tasks. Ken et al. [5] predicted ozone in real time on an Arduino Nano 33 BLE Sense, combining an MQ7 carbon-monoxide sensor with the board's own temperature and pressure sensors. They trained a linear-regression model in Edge Impulse on a public Indian air-quality dataset and obtained the best results when all three inputs were used. They traced the remaining errors to the lack of a humidity sensor and to sudden changes in the weather. Hayajneh et al. [6] studied transfer learning on microcontrollers, in which a general model is pre-trained on a large dataset and then fine-tuned on the device itself. Their experiments covered human-activity recognition on two public datasets rather than air quality, but they are relevant here because they show that small embedded models can still be adapted after deployment.

Table 1: Summary of related TinyML work

Ref.	Method and platform	Reported result	Limitation noted
[5]	Linear regression for ozone prediction (Edge Impulse); Arduino Nano 33 BLE Sense with MQ7; public Indian dataset	$R^2 \approx 0.92$ with all three inputs	No humidity sensor; errors under rapid weather change
[6]	CNN-LSTM with transfer learning on microcontrollers; Motion Sense and UCI datasets	Higher accuracy, shorter training time	Tested on activity recognition only

Ref.	Method and platform	Reported result	Limitation noted
[7]	Predictor plus missing-feature imputer, quantised; single microcontroller	R^2 above 0.70; imputer handles under 80 % missing data	Slight accuracy loss after quantisation
[8]	XGBoost AQI prediction within 2 MB; Raspberry Pi Pico W, onboard and web-sourced data	75.2 % accuracy; about 1.6 ms inference	About 60 % accuracy in real-time validation

Apart from these experimental studies, Warden and Situnayake [4] and Ray [3] describe how machine-learning models are reduced in size to run on microcontrollers, and the systematic review by Saini, Dutta and Marques [2] covers how IoT-based indoor air-quality monitors are designed and used.

Taken together, these works leave three gaps. First, most of them focus on one pollutant or one prediction, whereas the condition of a room depends on several parameters at the same time. Second, results are normally reported as numbers or predicted values and are not translated into guidance for the people in the room.

Third, safety functions such as gas-leak warnings are seldom kept apart from the learned model, so a weakness in the model could delay an alarm. The system described here addresses these gaps in three ways: it combines particulate, CO₂, climate and gas sensing on one board; it turns the result into a four-level label with matching advice; and it keeps the gas alarm as an independent threshold rule.

III. METHODOLOGY

The overall architecture of the system is shown in Fig. 1. The sensors pass their readings to the ESP32, which collects and normalises the data, runs the TinyML inference, calculates the AQI and checks the gas-safety rule before deciding what to display or sound. A single common power block supplies every part of the circuit.

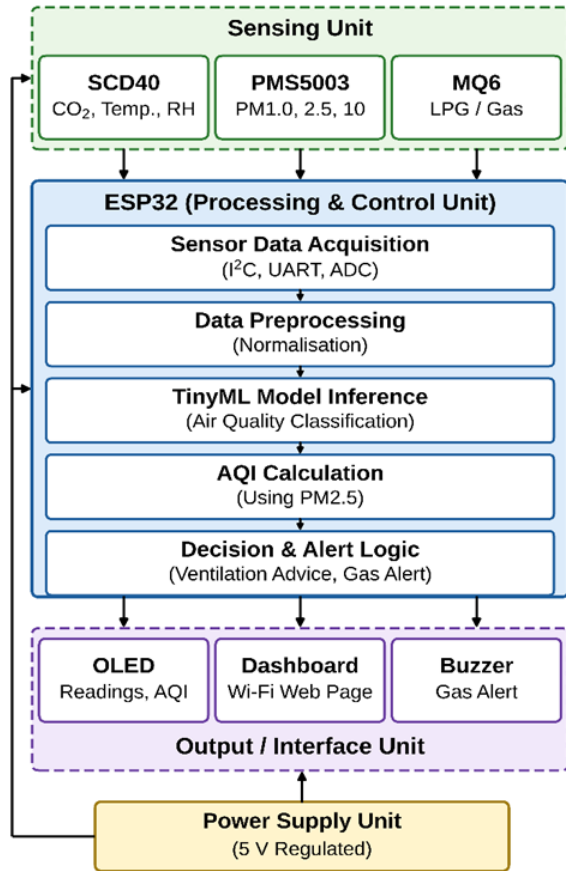


Fig. 1: Block diagram of the proposed system

Hardware

Table 2 lists the modules and the role of each. The ESP32 DevKit (Fig. 2) is a 32-bit microcontroller board with built-in Wi-Fi and Bluetooth, ADC inputs, and I²C and UART ports. It collects all the sensor data, runs the neural network, computes the AQI, drives the OLED and buzzer, and hosts the web dashboard. It operates at a 3.3 V logic level.

Table 2: Hardware modules and specifications

Module	Specification	Role in the system
ESP32 DevKit	32-bit controller, Wi-Fi and Bluetooth, ADC, I ² C, UART; 3.3 V logic	Acquisition, inference, AQI, display, web server
SCD40 (Sensirion)	Photoacoustic CO ₂ sensor with temperature and humidity; I ² C	CO ₂ , temperature and humidity inputs
PMS5003	Laser-scattering particulate sensor; UART; 5 V supply	PM1.0, PM2.5, PM10; PM2.5 also feeds the AQI

Module	Specification	Role in the system
MQ6	Semiconductor sensor for LPG and similar gases; analog output to ADC	Leak detection (not a laboratory concentration meter)
OLED	1.3-inch, 128×64, I ² C	Three rotating local display pages
Buzzer	Digital GPIO output	Audible gas alarm
Power	3.7 V, 2000 mAh Li-ion cell with switch and charging/booster module	Portable operation

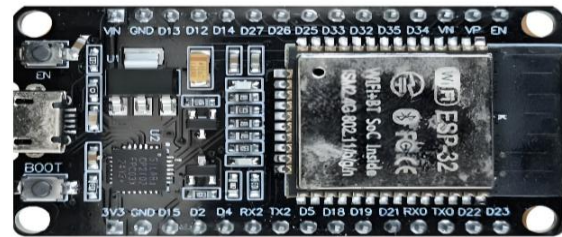


Fig. 2: ESP32 DevKit

The SCD40 (Fig. 3) measures CO₂ using the photoacoustic principle and reports CO₂, temperature and relative humidity over I²C [10]. A steady rise in CO₂ is a reliable sign of poor ventilation in an occupied room, and, as noted in Section 1.1, elevated CO₂ has been linked to poorer decision-making [4]. For this reason, CO₂ is one of the key model inputs.



Fig. 3: SCD40 sensor module

The PMS5003 (Fig. 4) runs on a 5 V supply and sends PM1.0, PM2.5 and PM10 values to the ESP32 as a serial data frame over UART.



Fig. 4: PMS5003 sensor

The MQ6 (Fig. 5) serves as a leak detector, and its analog output is connected to one of the ESP32's ADC pins.



Fig. 5: MQ6 gas sensor

The SCD40 and the OLED share one I²C bus with the ESP32 (SDA on D21, SCL on D22). The PMS5003 is connected through UART2 (RX on GPIO16, TX on GPIO17), and the MQ6 is read through an ADC pin. Table 3 lists the connections, and Fig. 7 shows the circuit diagram.

Table 3: Hardware interfaces

Module	ESP32 interface	Function
SCD40	I ² C (D21/D22)	CO ₂ , temperature, humidity
PMS5003	UART2 (16/17)	PM1.0, PM2.5, PM10
MQ6	ADC input	Gas abnormality
OLED	I ² C (D21/D22)	Local display
Buzzer	GPIO	Gas alarm
Wi-Fi	Built-in	Web dashboard

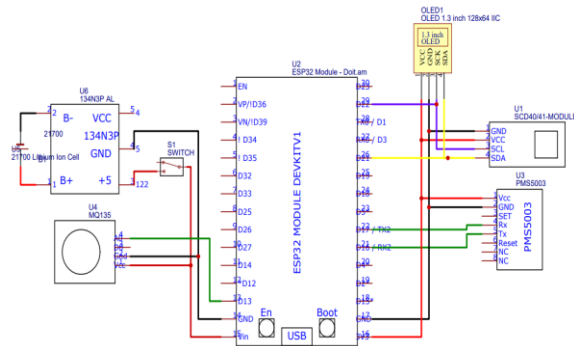


Fig. 6: Circuit diagram

Dataset and input data

The model has seven inputs: CO₂, temperature, humidity, PM1.0, PM2.5, PM10 and the MQ6 reading. Only the first six were used in training. The seventh input is reserved for the gas sensor; its weights are set to zero, so it has no effect on the classification.

The training data come from a synthetic indoor air quality dataset created specifically for this work, referred to here as SIAQ-3200. The dataset holds 3,200 labelled samples of the six quantities: 555 GOOD, 687 MODERATE, 765 POOR and 1,193 VERY POOR, with no missing values and no duplicate rows. Table 4 shows the range of values in each class. Because the ranges of neighbouring classes overlap, the classes cannot be separated perfectly. A stratified 80:20 split produced 2,560 training samples and 640 test samples. The test samples were set aside and used only once, for the final evaluation.

Table 4: Value ranges per class in the synthetic dataset

Class	CO ₂ (ppm)	PM2.5 (µg/m ³)	PM10 (µg/m ³)
GOOD	400–796	0–15	0–30
MODERATE	400–1195	0–35	0–75
POOR	452–1996	6–75	13–150
VERY POOR	713–3730	6–201	21–368

Data acquisition and preprocessing

The firmware reads every sensor once every five seconds. A PMS5003 frame is accepted only after its header, length and checksum have been checked. Because the MQ6 output is noisy, ten ADC samples are averaged for each reading. The seven values together form one feature vector, which is normalised as

$$X' = (X - \mu)/S \quad \text{---- (1)}$$

where the mean μ and the scale S are stored with the trained model and were computed from the training samples only.

TinyML model

A small feed-forward network was chosen because it fits easily in the ESP32's memory and needs no special runtime, in line with the common practice of running inference directly on the microcontroller [11]. The 7-

node input layer feeds a dense hidden layer of 12 neurons with ReLU activation, followed by a 4-neuron output layer whose values are converted into probabilities by a Softmax function. The class with the highest probability is taken as the prediction, and that probability is shown as the confidence. For example, outputs of 10 %, 15 %, 65 % and 10 % would give POOR with 65 % confidence. The trained weights are stored in a header file (`air_quality_tinyml_model.h`) that is compiled into the firmware, so the device needs neither a Python runtime nor a server.

Training was carried out offline in Python using scikit-learn's `MLPClassifier` with the Adam optimiser (learning rate 0.01, L2 penalty 10^{-4}), cross-entropy loss and a limit of 2,000 iterations. Training stopped after 177 iterations, once the loss no longer improved. These settings were selected by 5-fold cross-validation on the training samples alone, which gave a mean accuracy of about 87.6 %. The network has 136 trained parameters. Adding the 12 zero weights of the reserved input and the 14 normalisation constants gives 162 floating-point constants in the header, roughly 0.65 KB, and a single prediction takes about 130 multiply-accumulate operations.

AQI calculation

The AQI is calculated from PM_{2.5} by linear interpolation between concentration breakpoints [12]:

$$AQI = [(I_{high} - I_{low}) / (C_{high} - C_{low})] \times (C - C_{low}) + I_{low} \quad -- (2)$$
 where C is the measured PM_{2.5} concentration, C_{low} and C_{high} are the breakpoints on either side of it, and I_{low} and I_{high} are the corresponding index values. This AQI reflects particulate matter only, whereas the TinyML class draws on all the inputs, so the two outputs complement each other rather than repeat the same information.

Gas-safety layer

The averaged MQ6 reading is compared against a fixed threshold of 2000 on the 0–4095 ADC scale. When the reading reaches or exceeds this value, the system raises a gas alert, toggles the buzzer every 500 ms using `millis()` timing and shows a priority warning on the display. The alert clears once the reading drops back below the threshold. In the current model file, the weights on the MQ6 input are zero, so the gas value does not yet affect the classification and the alarm

relies entirely on the threshold rule. Section 5.4 treats this as a limitation.

Management logic

A rule layer maps the predicted class to advice, as shown in Table 5. During a gas alert, the fan and purifier advice switches to HIGH and the window is shown as closed.

Table 5: Recommendations per air-quality class

Class	Fan	Purifier	Window
GOOD	OFF	OFF	Optional
MODERATE	ON	ON	Check outdoor air
POOR	ON	ON	Closed

Software and tools

The firmware is written in C/C++ in the Arduino IDE with the ESP32 core, using the WiFi, WebServer, Wire, Adafruit GFX/SSD1306 and SensirionI2cScd4x libraries. Separate non-blocking `millis()` timers handle sensing, Wi-Fi reconnection, OLED page changes and the buzzer, so no task holds up another. The OLED cycles through three pages: the SCD40 values, then the particulate and gas values, and finally the AQI with the TinyML result. A web server on port 80 hosts a dashboard that refreshes every 5 seconds and shows all readings, the class, the confidence, the gas status and the recommendations. If the Wi-Fi connection drops, classification carries on locally.

Custom PCB

To remove loose wiring, a PCB was designed in EasyEDA and made in-house. The track pattern was transferred onto a copper-clad board, which was then etched, drilled and soldered. Before power-up, the board was checked with a multimeter for continuity and short circuits. Each sensor plugs into its own connector, so it can be replaced easily.

IV. RESULTS AND DISCUSSION

Evaluation procedure

The classifier was evaluated on the 640 test samples, which played no part in training or in choosing the training settings (Fig. 8). The assembled prototype was tested first module by module and then as a complete unit, covering every sensor and bus, the AQI routine, the OLED pages, the dashboard, the model output and

the buzzer. The values on the dashboard, the OLED and the serial monitor were compared to confirm that all three received the same processed data.

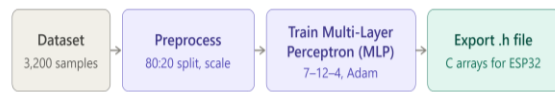


Fig. 8. Workflow for generating the model header file (.h)

Classification performance

On the test set, the network achieved 87.3 % accuracy, compared with 90.0 % on the training set. The macro-averaged precision, recall and F1-score were 0.86, 0.87 and 0.86, respectively. Table 6 lists the per-class results, and Fig. 9 shows the confusion matrix.

Table 6: Test-set performance per class

Class	Precision	Recall	F1-score	Samples
GOOD	0.797	0.955	0.869	111
MODERATE	0.857	0.745	0.797	137
POOR	0.847	0.830	0.838	153

True class \ Predicted class	GOOD	MOD.	POOR	V. POOR
GOOD	106	5	0	0
MOD.	27	102	8	0
POOR	0	12	127	14
V. POOR	0	0	15	224

Fig. 9: Confusion matrix on the test set

Every one of the 81 misclassified samples was assigned to a neighbouring class. For instance, 27 of the 137 MODERATE samples were predicted as GOOD, 12 POOR samples as MODERATE and 14 as VERY POOR, and 15 VERY POOR samples as POOR. No GOOD sample was labelled POOR or VERY POOR, and no VERY POOR sample was labelled GOOD or MODERATE.

The inference routine from the firmware was also compiled and run on a PC. For all 640 test samples it produced the same class as the Python model, with a maximum probability difference of about 10^{-6} , and its output stayed the same when the MQ6 input was set to a high value.

Functional verification

Table 7 summarises the checks carried out on the assembled unit.

Table 7: Functional verification of the prototype

Function	What was checked	Outcome
Sensor interfaces	Data received over I ² C, UART and ADC; PMS5003 frame verified by header, length and checksum	(Fig.10)
AQI routine	PM2.5-based AQI computed breakpoints from	(Fig.11)
OLED	Three pages rotate every 2 s	(Fig.11)
Web dashboard	Readings, class, confidence, class probabilities, gas status and advice refresh every 5 s	(Fig. 11)
Gas alarm	Below the threshold: gas normal, buzzer off. At or above it: alert shown and buzzer beeping; the alarm clears when the reading falls back	(Fig.12)
Data consistency	Dashboard, OLED and serial monitor show identical values	(Fig. 12)

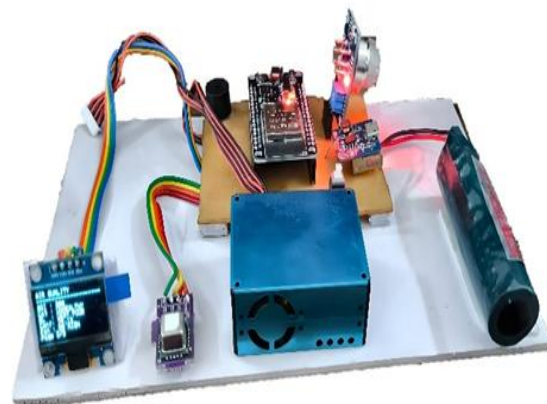


Fig.10: Sensor Interfaces

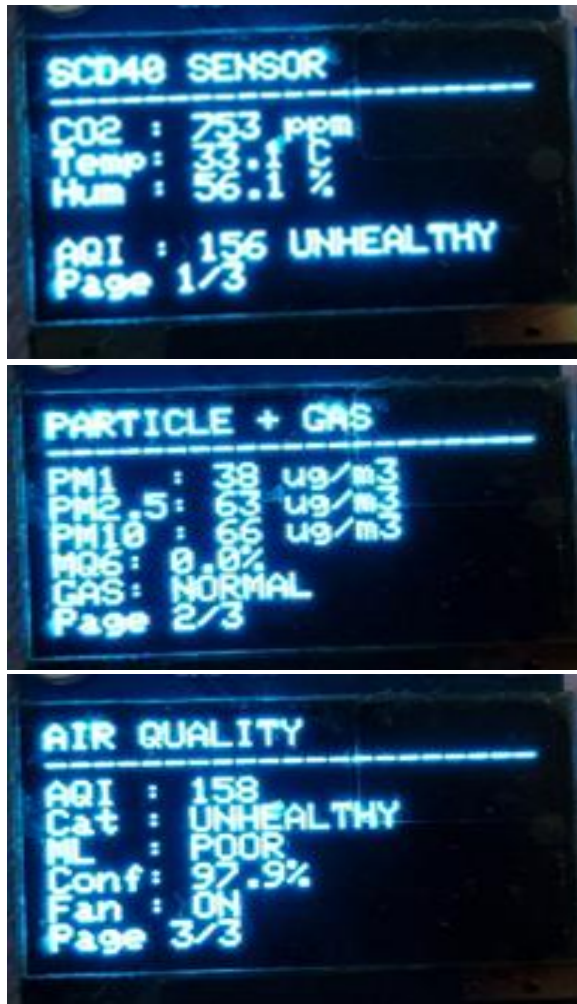


Fig.11: OLED Display

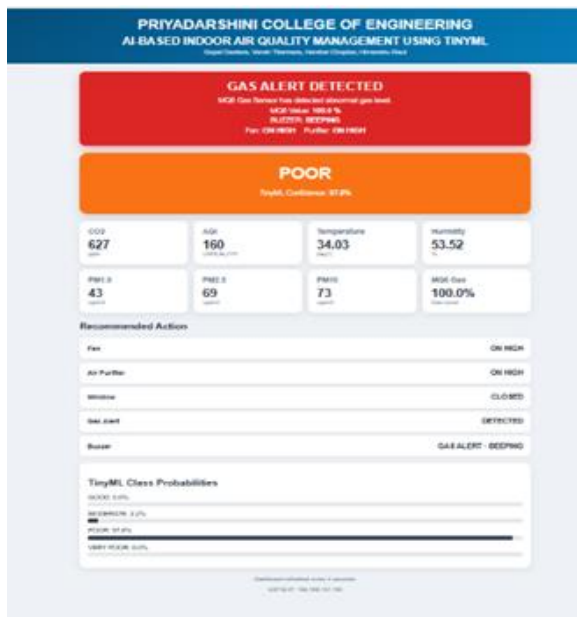


Fig. 12: Web Dashboard

DISCUSSION

Interpretation of the results

The four-level output is useful because it brings several signals together. CO₂ indicates ventilation, PM1.0–PM10 indicate dust and smoke, and temperature and humidity describe comfort; no single reading gives the same overall picture. Twelve hidden neurons are enough to combine these inputs, and this small size is why the network fits comfortably on the ESP32.

The errors mirror the structure of the data. Classes that are far apart are never confused; mistakes occur only between neighbouring classes (Fig. 11). This is to be expected, since the value ranges of neighbouring classes overlap (Table 4). MODERATE samples, for example, can have PM2.5 values as low as those of GOOD samples, which explains why 27 of them were predicted as GOOD. The small gap between training accuracy (90.0 %) and test accuracy (87.3 %) suggests that the network is not overfitting. The AQI and the class are related but not the same: the AQI depends on PM2.5 alone, while the class also takes CO₂, temperature and humidity into account. A high confidence value means the reading sits well inside one class, whereas a lower value indicates a condition close to the boundary between two classes.

Comparison with the literature

Ken et al. [5] reported an R^2 of about 0.92 for ozone prediction from three inputs, and Wardana et al. [7] reported an R^2 above 0.70 for their predictor. Both papers deal with regression of a single quantity, while our node performs four-class classification over several inputs, so the figures cannot be compared directly. Datta et al. [8] achieved 75.2 % accuracy for AQI prediction, which fell to about 60 % in real-time validation. Our 87.3 % is higher, but the comparison carries little weight: the tasks are different, and our figure comes from synthetic data, which are usually easier to classify than real sensor readings. The drop reported in [8] between test and real-time conditions suggests that our accuracy on real measurements would also be lower. What this work adds to these studies is a decision layer that turns the result into advice, together with a separate gas-safety path.

Strengths

Compared with the cited approaches, the system has three strengths: it combines particulate, CO₂, climate

and gas sensing on one board, it works entirely offline, and its safety alarm does not depend on the model.

Limitations

The most important limitation is that the model was trained and tested only on synthetic data. The reported accuracy therefore shows how well the network learns the defined classes, not how well it performs in real rooms. Further limitations are the uncalibrated low-cost sensors and the unused MQ6 weights. The MQ6 responds to several combustible gases and cannot tell which one is present, and its fixed threshold of 2000 has not been validated across different temperatures and humidity levels. Finally, the prototype tests were carried out on a single unit.

Practical implications

The unit is suitable for homes, classrooms, offices, laboratories and public halls, and it can also serve as a teaching platform for IoT and TinyML. In hospitals or industrial settings, it should be used only as a supplementary monitor, not as a replacement for certified instruments.

V. CONCLUSION

Low-cost indoor air monitors usually show numbers without telling people what to do about them, and many of them depend on the cloud. To address this, we built a node around a single ESP32 that measures CO₂, temperature, humidity, particulate matter and combustible gas. An embedded neural network places the room in one of four air-quality levels, a PM2.5-based AQI is calculated, and the node suggests when to run a fan or purifier or open a window. A separate threshold rule warns of gas leaks. On a held-out synthetic test set, the network achieved 87.3 % accuracy, and every error fell between neighbouring classes. The prototype also behaved consistently across its display, dashboard and alarm. The main contribution is a complete offline pipeline that senses, classifies, advises and warns using inexpensive hardware. Such a device can help homes, classrooms and offices ventilate in time, although certified instruments remain necessary in critical settings. Overall, the results show that TinyML can turn a simple low-cost sensor node into a practical tool that not only monitors indoor air but also guides people towards healthier and safer indoor spaces.

REFERENCES

- [1] U. Satish *et al.*, “Is CO₂ an indoor pollutant? Direct effects of low-to-moderate CO₂ concentrations on human decision-making performance,” *Environmental Health Perspectives*, vol. 120, no. 12, pp. 1671–1677, 2012, doi: 10.1289/ehp.1104789.
- [2] J. Saini, M. Dutta, and G. Marques, “Indoor air quality monitoring systems based on Internet of Things: A systematic review,” *International Journal of Environmental Research and Public Health*, vol. 17, no. 14, Art. no. 4942, 2020, doi: 10.3390/ijerph17144942.
- [3] P. P. Ray, “A review on TinyML: State-of-the-art and prospects,” *Journal of King Saud University—Computer and Information Sciences*, vol. 34, no. 4, pp. 1595–1623, 2022, doi: 10.1016/j.jksuci.2021.01.007.
- [4] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. Sebastopol, CA, USA: O’Reilly Media, 2019.
- [5] H. M. Ken, M. Behjati, A. S. Rafsanjani, S. Aslam, Y. K. Meng, A. P. P. Abdul Majeed, and Y. Zheng, “Advancing air quality monitoring: TinyML-based real-time ozone prediction with cost-effective edge devices,” *arXiv preprint arXiv:2504.03776*, 2025.
- [6] A. M. Hayajneh, M. Hafeez, S. A. R. Zaidi, and D. McLernon, “TinyML empowered transfer learning on the edge,” *IEEE Open Journal of the Communications Society*, vol. 5, pp. 1656–1672, 2024, doi: 10.1109/OJCOMS.2024.3373177.
- [7] I. N. K. Wardana, S. A. Fahmy, and J. W. Gardner, “TinyML models for a low-cost air quality monitoring device,” *IEEE Sensors Letters*, vol. 7, no. 11, Art. no. 6007804, 2023, doi: 10.1109/LSSENS.2023.3315249.
- [8] A. Datta, A. Pal, R. Marandi, N. Chattaraj, S. Nandi, and S. Saha, “Efficient air quality index prediction on resource-constrained devices using TinyML: Design, implementation, and evaluation,” in *Proc. 25th Int. Conf. Distrib. Comput. Netw. (ICDCN)*, 2024, pp. 304–309, doi: 10.1145/3631461.3631956.
- [9] A. P. Plageras *et al.*, “A gas leakage detection device based on the technology of TinyML,” *Technologies*, vol. 11, no. 2, Art. no. 45, 2023.

- [10] Sensirion AG, “SCD4x datasheet: Miniature CO₂, temperature and humidity sensor module,” Stäfa, Switzerland.
- [11] R. David *et al.*, “TensorFlow Lite Micro: Embedded machine learning for TinyML systems,” in *Proc. Mach. Learn. Syst. (MLSys)*, vol. 3, 2021.
- [12] U.S. Environmental Protection Agency, “Technical assistance document for the reporting of daily air quality—the Air Quality Index (AQI),” Rep. EPA-454/B-18-007, Research Triangle Park, NC, USA, 2018.