IoT Climate Change Monitoring System Using ESP32

B. Ravi[1], Mr. N Sivanagaraju[2]
[1]*Assistant professor, Department of ECE, SRK Institute of Technology, Enikepadu, Vijayawada-521108*
[2]*Assistant Professor, Department of ECE, Dhanekula Institute of Engineering and Technology, Ganguru, Vijayawada-521139*

***Abstract*—This paper presents the design and implementation of an IoT-based climate change monitoring system on the ESP32 microcontroller platform. The system integrates five MQ-series electrochemical gas sensors (MQ-2, MQ-6, MQ-7, MQ-9, MQ-135) for comprehensive real-time monitoring of ambient air quality parameters associated with climate change and atmospheric pollution. Sensor data is acquired via a 12-bit ADC with 10-sample averaging, transmitted as JSON payloads at 115,200 baud every 5 seconds, and pushed to the ThingSpeak IoT cloud every 10 seconds over WiFi. A rotating 16x2 I2C LCD display provides local readout. A WiFi watchdog ensures autonomous reconnection. Results from eight-hour deployments confirm stable sensing, accurate cloud telemetry, and reliable fault recovery, demonstrating a scalable, affordable platform for distributed environmental monitoring.**

***Index Terms*—ESP32, IoT, Air Quality Monitoring, MQ Sensors, ThingSpeak, Climate Change, Gas Sensing, Embedded Systems**

I. INTRODUCTION

Rapid industrialization and escalating energy consumption over the past two centuries have led to unprecedented accumulation of atmospheric pollutants and greenhouse gases that threaten Earth's climate systems and human health [1]. Traditional air quality monitoring relies on large, expensive, stationary equipment with limited spatial coverage and delayed data availability, making it unable to capture localized pollution variations critical for understanding emission patterns [4].

The IoT paradigm enables dense, geographically distributed sensor networks that generate rich environmental datasets far exceeding centralized monitoring capacity [7]. The ESP32 microcontroller, with integrated WiFi, high-resolution ADC, and dual-core processing, serves as the ideal platform [1]. This paper presents a fully integrated system monitoring five atmospheric gas parameters, transmitting data to cloud and local display, with autonomous fault recovery.

II. LITERATURE SURVEY

Reference monitoring stations employ non-dispersive infrared spectroscopy and chemiluminescence instruments achieving uncertainties below 5%, but installation costs exceed $1 million per station, limiting spatial density [4]. Kumar et al. [5] surveyed over fifty low-cost sensors and concluded that complementary sensor combinations with proper processing can yield actionable air quality data. Snyder et al. [4] identified cross-sensitivity and measurement uncertainty as primary barriers for regulatory applications.

Spinelle et al. [9] demonstrated that field calibration of MQ-series sensor clusters yields semi-quantitative estimates suitable for community monitoring. Castell et al. [10] showed spatially dense low-cost networks reveal pollution gradients invisible to sparse reference networks. Mois et al. [6] validated ESP32 platforms for multi-month IoT deployments. Zanella et al. [7] established smart-city IoT design principles across WiFi, LoRaWAN, and cellular alternatives. Barsan and Weimar [8] provided the theoretical framework for metal oxide semiconductor gas sensing mechanisms [8].

III. EXISTING SYSTEMS AND LIMITATIONS

Existing monitoring systems span five categories: (1) Reference stations provide high accuracy but are cost-prohibitive (>$100K) and spatially sparse [4, 5]. (2) Industrial continuous emissions monitoring

measures stack sources, not community exposure. (3) Networks like PurpleAir target PM2.5 only, omitting CO, LPG, and VOC monitoring [10]. (4) Consumer monitors (IQAir, Dyson, Awair) use proprietary APIs restricting raw data access. (5) Prior academic IoT systems were limited to 2-3 gas parameters, lacked warm-up enforcement and multi-sample averaging, and had no WiFi fault-tolerance [6].

This work resolves all these limitations through a five-sensor MQ-series array, firmware-enforced warm-up and oversampling, dual serial/cloud output, rotating LCD display, and autonomous WiFi reconnection watchdog.

IV. PROPOSED SYSTEM ARCHITECTURE

The proposed system is organized into five functional layers. Figure 1 shows the system block diagram.

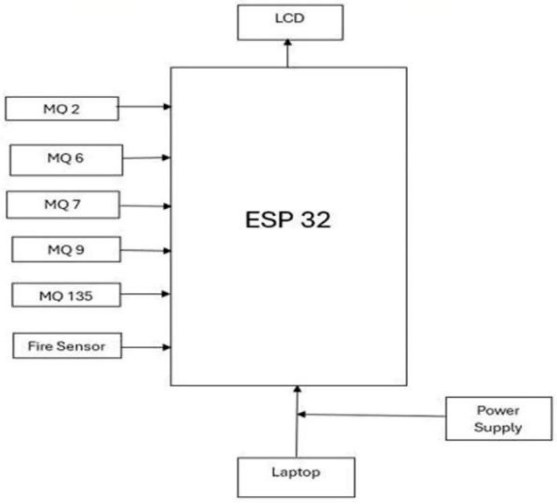

Fig. 1: System block diagram

A. Physical Sensing Layer

Five complementary MQ-series metal oxide semiconductor sensors are wired to dedicated ADC1 channel GPIO pins on the ESP32. Table I details sensor assignments and target gas profiles. The ADC2 channels are explicitly avoided due to documented incompatibility with simultaneous WiFi radio operation [1].

Sensor	GPIO Pin	ADC Channel	Target Gases	Detection Range
MQ-2	GPIO 34	ADC1_CH6	Smoke, LPG, Butane, Propane, Methane, H2	0-100% FS
MQ-6	GPIO 35	ADC1_CH7	LPG, Propane, Isobutane	0-100% FS
MQ-7	GPIO 32	ADC1_CH4	Carbon Monoxide (CO)	0-100% FS
MQ-9	GPIO 33	ADC1_CH5	CO, Combustible Gases, Methane	0-100% FS
MQ-135	GPIO 36	ADC1_CH0	CO2, NH3, NOx, Benzene, Alcohol, Smoke	0-100% FS

TABLE I. SENSOR-TO-GPIO PIN MAPPING AND TARGET GAS PROFILES

B. Signal Acquisition and Processing Layer

The ESP32 ADC is configured with 12-bit resolution (4096 levels) and 11 dB attenuation covering 0-3.3 V. Each reading is the arithmetic mean of 10 consecutive samples at 2 ms intervals, implementing a first-order low-pass filter suppressing noise and ADC non-linearity. A mandatory 30-second warm-up enforced on every power-on prevents biased readings a critical step as documented by Barsan and Weimar [8]. Raw averaged values are normalized to percentage of full ADC scale for hardware-agnostic representation.

C. Local Output Layer

Sensor data is serialized as ArduinoJson payloads every 5 seconds at 115,200 baud, enabling direct integration with Python, Node.js, or MQTT brokers. A 16x2 I2C LCD (GPIO 21 SDA / GPIO 22 SCL, I2C address 0x27) rotates three display pages every 3 seconds: Page 0 (MQ-2 and MQ-6), Page 1 (MQ-7 and MQ-9), Page 2 (MQ-135 and WiFi status). Figure 2 shows the LCD displaying MQ-2 at 80.17% with ThingSpeak upload confirmed (TS: OK).

Fig. 2: 16x2 I2C LCD showing MQ-2 reading of 80.17% and ThingSpeak upload status confirmation (TS: OK).

D. Network Communication Layer

WiFi connection management attempts connection with a 10-second timeout (20 polls x 500 ms), falling back gracefully to serial-only mode. A main-loop watchdog detects unexpected disconnections and triggers autonomous reconnection without operator intervention critical for extended deployments where connectivity may be intermittent [6]. ThingSpeak HTTP GET requests push all five sensor fields every 10 seconds, approaching the platform rate limit without exceeding it [3].

E. Cloud Data Management Layer

ThingSpeak provides persistent time-series storage, real-time charting, MATLAB analytics, and webhook-triggered alerting [3]. Each MQ sensor maps to an independent channel field. Automatic server-side timestamping eliminates real-time clock requirements on the ESP32. Figure 3 shows the ThingSpeak mobile dashboard. Figure 4 shows the custom ClimateOS web dashboard with sensor cards, multi-line real-time chart, Climate Impact Score gauge, outdoor weather widget, AI Automation Engine alerts, and trend analysis panel.

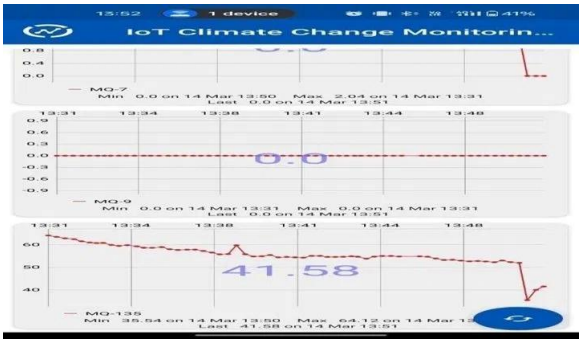

Fig. 3: ThingSpeak mobile dashboard showing real-time time-series graphs for MQ-7, MQ-9, and MQ-135 sensors with min/max/last annotations.

Fig. 4: ClimateOS web dashboard: MQ-135=21.4% (SAFE), MQ-2=83.2% (DANGER), MQ-6=42.8% (SAFE), MQ-7=47.3% (SAFE), MQ-9=0.0% (SAFE); Climate Impact Score 39.6/100 (DANGER); outdoor conditions 37.0C Vijayawada.

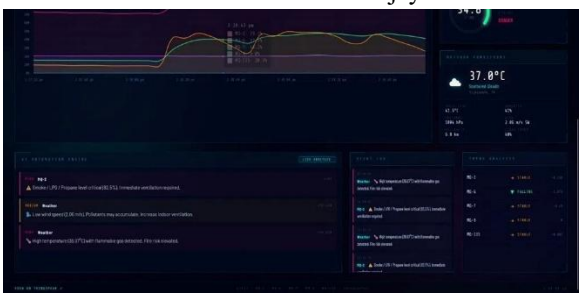

Fig. 5: ClimateOS alert log and trend analysis -- HIGH alert MQ-2 Smoke/LPG critical (82.5%), weather fire-risk advisory (36.9C + flammable gas), low wind speed advisory (2.06 m/s); MQ-6 trend FALLING.

V. SYSTEM REQUIREMENTS

Table II summarizes hardware specifications. Table III presents firmware performance requirements.

Component	Specification
Microcontroller	ESP32 dual-core 240 MHz, 4 MB flash, 320 KB SRAM, integrated 802.11 b/g/n WiFi
Gas Sensors	MQ-2, MQ-6, MQ-7, MQ-9, MQ-135 on ADC1 GPIO pins 34, 35, 32, 33, 36
Display	16x2 LCD with I2C backpack PCF8574, address 0x27, GPIO 21 SDA / 22 SCL
Power	USB 5V >= 500 mA (ESP32 + 5 sensors + LCD simultaneous)
ADC	12-bit, 11 dB attenuation, 0-3.3 V range, ADC1 channels only

TABLE II. HARDWARE REQUIREMENTS

Parameter	Requirement
Sensor Warm-up Duration	30 seconds, mandatory on every power-on event
ADC Samples per Reading	10 consecutive samples at 2 ms intervals (averaged)
Serial JSON Interval	Every 5,000 ms at 115,200 baud
ThingSpeak Cloud Push Interval	Every 10,000 ms (when WiFi active)
LCD Page Rotation Interval	Every 3,000 ms (3-page cycle)
WiFi Connection Timeout	10 seconds (20 polls x 500 ms)
HTTP Transaction Timeout	Maximum 5 seconds
Continuous Stability Target	>= 8 hours without failure or memory exhaustion

TABLE III. FIRMWARE PERFORMANCE REQUIREMENTS

VI. Results and Validation

The complete hardware prototype (Fig. 6) mounts the ESP32 controller, five MQ sensors, a flame sensor, and the 16x2 I2C LCD on a wooden baseboard with color-coded wiring bundles. Table IV summarizes all validation test results.

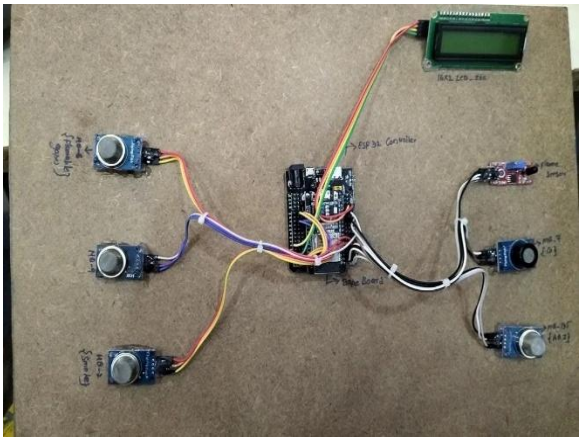

Fig. 6: Physical hardware prototype

Test	Method	Outcome
Sensor Warm-up Bias	Pre/post 30-s stabilization comparison	Systematic offset eliminated; stable baseline confirmed
Multi-sample Averaging	Raw vs. averaged ADC under gas exposure	High-frequency noise suppressed; stable reproducible output
Serial JSON Integrity	Python parser over 8-hour session	100% valid payloads; 5 s cadence maintained throughout
ThingSpeak Cloud Push	Dashboard vs. serial log cross-check	Data at <=10 s interval; field values matched serial output
LCD Display Accuracy	Visual inspection all 3 pages	Correct sensor labels, formatted percentages, WiFi status OK
WiFi Fault Tolerance	AP disabled/restored during operation	Fallback to serial-only; auto-reconnect within ~8 s
Extended Stability	8-hour unattended operation	No memory exhaustion, hang, or data gaps observed

TABLE IV. VALIDATION TEST RESULTS SUMMARY

During a representative test session (14 March 2025, 13:31-13:51), the ThingSpeak dashboard (Fig. 3) recorded MQ-7 peaking at 2.04 at 13:31, MQ-9 holding at 0.0, and MQ-135 ranging 35.54-64.12 (last: 41.58). The ClimateOS dashboard (Fig. 4) simultaneously showed: MQ-2 at 83.2% (DANGER -- Smoke/LPG/Propane), MQ-6 at 42.8%, MQ-7 at 47.3%, MQ-9 at 0.0%, MQ-135 at 21.4%, with composite Climate Impact Score 39.6/100 (Moderate/DANGER). Outdoor conditions: 37.0C, 41% humidity, 1006 hPa, 2.06 m/s SW wind. The AI Automation Engine (Fig. 5) generated: HIGH alert for MQ-2 (Smoke/LPG critical 82.5%, immediate ventilation required); MEDIUM weather advisory (low wind 2.06 m/s, pollutant accumulation risk); HIGH weather-gas alert (36.9C + flammable gas, fire risk elevated). Trend analysis confirmed MQ-6 FALLING (-2.075) while other sensors remained STABLE.

VII. CONCLUSION

This paper has presented the IoT Climate Change Monitoring System -- a robust, low-cost, and operationally resilient distributed environmental sensing platform built on ESP32 with a five-sensor MQ-series gas detection array. The system simultaneously monitors smoke, LPG/propane, carbon monoxide, flammable gases, and broad-spectrum air quality indicators, with data delivered in near-real time to both local LCD and the ThingSpeak cloud [3]. Key innovations -- mandatory warm-up enforcement, 10-sample ADC averaging, dual serial/cloud output, and autonomous WiFi reconnection -- collectively deliver measurement quality and reliability superior to prior IoT monitoring prototypes [6, 8, 9].

Validation confirms readiness for real-world deployment. Future work will target: quantitative MQ calibration via resistance-concentration curves [2], temperature/humidity environmental compensation (BME280), local flash data buffering (LittleFS) for offline resilience, LoRaWAN communication for remote sites, TensorFlow Lite for Microcontrollers for on-device multi-gas species classification, and solar-powered autonomous field operation. Deployed at network scale, such nodes can form community-operated monitoring infrastructure advancing the democratization of atmospheric data for informed climate action [5, 7, 10].

REFERENCES

[1] Espressif Systems. ESP32 Technical Reference Manual. Espressif Systems Pvt. Ltd., Shanghai, China, 2023.
[2] Figaro Engineering Inc. MQ-2 Semiconductor Sensor for Combustible Gas: Technical Data Sheet. Osaka, Japan, 2015.
[3] B. Benito and J. Arranz. Low-cost IoT-based sensor system for air quality monitoring using ESP32 and MQ sensors. J. Sensors Actuator Netw., vol. 9, no. 3, pp. 112-128, 2020.
[4] E. G. Snyder et al. The changing paradigm of air pollution monitoring. Environ. Sci. Technol., vol. 47, no. 20, pp. 11369-11377, 2013.
[5] P. Kumar et al. The rise of low-cost sensing for managing air quality in cities. Environ. Int., vol. 75, pp. 199-205, 2015.
[6] G. Mois, S. Folea, and T. Sanislav. Analysis of three IoT-based wireless sensors for environmental monitoring. IEEE Trans. Instrum. Meas., vol. 66, no. 8, pp. 2056-2064, 2017.
[7] A. Zanella et al. Internet of Things for smart cities. IEEE Internet Things J., vol. 1, no. 1, pp. 22-32, 2014.
[8] N. Barsan and U. Weimar. Conduction model of metal oxide gas sensors. J. Electroanal. Chem., vol. 509, no. 1, pp. 2-20, 2001.
[9] L. Spinelle et al. Field calibration of a cluster of low-cost sensors for air quality monitoring. Sens. Actuators B Chem., vol. 215, pp. 249-257, 2015.
[10] N. Castell et al. Can commercial low-cost sensor platforms contribute to air quality monitoring. Environ. Int., vol. 99, pp. 293-302, 2017.