Journal of Electrical Engineering, Electronics, Control and Computer Science – JEEECCS, Volume 11, Issue 38, pages 9-16, 2025

Design and Implementation of an ESP8266-Based IoT System for Real-Time Classroom Environmental Monitoring

1st Stanciu Dumitru-Cătălin
Faculty of Electronics, Communications and Computers
National University of Science and Technology Politehnica Bucharest
Pitesti, Romania
cstanciuzedx01c@gmail.com

3rd Drăguşin Sebastian-Alexandru
Department of Electronics, Computers and Electrical Engineering
National University of Science and Technology Politehnica Bucharest
Pitesti, Romania
dragusin.sebi@yahoo.com

2nd Dinică Răzvan-Andrei
Faculty of Electronics, Communications and Computers
National University of Science and Technology Politehnica Bucharest
Pitesti, Romania
razvanandreidinica@gmail.com

4th Bizon Nicu
Department of Electronics, Computers and Electrical Engineering
National University of Science and Technology Politehnica Bucharest
Pitesti, Romania
nicubizon@yahoo.com

Abstract –This paper presents the design and validation of an Internet-of-Things (IoT) architecture for real-time monitoring of classroom environmental conditions. Centered on a Wi-Fi–enabled embedded node (ESP8266) and a lightweight edge-to-cloud pipeline, the system acquires, transmits, and visualizes multivariate sensor data through a MATLAB-based virtual instrument. The approach emphasizes low-cost components, minimal power and code footprint, and a modular interface that supports rapid reconfiguration of sensing channels and alert logic. A proof-of-concept deployment in representative classroom scenarios demonstrates reliable telemetry, stable user interaction, and actionable feedback for indoor comfort and safety management. The results indicate that the proposed stack offers a practical pathway for scalable educational IoT, bridging physical sensing and data-driven decision support while remaining accessible for laboratory, pilot, and instructional use.

Keywords- IoT, environmental sensing, Wi-Fi telemetry, MATLAB, virtual instrument, real-time data, indoor air quality, low-cost design

I. INTRODUCTION

Indoor environmental quality (IEQ) in educational spaces has a direct impact on students' cognitive performance, attendance, and well-being. Parameters such as temperature, relative humidity, carbon dioxide (CO_2), and ambient light influence comfort and ventilation adequacy, while energy constraints and building heterogeneity complicate continuous supervision. Despite increased awareness, many classrooms still lack affordable, real-time monitoring solutions that can be rapidly deployed, maintained with minimal expertise, and integrated into existing digital workflows for instructional or facility-management use [1].

Recent advances in the IoT have lowered barriers to pervasive sensing. Low-cost microcontrollers with integrated Wi-Fi—most notably the ESP8266—enable battery-friendly data acquisition, lightweight edge processing, and direct IP connectivity. Complementary software frameworks now make it practical to prototype end-to-end pipelines that ingest sensor streams, visualize trends, and trigger alerts without enterprise-grade infrastructure. Nevertheless, institutions frequently encounter gaps around reproducibility, interoperability with analysis tools used in teaching, and transparent calibration and validation practices suitable for academic environments [2].

This work addresses those gaps by designing and implementing a classroom-oriented IoT monitoring system that combines an ESP8266-based sensing node with a MATLAB graphical user interface (GUI) acting as a virtual instrument over Wi-Fi. The hardware platform collects essential environmental variables at configurable sampling rates and publishes timestamped measurements to a local network endpoint [2]. The MATLAB interface provides real-time visualization, basic quality checks, data logging, and export, offering an approachable bridge between embedded acquisition and the data analysis workflows commonly taught in STEM (Science, Technology, Engineering and Mathematic) curricula [2], [3].

Beyond assembling commodity components, the approach emphasizes a reproducible, open, and low-cost bill of materials; a communication scheme that is

simple to firewall and maintain within campus networks; calibration and sanity-check routines aligned with classroom use; and a user interface (UI) design that supports both pedagogical exploration and facility diagnostics. Collectively, these choices aim to facilitate rapid adoption in teaching laboratories while remaining credible for operational monitoring pilots.

The remainder of the paper is organized as follows. Section 2 reviews related literature on IoT environmental monitoring, typical sensing modalities, and UI and analytics approaches. Section 3 details the hardware architecture, sensors, power, and networking. Section 4 presents the software stack, including firmware design, the data protocol, and the MATLAB GUI used as a virtual instrument. Section 5 discusses results and system behavior under representative scenarios, and Section 6 concludes with limitations and avenues for future work.

II. Literature Review

A growing body of literature examines IoT-based environmental monitoring for occupied indoor spaces, with a consistent emphasis on low-cost microcontrollers (ESP8266/ESP32, Arduino-class MCUs (Microcontrollers)), commodity sensors for temperature, humidity, light, CO_2 and noise, and lightweight connectivity stacks (Wi-Fi with HTTP (Hypertext Transfer Protocol) / MQTT (Message Queuing Telemetry Transport)). Across studies, two recurring design axes emerge: (i) edge vs. cloud processing—ranging from simple on-node filtering and timestamping to full cloud analytics—and (ii) human–machine interfaces—spanning LabVIEW/MATLAB virtual instruments, custom web dashboards, and mobile apps for real-time visualization, alerting, and data archival. Typical challenges reported include sensor calibration and drift, sampling strategy vs. battery life, network reliability and QoS (Quality of Service), time synchronization, and the handling of privacy and security in educational settings. Recent works also highlight reproducible data pipelines (CSV (Comma-Separated Values) / JSON (JavaScript Object Notation) exports, REST (Representational State Transfer endpoints), interoperable messaging, and comparative evaluations of accuracy and latency under classroom-like dynamics (occupancy changes, door/window events). This chapter situates the present study within these trajectories, distilling common architectures, trade-offs, and best practices that inform our subsequent design and implementation choices.

The paper [4] reports a low-cost IoT air-quality node that couples an ATmega328 master with an ESP8266 Wi-Fi coprocessor, integrating MQ135 gas sensing and DHT11 temperature/humidity to stream calibrated measurements to a cloud dashboard (Blynk/ThingSpeak) and trigger threshold-based alerts; communication is handled via software-serial, and the authors validate an end-to-end pipeline from sensing to web visualization on a functional prototype.

A closely related strand of work is the IoT-enabled environmental control for small cultivation spaces, [5] present a low-cost mini-farm built around an ESP8266 (NodeMCU) that acquires temperature and humidity from dual DHT11 sensors and exposes both monitoring and actuation (brushless fan, thermoelectric cooler, ultrasonic mist) through a Blynk mobile dashboard; power delivery (12 V rails, LM317 regulation) and a relay/ESC (Electronic Speed Controller) stage enable closed-loop thermal–humidity management, validated with a two-week dataset and step-wise scenarios (monitor-only, TEC (Thermoelectric Cooler) on, TEC + ice packs) showing measurable set-point correction and stability improvements.

[6] presents the design and prototyping of an industrial temperature monitoring system built around a NodeMCU ESP8266 and a DS18B20 digital thermometer, using MQTT for lightweight publish–subscribe messaging and Node-RED for visualization. A Raspberry Pi 3B+ hosts the Mosquitto broker and the Node-RED dashboard, while the ESP8266 node periodically publishes temperature readings (every 10 seconds) to a designated topic; data are shown both on a serial console and via web gauges/plots. The authors outline hardware and software components (ESP8266 characteristics, DS18B20 1-Wire interfacing, Arduino IDE (Integrated Development Environment) workflow), detail the system architecture and Node-RED flows, and report low-latency, low-bandwidth operation in the prototype. They conclude by proposing future extensions such as adding more sensor types and nodes and powering the system with green energy.

The paper [7] presents an ESP8266-based air-quality station that measures PM2.5 alongside temperature, relative humidity, and CO, and streams results to Google Sheets with dashboards in Google Data Studio for visual analysis. The prototype uses NodeMCU (ESP8266) with low-cost sensors (dust sensor, DHT11, MQ-series gas sensor) and a simple flow from acquisition to cloud visualization, enabling real-time monitoring and alerting. In month-long trials, PM2.5 readings were compared against Thailand's Air4Thai reference data, yielding average percentage differences of ∼3.9% for PM2.5 and acceptable deviations for temperature/humidity, indicating adequate accuracy for community monitoring. The authors highlight practical deployment considerations (site selection, seasonality effects) and argue that the low cost and ease of integration make the approach suitable for broader urban or campus deployments.

The authors of [8] present a low-cost IoT solution for data-center temperature supervision built around an ESP8266-based wireless sensor network that streams multi-point readings to a Ubidots cloud dashboard with real-time SMS/email alerts when thresholds are exceeded. The prototype uses DHT11/DHT22 sensors—deployed redundantly per location for data validation—wired to ESP8266 nodes that handle both on-board processing (firmware developed in the Arduino IDE) and Wi-Fi connectivity to a router acting as the WSN access point. Temperature data are posted at minute-level intervals, visualized on the cloud, and drive configurable multi-level alarms; a >24-hour run demonstrated stable operation and timely notifications during induced thermal excursions, highlighting reduced total cost of ownership versus XBee/Arduino architectures.

III. Monitoring System Hardware Architecture

This section describes the physical architecture of the hardware platform implements a compact, networked sensing node designed for continuous, classroom-scale environmental monitoring. At its core, the system couples a Wi-Fi–enabled microcontroller with a heterogeneous sensor suite and a minimal set of actuators, delivering local feedback (LCD (Liquid Crystal Display) / buzzer) and remote telemetry over IEEE 802.11 b/g/n. The architecture emphasizes low bill-of-materials cost, simple assembly, and electrical robustness (level-compatible interfaces, shared reference ground, decoupled rails), while preserving measurement fidelity through appropriate signal conditioning and wiring discipline.

The processing and communication substrate is a NodeMCU development board built around the ESP8266EX SoC (System on Chip). This device provides a 32-bit Tensilica MCU (Microcontroller Unit), integrated $2.4GHz$ Wi-Fi MAC (Media Access Control) / PHY (Physical Layer), up to $160MHz$ clocking, GPIO (General-Purpose Input/Output) with interrupt capability, a 10-bit ADC (Analog-to-Digital Converter) (A0), and I²C (Inter-Integrated Circuit) / SPI (Serial Peripheral Interface) / UART (Universal Asynchronous Receiver-Transmitter) peripherals. In the proposed topology, the ESP8266 orchestrates periodic sampling, debounced event detection, local rule-based actuation, and MQTT/HTTP publishing. Timer-driven tasks and interrupt service routines are used to separate deterministic acquisition from asynchronous network activity [9].

The sensing layer targets the principal variables of indoor environmental quality and occupancy. For clarity, the main transducers and their electrical interfaces are enumerated below.

- **Temperature/Humidity (DHT11)**: one-wire digital protocol to a GPIO pin; sampling period $2s \div 5s$ to remain within the sensor's recommended duty cycle. The module's pull-up resistor is placed near the MCU to improve line integrity [10].
- **Combustible gas/smoke (MQ-2)**: analog voltage proportional to gas concentration, routed to A0 through a resistor divider matched to the ESP8266's $\sim 1.0V$ ADC range; the heater element is powered from $5V$ with warm-up time accounted for in firmware [11].
- **Ambient illuminance (LDR (Light Dependent Resistors) / KY-018)**: passive photoresistor in a voltage divider, read by A0 in a time-multiplexed scheme with the MQ-2 (firmware arbitrates ADC access and applies per-channel calibration curves) [12].
- **Acoustic level (KY-038)**: digital threshold output (DO) to a GPIO for noise events; optional analog channel (AO) can be profiled during calibration, but routine operation favors the debounced DO to limit ADC contention [13].
- **Vibration/Shock (SW-420)**: digital comparator stage delivers a clean logic output to a GPIO; an RC (Resistor–Capacitor) network provides chatter suppression at the source [14].
- **Proximity/Distance (HC-SR04)**: ultrasonic ranging with distinct TRIG/ECHO pairs; echo is captured via interrupt-timestamping to improve time-of-flight precision. One unit supports door/approach detection, the other coarse occupancy sensing [15].

The HMI (Human–Machine Interface) and actuation elements provide immediate, in-room feedback and automated mitigation:

- **LCD 16×2 (HD44780 with I²C backpack)**: connected to the ESP8266 via SDA/SCL; the I²C expander lowers pin count and permits bus co-existence with other I²C devices. The display presents current measurements, alarm states, and network status [16].
- **Active buzzer**: driven from a GPIO through a current-limiting resistor for audible alarms under threshold violations (e.g., excessive temperature or gas concentration) [17].
- **Relay module + $9V$ DC fan**: a transistor-isolated, opto-coupled relay board switches the fan supply under firmware control; flyback protection and separate $5V$ relay rail avoid MCU brown-outs during coil energization [18], [19].

The power-delivery network is built around a step-down DC–DC (Direct Current to Direct Current) buck converter that derives regulated $5V$ from a higher input source; the ESP8266 and all $3.3V$-only peripherals are supplied either from the NodeMCU's on-board regulator or a dedicated LDO, while high-draw loads (relay coil, sensor heaters, LCD backlight) remain on the $5V$ rail. All modules share a single reference ground; local $100nF$ decouplers are placed at each device, and star-ground routing minimizes return-path interference. Signal-integrity precautions include short TRIG/ECHO leads, series resistors on fast GPIO lines, and ADC anti-alias filtering where applicable [20].

From a systems viewpoint, the node executes a deterministic acquisition loop (sensor polling, time-stamped aggregation, plausibility checks) and a supervisory layer that handles alarm logic and network telemetry. Raw and derived metrics are rendered on the LCD, while event flags trigger the buzzer and relay. The ESP8266's network stack batches payloads to reduce channel occupancy and applies reconnection back-off to maintain graceful behavior in congested Wi-Fi environments.

The wiring diagram synthesizing the above interconnections—controller, sensors, HMI, relay-driven fan, and power module—is shown in Figure 1.

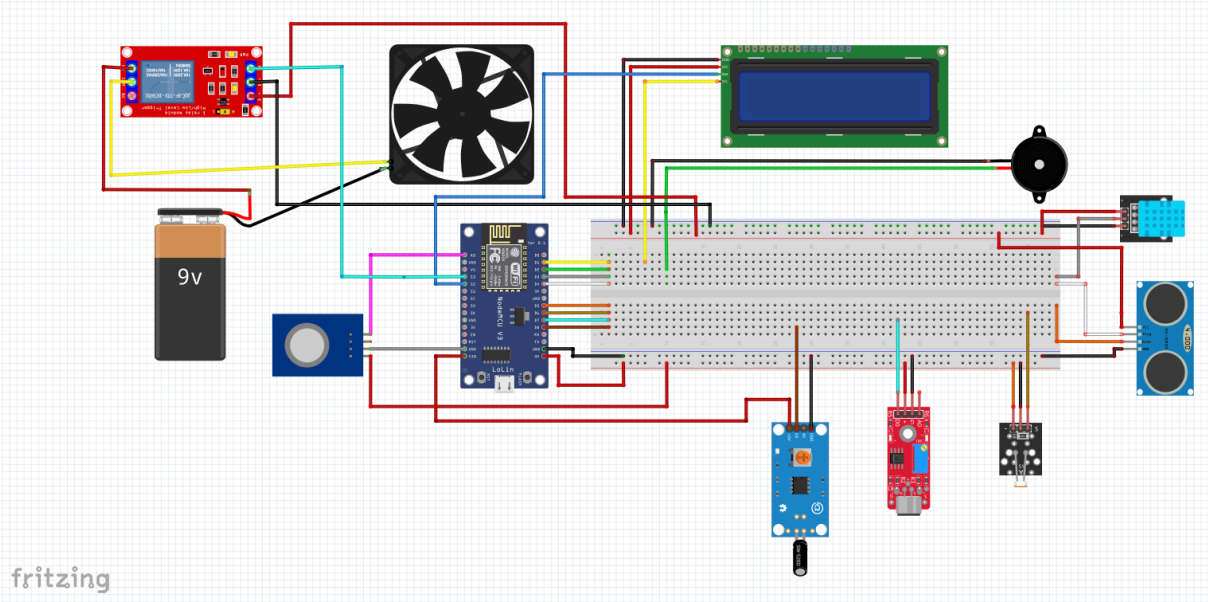

Figure 1. ESP8266-based classroom monitoring node: wiring diagram

IV. System Software Design

This chapter details the embedded and desktop software that enables end-to-end acquisition, transport, visualization, and supervisory control. The design follows a layered approach — *sensor/actuator drivers → data fusion & event logic → Wi-Fi transport → GUI presentation & control* — to ensure determinism on the microcontroller and responsive interaction in the user interface.

A. ESP8266 IoT firmware

The ESP8266 firmware is implemented as a non-blocking state machine in Arduino IDE (C/C++), using $setup()$ for one-time initialization and $loop()$ for cyclic tasks scheduled with $millis()$ timers [21]. The main software responsibilities are sensor sampling, data conditioning, local alarms, Wi-Fi telemetry, and reception of remote commands (fan, LEDs, buzzer).

- **Initialization**. On boot the MCU configures GPIO/ADC pins, initializes peripheral libraries (DHT, ultrasonic, MQ-2, LDR, SW-420, LCD, buzzer, relay), loads threshold values from flash (if present), and establishes a Wi-Fi connection (station mode) with exponential back-off and watchdog resets on persistent failure.
- **Acquisition & conditioning**. Each sensor has its own sampling period (e.g., $1s \div 2s$ for DHT, $100ms \div 200ms$ for acoustic/vibration/ultrasonic, $250ms \div 500ms$ for LDR/MQ-2). Readings are debounced and filtered with a short moving-average; boolean events (noise spike, vibration) use hysteresis and minimum hold times to avoid chatter. A lightweight plausibility check (range, NaN) guards against spurious values.
- **Event logic (edge computing)**. The firmware evaluates rules locally to ensure timely reaction without network dependency:
  - over-temperature or high gas → relay-driven fan ON and buzzer alert.
  - excessive noise or detected vibration → buzzer pulse.
  - proximity events → optional actions (e.g., door logic).
  - Events are time-stamped (epoch ms) to support GUI timelines.
- **Telemetry & remote control**. Measurements are serialized as a compact JSON/CSV frame and pushed over Wi-Fi using a simple REST POST or a persistent TCP (Transmission Control Protocol) socket (both options supported in the codebase). A typical JSON frame is:

```
{
  "t": 23.8,
  "rh": 41.2,
  "gas": 178,
  "light": 0.62,
  "sound": 0,
  "dist": 123,
  "vib": 0,
  "fan": 1,
  "buzz": 0,
  "ts": 1712665234123
}
```

The firmware also subscribes to short command messages (e.g., $\{"fan": 0, "buzz": 1\}$), validates them, and applies actuator updates with acknowledgment. All I/O (Inputs/Outputs) is non-blocking; time-critical paths use interrupts only where unavoidable.

- 🕮 **Reliability & security**. A reconnect routine restores links after AP (Access Point) changes; sequence numbers prevent out-of-order GUI updates; optional checksum guards payload integrity. For campus/LAN (Local Area Network) deployments, WPA2 (Wi-Fi Protected Access) plus an application token in the header is used; TLS (Transport Layer Security) can be enabled on capable toolchains.

B. MATLAB-based GUI

The desktop application is built in MATLAB as a virtual instrument that fuses data streams, renders real-time charts, logs datasets, and issues supervisory commands. The GUI, shown in Figure 2, opens a listener/socket (or polls an HTTP endpoint) to ingest MCU frames, parses the payload, and updates numeric indicators and strip-charts for Temperature, Humidity, Gas, Light, Sound, Distance, Vibration. Control widgets (buttons/toggles) publish JSON commands back to the ESP8266 for Fan/LED/Buzzer actuation.

Algorithmic flow (GUI):

1. Initialize UI components, networking object, and a circular buffer for each channel.
2. On message arrival: validate frame → parse fields → append to buffers → update numerical readouts and redraw plots; outliers are flagged but preserved in the log.
3. Compute derived metrics (e.g., rolling mean/max over 1–5 min) for stable visualization and annotate threshold crossings.
4. User actions (button presses) generate command frames with timestamps and a monotonic sequence; acknowledgments from the ESP are reflected in the status lamp ("System status" indicator).
5. Periodic tasks persist data to disk (MAT-files or CSV) and refresh connection health.

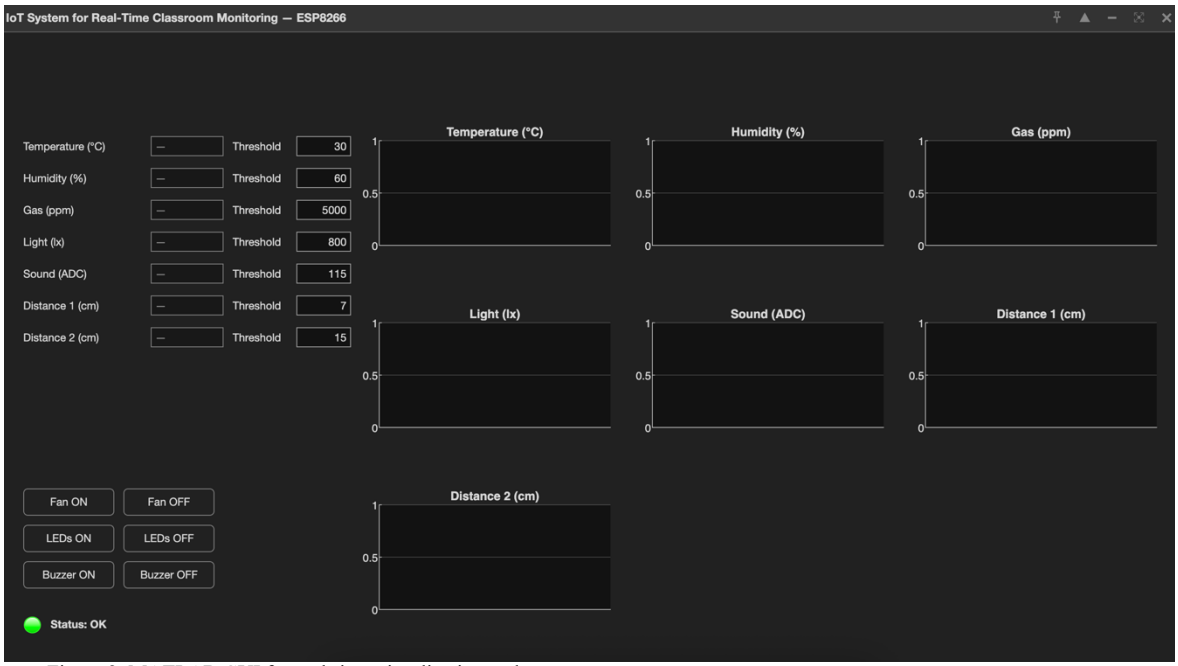

Figure 2. MATLAB GUI for real-time visualization and supervisory control of the ESP8266 classroom monitor

V. RESULTS

To evaluate the proposed IoT monitoring platform, we followed a staged test plan that combined bench validation with in-situ trials in a classroom mock-up. First, each sensing channel (temperature/relative humidity, gas, light, sound, vibration, ultrasonic distance) was verified individually against reference instruments and known stimuli (e.g., heat/cold packs, aerosol from a test spray, calibrated light and acoustic sources, discrete mechanical taps for the SW-420). Next, the full stack—ESP8266 firmware, Wi-Fi transport, and MATLAB GUI—was exercised end-to-end to assess acquisition stability, time synchronization, and command/actuation loops (fan via relay, buzzer, LEDs). Particular attention was given to start-up sequencing, reconnect logic after AP loss, and resilience to burst traffic when multiple widgets in the GUI simultaneously requested updates.

In continuous runs, the system sustained real-time streaming with responsive plots and controls; the GUI reflected state changes (e.g., fan ON/OFF, buzzer alarms) without noticeable lag, and no material packet loss was observed over extended logging sessions. Under rapid environmental changes (door open/close, directed airflow, localized sound bursts), the channels exhibited coherent responses and recovered cleanly after threshold-triggered alarms.

The physical integration of sensors around the miniature classroom facilitated repeatable scenarios (occupant approach, excessive noise, increased temperature), demonstrating that the platform can support both monitoring and simple supervisory control for indoor educational spaces.

testbed (ESP8266 node, sensors, relay-driven fan, LCD), can be observed in Figure 3, which we used to validate real-time monitoring and actuation.

Figure 3. Integrated GUI and physical testbed of the ESP8266-based classroom environmental monitoring system

CONCLUSION

The study has presented a complete, low-cost IoT platform for classroom environmental monitoring, built around an ESP8266 node and a MATLAB graphical interface.

After integration, the MATLAB GUI with live plots and controls, together with the classroom mock-up

The hardware integrates multi-domain sensing—temperature, humidity, gas concentration, light, sound, vibration and distance—together with local feedback and actuation via LCD, buzzer, LEDs and a relay-driven fan. On the software side, a compact C/C++ firmware manages time-stamped acquisition and reliable TCP transmission, while the MATLAB GUI

provides responsive visualization and supervisory control through live plots, status indicators and command buttons. Validation on a physical classroom mock-up confirmed end-to-end operation: streams were received in real time, commands were executed deterministically, and the system remained stable during extended runs with variable stimuli that emulate occupied classroom conditions.

Despite these results, several constraints remain. The current prototype operates as a single node, uses unencrypted TCP transport and relies on manual sensor calibration. Power delivery is bench-sourced rather than optimized for long-term deployment, and the user interface is desktop-bound, limiting accessibility and multi-site management. Control policies are intentionally simple and rule-based, and historical analytics are confined to session-level inspection rather than full life-cycle data management.

Future development will address these limitations along three axes: sensing fidelity, secure scalability and operational robustness. First, we plan to enrich the sensing stack with CO_2 / TVOC (Total Volatile Organic Compounds) and PM2.5 modules and to implement temperature- and humidity-compensated self-calibration, thereby improving air-quality inference and longitudinal comparability. Second, the communication layer will migrate to MQTT over TLS with certificate-based authentication, augmented by over-the-air firmware updates, watchdogs and structured telemetry for remote diagnostics; multi-node synchronization and aggregation will enable classroom- and building-level views with measured latency, jitter and loss. Third, we will replace ad-hoc rules with lightweight edge intelligence for anomaly detection and occupancy inference, and we will evaluate adaptive control strategies for ventilation.

In parallel, the data pipeline will target a time-series back end with role-based dashboards and programmatic access, while the hardware will be consolidated on a custom PCB (Printed Circuit Board) with EMI (Electromagnetic Interference) / ESD (Electrostatic Discharge) protection, fused power paths and efficient DC–DC regulation housed in a classroom-ready enclosure. A web-based client will replicate the MATLAB interface to broaden access across devices. Taken together, these directions outline a clear pathway from a functional demonstrator to a secure, scalable and analytically rich platform capable of supporting evidence-based decisions on indoor environmental quality and energy efficiency in schools.

REFERENCES

[1] G. Battista, S. Serroni, M. Martarelli, M. Arnesano, and G. M. Revel, "Innovative measurements for Indoor Environmental Quality (IEQ) assessment in residential buildings," *2022 IEEE International Workshop on Metrology for Living Environment, MetroLivEn 2022 - Proceedings*, pp. 170–173, 2022, doi: 10.1109/METROLIVENV54405.2022.9826982.

[2] S.-A. Drăgușin, N. Bizon, R.-M. Teodorescu, D. Toma, R.-N. Boștinaru, and G. Anghel, "Communication Protocols in Embedded Systems for Automotive Applications: Comparative Analysis and Implementation Through Virtual Instruments," pp. 1–8, Aug. 2025, doi: 10.1109/ECAI65401.2025.11095564.

[3] B.-G. Andrei-Alexandru, P. Diana-Ioana, C. Adriana-Andrada, B. Nicu, D. Sebastian-Alexandru, and B.-G. Maria-Luiza, "Interactive Educational Platform Integrating Electronic Components into Mathematics Courses for Numerical Computation," *Journal of Electrical Engineering, Electronics, Control and Computer Science*, vol. 10, no. 1, pp. 13–20, Jan. 2025, Accessed: Jan. 10, 2026. [Online]. Available: https://jeeeccs.net/index.php/journal/article/view/366

[4] P. K. Malik, A. S. Duggal, S. Aluvala, R. Sahithi, Geetanjali, and A. Gehlot, "Development of a low-cost IoT device using ESP8266 and Atmega328 for real-time monitoring of Outdoor Air Quality with Alert," *2023 3rd International Conference on Advancement in Electronics and Communication Engineering, AECE 2023*, pp. 125–129, 2023, doi: 10.1109/AECE59614.2023.10428098.

[5] M. S. Shafri Shaiful, R. Ambar, M. H. Abd Wahab, and M. M. Abd Jamil, "IoT-based Data Monitoring and Environment Controlling System for Oyster Mushroom Mini-farm," *IVIT 2022 - Proceedings of 1st International Visualization, Informatics and Technology Conference*, pp. 309–316, 2022, doi: 10.1109/IVIT55443.2022.10033410.

[6] P. S. B. MacHeso, T. D. Manda, A. G. Meela, J. S. Mlatho, G. T. Taulo, and J. C. Phiri, "Industrial Temperature Monitor Based on NodeMCU ESP8266, MQTT and Node-RED," *Proceedings - 2021 3rd International Conference on Advances in Computing, Communication Control and Networking, ICAC3N 2021*, pp. 740–743, 2021, doi: 10.1109/ICAC3N53548.2021.9725469.

[7] P. Saeng-On, N. Thaitae, and S. Sonasang, "Development of monitoring PM2.5 based on IoT and Google Data Studio," *Proceeding - 2023 International Electrical Engineering Congress, iEECON 2023*, pp. 64–67, 2023, doi: 10.1109/IEECON56657.2023.10126631.

[8] S. Saha and A. Majumdar, "Data centre temperature monitoring with ESP8266 based Wireless Sensor Network and cloud based dashboard with real time alert system," *Proceedings of 2nd International Conference on 2017 Devices for Integrated Circuit, DevIC 2017*, pp. 307–310, Oct. 2017, doi: 10.1109/DEVIC.2017.8073958.

[9] "ESP8266 Wi-Fi SoC | Espressif Systems." Accessed: Oct. 12, 2025. [Online]. Available: https://www.espressif.com/en/products/socs/esp8266

[10] D-Robotics, "DHT11 Humidity & Temperature Sensor".

[11] "MQ-2 Semiconductor Sensor for Combustible Gas".

[12] "KY-018 Photoresistor module KY-018 Photoresistor module Contents", Accessed: Oct. 12, 2025. [Online]. Available: https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code]

[13] "KY-038 Microphone sound sensor module KY-038 Microphone sound sensor module Contents".

[14] "Handson Technology User Guide SW-420 Vibration Sensor Module", Accessed: Oct. 12, 2025. [Online]. Available: https://handsontec.com

[15] "Ultrasonic Ranging Module HC-SR04", Accessed: Oct. 12, 2025. [Online]. Available: www.Elecfreaks.com

[16] "Handson Technology User Guide I2C Serial Interface 1602 LCD Module", Accessed: Oct. 12, 2025. [Online]. Available: www.handsontec.com

[17] "Active Buzzer Module", Accessed: Oct. 12, 2025. [Online]. Available: www.handsontec.com

[18] Wakefield-, "Wakefield-Vette Revision A DC FANS".

[19] "1 Channel 5V Optical Isolated Relay Module", Accessed: Oct. 12, 2025. [Online]. Available: www.handsontec.com

[20] "LM2596 SIMPLE SWITCHER ® Power Converter 150-kHz 3-A Step-Down Voltage Regulator," 2023, Accessed: Oct. 12, 2025. [Online]. Available: www.ti.com

[21] D. K. Halim, T. C. Ming, N. M. Song, and D. Hartono, "Arduino-based IDE for embedded multi-processor system-on-chip," *Proceedings of 2019 5th International Conference on New Media Studies, CONMEDIA 2019*, pp. 135–138, Oct. 2019, doi: 10.1109/CONMEDIA46929.2019.8981862.