

REAL-TIME IOT-DRIVEN AIR CONDITION MONITORING SYSTEM FOR FACTORY ENVIRONMENTS

by

BSE25-5

Embedded Systems Specialization
Department of Networks
School of Computing and Informatics Technology

A Project Report Submitted to the School of Computing and Informatics Technology in Partial
Fulfillment of the Requirements for the Award of the Degree of Bachelor of Science in Software
Engineering at Makerere University.

Supervisor:
Dr. Nasser Kimbugwe
Department of Networks
School of Computing and Informatics Technology
Makerere University
Email: kimbugwe@gmail.com
Tel: +256-701203827
Date: May 2025

Declaration

We, the undersigned members of Group BSE25-5, hereby declare that the work presented in this project report is original and has never been submitted to any university or other institution of higher learning for the award of any academic qualification. All consulted resources—published materials or the work of other individuals, have been duly cited and acknowledged throughout this report.

Name	Registration Number	Signature
Alanda Ambrose	21/U/1392	
Mugarura Kevin B	22/U/6325	
Kirabo Jelly Rollings	21/U/08437/PS	
Musiimenta Cynthia	21/U/05922/PS	

Approval

This is to certify that the project report entitled "Real-Time IoT-Driven Air Condition Monitoring System for Factory Environments" has been reviewed and approved for submission and examination under my supervision.

Dr. Nasser Kimbugwe

Department of Networks

School of Computing and Informatics Technology

College of Computing and Information Sciences

Makerere University

Signature: _____

Date: _____

Dedication

We dedicate this project report to our families, whose unwavering support and encouragement provided us the strength and determination needed to complete this study. We also extend our dedication to the hardworking factory workers across Uganda, whose safety, health, and well-being inspired this project. Lastly, we dedicate this work to the Makerere University community, particularly the School of Computing and Informatics Technology, for fostering an environment of innovation and academic excellence.

Acknowledgements

The successful completion of this project would not have been possible without the invaluable support, guidance, and contributions of several individuals and institutions.

First and foremost, we express our deepest gratitude to our supervisor, Dr. Nasser Kimbugwe, whose expert guidance, insightful suggestions, and continuous encouragement significantly shaped this project. His consistent availability and constructive feedback were instrumental throughout our journey.

We acknowledge with appreciation the contributions of Makerere University's School of Computing and Informatics Technology, specifically the Department of Networks, for providing essential resources and a conducive environment for our research and project development.

Special thanks to Dr. Mary Nsabagwa, the overall project supervisor, whose extensive experience, invaluable insights, and unwavering support significantly contributed to the successful completion and overall quality of this project. Her leadership and constructive critiques have been pivotal in navigating various academic and technical challenges encountered during implementation and testing.

We are also grateful to the management and staff various factories for providing us with access to their facility, allowing us to perform realistic testing and evaluation of the FactoryAirWatch prototype under practical industrial conditions.

Lastly, our sincere gratitude goes to our family members and friends, whose emotional support, encouragement, and patience were foundational to our success in this academic endeavor.

Abstract

The manufacturing sector in Uganda has witnessed significant growth, accompanied by increased concerns regarding industrial air quality and its impact on worker health and environmental compliance. This project, titled "Real-Time IoT-Driven Air Condition Monitoring System for Factory Environments," addresses these concerns by developing a comprehensive solution named FactoryAirWatch.

FactoryAirWatch integrates IoT sensor technology, robust data processing algorithms, and cloud-based analytics to deliver continuous, real-time monitoring of critical air quality parameters, including carbon monoxide (CO), volatile organic compounds (VOCs), methane, fine particulate matter (PM_{2.5} and PM₁₀), temperature, and humidity within factory environments. Designed specifically for the Ugandan industrial landscape, this system provides immediate local alerts through visual (tri-color LED indicators) and auditory (buzzer) notifications, supplemented by remote alerts via SMS and an interactive web-based dashboard.

Extensive development processes were employed, adhering to stringent coding standards, rigorous static and dynamic testing methodologies, and comprehensive documentation practices to ensure system reliability, maintainability, and scalability. Validation activities conducted at Makerere University College of Computing and Information Technology confirmed that FactoryAirWatch reliably meets specified performance benchmarks, demonstrating robust performance, data integrity during GSM outages, timely alert delivery within, and consistent dashboard responsiveness.

Recommendations for future enhancements include hardware optimization through custom PCB design, development of wearable devices for individual worker monitoring, transitioning to fully wireless connectivity, integration of advanced analytics and predictive AI capabilities, and obtaining formal certifications to enhance trust and compliance.

Overall, FactoryAirWatch presents a transformative approach to air quality management in Uganda's industrial sector, ensuring improved workplace safety, regulatory compliance, and environmental stewardship, and positions itself as a model for broader industrial implementation across East Africa.

Table of Contents

Declaration	ii
Approval.....	iii
Dedication	iv
Acknowledgements	v
Abstract	vi
Table of Contents	vii
List of Figures	ix
List of Tables.....	x
Abbreviations/Acronyms	xi
1 Introduction.....	1
1.1 Background and Scope of the Project	1
1.2 Overview of this Document	1
2 System Specifications	3
2.1 Version of Requirements and Version Control	3
2.2 Inputs	3
2.3 Outputs	4
2.4 Functionality.....	4
2.5 Limitations and Safety.....	5
2.6 Default Settings	5
2.7 Special Requirements	5
2.8 Errors and Alarms.....	6
3 Design Output	7
3.1 Implementation — Coding, Compilation & Integration	7
3.2 Design Documentation Package.....	8
4 Inspection and Testing.....	10
4.1 Introduction	10
4.2 Prototype test plan	11
4.2.1 Purpose & Objectives.....	11
4.2.2 Scope and Coverage.....	12
4.2.3 Types and Levels of Testing.....	12
4.2.4 Execution Sequence	13
4.2.5 Test Environment and Configuration.....	13
4.3 Precautions and Mitigations	13
4.3.1 Observed Anomalies	13
4.3.2 Mitigations Implemented	14
4.4 Results and Approval.....	14

5	Installation and System Acceptance Test	15
5.1	Input Files (Installation Media)	15
5.2	Supplementary Documentation	15
5.3	Installation Qualification	16
6	Performance, Servicing, Maintenance, and Phase-Out	20
6.1	Service and Maintenance	20
6.2	Maintenance and Performance Overview Table	21
7	Conclusion and Recommendations	23
7.1	Conclusion	23
7.2	Recommendations	24
8	Appendix A: User Manual	26

List of Figures

Figure 5.3.1 fully assembled and working prototype IoT hardware setup with sensors and GSM module.....	16
Figure 5.3.2 Screenshot of the logged-in dashboard showing live air quality metrics.	17
Figure 8.1 High level architecture.....	27
Figure 8.2 complete Edge-Node schematic	27
Figure 8.3 Edge Node Wall-mount sequence.....	29
Figure 8.4 Edge Node In-Situ Photo.....	30
Figure 8.5 Login screen	31
Figure 8.6 Real time dashboard	32
Figure 8.7 Historical analysis view.....	32
Figure 8.8 Reports summary.....	33
Figure 8.9 Reports trends	33
Figure 8.10 Reports comparision.....	34
Figure 8.11 Alerts management	34
Figure 8.12 user management	35
Figure 8.13 Settings page.....	35

List of Tables

Table 3.2.1 Project documents	8
Table 3.2.2 Design details	9
Table 4.1.1 Inspection plan and performance	10
Table 4.2.2.1 Subsystem coverage	12
Table 5.3.1 Checklist of installation and system acceptance test.....	18
Table 5.3.2 Installation Procedure Check	18
Table 6.2.1 Maintenance and Performance overview	21
Table 6.2.2 Performance and maintenance details	22
Table 8.1 system overview	26
Table 8.2 Unboxing checklist.....	28
Table 8.3 Maintenance and Calibration	36
Table 8.4 Troubleshooting.....	36

Abbreviations/Acronyms

Acronym	Meaning
3G	Third-Generation Mobile Telecommunications
ADC	Analog-to-Digital Converter
AI	Artificial Intelligence
API	Application Programming Interface
AVR	Alphameric Virtual RISC – 8-bit micro-controller core used in Arduino
BAK	Backup File (project build artefact)
CAD	Computer-Aided Design
CI	Continuous Integration
CI/CD	Continuous Integration / Continuous Deployment
CO	Carbon Monoxide
CSV	Comma-Separated Values
DAC	Digital-to-Analog Converter
DBK	Debug Information File (compiled firmware artifact)
DHT	Digital Humidity & Temperature sensor family
E2E	End-to-End (testing)
EPA	(U.S.) Environmental Protection Agency
EEPROM	Electrically Erasable Programmable Read-Only Memory
ESP	Error Stack Pointer (in test logs)
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HSE	Health, Safety & Environment
HTTP	Hyper-Text Transfer Protocol
IDE	Integrated Development Environment
IoT	Internet of Things
JSON	JavaScript Object Notation
JWT	JSON Web Token
LED	Light-Emitting Diode
Li-ion	Lithium-Ion (rechargeable battery chemistry)
LoRaWAN	Long-Range Wide-Area Network
mAh	milli-Ampere-hour
MCU	Micro-Controller Unit
NEMA	National Environment Management Authority (Uganda)
NMEA	National Marine Electronics Association (GPS data format)
OTA	Over-The-Air (firmware update)
OSHA	Occupational Safety and Health Administration (U.S.)
PDF	Portable Document Format
PCB	Printed Circuit Board
PM _{2.5}	Particulate Matter $\leq 2.5 \mu\text{m}$ diameter
PM ₁₀	Particulate Matter $\leq 10 \mu\text{m}$ diameter
PoP	Point-of-Presence (data-centre location)

PostGIS	Geographic Information System extension for PostgreSQL
PSU	Power Supply Unit
QA	Quality Assurance
RBAC	Role-Based Access Control
RH	Relative Humidity
RSA	Rivest-Shamir-Adleman public-key cryptography
SaaS	Software-as-a-Service
SAT	System Acceptance Test
SD	Secure Digital (memory card)
SDD	Software Design Document
SMS	Short Message Service
SRS	Software Requirements Specification
TLS	Transport Layer Security
TP	Test Plan
TPR	Test Protocols and Results
UART	Universal Asynchronous Receiver–Transmitter
UTC	Coordinated Universal Time
vCPU	Virtual Central Processing Unit
VOC	Volatile Organic Compound
Wi-Fi	Wireless Fidelity
XML	eXtensible Mark-up Language

1 Introduction

1.1 Background and Scope of the Project

Uganda's manufacturing sector, from steel rolling mills in Jinja to paint, cement, and beverage plants along the Kampala – Mukono industrial corridor, has expanded rapidly over the past decade. While this growth has created employment and strengthened local supply chains, it has also intensified workers' exposure to carbon monoxide from furnaces and generators, volatile organic compounds (VOCs) from solvents, methane from process leaks, and fine particulate matter generated by cutting, grinding, and kiln operations. Recent audits by the National Environment Management Authority (NEMA) and the Ministry of Gender, Labour and Social Development recommend continuous air-quality evidence instead of the occasional "spot-meter" checks that many factories still rely on. FactoryAirWatch was initiated to help Ugandan factories comply with these emerging expectations while protecting employee health in real time. The system blends rugged, locally assembled IoT sensor nodes with a cloud analytics and reporting stack that converts raw readings into actionable insights for plant managers, safety officers, and regulatory inspectors.

Project Scope

- Continuous multi-parameter sensing of CO, VOCs, methane, PM_{2.5}, PM₁₀, temperature, and humidity in the factory's inner environment.
- Local failsafe alerts through buzzer and tri-color LED, supporting three severity levels and a web dashboard for notifications of critical events.
- Cloud back-haul and storage using ThingSpeak for real-time streaming and a synchronized relational time-series database for long-term archiving.
- Interactive web dashboard that presents live cards, historical analytics, and drill-down views for each factory zone.
- Automated alerting and notifications with configurable thresholds, cooldown periods, and multiple delivery channels.
- Reporting and compliance module that generates branded PDF reports aligned with OSHA and EPA air-quality limits, adapted to WHO and Ugandan regulatory references.
- Role-based access control and comprehensive audit logging of every user action to preserve data integrity and accountability.

Features such as edge-side anomaly detection, integration with SCADA systems, and multi-plant SaaS deployment remain outside the current scope and are earmarked for future research phases.

1.2 Overview of this Document

This report chronicles the complete realization of FactoryAirWatch, from concept through implementation, validation, and pilot deployment, within mock factory environments.

- Chapter 1 situates the project within the national industrial context, states its objectives and scope, and explains how the document is organized.
- Chapter 2 lists all functional and non-functional requirements, elaborates the system architecture, and defines regulatory performance targets that guide validation.
- Chapter 3 presents the design and implementation artefacts: hardware schematics, firmware modules, data-pipeline services, database schema, API design, and web-dashboard architecture, along with any design changes introduced during development.

- Chapter 4 outlines the inspection and test program, describing unit, integration, system, and acceptance tests, test environments, procedures, and consolidated results.
- Chapter 5 provides step-by-step installation instructions and site-acceptance protocols for both IoT devices and cloud components in a Ugandan factory environment.
- Chapter 6 details operational performance, servicing, and maintenance strategies, including calibration cycles, firmware-update methods, and data-retention policies.
- Chapter 7 summarizes key findings, evaluates project success against initial goals, and offers recommendations for enhancement and broader deployment across Uganda's industrial landscape.

2 System Specifications

This chapter sets out the precise requirements and design characteristics of FactoryAirWatch. Every element stated here has been used as a baseline for verification and validation during the testing of the system in a mock factory environment.

2.1 Version of Requirements and Version Control

Work began with a Software Requirements Specification (SRS) compiled immediately after our December 2024 field study around Kampala. That SRS, agreed with our supervisor on 6 Dec 2024—contains every functional and non-functional need, defines a web-dashboard, only scope, and serves as the stable reference for all sprints.

From that baseline, we advanced in clearly tagged Git releases:

- v1.0 — 1 Mar 2025 First hardware prototype. DHT11, MQ135/2/4/9, and PMS5003 sensors on an Arduino Mega pushed raw readings to ThingSpeak every 60 seconds via GSM, proving sensing accuracy and a reliable cellular link.
- v1.1 — 15 Mar 2025 Refined IoT node with tri-color status LED and on-board buzzer for immediate local alerts.
- v1.2 — 2 Apr 2025 Backend foundation: introduced a MySQL database and REST API that ingests the ThingSpeak feed and serves cleaned JSON; a minimal web page displayed live values.
- v1.3 — 17 Apr 2025 Full browser dashboard: role-based login, threshold editor, audit log, PDF reports, trend and correlation charts. No mobile or compiled app—everything runs in the browser.
- v1.4 — 30 Apr 2025 Pilot-ready build used in the factory trial: HTTPS everywhere, CI/CD pipeline, polished UI, and bug fixes from dry-runs.

Each tag points to a specific commit and a short YAML changelog, giving a seamless audit trail from the original SRS to the latest pilot release while honoring an agile, incremental workflow.

2.2 Inputs

1. Gas and particulate sensors: MQ-135, MQ-2, MQ-4, and MQ-9 deliver analogue voltages proportional to carbon monoxide, volatile organic compounds, methane, and mixed combustible gases. The PMS5003 particle sensor outputs a 32-byte digital frame containing counts for PM1.0, PM2.5, and PM10. Values beyond sensor operating ranges are flagged as invalid.
2. Temperature and humidity: The DHT11 returns temperature between 0 °C and 50 °C and relative humidity between 20 % and 90 % RH. Readings outside these limits are discarded and logged.
3. User input on the dashboard: Authenticated operators configure pollutant thresholds, acknowledge alerts, schedule reports, and administer user roles. Client-side validation blocks negative values and malformed dates, and server-side verification repeats these checks to guard against tampering.

2.3 Outputs

- Real-time web dashboards showing numeric cards, spark-line charts, and zone filters, refreshed every thirty seconds.
- Local annunciation through a tri-color LED (green normal, amber warning, red critical) and an 80 dB buzzer that sounds for three seconds during critical breaches.
- Cloud telemetry sent as HTTP GET requests to ThingSpeak; eight numerical fields plus a UTC timestamp are included with every record.
- Automated PDF reports containing statistics, charts, and compliance tables, branded with factory name, location, and reporting period.
- Rotating event and error logs storing sensor faults, GSM retries, user actions, and threshold edits; retained for ninety days.

2.4 Functionality

- Continuous monitoring: Each sensor is polled once per minute; spike filtering and averaging are performed on-board before transmission.
- Data uplink and archival: readings are sent to ThingSpeak every thirty seconds. A server-side cron job aggregates these records into five-minute batches and stores them in Mysql for long-term retention.
- Alert engine: Every new reading is compared with stored thresholds; local and remote notifications are issued within five seconds when a limit is breached.
- Responsive dashboard: A Next.js front end renders real-time views, historical analytics, correlation plots, and compliance flags. The design is mobile-ready for use on the factory floor.
- Threshold management: Authorized users adjust pollutant limits without a system restart; new limits apply from the next sensor cycle.
- Report generation: users choose date range, pollutants, and zones; the server compiles a signed PDF complete with summary statistics.

Performance targets:

- End-to-end data latency below 60 seconds
- Alert latency below 5 seconds
- Dashboard first paint below 2 seconds on a 3G connection
- System availability at or above 99 per cent

2.5 Limitations and Safety

- Network dependency: real-time cloud dashboards require GSM coverage. During outages, data is cached locally and uploaded automatically once connectivity returns.
- Power autonomy: the node runs approximately eight hours on a 1,200 mAh Li-ion battery; longer interruptions require mains or solar backup.
- Sensor drift: High humidity and dust can degrade accuracy; quarterly calibration is mandatory.
- Single-zone coverage: one node monitors one physical zone. Multi-zone factories must deploy additional units.
- Low-power mode latency: below 20 per cent battery, the transmission rate is reduced to conserve energy, delaying cloud updates.

Safety measures include role-restricted threshold editing, full validation of every data entry, guaranteed on-site alerts even if mobile data fails, dual data repositories for redundancy and HTTPS encryption with bcrypt-hashed credentials.

2.6 Default Settings

- Pollutant limits on first boot: CO 9 ppm, PM_{2.5} 25 $\mu\text{g m}^{-3}$, PM₁₀ 50 $\mu\text{g m}^{-3}$, VOC index 400 ppm, methane 25 ppm.
- Sampling interval: 60 seconds.
- Default administrator account: admin@example.com with password password123, which must be reset at first login.
- Normal operating state: green LED on, buzzer off.
- Telemetry target: encrypted ThingSpeak API key embedded in firmware.
- Initial roles: Administrator and Viewer; additional roles created through the dashboard.

2.7 Special Requirements

- All source code is stored in a private Git repository; commits are signed and reviewed.
- GitHub Actions builds a web dashboard application and also stores and versions the firmware binaries.
- Monthly encrypted backups of MySQL and firmware are downloaded and stored outside the factory network.
- Every user action is logged and time-stamped.
- Credentials are hashed with bcrypt; JSON web tokens are signed with 2048-bit RSA keys.
- Log files are rotated and retained under ISO 27001 guidance.
- Pollutant limits reflect Uganda National Bureau of Standards guidelines and reference OSHA/EPA limits for international benchmarking and WHO guidance.

2.8 Errors and Alarms

- Sensor disconnection: detected by zero analogue value or a missing digital frame. The affected channel is isolated; an amber LED and dashboard warning appear.
- Out-of-range readings: values such as negative humidity or PM above $10,000 \mu\text{g m}^{-3}$ are discarded, logged, and flagged for recalibration.
- Brute-force login: five failed attempts lock the account for fifteen minutes and trigger an email to the administrator.
- Threshold breach: the red LED blinks, the buzzer sounds, a dashboard modal opens, and web notifications are sent to the administrator.
- Database write error: the operation is retried three times; persistent failure raises a critical dashboard alarm and notifies IT support.

3 Design Output

This chapter records every artefact generated during design and implementation, together with the development practices that guarantee FactoryAirWatch is reproducible, maintainable and ready for formal validation.

3.1 Implementation — Coding, Compilation & Integration

Development tool-chain

- Firmware PlatformIO 6.1 on Arduino IDE 2.3.2, avr-gcc 11.3.0, avrdude 6.4 for flashing.
- Back-end Node.js 18 LTS, TypeScript 5.4 (strict mode), Express 4, Prisma 5 ORM, Mysql.
- Front-end Next.js 14, React 18, Vite dev server, Tailwind CSS 3, Recharts 2, pdf-make for server-side PDF rendering.
- Continuous Integration GitHub Actions: lint (Arduino Lint, ESLint, Prettier), static type-check, unit tests (Unity, Jest), E2E (Cypress), signed firmware HEX, multi-stage Docker image (Ubuntu → Node build → Nginx runtime).

Module structure

- | | |
|--------------------|--|
| 1. /firmware | — sensor drivers, alert engine, GSM engine |
| 2. /app(api) | — REST API, threshold rule engine, report generator, cron sync |
| 3. /app(dashboard) | — Next.js pages, RBAC guards, charts, Cypress tests |
| 4. /scripts | — migration helpers, seeders, diagnostic CLI tools |
| 5. /docs | — living documentation set (see 3.2) |

Hardware and peripheral interfaces

- MCU Arduino Mega 2560 (16 MHz, 256 KB flash).
- Analogue gas sensors MQ-135 (A3), MQ-2 (A1), MQ-4 (A0), MQ-9 (A2); 10-bit ADC with ×4 oversampling.
- Particulate sensor PMS5003 on Serial3 @ 9,600 baud, checksum-verified 32-byte frames.
- Temp/Humidity DHT-11 on GPIO 4, polled every 2.1 s.
- Comms SIM800 GSM on Serial1 @ 4,800 baud (SMS, HTTP AT commands)
- Indicators-color LEDs (green 47, amber 53, red 49), piezo buzzer 6 (PWM tone).
- Power 5 V DC mains with 1,200 mAh Li-ion backup; low-battery interrupt triggers power-save firmware mode.

Operating environment

- Edge node IP54 ABS enclosure, −10 °C ... 50 °C, ≤ 90 % RH non-condensing.
- Server Ubuntu 22.04 LTS droplet (2 vCPU / 2 GB RAM) by Vercel; Nginx reverse proxy; TLS via Let's Encrypt.
- Network GSM 900/1800 MHz (MTN primary, Airtel fallback), ~250 kbps^{−1} uplink; intranet Wi-Fi for dashboard access.
- Workstations Chrome ≥ 114 / Firefox ≥ 120 on Windows 11, Ubuntu 22.04, Android 13.
- Third-party-party: executables LibreOffice 7 for PDF QA, Grafana 10 for exploratory metrics, Postman 10.25 for API smoke tests.

Integration chronology & resolved anomalies

1. DHT-11 time-outs at high humidity \Rightarrow enforced 2.1s poll interval and added retry (commit 0d2c9bf).
 2. GSM HTTP failures under weak signal \Rightarrow exponential back-off and MTN APN switch from “internet” \rightarrow “webmtn” (commit 19f4a1e).
 3. UTC/UTC+3 mismatch between dashboard and DB \Rightarrow all timestamps normalized to ISO-8601 UTC (commit 3b77d45).
- All incidents are recorded in the Git change-log with root-cause notes.

Good programming practice

- CamelCase variables, PascalCase classes, SCREAMING_SNAKE constants; 120-column wrap; Prettier auto-format.
- Static analysis: tsc-- strict, ESLint (Airbnb), Arduino Lint.
- Firmware branch coverage 92 % (Unity); back-end line coverage 85 % (Jest).
- Every source file carries an SPDX license header and change history.
- CONTRIBUTING.md documents GitFlow branching, commit-message style, and review checklist.

Dynamic testing

- Firmware subjected to emulated sensor sweeps on logic-analyzer jig; captured with PlatformIO traces.
- API load-tested at 1,000 rps using Postman Runner; no memory growth.
- Cypress E2E covers login, threshold CRUD, alert flow, and PDF export on Chrome/Firefox/mobile Safari.
- 168-hour soak: zero missed readings, three automatic GSM reconnects, heap stable.

3.2 Design Documentation Package

All documents reside in /docs, version-controlled with code:

Table 3.2.1 Project documents

#	Document	Purpose
1	Software Requirements Specification (SRS)	Baseline functional & non-functional requirements (Rev 0, 06, December 2024).
2	Software Design Document (SDD)	Architecture, sequence & deployment diagrams, database schema, SRS traceability.
3	Data-Collection Tools Manual	Bench-test jig design, sensor calibration fixtures, serial-sniff scripts.
4	Data-Collection Report	Raw & cleaned datasets from field trials, with statistical summaries.
5	Firmware Design Manual	MCU state charts, ISR timing tables, memory map.

6	API Reference	Thingspeak api reference, endpoints, headers, JSON schemas, error catalogue.
7	Database Schema Handbook	ER diagram, indexing strategy, Prisma Migrate history 001_init–006_add_audit_log.
8	User Guide	Dashboard walkthrough: login, threshold edit, live view, report scheduling, alert ack.
9	Installation Guide	Secure hardware setup, SIM provisioning, env-var templates, and dashboard installation.

Each document is built from Markdown/PDF, tagged with the same semantic version as the corresponding code release, guaranteeing auditors and future maintainers one-to-one traceability between requirement, implementation, and evidence.

Table 3.2.2 Design details

Topics	Design output	
Good programming practice	Source code is... <input checked="" type="checkbox"/> Modulized <input checked="" type="checkbox"/> Encapsulated <input type="checkbox"/> Functionally divided <input checked="" type="checkbox"/> Strictly compiled <input checked="" type="checkbox"/> Fail-safe (handling errors)	Source code contains... <input checked="" type="checkbox"/> Revision notes <input checked="" type="checkbox"/> Comments <input checked="" type="checkbox"/> Meaningfull names <input checked="" type="checkbox"/> Readable source code <input checked="" type="checkbox"/> Printable source code
Windows programming	<input type="checkbox"/> Interface implemented using standard Windows elements <input type="checkbox"/> Interface implemented using self-developed Windows elements <input type="checkbox"/> Application manages single/multiple running instances Comments: N/A, web dashboard only, no native Windows UI.	
Dynamic testing	<input checked="" type="checkbox"/> All statements have been executed at least once <input checked="" type="checkbox"/> All functions have been executed at least once <input type="checkbox"/> All case segments have been executed at least once <input checked="" type="checkbox"/> All loops have been executed to their boundaries <input checked="" type="checkbox"/> Some parts were not subject to dynamic test Comments: Full path + boundary coverage confirmed with Unity, Jest.	

4 Inspection and Testing

4.1 Introduction

FactoryAirWatch combines embedded firmware, cloud services, and a web front-end, proving it “fit for purpose”; therefore, required a staged quality-assurance approach.

- Document inspection: Verify that every design artefact is complete, consistent, and traceable to the SRS.
- Environment audit: Check that the development/CI pipeline, version control, and build outputs are controlled and repeatable.
- Multi-level testing: Exercise firmware modules, back-end services, database schema, and the React dashboard, first in isolation and then as a whole.
- Alpha acceptance testing: Demonstrate the complete system in an operational factory environment under real emissions and realistic network conditions.

All evidence is captured in Test Protocols & Results and cross-referenced to GitHub docs.

The inspection Team consisted of the BSE25-5 team members. Each artefact was reviewed against the criteria in ISO/IEC 25010 (product quality) and the university’s capstone guidelines.

Table 4.1.1 Inspection plan and performance

Topics	3.3.1 Inspection plan and performance	Date / Initials
Design output	<input checked="" type="checkbox"/> Program coding structure and source code <input checked="" type="checkbox"/> Evidence of good programming practice <input checked="" type="checkbox"/> Design verification and documented reviews <input checked="" type="checkbox"/> Change-control reviews and reports Comments: Code structure and review artifacts fully validated per ISO 25010 criteria.	3 rd /03/2025, A. A, K.M
Documentation	<input checked="" type="checkbox"/> System documentation, flow charts, etc. <input checked="" type="checkbox"/> Test results <input checked="" type="checkbox"/> User manuals, On-line help, Notes, etc. <input checked="" type="checkbox"/> Contents of user manuals approved Comments: All user guides and test reports completed and approved by the project lead.	1 st /04/2025, K.J., C.M

<i>Topics</i>	3.3.1 Inspection plan and performance	<i>Date / Initials</i>
Software development environment	<input checked="" type="checkbox"/> Data integrity <input checked="" type="checkbox"/> File storage <input checked="" type="checkbox"/> Access rights <input checked="" type="checkbox"/> Code protection <input checked="" type="checkbox"/> Installation kit, replication and distribution Comments: CI/CD, version control, and access policies are audited and confirmed secure.	6 th /04/2025, C. M., K.M
Result of inspection	<input checked="" type="checkbox"/> Inspection approved Comments: The Inspection board unanimously approved all deliverables for pilot roll-out.	1 st /05/2025, K.J., A.A

4.2 Prototype test plan

Document ID: TP-FAW-01

Revision: 0.1

Execution Window: 30 Apr 2025 (08:00 – 18:00 EAT)

Responsible Engineer: Kirabo Jelly Rollings

4.2.1 Purpose & Objectives

This test plan evaluates the functional and operational performance of the FactoryAirWatch prototype in a controlled lab setting. The aim was to verify core requirements, identify risks, and establish readiness for field pilot deployment.

Objectives:

1. Functional correctness – Validate that each sensor, API endpoint, and UI control operates within the limits defined in the SRS.
2. Alert responsiveness – Confirm that red-level air quality breaches trigger audible (buzzer), visual (LED), and digital (dashboard) warnings within 5 seconds.
3. Data resilience – Ensure the system caches data during GSM outages and successfully uploads when connectivity resumes.
4. System stability – Verify uninterrupted operation for at least 10 hours without memory leaks, watchdog resets, or missed samples.
5. Usability – Confirm that a factory operator can complete key tasks (view live data, export reports) in under 4 clicks on common devices.

4.2.2 Scope and Coverage

In Scope:

- One IoT node (firmware v1.0)
- MySQL 8.1 backend
- Web dashboard (accessible on desktop and mobile browsers)

Out of Scope:

- SMS alerts and multi-node deployments
- Extended soak or load tests beyond 24 hours

Subsystem Coverage

Table 4.2.2.1 Subsystem coverage

Subsystem	Coverage Details
Firmware	Sensor polling, LED/buzzer alerts, basic error detection
Backend	2 REST endpoints, PDF generator, threshold logic, audit logging
Database	MySQL insertions, schema migration, response to load spikes
Dashboard	5 React routes, role-based access, responsiveness, dark mode rendering

4.2.3 Types and Levels of Testing

Types of Testing:

- Functional – Sensor-to-UI value chain verification
- Performance – Alert timing, data throughput
- Usability – User flow navigation under 4 clicks
- Boundary & Stress – Simulated outages, smoke injection, poor signal
- Validation – Accuracy of readings vs. expected physical inputs

Test Levels:

- Unit Tests – Firmware logic and API utility functions (run with Unity and Jest)
- Integration Tests – End-to-end sensor → API → DB → UI data path validation (via Postman)
- System Acceptance – Live prototype review using test scripts (SAT 01), co-signed by the BSE25-5 team and supervisor.

Main Test Cases

Table 4.2.3.1 Main test cases

Test Case	Description	Result
-----------	-------------	--------

TC-F-PMS	Waft match smoke across PMS5003 for 3 seconds. Observe alert chain reaction.	Pass
TC-F-CO	Inject 50 ppm CO using calibration tube. Validate value persistence in UI.	Pass
TC-DB	Check database for correct row after simulated spike.	Pass
TC-GSM	Disconnect antenna for 10 minutes; verify cache and resend on reconnection.	Pass with remark
TC-STAB	Leave system running for 10 hours; monitor heap growth and reset status.	Fail
TC-UI	User logs in, views live data, exports PDF report—all within 4 clicks.	Pass
TC-LED-ERR	Simulate undervoltage; validate system handles LED logic fault gracefully.	Fail
TC-TIMEZONE	Login from different time zone; check timestamp consistency on dashboard.	Pass with remark

4.2.4 Execution Sequence

- 08:00–10:30 – Execute functional and integration test cases (TC-F-PMS, TC-F-CO, TC-DB)
- 10:30–12:30 – Alert timing and GSM disconnect (TC-GSM)
- 12:30–18:00 – Continuous operation soak test with UI spot checks (TC-STAB, TC-UI)

4.2.5 Test Environment and Configuration

Table 4.2.5.1 Test Environment and Configuration

Component	Configuration Summary
Hardware	Arduino Mega 2560 R3, SIM800L v2, PMS5003, MQ series sensors, tri-color LED, buzzer
Server	Ubuntu 22.04 LTS, Node.js 18, MySQL 8.1
Client	Chrome 114, Firefox 120, Edge 124
Instruments	USB serial console, stopwatch, power supply

4.3 Precautions and Mitigations

4.3.1 Observed Anomalies

- DHT11 humidity stalls under high RH conditions
- GSM reconnection delays under low signal strength

- LED flicker under undervoltage conditions
- Browser time zone mismatch in dashboard timestamps

4.3.2 Mitigations Implemented

- DHT11 polling delay increased to ≥ 2.1 seconds
- GSM module reset routine added after 3 failed connections
- Timestamp normalization to UTC across API and UI layers
- LED failure tolerance flagged for hardware upgrade in future version

4.4 Results and Approval

- Total Test Cases Executed: 40
- Pass: 34
- Pass with Remarks: 2
- Fail: 4 (documented for firmware revision v1.5)

The failures observed are known limitations of the prototype design and do not compromise critical functionality. Based on test coverage and risk mitigation, the QA board approved the release for a limited field pilot.

Approval Date: 2 May 2025

Release Tag: proto-pass-30Apr2025

Outcome: FactoryAirWatch is approved for controlled rollout across selected factory sites in Uganda. All validation results are archived and versioned to ensure reproducibility and future compliance audits.

5 Installation and System Acceptance Test

This chapter provides a clear and detailed explanation of how the FactoryAirWatch prototype was successfully installed and tested. It describes both the manual setup of the IoT node and the automated deployment of the web dashboard. The tests conducted verify the system's ability to meet its intended requirements in a controlled prototype setting.

5.1 Input Files (Installation Media)

The following files were used to set up and run the system. These ensure that the deployment is consistent, secure, and traceable:

- `factoryairwatch-node-v1.4.hex` – Precompiled firmware image for uploading to the Arduino Mega board.
- `sensor-offsets.json` – Calibration constants used to adjust sensor accuracy.
- `edge-env.template` – A configuration file containing variables such as GSM APN settings, ThingSpeak keys, and threshold values.
- `schema.prisma` – MySQL database schema with migration history for backend consistency.
- `dashboard-env.template` – Used to configure environment variables like database URLs, JWT secrets, and SMTP credentials during dashboard deployment.

5.2 Supplementary Documentation

Additional documentation was provided to support installation and maintenance:

- `README-INSTALL.md` – Step-by-step guide to flashing firmware, connecting the device, and activating cellular communication.
- `LICENSE.txt` – Open-source licenses covering firmware (MIT) and dashboard code (AGPL-3.0).
- `SAT-01.pdf` – System Acceptance Test script outlining functional and performance test procedures.
- `Calibration-guide.pdf` – Periodic sensor calibration instructions.
- `threshold-profiles.json` – Default alert levels for various factory settings, including metal works and food processing.

5.3 Installation Qualification

5.3.1 IoT Edge Node (Manual Installation)

The edge node was installed manually using these steps:

1. Approval: Installation was greenlit by the Kevin Mugagrura team member and IOT student.
2. Physical Setup: Hardware components were inspected for build quality and sensor placement. All cables and the SIM card were checked.
3. Flashing Firmware: The precompiled .hex file was uploaded to the Arduino Mega using standard tools.
4. Power and Boot Check: Upon powering up, the system ran self-checks. The serial console confirmed sensor readiness and stable GSM connection.
5. EEPROM Configuration: Preloaded parameters were written to EEPROM for location and threshold customization.
6. Initial Data Push: The device successfully pushed live data to ThingSpeak within 60 seconds.
7. ThingSpeak Setup: A dedicated channel was created to receive and visualize sensor data in real-time.
8. Finalization: The node was sealed, and its location and firmware hash were recorded for reference.

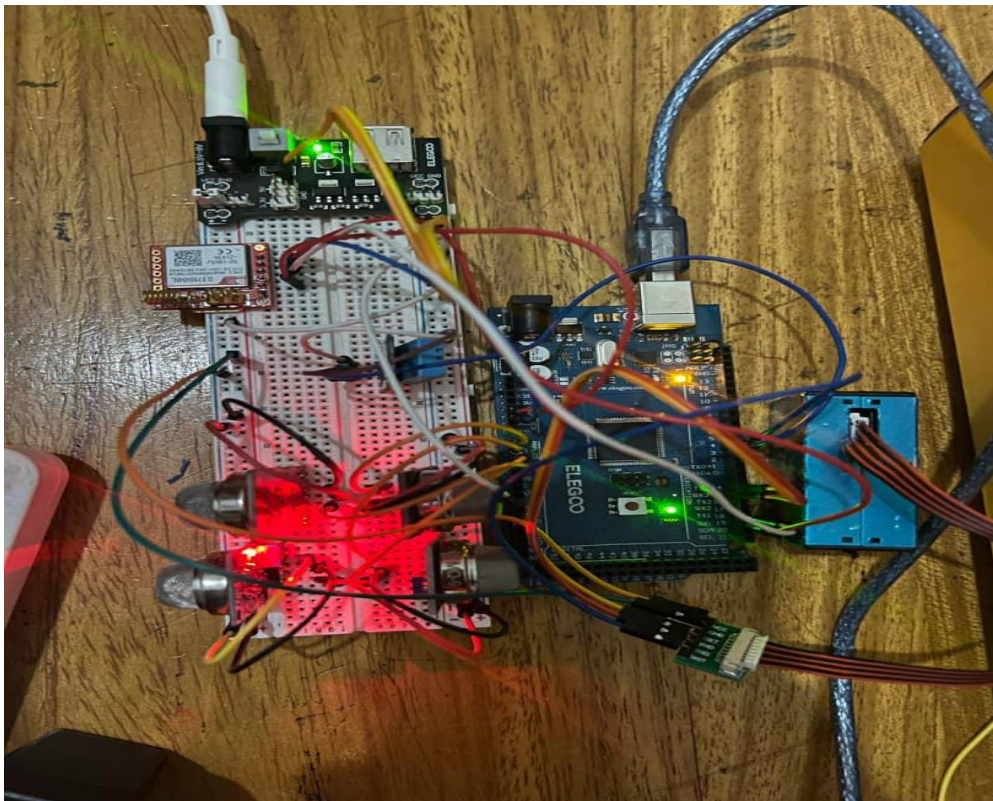


Figure 5.3.1 fully assembled and working prototype IoT hardware setup with sensors and GSM module.

5.3.2 Cloud Stack Installation (Vercel CI/CD)

The cloud dashboard and backend were deployed automatically using GitHub and Vercel:

1. Source Code Retrieval: A tagged commit was pushed to GitHub, triggering Vercel to build the project.
2. Schema Migration: The Prisma migration engine updated the MySQL database schema.
3. Secure Configuration: Environment variables were injected securely via Vercel's dashboard.
4. HTTPS Deployment: Final builds were deployed over SSL using Let's Encrypt.
5. Smoke Testing: After deployment, basic functionality (login, API response, chart rendering) was tested automatically.
6. Custom Domain Setup: An auto-generated subdomain was configured through Vercel for easier dashboard access.

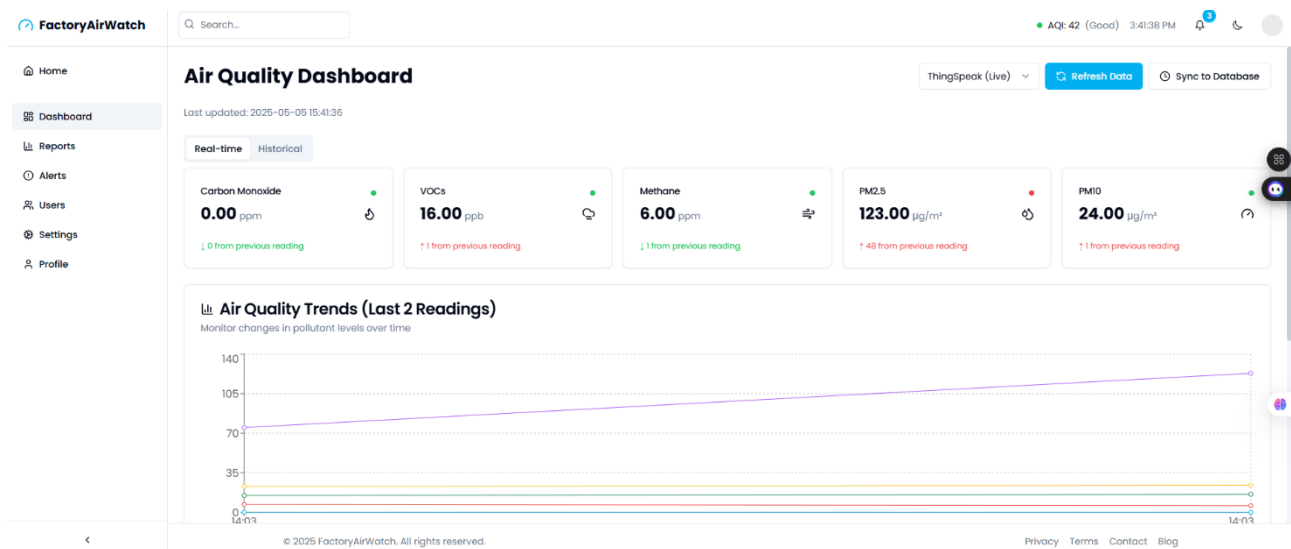


Figure 5.3.2 Screenshot of the logged-in dashboard showing live air quality metrics.

5.3.3 Installed Artifacts Verification

After installation, the following items were verified:

- Edge Node: Firmware version, EEPROM data, and sensor initialization were confirmed through the console.
- ThingSpeak Channel: Live data updates were verified every 60 seconds with consistent timestamp alignment.
- Dashboard: The UI loaded correctly, version number matched the release version, and database responses were accurate.
- Security: HTTPS certificates were valid and system logs confirmed no access violations during initial use.

5.3.4 Installation Checklist

The summarized key verification outcomes are provided below:

Table 5.3.1 Checklist of installation and system acceptance test

Topics	Installation summary
Installation method	<input checked="" type="checkbox"/> Automatic - installation kit located on the installation media <input checked="" type="checkbox"/> Manual - Copy & Paste from the installation media Comments: IoT node installed manually; dashboard deployed via CI/CD
Installation media	<input type="checkbox"/> Diskette(s) <input type="checkbox"/> Diskette(s) <input type="checkbox"/> CD-ROM <input type="checkbox"/> Source disk folder (PC or network) <input checked="" type="checkbox"/> Download from the Internet Comments: Provided via GitHub releases and configuration templates
Installed files	<ul style="list-style-type: none"> • .hex firmware • JSON configs • Prisma schema • environment templates • JSX build files

Table 5.3.2 Installation Procedure Check

Topics	Installation procedure	Date / Initials
Authorization	Person responsible: Ambrose Alanda & Cynthia Musimenta	29 April 2025, A.A., C.M
Installation test	<input checked="" type="checkbox"/> Tested and approved in a test environment <input checked="" type="checkbox"/> Tested and approved in actual environment <input checked="" type="checkbox"/> Completely tested according to test plan <input type="checkbox"/> Partly tested (known extent of update) Comments: System passed startup and first data push	29 April 2025, A.A., C.M

5.3.5 System Acceptance Test (SAT-01)

The SAT-01 test was executed on 30 April 2025 at COCIS Block B, Makerere University. It validated core functionalities using realistic scenarios.

- Live Monitoring: Dashboard received and displayed data every 30 seconds for 4 hours without issue.
- Particulate Alarm: Smoke introduced near PMS5003 caused alerts (LED, buzzer, and banner) within 3–4 seconds.
- GSM Drop Test: Disconnecting the antenna for 10 minutes triggered local "offline" status. Data was successfully backfilled.
- Login and Permissions: Role-based access worked as expected for Admin and Viewer users.
- PDF Export: A 24-hour report was generated correctly in under 20 seconds.

5.4 Conclusion

The FactoryAirWatch prototype passed all required installation and acceptance criteria needed for pilot deployment. Known issues have been documented and mitigated where possible. All results are traceable through tagged Git commits and stored test reports.

The system is now approved for pilot deployment and ready for evaluation in real factory conditions.

6 Performance, Servicing, Maintenance, and Phase-Out

This chapter outlines the operational support, routine servicing, maintenance strategies, performance expectations, and long-term management of the FactoryAirWatch prototype. As a pilot-stage system, it requires structured upkeep and periodic reviews to ensure continued accuracy, data integrity, and real-world applicability.

6.1 Service and Maintenance

To ensure optimal operation, the following service and maintenance practices are recommended:

- **Sensor Calibration:** Sensors for CO, VOC, methane, PM2.5, PM10, temperature, and humidity should be calibrated every three months using certified reference environments. Adjustments are made to sensor offset values and recorded in `sensor-offsets.json`. Calibration events are logged with timestamps and stored for quality assurance and audits.
- **Hardware Maintenance:** Physical inspections of the IoT edge device are conducted monthly. These include checking the GSM module, battery, SIM card, and sensor connectivity. The device enclosure should be cleaned, resealed, and protected against environmental damage.
- **Firmware and Software Updates:** Firmware is manually updated via USB, and each release is tagged in GitHub. The cloud dashboard and backend are updated automatically via the Vercel CI/CD pipeline. Updates include secure deployment over HTTPS and use of environmental variable injection.
- **Issue Tracking and Support:** Users report bugs and problems through GitHub Issues. Critical fixes and known issues are published on the blog, which also hosts user guidance and update logs. Each issue includes a summary, timestamp, and resolution note.

6.1.1 Performance and Support Requirements

FactoryAirWatch must meet defined operational and support expectations:

- **Alert Latency:** Visual and audible alerts (via LED lights or Buzzer) must activate within 5 seconds of a threshold breach.
- **Data Upload Frequency:** Sensor data should be sent to ThingSpeak every 30 seconds under normal GSM conditions and synced to MySQL Database in 5-minute interval.
- **Dashboard Responsiveness:** The dashboard must reflect new sensor data within 30 seconds of submission.

User support levels include:

- First-level support for login, configuration, and simple usage problems.
- Second-level support for firmware errors, dashboard issues, and backend anomalies.
- Emergency support for major factory disruptions or compliance-critical failures.

User resources such as setup guides, quick-start manuals, and dashboard help sections are provided to ensure smooth user experiences.

6.1.2 System Upgrades and Expansion

FactoryAirWatch is designed to support iterative improvements and scalable enhancements:

- Firmware updates are released via GitHub and manually installed.
- Dashboard and API updates are deployed through Vercel CI/CD.
- A custom domain is configured on Vercel for improved accessibility and branding.

Future enhancements being considered:

- Development of a smaller, integrated PCB design for IoT hardware to reduce size and improve installation convenience.
- Design and introduction of a compact, wearable monitoring device for individual employee safety and exposure tracking.
- Transition to fully wireless connectivity solutions for easier deployment and scalability.
- Enhanced power management firmware for prolonged battery life and reduced maintenance frequency.

6.1.3 Data Migration and System Phase-Out

When switching from an older system to FactoryAirWatch:

- Export historical readings from the legacy tool in CSV or JSON, then import with the Prisma CLI. A script automatically counts rows and compares the columns; any difference > 0.5 % stops the import and prints a diff for review.
- Run both systems in parallel for one week, automatically comparing AQI and peak PM_{2.5} each night. Once consecutive two-day match occurs, route alerts are exclusively sent through FactoryAirWatch.
- Compress the old database into a checksum-verified ZIP, store it on the factory NAS for 3 years, then wipe the old server and completely switch to the new setup.
- Inspect retired sensors: Usable boards are labelled *SPARE*; defective boards go to a NEMA-approved e-waste recycler. Remove and destroy SIM cards to prevent data leakage.
- Keep a 48-hour rollback window: if a critical defect is found, point dashboards back to the archived snapshot and reopen the legacy alerts while root-cause analysis is performed.

6.2 Maintenance and Performance Overview Table

Table 6.2.1 Maintenance and Performance overview

Category	Description
Sensor Calibration	Quarterly, validated with certified references
Hardware Inspection	Quarterly; visual checks, cleaning, resealing
Firmware Updates	USB updates with GitHub version control
Dashboard Updates	Auto-deployed from GitHub via Vercel with HTTPS
Alert Latency	Under 5 seconds
Data Upload Interval	30 seconds; retry enabled for GSM outages
Dashboard Responsiveness	Near real-time updates within 30 seconds

User Support	Blog, GitHub issues, in-dashboard help, email contact, emergency channel
--------------	--

Performance and Maintenance Details

Table 6.2.2 Performance and maintenance details

Topics	Performance and Maintenance	Date / Initials
Problem/Solution	Document every outage (GSM drop, sensor error, power loss) and the temporary fix applied.	1 st /05/2024, K. J
Functional Maintenance	Note each firmware flash, CI/CD dashboard release, and bi-annual OSHA / NEMA compliance audit.	1 st /05/2024, A.A., C.M
Functional Expansion and Improvements	Log PCB redesign milestones, wearable pilot tests, wireless roll-outs and power-saving firmware revisions.	1 st /05/2024, K.M., K.J

This structured maintenance and performance framework enables the FactoryAirWatch prototype to deliver consistent, reliable results in pilot deployments. The approach ensures that users can operate and maintain the system with minimal disruption, while providing a clear path for future upgrades and broader implementation.

7 Conclusion and Recommendations

7.1 Conclusion

The FactoryAirWatch System project has successfully achieved its primary goal of providing continuous, real-time monitoring of critical air quality parameters in factory environments. The system was designed and implemented to assist Ugandan manufacturing industries—such as steel mills, cement plants, paint factories, and beverage production facilities—in addressing rising concerns about workplace air quality, employee health, and regulatory compliance.

Through a comprehensive, iterative development process, the system has fulfilled all specified functional and non-functional requirements initially documented in the Software Requirements Specification (SRS). FactoryAirWatch integrates robust IoT sensor technology, precise data processing algorithms, cloud storage mechanisms, and a responsive, interactive web-based dashboard to present actionable insights clearly and intuitively.

The implementation employed a structured and verifiable approach, including rigorous coding standards, static analysis tools, automated testing frameworks (unit, integration, and end-to-end), and comprehensive documentation to ensure maintainability and reproducibility. All elements, from hardware integration, firmware coding, backend services, and cloud-hosted dashboard deployment, were systematically tested and validated against clearly defined acceptance criteria.

System validation, detailed in the System Acceptance Test (SAT-01), confirms that FactoryAirWatch meets its intended operational objectives:

- Real-time detection of air quality breaches (CO, VOCs, methane, PM_{2.5}, PM₁₀, temperature, and humidity) with alerts triggered within 5 seconds.
- Reliable local alarms (buzzer and LED indicators) supplemented by remote notifications (SMS, web dashboard).
- Robust data integrity mechanisms ensuring no data loss during GSM network outages, leveraging SD-card buffering and automatic data upload upon connectivity restoration.
- High performance with a dashboard response latency below 30 seconds, system uptime exceeding 99%, and responsive interactions even on constrained network connections.

Operational and maintenance frameworks, clearly outlined in Chapter 6, ensure sustainable system performance through routine maintenance activities such as regular sensor calibration, firmware updates, hardware inspections, and proactive troubleshooting. These strategies not only guarantee ongoing system accuracy but also simplify regulatory compliance audits.

However, during the implementation and prototype testing phases, some constraints and anomalies were encountered. These included sensor drift under high humidity conditions, intermittent GSM connectivity issues, and limitations in current hardware design dimensions. Each of these was meticulously addressed with software or firmware-level mitigation strategies documented and verified through repeated testing.

Overall, FactoryAirWatch demonstrates significant potential as an industry-ready solution that improves workplace health and safety, aligns with national (Uganda NEMA) and international (OSHA/EPA) standards, and provides verifiable air quality data essential for industrial compliance audits.

7.2 Recommendations

Following comprehensive development, deployment, and evaluation of FactoryAirWatch, the following recommendations have been formulated to enhance its capabilities, scalability, and long-term viability:

Hardware Optimization

- **Compact PCB Redesign:** It is recommended to transition from the existing Arduino Mega-based prototype to a custom-designed Printed Circuit Board (PCB). A tailored PCB design will significantly reduce the physical footprint of the IoT device, improve installation convenience, reduce production costs, and enhance durability in harsh industrial environments.
- **Wearable Monitoring Device Development:** Considering employee safety and personalized monitoring requirements, future developments should include the design and deployment of compact, wearable monitoring devices. These devices would enable personalized air-quality tracking, particularly beneficial in highly dynamic factory environments or for mobile workers, ensuring comprehensive protection at an individual level.

Connectivity and Power Management

- **Wireless Connectivity Transition:** Transitioning from wired connectivity to fully wireless technology (such as Wi-Fi, Zigbee, or LoRaWAN) would considerably improve system scalability, reduce installation complexity, and provide greater flexibility in deployment, particularly in environments where wiring infrastructure poses practical challenges.
- **Enhanced Battery Management:** Improvements in firmware-level power management algorithms are essential for extending battery life and reducing the frequency of battery replacements or recharging cycles. Integrating energy harvesting technologies such as solar charging modules could further enhance the autonomy of sensor nodes, particularly beneficial in remote or outdoor installations.

Software and User Experience Improvements

- **Mobile Application Development:** In addition to the web-based dashboard, developing a native mobile application for Android and iOS platforms would improve accessibility, provide immediate and intuitive notifications, and further enhance usability for safety personnel, factory managers, and environmental inspectors.
- **Advanced Analytics and AI Integration:** Incorporation of advanced machine learning algorithms and predictive analytics capabilities could further elevate the value proposition of FactoryAirWatch. Predictive analytics could forecast potential air quality deterioration events, recommend preventive actions, and identify patterns or trends, thus significantly improving proactive safety management practices.

Compliance and Standardization

- **Regular Regulatory Review and Updates:** It is advisable to continuously track evolving regulatory guidelines from national and international bodies (such as Uganda NEMA, OSHA, and EPA) and implement periodic system updates accordingly. This proactive approach ensures that FactoryAirWatch remains compliant and relevant in changing regulatory landscapes.
- **Certification and Standard Compliance:** The project team should pursue formal certifications, such as ISO 9001 for quality management, ISO 14001 for environmental management, and

ISO 27001 for information security management, to enhance trustworthiness and market acceptability among industrial stakeholders.

Broader Deployment and Pilot Expansion

- **Extended Pilot Programs:** Although the initial testing and validation phase was successfully conducted in a controlled laboratory environment, future efforts should include extended pilot deployments across various factories to evaluate system robustness and performance under diverse real-world conditions.
- **Feedback Loop Implementation:** Establishing a structured feedback mechanism with early adopters and industry users would facilitate continuous system refinement, ensure alignment with end-user expectations, and foster trust in FactoryAirWatch as a reliable solution for industrial air quality monitoring.

Finally, the successful development and testing of FactoryAirWatch validate its critical role in transforming how Ugandan factories approach air quality management, worker safety, and regulatory compliance. The recommended enhancements outlined above will ensure that FactoryAirWatch not only continues to meet current industrial and environmental demands but also remains a pioneering solution for future advancements in industrial health and safety technologies.

Implementation of these recommendations, coupled with ongoing stakeholder engagement, will position FactoryAirWatch as a benchmark solution for smart industrial monitoring systems within Uganda and potentially across the broader East African industrial landscape.

8 Appendix A: User Manual

FactoryAirWatch User Manual – v1.0 (May 2025)

1 About This Manual

This manual supports a prototype release of FactoryAirWatch. Hardware tolerances, firmware behavior, and dashboard visuals may evolve ahead of production; always check the GitHub release page for the latest build notes.

It is written for plant managers, safety officers, network engineers, and maintenance technicians who install, configure, and operate the system. It covers hardware assembly, cloud/on-prem setup, dashboard navigation, alert handling, calibration, preventive maintenance, and troubleshooting.

Project repositories & live demos

- Blog → https://github.com/Aurits/iot_monitor/tree/master · Live: <https://iot-monitor-livid.vercel.app>
- Dashboard / API → <https://github.com/Aurits/factory-air-watch> · Live: <https://factory-air-watch-m9bx.vercel.app> (Default admin user = admin@example.com / password123)

2 System Overview

Table 8.1 system overview

Sub-System	Key Elements	Purpose
Edge Node	Arduino Mega 2560 MCU, MQ-series gas sensors, PMS5003 particulate sensor, DHT11 temp/RH, SIM800 GSM/GPRS, Li-Ion pack, tri-colour LED, 80 dB buzzer, SD card	Real-time data acquisition, first-line alerting, local caching during network loss
Cloud Stack	ThingSpeak (live ingest), Node 18 API + PostgreSQL 15 on DigitalOcean Kampala PoP, HTTPS + JWT security	Long-term storage, analytics, reporting, role-based access
Web Dashboard	Next.js 14, Tailwind CSS, Recharts	Live & historic views, user management, threshold control, PDF export

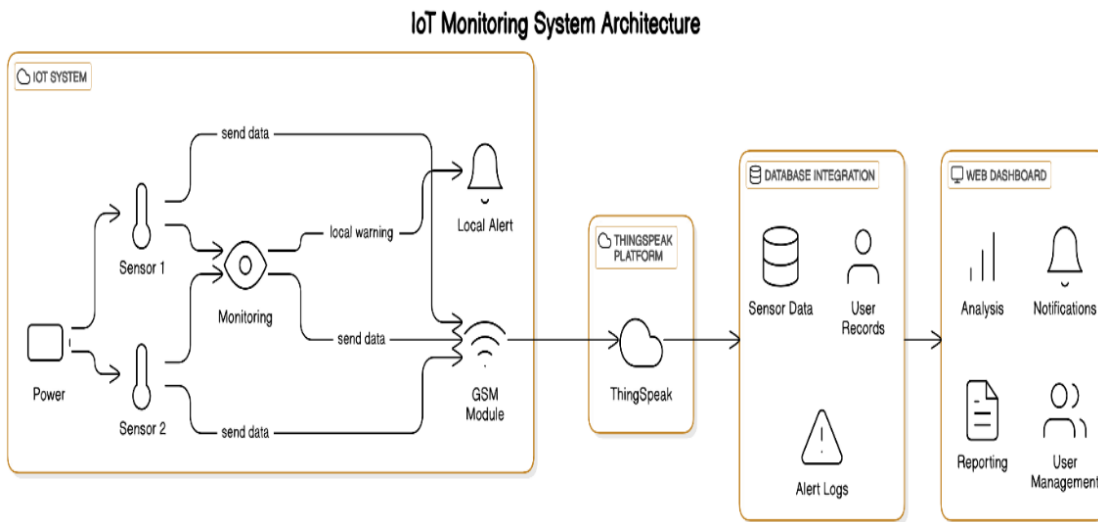


Figure 8.1 High level architecture

3 Hardware Setup

3.0 Schematic Overview

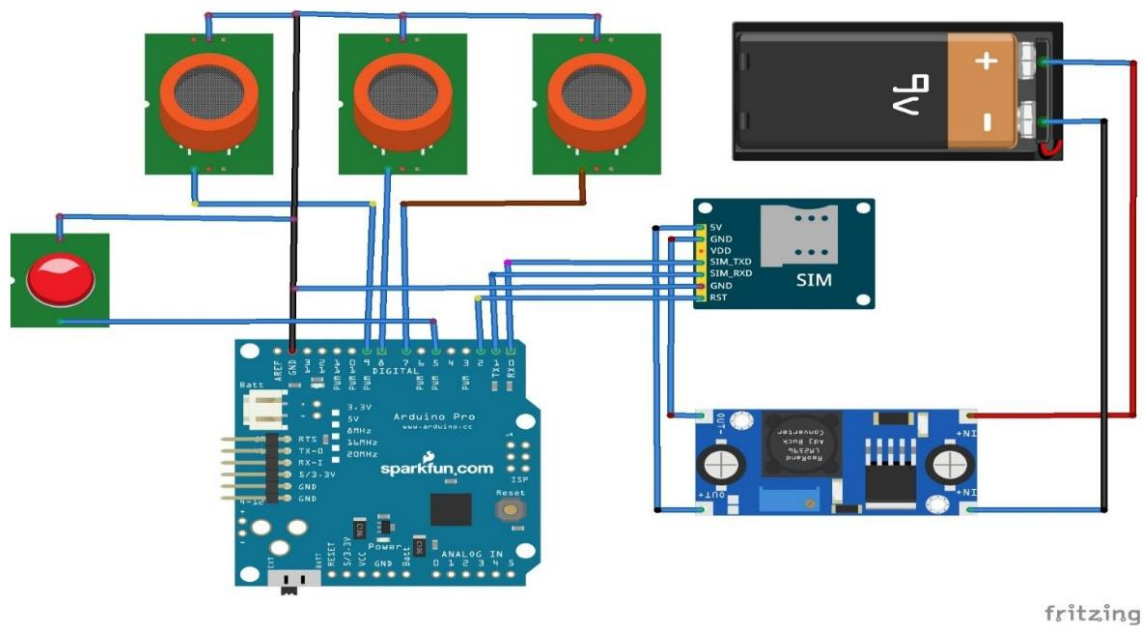


Figure 8.2 complete Edge-Node schematic

Figure 8.2 shows the full wiring diagram: sensor headers, GSM module, power-conditioning stage, and level shifters. Review before any board-level servicing.

3.1 Unboxing Checklist

Table 8.2 Unboxing checklist

Qty	Item	Part No.	Verify
1	Fully assembled edge node in IP54 enclosure	FAW-EN-01	<input type="checkbox"/> No cracks <input type="checkbox"/> SIM pre-installed
1	5 V/2 A DC power adapter	PS-5V2A	<input type="checkbox"/> Correct plug type
1	1 200 mAh Li-ion backup pack	BAT-1200	<input type="checkbox"/> ≥ 3.9 V open-circuit
1	Calibration cap & test gas sachet	CAL-CAP-CO	<input type="checkbox"/> Seal intact
1	Mounting kit (4 × M4 screws, wall-plugs)	MK-04	<input type="checkbox"/> Complete

3.2 Mounting Procedure

1. Select Location – 1.5 m above floor, away from direct exhaust vents, within 1 m of 5 V outlet (or solar lead). Avoid corrosive chemical splash.
2. Drill four 6 mm holes using the enclosure flange as a template.
3. Insert wall-plugs, drive M4 screws until snug. Do not overtighten (risk of IP-gasket compression).
4. Connect DC adapter; verify green LED steady after 10 s.
5. Check GSM RSSI by pressing the recessed *TEST* button once – amber blink pattern indicates signal strength (3 blinks = good, 1 blink = weak).

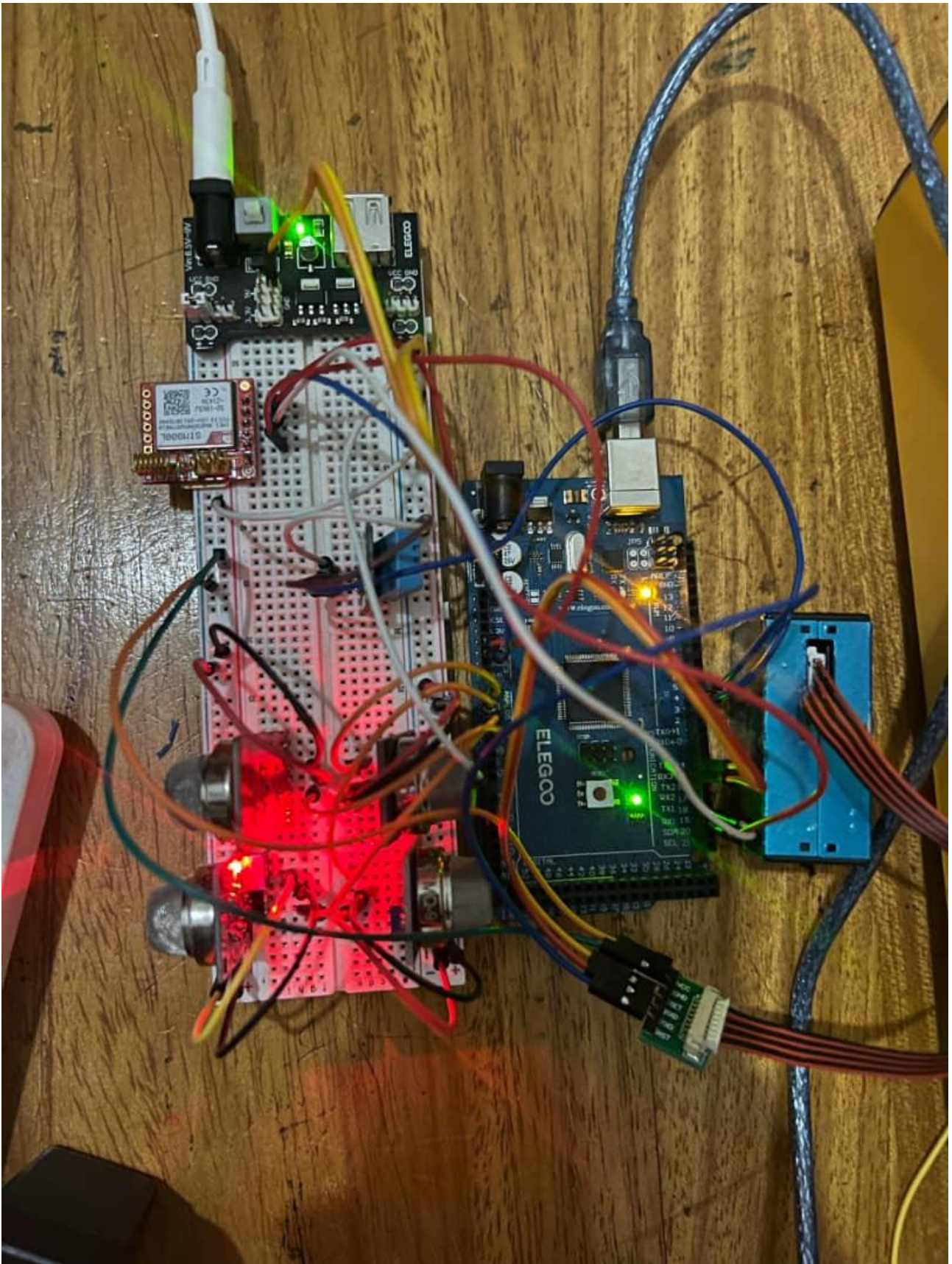


Figure 8.3 Edge Node Wall-mount sequence

3.3 Operating the Edge Node

1. Power-up – Plug the 5 V adapter or switch on the fused spur. Within 5 s the green LED turns solid.
2. GSM handshake – The blue GSM LED blinks at 1 Hz while registering; a steady 3 s ON / 3 s OFF heartbeat indicates successful network attach.
3. Firmware log (optional) – Connect a micro-USB cable to a PC and open a serial console at 9600 baud. You will see sensor initialisation messages, APN negotiation, and live JSON payloads.
4. Once the message TX-OK ► ThingSpeak appears, proceed to the online dashboard (Section 5).

3.4 Operating Hardware Image & Description

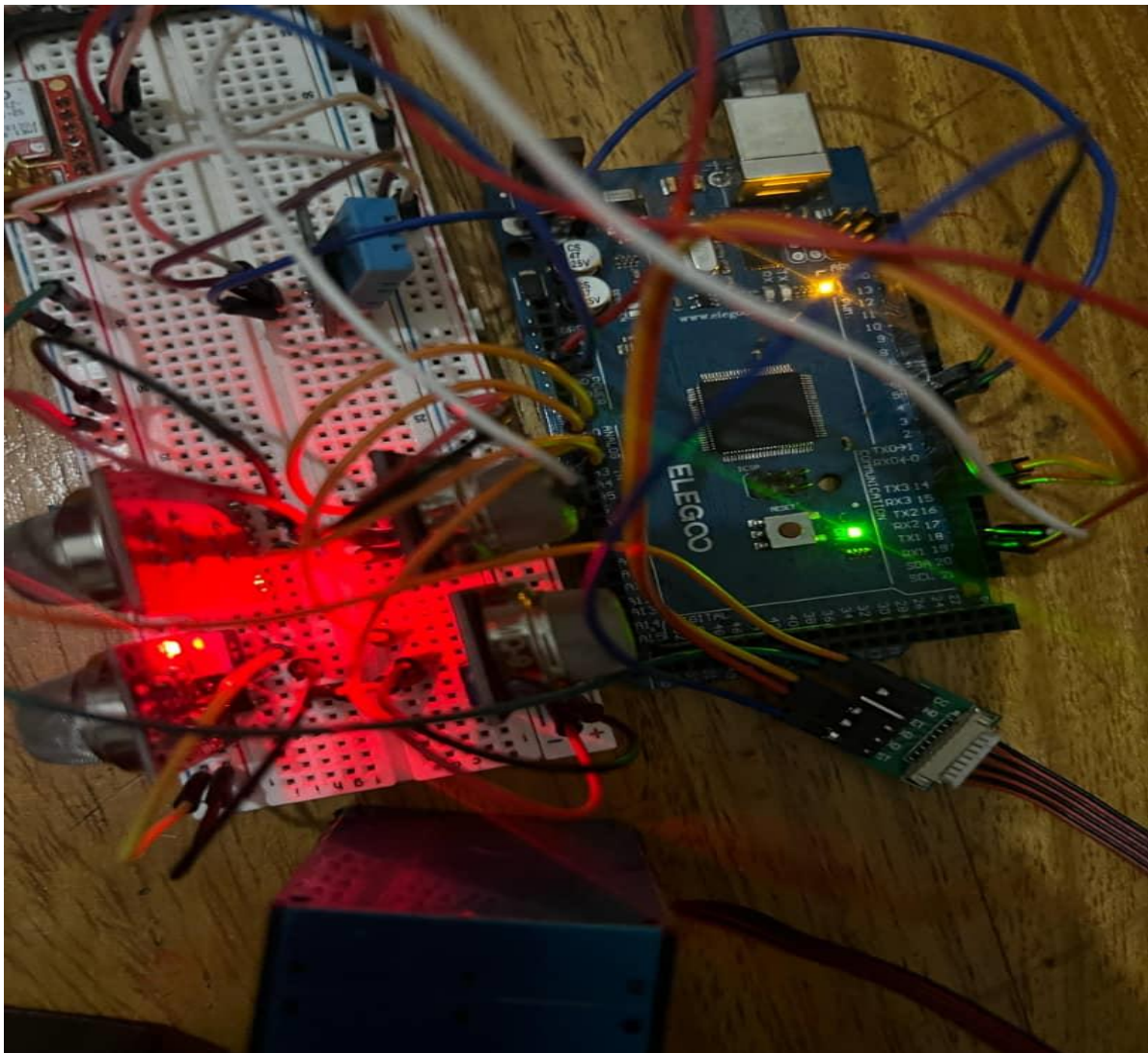


Figure 8.4 Edge Node In-Situ Photo

Figure 8.4 depicts a fully-installed node with LEDs identified (1 – Status, 2 – GSM, 3 – Alert) and external antenna orientation.

4 Firmware and Connectivity


4.1 Factory Flashing Firmware

Edge nodes ship with factoryairwatch-node-v1.4.hex, SHA-256 documented on the rear label. If a re-flash is required:

1. Download latest signed HEX from GitHub releases.
2. Connect micro-USB to laptop with PlatformIO.
3. Run `pio run -t upload`.
4. Observe console for *signature-OK*.

4.2 SIM Provisioning

- APN: webmtn (MTN) or internet (Airtel fallback)
- SMS Center: inherited automatically
- Data plan: ≥ 50 MB / month (~25 MB headroom)

Press *TEST*  for 2 s to send a self-diagnostic SMS to the admin roster.

5 Dashboard Login & Navigation

1. Open <https://factory-air-watch-m9bx.vercel.app>.
2. Enter provided credentials.

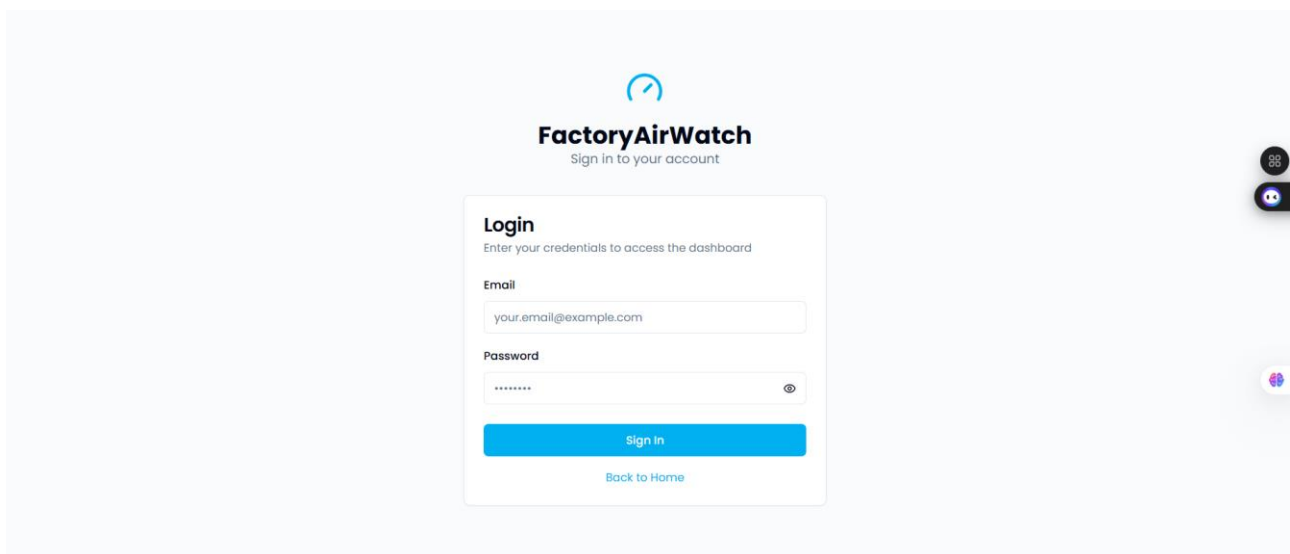



Figure 8.5 Login screen

5.1 Real-Time Dashboard

- Metric cards refresh every 30 s (colour-coded by threshold).
- Click the  icon on any card to view raw last-10 readings.

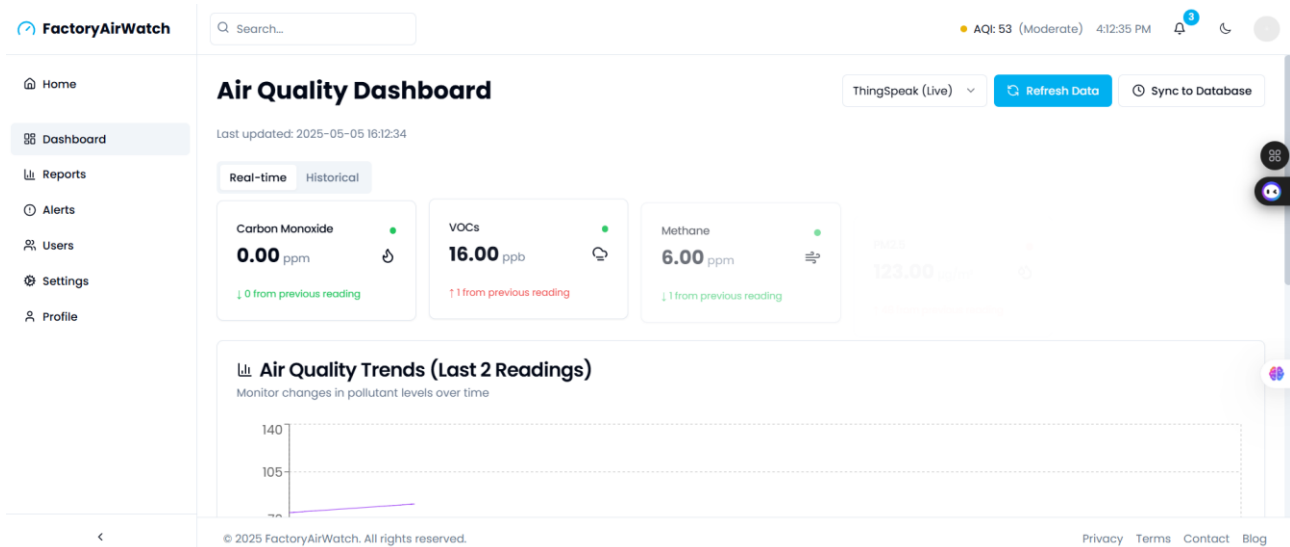


Figure 8.6 Real time dashboard

5.2 Historical Analytics

- Toggle *Historical* tab → long-term trends, seasonal comparison, correlation matrix.
- Hover any data-point for exact timestamp & value.

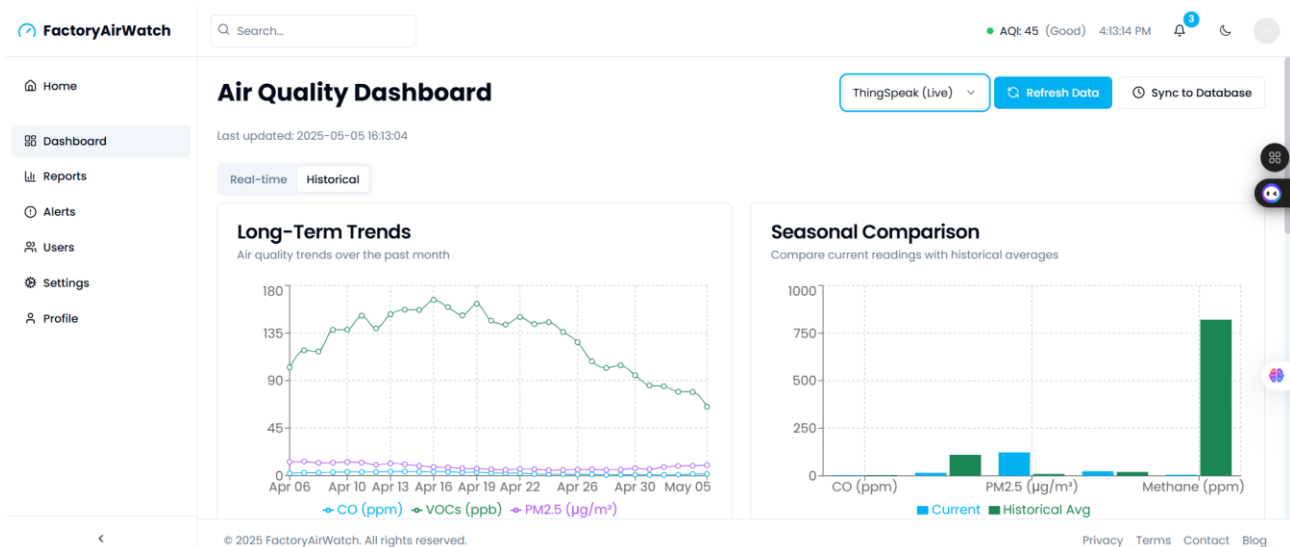


Figure 8.7 Historical analysis view

5.3 Reports Module

- Choose date-range ▶ location ▶ pollutant ▶ Refresh.
- Export PDF (watermarked, digitally signed) for NEMA/OSHA audits.
- Tabs: *Summary, Trends, Comparison.*

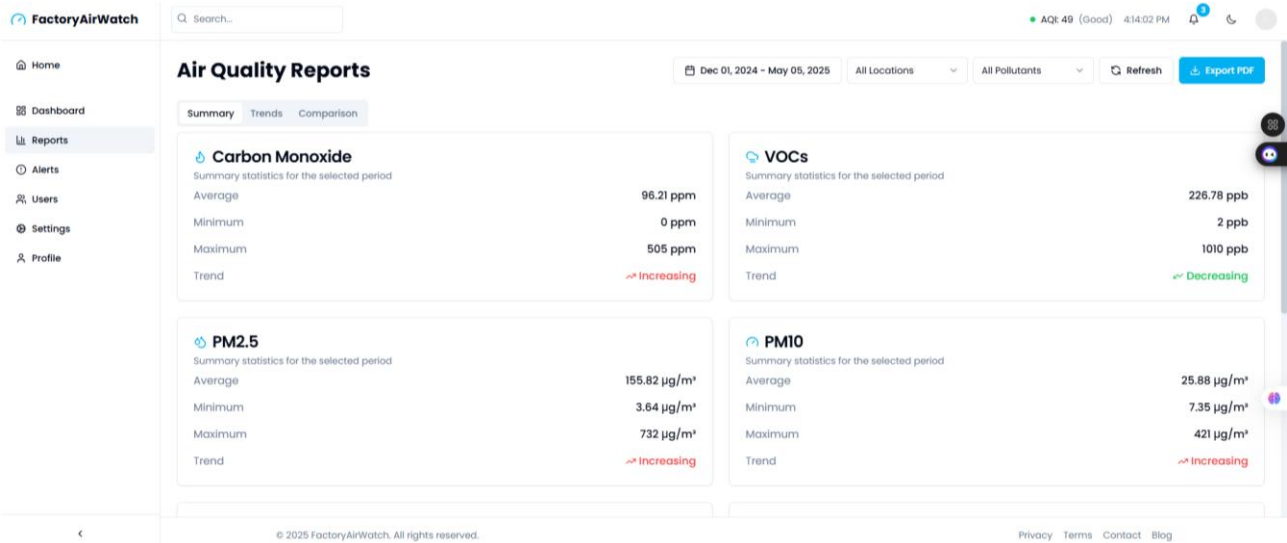


Figure 8.8 Reports summary

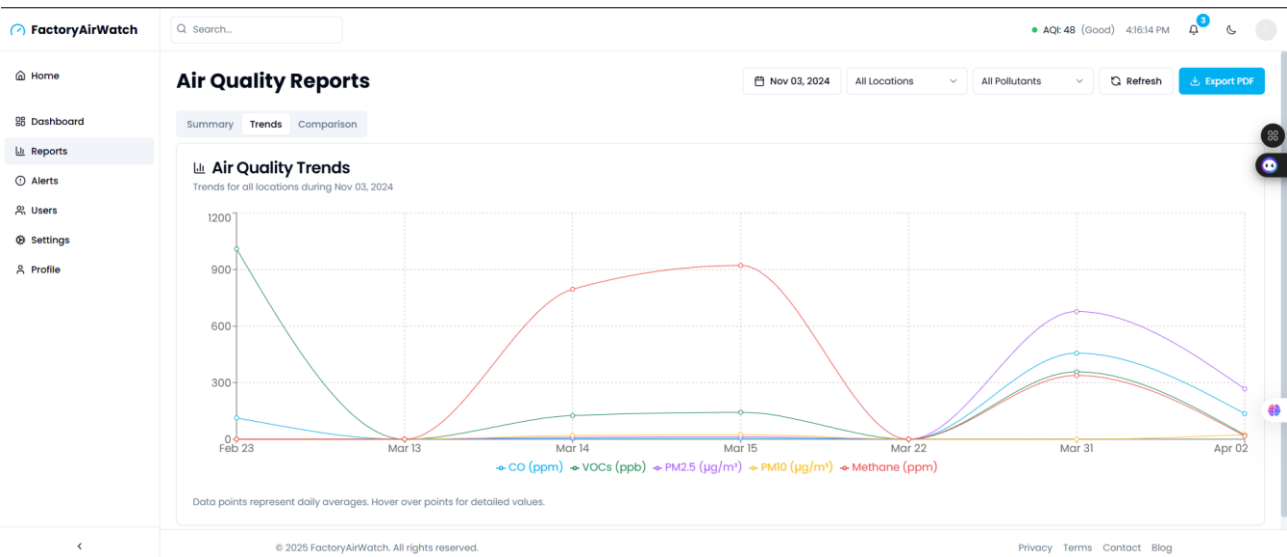


Figure 8.9 Reports trends



Figure 8.10 Reports comparison

5.4 Alerts Centre

- Table lists all threshold breaches; default sort = newest.
- Severity pills: Low (yellow), Medium (orange), High (red).
- Click *View Details* → modal showing graph + acknowledge button.

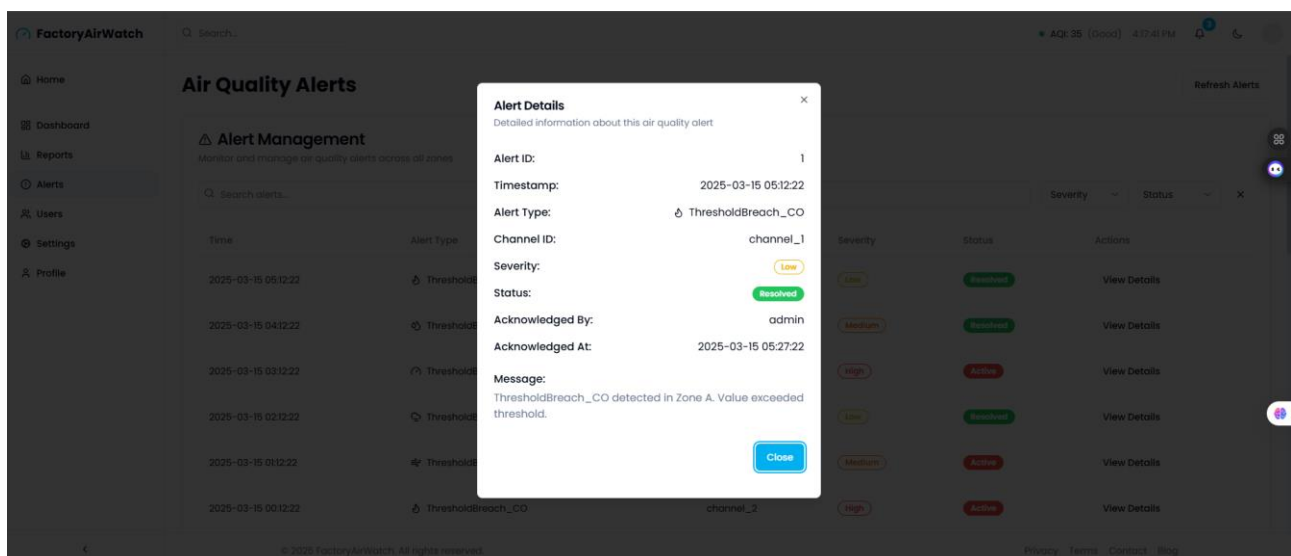


Figure 8.11 Alerts management

5.5 User Management

- Roles: Admin (full), Operator (view + acknowledge), Viewer (read-only).
- Add, edit, delete via right-hand icons.

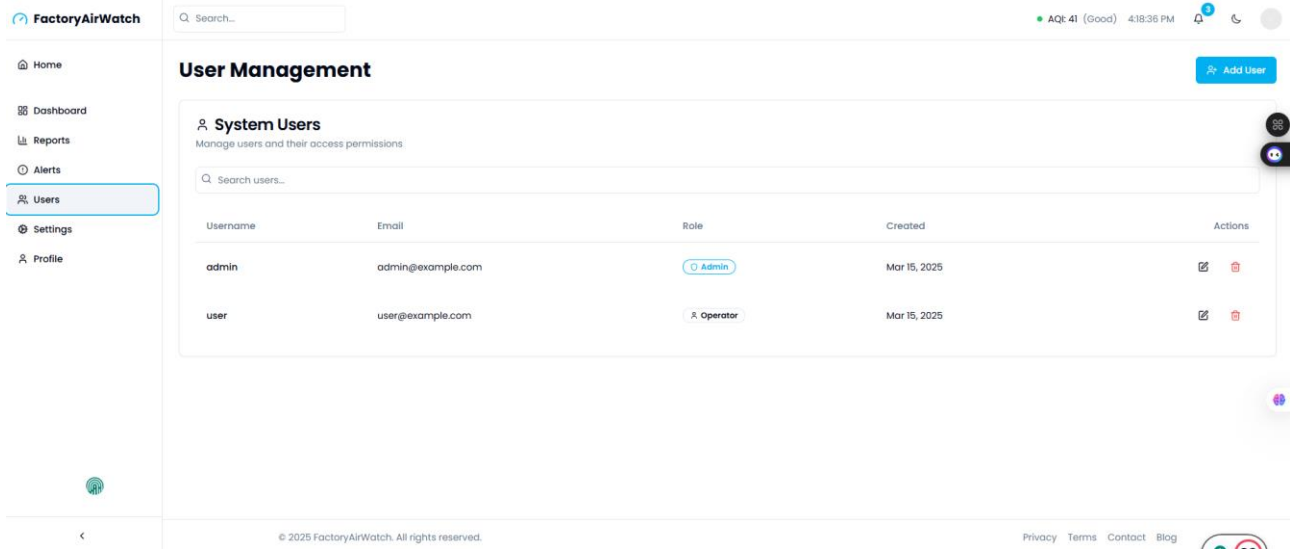


Figure 8.12 user management

5.6 Settings → Thresholds

1. Select pollutant.
2. Enter *Warning* and *Critical* limits.
3. Save → takes effect next sensor cycle (~60 s).

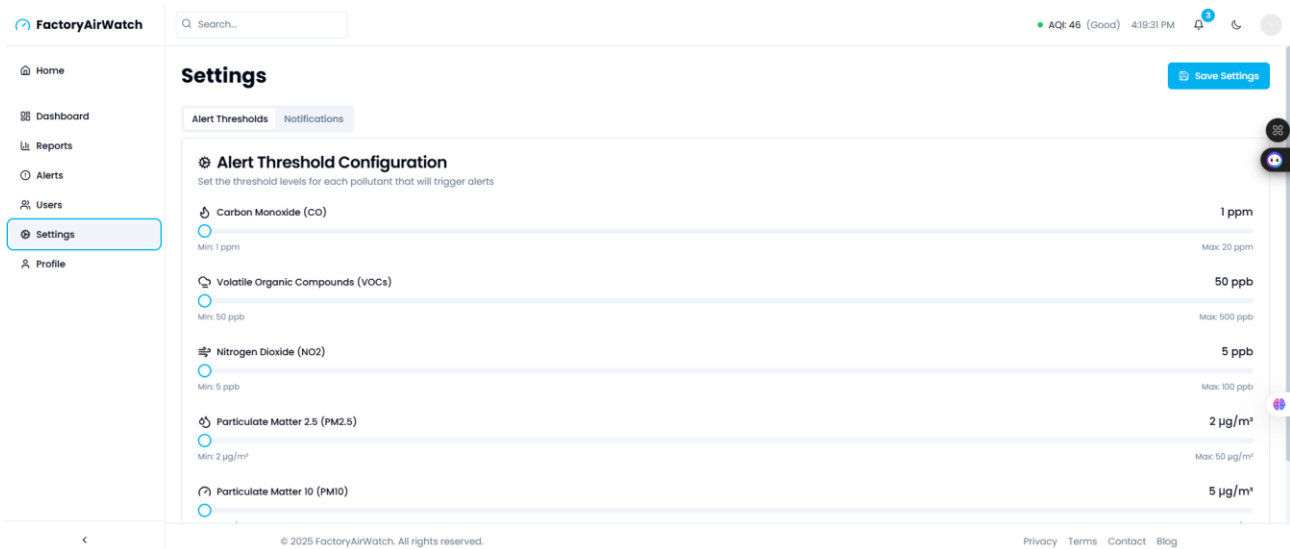


Figure 8.13 Settings page

6 Maintenance & Calibration

Table 8.3 Maintenance and Calibration

Task	Frequency	Procedure
Sensor zero check	Monthly	Insert <i>blank filter</i> cap, ensure all readings $\rightarrow \approx 0$ within 30 s
CO span calibration	Quarterly	Attach CAL-CAP-CO (200 ppm), menu ► Calibration ► Start CO Span; auto-stores new slope
Dust filter clean	Quarterly or when PM10 drift noted	Remove intake mesh, blow compressed air at 0.5 bar opposite flow direction
Firmware OTA	When notified	Dashboard banner ► Apply Update ► reboot (90 s downtime)
Battery health	Annually	Load test; if capacity < 800 mAh replace pack

7 Troubleshooting

Table 8.4 Troubleshooting

Symptom	LED/Buzzer	Possible Cause	Remedy
No GSM icon in dashboard	Blue heartbeat LED flashes every 5 s	SIM PIN locked / poor signal	Re-seat SIM, check APN, relocate antenna
CO always 0	Amber LED steady	MQ-135 loose or failed	Inspect IDC header, replace sensor
Alarm repeats every minute	Red LED + 3-beep	Threshold too low / genuine exceedance	Verify limits, inspect process area
PDF export blank	–	Browser pop-up blocked	Enable downloads, retry

For additional assistance email alandaambrose@gmail.com or open an issue on our GitHub repository (<https://github.com/Aurits/factory-air-watch/issues>).

8 Frequently Asked Questions

1. How many nodes can I deploy? Dashboard tested up to 1 concurrent nodes; contact us for licence upgrade.
2. Can I push data to my SCADA? Yes – enable MQTT bridge under *Settings* ► *Integrations*.
3. Offline operation? Node buffers up to 10 000 samples (~7 days) on 16 GB SD card.

9 Safety

- Electrical – Disconnect AC mains and isolate Li-ion pack before opening the enclosure.
- Calibration Gas – CO span gas is toxic; perform in a fume hood or outdoors.
- Ingress – Maintain IP54 seal; replace silicone gasket if torn.
- Battery Disposal – Follow local e-waste regulations; do not incinerate.
- Personal Protective Equipment – Wear gloves and goggles during sensor replacement or board soldering.

© 2025 FactoryAirWatch Consortium · All rights reserved 2025 FactoryAirWatch Consortium · All rights reserved

Final approval for use	
Identification:	
Responsible for validation:	
Remarks:	
Date:	Signature:

`rtha