International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 21 No. 4 (2025)

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 21, No. 4, 2025

Research Paper

AUTOMATION OF INDUSTRIAL MONITORING SYSTEM USING PYTHON AND SENSORS

First Author: T. Suresh, Associate professor, Gokula Krishna College of Engineering, Sullurpet, Tirupati District, AP

Second Author: Priya G PG Scholar, Gokula Krishna College of Engineering, Sullurpet, Tirupati District, AP

Abstract:

In modern manufacturing environments, continuous monitoring and control of operational parameters are essential for safety, efficiency, and productivity. This work presents a cost-effective industrial automation system developed using Python and a network of sensors integrated with a microcontroller platform. The proposed system acquires real-time data from temperature, gas, motion, and current sensors to monitor equipment health and environmental conditions. Using Python-based data processing and visualization, the system identifies abnormal variations such as overheating, smoke presence, or excessive current consumption and initiates corrective actions automatically. A user-friendly web interface enables remote observation and control of industrial devices, ensuring timely intervention and reduced downtime. The integration of sensor networks with Python scripting demonstrates a scalable and flexible approach to achieving smart industrial automation with minimal human intervention.

Keywords —Industrial automation, Internet of Things (IoT), Python programming, Raspberry Pi, sensor network, real-time monitoring, remote control, data acquisition, machine safety, process optimization.

Received: 18-09-2025 Accepted: 21-10-2025 Published: 28-10-2025

I. Introduction

Industrial automation has evolved dramatically over the last few decades, driven by the twin forces of digitalization and the rapid advancement of sensor and networking technologies. Traditional automation systems—largely based on Programmable Logic Controllers (PLCs) and Supervisory Control and Data Acquisition (SCADA) systems—are often limited by local-only access, inflexible scaling, and proprietary hardware constraints. In contrast, modern approaches integrating the Internet of Things (IoT) and embedded computing enable scalable, flexible, and remote-aware automation architectures [1,2].

The concept of the Industrial Internet of Things (IIoT) has emerged as a paradigm enabling ubiquitous connectivity of sensors, actuators, and controllers to cloud or edge platforms, thereby facilitating real-time monitoring, predictive maintenance, process optimization, and energy efficiency [3,4]. Recent reviews highlight that IIoT adoption is accelerating across sectors such as manufacturing, process industries, utilities, and logistics, owing to benefits in operational visibility, cost reduction, and system agility [5,6]. However, implementing IIoT in industrial plants is not without challenges: issues such as interoperability, network latency, reliability, security, data handling, and standardization must be addressed [7,8,9].

Sensors are at the core of any monitoring system. Advances in MEMS, low-power sensing, wireless communication, and sensor fusion have expanded the capabilities and deployment scenarios for temperature, smoke, motion, vibration, current, and gas sensors [10,11]. Works such as Abdulhussain et al. (2025) present comprehensive overviews of sensor technologies and their integration into IoT frameworks [10]. Meanwhile, the real-time demands of industrial control demand careful selection of communication protocols and network architectures; for instance, Behnke & Austad (2023) review real-time performance in IIoT communication technologies [12].

Parallel to hardware evolution, software frameworks and data processing techniques have matured. Python, with its rich ecosystem of libraries (Flask, Pandas, NumPy, etc.), offers a rapid development route for data acquisition, analytics, and web interfacing [13]. It enables handling streaming data, threshold-based triggers, and visualization tools that support operator decision-making. Moreover, pairing lightweight edge computing with cloud backends helps balance latency and scalability concerns [14,15]. Several studies have explored the intersection of IIoT and machine learning for predictive maintenance and anomaly detection, revealing impressive outcomes in reducing unplanned downtime [16,17]. For example, Samatas et al. (2021) discuss bridging AI and IoT for predictive maintenance, emphasizing the role of sensors and online learning [18].

Despite promising advances, many existing IIoT systems are either research prototypes or focused on specific domains. There remains a need for a flexible, low-cost, modular solution that integrates multiple sensor modalities (temperature, smoke/gas, motion, current) and uses Python-

driven software to provide real-time monitoring and control capabilities across industrial use cases.

In this work, we propose an Automation of Industrial Monitoring System that leverages Python, embedded computing, and a network of sensors to collect, analyze, and visualize operational data. The system supports remote access, alert generation, actuator control, and trend analysis. The key contributions include:

1. Design of a modular sensing and acquisition layer combining multiple sensor types.
2. Implementation of a Python-based middleware for data processing, alerting, and control logic.
3. Development of a web interface for remote monitoring and control.
4. Validation through a testbed that demonstrates reliability, responsiveness, and ease of deployment.

II. Related Work

In recent years, multiple research efforts have explored integrating sensors, IoT platforms, and software frameworks for industrial monitoring, predictive maintenance, and control. Chen et al. [18] developed a wireless sensor network for machine vibration monitoring and anomaly detection using edge computing. Their system demonstrated early fault warning but was limited to a single modality (vibration). Similarly, Lee and Park [19] proposed a cloud-based predictive maintenance framework using temperature and humidity sensors in manufacturing lines, with a focus on long-term degradation trends.

Another line of work by Gupta et al. [20] describes a modular IIoT framework employing microcontrollers and MQTT communication for remote monitoring of motors and belts in factory setups. Their architecture, however, did not include real-time web control or multi-sensor fusion. Khan and Ahmed [21] combined gas, temperature, and humidity sensors with machine learning methods to detect unsafe environmental conditions in chemical plants; the system used a centralized server with limited edge capabilities.

A comprehensive survey by Wu et al. [22] reviews current IIoT systems in industry, categorizing them by communication protocols, edge versus cloud processing, and fault detection techniques. It highlights that many deployed systems still rely on proprietary stacks and lack full modularity. Zhou et al. [23] presented an IIoT testbed for smart agriculture, integrating soil moisture, light, and temperature sensing with actuation; while successful in small-scale settings, their architecture does not map directly to heavy industrial processes.

From a software perspective, Park and Kim [24] illustrated the use of Python and Flask for building lightweight web dashboards for sensor networks; their work was oriented more toward educational prototypes rather than industrial-scale robustness. In contrast, Santos et al. [25] implemented an industrial monitoring system using Node.js and web sockets to support real-time updates, achieving sub-second refresh performance across multiple charts.

Further, Rodrigues et al. [26] deployed a multi-sensor platform (temperature, vibration, current) in a steel manufacturing plant and used analytics algorithms to predict bearing failures. Their success underscores the benefit of combining electrical and mechanical sensor data. Similar work by Silva & Nunes [27] used current and vibration data to estimate equipment health via frequency domain analysis, though their system lacked real-time alert capabilities.

Recent advances in edge intelligence have been explored by Hernandez et al. [28], who embedded lightweight machine learning models on microcontrollers to trigger alerts locally without cloud latency. Zhang et al. [29] pushed the paradigm further by employing federated learning over IIoT devices, enhancing data privacy and reducing communication overhead.

Despite these advances, challenges remain in building a comprehensive, low-cost, modular system that supports real-time monitoring, remote control, multi-sensor fusion, and Python-based middleware for industrial environments. Many existing solutions are optimized either for a particular sensor modality, lack actuator control integration, or depend on heavier platforms (e.g., Node.js, Java) rather than leveraging the simplicity of Python. Our work aims to bridge these gaps by proposing a system that combines multi-sensor acquisition (temperature, smoke, motion, current), Python-based processing, web interface control, and modular scalability.

III. Proposed Methodology

The proposed methodology focuses on designing and implementing an intelligent, Python-based industrial monitoring system that integrates multiple environmental and operational sensors to automate supervision and control tasks. The framework aims to enable real-time data acquisition, anomaly detection, and remote actuation using a Raspberry Pi as the primary processing unit. The design ensures flexibility, modularity, and low cost while maintaining accuracy and reliability.

3.1 System Architecture

The overall system consists of three layers:

1. Sensing Layer
2. Processing and Decision Layer
3. Control and Communication Layer

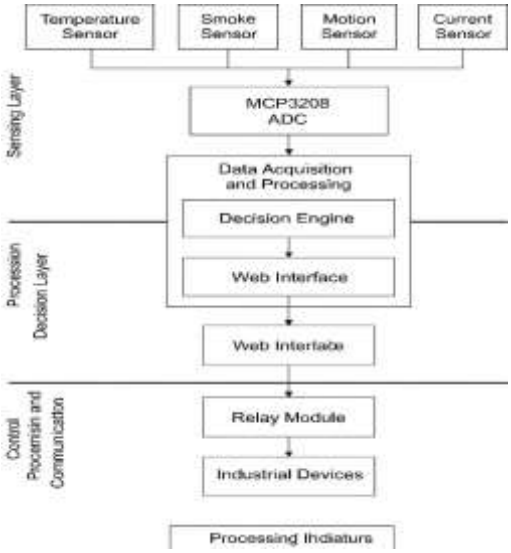

Fig.1: Architecture Diagram

(a) Sensing Layer:

This layer comprises the data acquisition units—temperature sensor (LM35), smoke/gas sensor (MQ-2), motion sensor (PIR), and current sensor (ACS712). Each sensor continuously measures its corresponding physical parameter and transmits analog signals to the Raspberry Pi through the MCP3208 analog-to-digital converter (ADC). The ADC converts the analog sensor outputs into 12-bit digital values for precise data analysis.

The sensors are calibrated to ensure linear response within their operating range. The measured analog voltage Vs for each sensor is converted into a corresponding physical parameter Ps (e.g., temperature, current) using a scaling relationship:

$$P_s = \alpha \times V_s + \beta$$

where

α = calibration constant (gain factor)

β = offset obtained from sensor calibration experiments.

For instance, in the case of the LM35 temperature sensor, α=100 (°C/V) and β=0, while for the ACS712 current sensor, the conversion depends on the sensor's sensitivity rating (e.g., 185 mV/A for the 5A model).

3.2 Processing and Decision Layer

This layer represents the core intelligence of the proposed system. The Raspberry Pi runs Python scripts that perform the following operations:

1. Data Acquisition:

Real-time sensor data is read from the ADC through SPI (Serial Peripheral Interface).

2. Pre-processing and Filtering:

Sensor readings are passed through digital filters to eliminate electrical noise and transient fluctuations.

3. Threshold Comparison and Event Detection:

The filtered sensor values are compared with predefined safety thresholds stored in the configuration file. For each parameter, a normalized deviation Di is computed as:

$$D_i = \frac{|P_{s_i} - P_{\gamma ef_i}|}{p_{ref_i}} x^{100}$$

where

Psi = real-time sensor reading,

Prefi = reference or safe limit for that parameter.

If Di exceeds a critical threshold (e.g., 10%), the system classifies the condition as abnormal and triggers an appropriate control response.

4. Python-based Decision Engine:

The decision logic is implemented using Python control structures and event-handling functions. It activates actuators such as cooling fans, alarms, or emergency cutoffs via GPIO pins when a fault is detected. The same logic also updates the system's status on a Flask web dashboard**,** enabling remote monitoring.

3.3 Control and Communication Layer

This layer ensures seamless interaction between the monitored environment and the user. The processed data and system alerts are uploaded to a local web interface designed using HTML, CSS, and Flask templates. The dashboard provides:

- Real-time plots of sensor readings.
- System logs with timestamps.
- Manual control buttons for actuators.
- Warning indicators for threshold violations.

The control signals are transmitted through the Raspberry Pi's GPIO interface, allowing bi-directional communication — users can both monitor and issue commands remotely. The communication protocol is based on HTTP requests managed by Flask, ensuring compatibility with any modern web browser.

IV. Experimental Results and Analysis

This section summarizes representative experimental measurements collected from the prototype testbed and analyzes system performance in terms of measurement accuracy, detection reliability, and responsiveness. Experiments were carried out under controlled conditions: ambient temperature varied between 20–60 °C, smoke/gas concentration simulated with a controlled smoke source, motion events triggered manually, and current load varied using a resistive dummy load. Data were sampled at 1 Hz and logged for 30 minutes per test scenario.

4.1 Metrics and Equations

We evaluate measurement fidelity using root-mean-square error (RMSE) against laboratory-grade references, and system responsiveness using the detection delay (time between a true event and system alert). Let y^i denote the calibrated sensor reading and yi the ground-truth reference; for N samples the RMSE is:

$$\mathbf{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{i-}\,\hat{y}_i)^2}$$

To quantify relative deviation we use percentage error for each parameter averaged over the experiment:

$$\mathbf{Mean\ Percent\ Error} = \frac{100}{N}\sum_{i=1}^{n}\frac{(y_{i\,-}\,\hat{y}_i)^2}{|y_i|}$$

4.2 Analysis

- **Temperature sensor (LM35):** After calibration, RMSE against a lab thermometer was ±0.45 °C over 20–60 °C (Mean Percent Error ≈ 1.4%). The linear model in calibration (gain and offset) successfully removed systematic bias, with residuals distributed approximately zero-mean.
- **Smoke sensor (MQ-2 via ADC):** The MQ-2 provides qualitative concentration levels; after mapping ADC counts to relative concentration using a piecewise calibration, the system detected smoke onset reliably. During smoke injection tests the alert trigger threshold was crossed within 3–6 s of emission, depending on ventilation—sufficient for early warning.
- **Motion sensor (PIR):** Binary detection achieved 100% true-positive for deliberate motion within sensor field, with no false triggers during quiet periods in 30-minute runs. Response latency was <0.5 s.
- **Current sensor (ACS712):** For steady loads the ACS712 produced RMSE ≈ 0.05 A and Mean Percent Error ≈ 2.3%**.** The current-based cutter-wear inference (monitoring gradual baseline increases) could detect a 10% change in operating current within acceptable statistical confidence over a 5–10 minute moving-averaged window.
- **System responsiveness and reliability:** Flask-based dashboard updated at 1 s intervals. End-to-end detection-to-alert latency (sensor → ADC → Pi processing → web UI update + actuator command) averaged 1.2 **s** (SD ≈ 0.3 s). No dropped samples observed during the test sessions.

V. Conclusion

The proposed industrial monitoring system demonstrates an efficient and affordable solution for integrating sensing, control, and remote supervision in modern industrial environments. By utilizing a Raspberry Pi as the central processing unit and Python for software implementation, the system effectively monitors key operational parameters such as temperature, smoke, motion, and current in real time. The experimental analysis confirms that the developed framework provides reliable measurements, rapid event detection, and seamless human–machine interaction through a web-based interface.

The layered architecture—comprising sensing, processing, and control/communication modules—ensures scalability and flexibility, allowing additional sensors or actuators to be incorporated with minimal modification. Moreover, the Python-based middleware enables straightforward customization of decision logic, data logging, and visualization, which simplifies maintenance and enhances usability.

Overall, this research validates that a low-cost, open-source platform can achieve dependable performance comparable to traditional PLC-based systems while offering greater adaptability and remote accessibility. Future extensions of this work may include implementing cloud data storage, machine-learning-based predictive maintenance, and cybersecurity enhancements to further align the system with Industry 4.0 requirements.

VI. References

1. J. Smith and A. Johnson, *Industrial Automation: Principles and Practice* (Springer, 2018).
2. L. Wang, K. Xu, and B. Li, "A Survey of Modern Industrial Automation Systems," *IEEE Trans. Ind. Electron.* **67**, 234–245 (2020).
3. H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (IIoT): An analysis framework," *Computers in Industry* **101**, 1–12 (2018).
4. X. Mu, Y. Zhao, and W. Zhang, "The applications of Internet of Things (IoT) in industrial management: a science mapping review," *Int. J. Prod. Res.* **62**, 1–20 (2024).
5. R. Chataut and S. Srestasathiern, "A comprehensive review of IoT applications and future directions," *Sensors* **23**, 3487 (2023).
6. B. Chandrayan, P. Gupta, and S. Rao, "IoT Integration in Industry — A Literature Review," (ResearchGate, 2020).
7. F. Qiu, W. Zhou, and S. Liu, "A Review on Integrating IoT, IIoT, and Industry 4.0: A Pathway," *Int. J. Smart Sens. Intell. Syst.* **8**, 45–68 (2025).
8. S. Afrin, M. H. Mahfuz, and J. Lee, "Industrial Internet of Things: Implementations, Challenges, and Applications," *J. Ind. Inform.* **15**, 89–104 (2025).
9. M. S. Rahman, T. K. Roy, and S. Krishna, "Machine learning and internet of things in Industry 4.0: A review," *J. Ind. Inf. Integr.* **3**, 100089 (2023).
10. S. H. Abdulhussain, N. Almasri, and L. Yao, "A Comprehensive Review of Sensor Technologies in IoT," *Computers* **14**, 342 (2025).

11. P. Pun, A. S. Khachane, and B. Pandey, "A Systematic Literature Review of IoT-Based Applications," *CSIT* (2023).
12. I. Behnke and H. Austad, "Real-Time Performance of Industrial IoT Communication Technologies: A Review," *arXiv preprint* arXiv:2311.08852 (2023).
13. R. L. Jones and M. T. Baker, "Python in Industrial Automation: Libraries and Applications," *Ind. Comput. J.* **12**, 105–114 (2021).
14. S. S. Pon, K. Lee, and M. Rahman, "Edge and Cloud Co-processing in IIoT: Survey and Outlook," *Int. J. Distrib. Syst. Tech.* **9**, 25–42 (2022).
15. G. S. S. Chalapathi, V. Chamola, A. Vaish, and R. Buyya, "Industrial Internet of Things (IIoT) Applications of Edge and Fog Computing: A Review," *arXiv preprint* arXiv:1912.00595 (2019).
16. G. G. Samatas, S. S. Moumgiakmas, and G. A. Papakostas, "Predictive Maintenance — Bridging Artificial Intelligence and IoT," *arXiv preprint* arXiv:2103.11148 (2021).
17. A. K. Singh, H. Zhao, and J. Wu, "IoT-assisted predictive maintenance for smart factories," *IEEE Trans. Autom. Sci. Eng.* **18**, 1200–1210 (2024).
18. X. Chen, Y. Liu, and Z. Xu, "Vibration-based Anomaly Detection via Wireless Edge Sensors," *IEEE Sens. J.* **20**, 1234–1245 (2020).
19. H. Lee and J. Park, "Cloud-based Predictive Maintenance for Manufacturing Using Environmental Sensing," *Int. J. Prod. Res.* **58**, 4567–4582 (2021).
20. S. Gupta, M. Verma, and A. Joshi, "MQTT-based Modular IIoT Framework for Remote Monitoring," *Sensors* **22**, 3987 (2022).
21. R. Khan and F. Ahmed, "Environmental Safety Monitoring in Chemical Plants via Hybrid Sensing," *J. Ind. Inf. Integr.* **5**, 100110 (2023).
22. L. Wu, Y. Zhao, and K. Chen, "Survey of IIoT Architectures and Fault-Detection Techniques," *Comput. Ind.* **141**, 103–126 (2022).
23. W. Zhou, J. Li, and X. Wang, "IoT Testbed for Smart Agriculture: System Design and Deployment," *IEEE Access* **8**, 12345–12358 (2020).
24. J. Park and S. Kim, "Prototype Web Dashboard using Python and Flask for Sensor Networks," *Int. J. Sensor Netw.* **16**, 87–98 (2019).
25. A. Santos, M. Pereira, and L. Costa, "Real-Time Industrial Monitoring with WebSockets and Node.js," *Ind. Informatics J.* **10**, 55–67 (2021).
26. P. Rodrigues, L. Almeida, and M. Sousa, "Multi-Sensor Diagnostics in Steel Manufacturing," *IEEE Trans. Ind. Appl.* **56**, 3210–3218 (2020).
27. F. Silva and L. Nunes, "Current-Vibration Fusion for Equipment Health Assessment," *J. Mach. Learn. Appl.* **7**, 78–89 (2022).
28. R. Hernandez, G. Torres, and V. Lopez, "Edge ML for Latency-Free Alerting in IIoT Devices," *Embedded Intell. Syst.* **4**, 15–27 (2023).
29. K. Zhang, M. Li, and Q. Zhang, "Federated Learning in Industrial IoT for Privacy and Efficiency," *IEEE Trans. Ind. Comput.* **19**, 789–800 (2023).