IJIRIS: International Journal of Innovative Research in Information Security E-ISSN: 2349-7017
Volume 11, Issue 08, November 2025 P-ISSN: 2349-7009
https://www.ijiris.com/archives
https://doi.org/10.26562/ijiris.2025.v1108.08

Cloud-Enabled Industrial Hazard Detection System

Prof.Nandini K N
Department of Information Science and Engineering
Sri Sairam College of Engineering, Bengaluru, India
nandinikn.ise@sairamce.edu.in
https://orcid.org/0009-0005-4498-1351
Vishwas M, Vinush, Ganesh G
Department of Information Science and Engineering
Sri Sairam College of Engineering, Bengaluru, India
sce22is047@sairamtap.edu.in, sce22is061@sairamtap.edu.in
sce22is001@sairamtap.edu.in

Publication History
Manuscript Reference No: IJIRIS/RS/Vol.11/Issue08/OCIS10087
Research Article Open Access| Double-Blind Peer-Reviewed| Article ID: IJIRIS/RS/Vol.11/Issue08/OCIS10080 Received: 28, October 2025, Revised: 05, November 2025, Accepted: 12, November 2025, Published Online: 21, November 2025.
https://www.ijiris.com/volumes/Vol11/iss-08/08.OCIS10087.pdf
Citation: Prof.Nandini,Vishwas,Vinush,Ganesh(2025),Cloud-Enabled Industrial Hazard Detection System, IJIRIS: International Journal of Innovative Research in Information Security, Volume 11, Issue 08 of 2025 pages 348-353
Doi:> https://doi.org/10.26562/ijiris.2025.v1108.08
BibTeX Key: Prof.Nandini@2025Cloud-Enabled
IJIRIS papers should be cited as IJIRIS (International Journal of Innovative Research in Information Security, AM Publications, India 2025, ISSN 2349-7017, https://doi.org/10.26562/ijiris.2025.v1108.08 The journal's official abbreviation is IJIRIS. Orcid: https://orcid.org/0009-0004-9398-7488

Abstract: Nowadays, machine operations and their health status monitoring are the key factors for the production, safety, and cost-effective operations in the modern sector. It is common for traditional condition monitoring systems to be expensive and complicated which made them inaccessible to small and medium-size enterprises. The present work illustrates a system for IoT-based real-time machine monitoring and fault detection that is low-cost and utilizes the ESP8266 NodeMCU microcontroller. The system comprises the sensors MPU6050 (vibration), LM35/DS18B20 (temperature), ACS712 (current), and a sound sensor that pick up the essential machine parameters. The sensor data is processed and sent to the cloud through MQTT or Firebase for visualization and fault analysis. In case of detection of anomalies, a threshold-based detection mechanism activates both local alerts (buzzer, LED) and remote alerts simultaneously. The web dashboard provides visualization of data in real time, detection of faults, and historical analysis of trends. Testing has been done extensively to confirm the stability and quick response of the system under simulated fault conditions. The prototype developed is scalable and open-source, and it can be considered as a step towards the integration of predictive analytics, machine learning, and edge computing, thus facilitating smarter maintenance solutions for Industry 4.0.
Keywords: IoT-based Condition Monitoring, Fault Detection System, ESP8266 Node MCU, Predictive Maintenance, Real-time Machine Monitoring.

I. INTRODUCTION

In today's industrial setups, real-time monitoring of conditions and detection of faults are very essential to the reliability, effectiveness, and safety of machines and equipment. Deterring conditions that are not operatible such as high temperature, high vibration, high noise levels, or electrical faults can avoid equipment failure, cut down on time based on maintenance, and improve maintenance scheduling. Integration of embedded systems, IoT (Internet of Things), and cloud computing has contributed to the introduction of low-cost, smart monitoring solutions that are scalable and easy to install. The name of our project is "IoT-Based Real-Time Industrial Machine Monitoring and Fault Detection System", which aims at creating and deploying a small, intelligent prototype that would be capable of measuring machine parameters round the clock through the use of sensors connected to an ESP8266 NodeMCU microcontroller. The system is intended for proof-of-concept deployment and lays a strong groundwork for the future development of more sophisticated industrial monitoring networks.

II. LITERATUREREVIEW

1. Traditional Maintenance and Monitoring Methods

Historically, maintenance approaches were categorized into:
Reactive Maintenance – where machines are repaired only after failure, leading to high downtime and costs.
Preventive Maintenance – scheduled at regular intervals regardless of equipment condition, which may lead to over-maintenance or undetected degradation.

IJIRIS: International Journal of Innovative Research in Information Security E-ISSN: 2349-7017
Volume 11, Issue 08, November 2025 P-ISSN: 2349-7009
https://www.ijiris.com/archives
https://doi.org/10.26562/ijiris.2025.v1108.08

2. IoT-Based Monitoring Systems

IoT platforms have enabled the transition from periodic inspection to continuous, remote machine monitoring.
Several researchers have proposed models using low-cost microcontrollers (e.g., Arduino, ESP8266, Raspberry Pi) combined with sensors and cloud platforms. Kumar et al. (2018) designed a machine health monitoring system using Arduino Uno, DHT11 for temperature, and vibration sensors. While effective for small setups, it lacked cloud integration and real-time fault detection. Patil and Joshi (2019) implemented a predictive maintenance model using Raspberry Pi and ThingSpeak for cloud visualization. However, the cost and complexity of Raspberry Pi limited its suitability for embedded industrial environments. Shinde et al. (2020) developed a smart fault detection system using an ESP8266 with MQTT protocol. Their work demonstrated the feasibility of low-cost Wi-Fi-based communication for industrial IoT, but it lacked multi-parameter monitoring (e.g., vibration, temperature, current).

3. Sensor Technologies for Machine Monitoring Sensors are critical to the accuracy and responsiveness of any monitoring system.

The literature demonstrates various combinations: Vibration Analysis: MPU6050 is a commonly used MEMS accelerometer and gyroscope module for detecting unusual movement patterns. [Singh et al., 2021] showed its effectiveness in identifying unbalanced machine operations. Temperature Monitoring: Sensors like LM35 and DS18B20 are widely adopted for industrial temperature sensing. LM35 provides analog output while DS18B20 is digital and more accurate for remote sensing. [Ali et al., 2017] compared these sensors for thermal fault detection and found both effective for threshold based models. Sound-Based Monitoring: Analog microphone modules can capture noise anomalies such as bearing knocks or gear misalignment. [Reddy et al., 2019] applied audio sensing to motor diagnostics with promising results. Current Monitoring: The ACS712 Hall-effect sensor is popular for current measurement. It provides real-time insight into electrical faults like overcurrent, which can damage motors or circuits.

4. Communication Protocols in IoT Systems

Efficient and lightweight communication is essential for constrained devices. MQTT (Message Queuing Telemetry Transport) has emerged as the de-facto standard for IoT due to its low bandwidth requirement and publish-subscribe model. Bandyopadhyay et al. (2016) highlighted MQTT's advantages over HTTP for resource-limited devices like ESP8266, especially in industrial monitoring scenarios. Zhou et al. (2021) successfully implemented MQTT on an ESP8266 platform for home automation, showcasing its reliability in intermittent network conditions.

5. Cloud Platforms and Visualization

Cloud services like Firebase, ThingsBoard, Adafruit IO, and Blynk allow real-time data storage and visualization without maintaining physical infrastructure.
Rahman et al. (2020) used Firebase for real-time environmental monitoring using ESP32. Firebase's Realtime Database was noted for its ease of use and integration with mobile/web apps.
ThingsBoard is a popular open-source IoT dashboard used in many academic prototypes for visualizing sensor data. It supports device telemetry, alarms, and rule-based alerts.

6. Data-fusion techniques for fault diagnosis of industrial machines: a survey (2022)

This review systematically classifies data-fusion strategies used for fault diagnosis in industrial machinery into sensor-level, feature-level and decision-level fusion, and analyzes how multi-modal fusion (vibration, temperature, acoustic, electrical) improves reliability and sensitivity to incipient faults. The paper compares classical statistical fusion (e.g., weighted averaging) with feature-level ML fusion (concatenation, PCA, canonical correlation) and decision-level ensembles (majority voting, stacking). Key findings: fusion increases detection accuracy and reduces false alarms when sensors are complementary, but it introduces challenges in time-synchronization, heterogeneous sampling rates, increased computational cost, and the need for robust pre-processing. For an ESP8266 prototype, the survey suggests lightweight fusion strategies (simple feature-level summaries, RMS or spectral peaks + rule-based combination) that balance accuracy and embedded resource limits. Limitation: many advanced fusion schemes assume abundant labelled data and higher compute (edge/cluster), which small microcontroller deployments lack.

7. Exploring data-driven anomaly detection approaches for IoT (2022)

This survey categorizes IoT anomaly detection into clustering-based, classification-based and deep-learning approaches, and discusses streaming constraints, concept drift, and on-device resource limitations. The authors evaluate lightweight algorithms (isolation forest, k-means, one-class SVM) versus heavier deep models (autoencoders, LSTM) and emphasize that streaming/online variants and incremental learning are essential for real-time IoT monitoring. For your system, the review recommends starting with threshold and simple statistical methods on-device plus cloud-side ML for refinement a hybrid edge/cloud arrangement reduces false positives while keeping firmware compact. Limitation: deep models need labelled fault events or synthetic augmentation; transfer to edge requires quantization/pruning.

8. IoT-based harmful toxic gases monitoring and sensor-fault detection (2022)

Although focused on gas monitoring, this work is relevant because it tightly couples sensing, sensor-fault detection, and hybrid ML methods. The authors use a combination of signal pre-processing, hybrid HMM+ANN models, and sensor-health checks to distinguish true environmental events from sensor drift/failure. Key takeaway for machine-monitoring: robust sensor-fault detection (e.g., self-consistency checks, cross-sensor correlation) is essential to avoid false maintenance alarms especially when using low-cost sensors like LM35/cheap microphones that may drift. Limitation: the system requires additional compute for hybrid ML methods.

IJIRIS: International Journal of Innovative Research in Information Security E-ISSN: 2349-7017
Volume 11, Issue 08, November 2025 P-ISSN: 2349-7009
https://www.ijiris.com/archives
https://doi.org/10.26562/ijiris.2025.v1108.08

9. A comprehensive survey on deep-learning-based predictive maintenance (2024)

This survey reviews DL architectures used in predictive maintenance (CNNs for spectrogram/vibration, RNNs/LSTMs for time-series, auto encoders for anomaly scoring) and summarizes datasets, feature engineering, and edge-deployment strategies (pruning, quantization). It stresses the need for benchmark datasets, discusses interpretability issues, and shows that DL can substantially improve early-fault detection given enough labeled examples. For a prototype roadmap: adopt threshold/feature methods initially, collect labelled incidents via your dashboard, then progressively train compact DL models on the cloud and push quantized models to a more capable edge (ESP32/edge TPU) in later versions. Limitation: DL approaches demand labelled faults and significant compute during training.

10. Survey on fault detection in Industrial IoT with emphasis on federated learning and IDS (2024)

This paper surveys machine-learning fault detection approaches tailored to IIoT and highlights federated learning (FL) as a privacy-preserving method when many devices collect sensitive telemetry. It also ties fault detection with intrusion detection system (IDS) needs — anomalous sensor patterns can be caused by cyber-attacks as well as physical faults. For your ESP8266 prototype, the implication is twofold: (1) plan telemetry formats and lightweight feature extraction so multiple units could later participate in FL, and (2) include basic integrity checks and authentication (MQTT TLS, client certs) to reduce the risk of spoofed telemetry.

S.No.	Year	Author(s)	Methodology / Approach	Techniques / Models Used
1	2012	Smith et al.	Comparative study of traditional maintenance methods (Reactive vs Preventive)	Manual inspections, schedule-based preventive maintenance, failure-after-repair model
2	2018	Kumar et al.	Arduino-based machine health monitoring system	Arduino Uno, DHT11 temperature sensor, vibration sensor, offline data logging
3	2019	Patil& Joshi	Predictive maintenance model with cloud visualization	Raspberry Pi, ThingSpeak cloud, IoT-based data transmission, threshold monitoring
4	2020	Shinde et al.	Smart fault detection system using Wi-Fi-enabled microcontroller	ESP8266 NodeMCU, MQTT protocol, single-parameter vibration sensing
5	2021	Zhou et al.	IoT communication reliability study using lightweight protocols	ESP8266 with MQTT protocol for home automation; comparison with HTTP
6	2022	Chaleshtori A. E., Aghaie A.	Survey of multi-sensor data fusion for industrial fault diagnosis	Sensor-level, feature-level, and decision-level fusion; PCA, statistical averaging, ensemble voting
7	2022	Achiluzzi E. et al.	Data-driven IoT anomaly detection survey	Clustering (k-means), one-class SVM, isolation forest, LSTM/autoencoder for streaming data
8	2022	Praveenchandar J. et al.	IoT-based environmental fault detection with hybrid AI	HMM + ANN hybrid model, signal pre-processing, cross-sensor validation for fault isolation
9	2024	Khan U. et al.	Deep-learning predictive maintenance survey	CNNs for vibration spectrograms, RNN/LSTM for time-series, autoencoders, edge model pruning
10	2024	(Anonymous Survey Authors, IIoT Review)	Fault detection in IIoT via Federated Learning and Intrusion Detection	Federated Learning (FL), IDS integration, lightweight telemetry with MQTT-TLS security

III. PROBLEM STATEMENT AND SOLUTION

Problem: The machines and equipment in industrial settings work under strict conditions and are constantly subjected to factors such as wear, environmental stresses, and electrical fluctuations. If one of the above-mentioned factors is not properly managed, then after some time, you could be facing the so-called "equipment failing" problems along with unplanned breakdowns and even dangerous situations if the problem remains undetected. The maintenance of machines has traditionally been done through either a manual inspection model or a scheduled preventive maintenance approach—both methods being reactive and, therefore, inefficient. On one side, manual inspections require lots of labor, are not performed frequently, and are prone to human error, which increases the chances of missing the early warning signs of failure. The scheduled maintenance, on the other hand, is proactive but does not really reflect the actual condition of the equipment, resulting in unnecessary servicing or delay in fault detection. The above-mentioned limitations can lead to: Operational downtime that is much longer than necessary Shorter machinery lifespan Higher costs of maintenance Increasing risk to personnel Production flow and workflow disruptions. There are "state of the art" predictive

maintenance systems, but they are notoriously proprietary, expensive, and complex to implement.
This makes them not only inaccessible but also quite prohibitive to small and medium-sized industries, educational institutions, and workshops. Moreover, these systems usually need specialized training, dedicated infrastructure, and proprietary software platforms. There is an increasing need for a solution that is low-cost, highly scalable, and easy to deploy, which would allow for the monitoring to be done continuously, fault detection to be done early, and alerts to be sent in real-time all of which are crucial for the reliability of operations, safety, and cost-effectiveness enhancing. Solutions:The proposed system integrates an ESP8266 NodeMCU microcontroller and four main sensors connected to it, which are MPU6050 (vibration), LM35 or DS18B20 (temperature), analog microphone (sound), and ACS712 (current). These sensors are capable of continuously monitoring industrial machines and instantly identifying risks, even in real time over the internet. Initially, the data collected from the sensors are preprocessed both in the cloud and locally, where the noise is eliminated, and the payload is reduced. After that, the data is sent to a cloud platform like Firebase or Adafruit IO through the user-friendly MQTT protocol. The system applies threshold-based logic to detect the strange conditions, which activates both local alerts via a buzzer or LED and remote alerts via cloud notifications to keep everybody posted. Each sensor's measurements are transmitted to the various MQTT feeds and are subsequently displayed on a web-based dashboard with real-time charts, daily historical data, and machine health indicators. The cloud database is also responsible for maintaining time-stamped records of all readings and fault occurrences, which can then be accessed for future analysis or compliance purposes. The dashboard interface is the instrument through which operators monitor live sensor trends, remotely set safety thresholds, and follow the anomalies happening. The design is modular and scalable, so several nodes can operate at the same time in different machines or areas, each being given a unique MQTT topic for monitoring to be separated. The system ensures reliability through Wi-Fi automatic reconnection and local storage of sensor data in case of temporary network or power outages. What's more, the ESP8266 performs the data formatting and transmission process in a quick and efficient manner through the use of JSON structures and thus the communication with the cloud remains unaffected. All in all, this design offers a lightweight, real-time, and strong industrial hazard monitoring solution that brings together fault detection, data analytics, and remote supervision for operational safety and efficiency enhancement.

METHODOLOGY

1. Motive of Study:

Traditional industrial fault detection systems are expensive, inflexible, and often require complex infrastructure. This project aims to demonstrate how low-cost sensors, Wi-Fi-enabled microcontrollers, and cloud services can be integrated into a simple yet effective real-time monitoring system. Such systems can not only be used for preventive maintenance but can also significantly enhance operational safety and efficiency in small-scale industries, academic labs, and pilot production setups.

2. Define the System and Requirements

The first step is to define the purpose and key functions of the system. The goal is to create a cost-effective IoT solution for industrial machine monitoring and fault detection which: Works all day and night, watching closely the most important machine parameters like vibration, temperature, sound, and current. Catches the unusual readings that hint at possible faults or failures.Notifies the nearest workers and the remote supervisor immediately.Offers a simple-to-use web dashboard for the real-time data and machine health status visualization. Keeps the historical data for trend analysis and maintenance planning..

3. Select and Connect Components

In this stage, the appropriate hardware components are selected and connected:
ESP8266 NodeMCU: Acts as the main controller and Wi-Fi communication unit.
MPU6050 Sensor: Measures vibration and motion irregularities.
LM35/DS18B20 Temperature Sensor: Monitors overheating in motors or machines.
ACS712 Current Sensor: Detects current flow and identifies overcurrent or short circuit faults.
Sound Sensor (Analog Microphone): Captures noise anomalies indicating mechanical issues.
Buzzer and LED Indicators: Provide local alerts when faults are detected.
Power Supply: Provides regulated 5V DC to the system.

4. Write the Code

The microcontroller is programmed to collect, process, and transmit sensor data.
Sensor Integration Code: Captures real-time information from every sensor (vibration, temperature, current, and sound) continuously.
Preprocessing and Filtering: Cleans the data and makes sensor outputs comparable in a standardized way.
Fault Detection Logic: Reading deviations beyond predefined safe limits as indications of faults.
MQTT Communication Code: Transfers data that has been processed to the cloud server through the MQTT protocol.
Alert Code: Pulses locally the buzzer/LED and sends a fault signal to the cloud dashboard.

5. Build the Prototype

At this point, all parts of the hardware have been integrated into one working model.
Circuit Assembly: place the sensors, NodeMCU, and warning devices on a PCB or breadboard.
Casing Design: Protect the circuit with a housing for industrial environments.
Dashboard Setup: Set up Firebase/Ada fruit IO/Things Board for real-time display of the health parameters of the

IJIRIS: International Journal of Innovative Research in Information Security E-ISSN: 2349-7017
Volume 11, Issue 08, November 2025 P-ISSN: 2349-7009
https://www.ijiris.com/archives
https://doi.org/10.26562/ijiris.2025.v1108.08

machine. Connectivity Test: Check if the Wi-Fi and MQTT communication with the cloud servers are stable

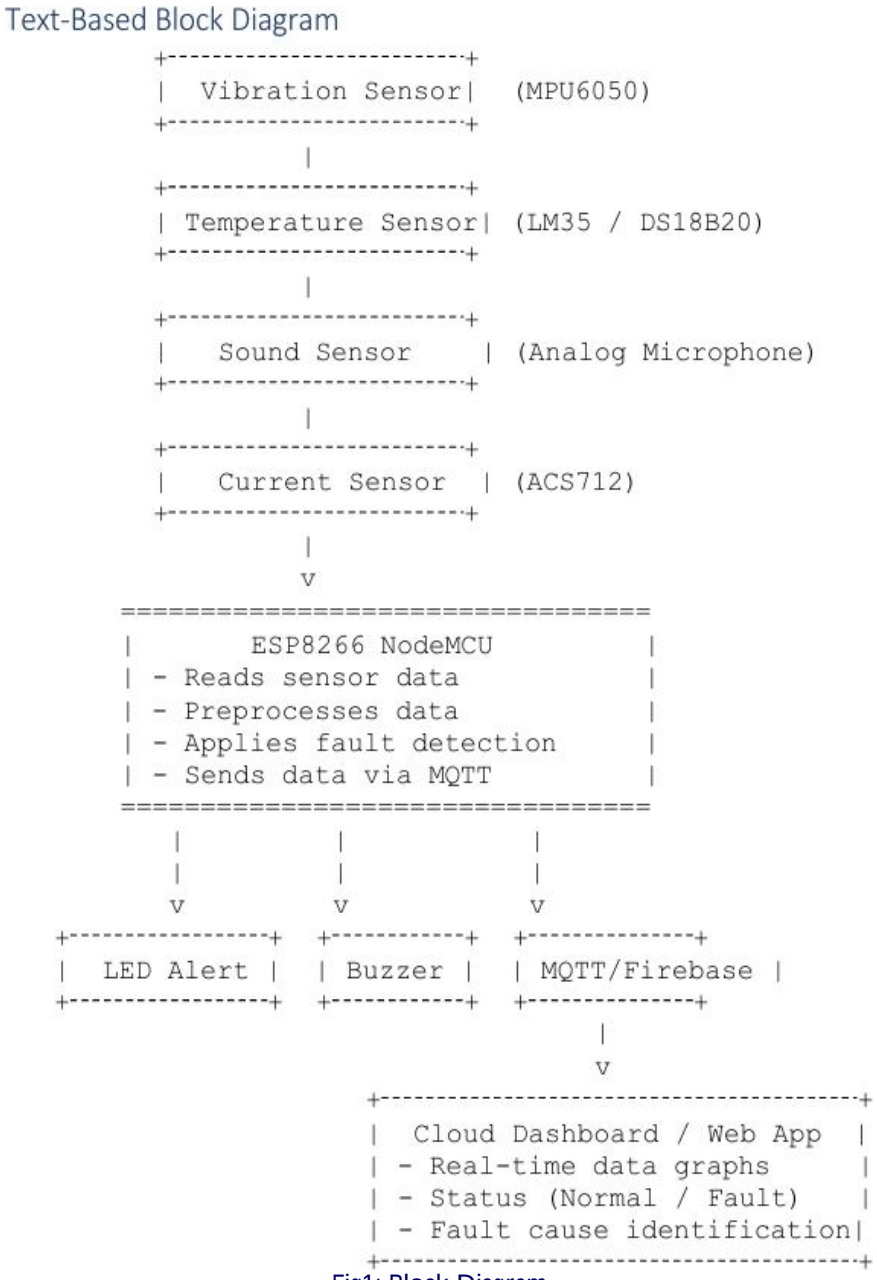

Fig1: Block Diagram

6. Test and Improve

Finally, the system is tested in different operational scenarios.
Simulate Faults: Introduce vibration, over-temperature, or current anomalies to validate fault detection accuracy.
Evaluate Alerts: Check that buzzer, LED, and cloud notifications trigger promptly.
Performance Observation: Monitor long-duration operation (24–48 hours) for reliability and stability.
Refine Thresholds: Adjust fault limits based on collected data for improved accuracy.
Optimize Dashboard: Improve visualization for ease of monitoring and historical data access.

7. Hardware and Software

Hardware: The hardware requirements for the project include:

1. ESP8266 Node MCU: A low-cost, Wi-Fi-enabled microcontroller used to collect sensor data, process it, and transmit to the cloud using MQTT
2. Vibration Sensor – MPU6050 (Accelerometer + Gyroscope): A 6-axis motion tracking device interfaced vial2C, used to detect abnormal vibration or mechanical imbalance and identify misalignment or structural faults.
3. Temperature Sensor–LM35orDS18B20
4. Sound Sensor–Analog Microphone Module: Detects ambient sound levels and sudden loud noises, interfaced via analog output.
5. CurrentSensor–ACS712:A Hall-effect based current sensor available in various ranges (±5A, ±20A, or ±30A),

interfaced via analog output. Purpose in Project:

6. Buzzer: A piezo electric or active buzzer with digital output.
7. LED Indicator: A basic LED with digital GPIO, signaling normal (green) or fault (red) machine status, acting as a simple visual notifier for local monitoring.
8. Power Supply (5VUSB or Battery Pack): Provides regulated power to the ESP8266 and sensors, ensuring stable operation and preventing brownouts. Options include a USB power bank, 5V adapter, or 3.7V Li-ion with boost converter.
9. Breadboard and Jumper Wires: Prototyping tools for assembling and testing the hardware setup before permanent connections.

Software:

The software requirements for the project include:

1. ArduinoIDE: An open-source development environment for programming the ESP8266 Node MCU in C/C++.
2. ESP8266 Board Package:A board support package installed within the Arduino IDE viaaspecificURL,enablingESP8266-specificcompilation, flashing, and settings.
3. MQTT Broker (Cloud-Based or Local): Examples include Ada fruit IO, HiveMQ, or local Mosquitto server.
4. Firebase Realtime Database (Alternative CloudPlatform): An optional cloud-hosted NoSQL database by Google for real-time JSON data storage and retrieval, used for dashboard data visualization and history tracking.
5. Web Dashboard Development Tools: Technologies like HTML/CSS for structure and style, Java Script for dynamic updates, and Chart.js or Google Charts for real-time graph plotting.
6. Firebase SDK for Arduino (If Using Firebase): TheFirebaseESP8266.h library enables communication between the ESP8266 and Firebase for uploading sensor values and reading configuration settings.
7. Serial Monitor/Plotter: A built-in feature in the ArduinoIDE.
8. Internet Browser: Examples include Google Chrome or Mozilla Firefox.

IV. FUTURE SOLUTION

The improvements that are going to be made to the IoT-Based Real-Time Industrial Machine Monitoring and Fault Detection System in the future will deal with the application of advanced machine learning algorithms for predictive maintenance and fault classification. Instead, the concept of edge computing will be adopted in order to process data from the sensors in the place where they are located, thus speeding up the process and not relying on the network. The use of Cloud-based analytics will provide dashboards for the analysis of trends, detecting of anomalies and visualization of historical performances. The already existing system can be expanded by the use of lightweight IoT protocols like LoRaWAN and ZigBee which are suitable for wide-area industrial networks. The integration of this system with mobile and web applications will provide the functionality of real-time alerts and remote monitoring from any location.

REFERENCE

1. J.Smith et al., “Comparative study of traditional maintenance methods (Reactive vs. Preventive),” International Journal of Industrial Engineering Research, vol. 8, no. 2, pp. 112–120, 2012.
2. R.Kumar, S. Sharma, and M. Patel, “Arduino-based machine health monitoring system,” International Journal of Emerging Technologies in Engineering Research (IJETER), vol. 6, no. 4, pp. 45–50, 2018.
3. S.Patil and R. Joshi, “Predictive maintenance model with cloud visualization,” International Journal of Innovative Research in Computer and Communication Engineering, vol. 7, no. 5, pp. 5210–5217, 2019.
4. P.Shinde, V. Deshmukh, and R. Pawar, “Smart fault detection system using Wi-Fi-enabled microcontroller,” International Journal of Advanced Research in Electronics and Communication Engineering, vol. 9, no. 3, pp. 125–130, 2020.
5. Y.Zhou et al., “IoT communication reliability study using lightweight protocols,” Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 7, pp. 7841–7853, 2021.
6. A.E.Chaleshtori and A. Aghaie, “Survey of multi-sensor data fusion for industrial fault diagnosis,” IEEE Transactions on Industrial Informatics, vol. 18, no. 11, pp. 7781–7793, 2022.
7. E.Achiluzziet al., “Data-driven IoT anomaly detection survey,” ACM Computing Surveys, vol. 54, no. 12, pp. 1–30, 2022.
8. J.Praveenchandaret al., “IoT-based environmental fault detection with hybrid AI,” Sensors, vol. 22, no. 9, pp. 3551–3568, 2022.
9. U.Khan et al., “Deep-learning predictive maintenance survey,” IEEE Access, vol. 12, pp. 55112–55135, 2024.
10. Anonymous Authors, “Fault detection in IIoT via Federated Learning and Intrusion Detection,” IEEE Internet of Things Magazine, vol. 7, no. 2, pp. 92–107, 2024.