International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

PredX: An IoT and Unsupervised Machine Learning-Based Predictive Maintenance Framework for Brush DC Motor Condition Monitoring

Aaryan Jai[1], Akshith Chiniwal[2], Gagan HS[3]
Department of Computer Science and Engineering
East West Institute of Technology
Bengaluru, Karnataka, India

Dr. Usha B.S.
Project Guide
Department of Computer Science and Engineering
East West Institute of Technology
Bengaluru, Karnataka, India

Abstract

Predictive maintenance aims to support maintenance decisions using current equipment-condition information before a developing abnormality becomes a disruptive failure. This paper presents PredX, a portable academic prototype for condition monitoring of a high-speed brush DC motor using Industrial Internet of Things (IIoT) connectivity and unsupervised machine learning. The system acquires vibration, temperature, electrical, and rotational-speed measurements using an MPU6050 inertial sensor, DS18B20 temperature sensor, INA219 voltage/current monitor, and A3144 Hall-effect sensor, with an ESP32 acting as the edge controller. Telemetry is transmitted through MQTT to a Python FastAPI backend, stored in SQLite through SQLAlchemy, and exposed through a React/Vite dashboard. A hybrid analytics layer combines configurable engineering thresholds, state-aware motor-off filtering, and an Isolation Forest anomaly detector operating on multivariate rolling-window features. The architecture is intended to provide interpretable rule-based warnings while retaining the ability to identify unusual combinations of sensor values without a large labeled fault dataset. The current contribution is an end-to-end, low-cost condition-monitoring and anomaly-detection workflow; it is not presented as a validated Remaining Useful Life predictor. Quantitative claims regarding detection performance, latency, false-alarm rate, or failure lead time require controlled physical experiments and reproducible ground-truth data.

Keywords: Predictive Maintenance, Industrial Internet of Things, Condition Monitoring, ESP32, Brush DC Motor, Vibration Monitoring, MQTT, Isolation Forest, Anomaly Detection, Industry 4.0.

I. INTRODUCTION

Industrial machines exhibit measurable changes in operating condition before many disruptive events occur. Vibration can reveal mechanical changes, temperature can indicate thermal stress, current can reflect loading or increased resistance, and rotational speed can expose speed loss or stall-like behavior. Condition monitoring therefore provides a basis for moving from corrective maintenance toward maintenance supported by observed machine condition. In Industry 4.0 environments, IIoT connectivity allows sensor measurements to be collected continuously, transported to software services, stored as a time series, and transformed into operator-facing information.

Recent predictive-maintenance research demonstrates several complementary directions. Multi-sensor IoT systems have been used to monitor industrial motors and feed machine-learning classifiers [1], [2]. Conveyor-motor studies have explored time-varying data representations and convolutional neural networks [3]. Event-based IoT and machine-learning architectures have connected industrial data streams to predictive workflows [4]. At the same time, systematic reviews emphasize that model selection depends strongly on the maintenance task, data characteristics, and validation method [5], [6].

PredX is designed around a related but distinct problem: demonstrating a complete low-cost condition-monitoring pipeline for a compact brush DC motor where the available fault data may be too limited to support a conventional supervised classifier. The system therefore combines interpretable engineering thresholds with unsupervised anomaly detection using Isolation Forest. The design includes an operating-state filter so that stationary accelerometer/gravity measurements are not incorrectly interpreted as active vibration faults.

The remainder of this paper describes the literature basis, problem definition, system architecture, sensing and telemetry design, feature engineering, hybrid anomaly-detection method, evaluation methodology, limitations, and future directions.

II. LITERATURE REVIEW

A. IoT and Multi-Sensor Motor Monitoring

Cakir, Guvenc, and Mistikoglu presented an IIoT-based condition-monitoring system in which temperature, current, noise, and revolution information were used with several machine-learning algorithms for predictive maintenance [2]. Their work demonstrates the usefulness of combining multiple operating features rather than relying on a single signal. PredX follows the same multi-parameter philosophy but uses a different embedded platform and emphasizes unsupervised anomaly detection plus rule-based state logic.

Akyaz and Engin developed an ESP32-based predictive-maintenance system for artificial-yarn machinery and evaluated multiple machine-learning approaches [1]. This work is particularly relevant to PredX because it shows the practicality of using an ESP32-class edge device and multiple sensors in an industrial maintenance workflow. The distinction is that PredX uses a compact brush DC motor and a hybrid threshold-plus-Isolation-Forest approach rather than a supervised/deep-learning evaluation of an industrial yarn machine.

Yousuf et al. proposed IoT-based health monitoring and fault detection for an industrial AC induction motor using temperature, vibration, current, voltage, and speed sensors [8]. The sensor diversity closely parallels PredX, although the monitored asset is an AC induction motor and the communication architecture differs. The comparison supports the design choice of combining mechanical, thermal, electrical, and speed measurements.

B. Machine Learning for Predictive Maintenance

Kiangala and Wang investigated predictive maintenance for conveyor motors using dual time-series representations and a convolutional neural network [3]. Their study highlights the importance of temporal behavior in motor-condition assessment. PredX also uses rolling windows so that the analytics layer can represent recent mean, variance, extrema, and other local changes rather than evaluating only isolated sensor samples.

Calabrese et al. proposed SOPHIA, an event-based IoT and machine-learning architecture for predictive maintenance in Industry 4.0 [4]. The architecture emphasizes integrating sensing, events, data processing, and predictive logic. PredX adopts a smaller local architecture but follows the same principle of connecting machine events to an operational alert lifecycle.

Carvalho et al. reviewed machine-learning methods applied to predictive maintenance and showed that model selection is closely related to data availability and the type of maintenance problem [5]. Zonta et al. further reviewed predictive maintenance in Industry 4.0 and highlighted the need for integration across data analytics, distributed systems, and engineering domains [6]. Sürücü, Gadsden, and Yawney reviewed machine-learning-based condition monitoring and emphasized the interaction between data processing, diagnostic task, and model selection [7].

C. Unsupervised Anomaly Detection

Isolation Forest was introduced by Liu, Ting, and Zhou as an unsupervised anomaly-detection method based on randomized partitioning trees [9]. The method isolates unusual observations through shorter partition paths and is useful when abnormal samples are scarce or not exhaustively labeled. PredX uses this algorithm as a condition-assessment layer rather than as a direct failure-time predictor.

Swana, Bokoro, and Doorsamy later examined Isolation Forest and Local Outlier Factor for anomaly detection on a rotating electrical machine [10]. Their work is directly relevant to PredX because it demonstrates the application of unsupervised anomaly detection to an electrical rotating asset. PredX extends this idea to a low-cost brush DC motor using a multivariate feature vector containing thermal, electrical, speed, and vibration information.

D. Literature Gap

The literature supports IoT-based sensing, multi-parameter motor monitoring, temporal feature extraction, supervised predictive maintenance, and unsupervised anomaly detection as separate or partially integrated capabilities. A practical gap for an academic demonstrator is a compact end-to-end system that combines these elements in one reproducible workflow: physical sensing → edge acquisition → MQTT transport → persistent storage → hybrid rule/ML analysis → alert lifecycle → browser visualization. PredX addresses this architectural gap while keeping its scientific claims limited to condition monitoring and anomaly detection until controlled measurements are collected.

TABLE I
COMPARISON OF RELATED STUDIES

Study	Main approach	Relation to PredX
Akyaz & Engin [1]	ESP32-based PdM; multiple ML/DL methods	Strong ESP32 + multi-sensor match; different supervised/deep-learning focus.
Cakir et al. [2]	IIoT CMS; temperature, current, noise, revolution	Supports multi-parameter monitoring; PredX uses hybrid rules + Isolation Forest.
Kiangala & Wang [3]	Temporal motor data + CNN	Supports rolling/time-varying features; different model and asset.
Calabrese et al. [4]	Event-based IoT + ML architecture	Supports event-driven integration; PredX is smaller/local.
Sürücü et al. [7]	Review of ML condition monitoring	Supports task/data/model selection rationale.
Swana et al. [10]	Isolation Forest + LOF on rotating machine	Closest support for unsupervised anomaly detection on electrical machinery.

III. PROBLEM STATEMENT AND OBJECTIVES

The objective is to develop a portable IoT-enabled condition-monitoring and anomaly-detection prototype for a 12 V brush DC motor that continuously measures vibration, temperature, electrical behavior, and rotational speed; transmits telemetry over MQTT; stores historical observations; detects abnormal patterns using deterministic thresholds and an unsupervised model; and exposes machine-health information through a web dashboard.

A. Objectives

- Measure three-axis acceleration and derive vibration magnitude.
- Measure motor-body temperature, bus voltage, and current.
- Measure rotational speed using Hall pulses.
- Publish timestamped telemetry using MQTT.
- Persist sensor readings, alerts, and anomaly predictions.
- Apply configurable/dynamic thresholds for interpretable warning and critical states.
- Detect multivariate anomalies using Isolation Forest on rolling-window features.
- Suppress misleading vibration alarms when the motor is off or idle.
- Maintain an operational alert lifecycle from normal to warning/critical and back to resolved.
- Provide live browser visualization suitable for an academic demonstration.

IV. PROPOSED SYSTEM ARCHITECTURE

PredX is organized into seven functional layers: physical asset, sensing, edge acquisition, IoT messaging, backend/storage, analytics/alerting, and operator visualization. The separation of layers permits independent replacement or enhancement of hardware, communication, storage, or analytics components.

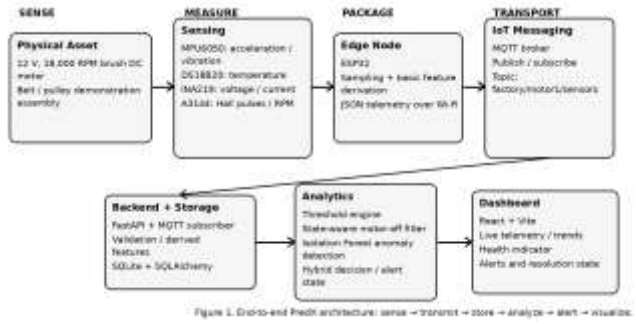


Figure 1. End-to-end PredX architecture: sense → transmit → store → analyze → alert → visualize.

A. Physical Asset and Sensing

The monitored asset is a non-gear brush DC motor rated at 12 V and approximately 18,000 RPM, driven through a BTS7960/IBT-2 motor driver. The sensing subsystem uses an MPU6050 for three-axis acceleration, a DS18B20 for temperature, an INA219 for bus voltage/current monitoring, and an A3144 Hall sensor with a magnetic target for rotational-speed measurement. The report emphasizes rigid vibration-sensor mounting and explicit documentation of sensor orientation, sampling rates, and pulses per revolution.

B. Edge and Messaging Layer

The ESP32 performs sensor initialization, sampling, basic feature derivation such as vibration magnitude and RPM, JSON packet construction, and MQTT publication. Heavy database and analytics operations remain on the backend. MQTT provides a lightweight publish/subscribe transport and decouples the sensor node from the processing service.

C. Backend, Storage, and Dashboard

A Python FastAPI backend receives and validates MQTT telemetry, derives additional features, stores records in SQLite through SQLAlchemy, evaluates rules and anomaly models, maintains alert state, and exposes statistics to the frontend. The React/Vite dashboard presents live sensor values, trends, health information, and alert status. Simulated inputs, when used for demonstrations, must remain clearly distinguished from measured physical data.

V. METHODOLOGY

A. Data Acquisition and Telemetry

The workflow begins by powering and initializing the ESP32 and all sensors. The controller connects to Wi-Fi and the configured MQTT broker, samples acceleration, temperature, voltage/current, and Hall pulses, derives vibration magnitude and RPM, and publishes a timestamped JSON record. A representative telemetry record contains machine ID, timestamp, voltage, current, temperature, RPM, acceleration components, and vibration magnitude.

B. Feature Engineering

For accelerometer components a_x, a_y, and a_z, PredX derives the total acceleration magnitude as:

$$A = \sqrt{(a_x^2 + a_y^2 + a_z^2)}$$

Because a stationary accelerometer can report approximately 1 g due to gravity, the implementation must document orientation, mounting, filtering, and whether vibration is represented by raw magnitude, a baseline-corrected value, or a dynamic component. For RPM, when N pulses are observed over Δt seconds and P pulses correspond to one revolution:

$$RPM = (N / P) \times (60 / \Delta t)$$

Rolling windows are used to reduce sensitivity to single noisy samples and to derive time-local statistics such as mean, variance, minimum, maximum, and slope. The conceptual feature vector is [temperature, current, RPM, a_x, a_y, a_z, vibration magnitude, vibration variance]. The final paper should record the exact feature order, window length, stride, preprocessing, and missing-value policy used by the implementation.

C. Hybrid Decision Logic

The analytics layer combines explicit engineering thresholds with the Isolation Forest anomaly score. Threshold rules provide transparent responses to conditions such as excessive temperature, elevated current, high vibration, or low RPM. Isolation Forest adds multivariate anomaly detection when labeled fault classes are unavailable. The final precedence between rule severity and ML anomaly state must match the implementation actually tested.

D. Motor-Off State Filtering

The documented prototype uses a motor-off rule based on very low current and low RPM. This state prevents stationary gravity-related accelerometer readings from being interpreted as an active-machine vibration fault. This filtering step is important because anomaly scores should be interpreted in the context of the machine operating state.

E. Alert Lifecycle

The alert manager represents four principal operating conditions: NORMAL, WARNING, CRITICAL, and IDLE/OFF. A normal reading updates telemetry without an active alert. A warning or moderate anomaly opens or maintains a warning alert. A critical rule violation or strong anomaly opens or maintains a critical alert. When the motor returns to a normal condition, the active alert can be resolved automatically. This lifecycle converts raw anomaly scores into an operational monitoring workflow.

VI. ISOLATION FOREST-BASED ANOMALY DETECTION

Isolation Forest is an unsupervised tree-based anomaly detector. Instead of learning a set of named fault classes, it isolates observations through randomized partitions. Unusual observations tend to be isolated using fewer partitions than ordinary observations [9]. This property is suitable for early prototype research where obtaining a large, correctly labeled set of bearing, overload, thermal, and speed faults may be difficult.

PredX uses a rolling multivariate feature vector combining thermal, electrical, speed, and vibration information. The model should be trained on a representative reference period dominated by acceptable operation. Important reproducibility parameters include n_estimators, max_samples, contamination, random seed, feature order, scaling or preprocessing, window size, stride, training period, and model version.

A key scientific limitation is that an Isolation Forest anomaly score is not equivalent to a future failure time. An anomaly indicates deviation from a learned/reference distribution. Remaining Useful Life (RUL) prediction would require time-to-failure observations, an appropriate prognostic formulation, and validation against ground-truth failure progression. PredX therefore describes the current model as unsupervised anomaly detection or condition assessment rather than validated prognostics.

VII. IMPLEMENTATION AND DATA MANAGEMENT

A. Software Stack

The documented implementation uses Arduino/ESP32 firmware for edge acquisition; Python with FastAPI, an MQTT client, SQLAlchemy, and scikit-learn for backend processing; SQLite for local persistence; and React/Vite with JavaScript, Recharts, and Lucide for the dashboard. The architecture is modular so that components can later be replaced by a stronger database, cloud service, or edge-inference deployment.

B. Data Model

Three logical data groups are defined: Sensor Readings, Alerts, and ML Predictions. Sensor readings contain timestamped telemetry including voltage, current, temperature, RPM, acceleration components, and derived vibration magnitude. Alerts store event type, severity, status, message, creation time, and resolution time. ML prediction records store the model version, anomaly score, and application-defined health score.

C. Data Quality and Reproducibility

The backend should reject malformed JSON, verify required fields, reject or flag impossible values, detect stale timestamps, handle missing sensor readings, and record software/model versions. For reproducibility, the final experiment should freeze firmware and library versions, sensor sampling rates, threshold configuration, model parameters, database schema, wiring/pin mapping, and feature-processing definitions.

VIII. EXPERIMENTAL EVALUATION AND DISCUSSION

The current project report does not contain a statistically sufficient experimental dataset and explicitly requires controlled measurements before numerical performance claims are made. Therefore, this paper defines the evaluation protocol rather than inventing accuracy, false-alarm, downtime-reduction, cost-saving, or industrial-validation results.

A. Controlled Test Conditions

- T1 — Motor off: current and RPM near zero; no active vibration fault should be attributed to gravity baseline.
- T2 — Normal operation: establish the baseline distribution of current, RPM, temperature, and vibration.
- T3 — Increased mechanical load: observe current rise and possible RPM reduction under a controlled load.
- T4 — Controlled vibration disturbance: verify that vibration magnitude/variance increases and is detected.
- T5 — Safe thermal stress: observe the temperature trajectory under a controlled and safe test condition.
- T6 — RPM/magnetic interruption: verify unstable or reduced RPM response using a safe test arrangement.
- T7 — Recovery: return conditions to normal and confirm alert resolution.
- T8 — Network interruption: measure reconnection behavior and data loss during a controlled MQTT/Wi-Fi interruption.

C. Discussion Framework

The final experimental discussion should identify which parameter changes first, whether the threshold engine triggers, whether the Isolation Forest also flags the condition, how long detection takes, whether false positives occur, and how the observations compare with related research [1]–[10]. The paper should distinguish measured evidence from engineering interpretation and should explicitly report sensor placement, motor load, environmental conditions, sampling rate, and model configuration.

IX. EXPECTED OUTCOMES AND APPLICATIONS

The intended outcome is a working prototype that demonstrates the complete monitoring loop: sense → transmit → store → analyze → alert → visualize → recover. The system is expected to provide continuous visibility into motor temperature, current, voltage, RPM, and vibration; maintain historical telemetry; generate interpretable threshold alerts; identify unusual multivariate behavior using Isolation Forest; and present machine status through a browser dashboard.

A. Applications

Potential applications include educational rotating-machine test benches, laboratory condition monitoring, fan/pump/conveyor motor prototypes, small-machine monitoring research, and Industry 4.0 teaching demonstrations. The architecture can also serve as a foundation for future multi-machine or cloud-connected monitoring systems.

B. Advantages

The documented advantages are low-cost and modular hardware, multi-parameter monitoring, MQTT decoupling between edge and backend, local persistence, hybrid engineering and ML analytics, real-time visualization, and portable deployment over a local hotspot.

X. LIMITATIONS AND FUTURE ENHANCEMENTS

The current prototype has several important limitations. A small brush DC motor is not equivalent to industrial machinery. Vibration measurements are sensitive to mounting, orientation, and filtering. Hall-based RPM depends on magnetic geometry and pulses per revolution. Isolation Forest can identify unusual behavior but cannot, by itself, prove a specific mechanical failure mode. SQLite is appropriate for a small prototype but not a large fleet-scale time-series deployment. Most importantly, the current system does not establish Remaining Useful Life.

Future work should add frequency-domain vibration features such as FFT-derived features; bearing-specific diagnostic indicators; supervised fault classification using labeled abnormal-condition data; validated RUL modeling; PostgreSQL/TimescaleDB for larger deployments; TLS and authentication for MQTT/API communication; cloud deployment and fleet monitoring; edge inference; maintenance recommendation/work-order integration; and digital-twin or richer mechanical simulation capabilities.

XI. CONCLUSION

PredX presents an end-to-end IoT-enabled condition-monitoring architecture for a high-speed brush DC motor. By combining MPU6050 vibration sensing, DS18B20 temperature measurement, INA219 electrical monitoring,

Hall-based RPM measurement, ESP32 edge acquisition, MQTT telemetry, FastAPI processing, SQLite persistence, React visualization, threshold rules, and Isolation Forest anomaly detection, the prototype connects physical machine behavior with a practical software monitoring workflow.

The primary research contribution at the current stage is architectural and demonstrative: a compact and reproducible pipeline from multi-sensor motor measurements to anomaly analysis and alert visualization. The scientific scope is deliberately limited to condition monitoring and unsupervised anomaly detection until controlled experiments produce ground-truth data and measured performance. Such validation will allow the study to quantify sensor stability, detection latency, false-alarm rate, anomaly-detection performance, and alert recovery behavior, and will provide a stronger basis for comparison with published predictive-maintenance systems.

XII. REFERENCES

[1] T. Akyaz and D. Engin, “Machine Learning-Based Predictive Maintenance System for Artificial Yarn Machines,” IEEE Access, vol. 12, pp. 125446–125461, 2024, doi: 10.1109/ACCESS.2024.3454548.

[2] M. Çakır, M. A. Güvenç, and S. Mistikoglu, “The experimental application of popular machine learning algorithms on predictive maintenance and the design of IIoT based condition monitoring system,” Computers & Industrial Engineering, vol. 151, p. 106948, 2021, doi: 10.1016/j.cie.2020.106948.

[3] K. S. Kiangala and Z. Wang, “An Effective Predictive Maintenance Framework for Conveyor Motors Using Dual Time-Series Imaging and Convolutional Neural Network in an Industry 4.0 Environment,” IEEE Access, vol. 8, pp. 121033–121049, 2020, doi: 10.1109/ACCESS.2020.3006788.

[4] M. Calabrese, M. Cimmino, F. Fiume, M. Manfrin, L. Romeo, S. Ceccacci, M. Paolanti, G. Toscano, G. Ciandrini, A. Carrotta, M. Mengoni, E. Frontoni, and D. Kapetis, “SOPHIA: An Event-Based IoT and Machine Learning Architecture for Predictive Maintenance in Industry 4.0,” Information, vol. 11, no. 4, p. 202, 2020, doi: 10.3390/info11040202.

[5] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. da P. Francisco, J. P. Basto, and S. G. S. Alcalá, “A systematic literature review of machine learning methods applied to predictive maintenance,” Computers & Industrial Engineering, vol. 137, p. 106024, 2019, doi: 10.1016/j.cie.2019.106024.

[6] T. Zonta, C. A. da Costa, R. da Rosa Righi, M. J. de Lima, E. S. da Trindade, and G. P. Li, “Predictive maintenance in the Industry 4.0: A systematic literature review,” Computers & Industrial Engineering, vol. 150, p. 106889, 2020, doi: 10.1016/j.cie.2020.106889.

[7] O. Sürücü, S. A. Gadsden, and J. Yawney, “Condition Monitoring using Machine Learning: A Review of Theory, Applications, and Recent Advances,” Expert Systems with Applications, vol. 221, p. 119738, 2023, doi: 10.1016/j.eswa.2023.119738.

[8] M. Yousuf, T. Alsuwian, A. A. Amin, S. Fareed, and M. Hamza, “IoT-based health monitoring and fault detection of industrial AC induction motor for efficient predictive maintenance,” Measurement and Control, vol. 57, no. 8, pp. 1146–1160, 2024, doi: 10.1177/00202940241231473.

[9] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation Forest,” in Proc. 8th IEEE International Conference on Data Mining (ICDM), 2008, pp. 413–422, doi: 10.1109/ICDM.2008.17.

[10] E. F. Swana, P. Bokoro, and W. Doorsamy, “Isolation Forest and Local Outlier Factor-Based Anomaly Detection on an Induction Machine,” in 2024 IEEE PES/IAS PowerAfrica, 2024, doi: 10.1109/PowerAfrica61624.2024.10759503.

[11] J. Sim, J. Min, S. Kim, S. W. Lee, and J.-H. Choi, “Construction of bearing health indicator under time-varying operating conditions based on Isolation Forest,” Engineering Applications of Artificial Intelligence, vol. 126, p. 107058, 2023, doi: 10.1016/j.engappai.2023.107058.

[12] OASIS Open, “MQTT Version 3.1.1,” OASIS Standard, 2014; ISO/IEC 20922:2016.

[13] Espressif Systems, “ESP32 Series Datasheet and ESP32-WROOM Module Documentation.”

[14] Texas Instruments, “INA219 Zerø-Drift, Bidirectional Current/Power Monitor With I2C Interface.”

[15] TDK InvenSense, “MPU-6000/MPU-6050 Product Specification, Rev. 3.4.”

[16] Analog Devices/Maxim Integrated, “DS18B20 Programmable Resolution 1-Wire Digital Thermometer, Rev. 6.”