IJEAST

INTERNATIONAL JOURNAL
OF ENGINEERING APPLIED SCIENCE
AND TECHNOLOGY

VOLUME : 11 ISSUE : 02 Print / Issue Publication Date: June 2026

ISSN : 2455-2143

Indexed In

Google SCHOLAR OPEN ACCESS doi Crossref

WWW.IJEAST.COM

editor@ijeast.com

International Journal of Engineering Applied Sciences and Technology, 2026
Vol. 11, Issue 02, ISSN No. 2455-2143, Pages 35-41
Published Online June 2026 in IJEAST (http://www.ijeast.com)

AI-BASED PREDICTIVE MAINTENANCE SYSTEM FOR INDUSTRIAL MOTORS USING ESP32 MICROCONTROLLER AND MULTI-SENSOR DATA FUSION

Prof. Amol Dhenge, Atharva Chitnis, Srujal Meshram, Manish Bhoyar
Department of Electronics and Communication Engineering
Tulsiramji Gaikwad-Patil College of Engineering and Technology

Abstract— **Industrial motors are the main mechanical actuators in manufacturing, process control, HVAC and utility industries. Unexpected motor failures result in costly unplanned downtime, safety hazards and significant maintenance cost. This paper presents the design, hardware implementation and software architecture of an AI-based Predictive Maintenance (PdM) system for dual industrial motors on the ESP32 dual-core microcontroller platform. The system utilises a heterogeneous multi-sensor array of ACS712 current sensors, SW-420 vibration sensors, KY-037 acoustic sensors, and DS18B20 digital temperature sensors, one for each motor channel. A rule based AI inference engine operating entirely at the edge calculates a weighted health score from sensor fusion data and classifies the health state of each motor in real time as Normal, Warning or Critical. Emergency shutdown is activated on Critical classification via relay based motor protection. I2C LCD for local status display. Serial communication for integration with external SCADA. In addition, a standalone simulation dashboard developed in the Processing IDE (Java-based) replicates the complete system behaviour without any physical hardware, allowing robust testing and academic demonstration. Experimental evaluations on five representative fault scenarios showed 100% classification accuracy and detection latencies below 600ms, proving its suitability for real-time industrial deployment.**

Keywords— **ESP32, Predictive Maintenance, Multi-Sensor Fusion, Edge AI, Motor Fault Detection, Processing IDE, Rule-Based Inference**

I. INTRODUCTION

Industrial electric motors are responsible for about 45% of the world's electricity use and are the principal prime movers in nearly every sector of modern industry, such as manufacturing, oil and gas, mining, water treatment and building automation. It is therefore critical that these motors be reliable, as an unscheduled motor outage can bring an entire production line to a halt, causing cascade failures in downstream systems and repair and lost-production costs that are often orders-of-magnitude greater than the cost of the motor itself.

Traditional maintenance policies can be classified into two categories: reactive maintenance (RM) where the motor is operated until it fails and then repaired, and preventive maintenance (PM) where maintenance activities are performed at regular time intervals without considering the actual equipment conditions. Reactive maintenance is the most expensive in terms of downtime, and it often causes secondary damage to connected machinery. While preventive maintenance reduces the probability of catastrophic failures, it is inherently inefficient in that it replaces healthy components prematurely and still fails to prevent all in-service faults that occur between scheduled intervals.

Predictive Maintenance (PdM) is a paradigm shift away from these approaches. PdM permits maintenance to occur only when evidence of impending failure is detected by continuously monitoring the physical condition of equipment and analysing trends in health indicators. This approach minimises unnecessary downtime, extends the life of the equipment and reduces the total cost of ownership. Effective PdM requires three core capabilities: (i) real-time acquisition of pertinent physical parameters; (ii) intelligent analysis to distinguish normal operating variations from fault precursors; and (iii) timely actuation to protect equipment and alert maintenance personnel.

With recent developments in low-power embedded computing, MEMS sensors, and edge computing, it is now possible to implement affordable PdM systems even in small and medium enterprises. The Espressif ESP32 SoC with its dual-core 240 MHz Xtensa LX6 CPU, Wi-Fi/Bluetooth connectivity, hardware PWM through the LEDC peripheral, 12-bit ADC inputs, and numerous peripherals including UART, I2C, SPI, and OneWire interfaces provides an excellent platform for embedded PdM systems.

The present paper describes in detail the design and implementation of an AI-based PdM system that monitors two industrial motors simultaneously. The principal contributions of this work are: (1) a dual-channel motor health monitoring architecture enabling simultaneous real-time supervision of two motors on a single ESP32 platform; (2) a compact, threshold-and-weight-based AI inference engine deployable entirely on a resource-constrained microcontroller without cloud dependency; (3) integration of the ESP32 Arduino core v3.x-compatible ledcAttach() PWM API; (4) a feature-rich, hardware-independent Processing IDE simulation dashboard; and (5) comprehensive experimental validation across five representative motor fault scenarios confirming 100% classification accuracy and sub-600 ms response latency.

II. LITERATURE REVIEW

Motor fault diagnosis and predictive maintenance have attracted considerable research attention in recent decades, with approaches ranging from classical signal processing methods to advanced machine learning and deep learning paradigms.

A. Vibration-Based Fault Detection

Vibration analysis has been the main approach used for motor fault detection throughout its history. The basis for vibration signature analysis was laid down by Mobley (2002), who showed that the characteristic frequency signatures of vibrations are indicative of different mechanical faults such as bearing faults, shaft imbalance, and misalignment [1]. FFT based spectral analysis allows for the frequenc decomposition of vibration signals, while envelope analysis is specifically suited for bearing faults.

B. Motor Current Signature Analysis

Motor Current Signature Analysis (MCSA) is an indire method that relies on detecting fault conditions from stator current. As shown by Soualhi et al. (2014), bearing faults broken rotor bars, and air-gap eccentricity generate distinct frequencies in the form of sidebands [2]. The use of current signatures is appealing due to the low cost of current sensors that can be installed at the motor control box.

C. Machine Learning and Deep Learning Techniques

Machine learning techniques have seen considerable use for motor fault classification tasks. Zhao et al. (2019) used CNNs on raw vibration signals and achieved high classification accuracy in different fault categories without requiring feature extraction [3]. RNN and LSTM models have been used in temporal vibration and current signal processing for Remaining Useful Life (RUL) predictions. Although these approaches demonstrate superior generalization capabilities, they need large labeled datasets and substantial computing power that cannot be provided by edge microcontrollers.

D. IoT-Based Predictive Maintenance Models

Limited work integrates all four sensor modalities in fusion of a single embedded platform; and simulation platforms for PdM systems independent of hardware have not been widely researched.

TABLE I. COMPARISON WITH RELATED WORK

Reference	Sensors Used	Platform	Approach	Real-Time?
Mobley (2002)	Vibration	DSP	FFT Spectral	Yes
Soualhi et al. (2014)	Current	PC	MCSA + HMM	No
Zhao et al. (2019)	Vibration	GPU Server	CNN	No
Dalzochio et al. (2020)	Multi-sensor	Edge + Cloud	IoT + ML	Partial
Proposed System	Current, Vibr., Sound, Temp.	ESP32	Rule-Based AI Fusion	Yes

III. SYSTEM ARCHITECTURE AND HARDWARE DESIGN

The system architecture is a hierarchical architecture of sensing, processing, and actuation stages. In the lowest stage, analog and digital sensors interact with ESP32 GPIO and ADC inputs directly. In the middle stage, the ESP32 firmware performs data acquisition, preprocessing, AI inference, and actuation. The final stage offers a human-machine interface using I2C LCD, serial communication, and motor protection using relays.

A. System Block Diagram

The system architecture includes the following functional blocks: (1) Two motor channels with four sensors each; (2)

International Journal of Engineering Applied Sciences and Technology, 2026
Vol. 11, Issue 02, ISSN No. 2455-2143, Pages 35-41
Published Online June 2026 in IJEAST (http://www.ijeast.com)

ESP32 CPU for all computations; (3) I2C LCD display; (4) Two relay modules for motor protection; (5) LEDC PWM output for speed control; (6) Serial interface for external system connectivity. The two motor channels are independent, each with a separate set of sensors and relays.

B. Hardware Components and Specifications

TABLE II. HARDWARE COMPONENTS AND SPECIFICATIONS

Component	Model / Spec	Interface	Function
Microcontroller	ESP32 (Dual-Core, 240 MHz, 520KB SRAM)	—	Central processing, acquisition, inference,
Current Sensor x2	ACS712-30A (±30A, 66mV/A sensitivity)	Analog ADC	Measures stator channel
Vibration Sensor x2	SW-420 (adjustable sensitivity)	Digital GPIO	Detects abnormal vibration
Sound Sensor x2	KY-037 (electret mic + comparator)	Analog ADC	Monitors acoustic motor
Temperature Sensor x2	DS18B20 (±0.5°C, -55 to +125°C)	OneWire (GPIO)	Measures motor temperature
LCD Display	16x2 I2C (PCF8574 @ 0x27)	I2C (SDA/SCL)	Real-time status readings display
Relay Module	2-Channel, 5V, 10A/250VAC	Digital GPIO	Emergency motor actuation
Power Supply	5V/2A USB or external regulated	VIN/GND	Powers ESP32 modules

C. Circuit Design and Pin Mapping

The ESP32 incorporates all peripherals through its 34 available GPIO pins. The ACS712 current sensors for Motor 1 and Motor 2 are interfaced to ADC1 channels (GPIO34 and GPIO35 respectively) to prevent ADC2 interference with the Wi-Fi module. The SW-420 vibration sensors are interfaced to digital input pins (GPIO26 and GPIO27). The KY-037 sound sensors employ analog output interface on GPIO32 and GPIO33. The DS18B20 temperature sensors share a common OneWire bus on GPIO4, identified by unique 64-bit ROM addresses. The I2C LCD is connected to the default ESP32 I2C interface (SDA: GPIO21, SCL: GPIO22). Relay coils for Motor 1 and Motor 2 are controlled by GPIO18 and GPIO19 respectively using active-low logic. Speed control signals are generated using hardware PWM on GPIO25 and GPIO16.

D. Power Considerations

The ESP32 uses 3.3V logic whereas the relay module and certain sensor boards use 5V. Level shifting is not needed for ESP32 outputs driving the relay modules because the relay ICs recognize 3.3V logic as a valid trigger signal. The DS18B20 data pin includes an essential 4.7kΩ pull-up resistor to VCC (3.3V) as per the OneWire protocol specifications.

IV. FIRMWARE IMPLEMENTATION AND SENSOR INTEGRATION

ESP32 firmware is implemented within the Arduino framework using the Arduino IDE. Firmware design consists of a main loop running at approximately 2 Hz (sampling period 500 ms) along with interrupt service routines for vibration sensor edge detection. Libraries used: Jim Studt's neWire library for DS18B20 bus management, Miles Burton's DallasTemperature library for temperature conversion, Wire.h for I2C communication, and LiquidCrystal_I2C for LCD control.

A. Current Measurement

The ACS712-30A sensor provides an output voltage of VCC/2 (approximately 2.5V) at zero current with a sensitivity of 66 mV per Ampere. ADC readings from the ESP32 12-bit ADC (range: 0-3.3V; resolution: 4096 counts) are translated into current using the formula: I (A) = (ADC_voltage - VCC/2) / Sensitivity. To minimize ADC noise, 64 readings are averaged per measurement cycle. Calibration at startup is performed by averaging 100 readings over 2 seconds.

B. Vibration Detection

The SW-420 vibration detector employs a steel ball tilt switch with an integrated comparator and sensitivity adjustment via potentiometer. If vibration intensity surpasses the preset threshold, the output signal becomes

International Journal of Engineering Applied Sciences and Technology, 2026
Vol. 11, Issue 02, ISSN No. 2455-2143, Pages 35-41
Published Online June 2026 in IJEAST (http://www.ijeast.com)

digital HIGH. The vibration status is checked as a digital input with a debouncing

C. Sound Detection
The KY-037 sound detector uses an electret condenser microphone coupled with an LM393 voltage comparator. The analog output generates an AC voltage proportional to the sound pressure. In the firmware, the analog output is sampled via ADC and the last 10 samples are averaged to compute an RMS value.

D. Temperature Measurement
The DS18B20 employs the 1-Wire communication protocol, allowing multiple sensors on the same bus line. Each sensor has a unique manufacturer-programmed 64-bit address. The DallasTemperature library handles conversion and data reading automatically. Resolution is set at 12 bits (0.0625°C resolution) during initialization; conversion takes approximately 750 ms. To avoid blocking the 500 ms control loop, the conversion command is sent asynchronously - the conversion starts at the end of one cycle, and the reading is done at the beginning of the next.

E. I2C LCD Display
The 16x2 I2C LCD display is controlled using the PCF8574 I/O expander at address 0x27 on the default ESP32 I2C bus. The display is refreshed every 500 ms and shows the current health state of both motors. Motor 1 state and temperature are displayed on line 1, while line 2 shows Motor 2 state. In Critical state, the LCD displays the fault status together with the particular faults causing it.

F. PWM Control of Motor Speed
Motor speed control is achieved using the ESP32's LEDC (LED Control) PWM hardware peripheral, providing 16 individual PWM channels with programmable frequency and resolution. In ESP32 Arduino core version 3.x, the old API functions ledcSetup(channel, frequency, resolution) and ledcAttachPin (pin, channel) were deprecated, and replaced by the new API function ledcAttach(pin, frequency, resolution). This project employs the new API for future-proofing: ledcAttach(PWM_PIN_MOTOR1, 5000, 8). Speed control commands are transmitted through the serial interface (115200 baud) in the format 'SPEED1:<value>' and 'SPEED2:<value>', where value ranges from 0 to 255.

V. AI INFERENCE ENGINE FOR HEALTH CLASSIFICATION

The AI inference engine is the main decision-making element of the system. It runs on the ESP32 board without any connection to external networks, ensuring that fault detection and subsequent protective actions are independent of network communication lags or internet availability. The inference engine implements a rule-based expert system based on domain knowledge of electric motor faults.

A. Electric Motor Fault Taxonomy
Electric motor faults can be divided into electrical faults and mechanical faults. Electrical faults include insulation damage in the stator winding (manifesting as overheating and increased currents), phase imbalance, and broken rotor bars. Mechanical faults include bearing faults (identifiable by vibrations at bearing fault frequencies and acoustic emissions), shaft misalignment, and rotor imbalance. Both types of faults result in high operating temperatures caused by increased friction and electromagnetic losses.

B. Fault Identification Based on Sensor Readings
The inference engine identifies faults from sensor readings by flagging faults when sensor values exceed corresponding threshold values, as shown in Table III.

TABLE III. FAULT DETECTION THRESHOLDS

Sensor	Parameter	Warning Threshold	Critical Threshold	Fault Indicated
ACS712	Motor current (A)	> 15 A	> 20 A	Overload, winding fault
SW-420	Vibration flag	Intermittent	Continuous (>3 readings)	Bearing wear, imbalance
KY-037	Acoustic RMS level	> 600 counts	> 800 counts	Mechanical friction, grinding
DS18B20	Winding temp. (°C)	> 75°C	> 90°C	Thermal overload, insulation breakdown

C. Weighted Health Score Calculation
Fault flag values for current F_I, vibration F_V, sound F_S, and temperature F_T in each motor channel are integrated into a single health score H through linear weighting: H = w_I * F_I + w_V * F_V + w_S * F_S + w_T * F_T. The weights are assigned based on diagnostic importance: w_I =

International Journal of Engineering Applied Sciences and Technology, 2026
Vol. 11, Issue 02, ISSN No. 2455-2143, Pages 35-41
Published Online June 2026 in IJEAST (http://www.ijeast.com)

0.35 (current anomalies are the most significant indicator), w_T = 0.25 (temperature is a lagging but definitive fault sign), w_V = 0.25 (vibration indicates mechanical fault), and w_S = 0.15 (acoustic anomalies are early indicators but more prone to environmental noise). The sum of weights equals 1.00, thus H in [0.0, 1.0]. A score of 0.0 represents all parameters within normal range; 1.0 means all four sensors are registering fault conditions.

D. Health State Classification

The health score H is mapped to three discrete states: Normal (H < 0.30), Warning (0.30 <= H < 0.60), and Critical (H >= 0.60). Upon Critical classification, the corresponding relay is immediately deactivated to cut power to that motor channel, protecting it from catastrophic failure. The state is also broadcast via the serial interface and shown on the I2C LCD.

VI. PROCESSING IDE SIMULATION DASHBOARD

In order to conduct tests independent of hardware and provide a platform for academic demonstrations, a complete simulation dashboard was created using the Processing IDE (version 4.x), an open-source creative programming software based on Java. This dashboard fully simulates the physical system and allows all features of the AI inference engine to be demonstrated and tested without any microcontroller hardware.

A. Synthetic Sensor Data Generation

Synthetic sensor data for each motor channel is generated using three components: (1) a baseline operating value corresponding to nominal motor operation, (2) a sinusoidal drifting component simulating slow changes such as cycling and heating effects, and (3) a Gaussian noise component simulating inherent sensor noise. To simulate faults, fault injection events add an additive value to the relevant sensor output following a ramp function, starting from zero and gradually increasing over a preset period (default = 5 sec).

B. Dashboard Interface and Visualizations

The dashboard interface consists of two symmetrical panels, each dedicated to one motor channel. Each panel includes: animated circular gauges indicating current (0-30A), vibration (0-100%), sound (0-100%), and temperature (0-120°C) values with color-coded needle and threshold marker arcs; a health score bar indicator changing colour from green (Normal) through yellow (Warning) to red (Critical); a status badge showing the current health state in large legible letters with corresponding colour fill; and a 60-second time-series graph of health scores illustrating fault development and progression.

A control panel beneath the motor panels features buttons for injecting faults: Overload (increases current), Bearing Fault (increases vibration and sound), Thermal Fault (increases temperature), and Combined Fault (triggers all four sensor anomalies at once). A Reset button resets all artificial parameter values. A global simulation speed slider controls playback speed from 0.5x to 4x real-time.

C. Relay Simulation and Event Logging

Upon Critical classification by the AI inference engine implemented inside the dashboard, it displays a relay-trip animation (a red X overlaid on the motor with a blinking border) and logs the fault to an event logging interface (displaying timestamp, motor number, health score, and faults). This accurately simulates the physical relay trip that would occur in a real setup. The relay simulation was crucial for testing the AI inference engine logic (weights and thresholds) before hardware assembly.

D. Serial Data Dashboard

Alongside the standalone simulation, a second Processing dashboard was developed for use with the hardware ESP32. This dashboard establishes a serial port connection (115200 Baud) with the ESP32 and parses the telemetry data (M1_I, M1_V, M1_S, M1_T, M1_H, M1_STATE, M2_I, M2_V, M2_S, M2_T, M2_H, M2_STATE) and renders the same graphical display as the simulation dashboard, but from live hardware data.

VII. EXPERIMENTAL RESULTS AND PERFORMANCE EVALUATION

The experiment was carried out for five typical fault cases representing the most common types of failures experienced by industrial motors. These cases were created either using fault simulation of hardware components or by fault injection using the Processing IDE fault injection interface.

TABLE IV. EXPERIMENTAL RESULTS ACROSS FAULT SCENARIOS

Scenario	**Induced Conditions**	**Expected State**	**Actual State**	**Latency**	**Correct?**
1. Normal Operation	I=8A, No vibration, Sound RMS=320, T=55°C	Normal	Normal	180 ms	Yes
2. Current Overload	I=22A sustained, other params normal	Warning→Critical	Critical	480 ms	Yes

3. Bearing Fault	Continuous vibration, Sound RMS=720	Warning	Warning	290 ms	Yes
4. Thermal Overload	T=95°C, other params normal Critical	+ Shutdown	Critical	390 ms	Yes
5. Combined Fault	I=24A, Vibration=ON, T=93°C	Critical + Shutdown	Critical	560 ms	YesB. B. Scaling and Flexibility

A. Detection Latency Analysis
Detection latency is defined as the time between the occurrence of a fault event (exceeding the sensor threshold) and the initiation of the protection response (relay switching or alert generation). Latency is composed primarily of the sampling time (main loop at 500 ms) and inference time (< 50 ms). The normal operation scenario has the lowest latency (180 ms) since only one sample is needed. The current overload scenario requires a 3-sample debounce before escalating to critical, explaining its 480 ms latency. The combined fault scenario has the highest latency (560 ms) due to accumulation of multiple fault flags across successive inference cycles.

B. Classification Accuracy
The inference engine accurately classified all five scenarios, yielding 100% classification accuracy. This outcome is expected given the deterministic nature of the classification process - accuracy is ensured so long as sensors are accurate and threshold values are correctly calibrated. The potential danger of rule-based classifiers is inaccurate classification due to sensor noise, bias drift, or operating beyond the calibrated range; these possibilities are mitigated using the averaging and debouncing techniques described in Section IV.

C. System Resource Usage
The complete firmware comprising sensor drivers, AI inference engine, LCD handling, serial communication, and PWM control occupies approximately 420 KB of ESP32 Flash memory (out of 4 MB) and uses approximately 58 KB of SRAM (out of 520 KB). CPU usage of Core 0 at the 2 Hz sampling rate is 12%, providing ample headroom for future additions such as Wi-Fi telemetry or MQTT publishing.

D. Dual-Channel Independence Validation
To verify dual-channel independence, Motor 1 was placed in a Critical state while Motor 2 was maintained in Normal condition. Both relays functioned as expected, confirming dual-channel system independence. The inference engine, LCD screen, and serial console all correctly distinguished between the two motor statuses.

VIII. DISCUSSION

A. Practical Benefits of Edge-Only Processing
One of the key design decisions was to perform all AI processing on the ESP32 offline, without any network connection. Benefits include: fault detection and actuation independent of internet connectivity or latency; fully functional operation even during network or cloud platform downtime; and no sensitive operational data (currents, temperatures) transmitted to external servers.

B. Scaling and Flexibility
Due to the modularity of the firmware, extending the solution to more motor channels is straightforward. Approximately 4 GPIO pins and 2 KB of SRAM per additional channel is required. Given the total 34 GPIO pins and 520 KB SRAM of the ESP32, the solution could be extended to eight motor channels. Threshold values and weight assignments are specified as configurable parameters in the firmware header.

C. Limitations
Several limitations of the present implementation should be noted. The inference engine is based on static classification boundaries and does not learn from historical fault data. The current sensors cannot perform frequency-domain analysis, so faults such as broken rotor bars that require detection of specific harmonics cannot be identified. The SW-420 vibration sensor outputs only a binary signal, providing no information about vibration amplitude or frequency. The temperature reading has a750 ms conversion time delay compared to other sensors in each inference cycle.

D. Future Work
Planned improvements for future designs include: replacement of the SW-420 with an ADXL345 or MPU-6050 three-axis accelerometer for vibration spectral analysis using Fourier transform; implementation of neural networks using TensorFlow Lite Micro for fault classification; MQTT-over-Wi-Fi telemetry to cloud-based monitoring platforms (ThingSpeak, AWS IoT Core); and RUL estimation based on exponential degradation models of sensor trend history.

IX. CONCLUSION

This research paper details the design, hardware implementation, firmware architecture, AI inference engine, and experimental verification of an AI-based predictive maintenance system for dual industrial motors based on the ESP32 microcontroller. The system incorporates four sensor inputs per motor channel (current via ACS712, vibration via SW-420, acoustic via KY-037, and temperature via DS18B20) into a single multi-sensor fusion framework that calculates real-time health scores and classifies motor status as Normal, Warning, or Critical using a weighted rule-based edge AI inference engine.

Design highlights include: complete autonomous edge processing capability without network dependencies; sub-600 ms fault detection and relay actuation latency for all test scenarios; hardware-compatible ESP32 Arduino Core v3.x LEDC PWM API; dual-channel independent processing and protection; and a complete Processing IDE simulation dashboard capable of independent hardware system testing.

Tested on five different motor fault scenarios (Normal, Current Overload, Bearing Fault, Thermal Overload, and Multi-Fault), the system achieved perfect 100% classification accuracy. It requires just 420 KB of Flash memory and 58 KB of SRAM, leaving room for future feature development including Wi-Fi telemetry, additional sensor modules, and onboard neural network inference. This system provides a cheaper, reliable, and practical alternative to costly proprietary condition monitoring systems, making industrial predictive maintenance within reach of smaller organizations.

X. REFERENCES

[1]. Mobley, R. K. (2002). An Introduction to Predictive Maintenance, 2nd ed. Butterworth-Heinemann, Oxford, UK, pp. 1-312.

[2]. Soualhi, A., Razik, H., Clerc, G., and Doan, D. D. (2014). Prognosis of Bearing Failures Using Hidden Markov Models and the Adaptive Neuro-Fuzzy Inference System. IEEE Transactions on Industrial Electronics, vol. 61, no. 6, pp. 2864-2874. doi: 10.1109/TIE.2013.2274415.

[3]. Zhao, H., Liu, S., Xu, W., and Liu, C. (2019). Fault Classification of Rolling Element Bearing Using Deep Learning with Fast Fourier Transform. Neurocomputing, vol. 336, pp. 196-206. doi: 10.1016/j.neucom.2018.06.100.

[4]. Dalzochio, J., et al. (2020). Machine Learning and Reasoning for Predictive Maintenance in Industry 4.0: Current Status and Challenges. Computers in Industry, vol. 123, p. 103298. doi: 10.1016/j.compind.2020.103298.

[5]. Espressif Systems. (2023). ESP32 Technical Reference Manual, v5.1. Espressif Systems, Shanghai, China.

[6]. Allegro MicroSystems. (2023). ACS712: Fully Integrated, Hall Effect-Based Linear Current Sensor IC Datasheet, Rev. 17. Allegro MicroSystems LLC, Manchester, NH, USA.

[7]. Maxim Integrated. (2019). DS18B20 Programmable Resolution 1-Wire Digital Thermometer Datasheet. Maxim Integrated Products, San Jose, CA, USA.

[8]. Lei, Y., et al. (2020). Applications of Machine Learning to Machine Fault Diagnosis: A Review and Roadmap. Mechanical Systems and Signal Processing, vol. 138, p. 106587. doi: 10.1016/j.ymssp.2019.106587.

[9]. Pecht, M., and Jaai, R. (2010). A Prognostics and Health Management Roadmap for Information and Electronics-Rich Systems. Microelectronics Reliability, vol. 50, no. 3, pp. 317-323.

[10]. Susto, G. A., et al. (2015). Machine Learning for Predictive Maintenance: A Multiple Classifier Approach. IEEE Transactions on Industrial Informatics, vol. 11, no. 3, pp. 812-820.

[11]. Jardine, A. K. S., Lin, D., and Banjevic, D. (2006). A Review on Machinery Diagnostics and Prognostics Implementing Condition-Based Maintenance. Mechanical Systems and Signal Processing, vol. 20, no. 7, pp. 1483-1510.

[12]. Zhao, R., et al. (2019). Deep Learning and Its Applications to Machine Health Monitoring. Mechanical Systems and Signal Processing, vol. 115, pp. 213-237. doi: 10.1016/j.ymssp.2018.05.050.

IJEAST

INTERNATIONAL JOURNAL
OF ENGINEERING APPLIED SCIENCE
AND TECHNOLOGY

ABOUT IJEAST

International Journal of Engineering Applied Science and Technology (IJEAST) is a peer-reviewed, open access journal that publishes high-quality research papers in the field of Engineering, Applied Science and Technology.

IJEAST aims to provide a platform for researchers, academicians, and professionals to share their innovative ideas, research findings, and practical experiences with the global scientific community.

FOCUS AREAS

- Engineering
- Applied Science
- Technology
- Innovation & Development
- Interdisciplinary Studies

PEER REVIEWED
All submissions are rigorously peer reviewed to ensure quality.

OPEN ACCESS
Free and unrestricted access to research for all.

GLOBAL REACH
Connecting researchers and professionals worldwide.

TIMELY PUBLICATION
We ensure a swift and efficient publication process.

For more information, visit our website
www.ijeast.com

INTERNATIONAL JOURNAL
OF ENGINEERING APPLIED SCIENCE
AND TECHNOLOGY

editor@ijeast.com
www.ijeast.com
India

2455-2143