Design and Preliminary Experimental Characterization of a Low-Cost ESP32-Based System for Electric Motor Condition Monitoring

Gael Rangel Castañón[1]*, Diego Ortega Pérez[1], Noel Montiel Elizalde[1], Yael Omar Acosta Rangel[1], Irving Yahir Barrera Nochebuena[1]

Instituto Tecnológico de Pachuca, Pachuca de Soto, Hidalgo, México

DOI: https://doi.org/10.47772/IJRISS.2026.102500007

Received: 21 October 2026; Accepted: 26 October 2026; Published: 14 November 2026

ABSTRACT

Small and medium-sized manufacturing enterprises without digital instrumentation detect incipient faults in electric motors by sight and sound, which leaves no record of the vibration and temperature changes that precede a breakdown. This study designed and preliminarily characterized a low-cost prototype, based on an ESP32 microcontroller, that measures vibration and temperature and classifies the motor state against approximate thresholds derived from the International Organization for Standardization (ISO) 10816-1 class limits. It also reports a preliminary proof of concept of a tiny machine learning (TinyML) anomaly detector intended to run on the microcontroller. The research was quantitative, applied and cross-sectional, in two stages. In stage 1, the prototype, with an MPU6050 accelerometer sampled at 20 Hz and a DS18B20 sensor, measured five workshop motors once each. The three motors with a known rated speed fell 71% to 85% below the approximate warning threshold (0.05 to 0.08 m/s² root mean square, RMS), a non-validated classification; a portable drill recorded 1.36 m/s² but could not be classified for lack of a rated speed. In stage 2, the sensor noise floor was characterized at 500 Hz, and a first capture of a running lathe motor gave an RMS 4.6 times that floor, with most energy between 65 and 130 Hz rather than at the rotation frequency. Because faults could not be induced, an INT8 autoencoder of about 3.9 kilobytes was trained on the 155 windows of that single session and tested with synthetic anomalies, reaching a sensitivity of 0.69 and a specificity of 0.95 with a chronological split; the threshold rule detected none. These optimistic figures were not measured on the microcontroller. Low-cost acquisition is feasible, but threshold conversion requires spectral velocity analysis before it can be considered equivalent to the standard. A bench calibration found gain errors below 0.6%, and the detector runs on the microcontroller in about 49 µs per window using 0.01% of the processor time.

Keywords: internet of things, predictive maintenance, TinyML, ESP32, ISO 10816, anomaly detection

INTRODUCTION

Maintenance of rotating machinery has evolved from corrective schemes (repairing after the failure) toward predictive schemes that anticipate deterioration from measurable physical variables, mainly vibration and temperature. This transition has strong economic support: poor maintenance strategies are estimated to reduce the productive capacity of a plant by 5% to 20%, and unplanned downtime costs industrial manufacturers about 50 billion dollars a year worldwide (Coleman et al., 2017).

Mechanical vibration accounts for a large share of the recent literature because it makes it possible to diagnose imbalance and incipient bearing defects before they turn into an unplanned stoppage (Maruthi & Hegde, 2016). The issue is particularly relevant in Mexico: micro, small and medium-sized enterprises represent 99.7% of the country's economic units and generate 71% of employment and 37% of national value added (OECD, 2024), yet they face investment barriers that systematically exclude them from the high-cost industrial monitoring solutions described by Magadán et al. (2022).

The miniaturization of microelectromechanical systems (MEMS) accelerometers and the availability of low-cost microcontrollers with built-in connectivity, such as the ESP32, open an affordable route to

instrumentation that was previously within reach only of professional vibration analyzers (Khan et al., 2019; Magadán et al., 2022).

Prototypes published with this kind of hardware usually compare vibration against a fixed threshold (Pandit et al., 2025) or delegate the diagnosis to an external computer or a cloud platform (Kolok et al., 2025; Janumala et al., 2026). In parallel, machine learning on microcontrollers, known as TinyML, makes it possible to run compact models directly on the sensor node, which removes the dependence on the network and reduces latency (Njor et al., 2024; David et al., 2021). However, the review by Njor et al. (2024) warns that most TinyML research concentrates on image and audio, and that predictive-maintenance datasets are scarce, which justifies generating original data from real motors.

This article presents an ESP32-based prototype that measures vibration and temperature, classifies the motor state with thresholds adjustable by standardized profile (ISO 10816-1 for vibration and NEMA-type insulation classes for temperature) and, in a second stage, designs a TinyML anomaly detector intended to run on the microcontroller itself, whose training and validation are still pending. The contribution is therefore a technology-development and preliminary characterization study, not a validated diagnostic system.

The problem addressed is the absence, in small-scale workshops and production lines, of a low-cost mechanism for the early detection of abnormal vibration and temperature conditions in electric motors. Without digital instrumentation, the difference between a normal reading and a fault reading, of up to twenty times the baseline value according to Auqui Fajardo and Farez Chasi (2024), is perceived only when it is already audible or visible, at which point the deterioration is usually advanced. The problem is delimited to low-power electric motors (up to 15 kW) in workshop and laboratory conditions, and at this stage it does not extend to a production line in continuous operation.

The research question is therefore: to what extent can a low-cost prototype based on an ESP32, a MEMS accelerometer and a digital temperature sensor, with motor-profile thresholds, measure and preliminarily classify the state of a low-power electric motor without depending on network infrastructure, and what experimental protocol is required to validate an on-device TinyML anomaly detector?

The technical justification rests on evidence that low-cost MEMS accelerometers perform adequately for vibration monitoring of low- and medium-power motors (Khan et al., 2019) and that an ESP32 with a MEMS accelerometer supports a functional alert system (Pandit et al., 2025; Kolok et al., 2025). The economic justification is direct: while a complete industrial monitoring system represents an investment of thousands of dollars (Magadán et al., 2022), the prototype described here is built with low-cost, commercially available components, consistent with the resource constraints of most manufacturing SMEs; a bill of materials and total cost have not yet been documented. Finally, technological autonomy characterizes the proposed design: threshold classification and the control panel run on the ESP32, and TinyML anomaly detection is designed to run on the same device.

The objective is to design and preliminarily characterize a low-cost ESP32-based prototype that measures the vibration and temperature of electric motors and classifies their state with approximate thresholds by motor profile, and to define the design and the validation protocol of a TinyML anomaly detector to be executed on the microcontroller.

LITERATURE REVIEW

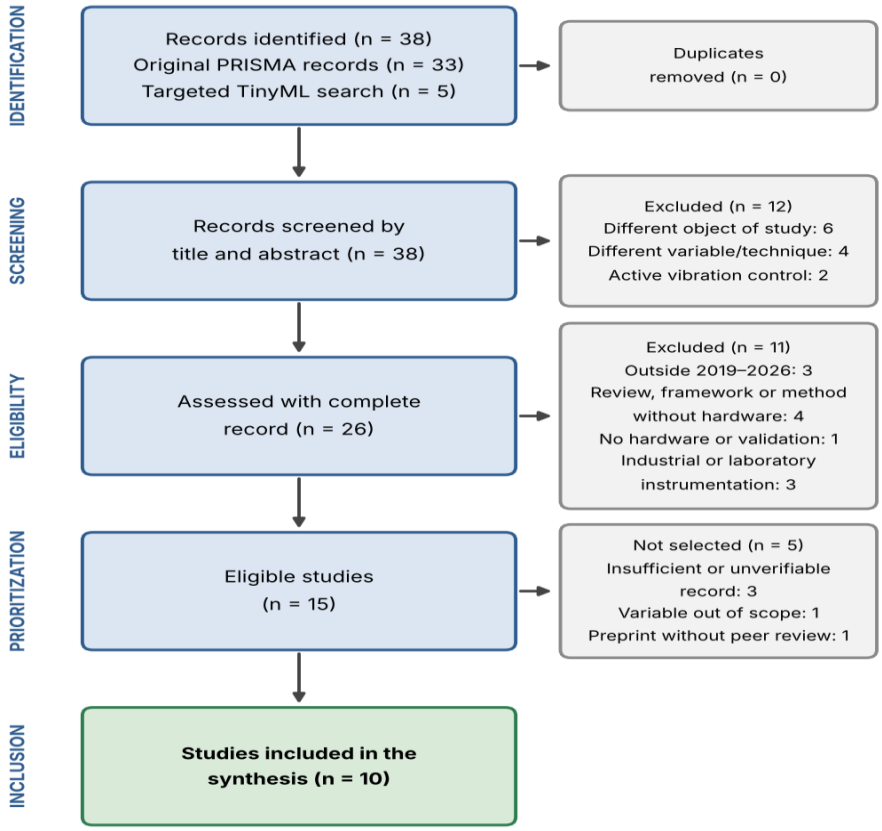
Search and Selection of Studies

Identification relied on open-access and institutional databases and repositories: IEEE Xplore, MDPI, Springer, ACM Digital Library, SciELO, Redalyc, Dialnet, university repositories from Ecuador, Colombia, Peru and Spain, and the arXiv preprint server. Each record found was entered in a form with eight fields (citation, year, location, objective, result, methodology, tools and limitations), which allowed the studies to be compared with the same criteria.

Descriptors in Spanish and English derived from the project keywords were then combined. The English search equation was ("predictive maintenance" OR "condition monitoring") AND ("electric motor" OR "rotating machinery") AND (vibration OR temperature) AND (ESP32 OR "low-cost" OR IoT OR MEMS), and an equivalent equation was used in Spanish. Together they yielded 33 records. When the project added a second, machine-learning stage on the microcontroller, a targeted search was added with ("TinyML" OR "edge") AND ("rotating machinery" OR motor) AND (ESP32 OR microcontroller OR quantization), which contributed five additional records.

The process is summarized in Figure 1. During identification, 38 records were gathered: 33 records from the original PRISMA and 5 from the targeted TinyML search. No duplicates were detected, so all 38 moved on to screening.

Figure 1 PRISMA Flow Diagram of Study Selection



Note. Adapted from "The PRISMA 2020 Statement: An Updated Guideline for Reporting Systematic Reviews," by M. J. Page et al., 2021, *BMJ, 372,* Article n71 (https://doi.org/10.1136/bmj.n71).

At the title-and-abstract screening stage, 12 records were excluded. Six had a different object of study (E1): monitoring of an overhead crane with LiDAR, of railway tracks with LoRa, of structures with NB-IoT, quality control of car bodies, diagnosis of an internal combustion engine, and classification of stoppages of weaving machines. Four more were based on a different variable or technique (E2), such as line current, audio, electrical tests or a non-contact microphone. The remaining two dealt with active vibration control (E3). Consequently, 26 records advanced to assessment with a complete record.

During eligibility, 11 additional records were excluded. Three fell outside the 2019–2026 time window (I4). Four were reviews, software frameworks or methods without a hardware implementation (E4), among them the review of the TinyML stack for predictive maintenance (Njor et al., 2024), the TensorFlow Lite Micro interpreter (David et al., 2021) and an autoencoder method evaluated only on datasets (Ahmad et al., 2021); these three were kept as context references but not as primary studies of the synthesis. One lacked a described

acquisition hardware and experimental validation, and three used industrial or laboratory instrumentation that cannot be replicated at low cost (E5). Fifteen studies were therefore eligible.

Finally, applying prioritization rules P1 to P4, 5 of the 15 eligible studies were discarded: three because of an insufficient or unverifiable record, one because it measured temperature and speed without vibration, and one because it was a preprint without peer review. The final synthesis thus comprised 10 studies. The detail of each excluded record and the reason is available from the authors.

Informative Quality and Characteristics of the Included Studies

To estimate how reliable the information in each included study is, three aspects were assessed from their records: whether it describes the hardware completely (Q1), whether it reports quantitative results (Q2) and whether it states its limitations (Q3). The assessment does not weigh overall methodological quality but the completeness of what each study reports (Table 1).

Table 1 Assessment of the Informative Quality of the Included Studies

Study	Q1 Hardware	Q2 Quantitative results	Q3 Limitations
Arciniegas et al. (2025)	Yes	Yes	Partial
Auqui Fajardo & Farez Chasi (2024)	Yes	Yes	Yes
Brito et al. (2025)	Yes	Yes	Yes
Carvalho de Lucena Filho & Valencia de Almeida (2026)	Partial	Yes	Partial
Gomez Garcia (2025)	Yes	Yes	Yes
Janumala et al. (2026)	Yes	Partial	Yes
Kolok et al. (2025)	Yes	Yes	Yes
Magadán et al. (2022)	Yes	Yes	Yes
Martínez Jiménez (2022)	Yes	Partial	Yes
Pandit et al. (2025)	Yes	Partial	Yes

Note. Partial = the information exists but is incomplete (for example, target platforms without a sensor, or qualitative results without metrics). Q1 = hardware described completely; Q2 = quantitative results reported; Q3 = limitations stated.

Overall, nine studies describe their hardware completely and one partially; seven report quantitative results and three do so partially; and eight state their limitations versus two that do so partially. This indicates that the set is suitable for a narrative synthesis, but not for comparing performance metrics across studies, because the datasets, the faults tested and the metrics do not coincide.

The ten studies were published between 2022 and 2026: two in 2022, one in 2024, five in 2025 and two in 2026. They come from Ecuador, Colombia, Spain, Portugal, Slovakia, India and Brazil, which gives international coverage with a presence of Latin American work. Regarding hardware, five use an ESP32-family board as the acquisition device and a sixth evaluates it as a deployment platform, while the rest rely on commercial modules or industrial instrumentation (Table 2).

Table 2 Characteristics of the Included Studies

Study	Country	Main hardware	Diagnostic method	Main result
Arciniegas et al. (2025)	Ecuador and USA	XIAO ESP32S3 and MPU6050 at 100 Hz	FFT, neural network and K-Means on the device	96.5% accuracy in the laboratory; inference of about 25 ms
Auqui Fajardo & Farez Chasi (2024)	Ecuador	SICK MPB10 sensor, S7-1200 PLC and Node-RED	Comparison of healthy versus forced imbalance	1.46 to 4.44 mm/s healthy; up to 29.36 mm/s with imbalance
Brito et al. (2025)	Portugal	ESP32-S3,	Transformer	25 of 150 configurations

	and Spain	piezoelectric sensor and thermocouples	autoencoder, GANomaly and Isolation Forest	at 100%; worst case 81.8%
Carvalho de Lucena Filho & Valencia de Almeida (2026)	Brazil	ESP32-S3, ESP32-WROOM and STM32F4 (MAFAULDA dataset)	MLP, 1D CNN, LSTM and Random Forest with INT8 quantization	37 KB INT8 MLP with F1 = 0.98; 1D CNN fell from 0.94 to 0.75
Gomez Garcia (2025)	Colombia	ESP8266, XBee modules and ThingSpeak	Visualization and e-mail alert	Peaks of up to 24 mm/s and 39 to 50 °C in a 55 kW motor
Janumala et al. (2026)	India	ESP32, MPU6050, ACS712 and temperature sensor	Fixed thresholds with Blynk and ThingSpeak alerts	Distinguished three simulated faults with no false alerts in normal operation
Kolok et al. (2025)	Slovakia	ESP32-C6 and MPU6050	Isolation Forest on a laptop	Accuracy of 72.9% ± 2.4%; cost under €30
Magadán et al. (2022)	Spain	SensorTile.box and Raspberry Pi 4 (BLE and MQTT)	FFT on the module and on the gateway	Resolution of 0.61 Hz (module) and 0.076 Hz (gateway)
Martínez Jiménez (2022)	Spain	STEVAL-MKSBOX1V1 (BLE) and Raspberry Pi 4	Envelope analysis with Fast Kurtogram	Interference harmonics at 100 Hz dominated the spectrum
Pandit et al. (2025)	India	ESP32 and ADXL345	Experimental threshold of 11.0 m/s² and Telegram alert	No performance metrics reported

Note. FFT = fast Fourier transform; MLP = multilayer perceptron; CNN = convolutional neural network; LSTM = long short-term memory; BLE = Bluetooth Low Energy.

Synthesis of the Included Studies

In the first group, most studies adopt the ESP32 and MEMS accelerometer pair as the acquisition platform. Pandit et al. (2025) combine an ESP32 with an ADXL345, compute the three-axis vibration magnitude and send a Telegram alert when an experimental threshold of 11.0 m/s² is exceeded; however, the method depends on a predefined value and does not integrate the diagnosis of multiple faults. Similarly, Janumala et al. (2026) integrate an ESP32 with vibration, temperature, current and sound sensors, and distinguished three simulated faults without generating false alerts in normal operation, although their prediction relies on fixed thresholds, depends on a stable WiFi connection and was tested only in the laboratory. For their part, Gomez Garcia (2025) validated a system with an ESP8266 and XBee modules on a 55 kW industrial motor, where peaks of up to 24 mm/s and temperatures of 39 to 50 °C were recorded depending on the operating stage, and reported difficulties with sensor adjustment and with the stability of data transmission.

A further step toward intelligent diagnosis is taken by Kolok et al. (2025), who with an ESP32-C6 and an MPU6050 trained an Isolation Forest only on healthy data and reached an accuracy close to 73% at a cost below 30 euros. Nevertheless, the algorithm ran on a laptop, only one fault (imbalance from a locked wheel) was tested at fixed speed and load, and the dataset comprised about 900 samples. Consequently, the four studies agree that cost is not a barrier to instrumenting a motor, but also that diagnosis remains anchored to thresholds or external processing.

A second group solves the instrumentation with commercial modules and a layered architecture. Magadán et al. (2022) compared computing the FFT on the sensor module versus the gateway and found accuracy comparable to that of a professional analyzer at a much lower cost; they also reported a reliable BLE range of

about 11 m and an 8- to 10-fold increase in latency in a plant with metal obstacles. In a similar vein, Martínez Jiménez (2022) implemented an edge, fog and cloud architecture on a real motor and found that the spectrum was dominated by electrical-interference harmonics at 100 Hz, in addition to a BLE communication bottleneck. Auqui Fajardo and Farez Chasi (2024), in turn, used industrial instrumentation and provided a useful quantitative reference: a motor without load vibrated between 1.46 and 4.44 mm/s at a stable temperature of 32 °C, whereas a forced imbalance with a 27 g mass raised the vibration to 29.36 mm/s and the temperature to 43 °C. These three works are valuable because they quantify the real problems of deployment, namely latency, interference and wireless bandwidth; however, none reports a complete diagnosis executed on a single low-cost device.

Finally, three studies take machine learning to the device itself or to its memory limit. Arciniegas et al. (2025) deployed on a XIAO ESP32S3 with an MPU6050 sampled at 100 Hz a neural-network classifier on spectral features and K-Means for anomalies, with 96.5% accuracy in the laboratory, an inference of about 25 ms and false positives caused by external vibrations. Brito et al. (2025), with an ESP32-S3, a piezoelectric sensor and thermocouples, evaluated autoencoders and other unsupervised detectors on a pump bench: of 150 configurations, 25 reached 100% accuracy, 43 reached at least 90% and the worst case was 81.8%. Lastly, Carvalho de Lucena Filho and Valencia de Almeida (2026) showed, on the MAFAULDA dataset, that an INT8 multilayer perceptron of about 37 KB reached F1 = 0.98 and was the only model that, according to its authors, fit on the three platforms evaluated, whereas a 1D convolutional network fell from 0.94 to 0.75 when quantized. These results indicate that small dense networks are a viable option on ESP32-family microcontrollers, although all of them come from laboratory benches or specific datasets.

Low-Cost Vibration and Temperature Monitoring with ESP32

The recent literature shows that the ESP32 and MEMS accelerometer pair is a common platform for motor monitoring. Khan et al. (2019) proposed an IoT system for the health of electric motors with low-cost sensors, and Magadán et al. (2022) reported a low-cost wireless industrial system for monitoring motor condition, at a cost far below that of commercial solutions. Pandit et al. (2025) use an ESP32 and an ADXL345 that compute the three-axis vibration magnitude and send a Telegram alert when a manually set threshold is exceeded, without reporting performance metrics in their abstract. Kolok et al. (2025) use an ESP32-C6 with an MPU6050 and process the data on a laptop with an Isolation Forest trained only on healthy data; the effective sampling rate was about 2 Hz, the accuracy close to 73% and the only fault tested was imbalance from a locked wheel. Janumala et al. (2026) integrate an ESP32 with a cloud platform and suggest, qualitatively, the probable cause of the fault.

Latin American engineering theses are also informative: Auqui Fajardo and Farez Chasi (2024) document that a healthy motor without load vibrates between 1.46 and 4.44 mm/s and that a forced imbalance raises the reading to 29.36 mm/s, while Gomez Garcia (2025) reports peaks of up to 24 mm/s and temperatures of 39 to 50 °C in a higher-power industrial motor depending on the operating stage. These values show that vibration and temperature depend on the working regime and not only on a specific fault, so a universal threshold is insufficient. Diagnostic methodologies with specialized equipment, such as the combination of MarPrime and Vibrotip applied to a marine engine (Oropesa-Márquez & Cecilia-Simón, 2018), confirm the value of vibration diagnosis but require instrumentation beyond the reach of an SME.

Normative Severity Criteria

The ISO 10816-1 standard evaluates vibration severity through the RMS velocity measured on non-rotating parts and groups machines into classes according to their size and type of support (International Organization for Standardization [ISO], 1995). That edition was superseded by ISO 20816-1:2016 (ISO, 2016), whose criteria for industrial machines are developed in other parts of the series; the limits used here correspond to the 1995 edition and should be checked against the current standard. For motors of up to 15 kW (class I), the reference limits used in this work are 1.8 mm/s for warning and 4.5 mm/s for the critical state. Because the MPU6050 delivers acceleration and not velocity, the limits are converted to acceleration by assuming that the dominant component is the vibration at the rotation frequency; this assumption is reasonable for imbalance and

misalignment but less valid for bearing defects, which appear at other frequencies.

Strictly, the relation $a = 2\pi fv$ holds for a sinusoidal component of known frequency; it is not a general conversion between broadband acceleration RMS and velocity RMS. The thresholds obtained with it are therefore treated in this work as approximate experimental criteria and not as a classification equivalent to the standard. A formal evaluation requires computing the velocity by spectral integration over a defined frequency band and checking the measurement conditions of the applicable part of the ISO 20816 series.

TinyML and Anomaly Detection in Predictive Maintenance

Njor et al. (2024) reviewed the complete TinyML stack for predictive maintenance and concluded that model optimization has received much attention, whereas data optimization and the availability of specific datasets are scarce. To run models on microcontrollers, TensorFlow Lite Micro offers an interpreter designed for devices with kilobytes of memory (David et al., 2021). Arciniegas et al. (2025) deployed on an ESP32S3 a neural-network classifier on spectral (FFT) features with an MPU6050 sampled at 100 Hz, together with K-Means for anomalies, and reported 96.5% accuracy in the laboratory, an inference of about 25 ms and false positives from external vibrations. Carvalho de Lucena Filho and Valencia de Almeida (2026) evaluated post-training quantization on the MAFAULDA dataset for three platforms, including the ESP32-S3 and the ESP32-WROOM, and found that a 37 KB INT8 multilayer perceptron reached F1 = 0.98, whereas a 1D convolutional network fell from 0.94 to 0.75 when quantized. Finally, Ahmad et al. (2021) showed that an autoencoder trained only on normal signals detects anomalies through its reconstruction error without requiring fault data, which is relevant for workshops where real faults are scarce. Table 3 summarizes the background reviewed in this section.

Table 3 Background on Motor Monitoring With ESP32 and TinyML

Study	Hardware	Diagnostic method	Reported scope or performance
Pandit et al. (2025)	ESP32 and ADXL345	Vibration magnitude with fixed threshold; Telegram alert	No performance metrics in the abstract
Kolok et al. (2025)	ESP32-C6 and MPU6050	Isolation Forest on a laptop (RMS and FFT/Welch)	Accuracy close to 73%; effective sampling of 2 Hz
Arciniegas et al. (2025)	ESP32S3 and MPU6050 at 100 Hz	FFT, neural network (Edge Impulse) and K-Means	96.5% in the laboratory; inference of about 25 ms
Ahmad et al. (2021)	Not embedded (datasets)	LSTM autoencoder trained on normal data	Average F1 of 99.6% on two datasets
Carvalho de Lucena Filho & Valencia de Almeida (2026)	ESP32-S3, ESP32-WROOM and STM32F4 (memory limits)	MLP, 1D CNN, LSTM and Random Forest with INT8 quantization	37 KB INT8 MLP with F1 = 0.98; 1D CNN fell from 0.94 to 0.75

Note. Compiled from Pandit et al. (2025), Kolok et al. (2025), Arciniegas et al. (2025), Ahmad et al. (2021), and Carvalho de Lucena Filho and Valencia de Almeida (2026).

Identified Gap

Three gaps emerge from the review. First, low-cost ESP32 prototypes usually rely on single thresholds or external processing (Pandit et al., 2025; Kolok et al., 2025). Second, TinyML works report performance on laboratory benches, without standardized motor profiles that adapt the alarm criterion to each machine (Arciniegas et al., 2025). Third, data from real workshop motors are scarce (Njor et al., 2024). proposed system proposes to address these gaps by combining ISO 10816-1 and insulation profiles with an anomaly detector to be trained on original data and executed on the microcontroller, with the normative rules as a safeguard; at present only the profile-based part has been tested.

METHODOLOGY

Research Approach, Type and Design

The research adopts a quantitative, applied approach, descriptive in its first stage and experimental in the second. The design is cross-sectional: each motor is measured in a limited time window rather than through longitudinal follow-up. The work is developed in two stages. In stage 1, five workshop motors are characterized with the threshold-based prototype. In stage 2, a dataset is built on a lathe motor, and the TinyML detector is designed and its validation protocol defined; at the time of writing, the noise floor and a first running capture have been obtained, but a preliminary autoencoder has been trained on that single session and evaluated with synthetic anomalies.

Population and Sample

The reference population is low-power electric motors (up to 15 kW) used in small-scale manufacturing processes. The sample is non-probabilistic and purposive: five accessible motors in a machine shop, with powers from 0.33 to 9.5 HP (Table 4). For stage 2, the 1.5 HP Siemens motor of a lathe is selected for its accessibility and its stable operation. This decision does not seek statistical representativeness but to demonstrate the technical feasibility of the system on real motors (Kolok et al., 2025).

Instruments and Data Collection

Vibration is acquired with an MPU6050 MEMS accelerometer connected through an I²C bus. In stage 1 it is sampled at about 20 Hz with a 21 Hz digital filter, and the RMS vibration magnitude is computed over a moving window of one second, according to expression (1):

$$vib_{\mathrm{RMS}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{1}$$

where n is the number of samples in the window, x_i is each reading of the acceleration magnitude (m/s²) and $\bar{x}$ is the window mean, so that the static component of gravity is removed and only the vibrational energy is retained. Temperature is measured with a DS18B20 digital sensor fixed to the casing and sampled asynchronously every second. The vibration thresholds are obtained by converting the velocity limits of ISO 10816-1 to acceleration through expression (2):

$$a_{\mathrm{lim}} = \frac{v_{\mathrm{lim}} \cdot 2\pi \cdot n}{60\,000} \tag{2}$$

where a_{lim} is the acceleration threshold (m/s²), v_{lim} is the velocity limit of the ISO class (mm/s) and n is the rated speed of the motor (rpm). The temperature thresholds are assigned by insulation class (A, B, F or H) as indicative casing-temperature values, not winding values; casing temperature does not equal winding temperature, and these thresholds have not been justified experimentally. Because expression (2) is strictly valid for a sinusoidal component at the rotation frequency, the resulting vibration thresholds are approximate experimental criteria and not an ISO-equivalent classification.

With both variables the firmware classifies the state as Normal, Warning or Critical, and allows a motor profile to be chosen or manual thresholds to be entered from the web panel or from an Android application connected through Bluetooth Low Energy. The nameplate data (power, speed, voltage, current and insulation class) were recorded in a spreadsheet with the formulas of the preceding expressions.

Design of the TinyML Detector

The stage 2 detector requires a higher sampling frequency than stage 1, because at 20 Hz (Nyquist limit of 10 Hz) the rotation frequency of the measured motors, from 1,450 to 1,740 rpm (24 to 29 Hz), cannot be resolved; the lathe's Siemens motor, connected directly to the three-phase 60 Hz mains, has a nameplate speed of 1,740 rpm at that frequency (about 29 Hz). An acquisition at 500 Hz was therefore designed, with the MPU6050

low-pass digital filter set to 94 Hz (44 Hz as an alternative). The 260 Hz setting was discarded because its bandwidth exceeds the Nyquist limit of 250 Hz and lets through noise without information, as the noise-floor characterization shows (Results section). From each window of 256 samples (about 0.5 s), computed on each axis after subtracting its mean and not on the vector magnitude, because the RMS of the magnitude reflects only the vibration in the direction of gravity and depends on sensor orientation, 12 features are extracted: RMS value, peak value, crest factor, kurtosis and the energy in eight frequency bands from 0 to 250 Hz obtained through the FFT; the RMS value is obtained as the square root of the sum of the variances of the three axes (vector RMS). With the 94 Hz filter, the bands above about 94 Hz are attenuated, and the definitive set of 12 features will be fixed after the capture. With these inputs, a dense autoencoder with a 12-8-4-8-12 architecture is trained only on healthy-operation windows. The reconstruction error of each window is computed with expression (3), and an anomaly is declared when it exceeds the threshold of expression (4):

$$e = \frac{1}{d}\sum_{j=1}^{d}\left(z_j - \hat{z}_j\right)^2 \tag{3}$$

$$\tau = \mu_e + k \cdot \sigma_e \tag{4}$$

where d = 12 is the number of features, z_j is the normalized feature, $\hat{z}_j$ its reconstruction, μ_e and σ_e are the mean and standard deviation of the error on the healthy validation windows, and k is a sensitivity factor set to 3 as an initial value and adjusted with the validation data. The model is quantized to 8-bit integers and deployed with TensorFlow Lite Micro (David et al., 2021). The choice of a small dense network rests on the fact that quantized multilayer perceptrons kept their performance on microcontrollers, unlike convolutional networks (Carvalho de Lucena Filho & Valencia de Almeida, 2026). The detector output is combined with the ISO rules so that the latter keep priority: a reading that exceeds the critical threshold raises an alarm even if the model detects no anomaly.

Collection and Validation Procedure

The procedure is organized in five steps. First, the sensor noise floor is measured at 500 Hz, with the sensor on a table and with the motor off, at filter settings from 21 to 260 Hz, recording the sensor orientation. Second, healthy-operation sessions of the lathe motor are recorded on different days and at different spindle speeds, and each session is labeled; each session follows the sequence of motor on with the 94 Hz filter, with the 44 Hz filter and with the 21 Hz filter, and motor off at the end; the date, spindle speed, load and sensor orientation are recorded. Third, because the workshop did not authorize inducing abnormal conditions in the motor, the test anomalies are synthetic: components that simulate an imbalance (an increase of energy in the band of the rotation frequency) are injected into healthy windows and labeled as simulated; validation with real faults is left as future work. In the preliminary test reported here, performed because no further sessions were available, the 155 healthy windows of the single session were split chronologically (60/20/20%) and the synthetic anomalies were generated by scaling the feature bands of the test windows (Results section).

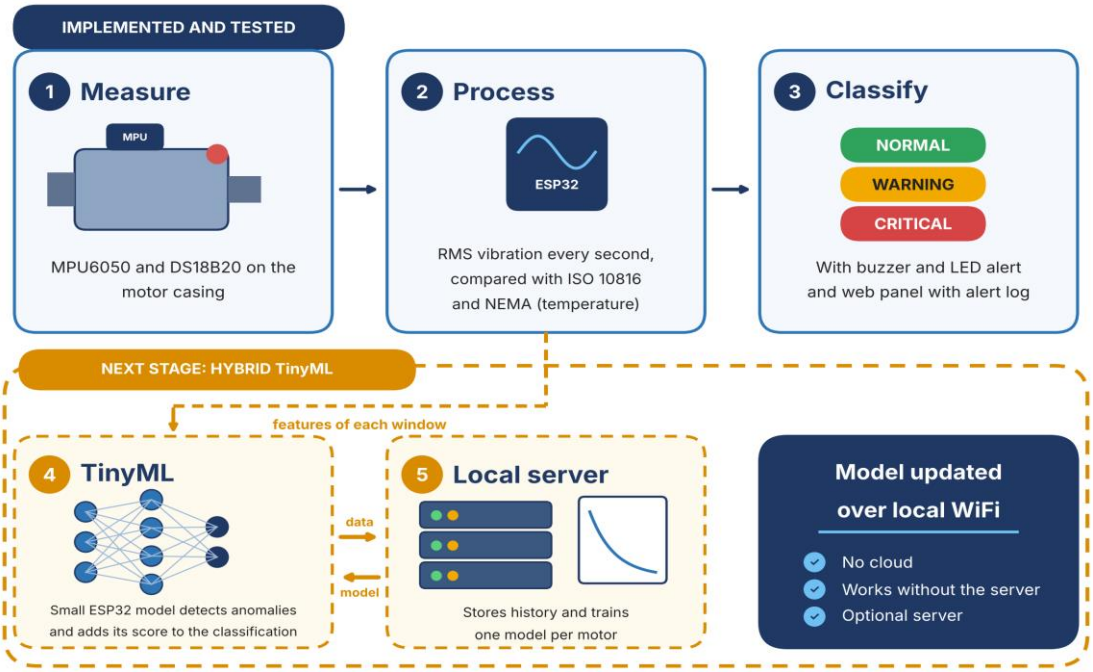
Fourth, the data are split by session and not by window, to prevent adjacent windows from appearing in both training and test: the healthy sessions are distributed among training, threshold validation and test, and the windows with synthetic anomalies are used only for testing. Fifth, three detectors are compared on the same test set: the ISO rules, a statistical baseline (z-score or Mahalanobis distance) and the autoencoder, in addition to the hybrid scheme. The metrics are sensitivity, specificity, F1, false alarms per hour, inference latency and use of ESP32 RAM and flash memory. Sensitivity, specificity and F1 are reported only on the synthetic anomalies, whereas false alarms per hour are measured on real healthy windows not used in training. As an initial acceptance criterion, a sensitivity of at least 0.90 with no more than one false alarm per hour is proposed, values that will be reviewed with the data obtained.

RESULTS AND DISCUSSION

Figure 2 summarizes the methodology of the proposed system. Blocks 1 to 3 (measure, process and classify) correspond to the prototype implemented and tested in stage 1, whose results are reported below. Blocks 4 and 5 correspond to the hybrid TinyML stage: a small model that runs on the ESP32 and adds its anomaly score to

the classification, and an optional local server that stores the history, trains a model for each motor and returns the updated model over the local WiFi network. The system works without the server and without the cloud, and the TinyML stage has not been completed at the time of writing.

Figure 2 Methodology of the Proposed System: Implemented Prototype (Stage 1) and Hybrid TinyML Stage



Note. Blocks 1–3 correspond to the prototype implemented and tested in stage 1; blocks 4–5 correspond to the hybrid TinyML stage, which had not been completed at the time of writing

Characterization of Five Workshop Motors

Table 4 presents the prototype measurements on the five motors. In the three motors with a known rated speed, the RMS vibration was 0.05 to 0.08 m/s², equivalent to 15.2% to 29.3% of the warning threshold calculated with expression (2), that is, between 71% and 85% below that threshold, and all three were preliminarily classified as Normal. This classification is not a validated diagnosis, because each motor was measured once at 20 Hz, a rate that does not resolve the rotation frequency. The equivalent velocity, obtained by dividing the acceleration by the angular rotation frequency, ranged from 0.27 to 0.53 mm/s, against the warning limit of 1.8 mm/s for class I. Casing temperatures were 19.4 to 30.6 °C. For the Siemens motor, of insulation class A, the 19.4 °C was far below the thresholds of 60 and 70 °C.

Table 4 Vibration and Temperature Measured in Five Workshop Motors

Motor	Speed (rpm)	Class	Vib. RMS (m/s²)	Threshold (m/s²)	% of threshold	Eq. velocity (mm/s)	T (°C)	Preliminary status
Industrial drill (3 HP)	1,700	I	0.06	0.320	18.7	0.34	21.9	Prelim. normal
Siemens lathe motor (1.5 HP)	1,740	I	0.05	0.328	15.2	0.27	19.4	Prelim. normal
JAFO FWF-32 milling machine (9.5 HP)	1,450	I	0.08	0.273	29.3	0.53	20.9	Prelim. normal
Truper drill (0.33 HP)	N/A	I	1.36	N/A	N/A	N/A	30.6	Unclassified
Bench drill (0.5 HP)	N/A	I	0.14	N/A	N/A	N/A	29.1	Unclassified

Note. Speed = rated speed; Threshold = ISO 10816-1 class I warning limit converted to acceleration; Eq.

velocity = equivalent velocity; N/A = not available for lack of a rated speed. Rated speeds were taken from the nameplate or catalog, without verification with a tachometer. The Siemens motor nameplate states 1,450 rpm at 50 Hz and 1,740 rpm at 60 Hz; the latter was used because of its direct connection to the 60 Hz mains.

The two portable drills could not be classified because the speed declared by their manufacturers corresponds to the spindle or the bit and not to the motor, so there is no motor rotation speed with which to apply expression (2). The Truper drill recorded 1.36 m/s², about 17 to 27 times the value of the three classified motors, and the other drill 0.14 m/s². Both also showed the highest temperatures of the group (30.6 and 29.1 °C).

Sensor Noise Floor at 500 Hz

For stage 2, the noise floor of the MPU6050 was characterized at 500 Hz (measured sampling frequency of 500.5 to 500.8 Hz, with a maximum deviation of 0.16%), with windows of 256 samples, a ±8 g range and the sensor at rest, in 14 sessions totaling 1,891 windows. As Table 5 shows, the RMS of the noise grows with the bandwidth of the digital filter: the power follows the relation $RMS^2 = N0 \cdot BW$ with $N0 = 7.98 \times 10^{-6}$ $(m/s^2)^2/Hz$ ($R^2 = 0.989$), and the free exponent of the RMS was 0.46 ± 0.02, close to the 0.5 expected for white noise. The coefficient of variation between windows was 0.29% on the table and 0.50% with the motor off, and the floor with the motor off was 0.84 to 0.85 times that measured on the table.

Table 5 MPU6050 Noise Floor at 500 Hz by Digital Filter Setting

Digital filter (Hz)	On the table (m/s² RMS)	Motor off (m/s² RMS)
260	0.0454	—
184	0.0377	—
94	0.0285	0.0241
44	0.0207	0.0176
21	0.0146	0.0124

Note. Data from 14 sessions (1,891 windows of 256 samples, ±8 g, fs ≈ 500.6 Hz). Values are in m/s² RMS; — = not measured. The 260 Hz setting exceeds the Nyquist limit (250 Hz).

The 94 Hz filter has a floor of 0.0285 m/s² on the table and of 0.0241 m/s² with the motor off, and with it the detection threshold $\mu + 3\sigma$ of the window RMS is about 0.028 m/s². The stage 1 readings (0.05 to 0.08 m/s² with the 21 Hz filter) exceed the floor of that filter by 3.4 to 6.5 times, so that noise accounts for between 2.4% and 8.5% of the measured power and the readings are not a sensor artifact.

The 94 Hz filter was adopted as the capture setting, with 44 Hz as an alternative if training shows that the high bands contribute no information. Each motor was measured only once at 20 Hz, so these margins remain indicative.

Siemens Motor Running at 500 Hz

On 21 October 2026 the Siemens motor was captured running for the first time, with the sensor fixed to its casing and the stage 2 capture program, with a direct connection to the three-phase mains (1,740 rpm on the nameplate at 60 Hz, about 29 Hz). The valid capture lasts 198 s with the motor on and is divided into three segments according to the digital filter (Table 6). With the 94 Hz filter, the RMS of the magnitude was 0.130 m/s² (vector RMS of 0.352 m/s²), 4.6 times the noise floor on the table; with the 44 and 21 Hz filters the margin was 3.8 and 2.8 times.

With the motor off and the 21 Hz filter, the RMS of the magnitude was 0.0105 m/s² (median of 1,075 windows), so the running motor exceeds it by 3.9 times, a value consistent with the range of 3.4 to 6.5 times of stage 1. The signal is stable: the standard deviation of the vector RMS within each segment was 6% to 9%.

Table 6 Siemens Motor Running: RMS Vibration by Digital Filter Setting (Session of 21 October 2026)

Filter (Hz)	Windows	Duration (s)	RMS magnitude (m/s²)	Vector RMS (m/s²)	Margin over floor
94	155	79	0.130	0.352	4.6×
44	96	49	0.078	0.225	3.8×
21	133	68	0.0405	0.117	2.8×
21 (motor off)	1,075	562	0.0105[a]	0.0208[a]	—

Note. [a] Median. Windows of 256 samples at 500 Hz, ±8 g. Margin = RMS of the magnitude divided by the table floor in Table 5. The first window, the span with intermittent starts, and the windows with fs below 499 Hz were excluded.

The energy is concentrated between 65 and 130 Hz and not at the rotation frequency. With the 94 Hz filter, the 65–94 Hz and 96–129 Hz bands contribute 0.19 and 0.25 m/s² RMS, whereas the 14–34 Hz band, which contains the rotation frequency, contributes 0.035 m/s², about 11% of the warning threshold of 0.328 m/s² calculated with expression (2) for 1,740 rpm.

The dominant spectral peak fell between 100 and 120 Hz, probably electromagnetic vibration at twice the 60 Hz line frequency, although this attribution was not verified. If the bands are converted to velocity (acceleration divided by 2πf of each band), the estimated RMS velocity is of the order of 0.6 mm/s, about 35% of the 1.8 mm/s warning limit; this is a preliminary estimate, limited to frequencies below about 130 Hz. The orientation of the sensor, with per-axis means of −7.77, −5.17 and −1.85 m/s², distributes gravity among the three axes, which is why the vector RMS is used.

Implementation Status of the System

At the close of this progress report, the following are operational: the firmware with motor profiles and manual thresholds, persistence of the configuration in flash memory, the self-hosted web panel with an alert log, and an Android application that shows vibration, temperature, state and probable cause in real time over Bluetooth Low Energy.

During link tests, disconnections were observed when WiFi and Bluetooth operated simultaneously, since both share the same ESP32 radio, although their frequency and the resulting data loss have not been quantified; the mitigation under way consists of tuning the connection parameters and, if necessary, disabling the access point while a Bluetooth client is connected.

Acquisition and Processing Code

The three programs used in this work are summarized in Figures 3 to 5, in the order in which they were used; they are presented in simplified form, without the web panel, the Bluetooth link, flash persistence or the output format, which do not take part in the reported measurements.

Figure 3 corresponds to the stage 1 prototype. It samples the MPU6050 at 20 Hz and computes the RMS of the magnitude over a moving window of 20 samples (expression 1); the casing temperature is read from the DS18B20 asynchronously every second.

The vibration thresholds are obtained from the ISO 10816-1 class through expression (2) and the temperature thresholds from the insulation class, and with both the state is classified as Normal, Warning or Critical. The magnitude-based RMS of this program depends on the orientation of the sensor and is not equivalent to the per-axis vector RMS used in stage 2 (Figure 5); the two metrics are therefore not directly comparable.

```
#define VENTANA 20                    // 20 samples ≈ 1 s at 20 Hz
const unsigned long SAMPLE_MS = 50;      // 20 Hz (vibration)
const unsigned long TEMP_MS   = 1000;    // DS18B20, asynchronous read
 // Reference limits by ISO 10816-1 class and NEMA insulation class
const float ISO_ADV_MMS[4]  = {1.8, 2.8, 4.5, 7.1};
const float ISO_CRIT_MMS[4] = {4.5, 7.1, 11.2, 18.0};
const float TEMP_ADV_C[4]   = {60, 70, 85, 100};
const float TEMP_CRIT_C[4]  = {70, 85, 100, 115};
 // Magnitude of the acceleration vector (includes gravity)
float magnitud() {
  mpu.getEvent(&a, &g, &t);
  return sqrt(a.acceleration.x*a.acceleration.x +
              a.acceleration.y*a.acceleration.y +
              a.acceleration.z*a.acceleration.z);}
// Expression (1): window RMS about the mean (removes gravity)
float vibRms() {
  float media = promedio(buf, VENTANA);
  float s2 = 0;
  for (int i = 0; i < VENTANA; i++) s2 += sq(buf[i] - media);
  return sqrt(s2 / VENTANA);}
// Expression (2): a_lim = v_lim * 2*pi * n/60   (v in m/s)
void aplicarPerfil(int rpm, int claseISO, int claseAisl) {
  float omega = 2 * PI * rpm / 60.0;
  umbVibAdv  = ISO_ADV_MMS[claseISO]  / 1000.0 * omega;
  umbVibCrit = ISO_CRIT_MMS[claseISO] / 1000.0 * omega;
  umbTempAdv  = TEMP_ADV_C[claseAisl];
  umbTempCrit = TEMP_CRIT_C[claseAisl];}

// Classification (critical rules take priority)
String estado(float vib, float temp, float vibPrev) {
  if (vib > umbVibCrit || temp > umbTempCrit)  return "CRITICO";
  if ((vib - vibPrev) > 1.5 * baseline)        return "ADVERTENCIA"; // sudden peak
  if (vib > umbVibAdv  || temp > umbTempAdv)   return "ADVERTENCIA";
  return "NORMAL";}
 void loop() {
  if (millis() - tVib >= SAMPLE_MS) {          // vibration, 20 Hz
    tVib = millis();
    guardaEnVentana(magnitud());
    vibPrev = vib;  vib = vibRms();
    edo = estado(vib, tempC, vibPrev);
    digitalWrite(LED,    edo != "NORMAL");
    digitalWrite(BUZZER, edo == "CRITICO");  }
  if (millis() - tTemp >= TEMP_MS) {           // temperature, 1 Hz
    tTemp = millis();
    tempC = ds18b20.getTempCByIndex(0);        // conversion already done
    ds18b20.requestTemperatures();             // starts the next one  }}
```

Figure 3 Stage 1 Firmware: 20 Hz Measurement, Profile-Based Thresholds, and Classification (Monitor_Etapa1.ino)

Note. Simplified version of the Monitor_Etapa1.ino firmware.

Figure 4 was used to obtain the noise floor of Table 5. It samples at 500 Hz through direct I²C, with ±8 g and an online-selectable digital filter (settings from 260 to 5 Hz). Every 256 samples it computes the RMS of the

magnitude about its mean, the standard deviation of each axis, the peak, the crest factor, the kurtosis and the actual sampling frequency, which serves to verify the 500 Hz.

Figure 4 Noise-Floor Capture Firmware at 500 Hz With the Motor Off (Reposo_Captura.ino)

```
#define WIN       256                  // samples per window (0.512 s)
#define FS_TARGET  500                   // Hz
#define PERIODO_US (1000000UL / FS_TARGET)
#define LSB_G      4096.0f               // ±8 g
 uint8_t dlpf = 4;                     // 0=260 1=184 2=94 3=44 4=21 5=10 6=5 Hz
 void mpuConfigura() {
  mpuEscribe(0x6B, 0x00);               // wake the sensor
  mpuEscribe(0x1C, 0x10);               // accelerometer ±8 g
  mpuEscribe(0x1A, dlpf);               // digital low-pass filter
  mpuEscribe(0x19, 0x00);               // minimum sample-rate divider
}
 void cierraVentana(uint32_t tAhora) {
  for (int i = 0; i < WIN; i++)          // vector magnitude
    mag[i] = sqrt(ax[i]*ax[i] + ay[i]*ay[i] + az[i]*az[i]);
  mm   = promedio(mag, WIN);
  rms  = sqrt(sumaCuadrados(mag, mm) / WIN);       // static component removed
  pico = max_i(abs(mag[i] - mm));
  cresta = pico / rms;
  curt   = momento4(mag, mm) / pow(rms, 4);
  fs_real = (WIN - 1) * 1e6f / (tAhora - t0Ventana); // checks the 500 Hz
  imprime(idx++, fs_real, dlpf, rms,
          desv(ax), desv(ay), desv(az), pico, cresta, curt);
}
 void loop() {
  while (Serial.available()) {           // a digit 0-6 changes the filter
    c = Serial.read();
    if (c >= '0' && c <= '6') { dlpf = c - '0'; mpuConfigura(); n = 0; }
  }
  if ((int32_t)(micros() - proximo) >= 0) {        // 500 Hz sampling
    proximo += PERIODO_US;
    leeAccel(&ax[n], &ay[n], &az[n]);
    if (++n == WIN) { cierraVentana(micros()); n = 0; }
  }
}
```

Note. Simplified version of the Reposo_Captura.ino firmware.

Figure 5 corresponds to the stage 2 capture. Acquisition runs on core 1 of the ESP32 with timing by micros() and the computation on core 0, so that the FFT does not delay the sampling. For each window the mean of each axis is subtracted, the vector RMS is computed, a Hann window and a radix-2 FFT are applied, and the energy in eight bands from 4 to 248 Hz and the spectral peak are obtained. These quantities are candidate features; the definitive set of 12 will be fixed after the capture.

Figure 5 Stage 2 Capture Firmware: 500 Hz Acquisition and FFT-Based Feature Extraction (Captura_Motor.ino)

```
#define FS      500      // Hz
#define WIN     256      // samples per window (0.512 s)
#define NB      8        // frequency bands
#define LSB_G   4096.0f  // sensitivity for ±8 g
 // ---- Core 1: 500 Hz acquisition ----
void loop() {
  if ((int32_t)(micros() - proximo) >= 0) {
    proximo += 1000000UL / FS;
    leeAccel(&ax[n], &ay[n], &az[n]);          // I2C 400 kHz, m/s²
    if (++n == WIN) {
      n = 0;
      fs_real = (WIN - 1) * 1e6f / (micros() - t_primera);
      copiaVentana();                          // copy to a buffer
      notificaNucleo0();                       // triggers the computation
    }
  }
}
 // ---- Core 0: per-window features ----
void calculaVentana(float *eje[3]) {
  for (int e = 0; e < 3; e++) {
    media[e] = promedio(eje[e], WIN);
    for (int i = 0; i < WIN; i++)
      re[i] = (eje[e][i] - media[e]) * hann[i];    // static component removed
    sd[e] = desvEst(eje[e], WIN);
    fft(re, im);                                   // radix-2
    for (int k = 1; k < WIN/2; k++)
      pot[k] += 2*(re[k]*re[k] + im[k]*im[k]) / (WIN * sum_hann2);
  }
  rms_vect = sqrtf(sd[0]*sd[0] + sd[1]*sd[1] + sd[2]*sd[2]);  // independent of sensor orientation
  for (int b = 0; b < NB; b++)                     // 4-248 Hz, df = FS/WIN = 1.95 Hz
    banda[b] = sqrtf(sumaPot(pot, K[b], K[b+1]));
  k_pico = argmax(pot, 3, WIN/2);                  // peak between 6 and 248 Hz
  f_pico = (k_pico + interpParabolica(pot, k_pico)) * FS / WIN;
  envia(fs_real, rms_vect, f_pico, banda);         // one CSV line per window
}
```

Note. Simplified version of the Captura_Motor.ino firmware.

TinyML Stage

Because no further sessions were available, a preliminary training was carried out with the single capture of the running Siemens motor (94 Hz filter, 155 valid windows, 12 features). The windows were ordered in time and split chronologically into 93 for training, 31 for validation (error threshold $\mu_e + 3\sigma_e$) and 31 for testing. Since no fault could be induced, anomalies were synthesized on the test windows by scaling the energy of the 14–34 Hz band (imbalance, factors 1.25 to 4) or of the 65–170 Hz bands (broadband increase, factors 1.1 to 2), updating the vector RMS and peak value accordingly. The autoencoder was trained with 10 random seeds and quantized to INT8; its inference was run with the quantized interpreter. The model occupies about 3.9 KB and requires 256 multiply–accumulate operations per inference. Table 7 gives the results, and Table 8 breaks the sensitivity down by synthetic condition.

The autoencoder reached a mean sensitivity of 0.69 (SD = 0.05), a specificity of 0.95 (SD = 0.05) and an F1 of 0.79 (SD = 0.03), with 1.6 false alarms on average among the 31 healthy test windows. A Mahalanobis-distance baseline obtained a sensitivity of 0.63, a specificity of 1.00 and an F1 of 0.78, so the autoencoder does not clearly outperform a simple statistical detector. The ISO-based rule, with the approximate threshold of 0.328 m/s², did not flag any synthetic window, because the vibration of the motor lies far below it even after scaling; therefore, in this test the hybrid scheme, in which the rule has priority, behaves identically to the autoencoder. Detection depends on the magnitude of the anomaly: it is practically complete for imbalance scaled by 4 and broadband increases of 1.5 or more, but low for the smallest factors (Table 8).

These results must be read with caution. They derive from one session; adjacent windows are similar, so the chronological split is optimistic compared with a split by session; the anomalies are synthetic and do not reproduce a real fault; and 31 healthy windows cannot be used to estimate false alarms per hour (even with zero false alarms, the 95% upper bound by the rule of three would be about 10% of the windows). Latency and memory use were measured on the ESP32 only for this preliminary model (Table 12), not for a model trained on several sessions.

Table 7 Preliminary Performance of the Detectors on Synthetic Anomalies

Detector	Sensitivity	Specificity	F1	False alarms/h	Latency (ms)	Flash / RAM (KB)
ISO 10816-1 rule	0.00	1.00	0.00	Not estimable	< 1 (rule)	—
Mahalanobis baseline	0.63	1.00	0.78	Not estimable	Not measured	—
INT8 autoencoder	0.69 (0.05)	0.95 (0.05)	0.79 (0.03)	Not estimable	Not measured	3.9 / not measured
Hybrid scheme	0.69 (0.05)	0.95 (0.05)	0.79 (0.03)	Not estimable	Not measured	3.9 / not measured

Note. Chronological split of 93/31/31 windows from a single session; anomalies are synthetic. Values for the autoencoder are the mean (standard deviation) over 10 seeds. The latency of the ISO rule is an estimate, not a measurement. Flash corresponds to the size of the quantized model.

Table 8 Sensitivity of the INT8 Autoencoder by Synthetic Condition

Condition	Factor	Autoencoder	Mahalanobis
Imbalance (14–34 Hz)	×1.25	0.16	0.13
Imbalance (14–34 Hz)	×1.5	0.44	0.23
Imbalance (14–34 Hz)	×2	0.91	0.84
Imbalance (14–34 Hz)	×4	1.00	1.00
Broadband (65–170 Hz)	×1.1	0.24	0.16

Broadband (65–170 Hz)	×1.25	0.80	0.71
Broadband (65–170 Hz)	×1.5	1.00	1.00
Broadband (65–170 Hz)	×2	1.00	1.00

Note. Proportion of the 31 synthetic test windows flagged; the autoencoder value is the mean over 10 seeds. The ISO rule flagged none.

Estimated Prototype Cost

Table 9 lists an estimated bill of materials for one monitoring node, kept to the minimum needed for the stage 2 configuration. Prices are approximate retail values in Mexican pesos (MXN) from common electronics vendors, without shipping or taxes, and were estimated by the authors rather than taken from invoices; they should be verified before being cited as a cost of the prototype. At roughly 19 MXN per US dollar, the total is about 25 USD, which is within the range of low-cost devices targeted for small enterprises. The cost excludes labor, the optional local server and the Android application.

Table 9 Estimated Bill of Materials per Monitoring Node

Component	Qty	Unit price (MXN)	Subtotal (MXN)
ESP32 development board	1	150	150
MPU6050 accelerometer module	1	60	60
DS18B20 temperature probe (waterproof)	1	60	60
4.7 kΩ resistor	1	2	2
Prototyping board and jumper wires	1	50	50
5 V USB power supply and cable	1	80	80
Plastic enclosure	1	70	70
Total			472

Note. Approximate values estimated by the authors; not based on invoices. Excludes shipping, taxes and labor.

Measurement Uncertainty and Verification Plan

Table 10 summarizes the manufacturer specifications of the two sensors and the measured quantities that bound the uncertainty of the readings reported in this article. The accelerometer values come from the MPU6050 datasheet and were not verified in this work, so they are presented as the expected performance and not as a calibration. The accelerometer gain tolerance (±3%) is the dominant term for the RMS values: treating it as a rectangular distribution, it contributes a relative standard uncertainty of about 1.7%, or about ±0.002 m/s² on the 0.130 m/s² RMS of the running Siemens motor. The quantization step (0.244 mg, 0.0024 m/s²) adds a noise of about 0.0007 m/s² RMS, negligible next to the measured noise floor of 0.0285 m/s². The zero-g offset (up to ±50 mg on X and Y and ±80 mg on Z) does not enter the RMS values, because the mean of each window is removed before computing them. The estimated velocity is more uncertain than the acceleration itself, since it relies on the approximation $a = 2\pi fv$ for a component at a single frequency (Equation 2), and it should be read as an order of magnitude.

Table 10 Sensor Specifications and Uncertainty Contributions

Parameter	Value	Source	Effect on results
Accelerometer range and sensitivity (MPU6050)	±8 g; 4096 LSB/g	Datasheet	Scale of all RMS values
Sensitivity tolerance	±3%	Datasheet	≈1.7% standard uncertainty on RMS
Zero-g offset	±50 mg (X, Y); ±80 mg (Z)	Datasheet	Removed by mean subtraction
Resolution (quantization step)	0.244 mg (0.0024 m/s²)	Datasheet	≈0.0007 m/s² RMS noise
Noise floor, 94 Hz filter	0.0285 m/s² RMS	Measured (Table 5)	Lower bound of detectable RMS

Sampling frequency (stage 2)	500.5 to 500.8 Hz	Measured	Frequency axis error below 0.2%
Temperature accuracy (DS18B20)	±0.5 °C (−10 to +85 °C)	Datasheet	Absolute error of the temperature reading
Temperature resolution	0.0625 °C (12 bits)	Datasheet	Negligible for 5 °C thresholds

Note. Datasheet values are the manufacturers' typical or maximum specifications and were not verified by the authors. RMS = root mean square. The 1.7% value is 3% divided by √3 (rectangular distribution).

Two verification procedures complete this characterization and are provided as open firmware with the project. The first is a static six-position calibration of the accelerometer, in which each axis is placed alternately facing up and down so that gravity is known: the offset is the mean of the two readings, the sensitivity is half their difference, and the sensitivity error is the deviation of that value from 1 g. The second is an on-device benchmark of the INT8 autoencoder on the ESP32, which measures the inference time per window with the microsecond timer over repeated runs, the program size and free memory, and checks the integer implementation against reference outputs of the TensorFlow Lite interpreter. Their results are reported in Tables 11 and 12.

Table 11 reports the six-position static calibration carried out on the accelerometer used in stage 2, and Table 12 reports the on-device benchmark of the INT8 autoencoder. The gain errors are between −0.5% and +0.2%, well inside the ±3% tolerance of the datasheet, and the offsets of the X and Z axes are close to the datasheet limits (±50 mg and ±80 mg, respectively). These offsets do not affect the RMS values, because the mean of each window is removed. The method has its own limits: the sensor was held against a table without a level, so a tilt of about 5° in some positions adds an uncertainty of roughly 0.4% to the gain, and the result comes from a single run. It is therefore a bench verification and not a traceable calibration.

Table 11 Six-Position Static Calibration of the MPU6050 (±8 g)

Axis	Offset (mg)	Sensitivity (g/g)	Gain error (%)
X	+52.4	1.0018	+0.18
Y	−8.5	0.9948	−0.52
Z	−50.9	0.9997	−0.03

Note. Single run, static positions with each axis up and down, about 2,000 samples per position at 500 Hz, 94 Hz filter, ±8 g. Offset = mean of the two readings; sensitivity = half their difference; gain error = sensitivity − 1. Datasheet limits: gain ±3%, offset ±50 mg (X, Y) and ±80 mg (Z). Tilt of the sensor adds about 0.4% of uncertainty.

The autoencoder runs on the ESP32 in about 36 µs per inference and in about 49 µs for the complete detector step (normalization, quantization, inference, reconstruction error and threshold), measured over 20,000 repetitions at 240 MHz. A window lasts 0.512 s, so the detector uses about 0.01% of the available time, which leaves the processor free for acquisition and feature extraction; those two stages were not included in the measurement. The integer implementation agrees with the reference TensorFlow Lite interpreter within 1 least significant bit on 16 test vectors, and the free heap and stack did not change during the benchmark. The model constants occupy 512 bytes of flash, while the whole sketch, which includes the Wi-Fi and Bluetooth libraries, uses 273,860 bytes (20% of the program storage) and 26,116 bytes of global variables (7% of dynamic memory). These figures verify that the detector fits comfortably within the microcontroller; they do not improve the detection performance, which remains preliminary.

Table 12 On-Device Benchmark of the INT8 Autoencoder (12-8-4-8-12)

Metric	Value
Microcontroller	ESP32-D0WD-V3, 240 MHz
Inference time, mean of 20,000 runs	35.5 µs (minimum 35, median 36, maximum 41 µs)
Complete detector step, mean of 20,000 runs	49.0 µs
Share of a 0.512 s window	About 0.01%

Difference from the TensorFlow Lite reference	Maximum 1 LSB; 4 of 192 outputs differ
Free heap before and after	327,136 bytes (no change)
Model constants in flash	512 bytes (TensorFlow Lite file: about 3.9 KB)
Whole sketch (includes Wi-Fi and Bluetooth libraries)	273,860 bytes of program (20%); 26,116 bytes of globals (7%)

Note. Benchmark sketch with 16 test vectors of the preliminary model. The time does not include the I²C acquisition or the fast Fourier transform used to extract the features. LSB = least significant bit.

Regarding industrial practice, ISO 20816-1 evaluates severity with broadband RMS velocity measured with calibrated instruments over a band that extends up to 1000 Hz, whereas this prototype has a 94 Hz filter, estimates velocity indirectly and has not been compared with a reference instrument. The system is therefore proposed as a screening and trend-recording tool for workshops that have no instrumentation, and not as a substitute for a certified vibration meter. A side-by-side comparison with a reference accelerometer or vibration meter on the same motor is left as future work.

DISCUSSION

The three classified motors operated with a wide margin with respect to their warning threshold, which is consistent with motors in normal operation and suggests, without validating it, that the approximate ISO-based thresholds are plausible. The equivalent velocity obtained (0.27 to 0.53 mm/s) is lower than the range of 1.46 to 4.44 mm/s that Auqui Fajardo and Farez Chasi (2024) report for a healthy motor without load; the difference should not be read as a better condition of the motors: these are different machines, with different mounting and measurement point, and the equivalent was computed under the assumption that vibration at the rotation frequency dominates.

The estimated cost of about 472 MXN per node (Table 9) supports the low-cost premise, although it is an author estimate and does not include installation or maintenance. The main limitation of stage 1 is the sampling at 20 Hz with a 21 Hz filter, which does not resolve the rotation frequency of motors of 24 to 29 Hz. The measured RMS value therefore represents low-frequency energy and not a vibration velocity in the strict sense, and the margins in Table 4 are indicative; nevertheless, the noise floor of the 21 Hz filter (0.012 to 0.015 m/s^2) is 3.4 to 6.5 times lower than the readings, so the margins are not explained by sensor noise. In addition, the stage 1 RMS was computed on the vector magnitude, which depends on sensor orientation, whereas stage 2 uses per-axis and vector RMS. This limitation motivated the 500 Hz acquisition of stage 2, and the threshold scheme will have to be recalibrated with that signal. The first step of future work is therefore to repeat the measurements of the five motors at 500 Hz, in several independent sessions, verifying the real sampling frequency and the rotation speed, and reporting confidence intervals and false-alarm rates. Other limitations are that the sample is five motors, that the rated speeds were not verified with a tachometer, and that each motor was measured only once, without repetitions that would allow estimating the variability of the measurement. The Truper drill value is an order of magnitude higher than the rest; one hypothesis to be verified is that it is a universal motor with brushes, whose operation is more vibrant, but without the rotation speed it is not possible to conclude this.

The capture of the running Siemens motor yields three readings for stage 2. First, the signal is distinguishable from noise at all three filter settings (margin of 2.8 to 4.6 times over the table floor) and stable within each segment. Second, the energy is concentrated between 65 and 130 Hz and not at the rotation frequency, so the acceleration threshold of expression (2), valid for the component at the rotation frequency, is not comparable with broadband acceleration, and to evaluate severity it is advisable to compute velocity by bands. Third, the results come from a single session of about three minutes, with the sensor mounted once and the nameplate speed not verified with a tachometer, and the dominant spectral peak does not correspond to the rotation frequency, so it cannot serve to confirm it; moreover, the velocity estimate is limited to frequencies below about 130 Hz.

Regarding TinyML, the background shows that the approach is feasible on ESP32-family microcontrollers (Arciniegas et al., 2025; Carvalho de Lucena Filho & Valencia de Almeida, 2026), although their results come from laboratory conditions and from other datasets, so they are not assumed for this work. It should also be considered that Kolok et al. (2025) obtained an accuracy close to 73% with a 2 Hz sampling rate, which supports the decision to increase the sampling frequency. The preliminary test shows that a 3.9 KB quantized autoencoder can separate scaled synthetic anomalies from healthy windows of the same session, but it does not outperform a Mahalanobis baseline and misses small anomalies; the fact that the approximate ISO rule detects none is consistent with the low overall vibration level and with its calibration for the rotation frequency. The main risk of stage 2 is the scarcity of real anomalies: since inducing anomalies in the workshop motor was not authorized, the test with anomalies is synthetic and does not replace validation with real faults, and training only on healthy operation mitigates the lack of fault examples but may produce false alarms under legitimate regime changes, such as a spindle speed not seen in training.

CONCLUSION

This study has three parts that should be kept distinct. What was proposed is a hybrid scheme in which a quantized autoencoder, trained on healthy operation of a lathe motor, complements the normative rules and runs on the ESP32. What was implemented is the stage 1 prototype (acquisition at 20 Hz, profile-based thresholds, web panel and Bluetooth Low Energy application) and the 500 Hz acquisition firmware. What was demonstrated experimentally is limited to the following: five workshop motors measured once each, of which three with a known rated speed fell 71% to 85% below the approximate warning threshold, a preliminary classification that is not a validated diagnosis; a sensor noise floor that follows $RMS^2 = N0 \cdot BW$ and that the stage 1 readings exceed by 3.4 to 6.5 times; a first capture of the running Siemens motor, whose energy is concentrated between 65 and 130 Hz and not at the rotation frequency; and a preliminary proof of concept in which an INT8 autoencoder trained on that single session detected synthetic anomalies with a sensitivity of 0.69 and a specificity of 0.95, figures that are optimistic and were not measured on the microcontroller.

Two consequences follow. First, the conversion of ISO 10816-1 velocity limits to acceleration through the rotation frequency is valid only for a sinusoidal component at that frequency, so the severity of the motor should be evaluated by velocity computed over frequency bands before the thresholds can be considered equivalent to the standard. Second, the TinyML result is preliminary: it rests on one session, a chronological split and synthetic anomalies, with false alarms per hour still unmeasured, although latency and memory use were measured on the ESP32 (Table 12), so it supports feasibility but not performance claims. Likewise, the calibration in Table 11 is a single bench run and no comparison with a reference instrument has been made.

As immediate work, the measurements of the five motors will be repeated at 500 Hz in independent sessions with the rotation speed verified by tachometer, the velocity will be computed by spectral integration, the bill of materials and connectivity stability will be documented, the autoencoder will be retrained with several independent sessions, deployed on the ESP32 to repeat the latency and memory measurements, and evaluated with synthetic anomalies, and the validation will be completed with real faults when the workshop permits.

Data Availability

The data are not publicly available because they were collected in a university workshop and are part of an ongoing project; they may be shared by the corresponding author on reasonable request

REFERENCES

1. Ahmad, S., Styp-Rekowski, K., Nedelkoski, S., & Kao, O. (2021). Autoencoder-based condition monitoring and anomaly detection method for rotating machines [Preprint]. arXiv. https://arxiv.org/abs/2101.11539
2. Arciniegas, S., Rivero, D., Piñan, J., Diaz, E., & Rivas, F. (2025). IoT device for detecting abnormal vibrations in motors using TinyML. Discover Internet of Things, 5, Article 41. https://doi.org/10.1007/s43926-025-00142-4

3. Auqui Fajardo, C. H., & Farez Chasi, C. O. (2024). Diseño e implementación de un sistema de monitoreo IoT de temperatura y vibración para protección de motores eléctricos [Design and implementation of an IoT temperature and vibration monitoring system for electric motor protection] [Undergraduate thesis, Universidad Politécnica Salesiana]. Repositorio Institucional UPS. https://dspace.ups.edu.ec/bitstream/123456789/29087/1/UPS-GT005779.pdf
4. Brito, S. D., Azinheira, G. J., Semião, J. F., Sousa, N. M., & Pérez Litrán, S. (2025). Non-intrusive low-cost IoT-based hardware system for sustainable predictive maintenance of industrial pump systems. Electronics, 14(14), 2913. https://doi.org/10.3390/electronics14142913
5. Carvalho de Lucena Filho, C. A., & Valencia de Almeida, F. (2026). TinyML for fault detection in rotating machinery: A post-training quantization assessment on the MAFAULDA dataset. In Anais do 20º Brazilian e-Science Workshop (pp. 33–40). Sociedade Brasileira de Computação. https://doi.org/10.5753/bresci.2026.249387
6. Coleman, C., Damodaran, S., Chandramouli, M., & Deuel, E. (2017). Making maintenance smarter: Predictive maintenance and the digital supply network. Deloitte Insights. https://www.deloitte.com/us/en/insights/industry/manufacturing-industrial-products/industry-4-0/using-predictive-technologies-for-asset-maintenance.html
7. David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Wang, T., Warden, P., & Rhodes, R. (2021). TensorFlow Lite Micro: Embedded machine learning for TinyML systems. Proceedings of Machine Learning and Systems, 3, 800–811. https://proceedings.mlsys.org/paper_files/paper/2021/hash/6c44dc73014d66ba49b28d483a8f8b0d-Abstract.html
8. Gomez Garcia, J. A. (2025). Sistema de adquisición y monitoreo en tiempo real para la detección temprana de fallas en motores mediante análisis de señales de vibración y temperatura bajo un ambiente (IoT) [Real-time acquisition and monitoring system for early fault detection in motors through vibration and temperature signal analysis in an IoT environment] [Degree thesis, Universidad Nacional Abierta y a Distancia]. Repositorio Institucional UNAD. https://repository.unad.edu.co/jspui/bitstream/10596/69921/4/jagomezgar.pdf
9. International Organization for Standardization. (1995). Mechanical vibration. Evaluation of machine vibration by measurements on non-rotating parts. Part 1: General guidelines (ISO Standard No. 10816-1:1995).
10. International Organization for Standardization. (2016). Mechanical vibration. Measurement and evaluation of machine vibration. Part 1: General guidelines (ISO Standard No. 20816-1:2016).
11. Janumala, T., Pavan, K. L., Ambareesh, K. V., Srirama, V., Khan, M. F., Keerthi, U. K., & Vedanth, S. (2026). Real-time industrial machine health monitoring and failure prediction using ESP32. International Journal of Engineering Research & Technology, 15(6). https://doi.org/10.5281/zenodo.20579053
12. Khan, N., Rafiq, F., Abedin, F., & Khan, F. U. (2019). IoT based health monitoring system for electrical motors. In 2019 15th International Conference on Emerging Technologies (ICET). IEEE. https://doi.org/10.1109/ICET48972.2019.8994398
13. Kolok, P., Hodoň, M., Ševčík, P., Hotz, L., & Remy, N. (2025). Low-cost IoT-based predictive maintenance using vibration. Sensors, 25(21), Article 6610. https://doi.org/10.3390/s25216610
14. Magadán, L., Suárez, F. J., Granda, J. C., & García, D. F. (2022). Low-cost industrial IoT system for wireless monitoring of electric motors condition. Mobile Networks and Applications, 28(1), 97–106. https://doi.org/10.1007/s11036-022-02017-2
15. Martínez Jiménez, N. (2022). Sistema IIoT para monitorización de motores eléctricos y detección de anomalías mediante el análisis de vibraciones en rodamientos [IIoT system for monitoring electric motors and detecting anomalies through bearing vibration analysis] [Master's thesis, Universidad de Oviedo]. Repositorio Institucional de la Universidad de Oviedo. https://digibuo.uniovi.es/dspace/bitstream/handle/10651/64200/TFM_NicolasMartinezJimenez.pdf
16. Maruthi, G. S., & Hegde, V. (2016). Application of MEMS accelerometer for detection and diagnosis of multiple faults in the roller element bearings of three phase induction motor. IEEE Sensors Journal, 16(1), 145–152.

17. Njor, E., Hasanpour, M. A., Madsen, J., & Fafoutis, X. (2024). A holistic review of the TinyML stack for predictive maintenance. IEEE Access, 12, 184861–184882. https://doi.org/10.1109/ACCESS.2024.3512860
18. OECD. (2024). Índice de Políticas para PyMEs: América Latina y el Caribe 2024 [SME Policy Index: Latin America and the Caribbean 2024] (Chapter 18: Mexico). https://www.oecd.org/es/publications/indice-de-politicas-para-pymes-america-latina-y-el-caribe-2024_807e9eaf-es/full-report/component-24.html
19. Oropesa-Márquez, Y., & Cecilia-Simón, N. (2018). Metodología combinada del uso de MarPrime y Vibrotip para diagnóstico del motor HYUNDAI HIMSEN 9H25/33 [Combined methodology using MarPrime and Vibrotip for diagnosing the HYUNDAI HIMSEN 9H25/33 engine]. Científica, 22(1), 71–79. https://www.redalyc.org/journal/614/61458000008/61458000008.pdf
20. Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., … Moher, D. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. BMJ, 372, Article n71. https://doi.org/10.1136/bmj.n71
21. Pandit, S. R., Patil, K., & Vaidya, V. (2025). Vibration-based motor health monitoring system using ESP32 and ADXL345. International Journal of Progressive Research in Engineering Management and Science, 5(4), 3550–3552. https://www.ijprems.com/ijprems-paper/vibration-based-motor-health-monitoring-system-using-esp32-and-adxl345