iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: https://doi.org/10.22214/ijraset.2026.85424

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

TinyML-Based Predictive Maintenance System for Low-Cost Industrial Equipment

Sudhanshu Shekhar, Sachin Kumar, Ashish Kumar, Priyanshu Kumar
Student, Department of Electronics Engineering, Government Polytechnic Sitamarhi India

Abstract: ***Small and medium enterprises run large numbers of low-cost motors, pumps, fans and compressors that are maintained only when they break down, because commercial condition-monitoring systems cost more than the machines themselves. This paper presents the design of a TinyML-based predictive maintenance system that runs the complete analysis on a microcontroller attached to each machine. An ESP32-S3 samples a three-axis ADXL345 accelerometer at 3200 Hz, a clamp-on current sensor and a casing temperature sensor, computes 113 time- and frequency-domain features every 0.32 s, and scores them with an 8-bit quantized autoencoder trained only on data from the healthy machine. The anomaly score is smoothed into a health index that drives green, amber and red alerts on the device, while only a 100-byte summary is sent over MQTT every 10 s; a velocity check against ISO 20816 zones and an optional fault classifier run in parallel. A review of 17 related studies shows that most TinyML fault detectors are trained and tested on laboratory datasets with labelled faults, which are rarely available for a specific low-cost machine. Design analysis shows that the autoencoder has 7,929 parameters and needs about 7,700 multiply–accumulate operations per window, that the signal buffers and model fit in under 30 KB of the ESP32-S3's 512 KB SRAM, and that on-device inference cuts the data sent from 1.66 GB to 0.86 MB per machine per day. A validation protocol using a motor test rig with seeded faults and public bearing datasets is provided.***

Index Terms - TinyML, predictive maintenance, anomaly detection, autoencoder, vibration analysis, ESP32-S3, edge computing, condition monitoring.

I. INTRODUCTION

Maintenance strategies range from run-to-failure, through fixed-interval preventive maintenance, to predictive maintenance (PdM), in which the condition of a machine is monitored and repairs are scheduled just before a failure is expected [1]. PdM reduces unplanned downtime and avoids replacing parts that still have useful life, but conventional implementations stream high-rate vibration data to a server or the cloud for analysis. The sensors, gateways, data charges and software licences then cost more than a small motor or pump, so such machines in small and medium enterprises are usually left to run until they fail.

Tiny machine learning (TinyML) runs trained neural networks on microcontrollers that cost a few dollars and consume milliwatts [2]. Frameworks such as TensorFlow Lite for Microcontrollers [3] and benchmarks such as MLPerf Tiny, which includes a machine-sound anomaly-detection task [4], make it practical to analyse vibration directly on the machine and send only the result. This removes the need to transmit raw data, keeps working when the network is down, and keeps sensitive production data on site.

Most published TinyML fault detectors, however, are supervised classifiers trained on laboratory datasets in which every fault type is labelled [11], [12], [14], [17]. A factory that wants to monitor its own fan or pump has no labelled fault data for that machine. This paper therefore designs a system that learns the healthy behaviour of each individual machine and flags departures from it. The contributions are:

1) A complete low-cost sensor-node design based on an ESP32-S3, a three-axis accelerometer, a clamp-on current sensor and a temperature sensor, with all analysis performed on the device.

2) An on-device pipeline that extracts 113 features from each 0.32 s window and scores them with an INT8 autoencoder trained only on healthy data, combined with an ISO 20816 velocity check and an optional fault classifier.

3) A health index with green, amber and red alert levels and a deployment lifecycle covering baseline collection, training, quantization, over-the-air update and retraining.

4) A review of 17 related studies, an analysis of model size, computation, memory and data volume, and a validation protocol using a motor test rig and public datasets.

Section II reviews related work. Sections III to V describe the architecture, hardware and on-device pipeline. Section VI sets out the experimental method, Section VII presents the results and discussion, and Section VIII concludes.

©IJRASET: All Rights are Reserved

II. LITERATURE REVIEW

The review covers 17 studies in four groups: surveys and enabling tools, datasets and baseline deep-learning methods, unsupervised anomaly detection on embedded devices, and TinyML fault diagnosis on microcontrollers. Table 1 summarises the studies that report a working model.

A. Surveys and enabling tools

Krupitzer et al. reviewed 150 PdM papers and identified affordable implementation for smaller companies, remote data handling and data fusion as open research gaps [1]. Ray reviewed the state of TinyML, its hardware, frameworks and applications, and its challenges such as limited memory and on-device training [2]. David et al. described TensorFlow Lite Micro, an interpreter-based framework that runs quantized models on microcontrollers with only a few kilobytes of runtime overhead [3]. Banbury et al. introduced MLPerf Tiny, which measures the accuracy, latency and energy of four tasks, including anomaly detection, on ultra-low-power hardware [4].

B. Datasets and baseline methods

Smith and Randall analysed the Case Western Reserve University (CWRU) bearing dataset with established signal-processing methods and showed which of its records contain clearly diagnosable faults, making it a benchmark for later work [5]. Purohit et al. released the MIMII dataset of normal and abnormal sounds from valves, pumps, fans and slide rails [6]. Ince et al. showed that a one-dimensional convolutional neural network (1-D CNN) can learn directly from raw motor current signals and detect faults in real time without hand-designed features [7].

C. Unsupervised anomaly detection at the edge

Givnan et al. trained autoencoders on normal operation of industrial motors and mapped reconstruction error to green, amber and red levels so that engineers could intervene before failure [8]. Ren et al. proposed TinyOL, which adds a trainable layer to a frozen network on an Arduino Nano 33 BLE Sense so that an autoencoder monitoring a fan can be fine-tuned with streaming data on the device [9]. Charan built a vibration autoencoder with a 7.5 KB footprint for an ARM Cortex-M4 with 256 KB SRAM, reaching accuracy and precision close to 80%, and noted the accuracy cost of post-training quantization [10].

D. TinyML fault diagnosis on microcontrollers

Liao deployed a lightweight CNN on an STM32H743 microcontroller and classified 10 CWRU bearing conditions with 98.9% accuracy, using about 142 KB of flash and 11 KB of RAM and 19 ms per diagnosis [11]. Yoo et al. reduced the input of a CWRU spectrogram CNN to 32×10 values and kept 99.86% accuracy with about 153,000 parameters [12]. Chung et al. instead built a 1-D CNN accelerator on a low-cost FPGA, with quaternary quantization that cut memory by 89%, reaching 96.37% accuracy and a 0.28 s response [13].

Al-Libawy deployed an INT8 1-D CNN on an Arduino Nano 33 BLE Sense for healthy, inner-race and outer-race bearing classes, reaching 98.64% test accuracy with 32 KB of flash and 20 ms latency [14]. Arciniegas et al. used a XIAO ESP32S3 Sense and an MPU6050 at 100 Hz with an 8-bit dense network on FFT features, reaching 96.5% laboratory accuracy and about 300 ms from acquisition to alert [15]. Ayadi et al. detected looseness and misalignment in DC motors with an MLP on an Arduino Nano 33 BLE Sense; validation accuracy was 97.1%, but on-device accuracy on unseen data fell to 73.19% [16]. Lucena Filho and de Almeida compared MLP, 1-D CNN, LSTM and Random Forest models on the MAFAULDA dataset and found that an INT8 MLP of 37 KB kept an F1 score of 0.98 and fitted ESP32 and STM32 devices, whereas the 1-D CNN lost accuracy under INT8 quantization [17].

Table 1 Summary of reviewed TinyML and embedded fault-detection studies

Ref.	Year	Hardware	Data / sensing	Model	Key result or limitation
[7]	2016	PC (real-time)	Motor current	1-D CNN	Learns from raw signal; no hand-made features
[8]	2022	Industrial motors	Machine sensors	Autoencoder	Green / amber / red alarms before failure
[9]	2021	Nano 33 BLE Sense	Fan vibration	AE + on-device learning	Fine-tunes last layer on streaming data
[10]	2023	Cortex-M4, 256 KB	Vehicle vibration	Autoencoder, 7.5 KB	~80% accuracy; PTQ costs accuracy

Ref.	Year	Hardware	Data / sensing	Model	Key result or limitation
[11]	2023	STM32H743	CWRU, 10 classes	Light CNN	98.9%; 142 KB flash, 11 KB RAM, 19 ms
[12]	2023	PC (lite model)	CWRU spectrograms	Small CNN, 153k params	99.86% with 32×10 input
[13]	2023	Low-cost FPGA	Bearing vibration	1-D CNN accelerator	96.37%; 0.28 s; 89% less memory
[14]	2025	Nano 33 BLE Sense	Bearing dataset, 3 classes	INT8 1-D CNN	98.64%; 32 KB flash; 20 ms
[15]	2025	XIAO ESP32S3	MPU6050 at 100 Hz	INT8 dense net + K-means	96.5% lab; ~300 ms to alert
[16]	2025	Nano 33 BLE Sense	IMU at 100 Hz, DC motor	MLP, 117 features	97.1% validation, 73.19% on device
[17]	2026	ESP32 / STM32 (fit)	MAFAULDA	MLP, CNN, LSTM, RF	INT8 MLP 37 KB, F1 0.98; CNN degrades

E. Research gap

Three gaps stand out. First, most microcontroller studies train supervised classifiers on public laboratory datasets with labelled faults [11], [12], [14], [17]; such labels do not exist for a particular low-cost machine in a factory. Second, accuracy measured on a validation set can fall sharply on the device with new data [16], so a deployed system needs a way to adapt to each machine and to be retrained. Third, several low-cost nodes sample at only about 100 Hz [15], [16], which covers imbalance and misalignment but not the higher-frequency content of bearing faults. The proposed system addresses these points with machine-specific unsupervised learning, a defined retraining lifecycle and 3200 Hz sampling (Table 2).

Table 2 Gaps in existing work and how the proposed system addresses them

Gap	Evidence in reviewed work	Approach in this paper
Need for labelled fault data	Supervised classifiers on CWRU, MAFAULDA [11], [12], [14], [17]	Autoencoder trained only on the healthy machine; classifier optional
Accuracy drop after deployment	97.1% validation vs 73.19% on device [16]	Per-machine baseline, threshold from healthy errors, retraining loop
Low sampling rate	IMU sampled at 100 Hz [15], [16]	ADXL345 at 3200 Hz, analysis band up to 1600 Hz
Single sensing modality	Vibration only in most nodes	Vibration + motor current + casing temperature
No link to maintenance practice	Accuracy reported, but not alarm levels	Health index, green/amber/red alerts, ISO 20816 velocity zones

III. PROPOSED SYSTEM ARCHITECTURE

The system has two layers (Fig. 1). A sensor node mounted on each machine acquires the signals, runs the complete feature extraction and inference pipeline, and shows the machine's state on a local LED and buzzer. It sends a short summary and any alert events over MQTT [25] to a plant or cloud layer, where a time-series store and dashboard give maintenance staff a view of all machines. The same layer collects baseline data, trains and quantizes the models, and sends model updates back to the nodes over the air.

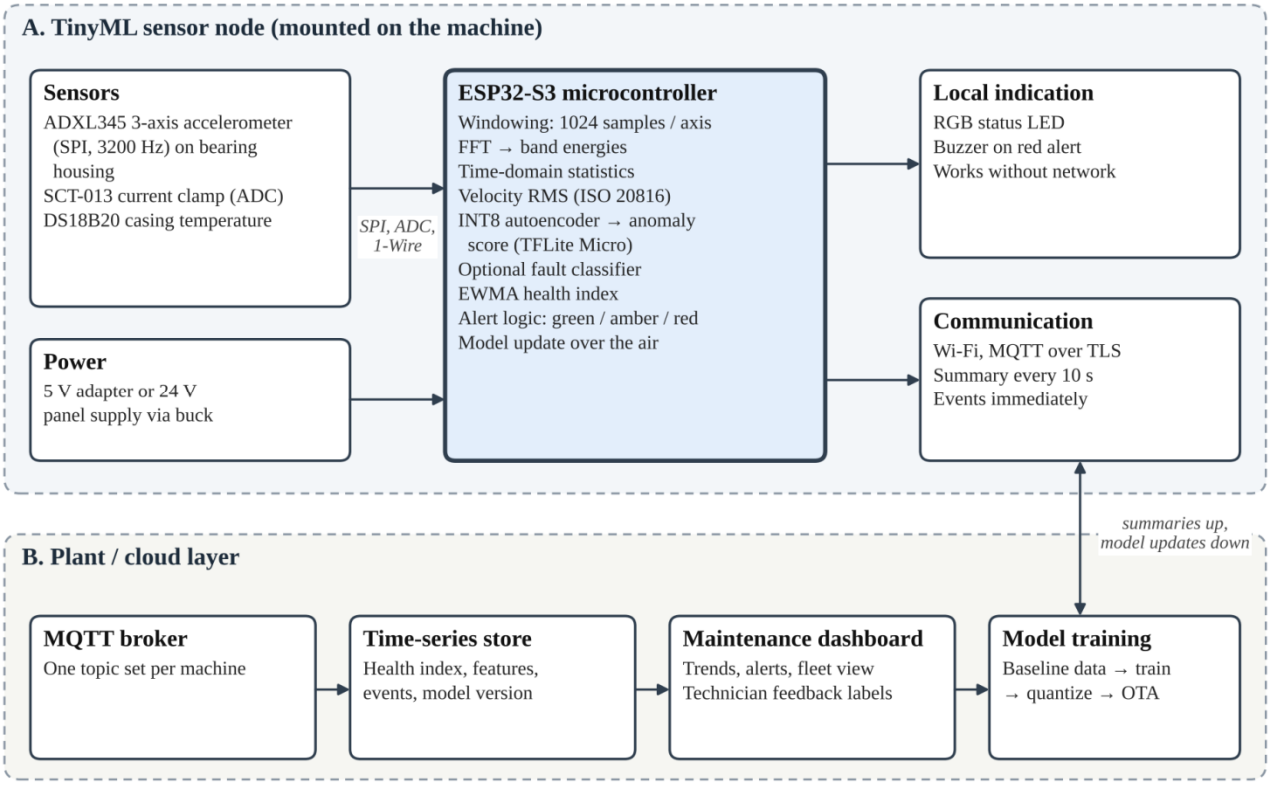


Fig. 1 Architecture of the TinyML predictive maintenance system

Because every decision is made on the node, the network is used only for reporting. If it fails, the node continues to monitor the machine and raise local alarms, and it uploads buffered summaries when the connection returns.

IV. HARDWARE DESIGN

Table 3 lists the components of one sensor node. The total cost of the parts is low compared with even a small industrial motor, which is the condition for monitoring every machine rather than only the most critical ones.

Table 3 Components of the sensor node

Component	Key specification	Role
ESP32-S3 module [21]	Dual-core Xtensa LX7 up to 240 MHz, 512 KB SRAM, vector instructions, Wi-Fi + Bluetooth LE	Acquisition, feature extraction, TinyML inference, communication
ADXL345 accelerometer [22]	3-axis, ±16 g, up to 3200 Hz output data rate, SPI, 32-sample FIFO	Vibration at the bearing housing
SCT-013 current transformer [23]	Split-core, clamps around one motor phase, no electrical contact	Motor current (load, electrical faults)
DS18B20 temperature sensor [24]	Digital, 1-Wire, ±0.5 °C typical accuracy	Casing temperature near the bearing
Status LED and buzzer	RGB LED, 5 V buzzer	Local green / amber / red indication
Power supply	5 V adapter or 24 V panel supply with buck converter	Powers the node
Enclosure and mount	IP-rated box; stud, adhesive or magnetic mount for the accelerometer	Protection and rigid coupling to the machine

The accelerometer is mounted as close as possible to the load-carrying bearing, with a rigid stud or adhesive mount; a loose mount acts as a mechanical filter and hides high-frequency content.

The ADXL345 samples at its maximum rate of 3200 Hz, giving a usable analysis band up to 1600 Hz. This band covers the shaft-speed harmonics that reveal imbalance, misalignment and looseness on machines running at 1500–3000 rpm (25–50 Hz), and the lower bearing defect frequencies. The current transformer clamps around one phase conductor without breaking the circuit, and its output passes through a burden resistor and a mid-supply bias network to an ADC input of the ESP32-S3, sampled at 1600 Hz.

V. ON-DEVICE PIPELINE

A. Signal processing and features

Fig. 2 shows the pipeline that runs on every window. The accelerometer FIFO is read over SPI into a double buffer, so a new window is collected while the previous one is processed. Each window holds N = 1024 samples per axis, or 0.32 s of data. After the mean is removed and a Hann window is applied, a radix-2 fast Fourier transform (FFT) [19] gives the spectrum X_k with a bin spacing of 3200/1024 = 3.125 Hz. The spectrum from 0 to 1600 Hz is divided into 32 bands of 50 Hz, and the energy E_b of each band is computed as in (1):

$$E_b = \Sigma_{k \in b} |X_k|^2 \quad (1)$$

For each axis the root-mean-square value, the peak, the crest factor CF and the kurtosis K are also computed (2). The crest factor and kurtosis rise when a defect produces short impacts, which is typical of early bearing damage.

$$x_{rms} = \sqrt{[(1/N)\,\Sigma\, x_i^2]}, \quad CF = x_{peak} / x_{rms}, \quad K = [(1/N)\,\Sigma\,(x_i - \mu)^4] / \sigma^4 \quad (2)$$

The overall vibration velocity, which ISO 20816-1 uses to judge machine condition [20], is obtained by integrating the acceleration spectrum in the frequency domain over the 10–1000 Hz band (3):

$$v_{rms} = \sqrt{[\ \Sigma_{10 \le f_\{k} \le 1000\}\ (\ |A_k| / (2\pi\, f_k)\ )^2\ ]} \quad (3)$$

where A_k is the RMS acceleration amplitude in bin k. The motor current gives its RMS value, crest factor and total harmonic distortion, and the temperature sensor gives one value. Table 4 lists the full feature vector of 113 values, which is normalised with the mean and standard deviation measured during the baseline period.

Table 4 Feature vector computed for each 0.32 s window

Source	Features	Count
Acceleration, 3 axes	Band energies, 32 bands of 50 Hz per axis, log-scaled	96
Acceleration, 3 axes	RMS, peak, crest factor, kurtosis per axis	12
Acceleration	Overall velocity RMS, 10–1000 Hz	1
Motor current	RMS, crest factor, total harmonic distortion	3
Temperature	Casing temperature	1
Total		113

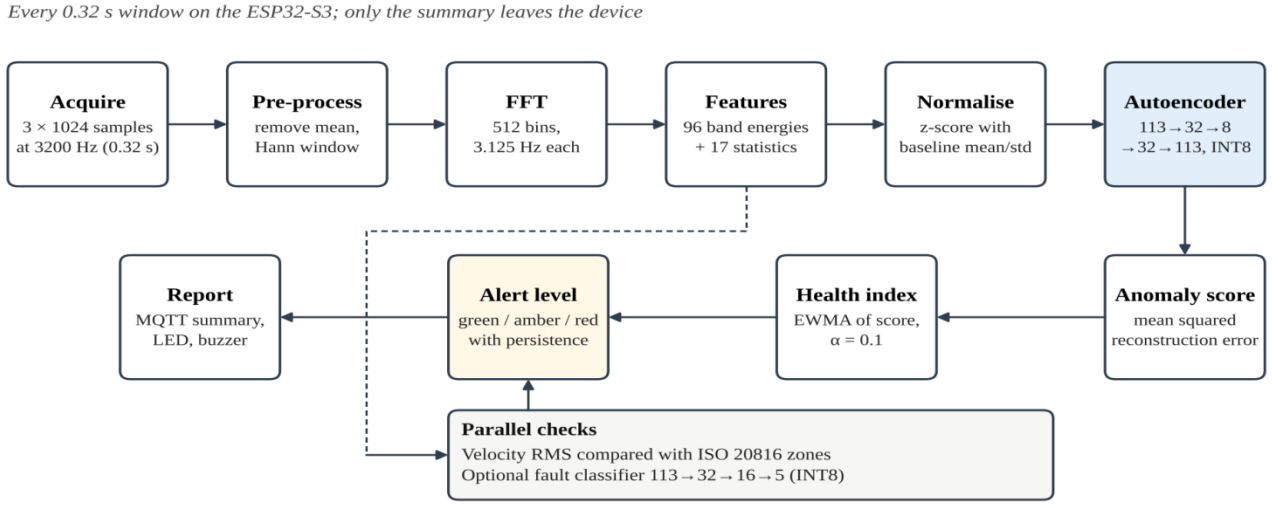


Fig. 2 On-device signal processing and inference pipeline

B. Autoencoder anomaly detection

The main model is a fully connected autoencoder with layer sizes 113→32→8→32→113. It is trained to reproduce the normalised feature vector z of the healthy machine through an 8-value bottleneck, so it learns the normal relationships between the features. A fault changes these relationships, and the reconstruction $\hat{z}$ becomes poor. The anomaly score e of a window is the mean squared reconstruction error (4), and the alarm threshold τ is set from the distribution of errors on held-out healthy data (5), with k = 4 as the starting value:

$$e = (1/n) \Sigma_{i=1}^{n} (z_i - \hat{z}_i)^2, \quad n = 113 \tag{4}$$

$$\tau = \mu_e + k \sigma_e \tag{5}$$

The model is trained offline, converted to TensorFlow Lite with full INT8 post-training quantization of weights and activations [18], and executed on the node by TensorFlow Lite Micro [3]. If labelled fault data become available for a machine type, a small classifier (113→32→16→5) is added to name the likely fault: imbalance, misalignment, looseness, bearing defect or electrical fault.

C. Health index and alert levels

A single window can be disturbed by a passing load change or a knock on the machine, so the score is smoothed into a health index H with an exponentially weighted moving average (6), where $\alpha = 0.1$ gives an effective memory of about 10 windows or 3 s:

$$H_t = \alpha e_t + (1 - \alpha) H_{t-1} \tag{6}$$

Alert levels are assigned as in Table 5. A level must persist for 30 consecutive windows (about 10 s) before it is raised, which suppresses short transients. The velocity check against ISO 20816 zones acts independently, so a machine with high absolute vibration is flagged even if it has been that way since the baseline was recorded.

Table 5 Alert levels

Level	Condition (persisting ≥ 30 windows)	Action
Green	$H < \tau$ and velocity in ISO 20816 zone A or B	Normal operation; summary every 10 s
Amber	$\tau \le H < 2\tau$, or velocity in zone C	Event sent; plan inspection at next stop
Red	$H \ge 2\tau$, or velocity in zone D	Event sent, buzzer on; inspect urgently

D. Deployment lifecycle

Fig. 3 shows how a node is brought into service. After installation, the node records feature vectors while the machine is known to be healthy, typically for three to seven days, so that the baseline covers its normal range of loads and ambient temperatures. The autoencoder and threshold are trained from this data, quantized and sent to the node over the air. During operation, technicians confirm or reject alerts on the dashboard; confirmed healthy periods and labelled faults are used to retrain the model, which addresses the drop in on-device accuracy reported in [16].

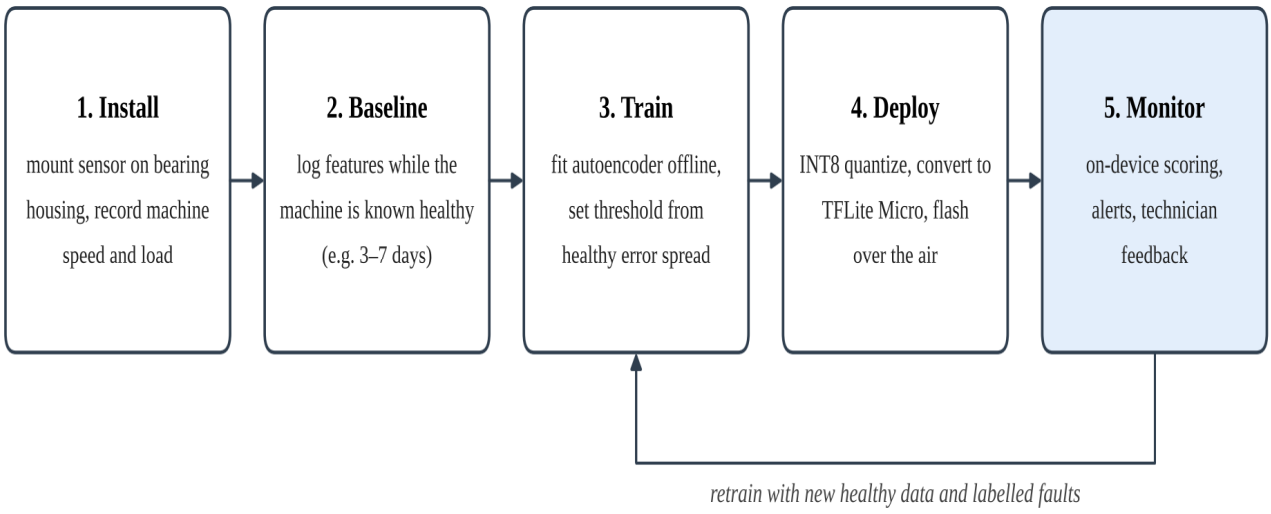


Fig. 3 Deployment and retraining lifecycle of a sensor node

VI. EXPERIMENTAL METHODOLOGY

A. Test rig and datasets

The system is validated in two stages. In the first stage, the feature pipeline and models are evaluated offline on public datasets: the CWRU bearing data [5], resampled to 3200 Hz, and the MAFAULDA rotating-machinery data used in [17]. The autoencoder is trained only on the normal records and tested on both normal and fault records.

In the second stage, the complete node is tested on a laboratory rig consisting of a 0.5 hp single-phase induction motor driving a disc and belt load, with the accelerometer on the drive-end bearing housing. After a baseline period, faults are introduced one at a time and removed between tests:

- Imbalance: bolt a known mass to the disc at a known radius.
- Misalignment: insert shims under one motor foot.
- Mechanical looseness: loosen one mounting bolt.
- Bearing defect: fit a bearing with a seeded outer-race defect or a worn bearing.
- Load change without fault: vary belt tension within the normal range to test for false alarms.

B. Evaluation metrics

Detection performance is measured with the area under the receiver operating characteristic curve (AUC) of the anomaly score, and with precision, recall and F1 score at the chosen threshold (7), where TP, FP and FN are true positives, false positives and false negatives:

$$P = TP / (TP + FP), \quad R = TP / (TP + FN), \quad F1 = 2PR / (P + R) \tag{7}$$

Table 6 Performance metrics and targets

Metric	How it is measured	Target
AUC of anomaly score	Normal vs fault windows, per dataset and rig fault	≥ 0.95
F1 at threshold τ	Eq. (7)	≥ 0.90
False alarms	Amber or red alerts per day on the healthy rig	≤ 1 per week
Detection delay	Fault introduced → amber alert	≤ 30 s
INT8 vs float accuracy loss	Change in AUC after quantization	≤ 0.01
Inference time	Feature extraction + model, measured with on-chip timer	< 320 ms (real time)
Memory use	Peak RAM and flash from the build map	< 64 KB RAM
Node power	Average current from a USB power meter	Report

VII. RESULTS AND DISCUSSION

A. Model size, computation and memory

Table 7 summarises the resource requirements of the proposed models, computed from their architecture, alongside values reported in the literature. The autoencoder has 7,929 parameters, so its INT8 weights occupy about 8 KB, and one inference needs 7,744 multiply–accumulate (MAC) operations. The optional classifier adds 4,261 parameters and 4,208 MACs. The three 1024-point FFTs need $3 \times (1024/2) \times \log_2 1024 = 15{,}360$ butterfly operations per window.

Table 7 Resource requirements: proposed models (computed) and reported values from the literature

Model	Parameters	Flash / model size	RAM	Source
Proposed autoencoder (INT8)	7,929	≈ 8 KB weights	See text	Computed
Proposed classifier (INT8)	4,261	≈ 4 KB weights	See text	Computed
CNN on STM32H743	36,390	142 KB	11.3 KB	[11]
INT8 1-D CNN, Nano 33 BLE	—	32 KB	8.3 KB	[14]
MLP, Nano 33 BLE	—	44.3 KB	1.9 KB	[16]
INT8 MLP, MAFAULDA	—	37 KB	—	[17]
Vibration autoencoder	—	7.5 KB	—	[10]

The main memory users are the signal buffers rather than the model. A double buffer of 3 axes × 1024 samples × 2 bytes needs 12 KB, the FFT working buffer of 1024 complex single-precision values needs 8 KB and is reused for each axis, and the feature vector needs under 0.5 KB.

Together with the model weights and a small tensor arena for the largest layer, the pipeline needs under 30 KB, less than 6% of the ESP32-S3's 512 KB of SRAM [21]. The computation of about 23,000 MACs and butterflies per 0.32 s window is a very small load for a 240 MHz dual-core processor, so the pipeline can run in real time on one core while the other handles communication; the actual inference time is measured in the tests of Section VI.

B. Data volume

Streaming raw vibration from three axes at 3200 Hz with 16-bit samples produces 19.2 kB/s, or 1.66 GB per machine per day. Sending every feature vector would still produce 127 MB per day. Running inference on the node and sending a 100-byte summary every 10 s reduces this to 0.86 MB per day, a reduction by a factor of about 1,900 compared with raw streaming (Fig. 4). This is what makes monitoring many machines practical over an ordinary Wi-Fi network without a data-collection server.

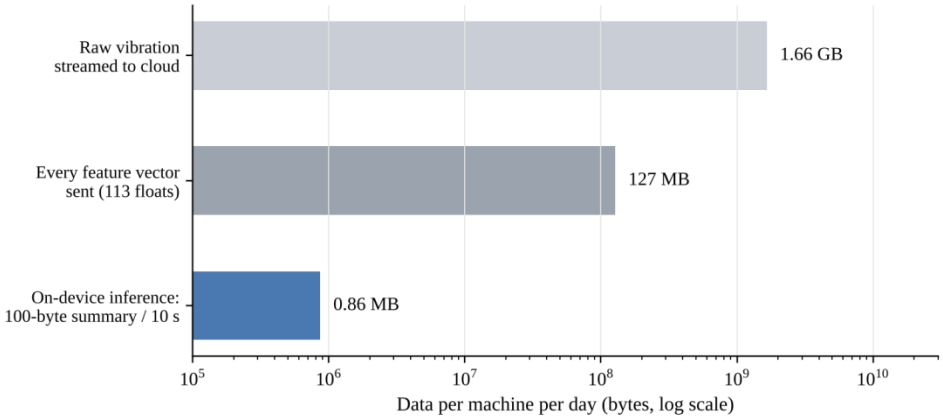


Fig. 4 Data sent per machine per day for three monitoring approaches

C. Validation results

Results of the validation in Section VI are to be entered in Table 8. Report the number of windows and test runs for each case, the threshold used, and all false alarms.

Table 8 Validation results (to be completed after testing)

Test case	AUC	F1 at τ	Detection delay (s)	Notes
CWRU (resampled to 3200 Hz)	[Insert]	[Insert]	—	Offline
MAFAULDA	[Insert]	[Insert]	—	Offline
Rig: imbalance	[Insert]	[Insert]	[Insert]	
Rig: misalignment	[Insert]	[Insert]	[Insert]	
Rig: looseness	[Insert]	[Insert]	[Insert]	
Rig: bearing defect	[Insert]	[Insert]	[Insert]	
Rig: load change only	—	—	—	False alarms per day: [ ]
INT8 vs float AUC change	[Insert]	—	—	
Inference time / RAM / power	—	—	—	[ms] / [KB] / [mA]

D. Discussion

The proposed design follows the direction suggested by the reviewed work but adapts it to the situation of a small factory. Compact dense networks were shown to survive INT8 quantization better than convolutional or recurrent models [17], and the autoencoder used here is of that type and similar in size to the 7.5 KB model in [10]. Training only on healthy data from the monitored machine removes the dependence on labelled fault datasets [11], [14], and the retraining lifecycle responds to the gap between validation and on-device accuracy observed in [16]. The traffic-light alert scheme follows [8] and links the output to a maintenance decision rather than a classification accuracy.

The design has limitations. An autoencoder detects that the machine has changed, not why; naming the fault still needs the optional classifier and labelled data. Machines whose speed or load varies widely need a baseline that covers all operating states, or separate models per state.

The ADXL345's 1600 Hz analysis band is enough for imbalance, misalignment and looseness, but early bearing damage often shows first at higher frequencies, which a faster MEMS or piezoelectric sensor would capture. The resource figures in Section 7.1 are computed from the design and must be confirmed by measurement.

VIII. CONCLUSION AND FUTURE SCOPE

This paper presented the design of a TinyML-based predictive maintenance system for low-cost industrial equipment. An ESP32-S3 sensor node extracts 113 vibration, current and temperature features every 0.32 s and scores them with an INT8 autoencoder trained only on the healthy machine, raising green, amber and red alerts locally and reporting a short summary over MQTT. The autoencoder needs 7,929 parameters and about 7,700 MACs per window, the whole pipeline fits in under 30 KB of RAM, and on-device inference reduces the data sent by a factor of about 1,900 compared with raw streaming. A review of 17 studies showed that this machine-specific, unsupervised approach addresses the lack of labelled fault data and the loss of accuracy after deployment seen in earlier TinyML work.

Future work will complete the validation on the test rig and public datasets, add on-device fine-tuning in the manner of [9], estimate remaining useful life from the trend of the health index, support machines with variable speed, and evaluate battery-powered nodes with duty-cycled sampling.

IX. ACKNOWLEDGMENT

[Insert acknowledgment of sponsors, institutions or individuals who supported this work.]

REFERENCES

[1] C. Krupitzer, T. Wagenhals, M. Züfle, V. Lesch, D. Schäfer, A. Mozaffarin, J. Edinger, C. Becker, and S. Kounev, “A survey on predictive maintenance for Industry 4.0,” arXiv:2002.08224, 2020.

[2] P. P. Ray, “A review on TinyML: State-of-the-art and prospects,” Journal of King Saud University – Computer and Information Sciences, vol. 34, no. 4, pp. 1595–1623, 2022.

[3] R. David, J. Duke, A. Jain, V. Janapa Reddi, N. Jeffries et al., “TensorFlow Lite Micro: Embedded machine learning for TinyML systems,” in Proc. Machine Learning and Systems (MLSys), vol. 3, pp. 800–811, 2021.

[4] C. Banbury, V. Janapa Reddi, P. Torelli, N. Jeffries, C. Kiraly et al., “MLPerf Tiny benchmark,” in Proc. NeurIPS Track on Datasets and Benchmarks, vol. 1, 2021 (arXiv:2106.07597).

[5] W. A. Smith and R. B. Randall, “Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study,” Mechanical Systems and Signal Processing, 2015, doi: 10.1016/j.ymssp.2015.04.021.

[6] H. Purohit, R. Tanabe, T. Ichige, T. Endo, Y. Nikaido, K. Suefusa, and Y. Kawaguchi, “MIMII dataset: Sound dataset for malfunctioning industrial machine investigation and inspection,” in Proc. Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), 2019, pp. 209–213.

[7] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, “Real-time motor fault detection by 1-D convolutional neural networks,” IEEE Transactions on Industrial Electronics, vol. 63, no. 11, pp. 7067–7075, 2016, doi: 10.1109/TIE.2016.2582729.

[8] S. Givnan, C. Chalmers, P. Fergus, S. Ortega-Martorell, and T. Whalley, “Anomaly detection using autoencoder reconstruction upon industrial motors,” Sensors, vol. 22, no. 9, Art. no. 3166, 2022, doi: 10.3390/s22093166.

[9] H. Ren, D. Anicic, and T. A. Runkler, “TinyOL: TinyML with online-learning on microcontrollers,” arXiv:2103.08295, 2021.

[10] K. S. Charan, “An auto-encoder based TinyML approach for real-time anomaly detection,” SAE International Journal of Advances and Current Practices in Mobility, vol. 5, no. 4, pp. 1496–1501, 2023, doi: 10.4271/2022-28-0406.

[11] W. Liao, “Real time bearing fault diagnosis based on convolutional neural network and STM32 microcontroller,” arXiv:2304.09100, 2023.

[12] Y. Yoo, H. Jo, and S.-W. Ban, “Lite and efficient deep learning model for bearing fault diagnosis using the CWRU dataset,” Sensors, vol. 23, no. 6, Art. no. 3157, 2023, doi: 10.3390/s23063157.

[13] C.-C. Chung, Y.-P. Liang, and H.-J. Jiang, “CNN hardware accelerator for real-time bearing fault diagnosis,” Sensors, vol. 23, no. 13, Art. no. 5897, 2023, doi: 10.3390/s23135897.

[14] H. Al-Libawy, “Efficient implementation of a tiny deep learning classifier based on vibration-related fault detection using limited resources hardware,” Journal of University of Babylon for Engineering Sciences, vol. 33, no. 1, pp. 88–105, 2025, doi: 10.29196/770r2493.

[15] S. Arciniegas, D. Rivero, J. Piñan, E. Diaz, and F. Rivas, “IoT device for detecting abnormal vibrations in motors using TinyML,” Discover Internet of Things, vol. 5, Art. no. 41, 2025, doi: 10.1007/s43926-025-00142-4.

[16] W. Ayadi, A. Amouri, S. Ali, N. Alzahmi, and M. N. Khalid, “Leveraging lightweight AI for anomaly detection in mechanical power transmission at the edge,” EPJ Web of Conferences, vol. 343, Art. no. 05014, 2025, doi: 10.1051/epjconf/202534305014.

[17] C. A. C. de Lucena Filho and F. V. de Almeida, “TinyML for fault detection in rotating machinery: A post-training quantization assessment on the MAFAULDA dataset,” in Proc. Brazilian e-Science Workshop (BRESCI), 2026, pp. 33–40, doi: 10.5753/bresci.2026.249387.

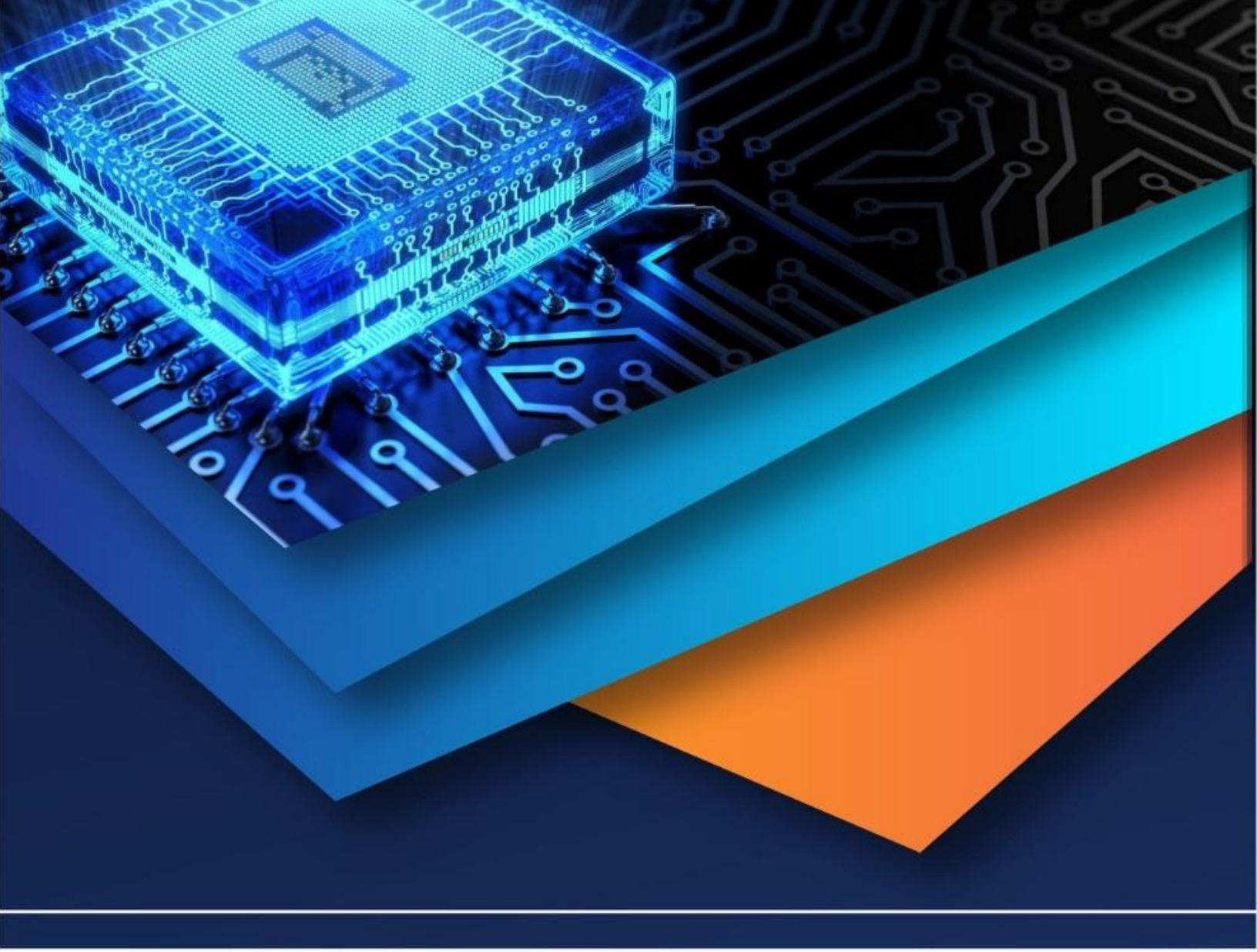
[18] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 2704–2713.

[19] J. W. Cooley and J. W. Tukey, “An algorithm for the machine calculation of complex Fourier series,” Mathematics of Computation, vol. 19, no. 90, pp. 297–301, 1965.

©IJRASET: All Rights are Reserved

[20] ISO 20816-1:2016, “Mechanical vibration — Measurement and evaluation of machine vibration — Part 1: General guidelines,” International Organization for Standardization, 2016.
[21] Espressif Systems, “ESP32-S3 series datasheet,” technical documentation.
[22] Analog Devices, “ADXL345 3-axis, ±2 g/±4 g/±8 g/±16 g digital accelerometer,” datasheet.
[23] YHDC, “SCT-013 split core current transformer,” datasheet.
[24] Analog Devices (Maxim Integrated), “DS18B20 programmable resolution 1-Wire digital thermometer,” datasheet.
[25] OASIS, “MQTT version 3.1.1,” OASIS Standard, A. Banks and R. Gupta, Eds., Oct. 2014.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)