A Context-Aware Edge-Cloud Multi-Agent PHM Framework for Multi-Tool CNC Turning Machines Using a Single Vibration Sensor

Ramesh Krishnamurthy[1], Shweta S[1], and Rachana Sreedhar[1]

[1] *Intellipredikt, Bengaluru, Karnataka, 560078, India*
ramesh@intellipredikt.com
shweta.s@intellipredikt.com
rachana.s@intellipredikt.com

ABSTRACT

CNC turning machines operate under highly variable and context dependent conditions, where multiple cutting tools and machine subsystems share a common structural vibration path. This creates a practical monitoring challenge: a single sensor captures a superposed vibration response, while tool wear and machine degradation evolve differently across tools, operations, and cutting conditions. This paper presents a context-aware edge cloud multi-agent Prognostics and Health Management (PHM) framework for multi-tool CNC turning machines using a single spindle mounted vibration sensor. The framework assigns a logical PHM agent to each tool, while a context router uses controller side process information such as tool identity, spindle speed, feed rate, and depth of cut, to transform the shared vibration stream into tool specific feature streams. Each agent performs in situ baseline calibration, health indicator (HI) estimation, monitoring mode selection, online degradation tracking, and, where applicable, Remaining Useful Life (RUL) estimation. To improve deployment robustness, the library includes adaptive threshold estimation, online threshold refinement, and explicit handling of anomaly dominant and degradation dominant operating regimes. The framework is implemented as an edge cloud architecture in which the edge performs feature extraction, context-aware routing, HI computation, and alert generation, while the cloud ingests telemetry and curated feature streams for visualization, storage, and lifecycle management. The study includes an industrial CNC case study with three tool types and an auxiliary evaluation on the UC Berkeley milling dataset. Results show that the framework can isolate tool-specific degradation behavior using a shared sensor, provide early warnings for rapidly degrading tools, and support low-latency edge deployment.

Keywords - Prognostics and Health Management, CNC turning, tool wear monitoring, edge AI, multi-agent monitoring, context aware feature fusion, health indicator, remaining useful life.

1. INTRODUCTION

CNC turning machines are inherently context dependent. A single production cycle may involve roughing, finishing, and grooving tools, each operating under different spindle speeds, feeds, and depths of cut. In addition, machine subsystems such as spindle bearings, the drive train, and structural elements contribute to the measured vibration response. As a result, vibration signatures in CNC turning are not stationary and cannot be reliably monitored using a single monolithic threshold.

A practical industrial requirement is to achieve low-cost deployment with minimal sensing complexity. This motivates the use of a single spindle-mounted vibration sensor. However, single-sensor deployment introduces a key PHM challenge: the same signal channel is shared across multiple tools and operating regimes. A viable solution therefore requires both tool-specific interpretation and strong use of process context.

This work addresses that challenge through a context-aware edge-cloud multi-agent PHM framework, extending the IntelliMaint health indicator and prognostics reference framework (Krishnamurthy, S, Sreedhar, & Chandrashekar, 2025). The framework is designed for deployment in industrial environments where low latency, explainability, and minimal sensor count are more important than training a large, task-specific deep model offline. Each cutting tool is represented by a logical PHM agent that maintains its own state, baseline, health index trajectory, mode, and alert logic. The term agent is meant as a dedicated logical monitoring entity for each tool. A context router uses controller-side information to associate the correct vibration-derived feature stream with the correct tool agent.

1.1. Research Objective

The objective is to develop a deployable PHM runtime that can monitor multiple cutting tools using a single vibration sensor, while supporting online baseline learning, tool-specific health tracking, early degradation warning, and low-latency deployment.

1.2. Contributions

The contributions outlined in this paper are listed below:

- A context-aware feature routing mechanism that converts a shared vibration stream into tool-specific PHM channels using controller context.
- A logical multi-agent formulation in which each tool is assigned an independent PHM agent with its own baseline, HI state, mode, alerts, and optional RUL state.
- A lightweight edge-compatible health-monitoring runtime based on baseline relative wear evidence, adaptive damage accumulation, and bounded HI construction.
- An edge cloud deployment architecture that separates low latency inference at the edge from storage, visualization, and lifecycle functions in the cloud.
- A practical industrial case study showing the feasibility of multi tool monitoring using a single sensor.

2. RELATED WORK

2.1. Tool Wear Monitoring in CNC Turning

Tool wear monitoring has been studied for several decades using indirect indicators such as cutting forces, motor current, vibration, and acoustic emission. Early reviews of intelligent tool wear monitoring techniques (Dan & Mathew, 1990) established the value of these signals as proxies for wear, while more recent surveys of artificial intelligence approaches to tool condition monitoring (Munaro, Attanasio, & Del Prete, 2023) summarize the wide adoption of machine learning models that learn non-linear mappings from such signals to wear or remaining useful life. Many recent contributions emphasize multi-sensor information fusion (Wang, Wang, Wu, & Xie, 2024) to improve robustness, but multi-sensor deployment increases installation cost and complexity, particularly on production CNC machines where multiple cutting tools share the same machine envelope.

A consistent finding across this body of work is that supervised wear models trained on one tool, material, or cutting regime transfer poorly to others without retraining (X. Li, 2002), (Wang et al., 2024); cutting parameters such as spindle speed, feed, and depth of cut materially shape the vibration signature, and this contextual variability is a primary obstacle to deploying a single fixed model across a multi-tool turning operation. Our framework attempts to address this transferability problem explicitly through context aware feature routing rather than through additional supervised retraining.

2.2. Vibration Based PHM and Edge Deployment

Vibration analysis is a primary modality for condition monitoring of rotating machinery, with comprehensive recent surveys of vibration sensors and their use in condition monitoring (Hassan, Panduru, & Walsh, 2024) cataloging how a single accelerometer can capture imbalance, looseness, gearmesh, bearing, and wear induced modulation effects. Within machining specifically, vibration based approaches have been combined with deep learning for tool wear monitoring; (Huang, Zhu, Lei, Li, & Tian, 2021) used short time Fourier transform features with deep convolutional networks for milling tool wear, and similar work has applied vibration analysis to bearing fault detection (Gaikwad, Mundhada, Nagre, & Patil, 2024). The deployment trend over the past five years has been to push more of the analytic stack onto industrial edge hardware, motivated by bandwidth, latency, and connectivity considerations. (Costa, Eberhardt, Chen, & Roßmann, 2023) demonstrate edge based vibration monitoring oriented toward Industry 4.0 predictive maintenance, while broader edge–cloud frameworks have been proposed for streaming machinery analytics (Calabrese, Regattieri, Bortolini, Gamberi, & Pilati, 2021), for modular industrial IoT predictive maintenance platforms (Resende et al., 2021), and for IIoT based condition monitoring with smart sensors at the edge (Y. Li, Meng, Zhang, & Song, 2020). Most of these systems, however, either assume a dedicated sensor per asset or treat the machine as a single asset; they do not address the shared sensor multi-tool monitoring problem that arises on a CNC turret where one accelerometer covers several inserts.

2.3. Multi-Agent PHM

Multi-agent system (MAS) approaches to PHM have been investigated as a way of structuring distributed monitoring across heterogeneous asset fleets in IIoT enabled environments. (Salvador Palau & Dhada, 2019) analyze multi-agent system architectures for collaborative prognostics and discuss the trade-offs between distributed and hierarchical topologies for failure prediction across fleets. More recent work positions multi-agent IIoT and machine learning approaches in the context of Industry 4.0 maintenance pipelines (Abd Elhaleem, Zanfal, & Hamdy, 2025), and machine-tool-specific studies have begun to incorporate variable working conditions and multi-sensor fusion into RUL prediction (X. Li, 2002); (Y. Li et al., 2020). Across this literature, however, each agent is typically tied to its own dedicated sensing path: an agent observes one asset through one or more sensors mounted on that asset and exchanges summary information with peer agents. The current problem inverts this assumption. Multiple logical agents - one per cutting tool, must share a single physical vibration sensor mounted on the spindle, be-

cause installing per tool sensors on a turret is mechanically impractical and economically unjustified. This shared sensor multi agent setup demands a context routing layer so that the same incoming signal stream can be interpreted differently by each agent depending on which tool is currently engaged. Such a layer is not a feature of the multi-agent PHM frameworks reviewed above and, to the best of our knowledge, has not been formalized in the multi-tool CNC turning context addressed here. The component agnostic IntelliMaint health indicator and prognostics framework (Krishnamurthy et al., 2025) provides the per agent HI and RUL primitives that the present paper extends with context aware multi-agent routing.

2.4. Research Gap

Combining the threads above, tool wear monitoring with conventional and AI based methods, vibration based and edge deployed PHM, tool RUL under variable working conditions, and multi-agent prognostic architectures each addresses a distinct subset of the problem space. The gap targeted by this paper is the intersection of four requirements that are seldom satisfied simultaneously in prior work: (i) single sensor deployment, in which one accelerometer covers multiple cutting tools; (ii) multi-tool monitoring, in which separate logical agents maintain independent health states for each tool; (iii) edge compatible runtime logic with low, deterministic latency; and (iv) tool specific context awareness that lets the system reinterpret a shared signal as different tools engage and disengage. The remainder of this paper presents a framework that addresses these four requirements jointly by extending the IntelliMaint HI and prognostics framework (Krishnamurthy et al., 2025) with a context aware feature router and per tool monitoring agents. The framework is validated on a production CNC dataset and on the public UC Berkeley milling dataset (Agogino & Goebel, 2007) and compared against standard RUL baselines.

3. PROPOSED FRAMEWORK

3.1. Edge Cloud Multi-Agent Architecture Overview

The proposed system consists of a two-part edge–cloud architecture (Fig 1), where real-time data collection and analysis are performed at the edge layer, while recording, storage, and other offline tasks are handled by the backend server, typically deployed on a cloud instance. The edge layer processes tri-axial vibration measurements collected from cutting tools and continuously estimates a HI representing the degradation state of the tool. In addition, the edge runtime performs context-aware feature routing, monitoring-mode logic, alert generation, and low-latency telemetry publishing.

The operational workflow implemented at the edge layer is summarized in Fig 2. The edge monitoring architecture consists of the following stages:

- Data ingestion
- Signal processing
- Context-aware routing
- Baseline learning
- Adaptive runtime logic
- Health Index (HI) and decision layer
- Publish outputs to cloud

A continuous monitoring loop updates the active tool agent state for each incoming snapshot and publishes the corresponding status, feature, and alert streams in real time.

The edge layer transmits information such as features and other information such as HI, alerts, and calculated RUL to the backend server. On the edge layer, the term agent is used in the operational PHM sense: each tool is assigned a dedicated logical monitoring entity. A tool agent maintains (i) its own baseline state, (ii) a tool-specific HI trajectory, (iii) mode state, (iv) thresholds and alert status, and (v) optional RUL state. The agents do not require independent physical sensors; instead, they consume context-routed feature streams generated from the shared sensor.

The backend layer stores telemetry and feature streams and supports later model analysis, validation, and configuration management. A dashboard is provided for operators that consumes data via the backend server.

Fig 3 presents the reusable IntelliMaint reference architecture. It defines four layers: the IntelliMaint Library (signal processing, anomaly detection, HI construction, diagnostics, and prognostics), the SDK / Workflow Builder, the Asset Agents, and the Runtime / Platform services. The proposed multi-tool turning solution is an application-specific instantiation of this generic stack.

Fig 4 shows the CNC-specific realization of the reference architecture for the multi-tool turning use case. Multiple Tool Agents operate under a shared edge gateway / orchestrator. Each agent receives a context-qualified feature stream, maintains its own baseline, health indicator trajectory, monitoring mode, and alert / RUL logic, and publishes outputs to telemetry, operator alerts, and dashboard services.

3.2. Context-Aware Feature Fusion

Context awareness is central to the proposed framework. Because the sensor observes a superposed vibration response from the active machining process and machine structure, a raw vibration feature by itself is insufficient to identify which tool it should be attributed to. For each analysis window, the feature vector is augmented with controller-side context including tool identity, operation code, spindle speed, feed rate, and depth of cut. This enriched representation is then routed into the corresponding tool agent. When a tool is inactive, the

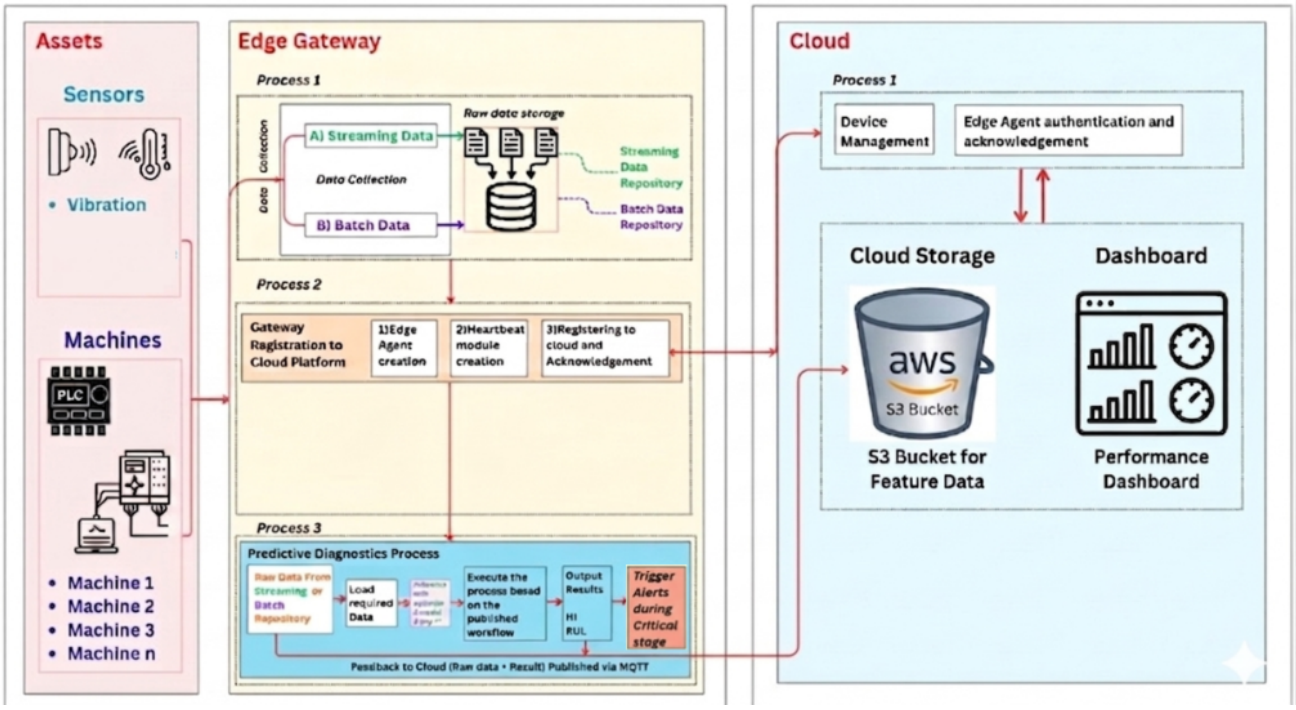


Figure 1. Overall system architecture. consists of a two-part edge–cloud architecture, where real-time data collection from sensors and analysis are performed at the edge layer, while recording, storage, and other offline tasks are handled by the backend server, typically deployed on a cloud instance.

framework marks its logical channel as idle; when it is active, the corresponding routed feature vector is treated as that tool's observation. Thus, context is a functional part of the PHM model because it determines feature attribution, baseline interpretation, and tool-specific decision logic.

3.3. Signal Processing and Virtual Sensor Construction

For each time window, on the edge, signal pre-processing and feature generation is performed. The raw tri-axial signals are de-trended, filtered to suppress drift and noise, and combined via vector magnitude where appropriate. Time-domain features include RMS, peak amplitude, variance, skewness, kurtosis, crest factor, and peak-to-peak range. Frequency-domain features are obtained through FFT-based spectral analysis and band-energy integration. Time frequency features may be derived through wavelet decomposition or related transforms to capture transient impacts and modulation effects.

These features are interpreted as virtual sensors representing different wear mechanisms, including impact-driven wear, frictional degradation, modulation effects, and mechanical instability. This virtual-sensor perspective is useful because it converts a raw vibration stream into a set of physically interpretable channels that can be monitored more robustly than a single scalar threshold.

3.4. Communication and Data Flow

The system generates two primary data streams:

Telemetry stream: This stream includes health indices, RUL estimates, anomaly flags, and contextual metadata (e.g., tool ID and timestamp). It is transmitted to the cloud via an MQTT broker with low latency, enabling real-time visualization and alerting.

Feature stream: This stream consists of context-aware, curated feature vectors produced by each agent. The data are compressed and batch-uploaded to a centralized data lake on AWS S3 to support offline analytics and model development.

On the cloud side, the backend architecture includes a centralized repository for storing both telemetry and feature data, organized by asset and time to enable efficient querying and retrieval and the operator dashboard.

3.4.1. Operational Interface and Deployment View

The monitoring output is presented through a live dashboard interface intended for direct use by operators and maintenance personnel. The interface provides a consolidated view of fleet condition monitoring, including active alerts, asset status, HI trends, and prognostic information such as RUL estimates where available.

As shown in Fig.5, the dashboard summarizes fleet-level health

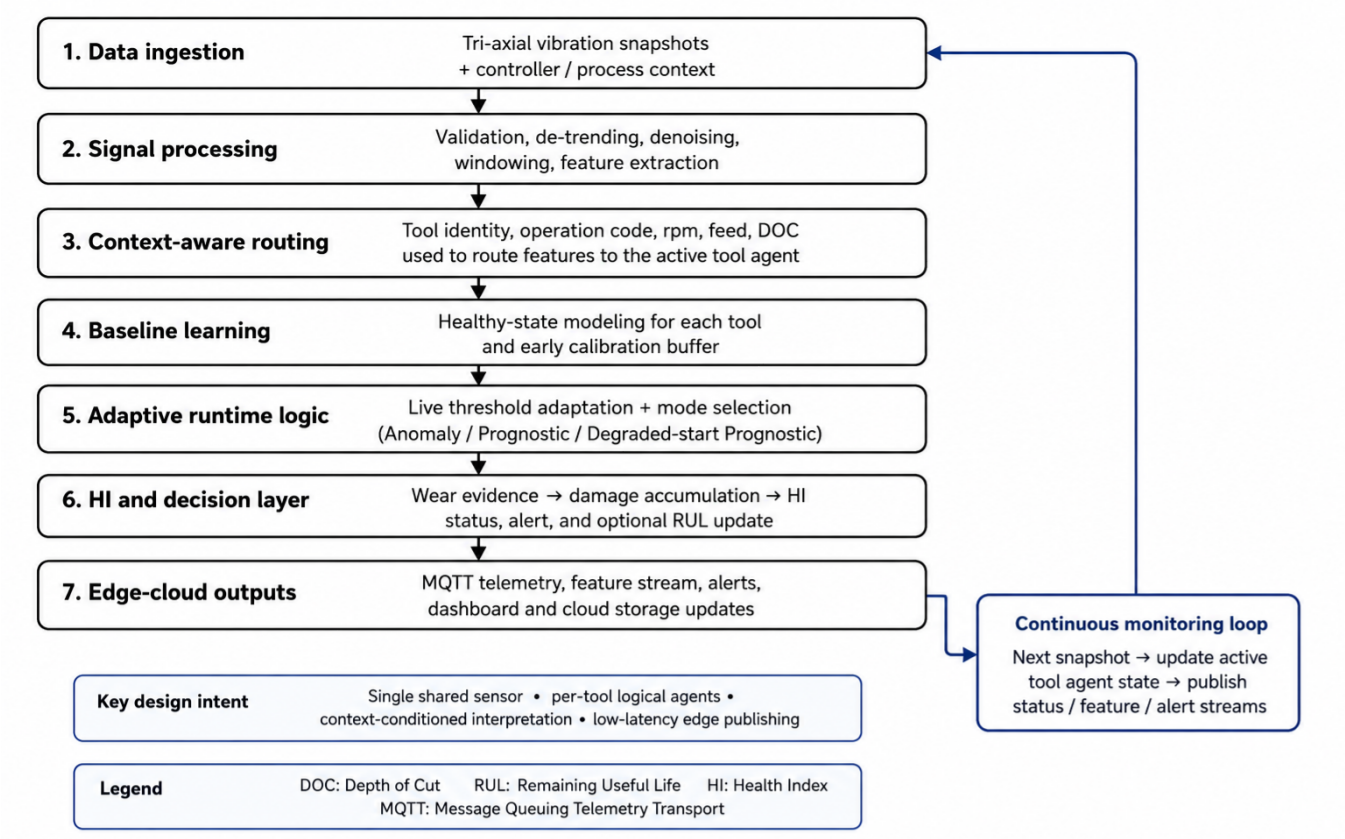


Figure 2. Edge processing flowchart.

information while also presenting tool-specific monitoring outputs. Active warning conditions are highlighted to support rapid operator response, and HI trend visualization enables observation of degradation progression over time.

3.5. Theory

3.5.1. Baseline Learning

Each tool agent begins in a short learning stage that uses the first early data points attributed to that tool through the context router. The agent records the routed feature vectors from these data points and estimates a per-feature healthy mean μ and a diagonal covariance σ_j^2, where the diagonal covariance captures the natural variability of feature j under healthy operation. These statistics define the tool's baseline distribution and serve as the reference against which all subsequent feature vectors are evaluated. The baseline stage also computes a healthy reference for the wear score,

$$s\_base = median(s\_healthy),$$

where $s\_healthy$ is the wear score computed on the baseline windows themselves;s_base anchors the subsequent normalization. Beyond statistics estimation, the stage supports baseline quality checks (rejection of windows with abnormal energy or insufficient feature diversity), context-aware calibration (separate baseline per (tool, regime) combination where the controller context partitions are sufficiently populated), and an optional online threshold adaptation mechanism described in Section 3.5.4.

3.5.2. Health Indicator Construction

The HI is constructed in three stages: a covariance-weighted wear score, a relative wear evidence with adaptive damage accumulation, and a bounded HI mapping.

(Stage 1) Covariance-weighted wear score: Given the per-feature healthy mean μ and diagonal variance σ_j^2 estimated, the wear score for a new feature vector x_t is computed in a Mahalanobis-style form that downweights naturally noisy features:

$$s_t = \sqrt{\sum_j \frac{(x_{tj} - \mu_j)^2}{\sigma_j^2 + \varepsilon}} \tag{1}$$

where ε is a small constant for numerical stability. Features with low healthy variance (i.e., features that were stable during baseline learning) carry higher weight, so deviations in

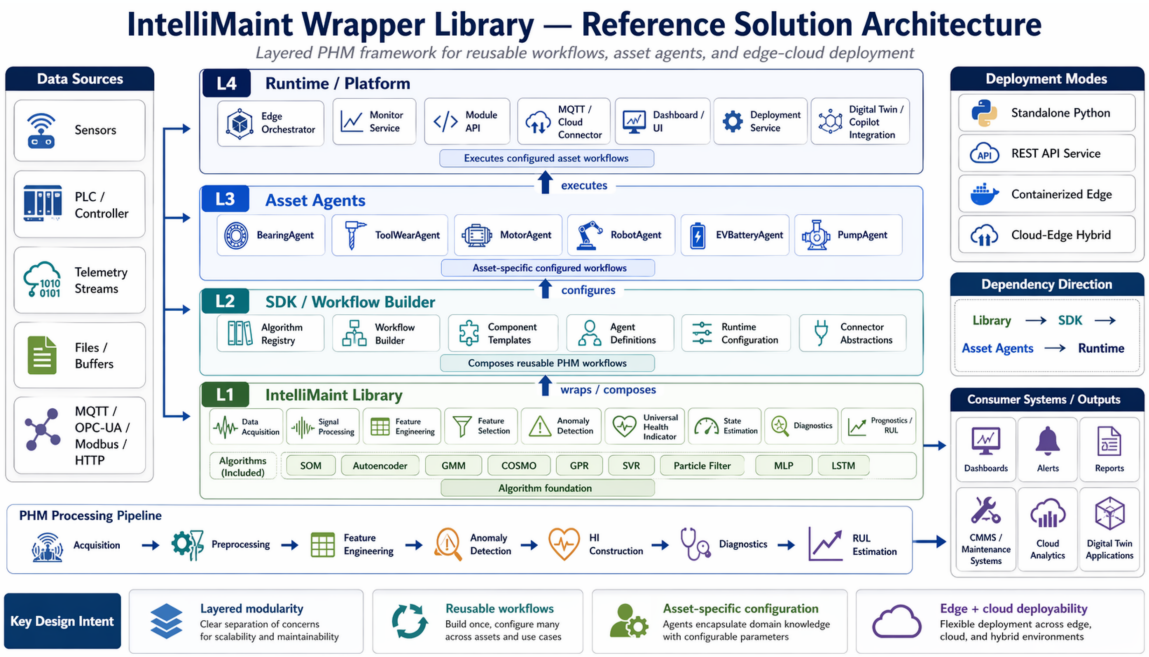


Figure 3. IntelliMaint Wrapper Library - reference solution architecture. It defines four layers: the IntelliMaint Library (signal processing, anomaly detection, HI construction, diagnostics, and prognostics), the SDK / Workflow Builder, the Asset Agents, and the Runtime / Platform services.

normally quiet features dominate the wear score over deviations in features that are intrinsically noisy under healthy operation.

(Stage 2) Relative wear evidence and damage accumulation: The wear score is normalized against the healthy reference s_base and rectified to be non-negative, yielding the relative wear evidence:

$$w_t = max(0, (s_t - s\_base)/s\_base) \tag{2}$$

The rectification is intentional: only deviations above the healthy reference are interpreted as wear evidence, while deviations below are treated as healthy variation and contribute zero. The wear evidence is then accumulated into a damage variable D_t with an adaptive sensitivity coefficient α_t:

$$\begin{aligned} D_t &= D_{t-1} + \alpha_t \, w_t \\ \alpha_t &= \alpha_0(1 + S\bar{w}) \end{aligned} \tag{3}$$

where α_0 is a base sensitivity, S is a sustained-wear scaling factor, and $\bar{w}$ is the average relative wear over a recent window. The form of α_t increases the accumulation rate when sustained wear is observed and reverts to the base sensitivity under transient or low-level variation, so isolated noisy windows do not inflate damage while genuinely persistent degradation accelerates appropriately.

(Stage 3) Bounded HI mapping: The damage variable is mapped into the operator-facing HI through a tanh transformation:

$$H_t = \tanh(D_t) \tag{4}$$

The use of tanh is motivated by deployment considerations. It yields a bounded, smooth, monotonic mapping from accumulated damage to a normalized health scale, which is more convenient for dashboarding and threshold interpretation than an unbounded damage variable. The adaptive coefficient α_t increases responsiveness when sustained wear evidence is observed while preventing overreaction to low-level variation under steady conditions.

3.5.3. Adaptive Runtime Logic

A key design goal of the proposed framework is to avoid forcing all tools into a single prognostic interpretation. In practice, different tools exhibit different degradation characteristics: some remain stable, some show gradual drift, and others

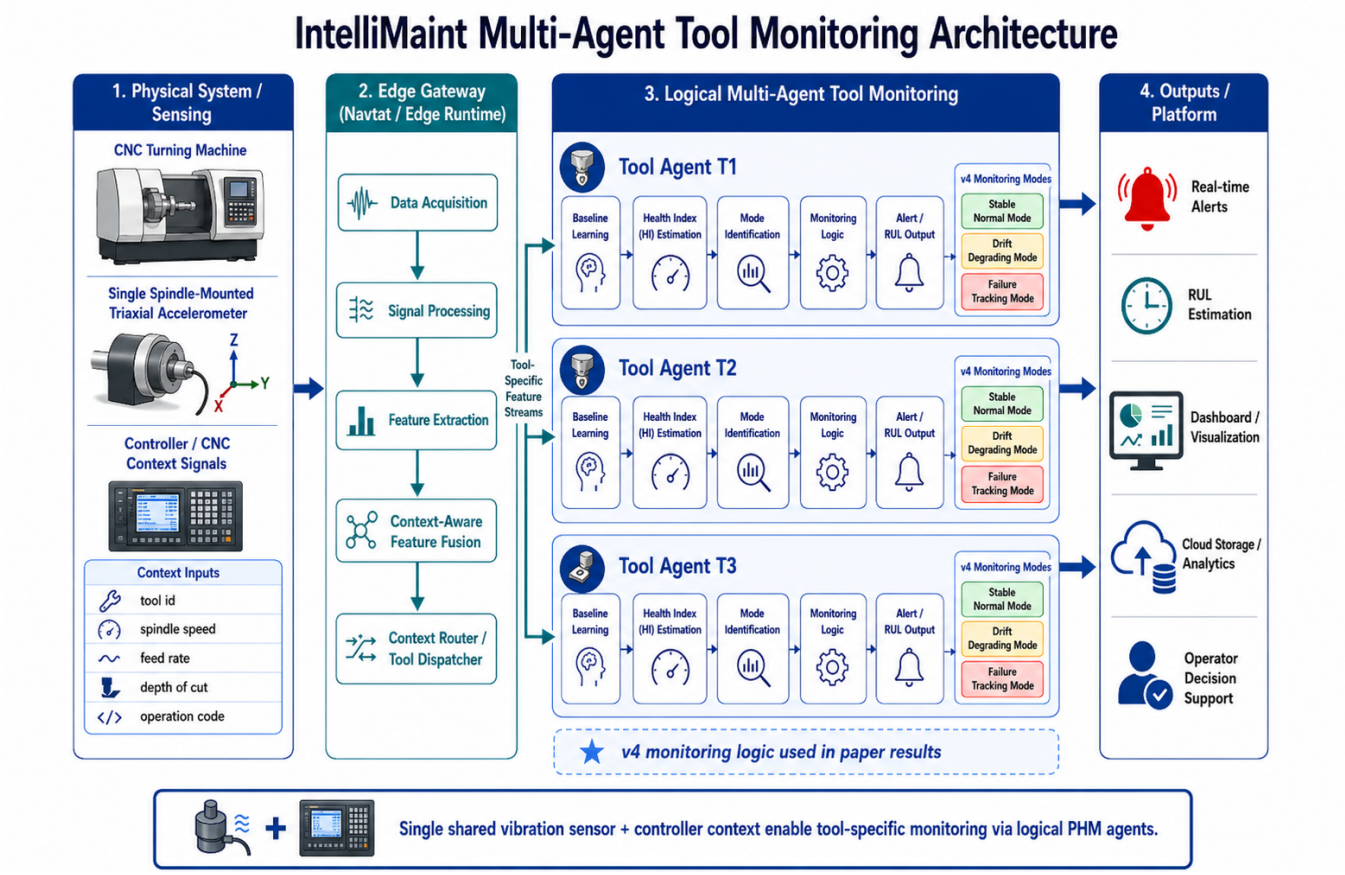


Figure 4. CNC-specific realization of the reference architecture for the multi-tool turning use case. Multiple Tool Agents operate under a shared edge gateway / orchestrator.

experience rapid late-stage degradation. To address this variability, the runtime incorporates an adaptive monitoring layer that classifies the observed HI behavior and activates the corresponding decision logic.

The runtime considers three behavioral categories:

- **Stable or anomaly-dominant behavior:** used when the HI remains near its baseline and the observed variation is primarily caused by operating-regime changes rather than persistent degradation.
- **Drift-degrading behavior:** used when the HI exhibits a slow but sustained upward trend, indicating gradual wear accumulation.
- **Rapid-degradation or failure-tracking behavior:** used when the HI increases sharply, indicating persistent and accelerated degradation.

In the IntelliMaint platform used for the reported results, these behavioral categories are implemented as three monitoring regimes: Stable Normal Mode, Drift Degrading Mode, and Failure Tracking Mode. A slope-based classifier is used to determine the active regime.

At each decision step, a linear regression model is fit over a recent sliding window of the HI trajectory and computes the slope β and the residual variance σ^2 around the fitted trend:

$$\beta = \frac{\mathrm{cov}(k, HI)}{\mathrm{var}(k)} \tag{5}$$

where k indexes the step.

The mode is then assigned by the following rules: a tool is placed in Stable Normal Mode when $|\beta|$ is small and σ^2 is low (no sustained trend, low residual scatter); in Drift Degrading Mode when $0 < \beta < \beta_{\max}$ and σ^2 is moderate (slow positive trend with bounded scatter); and in Failure Tracking Mode when β is high or σ^2 is very high (rapid trend or sharp variability indicative of late-stage degradation).

The thresholds $\beta_{\max}$ and the moderate/high cut-offs for σ^2 are configured per asset class; for the CNC turning case study, they are tuned during baseline learning so that the mode classifier remains in Stable Normal Mode under healthy operation.

Mode transitions drive the threshold logic of Section 3.5.4. It

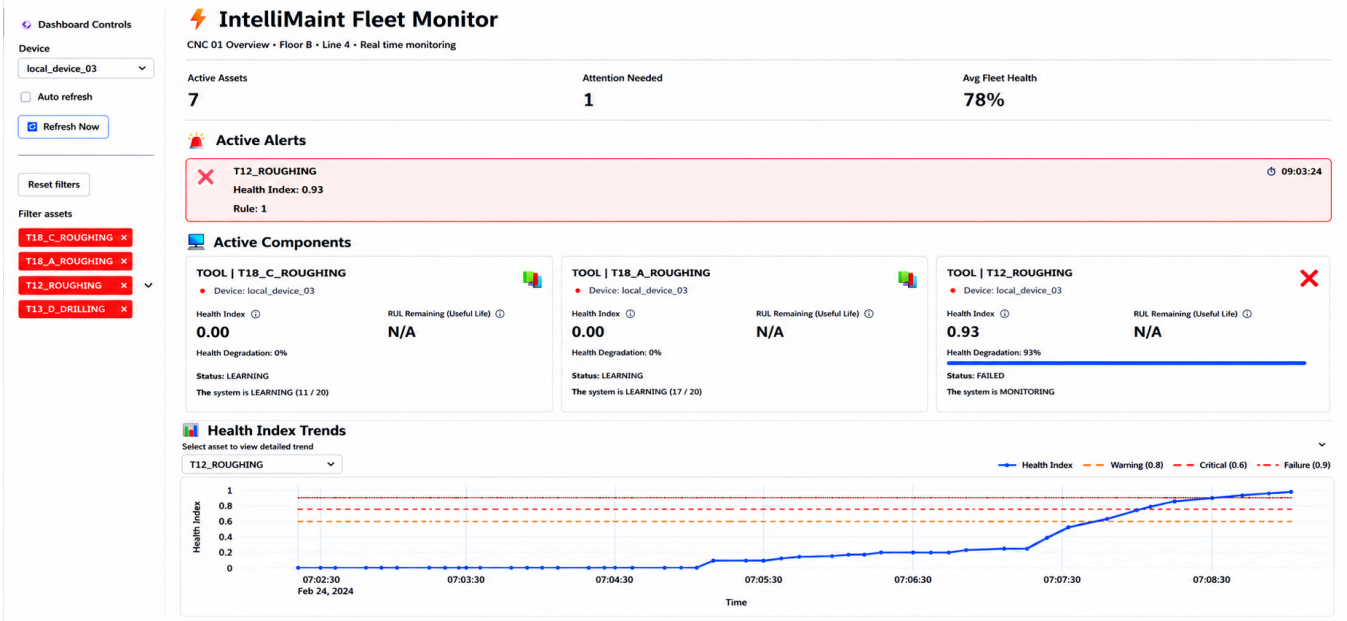


Figure 5. Operator-facing dashboard and deployment view.

also affects when the RUL is computed: RUL is computed only when the mode classifier places the tool in Drift Degrading or Failure Tracking, while in Stable Normal Mode, the runtime suppresses the RUL output as a deliberate design choice, since prognostic numbers issued in the absence of sustained degradation evidence would degrade operator signal-to-noise rather than improve it.

3.5.4. Threshold Estimation

Thresholds are estimated using the initial healthy operating state and the identified monitoring regime. Each mode employs a distinct threshold formulation for the degrading-level alert (HI_{deg}) and the failure-level alert (HI_{fail}). Let HI_{base} denote the median HI observed during the baseline learning stage. The threshold definitions used in the IntelliMaint library are summarized as follows.

Stable Normal Mode

$$HI_{\text{deg}} = 1.0, \qquad HI_{\text{fail}} = 2.0$$

Since

$$HI_t = \tanh(D_t)$$

is bounded within the interval $[0, 1)$, the value $HI_{\text{fail}} = 2.0$ is unreachable, while $HI_{\text{deg}} = 1.0$ is approached only asymptotically. These sentinel thresholds intentionally suppress threshold -based prognostic alerts in *Stable Normal Mode*. In this regime, the agent operates in an anomaly-surveillance posture and relies on excursion-based detection rather than fixed prognostic thresholds. RUL estimation is therefore not issued in this mode.

Drift Degrading Mode

$$HI_{\text{deg}} = HI_{\text{base}} + 0.12$$

$$HI_{\text{fail}} = \max(1.5,\ 10 \times HI_{\text{base}})$$

The degrading threshold is defined as a small fixed offset above the per-tool baseline, providing a sensitive yet non-trivial alert level for gradual wear progression. The failure threshold is lower-bounded by 1.5 to prevent extremely small baseline values from collapsing the failure threshold into the alert region. For larger baseline values, the failure threshold scales proportionally with the healthy operating point through a $10\times$ multiplier. RUL estimation is computed and issued in this mode.

Failure Tracking Mode

$$HI_{\text{deg}} = 0.25 \times HI_{\text{fail}}$$

$$HI_{\text{fail}} = \max(HI_{\text{traj}})$$

Once a tool enters a rapid late-stage degradation regime, fixed threshold offsets become less informative because the HI trajectory evolves rapidly. The thresholds are therefore re-derived directly from the observed trajectory. The failure threshold is defined as the running maximum of the HI trajectory, allowing the displayed failure level to evolve with the actual degradation progression. The degrading threshold is then defined

as a fraction of HI_{fail} to preserve a meaningful early-warning region within the late-stage degradation regime. In this mode, RUL estimation is computed using a tighter regression window.

Online Threshold Adaptation The runtime supports an optional label-free threshold adaptation mechanism in which the alert thresholds for the active mode can be gradually recalibrated using the running statistics μ_{HI} and σ_{HI} of the HI trajectory. A typical adaptive alert envelope is defined as:

$$HI_{\text{alert}} = \mu_{HI} + 3\sigma_{HI}$$

This adaptation mechanism is driven entirely by the observed HI history and does not require failure-time labels, thereby avoiding label leakage during benchmark evaluation.

In the experiments reported in this paper, online threshold adaptation was kept disabled during the test segment to keep the comparison with frozen baselines (SVR, LSTM) symmetric; the threshold values per mode are those derived from baseline learning at the start of the run.

Mode-Driven Threshold Reconfiguration When the mode classifier transitions a tool from one monitoring regime to another, the corresponding threshold family is re-derived using the predefined mode-specific rules described above. This process does not constitute online statistical learning; rather, it is a deterministic runtime reconfiguration triggered by regime transitions. This mechanism enables the framework to accommodate heterogeneous wear behaviors within a unified prognostic architecture.

3.5.5. Remaining Useful Life Estimation

For tools assigned to a prognostic regime like Drift Degrading Mode or Failure Tracking Mode, as determined by the mode classifier, the system estimates RUL using a local sliding window linear regression model applied to the accumulated damage trajectory $D(k)$. RUL estimation is not issued in Stable Normal Mode.

This design was selected to preserve edge-deployment viability and operator interpretability. The estimator is computationally lightweight, deterministic, and straightforward to explain in industrial settings.

Local Linear Fit: At each prognostic decision step k_{current}, a linear model is fit to the most recent W damage samples:

$$D(k) = mk + b \tag{6}$$

where m represents the local damage growth rate (slope) and b denotes the intercept. In IntelliMaint, the window size W is typically set to 10, which was empirically selected to balance responsiveness and noise suppression. A smaller window is used in Failure Tracking Mode to improve responsiveness during rapid late stage degradation.

Fitting is performed in damage space rather than HI space. This choice is deliberate because the $\tanh$ mapping saturates at high damage levels. Consequently, linear extrapolation in HI space would systematically underestimate progression near failure. In contrast, the accumulated damage trajectory remains approximately linear within the degradation regimes of interest and is therefore better suited for local linear projection.

Failure Projection in Damage Space: The mode specific failure threshold HI_{fail} is mapped into damage space using the inverse hyperbolic tangent transformation:

$$D_{\text{fail}} = \operatorname{atanh}(HI_{\text{fail}}) \tag{7}$$

The projected failure point is then estimated by inverting the fitted linear model at the failure-aligned damage level:

$$k_{\text{fail}} = \frac{D_{\text{fail}} - b}{m} \tag{8}$$

The Remaining Useful Life is subsequently computed as:

$$RUL(k) = k_{\text{fail}} - k_{\text{current}} \tag{9}$$

RUL estimation is issued only when the estimated local slope satisfies $m > 0$, ensuring that the projected degradation trajectory remains physically meaningful.

This formulation explicitly links the RUL estimate to the active monitoring regime. The threshold HI_{fail} (and therefore D_{fail}) depends on the active mode, while the regression parameters m and b depend on the local damage trajectory observed within that regime. The resulting RUL is therefore a direct projection of the current degradation state onto the mode defined failure boundary.

Monotonic Stabilizer: The raw RUL estimate may fluctuate as the locally estimated degradation slope changes between successive windows. To improve operator interpretability and output stability, the displayed RUL is constrained to be monotonically non increasing once valid estimation has begun:

$$RUL_{\text{display}}(k) = \min\left(RUL_{\text{display}}(k-1),\ RUL(k)\right) \tag{10}$$

This stabilizer is applied only to the displayed output and does not modify the underlying damage trajectory, regression

fit, or projected failure point. k_{fail}. Its purpose is to suppress artificial upward fluctuations caused by local slope variability, which could otherwise reduce operator confidence.

The proposed RUL model is not intended to be universally optimal. Rather, it is a lightweight, mode gated local estimator designed for low latency industrial deployment and real-time edge execution.

4. EXPERIMENTAL SETUP

The proposed platform was deployed in an industrial factory setting to achieve two primary objectives: (1) to evaluate its feasibility and performance in a real-world production environment, and (2) to validate the RUL prediction algorithm using data collected under real-time operating conditions. The overall experimental setup schematic is shown in Fig6, which summarizes the shop floor deployment used for tool wear monitoring and real-time data acquisition. To support this evaluation, data were collected from two categories of tools: tools exhibiting rapid failure and tools with longer degradation times.

The industrial study is presented as a deployment feasibility case study rather than a statistically exhaustive validation across many machines.

4.1. Implementation

The proposed framework was implemented on a Navtat industrial edge gateway (Fig 7) running a Linux operating system. The vibration sensor (piezoelectric triaxial accelerometer) is connected via an analog-to-digital converter. Data acquisition and processing are implemented in Python. MQTT was used for telemetry, and the backend server was hosted on AWS with dashboards displaying real-time data.

4.2. Industrial Machine and Sensor Setup

The industrial case study uses a Hwacheon Hi Tech 230CL CNC turning machine (Fig 8). A triaxial accelerometer is mounted on the spindle housing near the cutting region to capture process and structural vibrations. The sensor is sampled at 10 kHz. The machine time for one job (i.e. one workpiece) is approximately 6 minutes 30 seconds.

4.3. Tool and Process Description

The experiments involve three inserts corresponding to roughing, finishing, and grooving operations. The cutting conditions for these three inserts differed in terms of spindle speed (rpm), feed rate (f), and depth of cut (doc), as summarized in Table1. A total of 35 workpieces were processed in the reported industrial study. The machine's spindle bearings were healthy at the beginning of the experiment.

4.4. Data and Ground Truth Considerations

The acquired vibration streams were stored in Parquet file format. Each data set consists of individual 1-second vibration signal snapshots acquired at 1-minute intervals. Each file contains 10,000 data points. Tool T1 consists of 112 snapshots, which is significantly higher than the other two tools, since the insert did not wear out rapidly and therefore operated for a longer duration. Tool T2 contains 22 snapshots, while Tool T3 contains 35 snapshots. These represent the total number of snapshots collected during the complete operational period. The dataset is limited in size and should be interpreted accordingly. Ground truth labels were established through operator inspection and observed degradation and failure events.

The industrial study is valuable because it reflects real plant conditions, but it is not a large scale, repeated trial dataset. Therefore, the reported results should be interpreted as evidence of deployment feasibility and early operational usefulness rather than final proof of universal generalization.

5. RESULTS AND DISCUSSION

5.1. Tool-Wise Monitoring Behavior

Based on Fig 9a, The Health Index (HI) trend for Tool T1 shows an extended initial phase of near zero values, followed by a distinct step increase around snapshots 35–40, after which the HI stabilizes and increases gradually with a low slope. Despite this transition, the HI remains consistently below the degrading threshold (≈ 0.5), indicating that the tool does not enter a degradation regime. The visual inspection markers (green and red triangles) further support this observation, as all points after the transition are classified as not degraded (green triangles), confirming alignment between the model predictions and physical assessment. This behavior is characteristic of the Stable Normal mode, where the tool exhibits minor changes in operating conditions without sustained wear progression. The absence of threshold crossings and the agreement with inspection results suggest that the observed shift reflects a benign operational change rather than true degradation, indicating stable tool performance with no immediate maintenance requirement.

Based on Fig 9b, Tool T2 exhibits a clear and sustained increase in HI following an initial stable phase, reflecting continuous wear accumulation. The HI crosses the degrading threshold at snapshot S12, followed by rapid progression to the critical (S18) and failure (S19) thresholds within a short interval. This sharp escalation indicates an accelerated degradation process, characteristic of the Failure Tracking mode, where the system captures near failure dynamics. The relatively linear increase prior to threshold crossings supports reliable tracking of degradation, while the rapid transition between critical states highlights the severity of wear in later

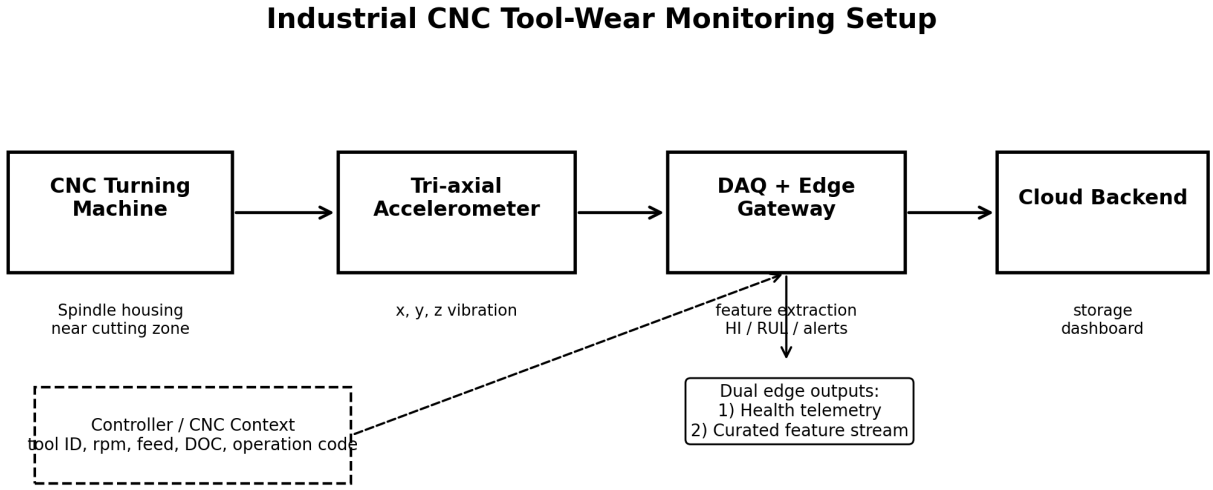


Figure 6. Experimental setup schematic

Table 1. Details of the tool wear experiments

Date	Tool ID	Tool Material	Operation Time for one workpiece	Parameters
13-02-2026	T1 ROUGHING	Carbide	4 min 30 sec	doc=0.8 mm; f=0.2; 1000 rpm
13-02-2026	T2 FINISHING	Cermet	40 sec	doc=0.3 mm; f=0.1; 2600 rpm
13-02-2026	T3 GROOVE	Carbide	1 min 20 sec	doc=0.5 mm; f=0.12; 2000 rpm

Table 2. Comparison between algorithm prediction and ground truth for Tool T2

Event	Snapshot	Time
Degradation detected (Analysis)	12	–
Critical alert generated (Analysis)	18	2:27 PM
Predicted failure time (Analysis)	19	2:37 PM
Actual failure observed (Ground Truth)	20	2:55 PM

stages. This pattern demonstrates the model's effectiveness in identifying imminent failure and providing timely warnings. Overall, T2 represents a fast degrading tool approaching end of life.

The comparison between the algorithm prediction and the ground truth observation for Tool 2 is summarized in Table 2. The results show that the algorithm predicted the imminent failure earlier than the observed failure event. Specifically, the system generated a critical alert approximately 28 minutes before the actual failure occurred, providing sufficient lead time for preventive maintenance actions.

Based on Fig 9c, The HI trend for Tool T3 shows a slow and steady increase over time, with very low slope and minimal variability. The degrading threshold is reached only at a late stage (around snapshot S30), and the HI remains significantly below critical and failure thresholds. This behavior aligns with the Drift Degrading mode, where wear progresses gradually without abrupt changes. The smooth and consistent increase indicates low intensity degradation, suggesting that the tool is wearing uniformly over time. The delayed threshold crossing reflects the system's sensitivity to subtle degradation while avoiding premature alerts. Overall, T3 demonstrates a slow degradation process with substantial remaining useful life.

Overall, the results demonstrate that the proposed health index based analysis can effectively detect degradation trends and provide early warnings of impending tool failure, supporting reliable predictive maintenance decisions. Experimental data and code will be made available on request.

5.2. Operational Usefulness

The primary operational benefit is the transition from periodic manual inspection to data driven monitoring. In the reported deployment, the operator's inspection effort is reduced from repeated physical inspection cycles to a short dashboard review, with physical intervention only when the framework indicates that the tool has crossed a wear threshold. For each

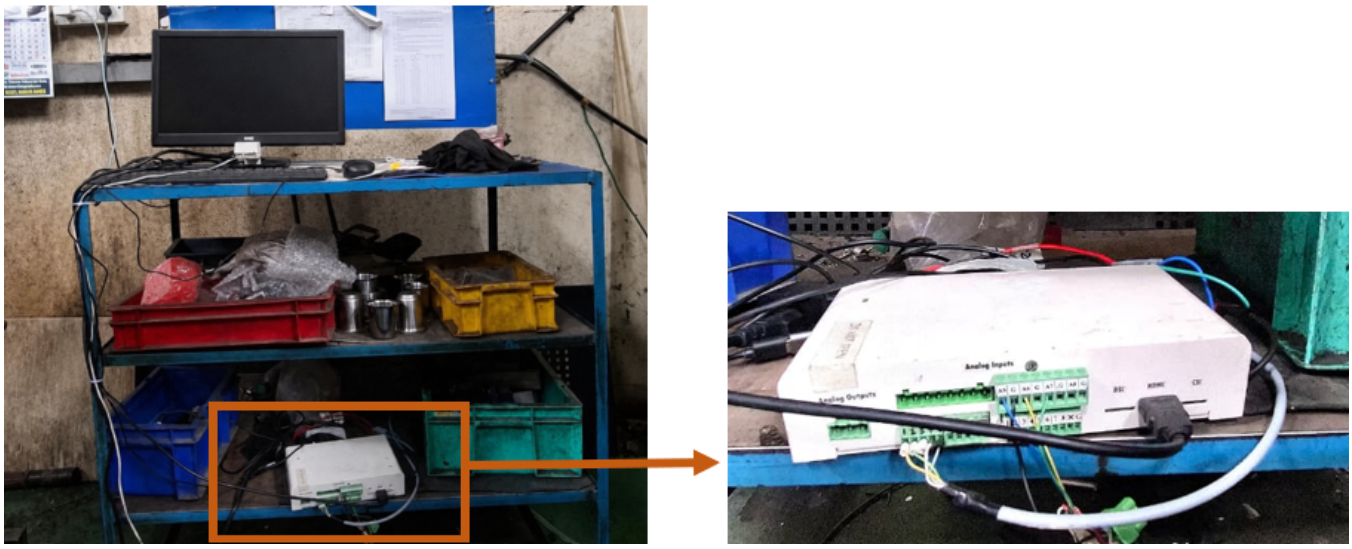
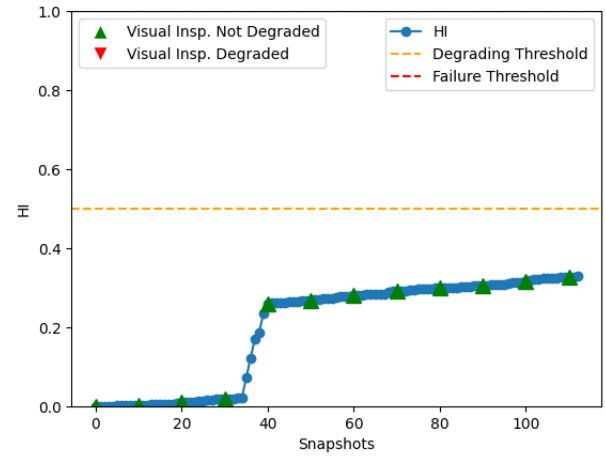
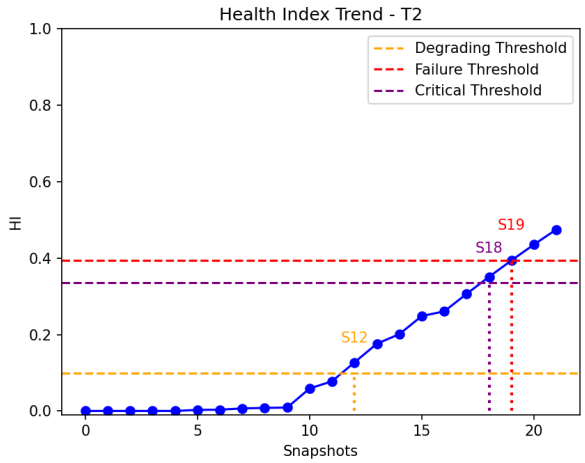


Figure 7. Gateway Setup

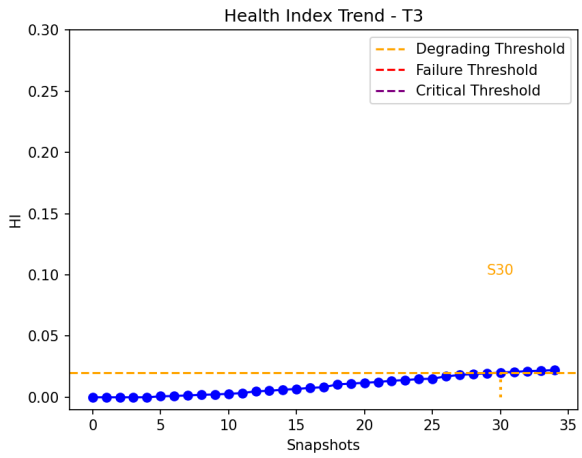
Figure 8. Hwacheon Hi Tech 230CL CNC turning machine



(a)



(b)



(c)

Figure 9. Health Index Evolution of the three tools (a) T1 (b) T2 (c) T3

tool, the dashboard indicates the different stages of wear, alerts if any, gives a bird's eye view of the overall fleet health and the estimated remaining life. This makes the system valuable even before full optimization of the prognostic model because it reduces unnecessary inspections and improves awareness of rapidly degrading tools.

6. EVALUATION

The platform is assessed using indicative benchmarking against baseline learning models on the collected industrial data as well as on open source data.

6.1. Evaluation on the collected dataset

We compare the proposed framework against two reference baselines: Support Vector Regression (SVR) using RMS centered handcrafted features, and an LSTM based deep learning baseline. These comparisons are intended as indicative baselines rather than exhaustive state of the art benchmarking.

These are the metrics used to evaluate performance: root mean square error (RMSE) for prognostic estimation quality where applicable, monotonicity of the predicted RUL trajectory as an indicator of operator facing stability and inference latency as an edge deployment suitability metric.

The LSTM model contained one LSTM layer with 50 units and ReLU activation, followed by a Dense regression output layer. The training utilized Adams optimizer with MSE loss for 30 epochs. The SVR model uses RBF kernel. Refer to Table 6 in the Appendix for more details.

6.1.1. Train–test protocol and threshold handling

The data splits are reported for reproducibility. We use a chronological 50/50 split: SVR and LSTM are fitted on the first half of the snapshots and evaluated on the second half. The proposed framework processes all snapshots sequentially: baseline learning uses the first N=10 snapshots per tool to estimate parameters such as mean, diagonal covariance and wear score; the remaining snapshots of the first half (approximately 7–8 snapshots for the industrial tools) are used for mode and threshold stabilization, after which the framework is evaluated on the second half of the snapshots (i.e. the same data as the reference baseline approaches).

The mode and threshold values derived from the baseline are held for the evaluation segment. The optional online threshold adaptation described in Section 3.5.4 was disabled during the test segment so that the comparison with the frozen baselines is symmetric. The framework's output during the evaluation segment therefore depends only on baseline statistics from the first half and on the test segment HI/damage trajectory itself, with no use of test segment failure labels at any stage. With approximately 17–18 snapshots per tool in the test segment, statistical confidence is limited, and the LSTM

in particular is operating well below the data scale at which deep recurrent models typically excel. The intent of including LSTM is not to claim a definitive comparison against deep learning, but to indicate the behavior of standard data hungry baselines under realistic small data industrial conditions.

6.1.2. A note on the values for T1 in Table 3

RUL is issued only when the slope based mode classifier places the tool in Drift Degrading or Failure Tracking; in Stable Normal Mode, the threshold values of $HIdeg = 1.0$ and $HIfail = 2.0$ are sentinel levels that suppress prognostic output. For T1, which is reported as remaining in a stable operating regime through the observed snapshots, RUL is not issued. To enable a head to head comparison with SVR and LSTM, which produce a regression output unconditionally, the RUL estimator was force evaluated for T1 by computing a local linear fit on the damage trajectory and projecting against a synthetic failure level, irrespective of mode. The T1 results therefore characterizes the predictor under a benchmark protocol rather than the deployed system; in operational use, the T1 RUL value would have remained suppressed, which is the framework's intended behavior in stable regimes.

6.1.3. RMSE

The proposed approach achieves the lowest RMSE across all tools, indicating a substantial improvement in prognostic accuracy. For T1, RMSE drops from 58.58 (SVR) and 31.42 (LSTM) to 16.48. Similarly, for T2 and T3, the proposed model reduces RMSE to 3.62 and 1.00 respectively, outperforming both baselines by a wide margin. This consistent reduction suggests that the model generalizes effectively across varying degradation patterns.

6.1.4. Monotonicity

The proposed framework reports a monotonicity score of 1.00 across all three cases. As stated in Section 3.5.5, the displayed RUL is constrained by equation 10, which enforces non increasing behavior at output time without altering the underlying linear fit. The SVR and LSTM baselines do not include such a stabilizer, so their lower monotonicity values reflect both the noise of their underlying regressors and the absence of an operator facing smoothing constraint. The monotonicity comparison should therefore be read as a deployability property of the proposed system rather than as evidence that its underlying predictor is intrinsically more monotonic.

6.1.5. Ground truth and per tool evidentiary weight

The RMSE values in Table 3 are computed against a synthetic linear countdown ground truth, applied uniformly to all three industrial tools. This convention is standard for prognostic baselines on run to failure data. It does not, however, mean that the RMSE values of each tool carry equal evidentiary weight. T2 reached an observed failure event during the experiment (Table 2), so its synthetic ground truth coincides at the final snapshot with a real end of life event, and its RMSE therefore reflects predictive accuracy against a physically grounded failure point. T3 exhibited gradual low intensity degradation but did not reach end of life within the reported run, and T1 remained predominantly in the stable regime; for these tools the synthetic countdown is a benchmark convention rather than a physical failure trajectory, and their RMSE values should be read as a measure of how closely each predictor tracks the convention, not as a measure of failure time accuracy. T2 therefore carries the strongest evidentiary weight, while T1 and T3 contribute predominantly to the latency, monotonicity, and ranking consistency story across heterogeneous tool behavior.

6.1.6. System level comparison

The three approaches in Table 3 differ not only in their predictors but also in their feature inputs and their output stabilization. SVR uses RMS centered handcrafted features, LSTM uses windowed feature sequences, and the proposed framework uses time, frequency, and wavelet features routed through the context aware fusion layer and processed through the covariance weighted wear evidence and adaptive damage accumulation. The asymmetric feature inputs across the three models are intentional and reflect typical practice: SVR and LSTM are reported with the simple, widely used RMS feature families that are common in the prognostic-baseline literature, while the proposed framework uses the richer feature set on which its design depends.

Latency results reveal a critical advantage of the proposed model for edge deployment scenarios. While SVR exhibits minimal latency (0.07 ms), its poor accuracy limits practical utility. Conversely, the LSTM model incurs significantly higher latency, ranging from 28.10 ms to 38.68 ms, which may be prohibitive for real-time or resource constrained applications. The proposed approach achieves a strong compromise, maintaining low latency (2.91–3.72 ms) while delivering superior accuracy and stability.

The reported numbers therefore characterize the end to end system, including context routing, mode gated prognostics, and monotonic stabilization, rather than the bare predictive capacity of an isolated regression model. The framework's contribution is positioned at this system level: it provides operationally usable, mode gated RUL with edge viable latency in a single sensor, multi tool, small data industrial setting where standard ML pipelines fitted to RMS features or trained without sufficient data do not perform competitively.

6.2. Evaluation on the UC Berkeley Milling Dataset

To validate the framework on open source data, it was run on the UC Berkeley Milling Dataset (Case 1) (Agogino &

Table 3. Comparative prognostic performance on collected data

Tool ID	Algorithm	RMSE (snapshots)	Monotonicity	Latency (ms)
T1	Baseline SVR	58.58	0.54	0.07
T1	Deep DL (LSTM)	31.42	0.36	28.10
T1	**Proposed**	**16.48**	**1.00**	**2.98**
T2	Baseline SVR	11.35	0.60	0.07
T2	Deep DL (LSTM)	5.16	0.22	35.03
T2	**Proposed**	**3.62**	**1.00**	**2.91**
T3	Baseline SVR	18.15	0.47	0.07
T3	Deep DL (LSTM)	8.65	0.47	38.68
T3	**Proposed**	**1.00**	**1.00**	**3.72**

Goebel, 2007). The dataset represents a rapid failure scenario where the cutting insert reaches its physical failure threshold in exactly 17 cutting cycles. The framework was evaluated against the same baselines as above: a Support Vector Regression (SVR) model utilizing time domain RMS features, and a Long Short Term Memory (LSTM) deep learning network. The evaluation focused on three core metrics: Root Mean Square Error (RMSE) for predictive accuracy, Monotonicity for RUL stability, and Inference Latency for edge viability. The first 3 cycles of the milling data were used for baseline calibration of the proposed agent; a 50/50 chronological split was applied in the case of SVR/LSTM. Refer to Table 6 in the Appendix for more details. The results of the proposed framework are shown in Table 4 and are discussed below.

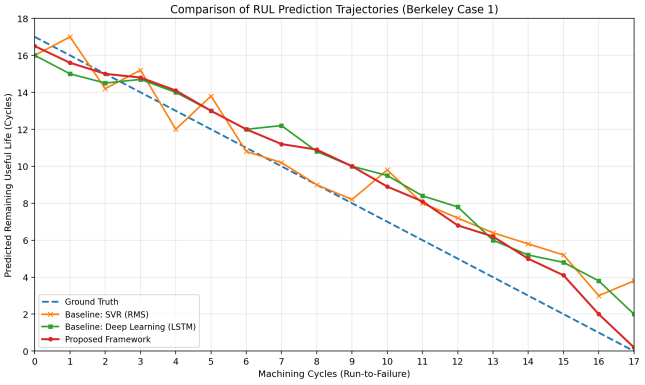


Figure 10. RUL predictions across different baselines (Berkeley Case 1).

Table 4. Comparative prognostic performance on the Milling dataset

Model Architecture	RMSE	Monotonicity	Latency
Baseline SVR (RMS Features)	5.99	0.25	0.08
Deep Learning (LSTM Network)	4.23	0.50	41.81
Proposed Framework	**3.18**	**1.00**	**3.66**

6.2.1. Discussion of results

The milling dataset reproduces the same ranking observed on the in house data: the proposed framework achieves the lowest RMSE (3.18 cycles) and edge viable latency (3.66 ms), with the standard ML baselines trailing on both metrics. Fig 10 compares the RUL trajectories predicted by the proposed framework against the ground truth and two benchmark models on the UC Berkeley Milling Dataset (Case 1). The proposed framework closely follows the true degradation trend and maintains a smooth, strictly monotonic decline throughout the 17-cycle run-to-failure sequence, whereas the SVR and LSTM models exhibit larger deviations and noticeable prediction fluctuations. While a single public dataset is not sufficient to establish universal generalization, this agreement between an in house industrial study and an independent benchmark strengthens the claim that the framework's behavior is not an artifact of a single deployment.

7. PRACTICAL RELEVANCE AND BUSINESS CONTEXT

For industrial adoption, technical accuracy alone is insufficient; the true value of the framework lies in its measurable operational impact. Practical metrics such as reduced manual inspection time, minimized unplanned downtime, fewer catastrophic tool breakage events, improved tool utilization, and lower scrap risk (Table 5) collectively define the business value proposition of the proposed system and should be systematically evaluated in a larger pilot deployment. By enabling a transition from reactive or schedule-based maintenance to proactive, condition-driven maintenance, the IntelliMaint framework supports more efficient maintenance planning and reduced production disruption. Furthermore, its ability to generate explainable and uncertainty-aware prognostic outputs, allows maintenance personnel and operators to make informed, risk-aware operational decisions with greater confidence.

8. LIMITATIONS

The current study has several limitations that should be considered when interpreting the results. First, the industrial case study is limited in scale, involving only three cutting tools

Table 5. ROI metrics for industrial pilot evaluation

Metric
Avg unplanned downtime/month
Cost per downtime hour
Tool replacement frequency per month
Avg tool cost
Tool breakage events per month
Avg failure repair cost
Preventive maintenance frequency per month
Maintenance cost per visit
Production rate in parts/hour
Scrap rate
Scrap cost per part

and 35 workpieces, which constrains the statistical strength and generalizability of the findings. Second, the effectiveness of the framework depends on the availability and accuracy of controller side contextual information, such as tool identity and cutting parameters; missing or erroneous context can reduce the accuracy of feature routing and tool specific monitoring. Third, although the single sensor design offers a cost effective and practical deployment approach, it introduces a signal superposition challenge in which vibration responses from multiple tools and machine components are combined, requiring accurate context-aware routing rather than direct physical isolation. In addition, the current RUL estimation approach is based on a lightweight local linear model chosen for explainability and edge compatibility, but it may not fully capture highly nonlinear degradation patterns, particularly during late stage wear. Finally, while the comparative benchmarking results are encouraging, the evaluation is not sufficiently broad to establish definitive superiority over all existing PHM and prognostic approaches.

9. FUTURE WORK

Future work will focus on expanding both the scope and capability of the proposed framework. A primary priority is to enlarge the industrial dataset to include a greater number of tools, repeated run-to-failure experiments, and deployments across multiple CNC machines in order to strengthen statistical validity and assess generalization under diverse operating conditions. The framework will also be enhanced by incorporating richer contextual information, such as part family identifiers, machining stages, and operation segment metadata, to further improve the accuracy of context-aware feature routing and tool specific interpretation. In addition, more advanced prognostic approaches, including nonlinear and probabilistic RUL models, will be investigated while maintaining the computational efficiency and low-latency requirements necessary for edge deployment. Beyond cutting-tool monitoring, the architecture will be extended to support simultaneous health assessment of other critical machine subsystems, including spindle bearings, gearboxes, and drive components, thereby enabling a more comprehensive machine level PHM solution. Finally, future industrial pilots will quantify the business impact of the framework by measuring key return-on-investment metrics such as reduced downtime, lower manual inspection effort, improved tool utilization, decreased scrap rates, and overall maintenance cost savings.

10. CONCLUSION

This paper presented a revised context aware edge cloud multi agent PHM framework for multi tool CNC turning machines using a single vibration sensor. The framework addresses the central deployment challenge of shared sensor monitoring by using controller context to route vibration derived features into tool specific logical agents. Each tool agent maintains its own baseline, health state, monitoring mode, alerts, and optional RUL trajectory. The revised manuscript clarifies the role of context, the meaning of the multi agent formulation, the construction of the HI, and the rationale for the lightweight edge compatible RUL model. The industrial case study demonstrates that the approach can isolate different tool behaviors, support early warnings for rapidly degrading tools, and reduce reliance on frequent manual inspection. Although further large scale validation is required, the results indicate that the framework is a practical and scalable foundation for deployable CNC PHM.

ACKNOWLEDGMENT

The authors thank Navtat Technologies for providing the industrial edge gateway and IntelliPredikt Technologies for providing AWS cloud resources. The authors also thank the operators and engineers who assisted with the CNC turning experiments at Hoshi Tools Pvt Ltd.

REFERENCES

Abd Elhaleem, S., Zanfal, A., & Hamdy, M. (2025, July). Predictive maintenance based on iiot and machine learning aligned with industry 4.0: a case study in waste-water treatment plant. *Neural Comput. Appl.*, *37*(24), 20383–20407. Retrieved from https://doi.org/10.1007/s00521-025-11463-4 doi: 10.1007/s00521-025-11463-4

Agogino, A., & Goebel, K. (2007). Milling data set. nasa ames prognostics data repository. *Moffett Field, CA*.

Calabrese, F., Regattieri, A., Bortolini, M., Gamberi, M., & Pilati, F. (2021). Predictive maintenance: A novel framework for a data-driven, semi-supervised, and partially online prognostic health management application in industries. *Applied Sciences*, *11*(8). Retrieved from https://www.mdpi.com/2076-3417/11/8/3380 doi: 10.3390/app11083380

Costa, V. L. L., Eberhardt, B., Chen, J., & Roßmann, J. (2023). Towards predictive maintenance: an edge-based vibration monitoring sys-

tem in industry 4.0. *2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, 1430-1437. Retrieved from https://api.semanticscholar.org/CorpusID:258335252

Dan, L., & Mathew, J. (1990). Tool wear and failure monitoring techniques for turning—a review. *International Journal of Machine Tools and Manufacture*, *30*(4), 579-598. Retrieved from https://www.sciencedirect.com/science/article/pii/0890695590900098 doi: https://doi.org/10.1016/0890-6955(90)90009-8

Gaikwad, P., Mundhada, V., Nagre, H., & Patil, H. (2024, 05). Bearing fault detection using vibration analysis. *International Journal for Research in Applied Science and Engineering Technology*, *12*, 4644-4653. doi: 10.22214/ijraset.2024.62569

Hassan, I. U., Panduru, K., & Walsh, J. (2024). An in-depth study of vibration sensors for condition monitoring. *Sensors*, *24*(3). Retrieved from https://www.mdpi.com/1424-8220/24/3/740 doi: 10.3390/s24030740

Huang, Z., Zhu, J., Lei, J., Li, X., & Tian, F. (2021). Tool wear monitoring with vibration signals based on short-time fourier transform and deep convolutional neural network in milling. *Mathematical Problems in Engineering*, *2021*(1), 9976939. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/9976939 doi: https://doi.org/10.1155/2021/9976939

Krishnamurthy, R., S, S., Sreedhar, R., & Chandrashekar, A. (2025, 10). Intellimaint: An intelligent component-agnostic framework for health indicator generation, and prognostics for electromechanical systems. *Annual Conference of the PHM Society*, *17*. doi: 10.36001/phmconf.2025.v17i1.4380

Li, X. (2002). A brief review: acoustic emission method for tool wear monitoring during turning. *International Journal of Machine Tools and Manufacture*, *42*(2), 157-165. Retrieved from https://www.sciencedirect.com/science/article/pii/S0890695501001080 doi: https://doi.org/10.1016/S0890-6955(01)00108-0

Li, Y., Meng, X., Zhang, Z., & Song, G. (2020). A machining state-based approach to tool remaining useful life adaptive prediction. *Sensors*, *20*(23). Retrieved from https://www.mdpi.com/1424-8220/20/23/6975 doi: 10.3390/s20236975

Munaro, R., Attanasio, A., & Del Prete, A. (2023). Tool wear monitoring with artificial intelligence methods: A review. *Journal of Manufacturing and Materials Processing*, *7*(4). Retrieved from https://www.mdpi.com/2504-4494/7/4/129 doi: 10.3390/jmmp7040129

Resende, C., Folgado, D., Oliveira, J., Franco, B., Moreira, W., Oliveira-Jr, A., ... Carvalho, R. (2021). Tip4.0: Industrial internet of things platform for predictive maintenance. *Sensors*, *21*(14). Retrieved from https://www.mdpi.com/1424-8220/21/14/4676 doi: 10.3390/s21144676

Salvador Palau, A., & Dhada, M. (2019, 12). Multi-agent system architectures for collaborative prognostics. *Journal of Intelligent Manufacturing*, *30*. doi: 10.1007/s10845-019-01478-9

Wang, K., Wang, A., Wu, L., & Xie, G. (2024). Machine tool wear prediction technology based on multi-sensor information fusion. *Sensors (Basel, Switzerland)*, *24*. Retrieved from https://api.semanticscholar.org/CorpusID:269311267

APPENDIX

This appendix documents the experimental configuration used in the Evaluation, for reproducibility (Table 6).

Table 6. Benchmark protocol and model settings

Item	Value
Data split — industrial production data	50% chronological train (SVR/LSTM) and baseline calibration (proposed); 50% chronological test for all three models. The proposed agent processes all cycles sequentially and is scored on the second half ($\approx$ 17–18 cycles per tool).
Data split — UC Berkeley milling	First 3 cycles used for baseline calibration of the proposed agent; 50/50 chronological split applied to the sequence dataset for SVR/LSTM (Berkeley Case 1, 18 cycle dataset).
Ground truth	Synthetic linear countdown $RUL(k) = total\_cycles - k - 1$, applied uniformly to all three industrial tools and to the Berkeley case. RMSE is computed against this synthetic ground truth on the test segment.
SVR kernel and parameters	SVR(kernel='rbf', C=100, gamma=0.1); epsilon left at default (0.1). Inputs standardized via StandardScaler.
SVR input features	Per cycle 3 axis RMS: $[\sqrt{mean(x)}, \sqrt{mean(y)}, \sqrt{mean(z)}]$
LSTM architecture	One LSTM layer with 50 units and ReLU activation, followed by a Dense regression output layer. Sequence length $min(3, \lfloor N/4 \rfloor)$, which is 3 for the 35 cycle industrial datasets.
LSTM training	Adam optimizer (default learning rate 0.001), MSE loss, 30 epochs, default batch size, no validation split, no early stopping, no dropout, no fixed random seed.
LSTM input features	Windowed feature sequences.
Proposed baseline calibration	N=10 snapshots for the industrial production data (T1, T2, T3); first 3 cycles for the UC Berkeley milling case to accommodate its shorter run to failure dataset.
Proposed input features	Time, frequency, and wavelet features routed through the context aware fusion layer and processed by the covariance weighted wear evidence pipeline.
Benchmark Comparisons	The proposed agent force evaluates RUL irrespective of the slope based mode classifier so that the predictor can be compared head to head with SVR and LSTM (which do not support mode gated suppression). In deployed operation, RUL would be issued only in Drift Degrading and Failure Tracking modes.
Proposed RUL post processing	3 sample moving average over valid RUL predictions, followed by the monotonic stabilizer $RUL\_display(k) = min(RUL\_display(k-1), RUL(k))$.
Online threshold adaptation	Disabled during the test segment so that the comparison with the frozen SVR/LSTM baselines is symmetric.
Reproducibility note	Reported numbers reflect a single run and may vary on re-execution within a few percent on RMSE and within 0.1 on the monotonicity score.
Evaluation metrics	RMSE in cycles (lower is better), monotonicity score (higher is better), and per-window inference latency in milliseconds.