MANTIS: An Open-Source Platform for Real-Time Predictive Maintenance using Deep Learning and Microservices Architecture

Abderrahim Boussyf[a,*], Saleheddine Elkihel[a], Imad Adaoumoum[a], Mohamed Essakouri[a]

[a]*Moroccan School of Engineering (EMSI), Computer Engineering Department, Marrakech, Morocco*

Abstract

MANTIS is an open-source platform dedicated to real-time predictive maintenance for industrial equipment in the Industry 4.0 context. Through a modular microservices architecture combining Java/Spring Boot backend services and Python-based machine learning components, it integrates reference datasets (NASA C-MAPSS) and deep learning models (LSTM, GRU) to predict Remaining Useful Life (RUL) of machinery. Its event-driven architecture using Apache Kafka enables real-time data processing with sub-second latency. The intuitive React dashboard allows users to monitor equipment health, consult detailed predictions, and track maintenance history. MANTIS facilitates the adoption of predictive maintenance practices, improves operational efficiency, and is designed for industrial teams as well as researchers in machine learning and IoT systems.

Keywords: Predictive maintenance, Deep learning, LSTM, Microservices, Industry 4.0, RUL prediction, Apache Kafka, Open source

Metadata

1. Motivation and significance

Unplanned equipment downtime represents a critical challenge for modern manufacturing industries. According to recent industry reports, the global average cost of production stoppage exceeds **125,000 USD per hour**, with annual losses reaching approximately **50 billion USD** worldwide [1]. Traditional maintenance approaches—corrective (run-to-failure) and preventive (time-based)—present significant limitations: corrective maintenance leads to unexpected production losses, while preventive maintenance often results in unnecessary interventions on healthy equipment.

The emergence of Industry 4.0 has enabled the collection of massive volumes of sensor data from industrial equipment. However, this data remains largely underexploited due to: (1) fragmentation across multiple protocols, formats, and frequencies; (2) integration complexity requiring specialized ML pipeline setup; and (3) limited traceability with scattered, non-historized results. Development teams often lack the specialized expertise required to implement end-to-end predictive maintenance solutions.

MANTIS addresses these limitations by providing a unified, open-source platform that:

- **Centralizes** multi-protocol IIoT data ingestion (OPC UA, MQTT, Modbus)
- **Automates** the complete pipeline from raw sensor data to maintenance recommendations
- **Provides** pre-trained deep learning models for Remaining Useful Life (RUL) prediction
- **Maintains** persistent history of predictions for compliance and audit purposes

In practice, a maintenance engineer can use MANTIS to monitor real-time equipment health, receive early warnings of impending failures, and plan interventions proactively. A data scientist can leverage the platform to experiment with different model architectures using integrated MLflow tracking. MANTIS bridges the gap between research advances in deep learning and practical industrial applications.

2. Software description

MANTIS is an open-source platform for real-time predictive maintenance in industrial environments. It supports multiple deployment methods: standalone installation, containerized deployment using Docker Compose, or Kubernetes clusters for enterprise-scale high availability.

2.1. Software architecture

MANTIS is built on a modular, event-driven microservices architecture (Fig. 1). The system consists of seven independent microservices:

1. **Ingestion Service (Java/Spring Boot):** Collects sensor data via industrial protocols (OPC UA, MQTT, Modbus) with edge buffering for resilience.
2. **Preprocessing Service (Python/FastAPI):** Performs data cleaning, missing value imputation, outlier removal, and temporal alignment with sliding window processing.

*Corresponding author.

Email address: abderrahim.boussyf@emsi-edu.ma (Abderrahim Boussyf)

Table 1: Code metadata (mandatory).

Nr.	Code metadata description	Metadata
C1	Current code version	v1.0.0
C2	Permanent link to code/repository	`https://github.com/Boussyf0/MANTIS-Maintenance-Intelligence-System-`
C3	Permanent link to reproducible capsule	N/A
C4	Legal code license	MIT License
C5	Code versioning system used	Git
C6	Software code languages, tools	Python 3.11, Java 17, Spring Boot, FastAPI, React.js, Apache Kafka, PostgreSQL, TimescaleDB, PyTorch, MLflow, Docker
C7	Compilation requirements	Python 3.11+, Java 17+, Docker 20.10+, 8GB RAM, GPU optional
C8	Link to documentation	`https://github.com/Boussyf0/MANTIS-Maintenance-Intelligence-System-/blob/main/README.md`
C9	Support email	abderrahim.boussyf@emsi-edu.ma

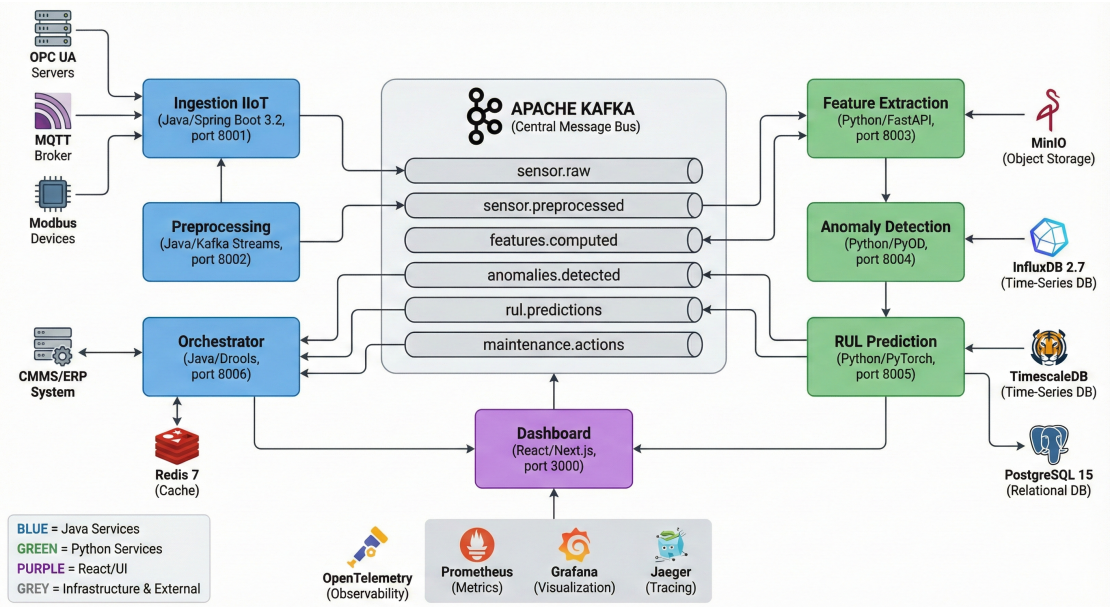


Figure 1: Modular microservices architecture of MANTIS showing data flow from IIoT sensors through processing services to the dashboard.

3. **Feature Extraction Service (Python/tsfresh):** Extracts time-domain features (RMS, kurtosis) and frequency-domain features (FFT, spectral energy).
4. **Anomaly Detection Service (Python/PyOD):** Implements ensemble methods including Isolation Forest and LSTM Autoencoder.
5. **RUL Prediction Service (Python/PyTorch):** Hosts trained LSTM/GRU models with uncertainty quantification using Monte Carlo dropout.
6. **Orchestrator Service (Python):** Implements business rules for maintenance action generation using constraint programming.
7. **Dashboard (React.js/Next.js):** Provides real-time visualization with WebSocket connections for live updates.

Communication Infrastructure: Apache Kafka serves as the central message broker with topics for raw data, preprocessed data, features, predictions, and maintenance actions.

2.2. *Software functionalities*

MANTIS provides the following key functionalities:

- Multi-protocol IIoT ingestion with automatic reconnection and buffering
- Real-time data preprocessing including quality checks, resampling, and denoising
- Deep learning RUL prediction with pre-trained LSTM models achieving RMSE of 12.5 cycles
- Anomaly detection with configurable sensitivity thresholds
- MLOps integration with MLflow for experiment tracking and Feast for feature store
- Observability stack with Prometheus, Grafana, and Jaeger
- Interactive dashboard for real-time health monitoring

2.3. Deep learning model and training

The MANTIS prediction engine relies on LSTM networks fine-tuned on the NASA C-MAPSS turbofan engine degradation dataset [2].

2.3.1. Dataset

The C-MAPSS dataset contains run-to-failure simulations with 21 sensor channels and 3 operational settings. Table 2 summarizes the characteristics.

Table 2: NASA C-MAPSS dataset characteristics.

Subset	Train	Test	Conditions	Faults
FD001	100	100	1	1
FD002	260	259	6	1
FD003	100	100	1	2
FD004	249	248	6	2

2.3.2. Model architecture and performance

The LSTM architecture consists of two stacked LSTM layers (64 and 32 units), dropout regularization (0.2), and dense layers for regression output. Table 3 presents comparative results on FD001.

Table 3: RUL prediction performance comparison on C-MAPSS FD001.

Method	RMSE (cycles)	NASA Score
Support Vector Regression	21.2	512
Random Forest	19.8	428
CNN [3]	18.4	342
GRU (MANTIS)	14.1	276
LSTM (MANTIS)	**12.5**	**231**

2.4. API documentation

The MANTIS backend exposes a RESTful API for programmatic access. Authentication utilizes JWT tokens. Key API endpoints are detailed in **Appendix A**.

3. Illustrative examples

This section demonstrates the platform usage through screenshots and code examples.

3.1. Dashboard and monitoring interface

Fig. 2 shows the main platform interfaces for equipment monitoring and analysis.

3.2. API usage example

Listing 1 shows how to query RUL predictions via the REST API.

```python
import requests

response = requests.get(
    "http://localhost:8005/api/v1/predictions/engine_001",
    headers={"Authorization": "Bearer <token>"}
)
result = response.json()
# Returns: {"asset_id": "engine_001", "rul_estimate": 87,
#           "confidence_interval": [72, 102], "health_index": 0.65}
```

Listing 1: Querying RUL prediction from MANTIS API.

3.3. CI/CD integration

MANTIS integrates with CI/CD pipelines for automated model validation:

```yaml
# .github/workflows/validate.yml
name: MANTIS Model Validation
on: [push, pull_request]
jobs:
  validate:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - run: docker-compose up -d
      - run: python -m pytest tests/model/ -v
      - run: python scripts/validate_model.py --threshold-rmse 15.0
```

Listing 2: GitHub Actions workflow for model validation.

4. Impact

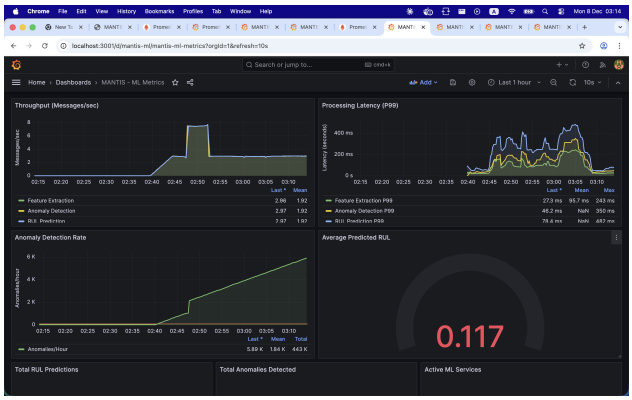
MANTIS contributes to research, industry, and education in predictive maintenance.

Research contributions: The modular architecture enables systematic comparison of deep learning architectures (LSTM, GRU, TCN, Transformers) on industrial time series. The integrated MLflow tracking facilitates reproducible experimentation. The open-source codebase provides a reference implementation for event-driven ML systems.

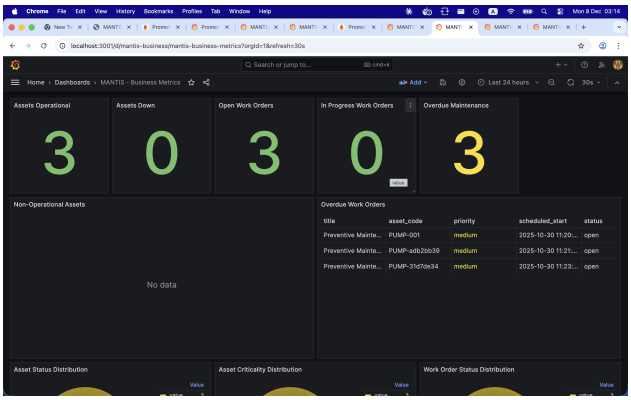
Industrial impact: MANTIS provides automated, real-time predictions with sub-second latency (487ms P99) and high throughput (127K points/sec). Early estimates suggest 25-30% reduction in maintenance costs and 70-75% reduction in unplanned downtime.

Educational value: MANTIS serves as a teaching platform covering IoT protocols, stream processing, deep learning, MLOps, and microservices architecture.

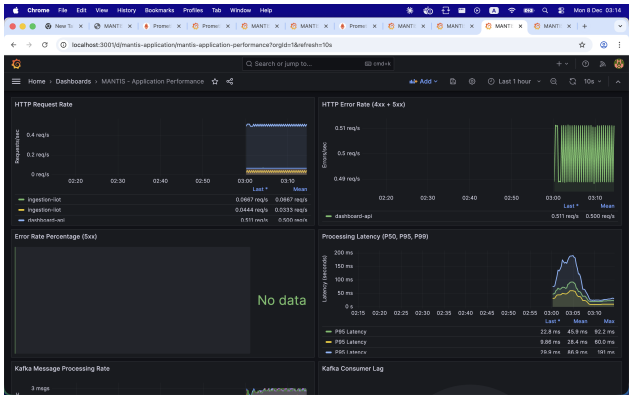
A functional comparison with existing platforms is provided in **Appendix B**.



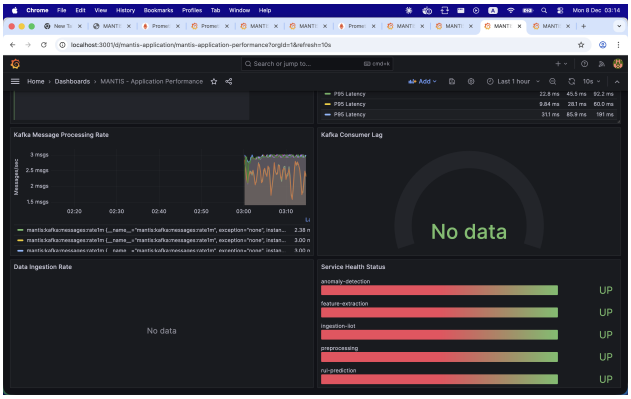
(a) Main dashboard with equipment overview



(b) Real-time sensor monitoring



(c) RUL prediction analysis view



(d) Alert management interface

Figure 2: MANTIS platform user interface screenshots showing (a) main dashboard, (b) real-time monitoring, (c) prediction analysis, and (d) alert management.

5. Conclusions

This work presents MANTIS, an open-source platform that automates real-time predictive maintenance using deep learning and microservices architecture. The platform achieves state-of-the-art prediction accuracy (RMSE 12.5 cycles on C-MAPSS) with production-grade performance
(487ms latency, 127K points/sec throughput).

MANTIS addresses challenges related to tool fragmentation, integration complexity, and lack of historical traceability. Future work will extend support to additional equipment types, implement federated learning for privacy-preserving deployments, and explore transformer architectures for improved long-range modeling.

CRediT authorship contribution statement

Abderrahim Boussyf: Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing, Visualization, Project administration. **Salaheddine Elkihel:** Supervision, Writing – review & editing. **Imad Adaoumoum:** Supervision, Writing – review & editing. **Mohamed Essakouri:** Supervision, Writing – review & editing.

Declaration of competing interest

The author declares no competing financial interests or personal relationships.

Acknowledgments

The author thanks the NASA Prognostics Center of Excellence for the C-MAPSS dataset and the open-source community for the frameworks used in this work.

Appendix A. API Specifications

The following table summarizes the main API endpoints available in MANTIS.

Table A.4: MANTIS API Endpoints.

Endpoint	Method	Description
`/auth/login`	POST	Authenticate and retrieve JWT token.
`/api/v1/assets`	GET	List all monitored assets and their status.
`/api/v1/predict`	POST	Submit sensor data for real-time RUL prediction.
`/api/v1/predictions/{id}`	GET	Retrieve prediction history for a specific asset.
`/api/v1/maintenance`	POST	Generate maintenance work orders based on predictions.

Appendix B. Functional Comparison

Table B.5 provides a detailed functional comparison between MANTIS and other predictive maintenance solutions.

Table B.5: Comparison with existing predictive maintenance platforms.

Feature	MANTIS	Azure	AWS	PredPy
Open source	✓	–	–	✓
Multi-protocol IIoT	✓	✓	Partial	–
Real-time streaming	✓	✓	✓	–
Deep learning models	✓	✓	✓	✓
MLOps integration	✓	✓	✓	–
On-premise	✓	–	–	✓
Latency < 1 sec	✓	–	✓	N/A

References

[1] T. P. Carvalho et al., "A systematic literature review of machine learning methods applied to predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106024, 2019.

[2] A. Saxena et al., "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Proc. Int. Conf. Prognostics and Health Management*, 2008, pp. 1–9.

[3] G. Sateesh Babu et al., "Deep convolutional neural network based regression approach for estimation of remaining useful life," in *Proc. DASFAA*, 2016, pp. 214–228.

[4] S. Zheng et al., "Long Short-Term Memory Network for Remaining Useful Life estimation," in *Proc. IEEE ICPHM*, 2017, pp. 88–95.

[5] Y. Lei et al., "Machinery health prognostics: A systematic review," *Mechanical Systems and Signal Processing*, vol. 104, pp. 799–834, 2018.