A Scalable Fog Computing Solution for Industrial Predictive Maintenance and Customization

Pietro D'Agostino *, Massimo Violante, Gianpaolo Macario

Posted Date: 30 October 2024

doi: 10.20944/preprints202410.2414.v1

Keywords: Fog Computing; Industrial IoT; Predictive Maintenance; LSTM; Platform Integration; Industrial Application

Article

A Scalable Fog Computing Solution for Industrial Predictive Maintenance and Customization

Pietro D'Agostino [1,*], Massimo Violante [1] and Gianpaolo Macario [2]

[1] DAUIN, Politecnico di Torino, Corso Castelfidardo, 34/d, 10138 Torino TO, Italia
[2] AROL Closure Systems, Viale Italia, 193, 14053 Canelli AT, Italia
* Correspondence: pietro.dagostino@polito.it

Abstract: This paper explores developing and deploying a fog computing platform tailored for industrial environments, focusing on predictive maintenance and advanced data management. The research investigates the integration of custom sensors and algorithms to meet specific industrial requirements, emphasizing the application of Long Short-Term Memory (LSTM) models for predicting equipment failures. The study enhances proactive maintenance strategies and real-time decision-making by applying machine learning techniques within a fog computing architecture. The findings underscore the platform's potential to advance industrial data processing, edge intelligence, and maintenance practices, contributing to improved efficiency and reduced downtime in industrial operations.

Keywords: fog computing; industrial IoT; predictive maintenance; LSTM, platform integration; industrial applications

1. Introduction

The escalation in the Internet of Things (IoT) devices and data-intensive applications has challenged industries to manage and utilize large volumes of data efficiently [1]. This increase in edge data production highlights the limitations of traditional cloud-centric computing, particularly the latency introduced by the distance between edge devices and cloud servers [2,3].

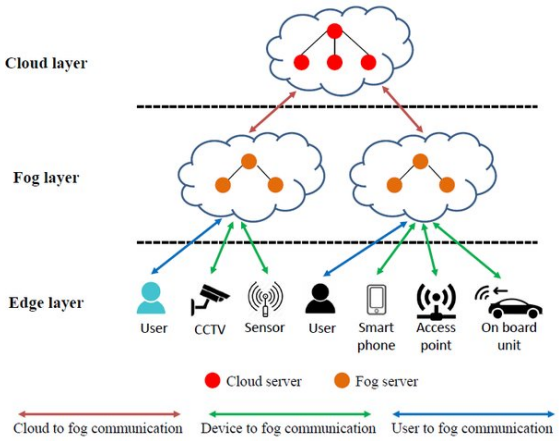

Figure 1. Fog computing architecture [4].

Fog computing addresses these challenges by extending cloud services to the peripherals, combining edge devices, fog nodes, and the cloud [5]. Fog nodes, positioned between field devices and the cloud, manage processing, storage, and content delivery, enhancing responsiveness and reducing latency [7–9]. This architecture accelerates data processing and improves privacy by minimizing data transfers over public networks [11]. Fog computing effectively combines cloud and edge strengths while addressing their limitations [12]. However, traditional services often lack customization for specific industrial needs. To overcome this, businesses are adopting mass customization [13].

 doi:10.20944/preprints202410.2414.v1

This paper introduces the Concentrator, a fog computing platform designed for industrial environments with numerous interconnected nodes. It centralizes data collection and allows extensive customization through Docker-based containerization, supporting real-time data processing and predictive maintenance [14,15].

Docker has proven effective in creating containers that run on the same kernel, allowing them to operate independently without interference unless specific permissions are granted. One important application of this technology is predictive maintenance, especially in industrial settings.

We demonstrate the application of this approach with AROL Closure Systems, a capping machine manufacturer that benefits from early issue detection through predictive maintenance. We use Long Short-Term Memory (LSTM) models to predict temperature thresholds and ensure timely maintenance [20]. LSTM models excel in time series forecasting by capturing long-term dependencies in sequential data, offering advantages over traditional models [21–23].

The main contributions of this paper are:

- Development and deployment of the Concentrator in a real industrial setting.
- Customization capabilities allowing integration of specific sensors and algorithms.
- Use of LSTM models for improved predictive maintenance and reduced downtime.

The paper is structured as follows: Section 2 reviews related work on fog computing and predictive maintenance. Section 3 outlines the design and implementation of the Concentrator system, with a focus on its customization and machine learning capabilities. Finally, Section 4 discusses the real-world deployment of the system and its performance in a production environment.

2. State of Art

This work primarily revolves around two key aspects: device customization and predictive maintenance functionalities.

Authors often pursue device customization by laying the groundwork for diverse application scenarios. For instance, Isaac Lera et al. [30] developed YAFS, a dynamic fog computing system enabling edge components to select the nearest and most secure link with various nodes, ensuring stable connections. Despite its utility for highly dynamic systems, YAFS does not add anything about the user's application, even though it can be useful for highly dynamic systems.

Cecil Wöbker et al. [31] present Fogernetes, a framework aimed at streamlining the deployment and management of fog computing applications. Fogernetes offers tailored features for fog environments, including edge node discovery, resource-aware scheduling, and dynamic adaptation to network conditions. While the authors successfully demonstrate the system's effectiveness, its adaptability is contingent upon the network; customers have limited scope for additional customization.

D. D'Alessandro et al. describe a similar project in [32], exploring mass customization in IoT wireless sensor systems. This study focuses on the need for flexible IoT solutions that adapt to different application requirements and environments. The authors present a modular approach to designing and developing IoT sensor systems, which allows for flexibility and scalability during deployment. Integrating modular components like sensors, communication modules, and processing units allows the system to be customized to meet specific user needs without requiring major redesigns or reconfigurations.

Unlike traditional methods that rely on hubs and separate radio modules, the Concentrator emphasizes embedding custom firmware directly into embedded systems used in production lines. This streamlined approach improves efficiency and reduces the need for additional hardware components unless required by the client. Additionally, by using Docker and containerization, as previously mentioned, it is possible to create adaptable environments tailored to each customer's needs. This method provides exceptional flexibility and customization options, making integrating into various environments and applications easy. This solution have also been adapted for use in

 doi:10.20944/preprints202410.2414.v1

microservices and IoT scenarios on embedded systems, such as a Raspberry Pi, an example of its usage is presented in [17], making them easily deployable in production lines.

Shifting the focus to predictive maintenance, [33] illustrates the critical significance of these methods for industries aiming to minimize downtime, cut costs, prolong equipment lifespan, enhance operational efficiency, and support safety measures. The authors employ charts to assess the Return on Investment (ROI) based on commonly used variables across various industries. However, they highlight a significant challenge: the indispensable requirement for high-quality data to ensure dependable predictions. They emphasize the crucial role of meticulous data preparation and cleaning in guaranteeing the reliability of predictive maintenance systems. This underscores the importance of data quality in deriving actionable insights and maximizing the effectiveness of predictive maintenance strategies.

Both [34] and [35] look into various machine learning methods, proving their usefulness for executing predictive maintenance on machines. However, their focus predominantly lies on classification techniques such as Support Vector Machine, Logistic Regression, K Nearest Neighbors, Random Forest, Gaussian Naive Bayes, Decision Tree, and Neural Network. In contrast, the Concentrator employs a different approach, leveraging Recurrent Neural Networks like Long Short-Term Memory (LSTM) to forecast the future behavior of measured parameters.

In [36], the authors illustrate an application of LSTM in estimating Exhaust Gas Temperature (EGT) within dynamic systems to monitor system deterioration. Their LSTM network architecture comprises 12 nodes in the input layer, 24 nodes in the hidden layer, and 1 node in the output layer. The model is trained using various batch sizes and validated with a real engine dataset encompassing steady and transient conditions. Although the proposed application and the models differ, the underlying objective of the algorithm remains consistent.

Several techniques can be employed to optimize LSTM models for these environments, for example:

- **Model Pruning and Quantization**: Reducing the size of the LSTM model through pruning unnecessary parameters and applying quantization techniques helps decrease the memory footprint and computational load, making the model more suitable for embedded systems [26].
- **Knowledge Distillation**: This involves training a smaller, less complex model (student model) to mimic the behavior of a larger, more complex LSTM model (teacher model). The smaller model can then be deployed on resource-constrained devices without significant loss of accuracy [27].
- **Hardware Acceleration**: Utilizing specialized hardware, such as Tensor Processing Units (TPUs) or Field-Programmable Gate Arrays (FPGAs), can significantly accelerate the computation of LSTM models on embedded devices, enabling real-time processing [28].
- **Edge AI Frameworks**: Leveraging edge AI frameworks like TensorFlow Lite or ONNX Runtime can assist in converting and optimizing LSTM models for execution on embedded systems, ensuring that they run efficiently while maintaining high accuracy [29].

By adopting these optimization strategies, industries can deploy LSTM-based predictive maintenance solutions effectively within embedded systems, ensuring that predictive maintenance remains robust and reliable even in environments with stringent resource constraints.

Compared to these articles, the LSTM model serves a dual purpose within the system. First, it is utilized for predictive maintenance, ensuring the machine operates correctly and alerting the user to potential issues if certain parameters deviate from expected values. Second, the model monitors the sensors' performance. Since the system is homogeneous, if predictions from one sensor diverge significantly from those of its neighboring sensors, it may indicate a problem with the measurement system itself. Therefore, the LSTM model aids in verifying both the machine's proper operation and the accuracy of the measurement system.

Numerous studies have attested to the superior performance of LSTM models compared to their counterparts, showcasing their ability to achieve remarkable accuracy and generalization on diverse datasets [24]. LSTM's adaptability to varying data complexities and ability to learn intricate patterns

 doi:10.20944/preprints202410.2414.v1

make it a favored choice for addressing real-world industry challenges. Its versatility and proven efficacy solidify LSTM's position as a key technology in deep learning [25].

While the LSTM model excels in predictive maintenance due to its ability to capture long-term dependencies, deploying such models on embedded systems poses unique challenges. Embedded systems, often characterized by limited computational power, memory, and energy resources, require efficient deployment of LSTM models to ensure real-time performance without exhausting system resources.

3. Methodology and Architecture

In this section, the entire system will be described, showing the architecture of the Concentrator, as seen in Figure 2 and its functionalities, as seen in Figure 3.

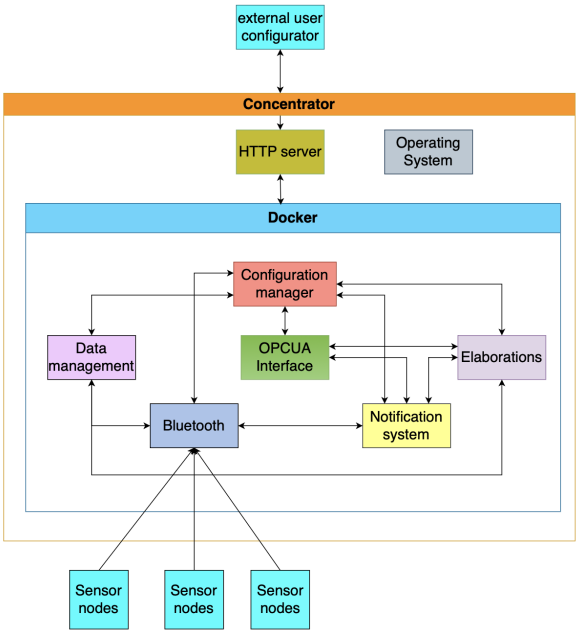

Figure 2. General architecture of the Concentrator.

The following schemes have been created using Capella, an open-source program using Arcadia as a model language to represent schematics.

The Concentrator system is a comprehensive framework for real-time data management and analysis within industrial environments. Essential functionalities, or Use Cases (UCs) include:

- UC.1: wireless data retrieval from sensors and devices
- UC.2: versatile handling of process data from both AROL and external systems
- UC.3: rigorous technical data processing for real-time monitoring and optimization
- UC.4: prompt anomaly detection with effective user notification
- UC.5: multi-device connectivity for holistic monitoring

These functionalities collectively empower the Concentrator to serve as a central data aggregation, processing, and dissemination hub, facilitating informed decision-making, predictive maintenance, and operational optimization. Moreover, with provisions for seamless integration of custom sensors and adapters, the Concentrator ensures interoperability across diverse industrial ecosystems.

3.1. Functional Requirements

The Concentrator system is tasked with various functionalities essential to operate within industrial contexts. Firstly, the system addresses the platform power management requirement, including startup, shutdown, and configuration procedures. The Concentrator then establishes robust

 doi:10.20944/preprints202410.2414.v1

connections with various external systems regarding system communication. Notably, the Concentrator is expected to communicate simultaneously with a maximum of 64 sensor node devices (usually mounted on a capping machine) while maintaining connectivity to additional devices as required. Moreover, to ensure future scalability and adaptability, compatibility with diverse IoT devices can be possible as long as the hardware limitations permit it.

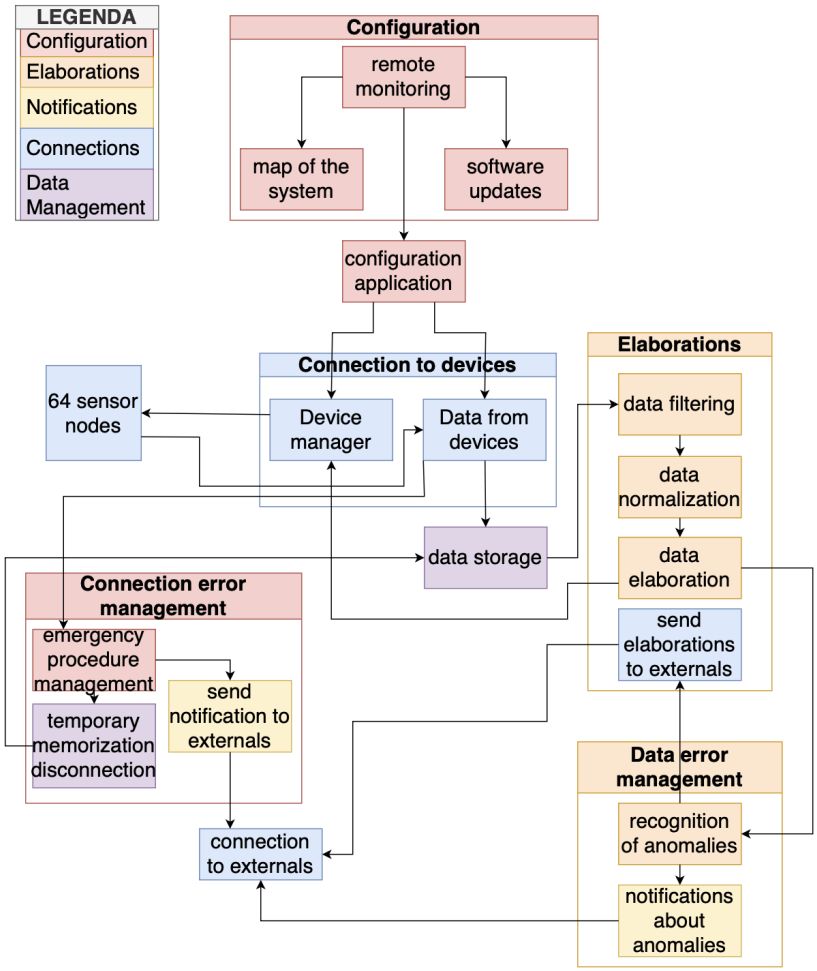

Figure 3. Functions of the Concentrator.

3.2. Protocols Requirements

The Concentrator system employs robust communication protocols tailored for seamless interaction with IoT and operational technology (OT) systems in industrial settings. For IoT device communication, it utilizes wireless connectivity, supporting protocols like Bluetooth Low Energy (BLE) version 4.0 and Wi-Fi. The Wi-Fi capability enables transmission using various protocols such as Message Queuing Telemetry Transport (MQTT), Hypertext Transfer Protocol (HTTP), Transmission Control Protocol/Internet Protocol (TCP/IP), and Constrained Application Protocol (CoAP) [37]. Notably, the platform adeptly manages multiple connections from IoT devices, handling connection establishment, maintenance, and termination while automatically reconnecting during temporary disruptions. While communication with OT systems is pending development, the Concentrator will interface with Modbus Transmission Control Protocol (Modbus/TCP) and (OPC Unified Architecture (OPCUA) protocols, facilitating data exchange with industrial automation devices such as Programmable Logic Controllers (PLCs) and Supervisory Control and Data Acquisition (SCADA) systems. Additionally, it features synchronization inputs for temporal alignment of devices or actions, ensuring precise coordination within industrial processes. Moreover, potential integration

 doi:10.20944/preprints202410.2414.v1

of Ethernet-based protocols may be necessary to accommodate client-specific devices, highlighting the Concentrator's adaptability to diverse industrial environments.

3.3. Data Processing

The Concentrator system plays a crucial role in enhancing the quality and utility of data received from IoT devices. Initially, it performs pre-processing on data from IoT devices to ensure compatibility and efficiency before transmitting it to external systems. It also filters data based on specific criteria, potentially removing invalid, out-of-range, or non-compliant data to maintain data integrity and relevance. Additionally, the system aggregates data from various IoT devices to offer a comprehensive overview of collected information, aiding informed decision-making. It can also apply calibrations or normalizations to ensure data aligns with external system standards, improving interoperability.

3.4. Error Management

The Concentrator is responsible for robust error detection and mitigation to maintain operational reliability and data integrity. It identifies errors and anomalies like connection failures, transmission errors, out-of-range data, or invalid values. Once detected, the system communicates these errors via messages, system logs, or other notifications for timely intervention and resolution. Additionally, it can manage critical errors through emergency procedures such as system shutdown or activating safety modes to prevent potential hazards. Moreover, the Concentrator supports temporary data storage during system disconnection, ensuring seamless data transmission upon reconnection.

3.5. Cybersecurity and Updates and Maintenance

The Concentrator system is tasked with maintaining data integrity and controlling access. Its primary function is to authenticate IoT devices and regulate data access to prevent unauthorized users from compromising the confidentiality and integrity of sensitive information. Additionally, the platform must support remote management and monitoring regarding system updates and maintenance. This capability enables smooth execution of updates and remote system monitoring, ensuring it stays current with the latest security patches and enhancements while enabling proactive maintenance and optimization.

4. Experiment and Discussion

The experiment consisted of two phases: the first was in a controlled laboratory setting, where 50 sensor nodes were connected to monitor and record the room's temperature. In the second phase, we used an actual machine and tested it in a thermal chamber instead of a real-world environment due to ongoing testing constraints.

The Nicla Sense ME has been used as a sensor node, and it exemplifies a trio of strengths: compact design, cost-effectiveness, and energy efficiency, while seamlessly integrating four state-of-the-art sensors engineered by Bosch Sensortec. The acronym "ME" stands for "Motion" and "Environment," highlighting its capability to accurately detect rotation, acceleration, temperature, humidity, pressure, air quality, and CO2 levels with industrial-grade precision. This latest version sets a new benchmark with its ultra-compact size, enabling effortless data transmission via Bluetooth Low Energy (BLE) 4.2, powered by the ANNA-B112 module. In the experiment, 50 Niclas were used, both in the laboratory and on the machine. Their primary role was to transmit temperature, humidity, and pressure data to the Concentrator using the wireless protocol.

The Raspberry Pi 4 Model B has been used as a Concentrator, the core of our embedded hardware system, operating on Raspberry Pi OS 64-bit, built on Debian 12 (Bookworm). This platform was our top choice for its robustness and versatility. To support virtualization and sandboxing for external users, we implemented Docker version 25.0.4 (build 1a576c5), with Docker Compose version 2.24.7 enabling efficient management of multiple containers concurrently. Given our reliance on Bluetooth for wireless communication, the BlueZ library version 5.66 was integrated into the system. The architecture

 doi:10.20944/preprints202410.2414.v1

is divided into two key segments: Native and Host. The Native segment represents the manufacturer's standard distribution providing essential system functionalities. It is represented in Figure 1. In contrast, the Host segment functions as a sandbox environment for general users, allowing them to upload their code and integrate additional sensors, enhancing the system's measurement capabilities. The overall experiment is represented in Figure 4. The Concentrator's primary function is to aggregate data from multiple peripherals. Utilizing a container built on Node.js version 16 and the open-source library node-ble, this system intelligently differentiates between Host and Native configurations based on peripheral addresses and settings. It establishes seamless connections with the targeted peripherals and carefully records the data streams from the Nicla Sense ME sensors. The acquisition frequency is calibrated to capture measurements at a rate of one per second. Once connections are established and the necessary data is acquired, the container publishes the raw data locally using a TCP server. The Native container employs the node-ble library and Bluetooth connections to broadcast native measurements. Each line of the data file includes the peripheral's MAC address, the measured characteristic's effective value, and the sensor it is using (internal or external, in case there is an external sensor attached).

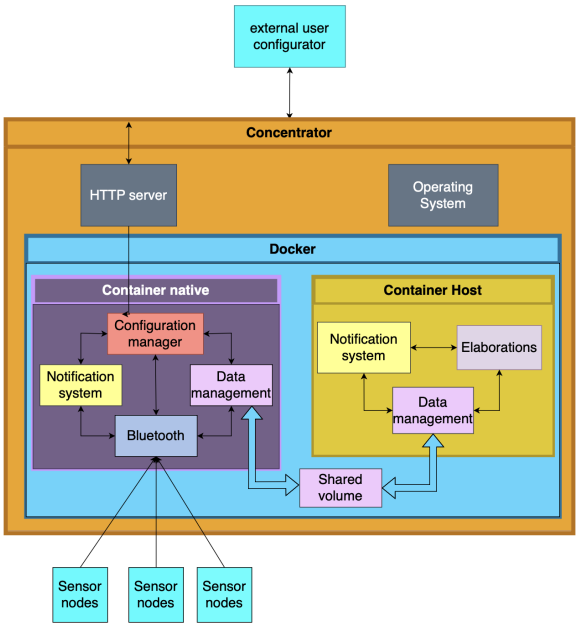

Figure 4. Experiment.

The Host container leverages an LSTM machine-learning model to predict sensor temperature measurements. Built on a dataset of 1,019,663 real measurements, the model was trained on 70% of the data, with the remaining 30% reserved for testing, using Python 3.11. Key libraries supporting this process include Numpy 1.26.3, Pandas 2.1.4, Ujson 5.4.0, TensorFlow 2.15.0, and sci-kit-learn 1.2.2. While TinyML was initially considered [38], a more conventional approach was chosen since the Raspberry Pi has ample memory and CPU capacity to run TensorFlow efficiently. The process begins by connecting to the TCP server and parsing the raw data file generated by the initial container, ensuring sufficient data is available for prediction. Once the data threshold is met, the system identifies outliers within the 20 to 43 degrees Celsius temperature range and any zero or NaN values. Next, the pre-trained model, named "model_j0700.keras," is loaded to evaluate the upcoming sequence of temperature measurements. The output includes the last recorded temperature in the file and the subsequent predicted temperature (e.g., temperature number 40 and the predicted temperature number 41). Predictions are generated using a sliding window of 20 values, updating with each new reading, to provide a single forecasted value. The model was trained on real AROL Closure System's J0700 machine data. It consists of three layers: one LSTM layer with 50 nodes and two Dense layers,

 doi:10.20944/preprints202410.2414.v1

with 25 nodes in the first and a single node in the second. The model achieves an RMSE of 0.1838, with a prediction time of approximately 200 ms on the embedded device. Here below the consumption of resources taken by the architecture:

Table 1. LSTM experiment data.

Container	Number of Nodes	Memory	CPU
None	0	570 MB	0%
Only Native	1	671 MB	10%
Only Native	50	700 MB	17%
Only Host	1	800 MB	10%
Only Host	50	1 GB	20%
Docker Compose	1	1,2 GB	35%
Docker Compose	50	1,4GB	55%

4.1. Controlled Laboratory Results

To check the behavior of the sensors, one of them, by chance, has been a little tossed around, resulting in a temperature change. This results in a change of its prediction without knowing which node has been altered. The analysis in Figure 5 demonstrates that minimal error metrics mark the system's predictive performance. The highest values for Mean Absolute Error (MAE) and Mean Squared Error (MSE) are noted for NODE_03 at 0.0340 and 0.0013, respectively. These values represent the greatest deviations among the nodes, yet they still indicate that even the node with the most significant errors remains within a relatively narrow accuracy range. In contrast, NODE_25 is identified as an anomaly within the dataset due to its intentional modification as a test case designed to assess the system's capability of checking the node's behavior. This deliberate alteration of NODE_25 provides insights into the system's behavior under abnormal conditions. The generally low error rates across most nodes highlight the effectiveness of the predictive maintenance system, underscoring its ability to produce accurate and reliable forecasts. The discrepancies observed, especially with NODE_25, illustrate how the Concentrator monitors and detects anomalies.

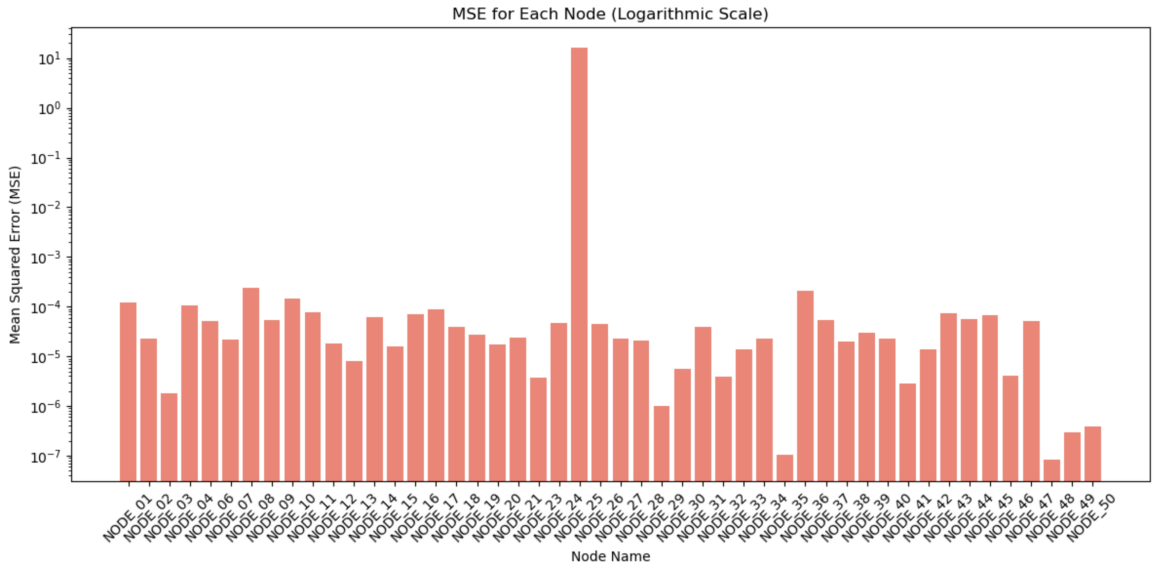

Figure 5. Mean Square Error of laboratory results (logarithmic scale).

4.2. Real Machine Results

In alignment with the laboratory results, the performance metrics obtained from the real machine data further affirm the system's efficiency, as depicted in Figure 6. No individual sensors were subject to modification or experimental alteration in the scenario. The consistency is reflected in the results,

 doi:10.20944/preprints202410.2414.v1

where all sensors exhibited comparable errors in predictive accuracy and recorded similar temperature measurements. The maximum error values were observed for NODE_08, which recorded a peak MAE of 0.0126 and an MSE of 0.0002. These values still indicate minimal deviation, underscoring the system's precision in predicting machine behavior. The absence of anomalies or sensor discrepancies in this test case resulted in uniformly low error rates, suggesting that the system operates optimally when all components function as expected. It effectively demonstrates the system's capability to maintain high levels of accuracy in the absence of external perturbations, further validating its reliability for practical applications in real-world environments.

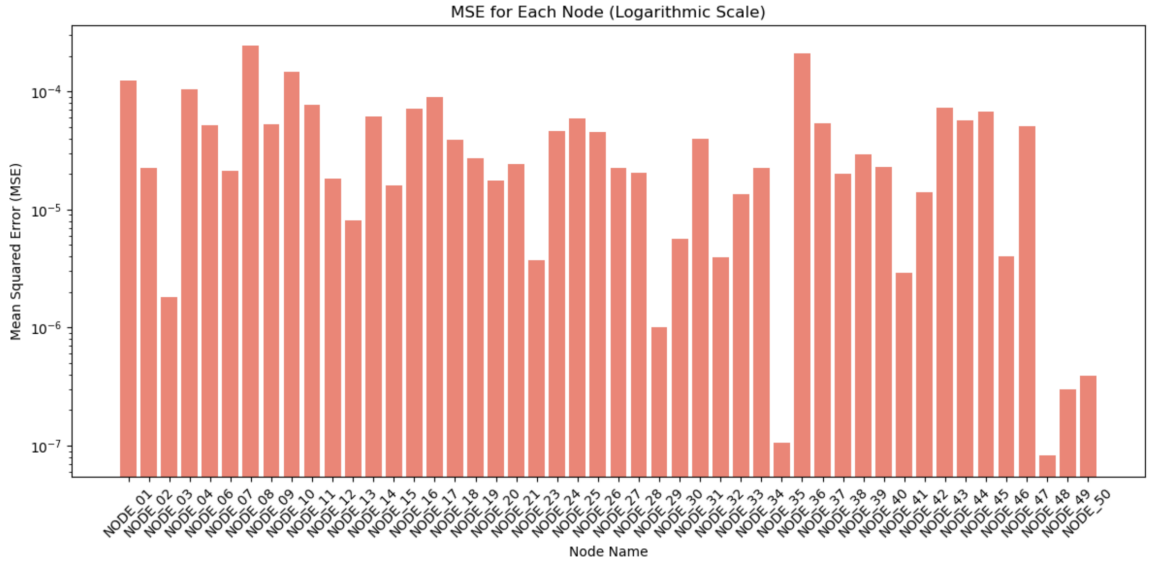

Figure 6. Mean Square Error of real machine results (logarithmic scale)

5. Conclusions

This study demonstrated the practicality of implementing a predictive maintenance system within a resource-constrained IoT framework. Utilizing Nicla Sense ME sensors and a Raspberry Pi-based Concentrator, the system successfully monitored, processed, and predicted real-time temperature measurements. Additionally, including a sandbox environment allows external users to customize the system by adding different codes or sensors without altering the core software, enhancing its adaptability for diverse applications.

Using an LSTM-based machine learning model on an embedded system showed that advanced AI techniques can be effectively applied even in environments with limited computational resources. The model's accurate predictions, with an RMSE of 0.1838, demonstrate the potential for these systems to be utilized in industrial settings to support proactive maintenance and reduce downtime.

Moreover, the architecture's scalability was confirmed, as the system maintained reliable performance across various connected devices. This scalability, combined with the successful integration of machine learning for predictive maintenance, underscores the system's potential for broader application in smart industrial environments.

Acknowledgments: The authors express their deepest gratitude to AROL Closure System for generously sharing their expertise and resources, which were instrumental in completing this project. Furthermore, they sincerely thank their colleagues, whose invaluable assistance greatly contributed to the development of this paper.

References

1. Saeed, A.; Khattak, M.A.K.; Rashid, S. Role of Big Data Analytics and Edge Computing in Modern IoT Applications: A Systematic Literature Review. *Proceedings of the 2022 2nd International Conference on Digital Futures and Transformative Technologies (ICoDT2)* **2022**, 1-5. doi:10.1109/ICoDT255437.2022.9787416.

2. Tianfield, H. Towards Edge-Cloud Computing. *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)* **2018**, 4883-4885. doi:10.1109/BigData.2018.8622052.
3. Rajanikanth, A. Fog Computing: Applications, Challenges, and Opportunities. *Journal of Autonomous Intelligence* **2023**, *24*, 5. doi:10.32629/jai.v5i2.545.
4. Weng, C.-Y.; et al. A Lightweight Anonymous Authentication and Secure Communication Scheme for Fog Computing Services. *IEEE Access* **2021**. doi:10.1109/ACCESS.2021.3123234.
5. Atlam, H.F.; Walters, R.J.; Wills, G.B. Fog Computing and the Internet of Things: A Review. *Big Data and Cognitive Computing* **2018**, *10*, 2. doi:10.3390/bdcc2020010.
6. Mouradian, C.; Naboulsi, D.; Yangui, S.; Glitho, R.; Morrow, M.; Polakos, P. A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges. *IEEE Communications Surveys & Tutorials* **2017**. doi:10.1109/COMST.2017.2771153.
7. Abdali, T.-A.N.; Hassan, R.; Aman, A.H.M.; Nguyen, Q.N. Fog Computing Advancement: Concept, Architecture, Applications, Advantages, and Open Issues. *IEEE Access* **2021**. doi:10.1109/ACCESS.2021.3081770.
8. Singh, J.; Singh, P.; Gill, S.S. Fog Computing: A Taxonomy, Systematic Review, Current Trends and Research Challenges. *Journal of Parallel and Distributed Computing* **2021**, *157*, 6. doi:10.1016/j.jpdc.2021.06.005.
9. Rasheed, M.; Saleem, J.; Hudabia, M.; Hafiz, T.; Mannan, R.; Mishaal, A. A Survey on Fog Computing in IoT. *VFAST Transactions on Software Engineering* **2021**, *9*, 4. doi:10.21015/vtse.v9i4.727.
10. Kumar, K.V.; Kumar, K.; Kumar, R.; Basha, S.M.; Praveen, M.; Reddy, P. Internet of Things and Fog Computing Applications in Intelligent Transportation Systems. In *IGI Global* **2019**, Chapter 008.
11. Mukherjee, M.; Matam, R.; Shu, L.; Maglaras, L.; Ferrag, M.A.; Choudhury, N.; Kumar, V. Security and Privacy in Fog Computing: Challenges. *IEEE Access* **2017**. doi:10.1109/ACCESS.2017.2749422.
12. Ahmed, K.; Zeebaree, S. Resource Allocation in Fog Computing: A Review. *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)* **2021**, *10*, 5. doi:10.5281/zenodo.4461876.
13. IEEE. IEEE Guide for General Requirements of Mass Customization. *IEEE Std 2672-2023* **2023**, 1-52. doi:10.1109/IEEESTD.2023.10210443.
14. d'Agostino, P.; Violante, M.; Macario, G. A User-Extensible Solution for Deploying Fog Computing in Industrial Applications. *Proceedings of the 2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE)* **2023**, 1-6. doi:10.1109/ISIE51358.2023.10227939.
15. D'Agostino, P.; Violante, M.; Macario, G. An Embedded Low-Cost Solution for a Fog Computing Device on the Internet of Things. *Proceedings of the 2023 Eighth International Conference on Fog and Mobile Edge Computing (FMEC)* **2023**, 284-291. doi:10.1109/FMEC59375.2023.10306045.
16. Fava, F.B.; et al. Assessing the Performance of Docker in Docker Containers for Microservice-Based Architectures. *Proceedings of the 2024 32nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)* **2024**, 137-142. doi:10.1109/PDP62718.2024.00026.
17. Mahmud, R.; Toosi, A.N. Con-Pi: A Distributed Container-Based Edge and Fog Computing Framework. *IEEE Internet of Things Journal* **2022**, *9*, 6, 4125-4138. doi:10.1109/JIOT.2021.3103053.
18. Forsström, S.; Lindqvist, H. Evaluating Scalable Work Distribution Using IoT Devices in Fog Computing Scenarios. *Proceedings of the 2023 IEEE 9th World Forum on Internet of Things (WF-IoT)* **2023**, 1-6. doi:10.1109/WF-IoT58464.2023.10539565.
19. Khan, Z.A.; Aziz, I.A.; Osman, N.A.B.; Ullah, I. A Review on Task Scheduling Techniques in Cloud and Fog Computing: Taxonomy, Tools, Open Issues, Challenges, and Future Directions. *IEEE Access* **2023**, *11*, 143417-143445. doi:10.1109/ACCESS.2023.3343877.
20. D'Agostino, P.; Violante, M.; Macario, G. Optimizing LSTM-Based Temperature Prediction Algorithm for Embedded System Deployment. *Proceedings of Emerging Technologies and Factory Automation (ETFA)* **2024**. doi:.
21. Masini, et al. Machine Learning Advances for Time Series Forecasting. *J Econ Surv* **2023**, *37*, 76-111. doi:10.1111/joes.12429.
22. Li, G.; Jung, J.J. Deep Learning for Anomaly Detection in Multivariate Time Series: Approaches, Applications, and Challenges. *Information Fusion* **2023**, *91*, 93-102. doi:10.1016/j.inffus.2022.10.008.
23. Tufail, S.; Riggs, H.; Tariq, M.; Sarwat, A.I. Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, and Algorithms. Electronics 2023, 12, 1789. https://doi.org/10.3390/electronics12081789

24. Ayankoso, S.; Olejnik, P. Time-Series Machine Learning Techniques for Modeling and Identification of Mechatronic Systems with Friction: A Review and Real Application. Electronics 2023, 12, 3669. https://doi.org/10.3390/electronics12173669
25. Bittner, et al. Benchmarking Algorithms for Time Series Classification. *Data Science and Knowledge Engineering* **2023**, *48*, 63-77. doi:10.1016/j.dks.2023.100091
26. Liu, Y.; Zhao, J.; Wang, L.; Wang, W. Unified Modeling for Multiple-Energy Coupling Device of Industrial Integrated Energy System. *IEEE Transactions on Industrial Electronics* **2023**, *70*, 1, 1005-1015. doi:10.1109/TIE.2022.3152019.
27. Xu, Q.; Chen, Z.; Wu, K.; Wang, C.; Wu, M.; Li, X. KDnet-RUL: A Knowledge Distillation Framework to Compress Deep Neural Networks for Machine Remaining Useful Life Prediction. *IEEE Transactions on Industrial Electronics* **2022**, *69*, 2, 2022-2032. doi:10.1109/TIE.2021.3057030.
28. Alam, S.; Yakopcic, C.; Wu, Q.; Barnell, M.; Khan, S.; Taha, T.M. Survey of Deep Learning Accelerators for Edge and Emerging Computing. Electronics 2024, 13, 2988. https://doi.org/10.3390/electronics13152988
29. Yu, Y.; Guo, L.; Gao, H.; He, Y.; You, Z.; Duan, A. FedCAE: A New Federated Learning Framework for Edge-Cloud Collaboration Based Machine Fault Diagnosis. *IEEE Transactions on Industrial Electronics* **2024**, *71*, 4, 4108-4119. doi:10.1109/TIE.2023.3273272.
30. Lera, I.; Guerrero, C.; Juiz, C. YAFS: A Simulator for IoT Scenarios in Fog Computing. *IEEE Access* **2019**, *7*, 91745-91758. doi:10.1109/ACCESS.2019.2927895.
31. Wöbker, C.; Seitz, A.; Mueller, H.; Bruegge, B. Fogernetes: Deployment and Management of Fog Computing Applications. *Proceedings of the 2018 IEEE/IFIP Network Operations and Management Symposium (NOMS)* **2018**, 1-7. doi:10.1109/NOMS.2018.8406321.
32. D'Alessandro, D.; Gunderson, W.; Staten, E.; Donastien, Y.K.; Rodriguez, P.; Bailey, R. Integrating Modularity for Mass Customization of IoT Wireless Sensor Systems. *Proceedings of the 2021 Systems and Information Engineering Design Symposium (SIEDS)* **2021**, 1-5. doi:10.1109/SIEDS52267.2021.9483737.
33. Jadhav, A.; Gaikwad, R.; Patekar, T.; Dighe, S.; Shaikh, B.; Patankar, N.S. Predictive Maintenance of Industrial Equipment Using IoT and Machine Learning. *Proceedings of the 2023 4th International Conference on Computation, Automation and Knowledge Management (ICCAKM)* **2023**, 1-5. doi:10.1109/ICCAKM58659.2023.10449546.
34. Dhanraj, D.; Sharma, A.; Kaur, G.; Mishra, S.; Naik, P.; Singh, A. Comparison of Different Machine Learning Algorithms for Predictive Maintenance. *Proceedings of the 2023 International Conference for Advancement in Technology (ICONAT)* **2023**, 1-7. doi:10.1109/ICONAT57137.2023.10080334.
35. Amer, S.; Mohamed, H.K.; Badr Monir Mansour, M. Predictive Maintenance by Machine Learning Methods. *Proceedings of the 2023 Eleventh International Conference on Intelligent Computing and Information Systems (ICICIS)* **2023**, 58-66. doi:10.1109/ICICIS58388.2023.10391130.
36. Ullah, S.; Li, S.; Khan, K.; Khan, S.; Khan, I.; Eldin, S.M. An Investigation of Exhaust Gas Temperature of Aircraft Engine Using LSTM. *IEEE Access* **2023**, *11*, 5168-5177. doi:10.1109/ACCESS.2023.3235619.
37. Al-Masri, E.; Kalyanam, K.R.; Batts, J.; Kim, J.; Singh, S.; Vo, T.; Yan, C. Investigating Messaging Protocols for the Internet of Things (IoT). *IEEE Access* **2020**, *8*, 94880-94911. doi:10.1109/ACCESS.2020.2993363.
38. Elhanashi, A.; Dini, P.; Saponara, S.; Zheng, Q. Advancements in TinyML: Applications, Limitations, and Impact on IoT Devices. Electronics 2024, 13, 3562. https://doi.org/10.3390/electronics13173562