

A Unified Namespace-Based Edge-to-Cloud Framework for Industrial Data Analytics

Joaquín D. López , Juliam A. Díaz , Natalia Duarte , Iván Hernández , Carlos A. Fajardo , and Juan M. Rey 

Abstract—Industry 4.0 has intensified the demand for Industrial Internet of Things (IIoT) platforms supporting scalable and interoperable industrial analytics. However, many existing architectures treat data acquisition, processing, and visualization as loosely connected components within fragmented designs, limiting scalability, cross-domain generalizability, and accessibility for organizations lacking specialized infrastructure. This paper presents a flexible edge-to-cloud framework for industrial analytics based on a Unified Namespace (UNS) architecture. Unlike existing IIoT architectures, which rely on point-to-point or gateway-mediated integration and recover semantic context from external mappings, the proposed framework makes the namespace itself the integration contract: every message is self-describing, and integration interfaces grow linearly rather than multiplicatively with the number of connected components. The framework is evaluated on physically independent, heterogeneous hardware against three design pillars: scalability, generalizability, and accessibility. Its practical viability is demonstrated through a case study involving induction motor fault diagnosis. Per-device latency remains near 0.5 s and is unaffected as independent edge nodes join. Substituting the diagnostic model, and subsequently the data type, sampling regime, and analytic paradigm, required no modification outside the analytics component, while classification accuracy remained above 98% after cross-dataset adaptation. The complete pipeline runs on single-board hardware within the free tiers of open-source and managed services, making the framework viable for resource-constrained industrial deployments.

Link to graphical and video abstracts, and to code:
<https://latam.t.ieee9.org/index.php/transactions/article/view/10869>

Index Terms—Unified Namespace, Industrial Internet of Things, Edge-to-Cloud Framework, Industrial Analytics, Fault Diagnosis, Induction Motors.

I. INTRODUCTION

INDUSTRY 4.0 represents a transformative shift in the industrial landscape, positioning data as the primary strategic asset for enabling the *Future Factory* and ensuring sustainable long-term competitiveness [1]. This paradigm drives the evolution from traditional cyber-physical systems toward fully

integrated cyber-physical production systems, in which intelligent and interconnected entities collaborate autonomously to enhance efficiency, flexibility, and responsiveness [2]. In the manufacturing sector, digital transformation is conceptualized as a value-oriented process that integrates digital technologies with domain-specific expertise to generate sustainable competitive advantage [3]. To successfully navigate this transition, organizations require structured transformation frameworks that guide organizations through systematic planning, effective implementation, and ongoing optimization [4].

A direct consequence of Industry 4.0 is the emergence of industrial data analytics, driven by the convergence of Internet of Things (IoT), big data, and cloud technologies [5]. At the core of this paradigm lies the Industrial Internet of Things (IIoT), in which interconnected machines equipped with sensors and actuators continuously generate and transmit operational data, thereby enabling near-real-time monitoring and informed decision-making [6]. However, the exponential growth of sensor-generated data streams necessitates a strategic allocation of computational resources across edge and cloud layers [7]. This distributed computing paradigm highlights the importance of advanced analytical tools capable of processing high-volume, high-velocity data, ultimately strengthening the reliability, safety, and resilience of industrial systems [8].

IIoT platforms are the architectural foundation that makes industrial data analytics possible, integrating sensing, communication, and computing capabilities to connect physical assets with the digital systems that analyze and operationalize their data [9]. These platforms typically adopt layered or distributed architectures designed to systematically acquire sensor data, perform meaningful aggregation and preprocessing, and expose standardized interfaces that facilitate analytics integration and interoperability [10]. Edge computing is central to this design: by deploying computational resources at the network periphery, edge nodes process latency-sensitive workloads locally, thereby reducing response times and mitigating network congestion before transmitting selected data streams to cloud infrastructures for further analysis and long-term storage [11].

Several IIoT platform architectures have been proposed in the literature, each tailored to specific industrial applications and requirements. For instance, [12] deploys edge servers as outlier-verification nodes to process high-dimensional sensor streams, thereby enhancing anomaly detection performance at the network periphery. Similarly, [13] applies a three-tier architecture for robotic fulfillment systems integrating wireless sensor networks with Message Queuing Telemetry Transport

The associate editor coordinating the review of this manuscript and approving it for publication was Guillermo Valencia-Palomo (*Corresponding author: Gilbert Delgado*).

J. D. López, J. A. Díaz, C. A. Fajardo, and J. M. Rey are with Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones (E3T), Universidad Industrial de Santander, Bucaramanga 680002, Colombia (e-mails: gilbert.delgado@e3t.uis.edu.co, juliam2238321@correo.uis.edu.co, cafajar@uis.edu.co, and juanmrey@uis.edu.co).

N. Duarte, and I. Hernández are with Diseño y Automatización Industrial DAUTOM S.A.S., Bucaramanga, Colombia (e-mails: natalia.duarte@dautom.com.co, and ivan.hernandez@dautom.com.co).

(MQTT)-based gateways. In a service-oriented perspective, [14] introduces a Digital Twin-as-a-Service (DTaaS) model that bridges physical and cyber spaces throughout an agile value life cycle. Addressing cybersecurity concerns, [15] introduces an artificial intelligence (AI) zero-trust intrusion detection framework for interconnected industrial units.

To overcome the scalability and interoperability limitations inherent in conventional hierarchical architectures, alternative structural paradigms have emerged. Notably, [16] replaces the traditional pyramidal model with a flat IIoT architecture built on Open Platform Communications Unified Architecture (OPC UA) and Software-Defined Networking (SDN). Similarly, [17] addresses cross-organizational data fragmentation through a Federated Learning architecture aligned with data sovereignty principles, enabling collaborative analytics without centralized data sharing.

Despite these advancements, significant challenges persist. As highlighted in [11], centralized processing paradigms face inherent scalability bottlenecks when faced with the exponential growth of connected devices. Moreover, the integration of deep neural network models within multi-layered hierarchies introduces considerable computational overhead, limiting horizontal scalability. Accessibility presents an additional barrier, as many existing frameworks require considerable infrastructural investment and specialized expertise, constraining adoption among resource-limited organizations.

More fundamentally, many existing solutions treat data acquisition and analytics as isolated concerns within fragmented architectures, with limited emphasis on end-to-end integration and semantic interoperability. Consequently, a critical gap remains: the absence of flexible, vendor-agnostic frameworks capable of unifying data acquisition, analytics, storage, and visualization within a single cohesive architecture that scales horizontally, generalizes across industrial domains, and remains accessible without specialized infrastructure.

To overcome these challenges, this paper presents a flexible edge-to-cloud framework for industrial analytics based on a Unified Namespace (UNS) architecture. Unlike conventional IIoT platforms that rely on point-to-point integrations or centralized orchestration mechanisms, the UNS provides a shared, semantically structured information space that decouples data producers from consumers through a publish-subscribe model. This architectural approach enables the modular and scalable integration of heterogeneous components, including edge devices, cloud-based analytics services, data historians, and visualization clients, without requiring structural modifications as the system evolves.

To demonstrate its practical viability, the proposed framework is validated through an induction motor fault diagnosis use case. This scenario represents a realistic and computationally demanding application, encompassing high-frequency sensor signal acquisition, feature extraction and processing, machine learning (ML)-based inference, and multi-user visualization. Through this validation, the framework's capability to support end-to-end analytics workflows within a cohesive and extensible architecture is systematically evaluated.

The main contributions of this work are threefold:

- 1) **Architectural formalization of the UNS for industrial**

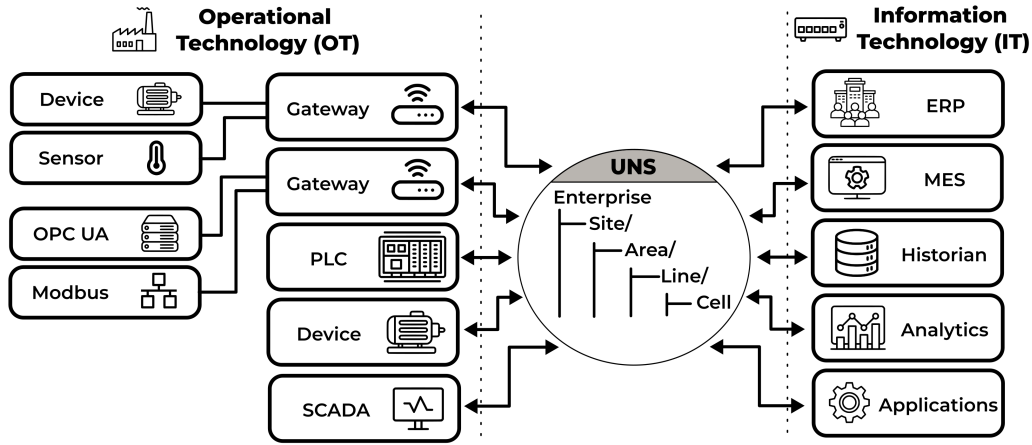
analytics integration: This work provides a structured formalization of the UNS as a semantically organized, event-driven integration paradigm for industrial systems. The UNS is defined as a shared information space that enforces decoupling between data producers and consumers while maintaining semantic consistency across distributed components. This formal treatment establishes a systematic foundation for its application in end-to-end industrial analytics pipelines.

- 2) **Design and evaluation of the modular edge-to-cloud analytics framework:** A technology-agnostic, five-component framework (Edge, UNS, Data Cloud, Historian, Client) is designed and implemented to operationalize the proposed integration paradigm. The architecture is evaluated against three explicitly defined design criteria: (i) *Scalability*, measured through end-to-end latency under increasing numbers of concurrent edge devices; (ii) *Generalizability*, assessed via cross-dataset transfer learning without architectural modification; and (iii) *Accessibility*, validated through deployment using exclusively open-source and free-tier infrastructure components.
- 3) **Experimental validation in an industrial fault diagnosis use case:** The proposed architecture is validated through an induction motor fault diagnosis application. Experimental results demonstrate that the UNS-based publish-subscribe model preserves semantic coherence across distributed nodes, and that per-node latency is unaffected by the addition of further independent edge nodes. Furthermore, classification accuracy remains above 98% after cross-dataset model adaptation.

The remainder of this article is structured as follows: Section II introduces the concept of Unified Namespace (UNS). Section III presents the architectural formalization of the proposed framework and defines the core design pillars: scalability, generalizability, and accessibility. Section IV presents a feature comparison against relevant IIoT architectures from the literature. Section V details the implementation of the system, utilizing an induction motor fault diagnosis case study as a practical validation of the framework's effectiveness. Section VI provides a thorough analysis of the experimental results and a discussion of the framework's performance. Finally, Section VII presents the main conclusions.

II. DESCRIPTION OF THE UNS CONCEPT

The UNS is an emerging concept within industrial automation and digital transformation. To date, most of the discussion related to UNS is available in sources such as blogs and websites [18]–[20]. This predominance of industry-driven discussion, contrasted with limited academic formalization, highlights the need for rigorous scholarly treatment. An important step in this direction is provided by [21], which offers a structured definition of UNS along with a formal articulation of its operational principles. Complementing this theoretical foundation, recent studies report practical experiences demonstrating successful implementations of UNS-based architectures in industrial applications [22]–[25].



Adapted from [26]

Fig. 1. Deployment of a UNS-based architecture in an industrial application.

The UNS is a data architecture designed for industrial automation that organizes all entities, resources, and devices within a central hub or broker of structured data. This hub functions as a single source of truth, offering a framework for collecting, organizing, and providing low latency access to information across all levels of the enterprise, thereby facilitating communication between IoT devices and enterprise systems. The UNS models both the organizational structure and its operational events, serving as a dynamic repository that continuously reflects the current state of the enterprise.

Building on this foundation, the UNS facilitates data interpretation and contextualization, thereby promoting interoperability across heterogeneous systems and platforms. By standardizing information produced by sensors and devices from diverse sources, it ensures uniform understanding and consistent data exchange across the enterprise. Its architecture is founded on a *publish-subscribe* model which is inherently compatible with MQTT protocol [27], [28]. Within this model, the broker functions as a central hub for the UNS, where data producers publish context-rich information and subscribing clients consume it within the shared namespace. This paradigm aligns with the architectural principles established by the ISA 95 standard (IEC 62264 [29]), which defines a hierarchical model for enterprise integration comprising Enterprise, Site, Area, Line, and Cell levels. The UNS operationalizes this hierarchy within a broker-based communication layer, mapping each level to a structured topic path that preserves the semantic context of the data as it traverses from field devices to enterprise applications.

Implementing a UNS architecture yields key operational benefits including scalability through elimination of point-to-point integrations, data normalization across heterogeneous sources, reduced time-to-value, and enhanced security via role-based access control [21]. To realize these benefits across a wide range of applications and system types, it is essential to assign a consistent topic structure that defines a predictable namespace, organized as Enterprise > Site > Area > Line > Cell [30]. This hierarchy directly reflects the ISA 95 functional levels, providing a normalized

mapping between the physical and logical organization of industrial assets and the semantic structure of the data namespace. A common hierarchy facilitates the convergence of operational technologies (OT) and information technologies (IT) by standardizing the organization of assets and data across systems. This standardization streamlines integration efforts, enhances scalability, and significantly improves system interoperability. Fig. 1 provides a general overview of the UNS concept. The UNS enables low-overhead data exchange by providing a unified semantic layer that bridges floor-level gateways with higher-level IT systems. This architecture facilitates the seamless integration of diverse clients, from data historians to specialized analytics platforms. The following section explores how this UNS architecture can be tailored to support industrial analytics, emphasizing the operational advantages of this unified and structured approach.

III. PROPOSED FRAMEWORK ARCHITECTURE

The proposed framework architecture is structured around a UNS and comprises five functional components: Edge, UNS, Data Cloud, Historian, and Client. Each component fulfills a clearly defined role and interacts through standardized protocols and data formats, thereby ensuring modularity, interoperability, and independence from specific technologies. As shown in Fig. 2, the architecture supports data flow from sensor acquisition at the industrial edge to cloud-based inference and visualization. The UNS acts as the integration hub, implementing topic-based messaging to coordinate communication across the system. The role of each component is defined independently of specific tools, ensuring applicability in diverse industrial deployments. Table I summarizes the main features of each component.

The UNS organizes all industrial data streams in a hierarchical topic structure aligned with enterprise operations, of the form {Enterprise}/{Site}/{Area}/{Line}/{Edge}/{AssetModel}/{DataType}. Within this structure, the Cell component represents the physical device and constitutes the smallest unit in the ISA-95 hierarchy. In the proposed architecture, this component is denoted as *Edge*,

TABLE I
COMPONENTS OF THE PROPOSED EDGE-TO-CLOUD FRAMEWORK FOR FAULT DIAGNOSIS

Component	Primary Function	Data Flow Role	Flexibility	Scalability Model
Edge	Sensor data acquisition and preprocessing	Data producer to UNS	Compatible with any edge device or gateway	Horizontal: linear addition of nodes
UNS	Contextual data hub and semantic routing	Bidirectional hub for all communications	Protocol-agnostic (MQTT, Kafka)	Vertical: clustering and load distribution
Data Cloud	Signal processing and ML inference	Data consumer/producer with UNS	Independent of cloud or hybrid platforms	Elastic: auto-scaling with computational demand
Historian	Time-series persistence and retrieval	Data sink from UNS, query provider to clients	Database-agnostic (non-relational or time-series databases)	Horizontal: distributed storage clusters
Client	Monitoring and Visualization	Consumer of historian and UNS	Platform-independent (web, desktop, mobile apps)	Concurrent: multi-user access

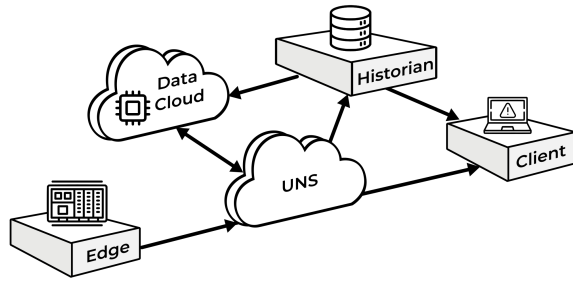


Fig. 2. Architecture of the proposed edge-to-cloud framework for fault diagnosis.

as it corresponds to the edge computing device responsible for acquiring and transmitting data from the field. Below the Cell (Edge) level, the subsequent elements, namely the asset model and data type, are incorporated to provide semantic context to the data, enabling a clear association between the physical device, its functional model, and the nature of the information being published. This topic schema ensures that every message published to the UNS carries sufficient metadata to identify its origin (device and location), its physical referent (asset model), and its informational nature (data type). As a result, the architecture eliminates the need for external lookup tables or ad hoc naming conventions, preserving semantic clarity and self-descriptiveness throughout the data flow.

The architecture mainly supports three distinct data flows. The edge data path follows $\text{Edge} \rightarrow \text{UNS} \rightarrow \{\text{Data Cloud}, \text{Historian}\}$, the processing path follows $\text{Data Cloud} \rightarrow \text{UNS} \rightarrow \text{Historian}$, and visualization occurs through $\text{Historian} \rightarrow \text{Client}$ for historical queries and $\text{UNS} \rightarrow \text{Client}$ for low-latency streams.

These data flows are not enforced through rigid orchestration logic. Instead, each component independently subscribes to the topics relevant to its function, which means that adding or removing a component does not require reconfiguration of the remaining elements. This decoupled routing model is what enables the scalability and generalizability properties evaluated in Section VI.

Interoperability within the proposed framework is achieved through the combination of standardized communication protocols and structured data representations. MQTT is

adopted as the primary protocol, configured with appropriate quality-of-service levels to ensure reliable message delivery within the UNS. The architecture remains protocol-agnostic, allowing alternative interfaces such as Advanced Message Queuing Protocol (AMQP), Apache Kafka, OPC Unified Architecture (OPC UA) [31], or Representational State Transfer Application Programming Interfaces (REST APIs) to be integrated based on deployment constraints, consumer capabilities, and system accessibility requirements. Data exchange relies on JavaScript Object Notation (JSON) for low-rate structured data such as diagnostic results, alarm states, and device metadata, where self-descriptiveness outweighs encoding cost, and on a compact binary encoding for the compute branch, where text encoding would otherwise dominate the payload. The topic schema is independent of the payload encoding, so both coexist in the same namespace and a consumer selects the branch appropriate to its function, while the architecture is structurally compatible with the Sparkplug B specification [32] for device birth and state monitoring within the UNS, although its validation falls outside the scope of this work. All communications are secured using Transport Layer Security (TLS) encryption, with access control enforced through username–password authentication, ensuring consistent, secure, and extensible interaction across UNS-connected components.

Algorithm 1 formalizes the data flows described above as the concurrent, event-driven behaviour of the components that exchange data through the UNS. The notation follows the topic schema defined in this section: R denotes the topic root built from the ISA-95 hierarchy, T a topic filter derived from it, and $\mathcal{P}$, $\mathcal{S}$ the publish and subscribe operations on the UNS. The essential property expressed by the algorithm is that no component invokes another directly. Each one declares the topics relevant to its function and reacts to messages as they arrive, so control flow is determined by the namespace rather than by an orchestrator.

Three architectural consequences follow directly from Algorithm 1. First, the Edge loop terminates at a publish operation and never references a consumer, so introducing an additional edge node requires no change anywhere else in the system. Second, subscriptions are expressed as topic filters rather than as device addresses. Because the single-

Algorithm 1 UNS-mediated edge-to-cloud data flow

Require: Topic root $R \leftarrow \{\text{Enterprise}\}/\{\text{Site}\}/\{\text{Area}\}/\{\text{Line}\}$
Require: Edge set E , window length L , sampling rate f_s

```

– Edge component (executes on each  $e \in E$ )
1: for each  $e \in E$  in parallel do
2:    $T_{raw} \leftarrow R/e/\{\text{AssetModel}\}/\{\text{DataType}\}$ 
3:   while acquisition active do
4:      $w \leftarrow$  acquire  $L$  samples at  $f_s$ 
5:      $m \leftarrow \text{ENCODE}(w, t_{stamp}, id_e)$   $\triangleright$  encoded payload
6:      $\mathcal{P}(T_{raw}, m)$   $\triangleright$  publish to UNS
7:   end while
8: end for

– Data Cloud component
9:  $\mathcal{S}(R/ + /\{\text{AssetModel}\}/\{\text{DataType}\})$   $\triangleright$  single-level
  wildcard: any device
10: while true do
11:    $m \leftarrow$  await message on subscribed topic
12:    $e \leftarrow \text{DEVICEOF}(m)$   $\triangleright$  device id recovered from the topic
  of  $m$ 
13:    $x \leftarrow \text{DECODE}(m)$ 
14:    $\phi \leftarrow \text{EXTRACT}(x)$   $\triangleright$  feature transform
15:    $y \leftarrow \text{INFER}(\phi)$   $\triangleright$  interchangeable analytic model
16:    $T_{proc} \leftarrow R/e/\text{Analysis}/\{\text{DataType}\}$ 
17:    $\mathcal{P}(T_{proc}, \text{ENCODE}(\phi, y))$ 
18: end while

– Historian component
Require: Ingestion set  $\mathcal{I} = \{(T_i, q_i, n_i)\}$   $\triangleright$  topic filter, parser,
  series name
19: for each  $(T_i, q_i, n_i) \in \mathcal{I}$  in parallel do
20:    $\mathcal{S}(T_i)$   $\triangleright$  explicit per-branch subscription
21:   while true do
22:      $m \leftarrow$  await message on  $T_i$ 
23:     if  $q_i(m)$  parses successfully then
24:        $\text{PERSIST}(n_i, q_i(m), \text{TAGS}(m))$ 
25:     else
26:       discard  $m$   $\triangleright$  preserves store consistency
27:     end if
28:   end while
29: end for

– Client component
30: for each panel  $p$  with binding  $(n_p, T_p)$  do
31:    $\text{QUERY}(n_p, T_p, \Delta t)$   $\triangleright$  historian, windowed
32: end for
33: optionally  $\mathcal{S}(T_p)$   $\triangleright$  direct UNS stream, low latency

```

level wildcard occupies the Edge position, one Data Cloud consumer or one ingestion binding serves every device whose topic conforms to the schema, and adding the n -th device of an existing data type requires no configuration change at all. Substituting INFER likewise changes analytic behaviour without altering any subscription. Third, the Historian binds each branch to its own parser and series name, so introducing a new data type is an additive operation. The same additive property holds for the Client, whose panels bind to a series name and a topic path. These properties are what the experiments in Section VI evaluate.

A. Core Design Pillars of the Framework

The proposed UNS-based architecture is evaluated according to three design principles, adapted from [33], that de-

termine its suitability for industrial environments: scalability, generalizability, and accessibility. These principles do not prescribe specific implementation choices; instead, they define the operational qualities expected from an industrial edge-to-cloud solution. Their role is to establish clear criteria against which the framework is later validated through controlled experiments.

- **Scalability:** The proposed architecture supports modular expansion by allowing additional sensing nodes, analytic modules, or visualization clients to be integrated without altering existing dataflows. Since all communication occurs through a hierarchical namespace, new components join the system by following predictable topic structures, thereby preserving interoperability. This property is evaluated in Section VI-A through an analysis of end-to-end latency as the number of concurrent edge devices increases. The results verify that the UNS-based routing mechanism and the downstream diagnostic pipeline sustain stable operation and predictable performance under progressively increasing load.
- **Generalizability:** Refers to the system's ability to integrate heterogeneous data sources and analytic models. A suitable architecture must support different machine types, sensing modalities, and diagnostic algorithms without requiring modifications to the communication layer or the semantic structure of the data. This property is validated in Section VI-B by incorporating a second diagnostic model trained on a different dataset through transfer learning, confirming that the cloud processing layer accommodates algorithm and dataset substitution while the UNS hierarchy, MQTT configuration, and dataflow definitions remain unchanged.
- **Accessibility:** This property addresses the practical adoption of the framework in both academic and industrial settings. Achieving accessibility requires the use of open protocols, lightweight deployment strategies, and tool-agnostic interfaces that reduce vendor lock-in and lower entry barriers. The architecture is designed to operate efficiently on resource-constrained edge devices while remaining capable of scaling to enterprise-grade infrastructures. This characteristic is validated in Section VI-C through a deployment based exclusively on open-source tools and free-tier cloud services. Furthermore, the complete implementation is made available in a public repository to ensure transparency and reproducibility.

IV. POSITIONING WITH RESPECT TO EXISTING IIOT ARCHITECTURES

Having defined the proposed architecture and its design pillars, its position with respect to existing IIoT architectures can now be established. Table II compares the framework against the architectures reviewed in the introduction using six capabilities, grouped in pairs under the three design pillars introduced in Section III-A, so that the comparison tests the same properties the framework is subsequently evaluated against. Each capability is architecturally objective, in the sense that it can be decided from the structure an architecture prescribes rather than from the performance it reports.

TABLE II
FEATURE COMPARISON BETWEEN THE PROPOSED UNS-BASED FRAMEWORK AND IIOT ARCHITECTURES FROM THE LITERATURE

Architecture	Scalability		Generalizability		Accessibility	
	Producer–consumer decoupling	Linear integration scaling	Rich context data	Cross-domain generalizability	Native historian	Fully open / vendor-agnostic
Edge anomaly verification [12]	×	×	×	×	×	×
Three-tier fulfillment system [13]	×	×	×	×	✓	×
DTaaS reference model [14]	×	×	✓	✓	×	×
Zero-trust intrusion detection [15]	×	×	×	×	×	×
Flat OPC UA + SDN [16]	×	×	✓	✓	✓	✓
Federated learning + data space [17]	✓	✓	✓	✓	✓	×
Proposed UNS framework	✓	✓	✓	✓	✓	✓

Note: A × denotes that the capability is not part of the architecture as reported rather than a documented deficiency. Several of the reviewed works pursue objectives for which a given capability is not a design goal; [12] and [15], in particular, contribute detection methods over generic layered models rather than integration architectures.

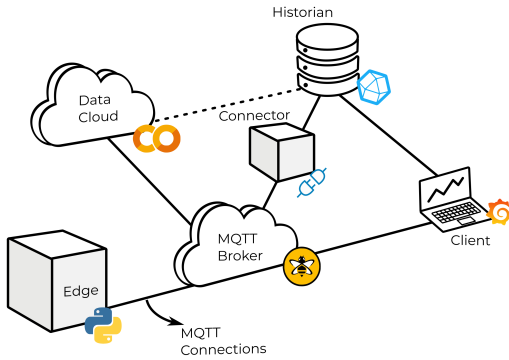


Fig. 3. Implementation of the UNS-based fault diagnosis framework.

Scalability is assessed through producer–consumer decoupling, which determines whether a component can be added or removed without reconfiguring the components around it, and linear integration scaling, which follows from it: when all exchange occurs through a shared information space, connecting n producers to m consumers requires $\mathcal{O}(n + m)$ subscriptions rather than the $\mathcal{O}(n \cdot m)$ mappings imposed by direct or gateway-mediated interfaces. *Generalizability* is assessed through rich context data, meaning that each message carries the metadata identifying its origin, physical referent, and informational nature, and through cross-domain generalizability, meaning that heterogeneous data sources and analytic models can be accommodated without altering the communication layer or the semantic structure of the data. *Accessibility* is assessed through native historian integration, since retrospective analysis otherwise requires additional infrastructure, and through openness, which conditions practical adoption by organizations without specialized infrastructure.

V. IMPLEMENTATION OF THE UNS-BASED ARCHITECTURE IN AN INDUSTRIAL CASE

A. Case Study: Induction Motor Fault Diagnosis

Induction motors constitute a foundational component of modern industrial and transportation systems. However, they frequently operate under harsh environmental and loading conditions that accelerate the degradation of critical components [34]. Statistical analyses indicate that bearing defects account for more than 40% of motor failures, while stator winding inter-turn short-circuit (ITSC) faults represent approximately 42% [35], [36]. Such failures result in significant economic losses due to reduced energy efficiency, unexpected downtime, and increased maintenance costs [37]. Although recent studies have reported remarkable diagnostic accuracy by emphasizing algorithmic optimization and feature engineering [38], comparatively less attention has been given to the data architectures required to operationalize these models in real-world industrial settings.

The proposed framework was implemented using open-source and cloud-based tools to validate its functionality across the components of the architecture. Fig. 3 illustrates the final deployment. The following subsections present a detailed description of each subsystem and its role within the overall implementation.

B. Implementation of the Functional Components

1) *Edge Component*: The edge component is responsible for sensor data acquisition and initial preprocessing. Due to the practical challenges of inducing physical motor failures under uncontrolled industrial conditions, the sensing of the induction motor was simulated for the case study using Raspberry Pi devices and a standard PC executing Python scripts. These scripts replayed pre-recorded vibration signals from the Korea Advanced Institute of Science and Technology (KAIST) dataset obtained from [39], covering both healthy and faulty motor states, including bearing defects, imbalance, and misalignment.

The vibration signatures in the dataset were acquired at a high sampling frequency of 25.6 kHz. Because standard MQTT communication is not optimized for individual message transmission at such high rates, the edge component implemented a window-based batching strategy. The high-frequency samples were grouped into discrete windowed data packets before being published to the UNS. This approach allows the system to preserve the high-fidelity spectral dynamics required for accurate fault diagnosis while ensuring stable throughput across the event-driven architecture. Each window is serialized as a compact binary array rather than as text, since vibration samples are numeric and textual encoding would otherwise inflate each value from four bytes to many characters. In the deployed pipeline, each compute-topic window is 0.39 MB, and combined with the downsampled storage-topic batch (approximately 0.12 MB), the total per-device publish rate is 0.51 MB/s. This payload representation is independent of the topic hierarchy, broker configuration, ingestion pipeline, and client queries, which is what allows the encoding to be selected per branch according to data rate.

Industrial sensors typically aggregate on-device, transmitting scalar indicators (root mean square RMS, peak, crest factor) at low rates per ISO 20816 severity criteria. Raw windowed transmission is adopted here deliberately: the deployed model classifies from the FFT magnitude spectrum, and scalar indicators discard the frequency-domain structure on which spectral fault signatures depend, which no downstream step can recover. This is a requirement of the analytic model, not of the architecture. Because the topic schema encodes the data type explicitly, aggregated indicators can be published to a dedicated branch (*.../MotorModel/rms*) alongside or instead of the raw stream, ingested and rendered without modifying the Historian or Client, as shown for temperature in Section VI-B2. Hybrid regimes such as continuous RMS trending with raw windows published on threshold exceedance are equally supported.

Replacing the simulation scripts with physical hardware would require changes only at the edge node level; all downstream components remain agnostic to the data source.

2) *Unified Namespace*: The UNS was implemented through the configuration of an MQTT broker. HiveMQ was deployed as the communication hub, applying the hierarchical topic structure described in Section III. As a result, all data exchanges among system components conformed to a consistent and well-defined semantic model.

To ensure secure communication, the broker was configured with TLS encryption and username–password authentication, thereby protecting message confidentiality and integrity. In addition, Topic level access control mechanisms were applied to regulate publishing and subscribing permissions according to each component’s functional role within the architecture. These controls were implemented in alignment with the security provisions described in Section III.

3) *Data Cloud Processing*: Google Colab was utilized as the data cloud environment. It subscribed to raw vibration topics within the UNS and executed the processing pipeline, which consisted of two main stages: 1) Fast Fourier Transform (FFT) for frequency-domain feature extraction; and 2) A

convolutional neural network (CNN) for classification of motor conditions, operating on the extracted FFT vectors.

The extracted features and inference outputs were subsequently published back to the UNS, where they became available for storage, visualization, and further downstream analysis. This bidirectional interaction with the broker, subscribing to raw data topics and publishing to processed data and diagnosis topics, exemplifies the decoupled routing model described in Section III. Within this model, the cloud processing layer functions simultaneously as a data consumer and producer, reinforcing the event-driven and modular characteristics of the proposed UNS-based architecture.

To demonstrate the model-agnostic nature of the proposed framework, a one-dimensional CNN was implemented. This architecture follows a baseline established in the literature for vibration-based bearing-fault classification and operates on frequency-domain representations [40]. The model receives as input a magnitude spectrum of size 2560×1 , obtained from applying a FFT to each vibration window. Its backbone consists of stacked convolutional blocks with batch normalization and max pooling layers, designed to capture hierarchical spectral patterns associated with bearing defects. The network was trained using the KAIST dataset [39]. When evaluated on Domain 0 of the KAIST dataset using five-fold cross-validation with a 70/15/15 training/validation/test split, the model achieved a macro average accuracy of 99.13% with a standard deviation of 0.17% across folds, confirming robust classification performance prior to deployment within the framework.

During deployment, only the backbone is utilized. Owing to its modular structure, the classification head can be replaced or the final layers fine-tuned to support transfer-learning scenarios. Fig. 4a illustrates the complete backbone architecture, highlighting the progression from low-level spectral filtering to high-level feature abstraction.

4) *Historian Integration*: InfluxDB was deployed as the historian to store both raw and processed data streams. Integration with the UNS was achieved through Telegraf, which subscribed to the relevant MQTT topics and ingested time-series data enriched with appropriate metadata and timestamps, extracting the timestamp, sensor identifier, and measurement fields required for structured ingestion. This configuration enabled persistent storage, historical querying, and traceability of the diagnostic process.

As formalized in Algorithm 1, Telegraf was configured with one ingestion binding per semantic branch, each pairing a topic filter with the parser and series name appropriate to that branch. Bindings were defined for the storage, diagnostic, alarm, and metric branches, all of which carry JSON payloads; the high-rate binary compute branch is consumed directly by the Data Cloud component and is not stored in raw form. Messages that failed to conform to the expected schema were discarded to maintain data consistency and integrity. Although the present deployment uses a single InfluxDB instance, the architecture inherently supports horizontal scalability through distributed storage clusters, as outlined in Table I.

5) *Client Visualization*: Grafana was utilized for system visualization and monitoring. Custom dashboards were de-

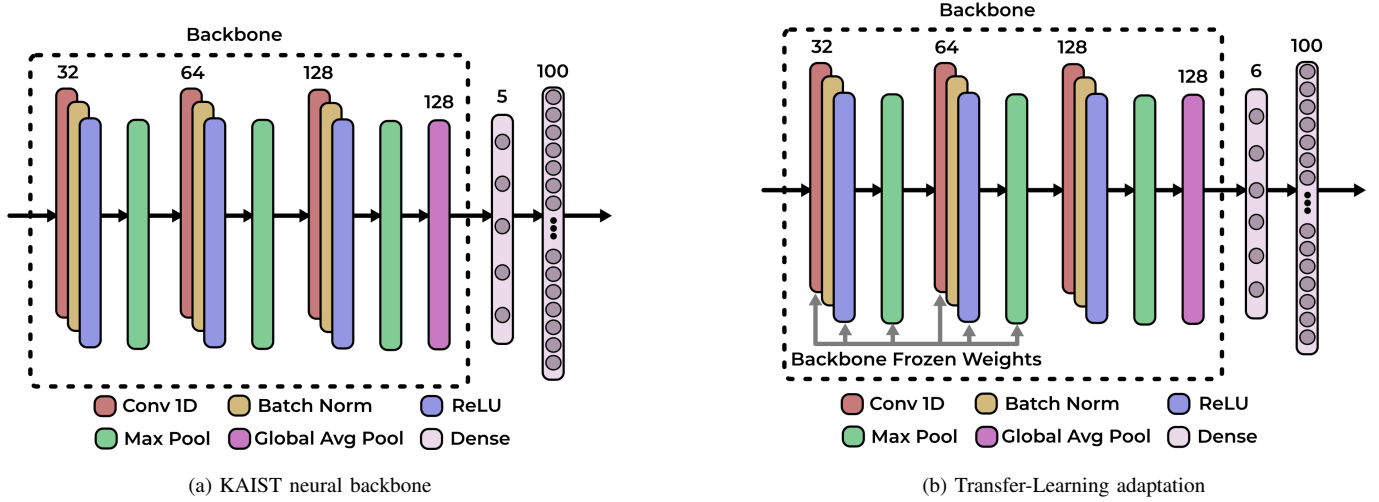


Fig. 4. Fault diagnosis model architecture: (a) backbone trained on KAIST [40] with its original five-class head; (b) transfer-learning adaptation, in which the same backbone is reused with frozen weights and only the six-class classification head is retrained for CWRU.

signed to display time-domain vibration signals, FFT-derived frequency-features, and CNN-based fault predictions. By integrating low-latency data streamed through the UNS with historical records retrieved from InfluxDB, the client interface provided operators with a unified view of raw measurements, processed features, and diagnostic outcomes.

The dashboard layout comprises three primary panels: raw vibration signals in the time domain, their corresponding FFT representations, and the CNN diagnostic outputs. This organization reflects the three data flows defined in Section III, offering end-to-end visibility within a single coherent interface, as illustrated later in Fig. 5 (Section VI-A).

VI. EVALUATION TESTS

The framework was evaluated through three experiments, each isolating one design principle defined in Section III-A while keeping the remaining components fixed.

A. Scalability: Adding Independent Edge Nodes

Scalability was evaluated by progressively increasing the number of concurrent edge devices publishing to the UNS and measuring the end-to-end latency between edge transmission and cloud-based diagnosis. Each edge node in this experiment is physically independent hardware with its own processor, operating system, and network interface: two Raspberry Pi 4 units running Raspberry Pi OS and an x86 laptop running Windows. Every node-added experiment was run from a fresh start. The cloud layer remained a single-threaded Google Colab process executing FFT extraction and CNN inference for all devices, and the historian and visualization stack ran on the laptop, which also acted as an edge node; this co-location was not a limiting factor, as CPU utilization remained low in all experiments.

Every node in this experiment publishes the same pinned source file at the same rate over the same binary-encoded compute topic, so offered load is equalized by construction rather than differing by device capability: each node sustains

TABLE III
STEADY-STATE END-TO-END LATENCY PER EDGE NODE AS PHYSICALLY INDEPENDENT DEVICES ARE ADDED CUMULATIVELY

Nodes	Device	Mean [ms]	Median [ms]	p95 [ms]
1	Motor1	609	601	661
2	Motor1	508	500	647
	Motor2	551	519	734
3	Motor1	532	498	782
	Motor2	495	476	740
	Motor3	401	386	589

0.39 MB compute batches plus 0.12 MB storage batches at approximately one batch per second, i.e. ≈ 0.51 MB/s per device regardless of host. Latency is reported per device in Table III, and statistics are computed over the steady-state portion of each series, excluding the first 30 s after a node joins. Fig. 5 shows the Grafana dashboard monitoring three devices publishing 25.6 kHz batched data in parallel.

Four observations follow. First, *no messages were lost*. Across this subsection's three- and nine-device experiments, 11 142 publish operations completed with `rc=0` and no failures, and every active node continued producing valid diagnostic outputs throughout.

Second, adding a node does not degrade the nodes already running, up to the capacity limit identified below. The Raspberry Pi's steady-state latency changes only modestly as further nodes join (609 ms mean at one device, 532 ms at three), and the second Raspberry Pi is essentially unaffected (551 ms to 495 ms). This is the behavior predicted by producer-consumer decoupling. A publisher's performance does not depend on how many peers share the namespace, because no publisher holds a connection to, or awaits acknowledgement from, any consumer other than the broker.

Third, edge nodes are never the bottleneck. Across this run, the two Raspberry Pis averaged 9.2% and 7.8% CPU



Fig. 5. Grafana dashboard monitoring three physically independent edge nodes concurrently.

TABLE IV
AGGREGATE END-TO-END LATENCY ACROSS DEVICES

Devices	Mean [ms]	Median [ms]	p95 [ms]
1	570	568	647
2	582	484	1201
3	446	446	667
4	508	544	801
5	670	588	1402
6	545	516	968
7	879	762	1765
8	986	938	1728
9	1789	1859	2546

respectively (max 90.2% and 63.6% in brief spikes) and the laptop averaged 0.76% CPU (max 15.0%) while sustaining their publish load. All three remained far from saturation throughout every configuration tested, including the highest-load configuration described next.

Extending the ramp to nine devices: To locate the point at which this architecture stops scaling linearly, the same three physical hosts were pushed further. Each of the three hosts (two Raspberry Pis, one laptop) ran three independent publisher stream processes instead of one, for nine simulated devices total on the same three-machine footprint used above. Devices were added one at a time, every 10 s, cycling across the three hosts. Table IV and Fig. 6 report latency immediately after each device count is reached, using the same steady-state windowing as Table III.

Latency is flat through six devices (446–588 ms median) and breaks sharply at seven, reaching 1859 ms median immediately after the ninth device joins. Fourth, and most importantly, *this knee is not caused by edge hardware, the network, or the historian stack*. CPU on every edge host remained low throughout the nine-device run, and Telegraf and InfluxDB themselves averaged 23.1% and 20.3% CPU respectively at nine devices, never approaching saturation.

Locating the Bottleneck: a measured, not assumed, capacity limit: The cloud diagnosis service logs a timestamp imme-

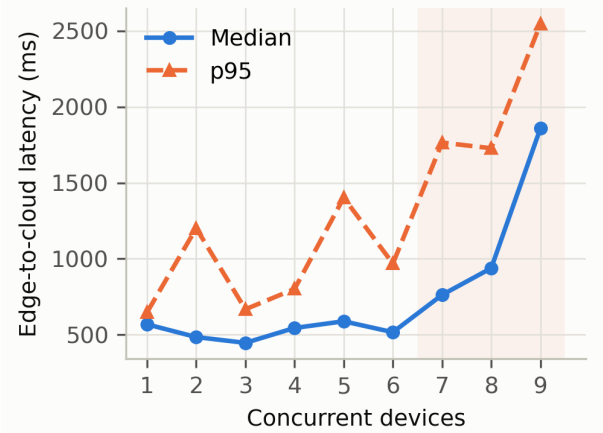


Fig. 6. Aggregate latency vs. concurrent device count.

diately before it begins processing each received batch. The interval between consecutive such timestamps is therefore a direct measurement of the service's single-threaded throughput, independent of network conditions. Across three independent runs, this measurement converges tightly: the processing floor (minimum interval, i.e. no queueing present) is 0.100–0.105 s in every run, and the sustained throughput under backlog is 7.6–8.2 messages/s, measured identically during the nine-device ramp and, separately, during the extended hold described below, with no drift between the two. At the measured ≈ 1 batch/s per device publish rate, this throughput predicts an unstable queue (arrival rate exceeding service rate) above 7.6–8.2 devices, matching the empirically observed knee within measurement noise. The single-threaded FFT extraction plus CNN inference stage, not any other architectural component, is the capacity limit of this deployment.

To confirm that the observed degradation is genuine queue instability and not a transient startup effect, the nine-device configuration was held for approximately 150 s beyond the point at which the ninth device joined. Fig. 7 shows the median latency grows from 1859 ms to a peak of 8575 ms over

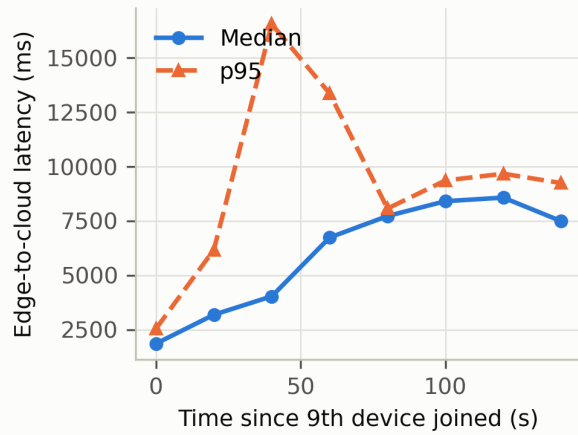


Fig. 7. Latency during a sustained hold at nine devices.

the hold, a $4.6\times$ increase, with the final messages fluctuating between 7500 and 8575 ms rather than returning toward the pre-saturation baseline. Throughout this window the cloud service’s own processing rate stayed constant, so the growth is not caused by the service getting slower under load, but by a fixed-capacity single server accumulating a backlog because devices offer more work than it can drain: the signature of an $M/M/1$ queue operating above its stability threshold ($\rho \approx 9/7.8 \approx 1.15$).

Edge and historian-stack resource usage remained low throughout this extended hold as well (edge hosts averaging 3.1-8.8% CPU per device, brief spikes to 65.6%; Telegraf 23.1%, InfluxDB 20.3%), reinforcing that the instability is confined to the cloud inference stage and does not implicate any other component of the architecture.

B. Generalizability

1) *Integrating New ML Models and Datasets:* Generalizability was first evaluated by incorporating a second diagnostic model based on a transfer learning strategy. Building upon the pretrained KAIST backbone [39], a new classification head was trained on the CWRU (Case Western Reserve University) dataset [41], which differs in operating conditions and class distribution. During adaptation, the backbone remained largely fixed while only the final layers were retrained, enabling the transfer of learned spectral representations across domains. The resulting architecture of the adapted model is illustrated in Fig. 4b.

The transfer learning procedure followed a staged adaptation strategy. First, batch normalization layers were recalibrated using mixed KAIST and CWRU samples, while all other layers remained fixed. The original KAIST classifier was then replaced with a new six-class output layer initialized from class prototype features, and only this head was trained to learn the CWRU decision boundaries. In a subsequent fine-tuning stage, the final convolutional block, global average pooling layer, dense layer, and batch normalization layers were unfrozen, allowing selective adaptation of higher-level representations while earlier feature extraction layers remained fixed. After a final batch normalization refresh, all

layers were frozen for inference. This progressive strategy preserves the general spectral features learned from KAIST while adapting higher-level representations to CWRU-specific vibration characteristics.

Importantly, integrating the new model required no modifications to the UNS hierarchy, MQTT configuration, or dataflow definitions. Once deployed, the adapted model subscribed to the same diagnostic topics, processed incoming FFT vectors within the Data Cloud component, and republished results to the corresponding UNS branches. This demonstrates that the architecture accommodates analytic variability without additional infrastructure engineering.

The confusion matrices for the model, separated by datasets, are presented in Fig. 8. They illustrate the diagnostic performance before and after adaptation, while demonstrating that the model can be replaced or retrained without modifying any component of the edge-to-cloud pipeline. Specifically, the KAIST backbone maintained its original classification accuracy of 99.13% after integration into the transfer learning pipeline, while the adapted CWRU head achieved 98.04% accuracy across six fault classes, with misclassifications concentrated between the Normal and Ball fault categories as shown in Fig. 8. These results confirm that the framework supports both model-level and dataset-level generalizability, enabling the integration of heterogeneous algorithms and previously unseen vibration sources within a consistent semantic and communication layer.

From an architectural perspective, the key outcome of this experiment lies not in the classification accuracy itself, which depends on model design and training strategy, but rather the confirmation that the UNS-based infrastructure supports model substitution as a modular operation. The only modification required was replacing the inference script in the Data Cloud component; the Edge, UNS, Historian, and Client layers remained entirely unchanged. This decoupling between the diagnostic algorithm and the data infrastructure is what distinguishes the proposed framework from monolithic fault diagnosis systems, where model updates typically propagate across the entire pipeline.

2) *Changing Data Type and Analytic Task:* The preceding experiment substituted the analytic model while holding the data type fixed. A stronger test of architectural generalizability is to change the nature of the data itself and the class of analysis performed on it, since these determine the semantic structure of the namespace rather than merely the contents of one component. This experiment therefore replaces high-frequency vibration analysis with low-rate temperature monitoring and threshold-based alarming: a different acquisition regime, a different payload, and an analytic task.

Two temperature sensors were attached to an edge node publishing to `.../Motor1/Edge/MotorModel/temperature`, at a sampling rate several orders of magnitude below the 25.6 kHz vibration case. In the Data Cloud component, the FFT-and-CNN pipeline was replaced by a rule-based evaluator that computes a running maximum and derives a binary alarm state, republishing both to

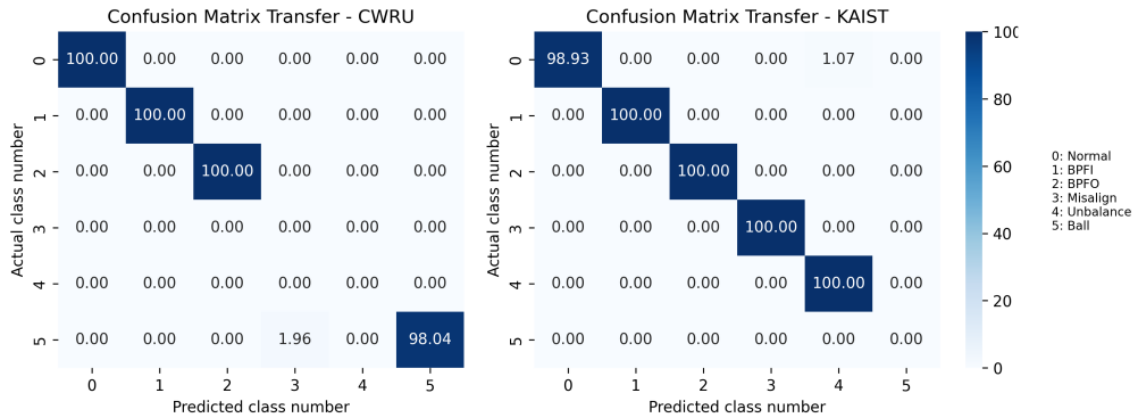


Fig. 8. KAIST and CWRU Confusion matrix for the trained model.

.../Motor1/Analysis/Diagnosis/temperature. Because the topic schema encodes the data type explicitly, the new stream occupies its own semantic branch alongside the vibration branches without collision or reconfiguration.

Fig. 9 shows the resulting client view. The raw two-sensor stream and its derived running maximum are rendered alongside the alarm state and its history, together with the same end-to-end latency measurement used in Section VI-A and the accumulated storage payload reported by the historian. Notably, the latency and storage panels required no adaptation: because they subscribe to structural properties of the namespace rather than to vibration-specific topics, they instrument the temperature deployment unchanged.

The architecturally significant result is the extent of the modification required. The new dashboard panels are ordinary queries against existing data sources rather than new integrations. The Edge, UNS, Historian, and Client components therefore remained unchanged across a transition that altered the sampling regime by several orders of magnitude, replaced a deep-learning inference stage with a threshold rule, and introduced a new analytic output type that had no counterpart in the original deployment.

Taken together, VI-B1 and VI-B2 experiments separate the two senses in which the framework generalizes: analytic generalizability, where the model changes while the data type is fixed, and semantic generalizability, where the data type, sampling regime, and analytic paradigm change while the namespace absorbs them as additional branches. It should be noted that both experiments remain within the domain of rotating-machine monitoring, but it shows that neither the analytic model nor the nature of the sensor data is tied to the architecture.

C. Accessibility: Leveraging Open and Free Tools

Accessibility was validated by implementing the proposed architecture exclusively using open-source tools and free-tier services. The resulting pipeline, comprising Python-based edge scripts, an MQTT broker, InfluxDB, Google Colab, and Grafana, operated without proprietary dependencies or licensing costs. The complete solution is publicly available in a Git repository and can be deployed locally using containerized

services, facilitating rapid experimentation, reproducibility, and on-premise integration. The lightweight deployment on Raspberry Pi devices further confirms accessibility under resource-constrained conditions.

To address accessibility quantitatively rather than by construction alone, the deployment was characterized along four dimensions: hardware requirements, software footprint, deployment effort, and recurring cost relative to managed alternatives. Table V summarizes the result.

Three points follow. First, the hardware requirement at the edge is a single-board computer costing a small fraction of an industrial gateway: the publishing node averages 9.1 % CPU on a quad-core Cortex-A72, with brief spikes to 64.2 % during batch assembly, leaving most of its processing capacity available on average. Second, the software footprint is modest and fully declarative. The server side comprises three official container images totalling 1188 MB, provisioned through a single Compose manifest derived from a public reference deployment [42], so no manual installation is required; the edge node needs only a Python runtime and an MQTT client library, which is what permits deployment on constrained hardware, and the whole containerized historian and visualization stack itself averages under 6 % aggregate CPU (Telegraf 3.1 %, InfluxDB 2.7 %, Grafana 0.3 %) and ≈300 MB of the server's 16 GB while serving one active device. Third, the complete pipeline operates within the free allowances published by the managed services used here, at no licensing cost.

Two qualifications apply to the cost comparison. The free-tier figure establishes that the pipeline fits within published free allowances, which is the relevant threshold for academic replication and pilot deployments; it is not a claim that the framework operates at zero cost in production. The constraints are material: the broker allowance of 10 GB per month and the 100-connection ceiling [43] bound deployment scale, and the retention windows of the free historian and client tiers [44], [45] preclude long-horizon historical analysis. Sustained operation at the measured 0.46 GB/h per device therefore requires either usage-based charges or self-hosting, the latter remaining available at no licensing cost since every component is open source. The managed-service rates are moreover



Fig. 9. Client view for the temperature monitoring case.

published list prices for comparable functionality rather than a measured alternative deployment; a controlled cost comparison against commercial IIoT platforms under equivalent industrial workloads was not performed and remains future work.

Overall, the experimental results demonstrate that the proposed framework preserves semantic consistency and operational flexibility while satisfying the three foundational design pillars. These findings position the solution as a practical and viable approach for edge-to-cloud industrial fault diagnosis in real-world production deployments.

VII. CONCLUSION

This paper has presented a flexible edge-to-cloud framework for industrial data analytics based on a UNS architecture. The primary contribution of this work is the architectural formalization of the UNS as a semantically organized, event-driven integration paradigm. By establishing a shared information space that enforces decoupling between data producers and consumers, the proposed architecture moves beyond fragmented system designs to provide a systematic foundation for edge-to-cloud industrial analytics.

The design and evaluation of this technology-agnostic, five-component framework (Edge, UNS, Data Cloud, Historian, and Client) demonstrate that high-performance analytics can be operationalized through three core design pillars. The framework's scalability was confirmed through low-latency performance under increasing loads; its generalizability was demonstrated via cross-dataset model adaptation without architectural changes; and its accessibility was validated through a cost-effective deployment using open-source and free-tier infrastructure.

Finally, the experimental validation in an induction motor fault diagnosis use case provided concrete evidence of the framework's effectiveness. The results show that the UNS-based publish-subscribe model preserves semantic coherence and per-node latency that is unaffected by the addition of further independent edge nodes. Furthermore, the system

sustained high diagnostic performance, with classification accuracy remaining above 98% even after model adaptation. Ultimately, this work fills the critical gap in the literature between isolated, siloed fault diagnosis mechanisms and fully integrated, enterprise-wide architectures capable of sustaining low-latency industrial processing and visualization.

Future work will address three open dimensions. Regarding scalability, subsequent studies will deploy the framework on higher-performance hardware with parallelized cloud inference to isolate architectural overhead from hardware constraints under high-load conditions. For generalizability, the architecture will be extended beyond motor fault diagnosis to heterogeneous industrial assets and industry-standard data protocols, validating the UNS as a plant-wide digital backbone. Finally, accessibility will be strengthened through granular access control, broker-level encryption, and semantic agents for automated legacy system integration.

DATA AND CODE AVAILABILITY

To ensure transparency and reproducibility, the complete implementation is publicly available at <https://github.com/giljoa/e2c-uns-fw>. The repository contains the edge publisher scripts for KAIST and CWRU data replay, the broker configuration, the Telegraf ingestion configuration, the Grafana dashboard definitions, the trained model weights for both the baseline and transfer-learning networks, and the Google Colab notebooks used for cloud-based inference.

Deployment parameters not reproduced in the text are recorded in these artifacts and can be inspected directly: the MQTT quality-of-service level, keep-alive interval and TLS settings used by every component are declared in the publisher scripts and in the Telegraf configuration; the acquisition window length, sampling rate and batching interval are defined in the edge publisher scripts; and the network hyperparameters, training schedule and the staged transfer-learning procedure are contained in the Colab notebooks together with the code that produced the reported accuracies. Hardware and software

TABLE V
MEASURED DEPLOYMENT CHARACTERISTICS OF THE
REFERENCE IMPLEMENTATION

	Edge node	Server node
Platform	Raspberry Pi 4B	Commodity laptop
Processor	Quad-core Cortex-A72 (ARMv8) 1.5 GHz	Intel Core i7 (x86-64)
Memory	4 GB LPDDR4	16 GB
Power supply	5.1 V / 3 A USB-C	Mains
Operating system	Raspberry Pi OS	Windows
CPU utilization	9.1 % avg. (peaks 64.2 %)	≈6 % avg. (peaks 18.2 %)
Memory utilization	≈6 % (≈263 MB)	≈2 % (≈300 MB)
Core temperature	57.9–58.9 °C, no throttling	n/a
Runtime dependency	Python 3, paho-mqtt	Docker Engine
Container images (server node)		
Grafana OSS 8.4.3	277 MB	
InfluxDB 2.7.11	406 MB	
Telegraf 1.33.1	505 MB	
Total	1188 MB	
Deployment and data		
Provisioning	Single docker compose up manifest [42]; edge node requires no container runtime	
Historised data	0.46 GB/h per device (0.13 MB/s)	
Recurring cost		
Free-tier total	USD 0/month: broker 100 connections and 10 GB/month [43], historian 30-day retention [44], client 10 k series [45], cloud runtime free	
Managed equivalent	Broker from USD 0.34/h plus 0.80/10 ⁶ messages [43]; client USD 19/month plus usage [45]; historian USD 0.0025/MB ingested [44]	

versions for the deployed stack are reported in Table V. The vibration datasets used for training and validation are cited and described in Sections V and VI.

DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE MANUSCRIPT PREPARATION PROCESS

During the preparation of this work the authors used Claude (Anthropic) in order to support writing and language refinement, and ChatGPT (OpenAI) in order to assist with code development for the public repository. After using these tools, the authors reviewed and edited the content as needed and took full responsibility for the content of the published article.

ACKNOWLEDGMENTS

This research was supported by Universidad Industrial de Santander under project 4624-2025 "Impulsando competencias

para la Transición Energética y la Agroindustria en Santander (ICOTEA)" and by MinCiencias with the project "Diseño, desarrollo e implementación de una plataforma IIoT para formación de profesionales en tecnologías de la cuarta revolución industrial", contract No. 82273 2022-0719.

REFERENCES

- [1] N. Anumbe, C. Saidy, and R. Harik, "A primer on the factories of the future," *Sensors*, vol. 22, no. 15, p. 5834, 2022, 10.3390/s22155834.
- [2] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, "Industry 4.0 and industry 5.0— inception, conception and perception," *J. Manuf. Syst.*, vol. 61, pp. 530–535, 2021, 10.1016/j.jmsy.2021.10.006.
- [3] C. H. Lee, C. L. Liu, A. J. Trappey, J. P. Mo, and K. C. Desouza, "Understanding digital transformation in advanced manufacturing and engineering: A bibliometric analysis, topic modeling and research trend discovery," *Adv. Eng. Inform.*, vol. 50, p. 101428, 10 2021, 10.1016/j.aei.2021.101428.
- [4] N. Bellantuono, A. Nuzzi, P. Pontrandolfo, and B. Scozzi, "Digital transformation models for the I4.0 transition: Lessons from the change management literature," *Sustainability*, vol. 13, no. 23, p. 12941, 2021, 10.3390/su132312941.
- [5] T. Zheng, M. Ardolino, A. Bacchetti, and M. Perona, "The applications of Industry 4.0 technologies in manufacturing context: a systematic literature review," *Int. J. Prod. Res.*, vol. 59, no. 6, pp. 1922–1954, 2021, 10.1080/00207543.2020.1824085.
- [6] M. Javaid, A. Haleem, R. P. Singh, R. Suman, and E. S. Gonzalez, "Understanding the adoption of Industry 4.0 technologies in improving environmental sustainability," *Sustain. Oper. Comput.*, vol. 3, pp. 203–217, 2022, 10.1016/j.susoc.2022.01.008.
- [7] M. Ghobakhloo, H. A. Mahdiraji, M. Iranmanesh, and V. Jafari-Sadeghi, "From industry 4.0 digital manufacturing to industry 5.0 digital society: a roadmap toward human-centric, sustainable, and resilient production," *Inf. Syst. Front.*, pp. 1–33, 2 2024, 10.1007/s10796-024-10476-z.
- [8] M. Ghobakhloo, "Industry 4.0, digitization, and opportunities for sustainability," *J. Clean. Prod.*, vol. 252, p. 119869, 2020, 10.1016/j.jclepro.2019.119869.
- [9] S. Selvarajan, G. Srivastava, A. O. Khadidos, A. O. Khadidos, M. Baza, A. Alshehri, and J. C. W. Lin, "An artificial intelligence lightweight blockchain security model for security and privacy in iiot systems," *J. Cloud Comput.*, vol. 12, pp. 38–, 3 2023, 10.1186/s13677-023-00412-y.
- [10] S. F. Tan and A. Samsudin, "Recent technologies, security countermeasure and ongoing challenges of industrial internet of things (IIoT): A survey," *Sensors*, vol. 21, no. 19, p. 6647, 2021, 10.3390/s21196647.
- [11] A. A. Mirani, G. Velasco-Hernandez, A. Awasthi, and J. Walsh, "Key challenges and emerging technologies in industrial IoT architectures: A review," *Sensors*, vol. 22, no. 15, p. 5836, 2022, 10.3390/s22155836.
- [12] G. Han, J. Tu, L. Liu, M. Martínez-García, and Y. Peng, "Anomaly detection based on multidimensional data processing for protecting vital devices in 6G-enabled massive IIoT," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5219–5229, 2021, 10.1109/JIOT.2021.3051935.
- [13] K. L. Keung, C. K. Lee, and P. Ji, "Industrial internet of things-driven storage location assignment and order picking in a resource synchronization and sharing-based robotic mobile fulfillment system," *Adv. Eng. Inform.*, vol. 52, p. 101540, 2022, 10.1016/j.aei.2022.101540.
- [14] S. Ahelerooff, X. Xu, R. Y. Zhong, and Y. Lu, "Digital twin as a service (DTaaS) in industry 4.0: an architecture reference model," *Adv. Eng. Inform.*, vol. 47, p. 101225, 2021, 10.1016/j.aei.2020.101225.
- [15] A. A. Laghari, A. A. Khan, A. Ksibi, F. Hajje, N. Kryvinska, A. Almadhor, M. A. Mohamed, and S. Alsubai, "A novel and secure artificial intelligence enabled zero trust intrusion detection in industrial internet of things architecture," *Sci. Rep.*, vol. 15, no. 1, p. 26843, 2025, 10.1038/s41598-025-11738-9.
- [16] R. Wang, C. Gu, S. He, Z. Shi, and W. Meng, "An interoperable and flat industrial internet of things architecture for low latency data collection in manufacturing systems," *J. Syst. Archit.*, vol. 129, p. 102631, 2022, 10.1016/j.sysarc.2022.102631.
- [17] B. Farahani and A. K. Monsefi, "Smart and collaborative industrial IIoT: A federated learning and data space approach," *Digit. Commun. Netw.*, vol. 9, no. 2, pp. 436–447, 2023, 10.1016/j.dcan.2023.01.022.
- [18] K. Manditereza, "Implementing unified namespace (UNS) with MQTT Sparkplug," 2022. [Online]. Available: <https://www.hivemq.com/article/implementing-unified-namespace-uns-mqtt-sparkplug/>
- [19] —, "What is unified namespace (UNS) and why does it matter?" 2022. [Online]. Available: <https://www.hivemq.com/article/what-is-unified-namespace-uns-iiot-industry-40/>

- [20] W. Reynolds, “Digital factory mastermind: Methodology and architecture to deploy industry 4.0 solutions for manufacturers,” 2024. [Online]. Available: <https://www.iiot.university/digital-factory-mastermind>
- [21] J. A. Díaz, J. M. Rey, I. Hernández, N. Duarte, and W. Reynolds, “Unified namespace: A new paradigm for industrial IIoT data integration,” *IEEE Internet of Things Mag.*, pp. 1–10, 2025, 10.1109/MIOT.2025.3633944.
- [22] S. Domínguez-Cid, D. F. Larios, F. J. Molina, J. Luque, A. Carrasco, and C. León, “SSOT-Based laboratory on IIoT for Industry 4.0 architectures,” *XVI Congreso de Tecnología, Aprendizaje y Enseñanza de la Electrónica (TAAE)*, 2024, 10.1109/TAAE59541.2024.10604981.
- [23] P. Koprov, S. Gadhwal, A. Walimbe, X. Fang, and B. Starly, “Systems and methods for authenticating manufacturing machines through an unobservable fingerprinting system,” *Manuf. Lett.*, vol. 35, p. 1009 – 1018, 2023, 10.1016/j.mfglet.2023.08.051.
- [24] J. Mäule, T. Kutzler, T. Leich, D. Schäfer, and C. Richter, “Unified namespace and asset administration shell: A winning combination for digital production,” *2024 IEEE 29th Int. Conf. on Emerging Tech. and Factory Autom.*, pp. 1–7, 2024, 10.1109/ETFA61755.2024.10710821.
- [25] L. Freitas, M. Silva, G. Vale, C. Avram, H. Lopes, F. Pereira, N. Leal, and J. Machado, “OPC UA and MQTT performance analysis within a unified namespace context,” *Internet of Things*, vol. 33, p. 101734, 9 2025, 10.1016/J.IOT.2025.101734.
- [26] HiveMQ. (2023) How does a unified namespace (UNS) work in IIoT and Industry 4.0? [Online]. Available: <https://www.hivemq.com/blog/how-does-unified-namespace-uns-work-iiot-industry-40/>
- [27] International Organization for Standardization, “ISO/IEC 20922:2016 information technology — message Queuing Telemetry Transport (MQTT) v3.1.1,” 2016, geneva, Switzerland.
- [28] A. Banks, E. Briggs, K. Borgendale, and R. Gupta, “MQTT Version 5.0,” OASIS Open, OASIS Standard, 2019. [Online]. Available: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>
- [29] International Electrotechnical Commission, “IEC 62264-1:2013 enterprise-control system integration — part 1: Models and terminology,” 2013, edition 2.0, Geneva, Switzerland. Also published as ANSI/ISA-95.00.01.
- [30] J. A. Díaz, J. Delgado, J. M. Rey, N. Duarte, and I. Hernández, “Performance evaluation of architectural design approaches for unified namespace in industrial data systems,” *Internet of Things*, p. 102076, 2026, in Press, 10.1016/j.iot.2026.102076.
- [31] International Electrotechnical Commission, “IEC 62541-1:2025 OPC unified architecture — part 1: Overview and concepts,” 2025, geneva, Switzerland.
- [32] Eclipse Sparkplug Contributors, “Sparkplug Specification, Version 3.0.0,” Eclipse Foundation, Tech. Rep., 2023. [Online]. Available: <https://sparkplug.eclipse.org/specification/version/3.0/>
- [33] S. Anbalagan, D. Agarwal, B. Natarajan, and B. Srinivasan, “Foundational models for fault diagnosis of electrical motors,” *2023 IEEE International Conference on Power Electronics, Smart Grid, and Renewable Energy (PESGRE)*, pp. 1–6, 2023, 10.1109/PESGRE58662.2023.10404206.
- [34] T. Aasar, K. Sultan, A. U. Khan, S. Czanner, A. Abbasi, and A. Ahmad, “Predicting stator winding short-circuit faults in induction motors using machine learning: a comparative study,” *Discov. Comput.*, vol. 29, no. 1, p. 103, 2026, 10.1007/s10791-026-09984-0.
- [35] S. Ali, A. Khizhik, S. Svirin, A. Ryzhikov, and D. Derkach, “Learning to hear broken motors: Signature-guided data augmentation for induction motor diagnostics,” *Eng. Appl. Artif. Intell.*, vol. 170, p. 114137, 4 2026, 10.1016/j.engappai.2026.114137.
- [36] X. M. Liu, R. M. Zhang, J. P. Li, Y. F. Xu, and K. Li, “A motor bearing fault diagnosis model based on multi-adversarial domain adaptation,” *Sci. Rep.*, vol. 14, pp. 1–14, 12 2024, 10.1038/S41598-024-80743-1.
- [37] S. Kang, X. Tang, Y. Wang, Q. Wang, and J. Xie, “Cross-domain fault diagnosis method for rolling bearings based on contrastive universal domain adaptation,” *ISA Trans.*, vol. 146, pp. 195–207, 3 2024, 10.1016/J.ISATRA.2023.12.019.
- [38] S. E. M. P. Kone, S. B. M. Sheriff, C. Shanmugam, G. Ramdas, H. Nakamura, and Y. Mizuno, “Non-invasive induction motor fault diagnosis using PCA optimized FFT features and SVM classification,” *Measurement*, vol. 266, p. 120401, 3 2026, 10.1016/j.measurement.2026.120401.
- [39] W. Jung, S. H. Kim, S. H. Yun, J. Bae, and Y. H. Park, “Vibration, acoustic, temperature, and motor current dataset of rotating machine under varying operating conditions for fault diagnosis,” *Data Brief*, vol. 48, p. 109049, 6 2023, 10.1016/J.DIB.2023.109049.
- [40] M. Usman, T. Komatsu, K. M. Htun, Z. Liu, and A. Beck, “Benchmarking sensor modalities with few-shot domain adaptation for cross-domain fault diagnosis,” *IEEE 20th International Conference on Automation Science and Engineering (CASE)*, pp. 3219–3224, 2024, 10.1109/CASE59546.2024.10711623.
- [41] Case Western Reserve University, “Bearing Data Center Website,” <https://engineering.case.edu/bearingdatacenter>, 2018, accessed Jan., 2026.
- [42] H. Johnston, “TIG stack: Telegraf, InfluxDB and Grafana with Docker Compose,” <https://github.com/huntabyte/tig-stack>, 2026, MIT License. Accessed: Aug. 2026.
- [43] HiveMQ GmbH, “HiveMQ pricing: Fully-managed MQTT platform,” <https://www.hivemq.com/pricing/fully-managed/>, 2026, accessed: Aug. 2026.
- [44] InfluxData Inc., “InfluxDB Cloud Serverless plans and pricing,” <https://docs.influxdata.com/influxdb3/cloud-serverless/admin/billing/pricing-plans/>, 2026, accessed: Aug. 2026.
- [45] Grafana Labs, “Grafana Cloud pricing: Free, pro and enterprise plans,” <https://grafana.com/pricing/>, 2026, accessed: Aug. 2026.



Sparkplug B, and the integration of AI tools for industry applications.



analytics.



Joaquín D. López received his B.S. degree in Electronic Engineering and an M.Sc. degree in Telecommunications Engineering from the Universidad Industrial de Santander (UIS), Colombia, in 2023 and 2025, respectively. He is currently pursuing a Ph.D. degree in Electronics Engineering at UIS, where he is affiliated with the GISEL research group. In parallel, he is involved with Dautom, a technology company focused on industrial digitalization. His research interests include smart electrical microgrids, Industrial IoT (IIoT), UNS architectures with MQTT

Juliam A. Díaz received his B.S. degree in Electronic Engineering, with the highest academic distinction Summa Cum Laude, from the Universidad Industrial de Santander (UIS), Colombia, in 2023, and the M.Sc. degree in Telecommunications Engineering from the same university in 2025. He is currently a Ph.D. student in Engineering at UIS. His experience and research interests include communication protocols, data processing, the Industrial Internet of Things (IIoT), and UNS architectures for the development of data pipelines and information

Natalia Duarte is an industrial designer with over 15 years of experience in business management. She holds a specialization diploma in Senior Management from UIS. She is co-founder and Business Director of Dautom, where she leads digital transformation strategy for manufacturing and energy industries. Her research interests focus on scalable IIoT solutions for regional industry.



research areas include IIoT, UNS, Data Cloud, and scalable industrial control systems.

Iván Hernández received his B.Sc. degree in electronic engineering from Universidad Industrial de Santander (UIS), Colombia, in 2006, and a specialization diploma in industrial automation from Universidad Autónoma de Bucaramanga, Colombia, in 2016. For the last two decades, he has worked as an Industry 4.0 integrator and Digital Systems Architect, integrating SCADA, MES, IIoT, and data cloud platforms for industrial solutions. He is currently the director of engineering and lead architect at Dautom, a company he co-founded. His



Carlos A. Fajardo is a faculty professor at UIS, Colombia. He holds a Ph.D. in Engineering (High-Performance Computing), an M.Sc. in Electronic Engineering, and a postgraduate certificate in University Teaching, all from UIS. He completed a postdoctoral fellowship at the Center for Brain-Inspired Computing (C-BRIC), Purdue University, specializing in edge AI through hardware-software co-design. His research focuses on AI applied to medical problems and advanced digital systems.



Juan M. Rey received his B.S. degree in Electrical Engineering and a Specialization Diploma in University Teaching from the Universidad Industrial de Santander (UIS), Colombia, in 2012 and 2016, respectively. In 2019, he received a Ph.D. degree in Electronics Engineering from the Technical University of Catalonia, Spain. In 2022, he was honored by Germany's Federal Ministry of Education and Research (BMBF) as a Green Talent. Since 2013, he has been affiliated with the Department of Electrical, Electronic, and Telecommunications

Engineering (E3T), UIS, where he is currently an Associate Professor. His research interests include power electronics, the design and control of electrical microgrids, and the use of IoT for industrial applications.