Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Resilient Edge-Historian Framework for Industrial Automation: Adaptive Multi-Trend HMI under Intermittent Network Conditions

Muhammed Ali Erbir
emuderris@gmail.com
Kırıkkale University
Fatma Gul Amil
Kırıkkale University
Huseyin Burak Akyol
Izmir Kâtip Çelebi University
Mahmut Altun
Kırıkkale University
Ramazan Enisoğlu
Kırıkkale University
Sakine Yalman
Kırıkkale University

Research Article

Keywords: Industrial IoT, Edge Computing, Time-Series Database, Historian, HMI Trend Analysis

Posted Date: April 30th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9327528/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Resilient Edge-Historian Framework for Industrial Automation: Adaptive Multi-Trend HMI under Intermittent Network Conditions

Muhammed Ali Erbir[1*], Fatma Gul Amil[2], Huseyin Burak Akyol[3], Mahmut Altun[4], Ramazan Enisoğlu[5], Sakine Yalman[6]

[1*]Department of Motor Vehicles and Transportation Technologies, Kirikkale University, Kirikkale, Türkiye.
[2,4,5]Department of Electronics and Automation Technology, Kirikkale University, Kirikkale, Türkiye.
[3]Department of Computer Engineering, Katip Celebi University, Izmir, Türkiye.
[6]Department of Computer Technologies, Kirikkale University, Kirikkale, Türkiye.

*Corresponding author(s). E-mail(s): emuderris@gmail.com;
Contributing authors: fatmagulamil@kku.edu.tr;
huseyinburak.akyol@ikcu.edu.tr; mahmutaltun@kku.edu.tr;
ramazanenisoglu@kku.edu.tr; sakineyalman@kku.edu.tr;

Abstract

Continuous recording of process data in production automation and industrial control applications is critical for fault analysis, quality monitoring, maintenance planning, and operational reporting. However, field environments often require systems to operate under constrained network conditions, such as limited bandwidth, high latency, or even intermittent connectivity. This study presents an edge-based time-series data historian and a connection-aware HMI trend visualization approach designed to operate reliably under such intermittent connectivity conditions. In the proposed architecture, the control layer runs on a TwinCAT-based industrial computer, where process variables are periodically collected and written to a time-series database deployed at the edge through a flow-based integration layer. On the HMI side, multiple trend charts can be visualized simultaneously. Each chart supports dynamic queries through selectable variable and time-range menus. Experimental results show that for queries

retrieving 180 data points over a 24-hour period, the P95 latency is approximately 1920 ms over VPN and approximately 355 ms over LAN. With adaptive refresh mechanisms, the risk of interface timeouts is reduced. The contributions of this work include: (i) a connection-aware refresh policy that adapts the refresh period according to LAN/VPN network conditions; (ii) dynamic downsampling that produces a fixed number of values (e.g., 180) regardless of the selected time range, thereby bounding query cost; and (iii) an operational continuity mechanism that ensures automatic service recovery after power interruptions or system restarts. The proposed system has been deployed on a real industrial field automation infrastructure, and measurements were collected during live operation using actual process data. The paper presents the system architecture, data model, and a reproducible experimental evaluation methodology. Also, it reports performance results based on metrics such as writing latency, query latency, and recovery time.

Keywords: Industrial IoT, Edge Computing, Time-Series Database, Historian, HMI Trend Analysis

1 Introduction

In production automation and Industrial Control Systems (ICS), the reliable collection of process variables (PVs) and their storage as a time-indexed operational memory (historian) constitute a critical requirement. These records serve as the primary data source for functions such as fault analysis, maintenance planning, quality assurance, post-event root cause analysis, and operational reporting. With the emergence of Industry 4.0, telemetry is no longer used solely for real-time monitoring. It must also be managed in a consistent, auditable, and long-term manner to support optimization, decision-making, and safe operation. Edge computing-based real-time monitoring and optimization approaches further reinforce this requirement by emphasizing the role of field data in operational decision-making [1].

However, many industrial assets deployed in the field do not operate under stable, high-bandwidth network assumptions. Remote or distributed facilities, mobile production units, temporary deployment sites, or ICS zones with restricted network access for security reasons often operate under constrained network conditions. These environments may experience low bandwidth, high latency, packet loss, and occasional complete connectivity outages [2]. Under such conditions, centralized or cloud-centric monitoring and data-logging architectures become directly dependent on network quality. This dependency can lead to data loss, delayed logging, synchronization issues, and user-visible problems on the HMI side, such as timeouts or interface freezes. Consequently, the primary requirement is not merely to transmit data but to ensure that it is reliably collected, stored, and presented to operators locally, even under fluctuating network conditions.

Edge-based Industrial Internet of Things (IIoT) approaches and edge-cloud collaboration models have therefore become widely adopted to reduce latency, minimize bandwidth consumption, and increase tolerance to connectivity uncertainties [3],[4].

Studies aiming to improve service continuity under intermittent connectivity typically focus on resilient communication mechanisms and reliable message delivery using multi-layer communication architectures [5, 6]. Similarly, data exchange mechanisms in edge–cloud IoT systems have been investigated with respect to real-time performance and end-to-end response latency [7]. However, while resilience in the literature is often addressed primarily from the perspective of communication layers or platform orchestration [5–7], two aspects remain insufficiently addressed in combination under intermittent connectivity conditions: (i) the continuity of a time-series historian deployed at the edge and (ii) the computational and interface stability challenges associated with multi-trend visualization and refresh operations in HMI systems. In practice, operator trend screens constitute a primary decision-support component during fault diagnosis and event analysis. Therefore, visual reliability, which refers to the correct, consistent, and timely representation of process data, becomes a system property that is as critical as reliable data acquisition in field environments.

For managing time-series data, Time-Series Databases (TSDBs) have become increasingly prevalent in industrial applications due to their high write throughput and efficient time-oriented query capabilities. For a TSDB operating at the edge, metrics such as write latency, query latency, and resource consumption directly affect system continuity in field deployments [8]. Furthermore, TSDB architectures designed for edge environments highlight the need for efficient storage and query mechanisms under resource-constrained conditions [9]. In this context, simply storing time-series data at the edge is not sufficient. As the selected time range increases, the amount of data required for visualization also grows, increasing data transfer and rendering costs. Therefore, a scalable trend visualization mechanism that bounds these costs is necessary.

This study proposes an integrated approach to address these challenges. The approach includes an edge-based time-series historian and a connection-aware multi-trend HMI visualization mechanism designed for intermittent connectivity. Data acquisition is performed through periodic reads from the TwinCAT environment via the ADS protocol [10]. The collected data are written to an edge TSDB through a flow-based integration layer. On the HMI side, multiple trend charts support dynamic queries based on selectable variables and time ranges. Although imputation techniques that synthetically fill missing data points have been discussed in various domains [11], their use in critical industrial monitoring and post-event analysis may affect data interpretability and reliability. Therefore, missing data segments are not filled through imputation in this study. Instead, data gaps are preserved and represented explicitly to maintain traceability.

The proposed system was evaluated on a real industrial control and data acquisition infrastructure operating under live field conditions (an automated well data system). This setup demonstrates that the proposed approach provides measurable performance not only under laboratory conditions but also under real operational constraints. Due to commercial confidentiality and security considerations, the identities of the field sites, facility locations, operational dates, and network addresses are anonymized. Nevertheless, the experimental protocol, metric definitions, and measurement methodology are presented in a reproducible manner.

Research Questions

This study aims to answer the following research questions:

RQ1 (Edge Historian Continuity): How can periodically acquired process variables from the control layer be reliably persisted at the edge under intermittent connectivity conditions while preserving traceability, and how can historian-like continuity be ensured?

RQ2 (Connection-Aware HMI Stability): Under VPN-induced latency and bandwidth constraints, how can multi-trend visualization remain responsive, and what type of adaptive refresh–query policy should be designed to reduce interface instabilities such as timeouts or freezes?

RQ3 (Predictable Trend Scalability): Is it possible to achieve visualization latency that is independent of the selected time range by bounding query and rendering costs through a fixed number of samples, and how does such an approach affect end-to-end performance metrics?

RQ4 (Resilience and Recovery): Following power interruptions or system restarts, how should recovery performance (e.g., Mean Time to Recovery—MTTR—and data continuity) be quantified, and what architectural mechanisms can improve automatic service restoration?

The main contributions of this paper are summarized as follows:

- **Edge-based historian architecture:** An edge-oriented time-series data storage architecture is designed to enable periodic acquisition of process variables from a TwinCAT-based control environment via the ADS protocol and reliable persistence of telemetry data at the edge.
- **Connection-aware HMI refresh policy:** A connection-aware refresh policy is introduced that dynamically adjusts HMI trend refresh intervals based on LAN or VPN connectivity conditions, improving interface responsiveness and stability under constrained network environments.
- **Predictable trend scalability through dynamic downsampling:** A dynamic downsampling mechanism is proposed to bound query and rendering costs by generating a fixed number of visualization points independent of the selected time range, thereby enabling predictable TSDB query workloads.
- **Operational resilience and recovery mechanisms:** Automatic service restart and recovery mechanisms are incorporated to maintain operational continuity under power interruption and system reboot scenarios, and recovery performance is quantified using metrics such as Mean Time to Recovery (MTTR).
- **Real-world experimental validation:** The proposed approach is evaluated on a live industrial field deployment, and a reproducible experimental framework is presented including metrics such as write latency, query latency, data loss, and MTTR.

The remainder of this paper is organized as follows. Section 2 reviews the related literature and positions this work within the existing research landscape. Section 3 formulates the problem definition and system requirements. Section 4 presents the proposed architecture, including the data acquisition, integration, TSDB, and HMI components, as well as the connection-aware policy. Section 5 describes the real field

deployment scenario, experimental design, and evaluation metrics. Section 6 reports the measurement results, discusses the findings, and outlines directions for future work. Finally, Section 7 concludes the paper.

2 Related Work

Edge-based IIoT architectures have been proposed as alternatives to traditional centralized SCADA approaches in critical industrial environments such as oil and gas production [12, 13]. While these studies provide valuable insights from practical field deployments, they do not examine in detail the management of time-series data using TSDB-based architectures or the impact of intermittent connectivity on HMI trend visualization. IIoT infrastructures require continuous data flow between field devices and higher-level information systems [14]. However, constraints such as latency, bandwidth limitations, and reliability issues [9, 15] become particularly prominent in intermittent connectivity scenarios. Studies examining the infrastructure and challenges of Industry 4.0 and IIoT systems [9, 16, 17] highlight issues related to data volume, real-time processing, and system integration [18]. Furthermore, edge intelligence frameworks published by the IEC [19] emphasize the importance of positioning storage and processing capabilities between the field and cloud layers. Approaches focusing on resilient edge processing under intermittent connectivity conditions have also been widely discussed in the literature [20–22]. For example, Salonidis and Pietzuch experimentally investigate resilient edge processing mechanisms capable of tolerating connectivity disruptions [23].

Time-series databases (TSDBs) are optimized for high ingestion rates and time-oriented queries. Their use has become increasingly widespread in industrial IoT and edge computing environments. Garcia et al. [24] employed a container-based edge architecture together with TimescaleDB to collect data from manufacturing production lines. Similarly, Medeiros et al. [25] integrated OBD-II edge devices, Node-RED, and TimescaleDB for processing vehicle telemetry data. Hendrawati et al. [26] implemented a PLC- and HMI-based IoT system for real-time load monitoring. These studies demonstrate that the integration of edge-based TSDB systems with HMI components is feasible across different industrial scenarios. In edge computing contexts, the selection of a TSDB depends on factors such as write throughput, query latency, and resource consumption. Grzesik and Mrozek present a comparative evaluation of TSDB systems in edge environments [8]. Benchmarking frameworks such as IoTDB-Benchmark compare ingestion and query performance across multiple TSDB systems, including InfluxDB [27]. Similarly, TSM-Bench, published in PVLDB, evaluates TSDB performance with metrics such as data ingestion throughput and query latency [28]. TSDBs are typically designed to handle regularly arriving data streams. However, Zeng et al. [29] propose a model enabling TSDB systems to efficiently handle irregularly arriving time-series data. Some studies further utilize time-series data collected at the edge primarily for anomaly detection and predictive analytics in industrial IoT systems [30, 31]. Nevertheless, these approaches generally do not comprehensively address the long-term continuity of time-series data together with the reliable and consistent presentation of such data to operators under unstable network conditions.

Flow-based programming approaches support rapid integration between devices, APIs, and services in IoT systems. Node-RED is a widely used platform that implements this paradigm through visual data flows. Studies on the use of Node-RED in IoT development [32, 33] discuss aspects such as flow design, testability, and orchestration requirements. Within the Beckhoff TwinCAT ecosystem, the Automation Device Specification (ADS) protocol is used for data exchange and control, operating over TCP/IP [10]. The ADS/AMS specification allows ADS clients to be implemented across different platforms. Studies addressing IoT edge resilience under intermittent connectivity conditions [5] generally focus on orchestrating heterogeneous communication layers with Quality of Service (QoS) mechanisms to maximize message delivery under degraded network conditions. Architectures such as CEFIoT aim to provide fault-tolerant edge–cloud data pipelines that maintain data and processing continuity under intermittent remote connectivity and node failures [20]. In addition, Node-RED–based dynamic offloading approaches direct workflows to local or cloud resources based on IoT gateway resource metrics, ensuring service continuity under varying operational conditions [34].

Under intermittent or fluctuating network conditions, the literature on IIoT and edge computing primarily focuses on (i) communication resilience and maximizing message delivery through multiple communication layers [5, 6], (ii) edge–cloud orchestration and distributed edge management [4, 14], and (iii) fault-tolerant data processing pipelines at the edge [20, 23]. In contrast, this study systematically addresses the continuity of industrial time-series data historians at the edge, along with maintaining a stable, network-aware multi-trend HMI experience for operators.

In the context of TSDBs, prior work has focused on TSDB architectures and time-series management suitable for edge deployment [9], comparative analyses of TSDB systems in edge environments [8], and benchmarking frameworks for evaluating TSDB performance [27, 28]. However, these studies largely overlook practical industrial requirements on the HMI side, such as user experience and the stability of multi-trend refresh. Similarly, the Node-RED and flow-based integration literature addresses workflow development, testing, orchestration [32, 33], and dynamic offloading [34], but does not consider the combination of store-and-forward time-series persistence with connection-aware HMI trend visualization as a primary problem under intermittent connectivity.

Table 1 summarizes selected literature along the dimensions of "edge TSDB," "HMI/trending," "connection-aware refresh," "fixed-complexity via downsampling," "operational recoverability (MTTR)," and "field validation." The table positions the contribution of this work by highlighting its end-to-end design pattern, which integrates (i) traceable data continuity on the edge TSDB, (ii) a connection-aware HMI refresh policy, and (iii) a fixed-point target dynamic downsampling component under network fluctuations.

Table 1: Literature comparison of edge TSDB and HMI trend visualization under intermittent connectivity.

Ref. No	Main focus	Edge/IIoT architecture	TSDB	HMI	Refresh policy	Fixed complexity downsampling	Recoverability MTRR	Field validation	Identified gap
[1]	Edge-based monitoring/optimization (energy management	✓	NA	NA	–	–	–	NA	HMI behavior / TSDB continuity is secondary
[3]	Open-source IoT edge system	✓	NA	NA	–	–	–	NA	HMI stability under intermittent connectivity not addressed
[4]	Distributed edge + cloud orchestration	✓	–	–	–	–	–	NA	Orchestration not focused on HMI/TSDB historian problem
[5]	Network resiliency with multi-communication (LPWAN + WiFi)	✓	–	–	✓	–	–	NA	Refresh/HMI policy not implemented at the client level
[6]	Network resiliency (front-loading)	✓	–	–	✓	–	–	–	Extended version of [5]; not targeting HMI/TSDB axis
[7]	Cloud–edge data exchange mechanism (real-time)	✓	–	–	Secondary	–	–	NA	Data exchange; no multi-trend HMI cost considered
[8]	Comparative analysis of TSDB in edge context	–	✓	–	–	–	–	–	TSDB selection criteria provided; HMI/policy missing
[9]	Edge-oriented TSDB design (EdgeDB)	✓	✓	–	–	–	–	NA	TSDB efficiency; HMI behavior not addressed

Continued on next page

Table 1: Literature comparison of edge TSDB and HMI trend visualization under intermittent connectivity (continued).

Ref. No	Main focus	Edge/IIoT architecture	TSDB	HMI	Refresh policy	Fixed complexity downsampling	Recoverability MTRR	Field validation	Identified gap
[20]	Edge–cloud fault-tolerant IoT architecture (CEFIoT)	✓	Secondary	–	–	–	✓	NA	Platform/architecture; HMI trend semantics missing
[22]	Edge–cloud collaboration (TSN), smart industrial platform	✓	NA	–	–	–	–	NA	Time-series historian and HMI policy not integrated
[23]	Resilient edge processing	✓	–	–	✓	–	–	NA	Processing resilience; HMI refresh policy missing
[24]	Time-series monitoring and visualization architecture in production	✓	✓	✓	–	NA	–	NA	Visualization exists; connection-aware refresh & fixed-complexity missing
[25]	OBD-II edge communication framework	✓	NA	–	–	–	–	–	Time-series historian / HMI trend not the focus
[26]	Real-time monitoring/control application with PLC + HMI	✓	–	✓	–	–	–	–	HMI exists; TSDB historian + intermittent network policy missing
[27]	TSDB benchmarking (IoTDB-Benchmark)	–	✓ (benchmark)	–	–	–	–	–	Measurement framework; system policy missing
[28]	TSDB benchmarking (TSM-Bench)	–	✓ (benchmark)	–	–	–	–	–	Measurement framework; HMI/edge policy missing

Continued on next page

Table 1: Literature comparison of edge TSDB and HMI trend visualization under intermittent connectivity (continued).

Ref. No	Main focus	Edge/IIoT architecture	TSDB	HMI	Refresh policy	Fixed complexity downsampling	Recoverability MTRR	Field validation	Identified gap
[32]	Node-RED IoT system development/testing	✓	–	–	–	–	–	–	Integration engineering; historian + HMI policy not addressed
[33]	Node-RED workflow orchestration	✓	–	–	–	–	–	–	Orchestration; trend refresh under intermittent connectivity not implemented
[34]	Node-RED–based dynamic offloading	✓	–	–	–	–	–	–	Offloading; TSDB historian + HMI trending not included
[This Study]	Edge TSDB historian with connection-aware multi-trend HMI	✓	✓	✓	✓	✓	✓	✓	End-to-end: data continuity + HMI stability + MTTR measurement

Note: NA indicates that the corresponding feature is not explicitly stated in the referenced study.

While most of the literature focuses on communication, orchestration, or TSDB performance, this study places the stability of multi-trend HMI refresh and predictable query/rendering costs at the core of the system design.

3 Problem Definition and System Requirements

3.1 Problem Definition

In the target scenario, the control system operates continuously at the edge layer. However, due to field conditions, the remote access link, particularly the VPN connection, may intermittently exhibit low bandwidth, high round-trip time (RTT) and jitter, or complete disconnections. Under such constraints, centralized or cloud-centric architectures render telemetry performance dependent on network quality, thereby increasing the risk of data loss and delayed logging. In addition, during simultaneous multi-trend updates on the operator interface (HMI), the cumulative cost of query processing, data parsing, and rendering may significantly increase. This behavior can lead to timeouts, request queuing, and instability phenomena such as interface freezes or degraded responsiveness. Therefore, the problem extends beyond reliable data transmission and involves: (i) process variables must be persistently stored on the edge as a historian, ensuring monitoring and control capabilities independent of network availability; (ii) under varying connectivity conditions (e.g., LAN, VPN, or offline modes), the HMI trend refresh rate and query workload must be adaptively regulated to preserve interface stability; and (iii) following power interruptions or system restarts, both data acquisition and time-series database (TSDB) services must automatically recover without human intervention to maintain operational continuity. Furthermore, as the selected time window increases, the associated data transfer and rendering costs grow proportionally. Maintaining these costs within a predictable upper bound is a fundamental requirement for scalable multi-trend visualization.

Therefore, the proposed system is designed to satisfy the functional and operational requirements listed and described in Table 2.

4 Proposed Architecture and Design

4.1 Architecture

The proposed system consists of four main components, as illustrated in Figure 1. First, the control layer is implemented on a Beckhoff industrial PC running TwinCAT. This layer is responsible for process control execution and the management of symbolic variables within the automation environment. Second, the data acquisition and integration layer is implemented using Node-RED with an ADS client interface. This layer performs periodic data reading from the control system, followed by validation and tagging operations prior to storage. Third, the edge time-series database (TSDB) layer is realized using InfluxDB OSS v2. This component enables persistent storage and efficient querying of timestamped process data at the edge, independent of external network connectivity. Finally, the HMI client layer provides multi-trend

Table 2 Functional and operational requirements that are satisfied by the proposed system.

ID	Title	Description
R1	Data Acquisition	Process variables shall be periodically acquired from the control computer at a predefined sampling interval. The acquisition mechanism shall ensure deterministic and continuous data collection during runtime
R2	Time-Series Storage	The acquired data shall be timestamped and persistently stored in a time-series database (TSDB). The storage mechanism shall preserve temporal consistency and data integrity
R3	Visualization	The HMI shall support the concurrent display of five trend graphs. For each graph, operators shall be able to select process variable(s) and time intervals. Visualizations shall be generated dynamically through runtime queries to the TSDB
R4	Connectivity Awareness	Distinct automatic refresh intervals shall be applied under LAN and VPN conditions. The system shall adapt refresh behavior according to network characteristics to maintain interface responsiveness and stability
R5	Operational Continuity	Following power interruptions or system restarts, both the data acquisition and TSDB writing services shall automatically resume without manual intervention
R6	Reproducibility	Field- and facility-specific details shall be anonymized due to commercial confidentiality. However, the experimental protocol, performance metrics, and baseline configuration parameters shall be reported in sufficient detail to ensure reproducibility

visualization capabilities. The interface supports dynamic selection of process variables and time intervals, as well as connectivity-aware refresh mechanisms that adapt to varying network conditions.

The system has been deployed on a real industrial field automation infrastructure. All measurements were conducted under live operational conditions using real-time process data. Due to field confidentiality constraints, well or facility identifiers, location information, network addresses, and similar operational details are not disclosed.

4.1.1 Control Layer

The control layer is implemented on an industrial PC running TwinCAT developed by Beckhoff (hereafter referred to as TwinCAT IPC). This layer is responsible for real-time process monitoring and closed-loop control functions. Field I/O modules are mapped directly to the TwinCAT runtime environment, and control logic is executed within deterministic, cycle-time-driven tasks configured to satisfy the timing constraints of the target industrial process. Within the proposed architecture, the control layer is positioned as the primary determinant of operational correctness. Process variables (PVs), control outputs, interlocks, and critical internal states are maintained with strict real-time priorities, independently of historian or HMI workloads. This separation ensures that telemetry-related operations do not interfere with control performance.

To prevent any adverse impact of data extraction on control determinism, the control layer exposes selected symbolic variables to external components via the ADS communication interface over TCP/IP. The exported variable set is deliberately constrained to represent essential analog PVs, critical control and status signals, and

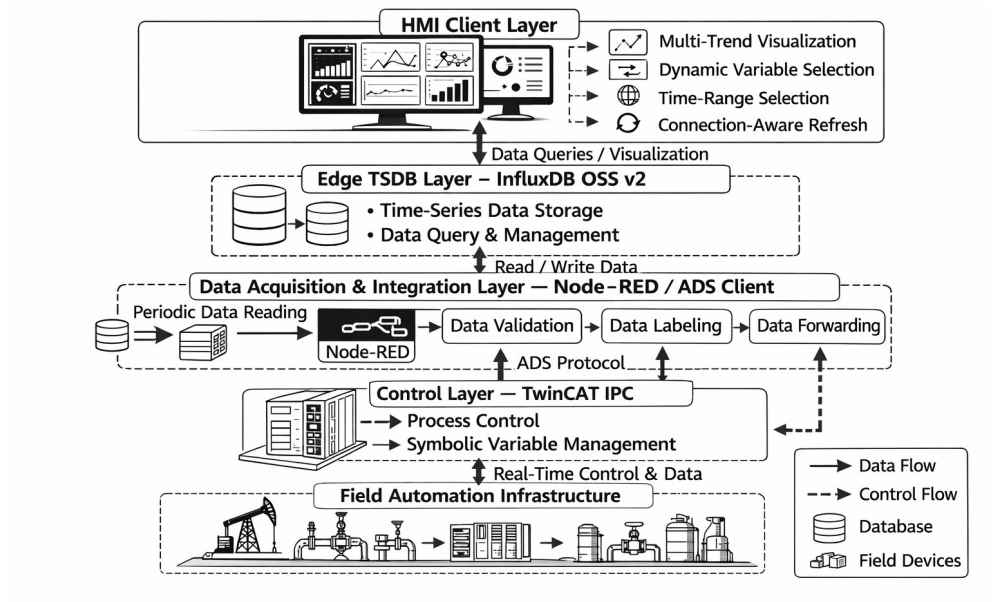


Fig. 1 Overview of the proposed system architecture.

key indicators required for fault diagnosis and post-event analysis. The historian and HMI subsystems operate without modifying the control program. Data acquisition is performed in a read-only manner and at a lower sampling frequency relative to real-time control tasks (e.g., 1 Hz in this study). This design choice minimizes jitter and cycle-time deviations while preserving sufficient observability for operator trend visualization and system monitoring. Furthermore, the control layer provides a consistent namespace and engineering semantics for downstream components, including standardized signal naming conventions, scaling definitions, and unit specifications. The subsequent data acquisition and integration layer connects to the ADS endpoint, periodically reads the symbolic variables, performs validation and tagging operations, and forwards timestamped samples to the edge historian pipeline.

4.1.2 Data Acquisition and Integration Layer

ADS is the native data exchange protocol used in TwinCAT systems [10]. Any device within the local area network can establish communication with a TwinCAT controller via the ADS protocol, as exemplified in the configuration illustrated in Figure 2 This communication mechanism enables structured and symbolic access to runtime variables without interfering with real-time execution tasks. In the present study, symbolic variables were periodically read over ADS within the integration layer and transferred together with their corresponding timestamps. The selected variable set represents analog process variables (PVs) monitored in the field automation system and reflects a realistic industrial deployment scenario.

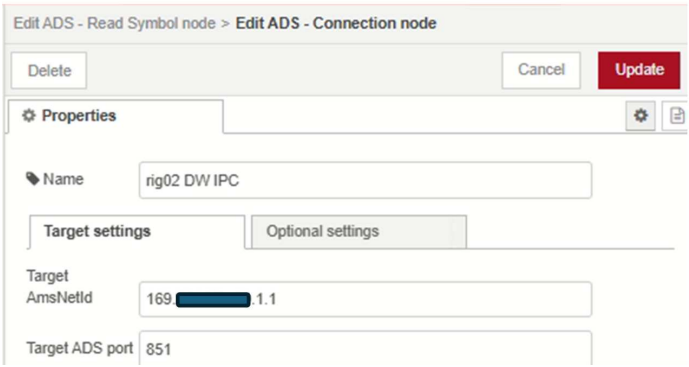


Fig. 2 ADS client connection and configuration screen.

The integration layer is implemented using Node-RED, which supports rapid prototyping in IIoT applications through a flow-based programming paradigm [32, 33]. As illustrated in Figure 3, Node-RED coordinates periodic triggering mechanisms to acquire symbolic variables via ADS. The acquired data are subsequently subjected to validation and format transformation procedures to ensure structural consistency and semantic correctness. Following preprocessing, the samples are mapped into the time-series database schema, including measurement names, tags, and field definitions, and are then written to InfluxDB OSS v2 for persistent storage. Through this modular architecture, the data acquisition and integration layer effectively decouples the real-time control environment from storage and visualization services. This separation preserves control determinism while simultaneously enabling structured, scalable, and query-efficient persistence of process data.

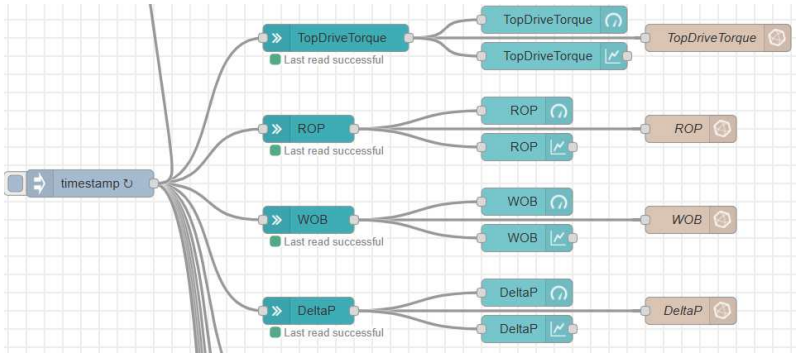


Fig. 3 Node-RED flow for periodic ADS data acquisition and forwarding to the edge TSDB.

4.1.3 Edge time-series database (TSDB) layer

InfluxDB OSS v2 is an open-source time-series database (TSDB) platform designed for high write throughput and low-latency queries, enabling efficient storage and analysis

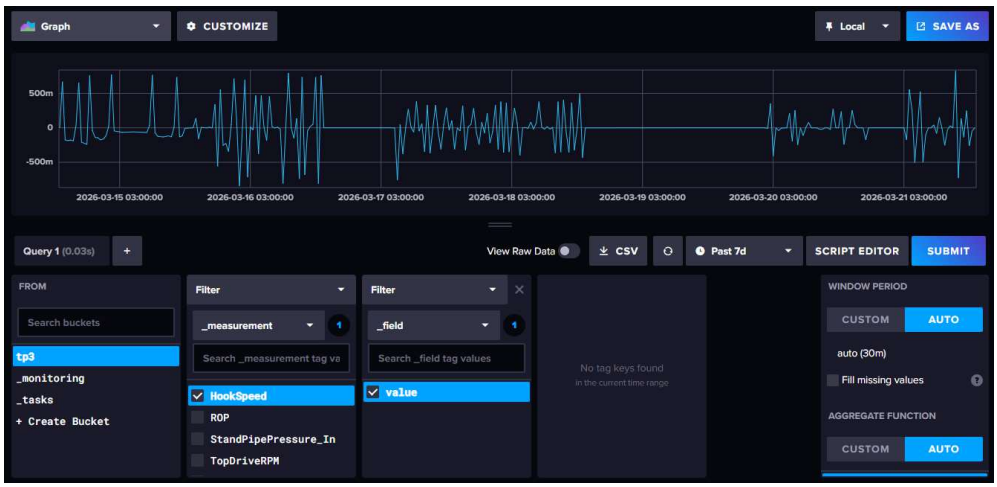


Fig. 4 InfluxDB interface showing a representative telemetry data interval.

of time-series data [35]. Figure 4 illustrates the InfluxDB interface for a representative data interval used in this study. The edge TSDB is deployed within the local area network to ensure low-latency access and independence from external connectivity.

The database schema is designed to facilitate separation of multiple subsystems, tag-based query filtering, and dynamic variable selection on the HMI. The example data model supports consistent, flexible, and scalable management of measurements. In this configuration, the measurement field is defined as telemetry, while provision is made for creating separate measurements on a per-signal basis when needed. The tag structure includes unit_id, subsystem, and signalgroup fields, enabling subsystem differentiation, tag-based querying, and dynamic filtering. The field component stores the primary measurement value under signal_value; additional fields may be defined depending on the signal type and application requirements. Timestamps are assigned based on the periodic read operations performed by Node-RED. When available, timestamps from the control layer may be included to enhance synchronization and temporal consistency. This design ensures that the edge TSDB provides a structured, high-performance, and scalable repository suitable for both real-time visualization and downstream analysis.

4.1.4 HMI client layer

The HMI provides five concurrently observable trend graphs, as illustrated in Figure 5. For each graph, the operator can select the process variable to be monitored via a pull-down menu and define the corresponding time interval using an additional selection menu. Based on these user-defined parameters, the relevant data are dynamically queried from the TSDB, and the resulting time-series samples are rendered in real time on the corresponding graphs.

This design enables operators to monitor multiple process variables simultaneously on a single screen and to perform rapid inspections over different time windows. During

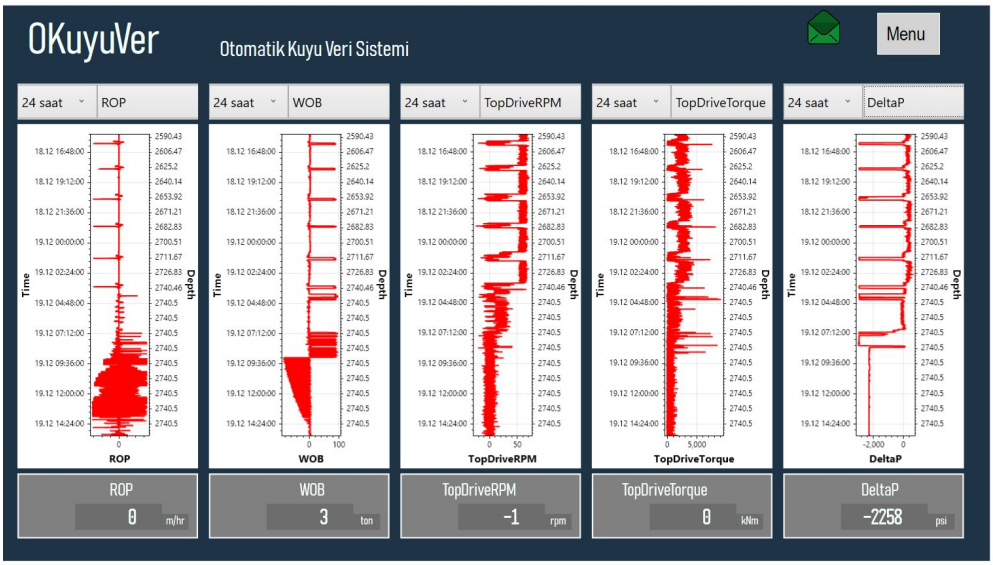


Fig. 5 Multi-trend HMI interface used in the study.

time interval selection, the sampling period within the window is determined according to the relationship defined in Equation 1.

$$\text{windowPeriod} = \frac{t_{\text{end}} - t_{\text{start}}}{N_{\text{target}}} \tag{1}$$

For database queries spanning large time intervals, the system minimizes data load by computing average values over the selected period, as expressed in Equation 2. This approach ensures efficient rendering while maintaining sufficient fidelity for operational monitoring and analysis.

$$\text{aggregateWindow}(\text{every}: \text{windowPeriod}, \text{fn}: \text{mean}, \text{createEmpty}: \text{false}) \tag{2}$$

4.2 Connectivity-Aware Refresh Policy

In remote access scenarios, particularly over VPN, increased round-trip time (RTT), jitter, and limited bandwidth can significantly prolong the end-to-end update time for multiple simultaneous trend graphs on the HMI (five graphs in this study), including TSDB query execution, data parsing, and rendering. When the refresh period is set below the total update duration, subsequent refresh cycles may be triggered before the previous cycle completes, potentially causing queuing, timeouts, and other interface instabilities on the client side. To address this, the refresh mechanism in the proposed system is not implemented as a fixed-period update. Instead, it is designed as a state-machine-based policy that continuously monitors network quality and the most

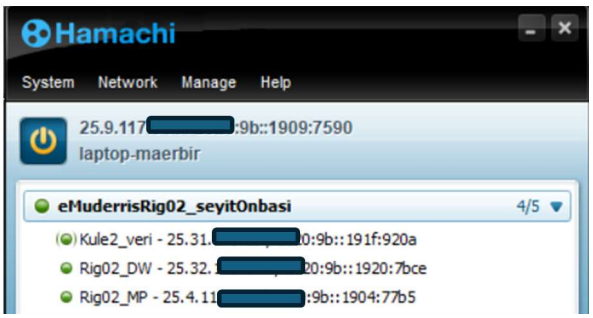


Fig. 6 VPN client application used for secure remote access.

recently observed update cost. Figure 6 presents a representative HMI screenshot; secure remote access over VPN was provided without additional cost.

The policy operates online by tracking the metrics including Tquery and Trender, where the former represents the time required to retrieve data from the TSDB; while the latter represents the time required to render the retrieved values on the HMI. The total duration of a refresh cycle for five trend graphs is defined in Equation 3.

$$T_{\text{update}} = \sum_{i=1}^{5} \left(T_{\text{query}}^{(i)} + T_{\text{render}}^{(i)} \right) \tag{3}$$

These metrics are used to adjust the refresh period and query timeout parameters in a deadline-aware manner, ensuring that interface responsiveness is maintained even under varying network conditions.

4.3 State Machine

The refresh policy treats the LAN as the primary path and the VPN as the secondary (fallback) path. Initially, the system always attempts the LAN connection. After a predefined number of consecutive LAN failures, the policy switches to VPN. During VPN queries, any failure in the first cycle triggers an immediate return to LAN. For all failed attempts, the retry interval is increased according to an exponential backoff strategy, as defined in Equation 4, with an upper limit of 300 seconds.

$$T_{\text{retry}}(n) = \min\left(T_0 \cdot 2^n,\ 300\ \text{s}\right) \tag{4}$$

In this policy, n represents the number of consecutive failed attempts (either a single counter for all modes or separate counters per mode), and T0 is the base interval (10 s). Upon a successful update, the counter is reset to zero. The complete state-machine diagram is presented in Figure 7 and the descriptions of the states composing the state machine are listed in Table 3.

4.4 Operational Continuity: Automatic Startup and Recovery

Field environments may experience power outages, or unexpected system restarts. Consequently, the automatic restart of the TSDB and integration layer services is critical to maintaining continuous data acquisition and HMI functionality. In this study,

Table 3 States of connectivity-aware HMI refresh policy and their descriptions.

ID	Title	Description
S0	LAN_ACTIVE	Normal refresh over LAN with standard (fast) refresh period
S1	LAN_BACKOFF	Retry attempts under LAN failure conditions; the retry interval grows exponentially. A fail_lan counter is maintained
S2	VPN_ACTIVE	Queries are performed over the VPN (secondary path). On VPN failure, an immediate return to LAN is attempted
S3	OFFLINE	Both LAN and VPN paths are unresponsive. The "last known data" is displayed, and retry intervals are scheduled with exponential backoff (≤ 300 s)

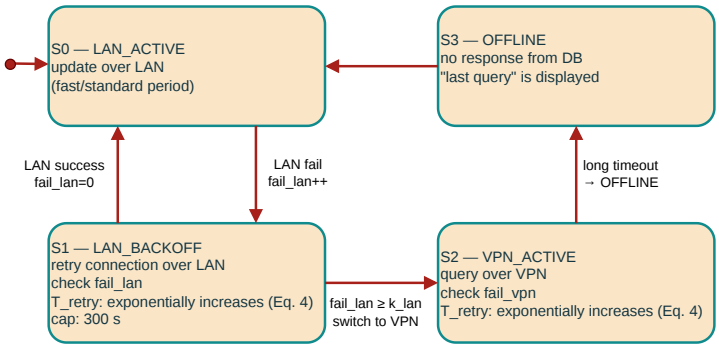


Fig. 7 State machine of the connectivity-aware HMI refresh policy.

the system is configured to automatically start the relevant services during system boot, ensuring that data logging and HMI query capabilities are restored without human intervention. As illustrated in Figure 8, on Windows operating systems, the necessary startup configurations are applied so that, following a system restart and user login, the data acquisition and database components are automatically launched. The corresponding execution script is registered in the Startup folder to guarantee seamless operational recovery.

5 Experimental Setup and Assessment Method

The experimental evaluation was conducted on an operational automated well control and data acquisition system. For field confidentiality, device make/model information and network addresses have been anonymized. The setup comprises a TwinCAT-based control computer with ADS access, providing real-time process and control variables. At the edge, a database node consisting of InfluxDB OSS v2 and Node-RED components is deployed on the local area network, handling time-series data storage and integration layer functions. Operator access is provided through HMI clients with multi-trend visualization capabilities running on a Panel PC. The network infrastructure is primarily based on an on-site of LAN, with remote access enabled over the Internet via a vendor-independent VPN solution.

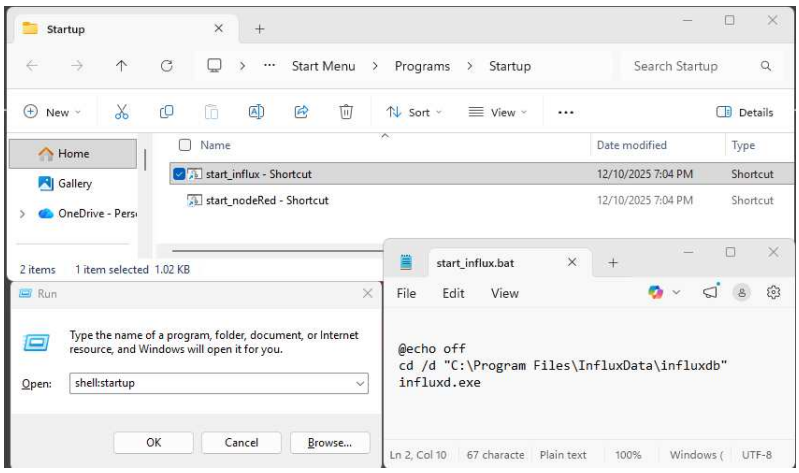


Fig. 8 Automatic restart configuration for OKUYUVER background services after system reset/restart.

The field system model is depicted in Figure 9. Experimental evaluation was conducted across four scenarios to investigate system behavior under different network and operational conditions. In the first scenario, the HMI sent queries to the TSDB and rendered results under LAN conditions, with an automatic refresh period of 10 seconds. In the second scenario, the same query and rendering process was performed under remote VPN access, with a refresh period set to 60 seconds. The third scenario examined a network outage situation during which data writing continued solely at the edge. Remote querying was suspended, and the system automatically resumed normal operation upon network restoration. In the fourth scenario, the edge node was restarted, and recovery times (MTTR) were measured.

Consistent with typical metrics used in the TSDB and edge computing literature [8], the following evaluation metrics were selected:

- M1 – Query latency (ms): Time elapsed from HMI query initiation to dataset retrieval.
- M2 – Rendering time (ms): Time elapsed from dataset retrieval to graph rendering.
- M3 – Recovery time (MTTR): Time from system restart to first successful data write and first successful graph rendering.

During the experiments, the edge node system clock was automatically synchronized via NTP to minimize timestamp drift and ensure temporal consistency of the measurements.

6 Results and Discussion

This section presents the performance results of the proposed edge historian and connectivity-aware multi-trend HMI approach under real field conditions and discusses the design decisions in light of experimental findings. The evaluation was conducted across multiple operational scenarios, including LAN access, VPN access, network

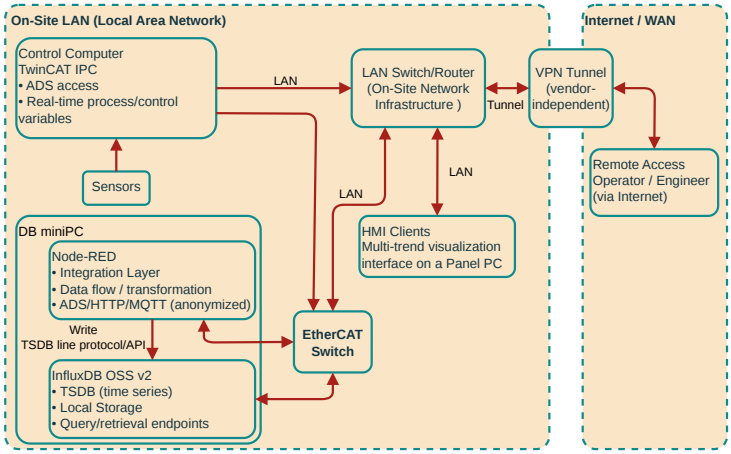


Fig. 9 Anonymized field deployment topology used in the experiments.

interruption, and system restart (power-cycle) conditions. Performance was analyzed using three primary metrics: query latency (M1), rendering time (M2), and recovery time (M3/MTTR). These metrics were selected to jointly assess data access performance, visualization responsiveness, and operational resilience of the end-to-end system architecture.

6.1 LAN and VPN Conditions: HMI Query and Rendering Performance

In the performance comparison under LAN and VPN conditions, the target number of data points per time window was empirically set to approximately 180 to ensure predictable visualization cost independently of the selected time range. This design choice provides sufficient visual resolution for operator perception while constraining both the data volume transferred from the TSDB and the client-side processing and rendering overhead. P50 and P95 statistics were computed over N = 200 independent measurements. All reported results represent the aggregated load of five concurrent trend graphs within the same visualization window.

The query and rendering performance during the initial HMI loading phase is summarized in Table 4 For a 24-hour time range with a target of approximately 180 points, the P95 query latency (M1) was measured as 355 ms under LAN conditions and 1920 ms over VPN. In contrast, the P95 rendering time (M2) remained at 44 ms for LAN and 43 ms for VPN. These results indicate that, particularly during the first load, the dominant contributor to total latency is the network and query chain rather than client-side rendering overhead.

Performance results for re-rendering the same time window while the HMI remained active are presented in Table 5 Compared to the initial load, latencies decreased substantially in this scenario. The P95 query latency (M1) was 140 ms under LAN and 440 ms over VPN, while the P95 rendering time (M2) remained at

Table 4 Initial HMI query and rendering performance under LAN and VPN conditions for a 24-hour time window (one window, five graphs).

For 1 Window (5 Graphs)							
Conn.	**Interval**	**Data Points**	**Refresh Rate**	**M1 (P50)**	**M1 (P95)**	**M2 (P50)**	**M2 (P95)**
LAN	24 Hours	180	10s	200ms	355ms	31ms	44ms
VPN	24 Hours	180	60s	1746ms	1920ms	32ms	43ms

approximately 9 ms in both cases. The findings demonstrate that client-side visualization overhead during re-rendering is both low and stable, and that performance differences are primarily attributable to connection-dependent query latency.

Table 5 Re-rendering performance under LAN and VPN conditions for the same 24-hour time window while the HMI remains active (one window, five graphs).

				Query Latency		Rendering Time	
Conn.	**Interval**	**Data Points**	**Refresh Rate**	**M1 (P50)**	**M1 (P95)**	**M2 (P50)**	**M2 (P95)**
LAN	24 Hours	180	10s	70ms	140ms	6ms	9ms
VPN	24 Hours	180	60s	361ms	440ms	6ms	9ms

These observations directly support the rationale for the proposed connectivity-aware refresh policy. Due to increased latency under VPN access, refresh intervals suitable for LAN environments (e.g., 10 s) may not be safe under all conditions. Transitioning to longer refresh intervals under VPN (e.g., 60 s) was observed to prevent overlapping update cycles, thereby reducing the risk of queuing effects, timeouts, and interface instability. In other words, treating the refresh period as a connectivity-sensitive parameter rather than a single fixed value constitutes a critical design factor for maintaining HMI stability. Furthermore, the use of a dynamic windowPeriod + aggregateWindow query strategy enabled the system to maintain an approximately constant number of retrieved samples even when users selected different time ranges. This approach bounds (i) the data volume transferred from the TSDB, (ii) client-side parsing and processing overhead, and (iii) rendering cost within a predictable range, thereby enhancing the scalability of multi-trend visualization. For the analyzed signals, the consistently low and stable rendering cost across varying time intervals aligns with the intended fixed-complexity design principle.

6.2 Power-Cycle Recovery and Operational Sustainability

Field environments are inherently subject to power outages and unplanned restarts; therefore, the ability of the system to autonomously resume operation without human

intervention constitutes a critical requirement. To address this, the proposed design ensures automatic startup of the data acquisition and TSDB components upon operating system boot. Recovery performance was explicitly treated as a measurable metric within the experimental evaluation. Recovery times following a full system restart are summarized in Table 6. Measurements indicate that after rebooting the database host, system components become operational within the order of minutes, and the first successful HMI rendering is achieved in approximately 159 s. The reported restart durations for the Node-RED service, the InfluxDB service, and the TwinCAT service further clarify the relative contribution of individual components within the recovery chain.

These findings demonstrate that the proposed architecture supports operational continuity not only under steady-state conditions but also in restart scenarios, thereby satisfying practical resilience requirements encountered in real field deployments.

Table 6 Recovery-related latencies measured after system restart.

Event	**M3 (s)**
Database PC restart	56
Node-RED service restart	162
InfluxDB service restart	148
First rendering of HMI	159
TwinCAT service start	144

6.3 Discussion of Design Decisions in Light of Experimental Findings

The experimental results substantiate the three principal design decisions adopted in this study.

First, maintaining the authoritative copy of process data at the edge within a TSDB aligns with the objective of ensuring historian continuity independently of network quality fluctuations. This architectural choice enables uninterrupted data acquisition and storage at the field level even when remote access experiences latency degradation or temporary disconnections. The findings confirm that decoupling data persistence from wide-area connectivity enhances operational robustness in practical deployments.

Second, the Node-RED–based flow architecture integrates data acquisition, validation, tagging, and TSDB write operations within a modular and manageable processing pipeline. Field operation results demonstrate that this layer provides a practical engineering solution capable of sustained continuous operation. The observed stability under real-world conditions reinforces that the proposed architecture is not merely conceptual but deployable and maintainable in live industrial environments.

Third, the combination of a connectivity-aware HMI refresh policy and a fixed-point-target aggregation strategy experimentally validates the decision to place

interface stability at the center of the design. Although query latency increases significantly under VPN conditions, rendering cost remains low and stable. With appropriate adaptation of the refresh interval, interface behavior remains consistent and free from instability phenomena such as update overlap or timeout-induced blocking. This observation directly supports the study's central argument that resilience should be considered not only at the communication infrastructure level but also at the level of operator-perceived visualization continuity.

Overall, the results exhibit a performance profile consistent with the objectives defined under RQ1–RQ4: edge historian continuity has been achieved; a connectivity-aware stability strategy for multi-trend HMI has been successfully implemented; query and rendering cost have been bounded through a fixed-point aggregation principle; and post-restart recovery performance has been quantitatively characterized.

7 Conclusions

This study presented an end-to-end architecture that integrates an edge historian approach with a connectivity-aware HMI refresh policy to maintain industrial process data continuity and stable multi-trend visualization under intermittent and fluctuating network conditions. The proposed framework comprises periodic data acquisition from a TwinCAT-based control layer via ADS, a Node-RED–based integration and tagging pipeline, an InfluxDB OSS v2–based edge TSDB storage layer, and dynamically queried multi-trend HMI components. Unlike laboratory-scale validation, the system was evaluated during live operation on a real-world automated well data system under actual field conditions.

The experimental results demonstrate the practical feasibility of the proposed approach and its alignment with stability objectives. For a 24-hour time window with an approximate target of 180 data points, the P95 query latency was measured at 355 ms under LAN conditions and 1920 ms over VPN. In contrast, rendering latency remained relatively low and stable, supporting the effectiveness of the fixed-point-target aggregation strategy in bounding client-side rendering cost. In re-rendering scenarios with the HMI already active, query latency further decreased (LAN P95 140 ms; VPN P95 440 ms), while rendering time remained consistently low. These findings confirm that the connectivity-aware refresh policy provides a practical and effective foundation for maintaining interface stability under varying access conditions. Additionally, power-cycle experiments showed that services automatically restarted after system reboot and that the first successful HMI rendering was re-established within the order of minutes, demonstrating unattended recovery capability.

The primary contribution of this work lies in defining resilience not solely in terms of network transport or platform orchestration, but also through the lens of operator-perceived interface stability and visualization continuity. In this respect, the study unifies edge TSDB historian continuity, connectivity-aware HMI refresh management, fixed-complexity downsampling, and MTTR-oriented recovery evaluation within a single field-validated architectural pattern.

Several limitations should be acknowledged. Due to commercial confidentiality and field security considerations, raw datasets and facility identifiers could not be disclosed.

Furthermore, the evaluation was conducted on a single real-world deployment with a specific software and configuration stack. The 1 Hz sampling rate, while adequate for the examined application, may not satisfy higher-frequency industrial scenarios. Nevertheless, architectural components, experimental procedures, and performance metrics have been described in sufficient detail to support reproducibility.

Future work will focus on extending the edge historian layer with long-term retention policies, multi-level downsampling strategies, change-based logging, and repeated-value suppression techniques. The connectivity-aware HMI policy will be refined using finer-grained network estimation mechanisms, including RTT, jitter, and bandwidth-aware adaptation with anti-flapping hysteresis control. Additional enhancements may include secure replication with central systems, alarm/event correlation, and online anomaly detection. Comparative evaluations across alternative TSDB platforms are also planned to better characterize the generalizability boundaries of the proposed design.

Declarations

- **Author Contributions:** Muhammed Ali Erbir: Conceptualization, methodology, system design, software implementation, investigation, data interpretation, visualization, and writing—original draft. Fatma Gul Amil: Methodology, validation, and writing—review and editing. Huseyin Burak Akyol: Technical review, validation, and writing—review and editing. Mahmut Altun: Investigation, resources, and writing—review and editing. Ramazan Enisoğlu: Formal analysis, visualization, and writing—review and editing. Sakine Yalman: Supervision, project administration, and writing—review and editing. All authors read and approved the final manuscript.
- **Ethics Approval:** Not applicable. This study does not involve human participants, animals, or identifiable personal data.
- **Informed Consent:** Not applicable.
- **Clinical Trial Number:** Not applicable.
- **Conflict of Interest:** The authors declare that they have no conflict of interest.
- **Data Availability:** Due to commercial confidentiality and field security considerations, raw field data cannot be publicly shared. However, the experimental protocol, performance metrics, and anonymized evaluation methodology are described in sufficient detail to support reproducibility.
- **Confidentiality and Anonymization:** Field ownership, location details, operational dates, well/facility identifiers, and network addresses have been masked or anonymized.
- **Permissions and Acknowledgments:** The study was conducted on a live field system with the permission of the system owner/operator. The authors gratefully acknowledge KOTAR Bilişim Ltd. Şti. for their technical support.
- **Use of Artificial Intelligence Tools:** AI-assisted tools were used solely for language editing and formatting purposes during manuscript preparation. The final content, technical decisions, and interpretations are the sole responsibility of the authors.

- **Code Availability:** The source code used in this study can be made available upon reasonable request.

Funding

Not applicable.

Ethics, Consent to Participate, and Consent to Publish

Ethics Approval Not applicable. The study does not involve human participants, animals, or identifiable personal data. Experiments were conducted on a real industrial petroleum well control and data acquisition system with the permission of KOTAR Bilişim. All field-specific identifiers and sensitive operational details were anonymized.

Consent to Participate Not applicable.

Consent for Publication Not applicable.

References

[1] Huang, J., Zhou, S., Li, G., Shen, Q.: Real-time monitoring and optimization methods for user-side energy management based on edge computing. Scientific Reports **15**(1), 24890 (2025)

[2] Bollam, A.: Emerging trends in data synchronization for edge computing. Journal of Computer Science and Technology Studies **7**(5), 102–108 (2025)

[3] Romero, D.A.V., Laureano, E.V., Betancourt, R.O.J., Álvarez, E.N.: An open source iot edge-computing system for monitoring energy consumption in buildings. Results in Engineering **21**, 101875 (2024)

[4] Martins, T.W., Rambo, F., Viezens, C., Clauer, E., Sandner, A., McConnell, T., Mirtov, B., Noce, R., Karnouskos, S.: Driving Industry 4.0 at distributed edges with cloud orchestration. IIC Journal of Innovation (2021). Accessed 2026-03-06

[5] Kumar, V., Yadav, P., Indrusiak, L.S.: Resilient edge: Building an adaptive and resilient multi-communication network for iot edge using lpwan and wifi. IEEE Transactions on Network and Service Management **20**(3), 3055–3071 (2023)

[6] Kumar, V., Yadav, P., Indrusiak, L.S.: Resilient edge: Can we achieve network resiliency at the iot edge using lpwan and wifi? arXiv preprint arXiv:2205.03729 (2022)

[7] Miao, W., Zeng, Z., Huang, J., Li, S., Xia, Y., Bi, S., Wang, X.: Data exchange mechanism for real-time object detection in cloud-edge iot system. Wireless Communications and Mobile Computing **2023**(1), 3842738 (2023)

[8] Grzesik, P., Mrozek, D.: Comparative analysis of time series databases in the context of edge computing for low power sensor networks. In: International Conference on Computational Science, pp. 371–383 (2020). Springer

[9] Yang, Y., Cao, Q., Jiang, H.: Edgedb: An efficient time-series database for edge computing. IEEE Access **7**, 142295–142307 (2019)

[10] Beckhoff: Beckhoff Information System - ADS. https://infosys.beckhoff.com/english.php?content=../content/1033/tcinfosys3/11291871243.html. Accessed: 2025-12-16 (n.d.)

[11] Gkoulis, D., Tsadimas, A., Kousiouris, G., Bardaki, C., Nikolaidou, M.: Exploring the performance of real-time data imputation to enhance fault tolerance on the edge: A study on environmental data. Simulation Modelling Practice and Theory, 103178 (2025)

[12] Ramzey, H., Badawy, M., Elhosseini, M., A. Elbaset, A.: I2ot-ec: A framework for smart real-time monitoring and controlling crude oil production exploiting iiot and edge computing. Energies **16**(4), 2023 (2023)

[13] Wang, X., Zhang, Y., Jin, L., Ni, J., Zhu, Y., Cao, W., Jiang, X.: Design of an integrated gate irrigation system with measurement and control based on cloud-edge-end collaboration and fuzzy algorithm. Computers and Electronics in Agriculture **239**, 111035 (2025)

[14] Nie, Q., Tang, D., Liu, C., Wang, L., Song, J.: A multi-agent and cloud-edge orchestration framework of digital twin for distributed production control. Robotics and Computer-Integrated Manufacturing **82**, 102543 (2023)

[15] Elsisi, M., Amer, M., Ali, M.N., Su, C.-L.: A resilient iot-enabled framework using hybrid decision tree and wavelet transform for secure and sustainable photovoltaic energy management. Sustainable Computing: Informatics and Systems, 101221 (2025)

[16] Akande, J.O., Chukwunweike, J.: Developing scalable data pipelines for real-time anomaly detection in industrial IoT sensor networks. International Journal of Engineering Technology Research & Management (IJETRM) **7**(12), 497–513 (2023)

[17] Papcun, P., Kajati, E., Cupkova, D., Mocnej, J., Miskuf, M., Zolotova, I.: Edge-enabled iot gateway criteria selection and evaluation. Concurrency and Computation: Practice and Experience **32**(13), 5219 (2020)

[18] Ahmed, S.F., Alam, M.S.B., Hoque, M., Lameesa, A., Afrin, S., Farah, T., Kabir, M., Shafiullah, G., Muyeen, S.: Industrial internet of things enabled technologies, challenges, and future directions. Computers and Electrical Engineering **110**, 108847 (2023)

[19] International Electrotechnical Commission (IEC): Edge Intelligence. https://www.iec.ch/basecamp/edge-intelligence. Accessed: 2026-03-06 (n.d.)

[20] Javed, A., Heljanko, K., Buda, A., Främling, K.: Cefiot: A fault-tolerant iot architecture for edge and cloud. In: 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), pp. 813–818 (2018). IEEE

[21] Enemosah, A., Chukwunweike, J.: Next-generation scada architectures for enhanced field automation and real-time remote control in oil and gas fields. Int J Comput Appl Technol Res **11**(12), 514–29 (2022)

[22] Wang, Y., Yang, S., Ren, X., Zhao, P., Zhao, C., Yang, X.: Industedge: A time-sensitive networking enabled edge-cloud collaborative intelligent platform for smart industry. IEEE Transactions on Industrial Informatics **18**(4), 2386–2398 (2021)

[23] O'Keeffe, D., Salonidis, T., Pietzuch, P.: Frontier: Resilient edge processing for the internet of things. Proceedings of the VLDB Endowment **11**(10), 1178–1191 (2018)

[24] Garcia, A., Oregui, X., Franco, J., Arrieta, U.: Edge containerized architecture for manufacturing process time series data monitoring and visualization. In: IN4PL, pp. 145–152 (2022)

[25] Medeiros, M., Azevedo, M., Amaral, M., Viegas, C.M., Silva, M., Silva, I., Costa, D.G.: A framework for efficient communication in iot applications in the automotive industry with obd-ii edge. In: 2023 Symposium on Internet of Things (SIoT), pp. 1–5 (2023). IEEE

[26] Hendrawati, D., Hakim, E.A.: Iot enabled real time load height monitoring and control system using plc and hmi for smart industrial solutions. Jurnal Profesi Insinyur Indonesia UMM **4**(2) (2024)

[27] Liu, R., Yuan, J., Huang, X.: Benchmarking time series databases with iotdb-benchmark for iot scenarios. arXiv preprint arXiv:1901.08304 (2019)

[28] Khelifati, A., Khayati, M., Dignös, A., Difallah, D., Cudré-Mauroux, P.: Tsm-bench: benchmarking time series database systems for monitoring applications. Proceedings of the VLDB Endowment **16**(11), 3363–3376 (2023)

[29] Zeng, W., Zhang, S., Yen, I.-L., Bastani, F.B., Hwang, S.-Y.: Monitoring data management services on the edge using enhanced tsdbs. In: 2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA), pp. 9–16 (2019). IEEE

[30] Oyekanlu, E.: Predictive edge computing for time series of industrial iot and large scale critical infrastructure based on open-source software analytic of big data. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 1663–1669 (2017). IEEE

[31] Baimukhanov, S., Ali, H., Yazici, A.: Enhancing ml-based anomaly detection in data management for security through integration of iot, cloud, and edge computing. Expert Systems with Applications **293**, 128700 (2025)

[32] Clerissi, D., Leotta, M., Reggio, G., Ricca, F.: Towards an approach for developing and testing node-red iot systems. In: Proceedings of the 1st ACM SIGSOFT International Workshop on Ensemble-Based Software Engineering, pp. 1–8 (2018)

[33] Simpkin, C., Taylor, I., Harborne, D., Bent, G., Preece, A., Ganti, R.K.: Efficient orchestration of node-red iot workflows using a vector symbolic architecture. Future Generation Computer Systems **111**, 117–131 (2020)

[34] Sosa, R., Kiraly, C., Rodriguez, J.D.P.: Offloading execution from edge to cloud: a dynamic node-red based approach. In: 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 149–152 (2018). IEEE

[35] InfluxData: InfluxDB OSS v2 Documentation. https://docs.influxdata.com/influxdb/v2/. Accessed: 2026-03-06 (n.d.)