IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 29 August 2025, accepted 17 September 2025,
date of publication 22 September 2025, date of current version 30 September 2025.

*Digital Object Identifier 10.1109/ACCESS.2025.3612701*

RESEARCH ARTICLE

# Analyzing Microservice Communication Performance in Industrial Edge Platform

**RICARDO P. PONTAROLLI[1], (Member, IEEE), MASSIMILIANO GAFFURINI[2], (Member, IEEE), EDUARDO P. GODOY[3], (Member, IEEE), AND PAOLO FERRARI[2], (Member, IEEE)**

[1]Electrical Engineering Department, São Paulo State University (Unesp), Bauru 17033-360, Brazil
[2]Information Engineering Department, University of Brescia (UNIBS), 25123 Brescia, Italy
[3]Electrical Engineering Department, São Paulo State University (Unesp), Sorocaba 18185-000, Brazil

Corresponding author: Ricardo P. Pontarolli (pasquati@ifsp.edu.br)

This work was supported in part by São Paulo Research Foundation (FAPESP) under Grant 2020/09850-0 and Grant 2024/21954-7; and in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil, under Finance Code 001.

**ABSTRACT** The integration of legacy manufacturing systems into Industry 4.0 ecosystems necessitates architectures that balance interoperability and performance across infrastructures. This paper addresses these challenges by proposing a hybrid system leveraging Docker containers, the Moleculer framework, and the Asset Administration Shell (AAS) to enable semantic interoperability while quantifying communication latencies in bidirectional industrial workflows, analyzing microservice communication performance in edge platforms through a hybrid architecture that combines legacy protocols with edge virtualization, and contextualizing it against fieldbus-based systems and fully virtualized containerized microservices. Experimental validation on a Siemens S7-1500 Programmable Logic Controller (PLC) evaluates latency distributions across architectures, revealing average round-trip times of 7.23 ms for hybrid sensor updates and 5.94 ms for virtualized command writes, with 75% of operations under 10 ms. Performance tradeoffs emerge between protocol translation overhead in hybrid systems and transporter-induced variability in virtual deployments. By correlating temporal metrics with architectural choices, this work provides actionable insights for modernizing brownfield installations, enabling manufacturers to select optimal strategies based on legacy constraints and real-time requirements. The results highlight the viability of gradual migration paths, preserving existing investments while adopting Industry 4.0 capabilities through performance-aware microservice orchestration.

**INDEX TERMS** Industrial edge computing, communication, legacy system integration, hybrid architectures, virtualized systems, latency analysis, Docker containers, asset administration shell.

## I. INTRODUCTION

The Fourth Industrial Revolution promises to transform manufacturing through the integration of cyber-physical systems, Industrial Internet of Things (IIoT), and asset digitalization. However, global factories still rely on legacy systems with proprietary protocols such as Siemens S7 Communication Protocol (S7comm), creating a technological gap [1].

The virtualization of sensors and assets via edge computing emerges as a strategic solution, enabling the coexistence of legacy infrastructures with modern digital frameworks. Automation Markup Language (AutomationML), Process Automation - Device Information Model (PA-DIM), and AAS offer semantic modeling for interoperability, but their application in legacy environments remains underexplored.

Legacy systems face three critical challenges: (1) incompatibility of fieldbus protocols (S7comm/Process Field Network (PROFINET)) with modern IIoT standards (Open Platform Communications Unified Architecture (OPC UA)/ Message Queuing Telemetry Transport (MQTT)), (2) monolithic architectures that hinder gradual upgrades, and (3)

The associate editor coordinating the review of this manuscript and approving it for publication was Stefano Scanzio.

excessive latency in bidirectional communications for real-time control.

This research focuses on creating a microservices-based automation architecture for Industry 4.0 applied to a Legacy System and measure its communication performance. This proposed architecture makes it possible to run in parallel with the legacy system, enabling a gradual migration of pre-existing industrial plants, being economically viable.

This architecture aims to integrate and support diverse industrial assets related to manufacturing and process control, including services, and is not limited solely to industrial devices.

The architecture will enable the automated deployment of services across distributed assets in a granular manner, as well as modular, open-source configuration and commissioning in the context of industrial manufacturing and process control, thereby enabling new functionalities for a Legacy System, connecting Operational Technologies (OT) with Information Technologies (IT).

This will be achieved through network asset discovery and the use of a standardized information model (AAS), ensuring the identification and recognition of asset functionalities. This will allow assets to be composed in a distributed manner (using microservices) to build applications for manufacturing or industrial process control.

The transition to Industry 4.0 requires significant investments by 2030, primarily aimed at modernizing existing plants [2]. Our solution enables the reuse of legacy infrastructure, reducing migration costs compared to "rip-and-replace" approaches. Experimental validation on a Siemens S7-1500 PLC demonstrates latency compatible with industrial applications.

The proposed architecture enables extensibility for new applications, including Industry 5.0, through its modular microservices-based approach and integration with the AAS. The virtualization of legacy assets via Docker containers and flexible orchestration with the Moleculer framework allows the incorporation of emerging technologies, such as collaborative robots and human-centric Artificial Intelligence (AI), without replacing existing infrastructure [3], [4].

Moreover, the semantic modeling of AAS facilitates interoperability with hyper-personalization and sustainability systems, key pillars of Industry 5.0. This scalability is demonstrated by the ability to deploy new edge-layer services (e.g., predictive analytics modules or intuitive human-machine interfaces) while maintaining industrial grade performance. Thus, the architecture not only addresses legacy integration challenges but also provides a foundation for evolving toward more resilient, collaborative, and human-value-oriented industrial ecosystems [5], [6], [7].

The proposed architecture combines three key components: - Virtualization via Docker containers at the edge layer - Microservice orchestration using Moleculer and Node-RED - Semantic information modeling using AAS (Shown in Fig. 2) The experimental validation employs statistical analysis of bidirectional latency, using hardware sniffers for precise temporal capture (Section III-B).

The main contributions are: 1. Legacy virtualization framework using AAS, reducing integration costs and extension of applications to higher level Enterprise Resource Planning (ERP)/Manufacturing Execution System (MES). 2. Hybrid transport architecture combining Neural Autonomic Transport System (NATS) and S7comm 3. Network Time Protocol (NTP) synchronization methodology with uncertainty <0.5ms (Section III-C)

The remainder of this paper is structured as follows: Section II Literature Overview. Section III details the proposed architecture. Section IV describes the experimental setup. Section V presents results and latency analysis. Conclusions and future work are discussed in Section VI.

## II. LITERATURE OVERVIEW

Three typical architectures found in industry in the Figure 1 presents, (a) traditional system, (b) hybrid system, and (c) full virtual system. In the traditional system shown in (a) in Figure 1, the PLC/Gateway (red square) controls a set of lower-layer devices (sensors and actuators) that are not service-enabled. Each of these devices is exposed through the Gateway as a service-enabled device. This approach allows for the gradual replacement of legacy devices with devices that are natively compatible with web services, without impacting the applications that use these devices [8].

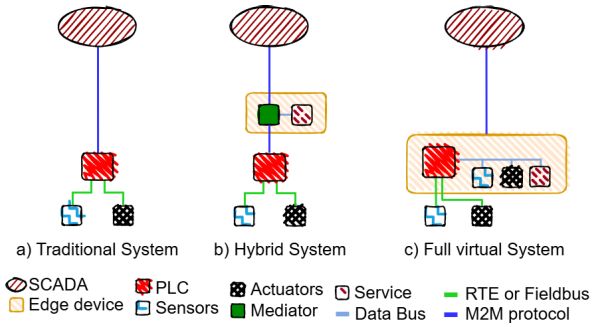


**FIGURE 1. Diagram of implementation, commissioning and distribution of hardware components and their respective communication.**

In the hybrid system shown in (b) in Figure 1, a mediator (green square) can contain multiple data sources, which may or may not be processed before being made available as a service. Reference [8] used the mediator to aggregate several non-service-enabled devices, allowing applications in the upper layers to communicate with the mediator instead of using proprietary interfaces.

In the fully virtual system shown in (c) in Figure 1, unlike the development of traditional remote Inputs/Outputs (I/Os) [9], [10], or a mediator service [8], a distributed I/O is deployed directly connected to the service-oriented network. As a result, all information acquired by the distributed I/O is automatically available for use by any other service or application in a standardized and interoperable manner.

A Microservice-Oriented Architecture (MOA) for automation and control in Industry 4.0 scenarios was developed in [11] and [12] by the authors, focusing on the development of microservices for process control.

Recent studies have leveraged these architectures to explore the virtualization of industrial controllers and the orchestration of microservices in automation environments. The work presented in [12] examines the composition of microservices for process control in Industry 4.0, utilizing the Moleculer framework for orchestration. The authors demonstrate that MOA can be effectively applied in real industrial scenarios, achieving response times suitable for process control. However, communication latency varies depending on the network topology (wireless or wired). This study underscores the flexibility and modularity provided by microservices, while also highlighting the need for precise synchronization to ensure real-time performance.

Additionally, [13] proposes a methodology for evaluating the performance of virtual PLCs (vPLCs) on edge computing platforms, focusing on Machine-to-Machine (M2M) communication for supervision and control. The results show that vPLCs can achieve latencies comparable to those of traditional PLCs, although the implementation of the Internet Protocol (IP) stack in the vPLC occasionally introduces spurious random delays of about 50 ms, when excluding these outliers the measured round-trip time is in the range of 5–6 ms. This study reinforces the feasibility of industrial controller virtualization but emphasizes the importance of optimizing the communication stack to meet stricter real-time requirements. The proposed methodology enables the creation of analytical models that can be used for simulations and worst-case analysis.

Finally, [14] advances the evaluation of real-time performance of vPLCs, focusing on communication with sensors and actuators via PROFINET. The authors demonstrate that vPLCs can operate with cycles as short as 1 ms, although they exhibit 50% higher jitter compared to conventional PLC systems. The study also develops an analytical model that correlates system parameters with performance metrics, with errors below 3%. Collectively, these studies highlight the growing maturity of virtualized architectures in industrial automation, while also identifying challenges such as reducing jitter and optimizing communication stacks for critical real-time applications.

The adoption of virtualization extends beyond production monitoring, functioning as a unified platform for diverse analytical capabilities [15]. Operationally, this technology enhances predictive maintenance frameworks [16] and refines cyberphysical system interoperability [17], while enabling seamless integration with MES through proactive modeling [18], [19]. From a diagnostic perspective, it supports automated anomaly detection [20], dynamic bottleneck forecasting [21], and virtual commissioning of production scenarios [22], [23].

In contrast to OPC UA converters and TSN-based approaches, which primarily focus on enabling deterministic, high-performance communication in modernized industrial networks, our work targets the adaptation of existing legacy infrastructures without requiring complete replacement of hardware or protocols. For example, [8] address the integration of legacy devices into SOA-based architectures, while [12] explore microservice orchestration for process control, and [13] evaluate virtual PLC performance in edge computing scenarios. Unlike these works, the proposed architecture combines microservice orchestration with AAS semantics to integrate conventional PLCs via S7comm, demonstrating that even non-state-of-the-art protocols can be incorporated into Industry 4.0 ecosystems with acceptable latency for supervisory and certain control applications. This novelty lies in bridging the gap between purely modernized communication stacks and the practical constraints of brownfield deployment.

Nevertheless, deploying these solutions encounters technological constraints, particularly in harmonizing with legacy infrastructure. Existing systems often lack the modular architectures and standardized communication protocols required for Industry 4.0 ecosystems, hindering their compatibility with virtualization paradigms.

Following this trend of using microservices in industrial automation and Industry 4.0 applications, this article describes the development of an open-source distributed I/O as a microservice for sensor virtualization. In this project, traditional distributed I/Os were used within a mediator service, corresponding to the hybrid system architecture shown in (b) in Figure 1, which was created to collect data from legacy distributed I/Os and replicate it for MOA services.

## III. PROPOSED ARCHITECTURE

This section presents a description of the architecture proposed in this work, showing the essential concepts involved in its development, as well as the selected technologies or tools.

### A. ARCHITECTURE

The proposed architecture, as illustrated in Figure 2, mainly comprises an Edge device and an Industrial Manufacturing Plant. It is imperative to note that all devices are interconnected via a local network, which is isolated from the internet for security reasons. This ensures reliable system operation and protects against any potential external access.

Within the Edge device, the Mediator, Transporter, and Sensor Service are executed within Docker containers. The Mediator orchestrates the assets through the transporter communication for the sensor service to read or update its value and via an M2M protocol for reading the value of the Sensor stored in the PLC.

The Industrial Manufacturing Plant is controlled by a PLC that is connected to sensors and actuators. In Figure 2, only the target sensor is represented.

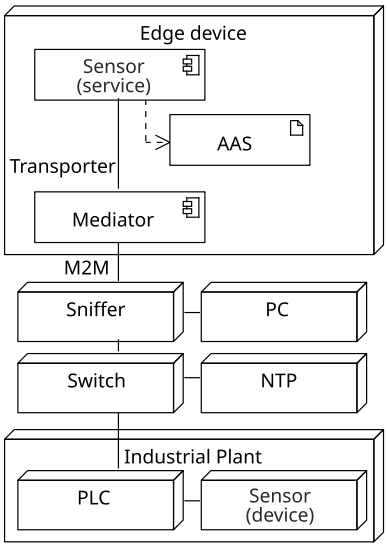


**FIGURE 2.** Diagram of implementation, commissioning and distribution of hardware components and their respective communication.

The communication latency in the general architecture is evaluated by the introduction of a Sniffer and Personal Computer (PC). The function of the Sniffer is to duplicate all data packets within the communication link between the Edge device and the Industrial Manufacturing Plant, and to forward them to the PC to analyze and store them.

### B. METRICS DEFINITION FOR THE TWO EXPERIMENTS (READ AND WRITE)

The latency evaluation is divided into Experiments A and B, as shown in Figure 3 and Figure 4.

Latency calculation was performed through sequential differential analysis ($T_{n;n+1} = T_{n+1} - T_n$). Temporal measurements from the PLC were excluded to reduce uncertainty [13]. This limitation was addressed by measuring the computational time required by the PLC using the Sniffer.

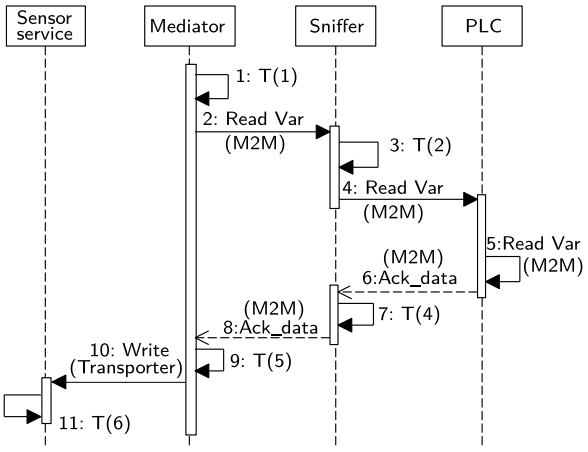


**FIGURE 3.** Experiment A: Update Sensor service value.

As illustrated in Figure 3, the sequence diagram of Experiment A depicts a centralized orchestration of steps initiated by the Mediator at time T(1). The Mediator requests the Sensor's value (device) from the PLC via an M2M protocol using a Read Var message. At time T(2), the Read Var request is captured by the Sniffer. The PLC then receives the Read var message and, upon processing, responds with an Ack_data containing the Sensor's value to the Mediator, thereby confirming the successful Read. This is subsequently captured by the Sniffer at time T(4). At time T(5), the Mediator forwards the Sensor's value to the Sensor service via a Transporter service, which receives the value at time T(6).

- $L_{1;2} = T(2) - T(1)$, Read var(M2M) traverse latency from Mediator to the Sniffer;
- $L_{2;4} = T(4) - T(2)$, latency introduced by the PLC for the elaboration of the Read(M2M) request and the generation of the Ack_Data(M2M).
- $L_{4;5} = T(5) - T(4)$, Ack_Data(M2M) traverse latency from the Sniffer to the Mediator;
- $L_{5;6} = T(6) - T(5)$ Write(Transporter) traverse latency from Mediator to the Sensor Service,
- $L_{1;6} = T(6) - T(1)$, Read operation round-trip time. From the Mediator to the Sensor Service.

In Figure 4, the sequence diagram of Experiment B is shown, demonstrating the reverse process of Experiment A, writing a value from the Sensor service to the PLC. The Mediator at time T(7) sends a Read request to the Sensor service, which at time T(8) responds with a Return message. At time T(9) the Mediator forwards the value, starting the write process, to the PLC using a M2M protocol. The PLC receives the write command and, when processed, sends an Ack_data to the Mediator, confirming the successful write, which is captured by the Sniffer at time T(12).

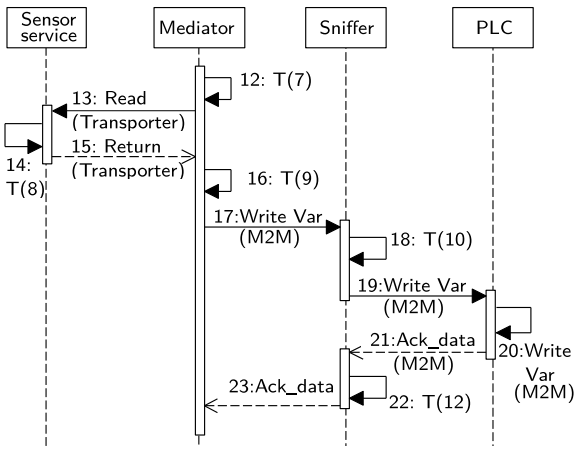


**FIGURE 4.** Experiment B: Update PLC value.

- $L_{7;8} = T(8) - T(7)$, Read(Transporter) traverse latency from Mediator to the Sensor Service;
- $L_{8;9} = T(9) - T(8)$, Return(Transporter) traverse latency from the Sniffer to the Mediator;
- $L_{9;10} = T(10) - T(9)$, Write var(M2M) traverse latency from Mediator to the Sniffer;

- $L_{10;12} = T(12) - T(10)$, latency introduced by the PLC for the elaboration of the Write(S7) request and the generation of the Ack_Data(M2M);
- $L_{7;12} = T(12) - T(7)$, Write operation round-trip time. From the Mediator to the PLC.

Timestamps $T(1)$, $T(5)$, $T(7)$, and $T(9)$ are produced by the Mediator, whereas $T(6)$ and $T(8)$ are generated by the Sensor service and both stored in log files. In contrast, timestamps $T(2)$, $T(4)$, $T(10)$, and $T(12)$ are obtained from the PC responsible for packet collection. The duplicated packets captured by the Sniffer, which is implemented using a Test Access Port (TAP) device, are stored in a pcap file after being captured with tcpdump v4.99.0 with the timestamp precision set to the millisecond level.

### C. SYNCHRONIZATION (NTP)

In the distributed measurement system shown in Figure 2, synchronization of the measurement devices is crucial, as any drift or offset between devices capturing the source and destination timestamps can affect the accuracy of the metrics (e.g., latency). To address this, the NTP box shown in Figure 2 is implemented using a Global Positioning System (GPS) receiver with NTP server capabilities. In our implementation, the Edge device, PC and PLC all natively support the NTP protocol and act as clients. The Sniffer does not support the NTP protocol, but it is a FPGA based hardware that forwards the copied packets to a PC; then the PC assigns a synchronized timestamp. Resuming, each device (which needs to timestamp a packet) is directly retrieving date and time information from the same NTP server, avoiding any intermediate push/pull mechanisms and guaranteeing uniform synchronization across the measurement system. Moreover, all the devices are connected directly via a single Ethernet switch, minimizing the NTP packet delay and jitter. Finally, long and short-term stability of the clock reference in the NTP server is assured by the GPS receiver.

To quantify the synchronization uncertainty of a device (denoted as $m$) Equation 1 is used:

$$u_{sm} = \sqrt{\mu_{sm}^2 + \sigma_{sm}^2} \tag{1}$$

where $\mu_{sm}$ represents the systematic time offset error (no calibration is performed in the experimental setup) and $\sigma_{sm}$ is the standard deviation of the time offset.

Many of the metrics discussed in Section III-B are evaluated between two devices (denoted $m$ and $n$). The standard uncertainty, $u_{mn}$, of these latencies is computed as shown in Equation 2:

$$u_{mn} = \sqrt{u_{sm}^2 + u_{sn}^2} \tag{2}$$

For the case where the same device captures both timestamps, the uncertainty simplifies to Equation 3:

$$u_{mm} = \sqrt{2}u_{sm} \tag{3}$$

## IV. EXPERIMENTAL SETUP

This section presents the experimental setup and the components of the study environment, covering software, hardware, and system integration. The system's technical requirements are detailed, including containerized software platforms, industrial communication protocols, and the physical infrastructure used. Its generic structure has already been presented previously in Figure 2.

Additionally, the microservices architecture adopted for orchestration is described, highlighting the interaction between distributed services and industrial devices. Furthermore, the structure of an automated manufacturing plant is explored, along with its operational cycle and process integrity validation, illustrated through diagrams and functional flows. Finally, the implementation of an edge computing solution for secure application management is discussed, emphasizing interoperability between virtualized environments and physical systems.

This description aims to provide a reproducible basis for understanding the practical implementation and the tests conducted. It is important to highlight that the Industrial manufacturing plant controlled by the PLC is used only to load the CPU with real tasks.

The components illustrated in Figure 2 are implemented as follows: The Sensor Service is realized using the Moleculer framework (Section IV-D) and hosted within Docker containers on the Siemens Industrial Edge platform (Section IV-C). The Mediator orchestrates communications via the Moleculer framework and interfaces with the PLC through the S7comm protocol (Section IV-E). The Transporter (NATS message broker) facilitates microservice communication. The NTP synchronization mechanism (Section III-C) ensures timestamp accuracy across devices, while the Sniffer and PC (Section III-B) capture network traffic for latency analysis. Hardware specifications (e.g., Edge device, PLC) are detailed in Section IV-B, and the Industrial Manufacturing Plant is described in Section IV-A.

### A. INDUSTRIAL MANUFACTURING PLANT

The Industrial manufacturing plant shown in Figure 5, is designed to perform the automated screwing of four threaded caps onto a solid block and to verify their seal integrity in a continuous cycle.

The operating cycle consists of two main phases: the first involves assembling the caps onto the block followed by a seal test, while the second focuses on disassembly and repositioning of the initial components back into the loading tray.

Each cycle processes eight blocks and 32 threaded caps, with four caps per block. Once started, the Selective Compliance Assembly Robot Arm (SCARA) Robot, a FANUC SCARA SR-6iA, picks up a block from the loading tray and moves it to the tag reading unit before placing it onto the screwing unit (position 1). The robot then retrieves a cap, pre-screws it into the first threaded hole of the block,

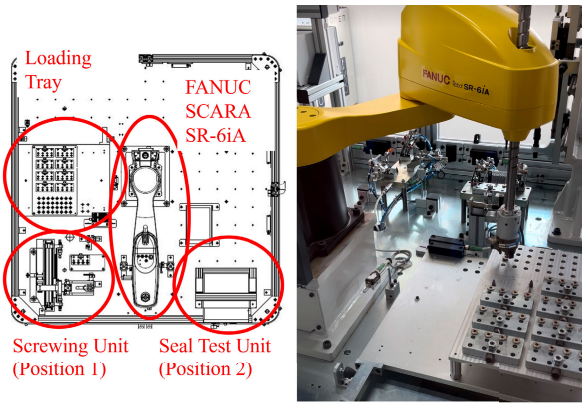


**FIGURE 5.** Industrial manufacturing plant: a) top view b) picture of the system.

and the electric screwdriver completes the tightening process with torque and angle control. Upon completion, the robot transfers the block to the next fixture position, picks up another cap, and repeats the screwing operation. This sequence is repeated until all four caps are secured. The robot then moves the assembled block to the seal test unit (position 2). The cycle is repeated for all blocks on the loading tray.

Once all units have been processed, the disassembly cycle begins. The robot places a completed component onto the screwing unit fixture, where the screwdriver partially unscrews the first cap, leaving approximately one thread engaged. The robot then fully unscrews the cap and places it back in the loading tray. The cycle continues until all assembled components have been disassembled and returned to their initial state.

The entire process is controlled by a PLC, which coordinates the SCARA robot, the electric screwdriver, and the various fixtures. The PLC communicates with the devices over a PROFINET network with a scan cycle of 4 ms.

### B. SOFTWARE AND HARDWARE

The architecture consists of a combination of software and hardware components that work together to ensure satisfactory operation. The software stack includes services running in Docker containers. The hardware configuration consists of industrial-grade devices, including PLCs and networking devices. Tables 1 and 2 summarize the software and hardware specifications, respectively.

**TABLE 1.** Software and corresponding docker images.

| Software | Docker Image | Version |
|---|---|---|
| Mediator | flow creator | 1.17.2 |
| Sensor Service | moleculer:sensor | 1.0.12 |
| Transporter | nats | 1.3.0 |
| S7comm | N/A | 1.1.6 |
| TIA Portal | N/A | 15.0 |

**TABLE 2.** Hardware components and OS versions.

| Name | Model | Firmware/OS version |
|---|---|---|
| Edge device | IPC127E | v2-56e330f6-x86-64 |
| PLC | S7-1500 CPU 1513F | 2.6 |
| NTP | HEOL T103C | N/A |
| Sniffer | KUNBUS-TAP 2100 | 0.90 |
| PC | Siemens IOT2050 | Debian 11 |

### C. SIEMENS INDUSTRIAL EDGE

The edge computing platform based on container services, executed in the Edge device, is Siemens Industrial Edge (SIE) from SIEMENS. SIE solution consists of three main components, as shown in Figure 6:

- Industrial Edge Hub (IEH): a cloud-based repository that stores on-premises middleware and official containerized applications.
- Industrial Edge Management (IEM): a local management system responsible for configuring and supervising edge devices and containerized applications.
- Industrial Edge Devices (IEDs), the physical devices that run the Industrial Edge Runtime, a software layer based on a Debian Linux distribution to execute Docker containerized applications. Containerized applications are called Industrial Edge App. The Industrial Edge Runtime can be executed in Virtual Machine, in this case the device is called Virtual Industrial Edge Device (VIED).

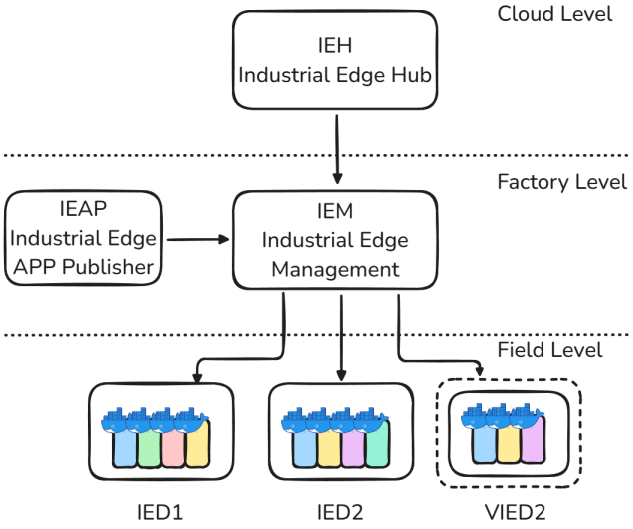


**FIGURE 6.** Siemens industrial edge components.

IEDs connect to both the factory network (for IEM access) and the field network (for data acquisition and control tasks). To ensure secure communication, all traffic is managed through Hyper Text Transfer Protocol Secure (HTTPS) proxies: an Nginx reverse proxy handles inbound traffic redirection, while a redsocks-based proxy manages outbound traffic. Although direct Transmission Control

Protocol (TCP)/IP port exposure is possible, it is generally discouraged. Instead, containerized applications typically utilize internal virtual networks created by Docker.

By default, proxy redirection is preconfigured and can be integrated into the application's Docker Compose file. This mechanism facilitates communication between applications and the Industrial Edge Databus (IE Databus), which is implemented as an MQTT Broker. Before data exchange can occur, the relevant MQTT topics must be pre-configured within the IEM.

A notable feature of the Siemens Industrial Edge ecosystem is the Industrial Edge Flow Creator, a customized implementation of the Node-RED browser-based development tool. This enables users to create applications using JavaScript (JS) functions, allowing for efficient manipulation of data obtained via MQTT connectors linked to the IE Databus.

In order to test and use customs containerized applications a software called Industrial Edge App Publisher (IEAP) provides the possibility to import the customs applications inside the ecosystem.

### D. FRAMEWORK MOLECULER

The Moleculer framework [25] (Figure 7) is an open-source framework for building microservices on Node.js, using loosely coupled services to enhance modularity and streamline development, testing, and deployment. Our tests within Siemens Industrial Edge highlight its adaptability to edge computing environments.

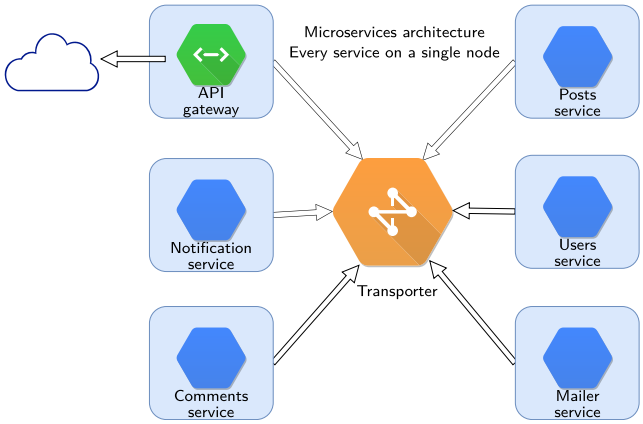


**FIGURE 7.** Microservices architecture.

Moleculer's extensibility enables integration of modules and plugins to enhance functionality, supported by monitoring tools that streamline debugging and maintenance. Key features in version 13 include a masterless architecture with equal nodes, support for transporters like TCP, NATS, MQTT, Kafka, and Advanced Message Queuing Protocol (AMQP), promise-based async/await and request-response patterns, dynamic service discovery, load balancing, and built-in Application Programming Interface (API) gateway and database modules [25].

The microservices architecture in Figure 7 employs individual containerized nodes that communicate via a transporter and message broker, balancing non-negligible inter-service latency with resilience through replication. Moleculer's masterless design treats all services equally, eliminating hierarchy while offering automated service registration, discovery, and fault tolerance.

The architecture in Figure 7 uses blue management nodes for development, discovery, and configuration, with each node hosting one or more services that manage their own data (one database per service). Existing services are automatically notified of new or updated functionalities, while built-in load balancing dynamically distributes requests across the masterless system, ensuring even workload distribution.

The architecture in Figure 7 uses blue hexagons to represent services, which execute specific tasks (e.g., data acquisition actions), while the green hexagon denotes the API gateway. This gateway bridges internal services (communicating via Ethernet) with external applications through REST protocol calls over the internet or cloud networks, enabling seamless integration between local and remote systems.

The orange hexagon in Figure 7 represents the Transporter, a message broker (e.g., MQTT, NATS) that enables communication between microservices by queuing messages until processed. This critical component eliminates the need for direct service discovery, as producers send messages to the Transporter, which holds them until consumers retrieve and process them. Though dependent on external installation, it ensures reliable, decoupled message handling across the architecture.

Moleculer simplifies service discovery for dependency-free protocols like TCP by using the Gossip protocol to share node status, service lists, and heartbeats, alongside built-in User Datagram Protocol (UDP) discovery to automatically detect network nodes. This eliminates manual node listing, requiring only a minimal "gossiper" node (with no services) to distribute remote node addresses, ensuring all nodes need only the gossiper's address to communicate across the network.

Moleculer's message-based communication supports patterns like pub-sub and one-way requests, with the Transporter automatically load-balancing requests across multiple service instances (deployed for availability) to active nodes. Switching between communication mechanisms (e.g., MQTT to NATS) is seamless, enabling developers to adapt strategies flexibly without overhauling the architecture.

Moleculer's microservices architecture uses a messaging protocol to decouple producers and consumers of events/data, enhancing availability and simplifying development compared to distributed transactions. The Transporter acts as a message broker, queuing unprocessed events until a consumer is available, while enabling reliable, decoupled communication between disparate systems. Data serialization is managed by the built-in Serializers module, ensuring seamless message exchange.

### E. S7COMM PROTOCOL

To facilitate seamless communication between PLCs and supervisory systems, specific communication protocols are commonly employed. Siemens developed S7comm as the primary protocol for M2M, Controller-to-Controller (C2C), and Supervisory Control and Data Acquisition (SCADA) communication. This protocol is used by Siemens S7-300, S7-400, S7-1200, and S7-1500 PLC families, as well as external devices [51].

S7comm operates over ISO transport services on top of TCP (TPKT), with all communications taking place on port 102. TPKT is used to handle the block-oriented transfer of data, ensuring that messages are transmitted in discrete, well-defined units suitable for industrial communication. The protocol encapsulates data within Connection-Oriented Transport Protocol (COTP) packets.

Establishing an S7 communication session with a PLC involves three key steps: (i) initiating a COTP connection by sending a request and receiving an acknowledgment (ACK), (ii) setting up the S7 communication session, and (iii) exchanging S7 function codes related to the transaction.

Figure 8 illustrates an example of Read Var and Write Var jobs, which retrieves data from a specific memory area of the Server and transfers it to the corresponding data area in the Client through the Ack_Data.

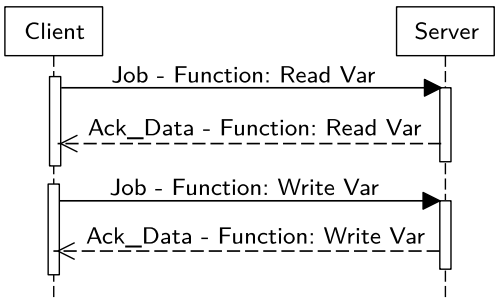


**FIGURE 8.** S7comm Read Var and Write Var jobs.

## V. EXPERIMENTAL RESULTS

Two experiments (A and B) were conducted to evaluate the bidirectional communication capability of the proposed architecture, focusing on the measurement of latency during message sequencing. Experiment A (Update Sensor Service Value) focused on reading data from the physical sensor connected to the PLC (S7) and subsequently updating the sensor service by writing to the information model. Experiment B (Update PLC value) reversed this flow, writing values from the sensor service to the PLC (S7) as explained previously.

### A. SYNCHRONIZATION UNCERTAINTY

Time synchronization is a critical aspect of distributed systems, particularly in real-time and industrial applications. Following Section III-C synchronization accuracy across different devices and associated uncertainties are evaluated.

In order to assess the accuracy of the synchronization, the average and standard deviation of time offsets (shown in Figure 9) between the NTP server and the synchronized devices are measured. The distribution of the offsets captured during the experiments is reported to highlight the variability of synchronization over time.

The experiments are conducted over a duration of 4 hours, corresponding to the complete time windows of the tests. A total of 2000 samples are collected for the NTP server self-timestamping using hardware-assisted measurements, ensuring precise reference timing. For the synchronized devices, 34 samples are collected for the PC and 34 samples for the Edge device from the NTP system logs, allowing a comparison of their offsets relative to the NTP server.

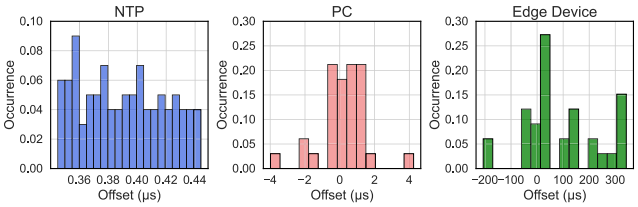


**FIGURE 9.** Offset histogram.

The results are summarized in Table 3:

**TABLE 3.** NTP synchronization statistics.

| Device | Avg Offset (µs) | Std Offset (µs) |
|---|---|---|
| NTP | 0.4 | 0.2 |
| PC | 0.3 | 1.2 |
| Edge device | 88.7 | 157.6 |

The NTP server and PC exhibit minimal time offset and standard deviation, indicating high synchronization accuracy. In contrast, the Edge device shows significantly higher average offset and standard deviation.

Quantification of uncertainty is crucial for determining the reliability of time-synchronization measurements during the two experiments. Starting from Table 3 is possible to analyze Standard ($u_c$) and Expanded ($U$), calculated with a coverage factor $k = 2$, uncertainties in the different link configurations. The results are presented in Tables 4-5. Depending on the links, the Expanded uncertainties range from 0.003 ms up to 0.5 ms.

**TABLE 4.** Standard and expanded uncertainties for experiment A.

| | L1;2 | L2;4 | L4;5 | L5;6 | L1;6 |
|---|---|---|---|---|---|
| $u_c$ (ms) | 0.18 | 0.0017 | 0.18 | 0.26 | 0.26 |
| $U$ (ms) | 0.4 | 0.003 | 0.4 | 0.5 | 0.5 |

### B. EXPERIMENTS AND DISCUSSIONS

Figure 10 graphically represents the histogram of latencies collected in Experiment A and Experiment B, providing an overview of the time distribution in each sequence of

**TABLE 5.** Standard and expanded uncertainties for experiment B.

| | L7;8 | L8;9 | L9;10 | L10;12 | L7;12 |
|---|---|---|---|---|---|
| $u_c$ (ms) | 0.26 | 0.26 | 0.18 | 0.0017 | 0.18 |
| $U$ (ms) | 0.5 | 0.5 | 0.4 | 0.003 | 0.4 |

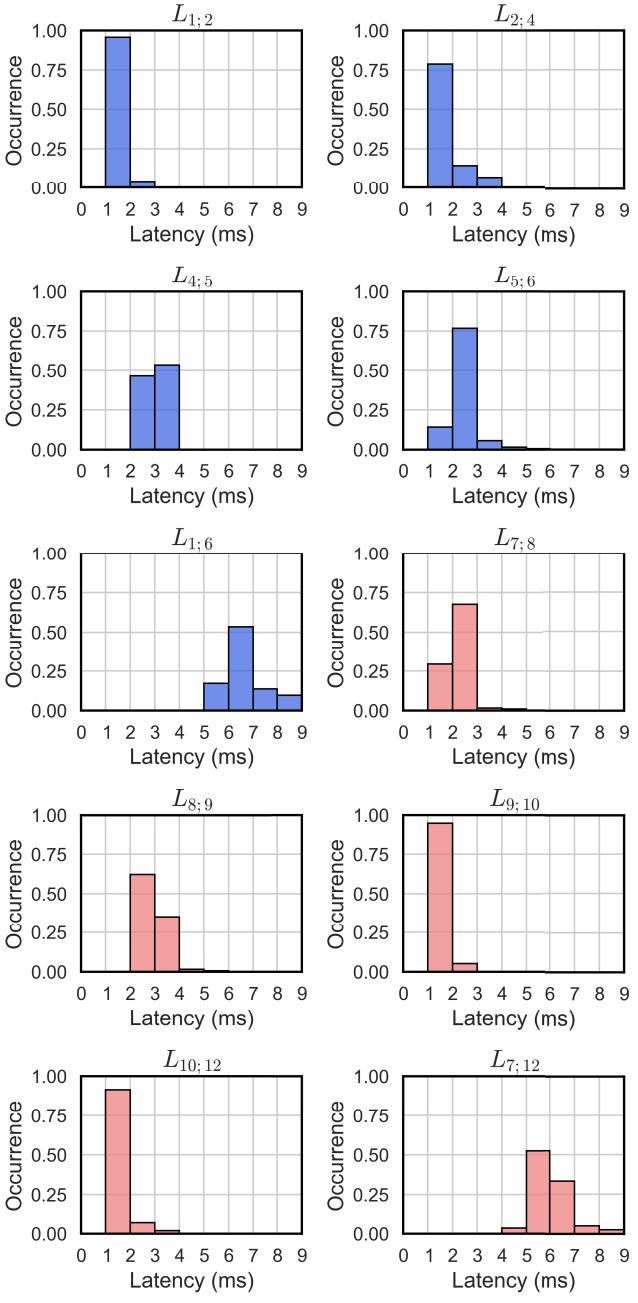


**FIGURE 10.** Latency Distribution of values under 95th percentile: in blue Experiment A while in red Experiment B.

the experiments, which will be discussed next with statistical data compiled into tables. For both experiments, 200 samples (N) were collected each. In order to facilitate enhanced visualization of the behaviour, solely valid values (not under resolution) are represented in the distribution, with the values situated below the 95th percentile.

The results of Experiment A (Update sensor service value) are shown in the Table 6 presents the transmission latencies (in milliseconds) between devices during the update of the sensor service value with the real sensor value. Data include average (Avg), standard deviation (Std), minimum (Min), percentiles (%), and maximum (Max).

**TABLE 6.** Transmission latencies (ms) between devices during Experiment A. ("-" means under resolution).

| Lat | Avg | Std | Min | 25% | 50% | 75% | 95% | Max |
|---|---|---|---|---|---|---|---|---|
| $L_{1;2}$ | 1 | 0.6 | - | - | 1 | 1 | 1 | 3 |
| $L_{2;4}$ | 1.2 | 0.77 | - | 1 | 1 | 1 | 3 | 4 |
| $L_{4;5}$ | 3 | 2.7 | 2 | 1 | 3 | 3 | 4 | 5 |
| $L_{5;6}$ | 3 | 5.6 | 1 | 2 | 2 | 2 | 3 | 54 |
| $L_{1;6}$ | 7 | 6.1 | 5 | 6 | 6 | 7 | 9 | 57 |

The results indicate that the average latency varies among the pairs of devices analyzed. The lowest average latency was observed for $L_{1;2}$ (1 ms), while the highest was observed for $L_{1;6}$ (7 ms), which was expected as it represents the total time required for process completion. The standard deviation follows this variation, being higher for connections with elevated average latencies, such as $L_{1;6}$ (57 ms), followed by $L_{5;6}$ (54 ms), indicating a high dispersion in transmission times.

The percentile analysis reveals that connections such as $L_{1;2}$, $L_{2;4}$ and $L_{4;5}$ exhibit low and stable latency distributions, while $L_{5;6}$ and $L_{1;6}$ show a greater dispersion, suggesting variations in communication performance. Some latency spikes appeared, with the maximum recorded value (57 ms) occurring in the $L_{1;6}$ connection, followed by $L_{5;6}$ (54 ms), which may indicate moments of congestion or transmission interference. Excluding these spikes, communications remained within a range of up to 10 ms.

The analysis of percentile data, such as the 95th percentile of 9 ms reported in Table 6, indicates that tail latency may approach or exceed the real-time thresholds required for certain industrial control applications. While the average and median latencies remain within acceptable limits for supervisory and non-critical control tasks, the upper-tail delays could compromise performance in closed-loop or time-sensitive processes. In such cases, strategies such as tighter timeout configurations, traffic prioritization, or the adoption of time-sensitive networking (TSN) profiles may be necessary to ensure deterministic behavior and maintain system stability under peak load conditions.

The results of Experiment B (Update PLC value) are shown in the Table 7 presents the transmission latencies (in milliseconds) between devices during the updating PLC value that comes from sensor service. Data include average (Avg), standard deviation (Std), minimum (Min), percentiles (%), and maximum (Max).

The results indicate that the average latency varies between 0.9 ms ($L_{10;12}$) and 6 ms ($L_{7;12}$). The standard deviation also follows this variation, being higher for connections with

**TABLE 7. Transmission latencies (ms) between devices during Experiment B. ("-" means under resolution).**

| Lat | Avg | Std | Min | 25% | 50% | 75% | 95% | Max |
|---|---|---|---|---|---|---|---|---|
| $L_{7;8}$ | 2 | 3.1 | 1 | 1 | 2 | 2 | 2 | 45 |
| $L_{8;9}$ | 3 | 2.3 | 2 | 2 | 2 | 3 | 3 | 34 |
| $L_{9;10}$ | 1 | 0.6 | - | - | - | 1 | 1 | 3 |
| $L_{10;12}$ | 0.90 | 0.59 | - | 1 | 1 | 1 | 2 | 4 |
| $L_{7;12}$ | 6 | 3.9 | 4 | 5 | 5 | 6 | 8 | 50 |

high latency, such as $L_{7;12}$ (3.9 ms), $L_{7;8}$ (3.1 ms), and $L_{8;9}$ (2.3 ms), showing greater dispersion in transmission times.

The percentile analysis shows that the connections $L_{9;10}$ and $L_{10;12}$ have low and well-distributed central values, indicating more stable transmissions. In contrast, $L_{7;12}$, $L_{7;8}$, and $L_{8;9}$ show higher latency values compared to other pairs, with maximum values of 50 ms, 45 ms, and 34 ms, respectively, suggesting possible variations in the network load. Excluding these spikes, communications remained within a range of up to 10 ms.

It is important to note that the experiments were designed to access a single input or output of the PLC, representing a worst-case scenario in terms of protocol efficiency. In S7comm, each transaction carries a fixed header and control fields whose size is independent of the payload, resulting in a proportionally high communication overhead for small data transfers. For example, reading a single variable of 8 bytes requires 105 bytes in the request and 107 bytes in the response, totaling 212 bytes exchanged; writing the same 8 bytes demands 117 bytes in the request and 96 bytes in the response, totaling 213 bytes. Although the parsing and message assembly costs were not quantitatively measured in this work, the results suggest that such factors, together with potential TCP-level optimizations (e.g., Nagle's algorithm), may contribute to the observed latency fluctuations. In practical deployments, burst-oriented transfers grouping multiple I/O points into a single transaction could significantly improve bandwidth utilization and reduce the relative impact of protocol overhead, which is a relevant direction for future work.

Based on the experimental results presented, it is clear that the proposed architecture, while functional, shows significant variations in communication latency between devices. A detailed analysis of the transmission latencies, both in the virtual sensor update and in writing the value to the PLC, highlights the importance of considering not only the average but also the variability of response times. The presence of latency spikes, evidenced by high maximum values in some connections, suggests the need for optimizations in the system to ensure greater predictability and reliability in operations.

A possible cause for these latency variations may be related to the configuration of the Moleculer framework, specifically the *requestTimeout* parameter. As observed in version 0.13.13, the default value for this parameter is set to 0 ms, meaning requests have no timeout. This configuration can lead to indefinite wait times, depending on the implementation of services, contributing to the observed latency spikes.

The Moleculer framework's *requestTimeout=0* parameter was maintained to observe industrial network latency values without filtering, such as the peaks of up to 54 ms recorded in Table 6. When set to values greater than zero, this parameter acts as a cutoff mechanism for late responses, forcing the termination of requests that exceed the configured maximum time. As a result, the influence of anomalous events is eliminated, ensuring a more uniform distribution of response times, improving communication predictability in industrial scenarios, and reducing unwanted variability in read and write operations.

It is therefore recommended to investigate and adjust this parameter, setting an appropriate timeout value for the specific industrial environment, to mitigate the risks of excessive delays in communication between devices. Additionally, implementing real-time monitoring and diagnostic mechanisms can help identify and resolve latency issues, ensuring the efficiency and stability of the system in production scenarios.

These results demonstrate the importance of considering not only the average latency but also its variability when designing distributed systems that rely on rapid updates between virtual and real sensors.

## VI. CONCLUSION

The integration of legacy manufacturing systems into Industry 4.0 ecosystems remains a critical challenge, requiring architectures that balance interoperability, scalability, and backward compatibility. This work demonstrates that microservices-based architectures, when combined with the AAS, effectively bridge the gap between conventional PLCs and modern frameworks. By encapsulating legacy functions as containerized services and exposing them through standardized interfaces, the proposed approach preserves existing infrastructure investments while enabling new Industry 4.0 capabilities.

Central to this architecture is the strategic use of Docker containers and the Moleculer framework, which provide modularity and technology-agnostic communication. Docker ensures consistent execution environments for legacy components, while Moleculer's transporter layer enables seamless integration between S7comm fieldbus protocols and modern message brokers like NATS. This hybrid design allows gradual modernization, where legacy systems coexist with microservices, reducing migration risks and costs.

Experimental validation on a Siemens S7-1500 PLC demonstrated average latencies of 7.23 ms for sensor updates and 5.94 ms for command writes, with 75% of operations under 10 ms—performance suitable for industrial supervision and certain non-critical real-time control applications where timing constraints permit this latency profile. The hybrid approach, integrating the S7comm protocol with modern transporters, allows the coexistence of legacy systems

with microservices, reducing migration costs compared to full replacement strategies.

Three main contributions emerge from this research: (1) an AAS-mediated legacy asset virtualization framework that reduces integration costs compared to monolithic solutions; (2) a hybrid transport architecture combining S7comm and NATS, maintaining latencies below 10 ms in 75% of cases; and (3) an NTP synchronization methodology with uncertainty below 0.5 ms, which is critical for accurate temporal measurements in distributed environments.

The implementation of AAS ensures semantic standardization, transforming legacy components into Industry 4.0-compatible assets, while orchestration via Node-RED and Moleculer enables the dynamic composition of services, expanding application capabilities into higher layers and integrating with information technology.

Future work will focus on extending support for protocols such as PROFINET and Modbus TCP, implementing Time-Sensitive Networking (TSN) profiles to reduce jitter in transporter-mediated communications, and integrating trusted execution environments for critical services.

The results demonstrate that gradual modernization via microservices offers an economically viable path for transitioning to Industry 4.0, preserving existing investments and ensuring scalability without premature obsolescence.

## REFERENCES

[1] Y. Chen, H. Ding, J. Gao, X. Tang, Y. Liu, and M. Yang, "A novel strategy for enhancing the thermal conductivity of shape-stable phase change materials via carbon-based in situ reduction of metal ions," *J. Cleaner Prod.*, vol. 243, Jan. 2020, Art. no. 118627, doi: 10.1016/j.jclepro.2019.118627.

[2] M. Ghobakhloo, M. Iranmanesh, B. Foroughi, E. B. Tirkolaee, S. Asadi, and A. Amran, "Industry 5.0 implications for inclusive sustainable manufacturing: An evidence-knowledge-based strategic roadmap," *J. Cleaner Prod.*, vol. 417, Sep. 2023, Art. no. 138023.

[3] K. Souaïbou Hawaou, V. C. Kamla, S. Yassa, O. Romain, J. E. Ndamlabin Mboula, and L. Bitjoka, "Industry 4.0 and industrial workflow scheduling: A survey," *J. Ind. Inf. Integr.*, vol. 38, Mar. 2024, Art. no. 100546.

[4] M. Sharma, A. Tomar, and A. Hazra, "Edge computing for industry 5.0: Fundamental, applications, and research challenges," *IEEE Internet Things J.*, vol. 11, no. 11, pp. 19070–19093, Jun. 2024.

[5] I. L. Dourado, L. C. Moreira, M. Kauffman, and B. Horan, "A wearable AR system for maintenance in industry 5.0: Assessing mental workload and usability," in *Proc. 29th Int. Conf. Autom. Comput. (ICAC)*, Aug. 2024, pp. 1–6.

[6] M. D. Choudhry, S. Jeevanandham, M. Sundarrajan, A. Jothi, K. Prashanthini, and V. Saravanan, "Future technologies for industry 5.0 and society 5.0," in *Automated Secure Computing for Next-Generation Systems,*. Hoboken, NJ, USA: Wiley, 2023, pp. 403–414.

[7] Š. Kozák, E. Ružický, J. Lacko, and A. Kozáková, "From industry 4.0 to industry 5.0: Current trends in research and education for industry 5.0," in *Proc. Cybern. Informat.*, 2025, pp. 1–7.

[8] S. Karnouskos, T. Bangemann, and C. Diedrich, "Integration of legacy devices in the future SOA-based factory," *IFAC Proc. Volumes*, vol. 42, no. 4, pp. 2113–2118, 2009, doi: 10.3182/20090603-3-ru-2001.0487.

[9] M. Haizad, R. Ibrahim, A. Adnan, T. D. Chung, and S. M. Hassan, "Development of low-cost real-time data acquisition system for process automation and control," in *Proc. 2nd IEEE Int. Symp. Robot. Manuf. Autom. (ROMA)*, Sep. 2016, pp. 1–5, doi: 10.1109/ROMA.2016.7847826.

[10] M. Abdallah and O. Elkeelany, "A survey on data acquisition systems DAQ," in *Proc. Int. Conf. Comput., Eng. Inf.*, Apr. 2009, pp. 240–243.

[11] J. A. Bigheti, M. M. Fernandes, and E. P. Godoy, "Control as a service: A microservice approach to industry 4.0," in *Proc. II Workshop Metrology Ind. 4.0 IoT*, Jun. 2019, pp. 438–443, doi: 10.1109/METROI4.2019.8792918.

[12] R. P. Pontarolli, J. A. Bigheti, M. M. Fernandes, F. O. Domingues, S. L. Risso, and E. P. Godoy, "Microservice orchestration for process control in industry 4.0," in *Proc. IEEE Int. Workshop Metrol. Ind. 4.0 IoT*, Jun. 2020, pp. 245–249, doi: 10.1109/METROIND4.0IOT48571.2020.9138228.

[13] M. Gaffurini, P. Bellagente, A. Depari, A. Flammini, E. Sisinni, and P. Ferrari, "Virtual PLC in industrial edge platform: Performance evaluation of supervision and control communication," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–10, 2024, doi: 10.1109/TIM.2024.3370746.

[14] M. Gaffurini, D. Brandão, S. Rinaldi, A. Flammini, E. Sisinni, and P. Ferrari, "Characterizing the real-time communication performance of virtual PLC in industrial edge platform," *IEEE Open J. Instrum. Meas.*, vol. 4, 2025, Art. no. 5500311.

[15] M. Singh, E. Fuenmayor, E. Hinchy, Y. Qiao, N. Murray, and D. Devine, "Digital twin: Origin to future," *Appl. Syst. Innov.*, vol. 4, no. 2, p. 36, May 2021, doi: 10.3390/asi4020036.

[16] V. M. Caldana, F. D. G. D. Silva, R. A. D. Oliveira, and J. F. Borin, "Internet of Things and artificial intelligence applied to predictive maintenance in industry 4.0: A systematic literature review," *Sustainability Anal.*, vol. 8, no. 3, pp. 1–12, 2019, doi: 10.46254/sa02.20210582.

[17] J. Liu, J. Vatn, and S. Yin, "A case study for enhancing maintenance of cyber-physical systems with digital twin," *IEEE Trans. Ind. Cyber-Phys. Syst.*, vol. 2, pp. 597–605, 2024, doi: 10.1109/TICPS.2024.3485038.

[18] M. Stolze, A. Belyaev, C. Kosel, C. Diedrich, and A. Barnard, "Realizing automated production planning via proactive AAS and business process modelsels," in *Proc. IEEE 29th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2024, pp. 1–8, doi: 10.1109/ETFA61755.2024.10710731.

[19] A. Ullah and M. Younas, "Development and application of digital twin control in flexible manufacturing systems," *J. Manuf. Mater. Process.*, vol. 8, no. 5, p. 214, Sep. 2024, doi: 10.3390/jmmp8050214.

[20] E. Negri, S. Berardi, L. Fumagalli, and M. Macchi, "MES-integrated digital twin frameworks," *J. Manuf. Syst.*, vol. 56, pp. 58–71, Jul. 2020, doi: 10.1016/j.jmsy.2020.05.007.

[21] E. Mahmoodi, M. Fathi, and M. Ghobakhloo, "The impact of industry 4.0 on bottleneck analysis in production and manufacturing: Current trends and future perspectives," *Comput. Ind. Eng.*, vol. 174, Dec. 2022, Art. no. 108801, doi: 10.1016/j.cie.2022.108801.

[22] L. Ragazzini, E. Negri, L. Fumagalli, and M. Macchi, "Digital twin-based bottleneck prediction for improved production control," *Comput. Ind. Eng.*, vol. 192, Jun. 2024, Art. no. 110231, doi: 10.1016/j.cie.2024.110231.

[23] A. Florescu, "Digital twin for flexible manufacturing systems and optimization through simulation: A case study," *Machines*, vol. 12, no. 11, p. 785, Nov. 2024, doi: 10.3390/machines12110785.

[24] G. Bitsch, P. Senjic, and J. Askin, "Integration of legacy systems to cyber-physical production systems using semantic adapters," *Proc. CIRP*, vol. 118, pp. 259–263, Jan. 2023, doi: 10.1016/j.procir.2023.06.045.

[25] Moleculer. *Moleculer Microservices Framework for Node.js*. Accessed: 2025. [Online]. Available: https://moleculer.services

[26] Node-RED. *Node-RED—Low-Code Programming for Event-Driven Applications*. Accessed: 2025. [Online]. Available: https://nodered.org/

[27] ECLASS. *ECLASS Basic & Advanced Content Suche—ECLASS*. Accessed: 2025. [Online]. Available: https://eclass.eu/eclass-standard/content-suche

[28] *Plattform Industrie 4.0—Details of the Asset Administration Shell—Part 1*. Accessed: 2025. [Online]. Available: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html

[29] *Plattform Industrie 4.0—Submodel Templates of the Asset Administration Shell—Generic Frame for Technical Data for Industrial Equipment in Manufacturing*. Accessed: 2025. [Online]. Available: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Submodel_templates-Asset_Administration_Shell-Technical_Data.html

[30] Eclipse Foundation. *Eclipse AASX Package Explorer | Projects.eclipse.org*. Accessed: 2025. [Online]. Available: https://projects.eclipse.org/projects/dt.aaspe

[31] Eclipse Foundation. *Eclipse Digital Twin | Projects.eclipse.org*. Accessed: 2025. [Online]. Available: https://projects.eclipse.org/projects/dt

[32] IDTA. *Admin-Shell-IO/AASX-Package-Explorer: C# Based Viewer/Editor for the Asset Administration Shell*. Accessed: 2025. [Online]. Available: https://github.com/admin-shell-io/aasx-package-explorer
[33] IDTA. *IDTA—Working Together to Promote the Digital Twin*. Accessed: 2025. [Online]. Available: https://industrialdigitaltwin.org/en/
[34] A. Shojaeinasab, T. Charter, M. Jalayer, M. Khadivi, O. Ogunfowora, N. Raiyani, M. Yaghoubi, and H. Najjaran, "Intelligent manufacturing execution systems: A systematic review," *J. Manuf. Syst.*, vol. 62, pp. 503–522, Jan. 2022, doi: 10.1016/j.jmsy.2022.01.004.
[35] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018.
[36] DOCKER. (2025). *Docker Overview | Docker Docs*. [Online]. Available: https://docs.docker.com/get-started/overview/
[37] A. Boltaboyeva, Z. Baigarayeva, O. Postolache, M. Mansurova, N. Karymssakova, and B. Belgibayev, "Smart sensors and IoT applied in digital twin for industry 4.0," in *Proc. Int. Symp. Sens. Instrum. 5G IoT Era (ISSI)*, Aug. 2024, pp. 1–6, doi: 10.1109/issi63632.2024.10720505.
[38] M. Etxegarai, M. Camps, L. Echeverria, M. Ribalta, F. Bonada, and X. Domingo, "Virtual sensors for smart data generation and processing in AI-driven industrial applications," in *Industry 4.0-Perspectives and Applications*. London, U.K.: IntechOpen, 2022.
[39] I. S. Acharyya, A. Al-Anbuky, and S. Sivaramakrishnan, "Software-defined sensor networks: Towards flexible architecture supported by virtualization," in *Proc. Global IoT Summit (GIoTS)*, Jun. 2019, pp. 1–4, doi: 10.1109/GIOTS.2019.8766429.
[40] C. Gao, Z. Tian, Y. Chen, and Z. Wang, "A novel virtual sensor management scheme for manufacturing network," in *Proc. Int. Conf. Netw. Netw. Appl. (NaNA)*, Oct. 2018, pp. 29–35, doi: 10.1109/NANA.2018.8648727.
[41] I. S. Acharyya and A. Al-Anbuky, "Software-defined wireless sensor network: WSN virtualization and network re-orchestration," *Adv. Intell. Syst. Comput.*, vol. 1136, pp. 79–90, May 2020, doi: 10.5220/0009194600790090.
[42] H. Darvishi, D. Ciuonzo, E. R. Eide, and P. S. Rossi, "Sensor-fault detection, isolation and accommodation for digital twins via modular data-driven architecture," *IEEE Sensors J.*, vol. 21, no. 4, pp. 4827–4838, Feb. 2021, doi: 10.1109/JSEN.2020.3029459.
[43] M. Majid, S. Habib, A. R. Javed, M. Rizwan, G. Srivastava, T. R. Gadekallu, and J. C.-W. Lin, "Applications of wireless sensor networks and Internet of Things frameworks in the industry revolution 4.0: A systematic literature review," *Sensors*, vol. 22, no. 6, p. 2087, Mar. 2022, doi: 10.3390/s22062087.
[44] E. Garcia, N. Montés, J. Llopis, and A. Lacasa, "Miniterm, a novel virtual sensor for predictive maintenance for the industry 4.0 era," *Sensors*, vol. 22, no. 16, p. 6222, Aug. 2022, doi: 10.3390/s22166222.
[45] V. Alcácer and V. Cruz-Machado, "Scanning the industry 4.0: A literature review on technologies for manufacturing systems," *Eng. Sci. Technol., Int. J.*, vol. 22, no. 3, pp. 899–919, Jun. 2019, doi: 10.1016/j.jestch.2019.01.006.
[46] A. Martínez-Gutiérrez, J. Díez-González, R. Ferrero-Guillén, P. Verde, R. Álvarez, and H. Perez, "Digital twin for automatic transportation in industry 4.0," *Sensors*, vol. 21, no. 10, p. 3344, May 2021, doi: 10.3390/s21103344.
[47] Q. Qi, F. Tao, and Y. Zuo, "Digital twin service towards smart manufacturing," *Proc. CIRP*, vol. 104, pp. 1098–1103, Jan. 2018, doi: 10.1016/j.procir.2021.11.165.
[48] D. Romero, J. Stahre, and M. Taisch, "The operator 4.0: Towards socially sustainable factories of the future," *Comput. Ind. Eng.*, vol. 139, Jan. 2020, Art. no. 106128.
[49] X. Wang, Y. Han, and V. Leung, "Energy-efficient task offloading in UAV-assisted mobile edge computing," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 356–368, Jan. 2021.
[50] ISA. (2021). *Beyond the Pyramid: Using ISA95 for Industry 4.0 and Smart Manufacturing*. [Online]. Available: https://www.isa.org/intech-home/2021/october-2021/features/beyond-the-pyramid-using-isa95-for-industry-4-0-an
[51] (2025). *S7 Communication (S7comm)*. [Online]. Available: https://wiki.wireshark.org/S7comm

**RICARDO P. PONTAROLLI** (Member, IEEE) received the B.Eng. degree in computer engineering from the Virtual University of the State of São Paulo, Brazil, in 2024. He is currently pursuing the Ph.D. degree in electrical engineering with Unesp. He is a Laboratory Technician at IFSP, Boituva, Brazil.

**MASSIMILIANO GAFFURINI** (Member, IEEE) received the M.Sc. degree in electronics engineering from the University of Brescia, Brescia, Italy, in 2023, where he is currently pursuing the Ph.D. degree in technology for health. He is working in the fields of the Industrial Internet of Things, Industry 4.0, and sustainable mobility.

**EDUARDO P. GODOY** (Member, IEEE) received the B.Eng. degree in control and automation engineering from Itajubá Federal University, Brazil, in 2003, and the Ph.D. degree from the University of São Paulo at São Carlos, Brazil, in 2011. Currently, he is an Associate Professor with São Paulo State University (Unesp), Brazil.

**PAOLO FERRARI** (Member, IEEE) received the M.Sc. degree in electronics engineering and the Ph.D. degree in electronic instrumentation from the University of Brescia, Brescia, Italy, in 1999 and 2003, respectively. He is currently a Full Professor with the Department of Information Engineering, University of Brescia. His research interests include measurement systems for industrial and smart city applications, including performance and security analysis of real-time networks and the Internet of Things (IoT) applications, wired and wireless sensor networks, and the clock synchronization of distributed systems.

Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - ROR identifier: 00x0ma614