Virtual PLC in Industrial Edge Platform: Performance Evaluation of Supervision and Control Communication

Massimiliano Gaffurini, *Member, IEEE*, Paolo Bellagente, *Member, IEEE*, Alessandro Depari, *Member, IEEE*, Alessandra Flammini, *Fellow, IEEE*, Emiliano Sisinni, *Member, IEEE*, and Paolo Ferrari, *Member, IEEE*

***Abstract*— Edge computing allows for data processing at reduced latency since the computational power is moved close to the data sources. Traditionally, edge computing has been often used in industrial scenarios for implementing gateways between the operational technology (OT) world and the IT (cloud) world. Recently, big manufacturers of industrial programmable logic controller (PLC) started promoting the use of containerized virtual PLC (vPLC) hosted inside edge computing platforms. They foresee an innovative integration of container based applications, including automation control, with all the data-centric services and applications already available for edge ecosystems. Even if a clear advantage from the scalability and maintainability could be expected, would vPLCs meet the stringent requirements of industrial automation? This article is part of a multistage research work, and as a first step, it is focused on the evaluation of the performance of vPLC when exchanging data with other machines, controllers, supervisors, and data acquisition systems in a machine-to-machine (M2M) scenario. After a brief overview of the involved technology, the design of a methodology for comparing real PLC and vPLC is described. Then, performance metrics, and an experimental setup for the evaluation of existing devices are defined taking care of the sources of uncertainty. The effectiveness of the proposed methodology is demonstrated by considering a real use case. Through the use of the suggested methodology, important insights into the use case are revealed: for instance, the considered vPLC could work as fast as a real PLC with minimum communication latency in the order of 3 ms but, currently, there is a random delay with an average of 50 ms whose source has been identified to be the IP stack implementation of the vPLC. Finally, the proposed methodology allows for the creation and validation of analytical models of the use case.**

***Index Terms*— Container based virtualization, controller to controller (C2C), machine-to-machine (M2M), programmable logic controller (PLC), supervisory control, and data acquisition (SCADA).**

I. Introduction

THE edge computing paradigm is rapidly evolving and it has been adopted in many scenarios since edge computing can reduce latency compared to cloud computing [1].

Manuscript received 22 July 2023; revised 3 January 2024; accepted 9 January 2024. Date of publication 27 February 2024; date of current version 6 March 2024. This work was supported by the Next-GenerationEU (Italian Piano nazionale ripresa resilienza (PNRR)—M4 C2, Invest 1.3—D.D. 1551.11-10-2022) under Grant PE00000004. The Associate Editor coordinating the review process was Dr. Guanfeng Du. *(Corresponding author: Massimiliano Gaffurini.)*

The authors are with the Department of Information Engineering, University of Brescia, 25123 Brescia, Italy (e-mail: massimiliano.gaffurini@unibs.it; paolo.bellagente@unibs.it; alessandro.depari@unibs.it; alessandra.flammini@unibs.it; emiliano.sisinni@unibs.it; paolo.ferrari@unibs.it).

Digital Object Identifier 10.1109/TIM.2024.3370746

The industrial automation is now following this trend. The first version of a virtualized programmable logic controller (PLC) is appearing [2]. Traditional PLCs have custom firmware running on proprietary hardware, with the aim of ensuring real-time availability. More recently SoftPLCs have control software running on standard PC hardware and real-time operating system (RTOS), with the aim to reduce cost, assure portability, and provide multiple vendor sources. The newest approach proposes virtual PLCs (vPLCs) that are the containerized version of PLC firmware: they can be executed on any platform that supports containers, assuring easily maintainable, lightly virtualized, solutions with full independency from both the hardware and the operating system. Moreover, the container based automation approach allows for microservice architectures, enabling new features such as scalability, observability, traceability, and accountability. In other words, the industrial machine can (independently of hardware) run exactly the required/licensed/verified services needed to produce the desired product together with its up-to-date/certified metadata (necessary for accounting for the service). Maintenance and update of applications is centrally managed assuring the security and integrity of the whole system [3], [4], [5].

All the previously listed advantages are clear to machine builders that currently use traditional PLC, but an underlined question remains: what is the performance of vPLCs compared to real PLCs? As a matter of fact, the automation experts from the operation technology (OT) field are obsessed with real-time constraints and they perfectly know that a new fancy controller that fails to control deadlines would result in a useless solution (i.e., usually industrial applications cannot tolerate jitter and high latency [6]).

A. Objectives

Considering the described situation, the goal of the project is to provide: a methodology, an experimental procedure, and a set of metrics to evaluate the performance of the communication and data exchange of PLC and vPLC.

Since the PLCs are placed at the center of the automation stack (also known as the computer-integrated manufacturing (CIM) automation pyramid [7]), they have two types of data exchange: 1) they are connected to other machines, supervisory control, and data acquisition (SCADA) systems [8],

and controllers for the supervision and coordination of the production line; and 2) they are connected to sensors and actuators to perform their own control actions. Both aspects have been well investigated in literature [9] with respect to traditional PLCs, but a general lack of research works on containerized vPLCs has been noted.

To clearly present and discuss results, the project has been organized into two parts, and this first article will deal only with the evaluation of the machine-to-machine (M2M) data exchange between PLCs or between PLC and SCADA.

The main contributions of this article are.

1) The definition of a methodology to compare the performance of PLC and vPLC from the point of view of the flow of data between machines (or supervisors).
2) The definition of an experimental setup with associated experimental procedure.
3) The definition of metrics to compare performance.
4) The application of the proposed methodology to a real industrial use case demonstrates its usefulness for modeling the system, drawing conclusions, and suggesting improvements to real-time behavior.

In the following, after the overview of the involved technologies and the existing literature, the proposed methodology is introduced and the use case is discussed. Finally, conclusions are reported.

II. Overview of Technology

It is important to point out the context of this work and the involved technologies that are used in the rest of the article.

A. Classical SCADA and PLCs-Based Architecture

A classic industrial system based on SCADA and PLC devices is shown in Fig. 1(a), it combines software and hardware components to supervise, coordinate, and control industrial processes.

The SCADA system serves as a centralized control system that collects, monitors, and analyzes data from multiple remote locations within the industrial environment. It consists of a supervisory computer, a human–machine interface (HMI), and a communication infrastructure. The PLC, on the other hand, is a specialized computer based controller that performs strict real-time control functions within the industrial processes; it talks with field devices, sensors, and actuators. SCADA retrieves soft real-time data from the PLCs, providing a centralized view of the whole production line.

Communication between SCADA and PLCs relies on M2M or on controller to controller (C2C [10]) protocols for sending commands and configuration parameters (to PLCs), and production-related information (to the SCADA).

B. Virtualization-Based Architecture

As described in [11], virtualization and containerization systems are speeding up the digital transformation of manufacturing. The rapid growth of virtualization technologies has opened new possibilities for industrial applications. Real devices often require specialized and costly hardware, making them less flexible and scalable. In contrast, virtual devices can leverage general-purpose hardware, which is more affordable and easily scalable and there is also an environmental aspect [12].

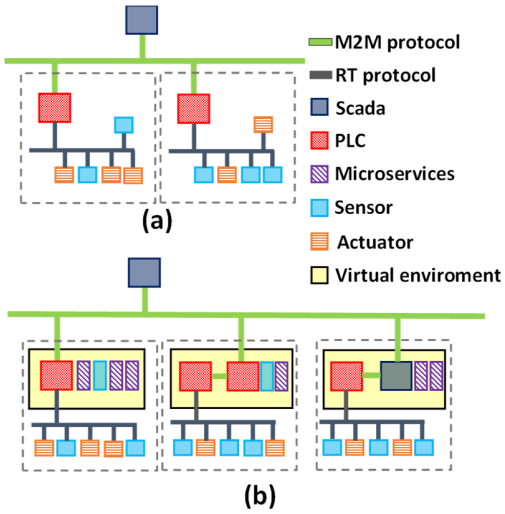

Fig. 1. (a) Classical automation architecture based on PLC and SCADA devices. (b) Virtualization applied to automation: old hardware devices are mapped to software services running inside virtual environments.

With the advent of virtualization techniques in the industry, the traditional architecture shown in Fig. 1(a) is still valid at the topological and communication level, but the single components implementation can be different.

In Fig. 1(b), it is shown an example of an architecture based on virtual environments. The components are the same as the traditional approach but there are virtual environments where PLCs, sensors, and SCADA can be virtualized. The protocols for M2M communication and sensor communication remain the same; they can be implemented directly in the virtual SCADA, in the vPLC, and also separately (as a microservice).

However, the adoption of virtual devices necessitates a thorough evaluation and comparison to determine their suitability for specific industrial use cases (e.g., real-time constraints). While real devices are bare-metal, so the performance is related to the hardware characteristics, the performance of virtual devices depends on many aspects such as: 1) virtualization technique, 2) operating system, and 3) hypervisor.

It is possible to define several virtualization techniques, the main ones are.

1) Full virtualization: this technique provides a high level of isolation and allows running multiple operating systems simultaneously on a single physical machine.
2) Containerization (light virtualization): containers offer a lightweight form of virtualization where the host operating system kernel is shared among multiple containers [13].

Each technique offers different levels of isolation, resource allocation, deployment systems, and flexibility. For this reason, it is necessary to pay special attention to the implementation of the virtual device.

III. Related Works and Research Objectives

In the literature, several works on the evaluation of custom vPLCs and microservices-based architectures can be found.

Software-defined automation solutions are analyzed by Javier Perez et al. [14], where they compared virtualized SoftPLC to a SoftPLC without hypervisor concluding that the vPLC can deliver similar performance in terms of switching time while having an increased period jitter.

Mellado and Núñez [15] proposed a containerized Internet of Things (IoT)-PLC (not fully IEC 61 131 compliant) running in a Raspberry Pi 4B board, they evaluated a four tanks control system scenario with a wireless communication system, obtaining latencies suitable for control applications if process variables change slowly.

Cruz et al. [16] proposed a vPLC that presents a convergent approach by virtualizing and co-hosting isolated PLC devices on the same physical equipment. This convergence consolidates distributed I/O on a networked I/O fabric, resembling the integration seen in datacenter architectures. Evaluation results indicate the feasibility of vPLC from a systems virtualization perspective, especially on ×86 platforms with room for improvement.

Dai et al. [17] designed an orchestration method and deployment procedures, IEC 61 499 compliant, based on microservice for industrial edge applications. A combined cloud and edge approach is described in [18].

Sollfrank et al. [19] evaluated a lightweight virtualization system for distributed and time-sensitive applications in industrial automation; they concluded that Docker containers can meet the soft real-time constraints of automation applications.

Catuogno et al. [20] proposed a methodology for the measurements of the computational resources used by a specific container.

However, different from the works discussed above, the goal of this research work is not to (propose and then) evaluate "custom" container based automation architectures. On the contrary, it is to design a methodology for the evaluation of existing architectures, with a special focus on the data exchange performance of commercially available solutions.

Rosa et al. [21] developed a framework comprising a basic vPLC running in a Docker container, equipped with an Open Platform Communications United Architecture (OPC-UA) middleware for IT and OT communications. For OT communications, a custom time-sensitive networking (TSN)-based OPC-UA configuration was utilized. The framework was evaluated on a practical testbed, which consisted of two edge nodes and an industrial network switch. The researchers concluded that the test environment demonstrated that the framework has low overhead, enhances determinism, and still maintains all of the benefits of virtualization.

In detail, the scope of this first article is to propose a method for evaluating communication performance at the supervision level (M2M, C2C, and SCADA). Operatively, this article includes also the discussion of a use case, where the M2M data exchange between vPLCs (virtualized with different light virtualization techniques) will be compared with the reference performance of their "equivalent" real PLCs.

IV. Proposed Methodology

The proposed setup for testing is illustrated in Fig. 2(a). Inside the architecture under test, the first step is to identify the two partners, referred to as C (Client PLC) and E (Edge PLC), that connect to each other using the M2M (or C2C or SCADA) protocol. The second step is to identify the physical network they use to exchange data. As a matter of fact, to assess the network latencies, and the communication stack delays of C and E, a physical network access, called T, is needed. By means of T, all the relevant data packets can be captured and analyzed.

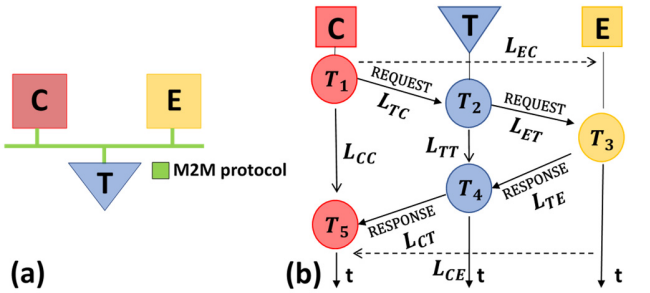

Fig. 2. Proposed methodology. (a) Measurement setup. (b) Exchange data diagram.

The third step is to isolate, in the M2M communication between C and E, the transaction type to be evaluated. For instance, in Fig. 2(b), it is shown that the case of a transaction of type "Request and Response." This case is very common in many supervision protocols. The method requires that for the two partners (E and C), and for T, a timestamp is taken and permanently saved for every event related to the identified transaction.

The last step of the method is to ensure that transactions are not time correlated. For this reason, a suitable randomization of the request must be introduced.

A. Metrics

The metrics of the proposed methodology are defined, without lack of generality, in the case of transaction of type "Request and Response." As a matter of fact, the other type of transaction is the "Publish," where one partner emits a message without being asked for. In practical systems "Publish with Acknowledge" and "Publish without Acknowledge" styles are possible, and the approach proposed here can deal also with them, as described after the metrics definitions.

The interaction between C and E is shown in Fig. 2(b): 1) at time $T1$ has generated the request; 2) at time $T2$, the request is seen on the network via T; 3) at time $T3$, the request is read from E; 4) at time $T4$, the response is visible on the network; and 5) at time $T5$, the response is read by C.

The following latencies are defined and evaluated.

1) $L_{\mathrm{CC}} = T_5 - T_1$, the request–response round trip time.
2) $L_{\mathrm{TT}} = T_4 - T_2$, latency is introduced by the elaboration of the request from the communication stack of E and the subsequent step of sending the response.
3) $L_{\mathrm{EC}} = T_3 - T_1$, the latency from the generation of the request to the reception of the request.
4) $L_{\mathrm{CE}} = T_5 - T_3$, the latency from the generation of the response to its reading.
5) $L_{\mathrm{TC}} = T_2 - T_1$, the request traverses latency from C to the bus.

6) $L_{ET} = T_3 - T_2$, the request traverses latency from the network to E, including the communication stack of E.
7) $L_{TE} = T_4 - T_3$, the response traverses latency from E to the network.
8) $L_{CT} = T_5 - T_4$, the response traverses latency from E to the network, including the communication stack of C.

For systems that use the "Publish with Acknowledge" transaction, the metrics are the same since the publish message coincides with the request and the acknowledge message is equivalent to the response.

For systems that use "Publish without Acknowledge" transaction, the subset of metrics valid for C to E directions is (L_{EC}, L_{TC}, L_{ET}), while the subset of metrics valid for C to E direction is (L_{CE}, L_{TE}, L_{CT}). L_{CC} and L_{TT} do not apply.

B. Synchronization

The proposed setup of Fig. 2 is a distributed measurement system. The measurement of traverse latency is affected by the drift and the offset between the time references of the devices that take the source and destination timestamps. In this work, as proposed in [21], all the devices must be time synchronized to compensate for the effect of drift and offset. For the synchronization, a specialized time transfer protocol called network time protocol (NTP) is used. The NTP synchronization protocol is based on exchanging packets between clients and servers, through the determination of: 1) offset of the client's local clock with respect to the server's clock, and 2) latency of the network connection. Observing the clock offset, the client can correct its local clock to match the server's time.

Still referring to [21] it is possible to evaluate the synchronization standard uncertainty as follows:

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

where σ_{sm} represents the standard deviation of the device m that takes the timestamp, and μ_{sm} is the systematic error that is necessary to consider because, in the experimental setup, no calibration is performed.

The standard uncertainty u_{mn} of any latencies evaluated between two devices (m and n), introduced in Section IV, is calculated as follows:

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

When the evaluated latency is calculated between the same device (2) becomes equal to $u_{mm} = \sqrt{2u_{sm}^2}$.

V. Use Case

The goal of the use case is to demonstrate the effectiveness of the proposed methodology. Currently, most vPLC solutions available on the market are built upon open-source Soft PLC IEC61131-3 compliant platforms and are executed on vendor-dependent Automation Platforms and/or Hardware. For example: 1) PLCNext by Phoenix Contact, featuring the PLCNext Control PLC based on the Linux kernel [23]; 2) ctrlX by Bosch Rexroth, offering a PLC App that supports target platforms based on ARM64 or x64, and Linux Ubuntu Core with real-time extension (called ctrlX OS [24]); and 3) software-defined automation solutions, previously introduced in [14].

The aforementioned solutions do not allow for a direct comparison between vPLCs and their real counterparts; thus, they are not the best choice for evaluating the proposed methodology.

A. Industrial System Used in the Use Case

In this use case, the Siemens vPLC CPU1582v is specifically addressed as an "equivalent" to real PLCs of the S7-1500 product family. This vPLC runs within a Docker container on the Siemens Industrial Edge (SIE) platform. The main components of the SIE platform are as follows: 1) the Industrial Edge Hub (IEH), located in the Cloud, which serves as a repository for documentation and containerized applications available on the marketplace; 2) the Industrial Edge Management (IEM), which runs locally and oversees the configuration and setup of edge devices and applications; and 3) the Industrial Edge Devices (IEDs), which refer to the actual machines running the containerized applications.

B. Experimental Setup for the Use Case

In this use case, the PLC models S7-1512C-1 PN (v2.6) and CPU1582V v0.30 (now called S71517V) assume the role of C and E in Fig. 2. Three types of devices have been used/implemented.

1) PLC S7-1512C-1 PN (called R) is a compact and powerful PLC from Siemens SIMATIC S7-1500 series. It features a fast CPU, expandable I/O modules, 250 Kbyte program memory, 1 Mbyte data memory, and support for real-time protocols (e.g., Profibus, Profinet, and Ethernet/IP).
2) CPU1582V hosted by Simatic IPC227E [called V_{IPC}, shown in Fig. 3(b)], a compact industrial PC boasting a quadcore Intel Celeron N2930 processor running at 1.83 GHz (burst frequency 2.16 GHz), 8 GB of main memory and a 240 GB SATA SSD. It executes the "IED OS" (version ied-os-1.9.0-27-amd64), which includes the Mentor Industrial OS (based on Debian Linux real-time) and the additional middleware for containerization.
3) CPU1582V hosted by desktop PC [called V_{PC}, shown in Fig. 3(a)], boasting a CPU Intel Core i7-7700b running at 3.60 GHz, 16 GB of main memory, and 500 GB SATA SSD and Windows 10 pro as Host OS. VMware Workstation pro 17.1 hypervisor is installed on the SSD it executes the "IED OS" (version 1.9.0-5-a-rc2). At the virtual machine are assigned 4 GB of memory, two processors, and two network bridge adapters (one for the connection with supervision IEM and the other for connection to the field level).

For the use case experiment, two devices of each type are used, to evaluate all the combinations in Table I. At each experiment, a reference number # *is*used in the following.

Referring again to Fig. 2, the network of the use case is Ethernet, and the network access is obtained using an Ethernet Tap (Profitap C1AP-100). The T duplicates all Ethernet traffic on the link and forwards it to an embedded system (Siemens IOT2050), which assigns timestamps and stores each packet.

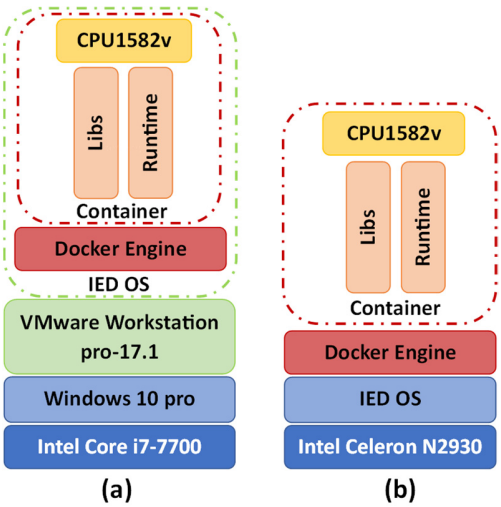

Fig. 3. Architectures of different virtualization solutions for CPU1582v: (a) commercial PC and (b) IPC227E.

TABLE I
EXPERIMENT MATRIX FOR THE COMPARISON (R: REAL PLC, V_{IPC}: VPLC ON IPC, V_{PC}: VPLC ON DESKTOP PC)

Ref. #	*Client (C)*	*Edge (E)*
1	R	R
2	R	V_{IPC}
3	R	V_{PC}
4	V_{IPC}	R
5	V_{IPC}	V_{IPC}
6	V_{IPC}	V_{PC}
7	V_{PC}	R
8	V_{PC}	V_{IPC}
9	V_{PC}	V_{PC}

C. M2M Protocol Used in the Use Case (S7comm)

To enable seamless communication between PLCs and supervisory systems, specific communication protocols are often employed. Siemens developed S7comm, which is the primary communication protocol for M2M, C2C, and SCADA. It is used by Siemens S7-300, S7-400, S7-1200, and S7-1500 families and external devices [25]. The protocol runs on ISO transport services on top of the TCP (TPKT) and all the communications occur on port 102. S7comm data are encapsulated in connection-oriented transport protocol (COTP) packets. The protocol incorporates security mechanisms such as authentication, integrity checks, and confidentiality using encryption algorithms. However, S7comm has faced vulnerabilities and attacks (see [26], [27], [28]).

There are three steps to establish a S7 connection with the PLCs [29]: 1) establish a COTP connection by sending a request and receiving the corresponding ACK, 2) S7 communication setup, and 3) exchange of S7 function code related to the transaction. In Fig. 4, it is shown an example of GET_DB instruction, it gets data from the desired data area of the server PLC and assigns them to the data area in the client PLC.

The S7 request is handled by the PLC operating system and it allows access to the specified memory area without disturbing the normal behavior of the PLC program.

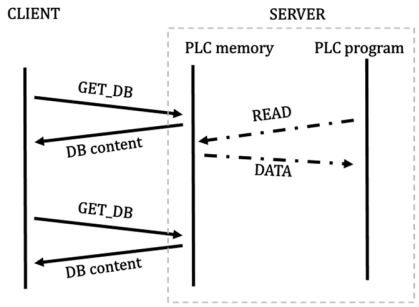

Fig. 4. Sequence diagram of S7comm GET Request.

TABLE II
NTP STATISTICS

	Poll time (s)	*Average Offset (ms)*	*Std. dev. Offset (ms)*
R	32	0.001	0.02
V_{IPC}	256	-0.258	0.52
V_{PC}	1024	-0.051	0.14
Tap	32	-0.016	0.56

D. Synchronization for the Use Case

In the proposed measurement setup, the connection with NTP Server (ntp1.inrim.it) is implemented as follows: 1) R via IOT2040; 2) V_{IPC} and V_{PC} via settings in SIE device management; and 3) Tap analyzer via IOT2050.

The performance statistics are taken through the NTP daemon, as shown in Table II.

From the values reported in Table II, the synchronization standard uncertainty is evaluated following Section IV

The resulting standard uncertainty and the corresponding expanded uncertainty, U, for this use case, calculated with a coverage factor of $k = 2$ are reported in Table III. The obtained resolution ranges from 0.1 to 1.7 ms, depending on the experiment. These values align with round-trip time latencies for industrial applications, as reported in [30].

In this article, the timestamping uncertainty , u_{tm}, is not taken into account because, with the considered hardware, it has a negligible impact (note that in previous work [21], the timestamping uncertainty has been estimated on the order of 0.01 ms).

E. Data Validation and Preprocessing

In this use case, the data exchange is implemented using S7 communication between C and E. The transaction type is "Request and Response;" specifically, C generates a GET Request to E for reading a variable ($T3$) inside one of its internal DataBlock (DB). The transaction is repeated every 10 s plus a random time between 0 and 999 ms.

In Table IV, statistics relative to the S7 protocol are listed for all evaluated experiments. In detail, Table IV shows that in the different experiments, the rates of S7 packets are similar, while the number of the total packets may change between the experiments.

TABLE III
STANDARD UNCERTAINTY AND EXPANDED UNCERTAINTY FOR EVERY METRIC AND EXPERIMENT

	L_{CC} (ms)		L_{TT} (ms)		L_{TC} (ms)		L_{ET} (ms)		L_{TE} (ms)		L_{CT} (ms)		L_{EC} (ms)		L_{CE} (ms)	
#	u_c	U	u_c	U	u_c	U	u_c	U	u_c	U	u_c	U	u_c	U	u_c	U
1	0.03	0.1	0.79	1.6	0.64	1.3	0.64	1.3	0.64	1.3	0.64	1.3	0.03	0.1	0.03	0.1
2	0.03	0.1	0.79	1.6	0.64	1.3	0.87	1.7	0.87	1.7	0.64	1.3	0.59	1.2	0.59	1.2
3	0.03	0.1	0.79	1.6	0.64	1.3	0.66	1.3	0.66	1.3	0.64	1.3	0.14	0.3	0.14	0.3
4	0.74	1.5	0.79	1.6	0.87	1.7	0.64	1.3	0.64	1.3	0.87	1.7	0.59	1.2	0.59	1.2
5	0.74	1.5	0.79	1.6	0.87	1.7	0.87	1.7	0.87	1.7	0.87	1.7	0.74	1.5	0.74	1.5
6	0.74	1.5	0.79	1.6	0.87	1.7	0.66	1.3	0.66	1.3	0.87	1.7	0.60	1.2	0.60	1.2
7	0.20	0.4	0.79	1.6	0.66	1.3	0.64	1.3	0.64	1.3	0.66	1.3	0.14	0.3	0.14	0.3
8	0.20	0.4	0.79	1.6	0.66	1.3	0.87	1.7	0.87	1.7	0.66	1.3	0.60	1.2	0.60	1.2
9	0.20	0.4	0.79	1.6	0.66	1.3	0.66	1.3	0.66	1.3	0.66	1.3	0.20	0.4	0.20	0.4

TABLE IV
S7 PROTOCOL PACKETS ANALYSIS

#	*Total*	*S7*	*S7 (%)*	*S7 Rate (packets/s)*
1	3222	504	15.6	0.2
2	17592	1376	7.8	0.2
3	11037	1480	13.4	0.2
4	116687	980	0.2	0.2
5	30444	708	2.3	0.2
6	15542	1723	11.1	0.2
7	11457	1387	12.1	0.3
8	18797	1959	10.4	0.2
9	19628	1128	5.7	0.2

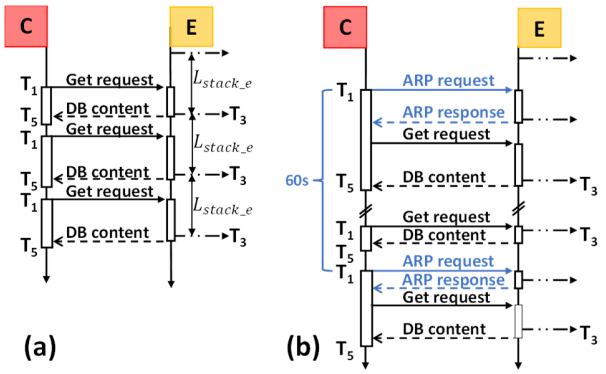

Fig. 5. Example of exchange in experiment #2: (a) normal exchange and (b) with ARP activity.

During the data analysis, it is observed that.

1) When the two partners C and E are both a real PLC, the transaction is very fast and the variability is extremely low. Such a situation suggests that the real PLC handles the IP stack (on which the S7 protocol lays) on interrupt.
2) When one of the communication partners is a vPLC (either V_{IPC} or V_{PC}), a large variability of the latency metrics is found. Such a situation triggered a deeper analysis of the reason for this behavior.

The analysis of the sequences of network packets, captured by the Ethernet Tap T, revealed that the implementation of the containerized CPU1582V (v0.30) seems to have an IP stack with a polling cycle equal to 100 ms [$L_{\text{stack\_e}}$ in Fig. 5(a)].

For the sake of clarity, Fig. 5 illustrates a temporal diagram that represents an example of data exchange in experiment #2 (see Table I). During normal operation [see Fig. 5(a)], C generates the GET Request at time $T1$. The request received from E is then processed at the end of the cycle introducing, at every request, a delay equal to the time required to reach the end of the IP stack cycle.

While investigating, another, less frequent, behavior of the IP stack has been observed. As shown in Fig. 5(b), C creates as usual the S7 GET Request at time $T1$ but, every 60 s, this request is queued because the IP stack of C is busy sending an address resolution protocol (ARP) request and then waiting until the ARP response arrives. This situation, in addition to the normal data exchange, results in a further delay on the order of hundreds of milliseconds.

In this article, for the metrics evaluation, only data pertaining to the normal behavior are considered and the data exchanges involving ARP activity are filtered out with preprocessing. This approach is justified by the sporadic nature of this behavior, and by the possibility to increase (or even eliminate) the ARP request configuring the IP parameters of clients. For sake of completeness not filtered metrics are reported in Tables V and VI. Fig. 6 shows the comparison between L_{CC} distribution with ARP activity Fig. 6(a) and filtered L_{CC} distribution Fig. 6(b).

VI. EXPERIMENTAL RESULTS FOR THE USE CASE

The primary goal of the proposed methodology is to provide useful insights into the system under test. For this use case, Tables VII and VIII report the results of the experiments. It is possible to observe the described behavior in Section V-E: when one of the communication partners is a vPLC, the round-trip time L_{cc} is on the order of 100 ms. Two examples of distributions are shown in Fig. 7(a) and (b).

In Fig. 7(a), it is possible to observe the evaluated latency for experiment #1 (i.e., only real PLCs) that works as a reference. The round-trip time (L_{CC}) has an average value of 3.3 ms.

Fig. 7(b) shows the results of experiment #5. Two cycles of 100 ms exist in the communication, one for C and one for E. The cycle on E is described by the L_{ET} latency, in particular is characterized by: average value of 51 ms, a maximum value of 100 ms, and a minimum value lower than U (see Table III). Similarly, the cycle on C is described by L_{CT} latency that is characterized by: an average value of 65 ms, maximum value of 74 ms, and a minimum value of 4 ms. These values are confirmed by analyzing L_{CC} (which is the sum of L_{CT} and L_{ET}) having an average value of 118 ms. In the latency L_{ET}, thanks to the random time added to the

TABLE V
ROUND-TRIP LATENCIES, FORWARD AND BACKWARD LATENCIES WITH ARP ACTIVITY ("—" MEANS UNDER RESOLUTION)

	L_{CC}(ms)				L_{TT}(ms)				L_{EC}(ms)				$L_{CE}(ms)$			
#	*Avg*	*max*	*min*	*Std*	*Avg*	*max*	*min*	*Std*	*Avg*	*max*	*min*	*Std*	*Avg*	*max*	*min*	*Std*
1	3.3	4.6	2.6	0.3	---	---	---	---	0.9	2.2	0.2	0.4	2.41	3.4	1.3	0.42
2	70.5	202.4	3.5	46.6	60	100	---	32	68	200	1	47	2	4	---	---
3	67.8	203.4	3.4	47.8	57	101	---	33	64.8	200.8	0.5	47.8	1.5	2.8	0	0.5
4	58	201	3	47	8	101	---	21	---	3	---	---	59	207	0	47
5	118	175	10	37	52	101	1	30	51	100	---	30	61	71	---	22
6	134	188	82	29	52	106	1	29	52	107	---	29	82	83	81	29
7	71.6	208.9	4.0	45.8	9	107	---	21	1	2	0	---	68	207	6	46
8	151	201	95	30	53	101	---	29	54	104	---	30	96	98	95	---
9	137	191	86	29	48	100	---	29	50	106	---	29	95	98	93	---

TABLE VI
TRANSMISSION LATENCIES BETWEEN DEVICES WITH ARP ACTIVITY ("—" MEANS UNDER RESOLUTION)

	L_{TC}(ms)				$L_{ET}(ms)$				$L_{TE}(ms)$				$L_{CT}(ms)$			
#	*Avg*	*max*	*min*	*Std*	*Avg*	*max*	*min*	*Std*	*Avg*	*max*	*min*	*Std*	*Avg*	*max*	*min*	*Std*
1	---	2	---	---	---	---	---	---	---	2	---	---	---	---	---	---
2	10	101	---	22	57	100	---	32	3	4	---	---	---	2	---	---
3	10	103	---	22	54	99	---	32	---	2	---	---	---	---	---	---
4	---	3	---	---	1	3	---	---	8	102	---	21	43	100	---	33
5	2	3	---	---	51	100	---	30	---	2	---	---	65	74	4	22
6	---	---	---	---	54	109	2	29	---	2	---	---	79	79	77	---
7	---	---	---	---	---	---	---	---	10	108	---	2	66	114	5	32
8	---	---	---	---	47	99	---	30	---	2	---	---	102	103	101	---
9	---	2	---	---	52	100	6	23	3	4	---	---	91	94	89	---

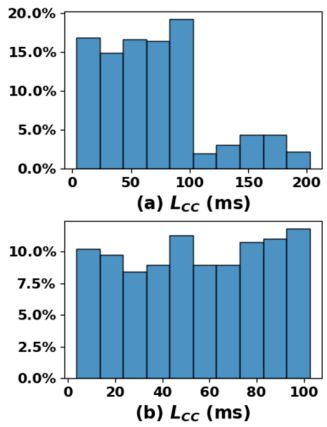

Fig. 6. L_{CC} distribution for experiment #2: (a) with ARP activity and (b) excluding ARP activity.

GET Request, all the possible values of the delays are tested and as shown in Fig. 7(b), a uniform distribution is obtained. On the contrary, in L_{CT} the situation is different because there is not the possibility to add random time to the response in the S7 protocol. Thus, the response arrives more frequently at the same point of the cycle of C, leading to a distribution where most of the samples are at 70 ms.

These cycles are well described in Table VIII. When C is a V_x (experiments #4, 5, 6, 7, 8, 9), L_{CT} latency has average values that ranged from 37 to 102 ms; these values may change depending on the synchronization status of C and E when the experiment is started. When E is based on V_x (experiments #2, 3, 5, 6, 8, 9), L_{ET} latency has average values that range from 47 to 54 ms and are characterized by a uniform distribution.

Again, the validity of the setup is confirmed by observing Table VII that the average value of L_{CC} is the sum of the average value of L_{EC} and L_{CE} (considering the expanded uncertainty).

In conclusion, comparing the average performance in this use case, the real PLC appears to be faster than the vPLC in completing the S7 transaction. However, if the minimum values of the latencies are compared, it is clear that the vPLC is as fast as the real PLC; the additional delay depends only on the implementation of the IP stack. Moreover, no noticeable differences are visible between the two types of virtual environments used for the vPLC. Hence, for this specific use case, the main suggestion for boosting performance is to ask developers to focus on enhancing the IP stack implementation of the vPLC.

A. Derived Analytical Model of the Use Case

The proposed methodology allows for creating an analytical model of the systems under test. For instance, considering the experimental results and referring to Fig. 5, an analytical model for the evaluation of the round-trip time L_{cc} can be derived as follows:

$$L_{cc,max} = L_{elab\_c,max} + L_{elab\_e,max} + L_{cable\_req,max} + L_{cable\_res,max} + \max(L_{stack\_c}) + \max(L_{stack\_e}) \quad (3)$$

where L_{elab_c} and L_{elab_e} represent the elaboration time of the C and E, L_{cable_req} and L_{cable_res} the request/response transmission time on the cable, and L_{stack_c} and L_{stack_e} the time taken by the IP stack of PLCs to read the request/response. Equation (3) is the worst-case scenario.

TABLE VII
ROUND-TRIP LATENCIES, FORWARD AND BACKWARD LATENCIES ("—" MEANS UNDER RESOLUTION)

	L_{CC}(ms)				L_{TT}(ms)				L_{EC}(ms)				$L_{CE}(ms)$			
#	*Avg*	*max*	*min*	*Std*	*Avg*	*max*	*min*	*Std*	*Avg*	*max*	*min*	*Std*	*Avg*	*max*	*min*	*Std*
1	3.3	4.6	2.6	0.3	---	---	---	---	0.9	2.2	0.2	0.4	2.41	3.4	1.3	0.42
2	54.6	102.8	3.3	28.8	52	100	---	29	51	100	---	29	2	4	---	---
3	50.8	103.6	3.4	28.7	48	101	---	29	48	101	1	29	1	3	---	0.5
4	45	102	3	32	---	2	---	---	---	3	---	---	38	97	---	30
5	118	175	10	37	52	101	1	30	51	100	---	30	61	71	---	22
6	134	188	82	29	52	106	1	29	52	107	---	29	82	83	81	29
7	57	114	4	29	---	2	---	---	1	2	---	---	60	116	7	29
8	151	201	95	30	53	101	---	29	54	104	---	30	96	98	95	---
9	137	191	86	29	48	100	---	29	50	106	---	29	95	98	93	---

TABLE VIII
TRANSMISSION LATENCIES BETWEEN DEVICES ("—" MEANS UNDER RESOLUTION)

	L_{TC}(ms)				$L_{ET}(ms)$				$L_{TE}(ms)$				$L_{CT}(ms)$			
#	*Avg*	*max*	*min*	*Std*	*Avg*	*max*	*min*	*Std*	*Avg*	*max*	*min*	*Std*	*Avg*	*max*	*min*	*Std*
1	---	2	---	---	---	---	---	---	---	2	---	---	---	---	---	---
2	2	2	1	---	50	98	---	29	3	4	---	---	---	2	---	---
3	2	2	---	---	46	99	---	29	---	2	---	---	---	2	---	---
4	---	3	---	---	---	2	---	---	---	3	---	---	37	97	---	29
5	2	3	---	---	51	100	---	30	---	2	---	---	65	74	4	22
6	---	---	---	---	54	109	2	29	---	2	---	---	79	79	77	---
7	---	---	---	---	---	2	---	---	2	3	---	---	59	116	6	29
8	---	---	---	---	47	99	---	30	---	2	---	---	102	103	101	---
9	---	2	---	---	52	100	6	23	3	4	---	---	91	94	89	---

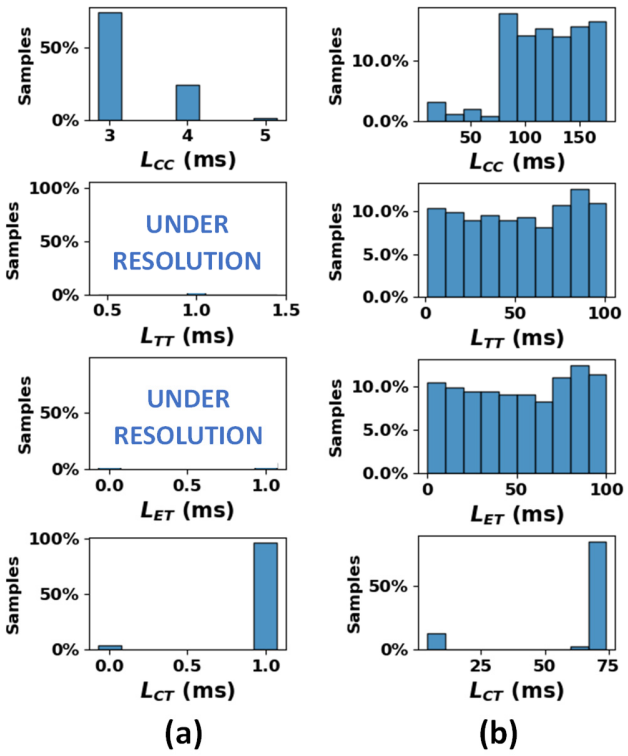

Fig. 7. Distributions of the evaluated latencies: (a) experiment #1 and (b) experiment #5.

Considering PLCs elaboration time equal for C and E, we can accumulate it into a single variable L_{elab}. The same can be done for the transmission time on cable L_{cable}.

Since communication in real PLCs works with interrupts, the real PLCs are considered $L_{\text{stack}} = 0$ ms. Due to these considerations, it is possible to simplify (3) in the following equation:

$$L_{\text{cc,max}} = 2L_{\text{elab,max}} + 2T_{\text{cable,max}} + j\max(T_{\text{stack}}) \quad (4)$$

where j is equal to the number of vPLC s involved in the scenario to be modeled.

A model of the average latency can be also obtained. As previously described if the GET request is not correlated to the response of the previous transaction, a uniform distribution is obtained for the latency. As it is possible to see in Table VIII, the average time spent in the stack is equal the half of L_{stack}. Thus, the model can be written as follows:

$$L_{\text{cc,avg}} = 2L_{\text{elab,avg}} + 2L_{\text{cable,avg}} + j\frac{L_{\text{stack}}}{2}. \quad (5)$$

From (5), it is clear that with two real PLCs involved in the communication ($j = 0$), the round-trip time is due only to the elaboration time of PLCs and the propagation time on the cables. On the other hand, with two vPLCs ($j = 2$), L_{cc} is mainly given by the latency introduced by the IP stack.

VII. CONCLUSION

The new vPLCs can be executed on any platform that supports containers, assuring independence from both the hardware and the operating system. They are maintainable, scalable, traceable, and open to new concepts of micro-service architectures in industrial automation. But what is the performance of vPLCs compared to (proven in use) real PLCs?

This article, the first of a multistage research work, provides a methodology for the evaluation of the communication performance of vPLCs when exchanging data for supervision, coordination, and control with other machines. For comparing

the performance, a set of metrics has been defined corresponding to the round trip time of the transaction, and to the transmission latencies between the devices. The methodology is completed by the proposal of a general experimental setup for measuring the relevant metrics across a distributed measurement environment. The synchronization of the devices under test (and of ancillary devices) is discussed, and their expanded uncertainty is taken into account.

The effectiveness of the proposed has been demonstrated by a use case, where real and vPLCs are compared. In detail, Siemens vPLC (CPU1582V) and real PLCs of the S7-1500 family are used in a scenario where they exchange supervision data by means of S7comm protocol. The vPLC is hosted by two different virtual environments, allowing for a comparative assessment of the type of virtualization.

Utilizing the recommended approach, the analysis of the use case results points out that: a vPLC could work as fast as a real PLC with average data exchange latencies L_{CC} in the order of 3 ms (and almost identical time distributions), but the IP stack implementation introduces a higher delay up to a maximum of 100 ms and an average of 50 ms. This insight information is useful for developers of vPLC that can work on reducing such delays.

Finally, the full access to the network traffic given by the proposed setup, combined with the fully synchronized timestamping, allows for the creation of an analytical model of latencies of the use case under test. The analytical model can be used for simulators or worst-case analyses.

In conclusion, in this article, a general methodology for the evaluation of the performance of vPLC is provided, when exchanging data with other machines and SCADA in a M2M scenario.

The second stage of the ongoing project will involve evaluating the real-time communication performance between the PLC and sensors/actuators. This evaluation will be conducted using advanced time measuring devices, including direct 1-PPS synchronization signal and GPS receivers, with a resolution of around 50 μs. However, it is important to note that this may require additional hardware and incur extra costs, resulting in a more expensive setup for real-time measurement.

Future evolution of virtual components (like vPLC, but not limited to) can greatly benefit from measurement and test procedures (like the ones described in this article) of their performance. As a matter of fact, vPLC implementations (being a full software approach) can be quickly improved by means of a combined cycle of design, test, and redesign.

ACKNOWLEDGMENT

The authors would thank Siemens, Italy, for the support during the implementation of the use case. This study is within the Made in Italy—Circular and Sustainable (MICS) Extended Partnership.

REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[2] T. Goldschmidt, S. Hauck-Stattelmann, S. Malakuti, and S. Grüner, "Container-based architecture for flexible industrial control applications," *J. Syst. Archit.*, vol. 84, pp. 28–36, Mar. 2018.

[3] D. Taibi, V. Lenarduzzi, and C. Pahl, "Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation," *IEEE Cloud Comput.*, vol. 4, no. 5, pp. 22–32, Sep./Oct. 2017.

[4] S. Sarkar, G. Vashi, and P. P. Abdulla, "Towards transforming an industrial automation system from monolithic to microservices," in *Proc. IEEE 23rd Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, vol. 1, Turin, Italy, Sep. 2018, pp. 1256–1259.

[5] P. Ferrari et al., "On the use of LoRaWAN and cloud platforms for diversification of mobility-as-a-service infrastructure in smart city scenarios," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–9, 2022.

[6] R. Queiroz, T. Cruz, J. Mendes, P. Sousa, and P. Simões, "Container-based virtualization for real-time industrial systems—A systematic review," *ACM Comput. Surv.*, vol. 56, no. 3, pp. 1–38, Mar. 2024.

[7] B. Scholten, "The road to integration: A guide to applying the ISA-95 standard in manufacturing," ISA, Pittsburgh, PA, USA, Tech. Rep., 2007.

[8] D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis, "A survey on SCADA systems: Secure protocols, incidents, threats and tactics," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1942–1976, 3rd Quart., 2020.

[9] M. Ayiad, E. Maggioli, H. Leite, and H. Martins, "Communication requirements for a hybrid VSC based HVDC/AC transmission networks state estimation," *Energies*, vol. 14, no. 4, p. 1087, Feb. 2021.

[10] A. Eckhardt and S. Müller, "Analysis of the round trip time of OPC UA and TSN based peer-to-peer communication," in *Proc. IEEE Int. Conf. Emerging Technol. Factory Automation*, 2019, pp. 161–167.

[11] T. Borangiu, D. Trentesaux, A. Thomas, P. Leitão, and J. Barata, "Digital transformation of manufacturing through cloud services and resource virtualization," *Comput. Ind.*, vol. 108, pp. 150–162, Jun. 2019.

[12] N. Carvalho, O. Chaim, E. Cazarini, and M. Gerolamo, "Manufacturing in the fourth industrial revolution: A positive prospect in sustainable manufacturing," *Proc. Manuf.*, vol. 21, pp. 671–678, Jan. 2018.

[13] E. Sisinni et al., "Assessment of time performance of lightweight virtualization for edge computing applications," in *Proc. IEEE 19th Int. Conf. Factory Commun. Syst. (WFCS)*, Pavia, Italy, Apr. 2023, pp. 1–4.

[14] D. Javier Perez, J. Waltl, L. Prenzel, and S. Steinhorst, "How real (time) are virtual PLCs?" in *Proc. IEEE 27th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Stuttgart, Germany, Sep. 2022, pp. 1–8.

[15] J. Mellado and F. Nùñez, "Design of an IoT-PLC: A containerized programmable logical controller for the industry 4.0," *J. Ind. Inf. Integr.*, vol. 25, Jan. 2022, Art. no. 100250.

[16] T. Cruz, P. Simoes, and E. Monteiro, "Virtualizing programmable logic controllers: Toward a convergent approach," *IEEE Embedded Syst. Lett.*, vol. 8, no. 4, pp. 69–72, Dec. 2016.

[17] W. Dai, Y. Zhang, L. Kong, J. H. Christensen, and D. Huang, "Design of industrial edge applications based on IEC 61499 microservices and containers," *IEEE Trans. Ind. Informat.*, vol. 19, no. 7, pp. 7925–7935, Jul. 2023.

[18] C. Pallasch et al., "Edge powered industrial control: Concept for combining cloud and automation technologies," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, San Francisco, CA, USA, Jul. 2018, pp. 130–134.

[19] M. Sollfrank, F. Loch, S. Denteneer, and B. Vogel-Heuser, "Evaluating Docker for lightweight virtualization of distributed and time-sensitive applications in industrial automation," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 3566–3576, May 2021.

[20] L. Catuogno, C. Galdi, and N. Pasquino, "An effective methodology for measuring software resource usage," *IEEE Trans. Instrum. Meas.*, vol. 67, no. 10, pp. 2487–2494, Oct. 2018.

[21] L. Rosa, A. Garbugli, L. Patera, and L. Foschini, "Supporting vPLC networking over TSN with kubernetes in industry 4.0," in *Proc. 1st Workshop Enhanced Netw. Techn. Technol. Ind. IoT Cloud Continuum*. New York, NY, USA: Association for Computing Machinery, Sep. 2023, pp. 15–21.

[22] P. Ferrari, A. Flammini, E. Sisinni, S. Rinaldi, D. Brand ao, and M. S. Rocha, "Delay estimation of industrial IoT applications based on messaging protocols," *IEEE Trans. Instrum. Meas.*, vol. 67, no. 9, pp. 2188–2199, Sep. 2018.

[23] (2023). *PLCnext Technology*. Accessed: Jul. 1, 2023. [Online]. Available: https://www.plcnext-community.net/infocenter/home/

[24] (2023). *ctrlX*. Accessed: Jul. 1, 2023. [Online]. Available: https://docs.automation.boschrexroth.com/doc/1368933374/ctrlx-core-runtime-application-manual/latest/en/

[25] (2023). *S7comm*. Accessed: Jul. 1, 2023. [Online]. Available: wireshark.org
[26] L. Martín-Liras, M. A. Prada, J. J. Fuertes, A. Morán, S. Alonso, and M. Domínguez, "Comparative analysis of the security of configuration protocols for industrial control devices," *Int. J. Crit. Infrastruct. Protection*, vol. 19, pp. 4–15, Dec. 2017.
[27] D. Beresford, "Exploiting Siemens simatic S7 PLCS," *Black Hat USA*, vol. 16, no. 2, pp. 723–733, 2011.
[28] R. Spenneberg, M. Bruggemann, and H. Schwartke, "PLC-blaster: A worm living solely in the PLC," *Black Hat Asia*, vol. 16, pp. 1–16, Jul. 2016.
[29] F. Xiao, E. Chen, and Q. Xu, "S7commtrace: A high interactive honeypot for industrial control system based on S7 protocol," in *Information and Communications Security*. Beijing, China: Springer, 2018.
[30] S. Figueroa-Lorenzo, J. Anorga, and S. Arrizabalaga, "A role-based access control model in modbus SCADA systems. A centralized model approach," *Sensors*, vol. 19, no. 20, p. 4455, Oct. 2019.

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 industrial Internet of Things, Industry 4.0, and sustainable mobility.

Paolo Bellagente (Member, IEEE) is currently an Assistant Professor with the Department of Information Engineering, University of Brescia, Brescia, Italy. His research field is about Internet of Things (IoT) systems for smart city and cognitive buildings applications.

Alessandro Depari (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 2002 and 2006, respectively.

He is currently an Associate Professor with the Department of Information Engineering, University of Brescia. His current research interests include sensor signal conditioning and processing, embedded systems, and design and development of systems for mobile health (mHealth) and Internet of Things (IoT) applications.

Alessandra Flammini (Fellow, IEEE) received the master's degree (Hons.) in physics from the University of Rome, Rome, Italy, in 1985.

After eight years with Ansaldo Industria, Milan, Italy, in 1995, she joined the University of Brescia, Brescia, Italy, where she is currently a Full Professor of electronic measurements. She has authored or coauthored more than 200 international articles. Her current research interests include electronic instrumentation, digital processing of sensor signals, smart sensors, and wired and wireless sensor networks synchronization.

Emiliano Sisinni (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 2000 and 2004, respectively.

He is currently a Full Professor of electronics with the Department of Information Engineering, University of Brescia. His research interests include wireless and wired networking for the Internet-of-Things (IoT) applications.

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 is about measurement systems for industrial and smart city applications, including performance and security analysis of real-time networks and Internet-of-Things (IoT) applications, wired and wireless sensor networks, and clock synchronization of distributed systems.

Open Access funding provided by 'Università degli Studi di Brescia' within the CRUI CARE Agreement