Industrial Internet of Things and Operational Technology Guided Industrial Evolution towards Digital Transformation in the Industry 4.0/5.0 Context.

Habilitation Thesis

Adrian Korodi

Contents

1 An overview of scientific, professional, and academic results

The current chapter presents briefly the author's evolution after defending the Ph.D. thesis considering scientific and professional perspectives, as well as regarding teaching and tutoring within the Automation and Applied Informatics department.

1.1 Scientific and Professional Activity.

After defending the Ph.D. thesis entitled "Contributions to the dependability analysis for automatic systems" in 16.11.2007, the author published a number of 57 works ([K-2]-[K-58]), and 1 paper [K-1] being submitted to a journal. 42 scientific papers are indexed in WoS.

Before stepping in other direction with the research, works [K-54]-[K-58] we more or less related to some parts of the Ph.D. thesis, focusing on availability and control of mobile robots.

Then, the author approached several different domains, being involved in interdisciplinary research topics in the following years. These research paths assured some scientific outcomes that were published as follows:

- [K-52]-[K-53], representing research that involved fuzzy models in the financial domain. The works were focusing on predicting bankruptcy based on current financial indicators. Currently both journals are WoS indexed Q3 journals with IF: 1.
- [K-45]-[K-51], approaching the biomedical engineering domain, particularly modeling components of the nervous control system regarding the cardiovascular system on short term. The modelling was focused on the vestibular nucleus and on vestibular receptors involved in scenarios as orthostatic stress, but also in the cardiovascular system elastance function and valve dynamics. Two of the 7 publications were WoS indexed proceedings.
- [K-44], representing an interdisciplinary research involving Markov models based predictions in the social work domain, regarding social marginalization of the elderly. The paper was indexed in WoS.
- [K-41]-[K-43] are researches referring automation in the photovoltaics domain. The works focused on modeling a PV panel using interpolation, followed by approaches assuring to reach the maximum power point in photovoltaic systems. One of the 3 works was published in a WoS indexed proceedings.

Paper [K-40], indexed in WoS, was conceived after several years of working with the industry, mainly as consultant in automation and SCADA. After year 2014, the author channeled all research focus on IIoT/IoT, Industry 4.0/5.0, Industrial automation and SCADA domains. Work [K-26] published in a Q1 WoS indexed journal presented an overview over the Industry 4.0 and IIoT development directions.

Other several works that were not included in the thesis, however within or related to the main domains are briefly enumerated in the followings:

- Some aspects referring digital transformation were published in two recent conference proceedings papers. One [K-6] is starting from the OT level to the cloud data transmission using MQTT and JSON, and one [K-5] approached the digital transformation in the context of legacy asset management system.
- The research approached some security considerations on the OT level, for automation and Supervisory Control and Data Acquisition (SCADA) systems. Legacy protocol level communication security is proposed using Trusted Platform Modules (TPM) and Elliptic curve cryptography (ECC). The legacy protocols related two papers focused on a Modbus TCP case study with MITM attack scenario [K-27], [K-13] were published and indexed in Q1 and Q2 WoS journal. ECC for OPC UA study was approached [K-11] and published in a WoS indexed proceeding. Also, while protocol conversion and data packing are essential for interoperability and interoperation, a honeypot for a water pumping station inside an OPC UA wrapper [K-29] was published in a WoS indexed proceeding.
- Some studies were somehow independent and were not integrated in a generic topic. Work [K-9] published in a Q2 Wos indexed journal is approaching image compression techniques. Paper [K-25] published in a Q2 WoS indexed journal is focused on enhancing the driver safety by protection against sun-glare in the automotive sector, while [K-37] published in WoS indexed proceedings is focusing on home-security systems in an IoT context.

Due to the large number of published works since defending the Ph.D., only part of them were selected to be presented in the current thesis. These studies are grouped into 4 chapters.

Chapter 2 depicts studies focused on IIoT/Industry 4.0 interoperability and relies on 10 published works, all WoS indexed. It presents solutions that are conceived for integrating legacy protocols, followed by works focusing OPC UA key enabler of the industrial revolution, new specifications, improvements and perspectives for industrial scenarios. Finally, emerging protocol studies, and protocol coexistence solutions are depicted for the automotive sector. Works [K-30], [K-31], and [K-38] were first important steps in the interoperability direction and were published in WoS indexed proceedings. Articles [K-19] and [K-21] were published as Q1 WoS indexed journals, while [K-2], [K-8], [K-15], [K-16], [K-18] papers were indexed in Q2 WoS journals.

The author's research was focusing also on new technologies referring to SCADA. Besides approaching Ignition software, considered one of the most influential and complex SCADA environments, 6 studies presented in chapter 3 were realized and published referring to IGSS, Android SCADA, and Node-RED SCADA. IGSS optimal resource allocation concept was published in [K-39], respectively a web module development in [K-34], both works being in WoS indexed proceedings. Mobile Android and OPC UA based SCADA solution was conceived and developed, first as a basic diagram and OPC UA client-server application [K-35] published in WoS indexed proceedings, and then as a complex runtime and development system that was published [K-10] in a Q2 WoS journal. Node-RED based SCADA was approached and published as a generic solution [K-21], and then developed in a complex application that was validated in industry [K-8], both works being published in Q2 WoS journals.

Eleven studies [K-7], [K-12], [K-14], [K-17], [K-20], [K-23], [K-24], [K-28], [K-32], [K-33], [K-36], oriented on increasing efficiency in an IIoT and Industry 4.0 oriented industrial evolution context are grouped in chapter 4. The works targeted industrial scenarios mainly in the water sector (9 articles), but also in the automotive manufacturing (2 articles). Papers [K-32], [K-33], [K-36] and [K-14] were published in conference proceedings, the first 3 being WoS indexed. Article [K-23] was published in a Q3 WoS indexed journal. Works [K-7], [K-12], [K-17], [K-24], [K-28] were published in Q2 WoS indexed journals, while [K-20] is Q1 WoS indexed.

Three very recent studies [K-1], [K-3], [K-4] are constituting chapter 5, and are focused on structured and contextualized data propagation in an Industry 5.0 and digital transformation context. Works [K-3] and [K-4] were published in the summer of 2025 in conference proceedings, while [K-1] is being submitted to journal.

Since defending the Ph.D. thesis, the author was the director of the following research and development projects:

1. Efficiency increase in water domain systems functioning through proactive supervision (EFICIENT)/: 77PTE/2022, 27/06/2022-27/12/2023 – Director at partner.

Works [K-7], [K-10], [K-11], [K-12], [K-13] were between the outcomes of project 1.

2. Centralizing and optimizing SCADA in the water sector / Bridge Grant cod PN-III-P2-2.1-BG-2016-0208, 2016-2018 - Grant director.

Works [K-31], [K-32], [K-33], [K-34], [K-35], [K-36], [K-38], [K-39], were among the outcomes of project 2.

3. Project with Continental Automotive entitled: Artificial Intelligence Based Prediction in the Electronic Manufacturing, 01.04.2022-31.01.2023 – Project director

Work [K-14] was among the outcomes of project 3.

4. Project with Continental Automotive entitled: Industry 4.0 Node-RED Integration solutions for Building Management System Components, 01.02.2023-01.06.2023 – Project director
5. Project with Continental Automotive entitled: Industry 4.0 Node-RED Integration Solutions for Building Management System Components – Extended Research, 01.06.2023-01.10.2023 – Project director
6. Grant Continental Automotive entitled: Researching Facility Management Industry 4.0/IIoT Solutions Regarding Integrability/Interoperability and Supervision, 01.10.2021-31.03.2022 – Grant director.
7. Project with Continental Automotive entitled: Soluție software în Node-RED de interfațare, integrare, monitorizare, stocare date de proces, 01.04.2020-01.06.2020 – Project director
8. Project with Continental Automotive entitled: Researching and Developing Node-RED Integration Solutions for Building Management System Entities, 01.05.2022-01.07.2022 – Project director

Work [K-8] was among the outcomes of projects 4-8.

9. Project with Hella entitled: Image processing solutions for equipment testing in the automotive industry, 2017 – Project director.
10. Project with Hella entitled: Prototype research and development for image processing solution for ECU testing in automotive manufacturing, 2018-2019 – Project director

Work [K-20] was among the outcomes of project 9-10.

The Hirsch index of the author is 12 in WoS, 14 in Scopus, and 17 in Google Scholar.

The author was invited to review for various journals, such as:

- IEEE Transactions on Industrial Informatics;

- IEEE IoT Journal;
- IEEE Access Journal;
- Journal of Manufacturing Systems;
- Sensors;
- IEEE Open Journal of the Industrial Electronics Society;
- Applied Sciences;
- International Journal of Critical Infrastructure Protection;
- Sustainability;
- Journal of Photovoltaics;
- Journal of Process Control;
- Energy Sources, T&F;

etc.

The professional experience of the author consisted also in a significant consulting and development activities in the automation/SCADA domain for companies as Louis Berger, Eddacon, CCAT, Tadeco, etc., in various locations (e.g. Timis, Constanta, Bihor, Ilfov, Satu-Mare, etc.). The activities provided access to latest industrial technologies regarding equipment and solutions.

Also, the author was external evaluation expert in the European Commission's Horizon programme. This activity provided access to state-of-the-art research directions.

1.2 Didactic Activity

The activity within the Department of Automation and Applied Informatics took place in the following periods of time: 2003 – 2008; 2009 – present, as Ph.D. student, Assistant Professor, Lecturer, Associate Professor.

The teaching activity consisted of the followings:

- Undergraduate studies:
    - Industrial SCADA solutions,
    - Industrial Internet of Things.
    - Industrial IoT and Microcontroller Systems Project.
    - Automation Elements,
    - Linear Systems Theory,
    - Nonlinear Systems Theory,
    - Systems Theory and Automation,
    - Computer Programming,
    - Object Oriented Programming,
    - Mechatronic Project,
- Graduate studies (Master Programs):

- Automation in Photovoltaic Systems (Renewable energy, solar energy),
- Complements of Systems Theory (Automatic Systems Engineering)
- Quality Engineering (Automatic Systems Engineering)

Among the listed subjects, the author introduced the following courses within the Department of Automation and Applied Informatics, and contributed significantly for providing the students possibilities to acquire practical skills along with theoretical concepts, in the context of the current industrial world:

- Industrial SCADA solutions – 4th year Systems Engineering – Course and Applications,
- Industrial Internet of Things – 4th year Systems Engineering – Course and Applications,
- Industrial IoT and Microcontroller Systems Project – 2nd year Informatics - Project.

Regarding undergraduate and graduate student coordination after obtaining the Ph.D. degree, the didactical activity consisted of the followings:

· Scientific coordinator of more than 140 diploma and dissertation projects.
· Close advisor of 6 Ph.D. students, coordinated by Prof. Ioan Silea.
· Coordinating the research activity of various students within research groups and guidance towards scientific publications.

Between 2008-2009 the author was a Visiting Professor at the University Tecnologico de Monterrey, Mexico, where the teaching activity consisted of the following courses:

(Department of Mechatronics)

- Digital Control,
- Control Engineering,
- Microcontrollers,
- Automatic Control Laboratory,
- Mechatronic Projects.

(Department of Computer Science)

- Intelligent Systems,
- IT Project Management.

Following the Ph.D. thesis, two books were published, [K-59] and [K-60]. [K-60] was published in 2008 and it is a book supporting the teaching activity that involves C based programming. [K-59] was published in 2015 and it supports SCADA, industrial automation and also IIoT related teaching activities being focused on IGSS and Ignition SCADA environments, PLC and HMI touch panel programming and OPC interfacing.

2 Industrial Interoperability Issues and Solutions in Industry 4.0

Section 2.1 presents solutions that assure interoperability for legacy systems. These legacy systems comprise of various protocols and local technologies that represented challenges throughout the years. The target always represented interoperability/interoperation and open platform technologies that assure a high TRL. The desired output protocol was Open Platform Communication Unified Architecture (OPC UA), but there were situations where for digital transformation Message Queue Telemetry Transport (MQTT) protocol was also envisioned. Information in Section 2.1 is relying on papers [K-8], [K-31], [K-38].

Section 2.2 takes further the research regarding the OPC UA protocol and approaches new specifications that rely on publish-subscribe mechanism and bring the interfacing closer to real-time. Improvements were approached, consisting of synchronization algorithm and multithreading broker over UDP, respectively multi-channel communication and image transmission. The majority of implemented industrial products based on OPC UA do not include the publish-subscribe mechanism. Regarding the fact that the Data Distribution Service (DDS) protocol emerged in robotic manipulators, a DDS-OPC UA protocol coexistence solution in real-time using non-ideal infrastructure was conceived. Information from Section 2.2 was published in works [K-16], [K-18], [K-19].

Section 2.3 is focusing on emerging protocols, gateway and protocol coexistence solutions in the automotive sector, due to the fact that in-car automotive solutions rely on some accepted protocols, different from the manufacturing industry. The automotive sector is slowly including Ethernet-based protocols like DDS, Scalable Service-Oriented Middleware over IP (SOME/IP), enhanced Communication Abstraction Layer (eCAL). In the context of vehicle to everything (V2X) concept, protocol coexistence solutions were approached for automotive in-car protocols, also OPC UA being considered. Zenoh emerging protocol is a new candidate for the automotive sector, and the current section aims to provide a useful comparison between Zenoh and DDS. Information from Section 2.3 was published in works [K-2], [K-15], [K-21], [K-30].

2.1 Providing Interoperability for Legacy Systems.

The Industrial Internet of Things (IIoT) means practically a world of interconnected devices. Besides the physical communication support, the most important enabler of Industry 4.0 is the interfacing. Protocol related advancements are responsible for eliminating language barriers between industrial devices and to create a proper frame for exchanging data. The key enabler in the operational technology (OT) level is the OPC UA protocol, previously called Object Linking and Embedding for Process Control Unified Architecture.

In industrial environments, automation equipment typically has a long operational lifespan, and systems are often designed with proprietary protocols, making interoperability a challenging endeavor. Ensuring seamless integration within local automation systems can be very complex, and many times changes in functional structures must be minimal. Legacy systems refer to outdated computing software, hardware, technologies, and protocols that are still widely used by many companies and remain essential for daily operations.

Moving basic automation on the OT level towards more complex software structures demands interfacing structures and properly trained integrators. Field elements are usually integrated and controlled by the first level automation structures based on PLCs. First level and further higher-level SCADA integration are necessary for the operators. Furthermore, data will have to be unified with IT level data coming from other software applications (e.g. ERP). IIoT/IoT concepts are relying on universal interfacing and communication. The main goal of IIoT/IoT is to obtain communication in a proper language among all equipment, even on the same hierarchical level, and therefore each device must have the capability to exchange data in a universal way.

The OPC UA interfacing is the key enabler from the integration/interoperability point of view. Practical experience in automation/SCADA implementations points out that OPC UA is growing exponentially in coverage, but still many industries are facing issues in moving forward from legacy protocols. OPC UA is viewed differently by the integrators when it refers to local automation integration. PLC or field device integration into higher-level structures through OPC UA is a target, but many times the protocol is not exposed for interoperability. The industry becomes more and more involved towards providing products that include OPC UA Client and/or Server for the local level. The industry is active in developing embedded OPC UA solutions for PLCs (e.g. Beckhoff, Siemens, Schneider), HMI panels (e.g. Siemens, Schneider Electric), gateways (e.g. Softing, Matrikon), SCADA (e.g. Inductive Automation, Schneider Electric, Siemens). But, still considered in many cases are centralizing OPC UA Servers residing on a central processing unit (e.g. OPC UA Server from PTC – each specific local protocol with the specific driver license, Telecontrol Server Basic – product oriented for Siemens PLCs, etc.).

Technologies evolved at the PLC level and products exist with OPC UA server included. Obviously some of them are missing essential specifications (e.g. security), and many integrators are not making the interfacing available because of incomplete tendering documents.

Initial important studies referring OPC UA dating around years 2016-2017 were focused on different aspects related to interoperability/integration. Authors in [1] are presenting implementations regarding OPC UA Servers applied on PLCs and OPC UA Client development for Onevue. In [2], OPC UA is used for the vertical plant integration to provide the process data to all higher-level applications. In [3], OPC UA interface is considered in characterizing intelligent cyber-physical sensor systems. Other papers like [4], [5] are presenting developments focused on integrating OPC UA servers for monitoring and controlling production processes, respectively in [6] the authors are considering OPC UA to achieve interoperability of micro-grid platforms. Using the above-mentioned information, when starting a new implementation there are few choices to make the local automation panel interoperable through OPC UA. In practice, the chronology of developments and the life cycle of the structures are highly reflected in the interfacing and interoperability. As noted in [7], manufacturing systems often require local processing units to communicate via OPC UA and existing architectures present challenges that make OPC UA interfacing difficult to implement. Also, papers like [3], [7], [8] were mentioning the need of a middleware structure used as wrapper to obtain interoperability.

In the following years, dating to year 2018, the industry continued to be concerned with connecting its physical part with the digital infrastructure, respectively to provide interoperability to the entities. Following Industry 4.0 principles and corresponding studies, the research and industrial community continued to ground OPC UA as the key IIoT protocol on the OT level (e.g. [9], [10], [11], [12]). The OPC UA continued to function on client-server basis, being is platform independent, including security modes and policies, allowing easy addressing. It provides an address space on the server side containing brows-able nodes, it may include classic OPC features (DA - Data Access, A&E - Alarms and Events, HDA - Historical Data Access), etc. Research studies related to OPC UA were implementing OPC UA servers and clients associated for different hardware-software equipment. The OPC UA sever research and development was approached for various process structures (e.g. on the sensorial level in [13], [14], various OPC UA server developments for the process parts in [15]), and some are considering various issues/applications related already functional OPC UA servers (e.g. redundancy in [16], web-based platform for OPC UA in [17], etc.).

Automation equipment typically operates for extended periods, resulting in numerous legacy systems where invasive modifications are generally avoided. Industry efforts focus on making these systems interoperable and integrating them into higher-level supervisory applications. At the same time, horizontal communication between entities is sought to enhance flexibility and adaptability. To meet these requirements, non-invasive interfacing with local automation, protocol conversion for legacy systems, and deployment of local OPC UA servers enabling both horizontal and vertical interoperability, a middleware solution is essential.

OPC UA based middleware solutions for the industry were researched in 2016-2018 (e.g. [18], [19]) to provide interoperability for the local equipment. As generally known, serial Modbus was one of the most widespread protocol used in the industry. Even now, serial Modbus and Modbus TCP are omnipresent on the first level PLC integration (e.g. measuring equipment, frequency converters).

Beyond ensuring interoperability, local control structures often require further development or improvement without altering existing PLC software due to constraints such as warranty restrictions, limited implementation details, or missing development licenses. In such cases, an OPC UA hardware gateway that cannot augment the software application proves unsuitable, even when the local PLC itself offers compatibility.

When implementing industrial structures, several key factors are prioritized: minimizing development time, reducing process downtime to near zero, controlling costs, and ensuring ease of maintenance and future scalability.

Considering the above-mentioned aspects, section 2.1.1 presents based on information from papers [K-38], [K-31] two researched OPC UA wrapping structures. The two wrapping structures approach Modbus serial and Modbus TCP as basic protocol. Both solutions were applied in the water industry and present flexibility to be adapted for other basic legacy protocols. The first one is a low-cost middleware OPC UA wrapping structure based on Node-RED and Raspberry Pi. The second solution presents a serial Modbus to OPC UA wrapping solution with IoT-2040 as hardware and Node-RED as software environment. The wrappers provide the possibility to monitor and control the local system, to store and query data into/from a local database, to further implement control algorithms for existing structures without modifying the local software.

Currently, the integration of networking, interfacing technologies, and smart computing in manufacturing continues under the Industry 4.0 paradigm [20]. Core IIoT principles like interconnection of devices anytime and anywhere are applied to improve safety, efficiency, and productivity. The rapid evolution and widespread adoption of IIoT and Industry 4.0 in recent years have significantly impacted multiple domains and reshaped traditional manufacturing organizations [21, 23]. The key towards transformation is to assure interoperability. This can be solved through integration of IIoT legacy and new protocols and technologies [22], but also legacy IoT web-based techniques which sometimes represent the only available option. The objective is to design systems capable to monitor, collect, exchange, analyze, and deliver information, structured as networks of interconnected industrial devices that employ communication technologies to achieve interoperability [23].

Although legacy technologies are outdated from a modern perspective [24], many remain essential for enterprise infrastructure, making replacement difficult. Current industry trends highlight the need for evolution and expansion of manufacturing and monitoring processes, driving the integration of legacy solutions into IIoT networks [23, 25, 26]. When targeting digital transformation and the requirement to unify OT-IT levels, a current important approach is towards the MQTT protocol, as a broker based solution with fast deployment. Therefore, classic OPC UA to MQTT conversion is important for OT systems to reach a common ground with the IT and cloud level.

Section 2.1.2 presents based on paper [K-8] the integration issues and solutions in the automotive manufacturing industry, particularly in a Building Management System (BMS) facility, where various legacy systems are functioning. Also, an OPC UA to MQTT conversion solution in Node-RED is provided.

2.1.1 Assuring Interoperability through Modbus to OPC UA conversion

The section introduces first a cost-effective middleware solutions based on OPC UA wrapping structure, aimed at facilitating interoperability within local automation networks. The OPC UA wrapper not only enables system monitoring and control but also supports the creation of a local data archive and the implementation of advanced control algorithms. It also enhances tag packaging for structured data integration into SCADA systems without requiring changes to the existing local software. The wrapper is conceived to be a complete hardware-software solution that complements the local automation structure.

The research considered various hardware and software environments in order to choose the most suitable variant for a high TRL, low cost, and minimized implementation times and local process downtime until deployment. From the hardware point of view several choices were taken into consideration, with selection criteria oriented towards the cost issue, capabilities, industry focus (e.g. powering, physical communication support, enclosure and industrial deployment possibilities, operating system), and device popularity (e.g. higher chance of adoption by integrator, implicit higher reliability).

From a software perspective, the analysis began with OPC vendors to identify environments capable of addressing integration challenges. OPC DA/UA wrapper solutions were examined for their role in connecting local SCADA servers to higher-level control centers, including products from Unified Automation, Matrikon, etc. Although the Windows-oriented DA client was not a primary objective, mature wrapping solutions on similar equipment were reviewed to assess their evolution. The main focus remained on OPC UA servers and clients, implemented either through SDKs or installation-ready products, with several key issues considered: platform independence and easy deployment on hardware suitable for automation panels, with devices such as Raspberry Pi providing sufficient performance; protocol conversion via middleware, enabling local protocols to be integrated into OPC UA servers, targeting open-source solutions; high technological readiness to ensure rapid implementation in real applications; modularity and flexibility to support new algorithms and seamless integration with both local and higher-level modules (e.g., OPC UA servers); ease of knowledge transfer to automation and SCADA integrators; low cost for both usage and development; additional features, such as local database support and lightweight SCADA functionalities.

One environment of an OPC products developing company could not be chosen considering the upper mentioned issues. Two java based software environments were considered to be the most appropriate for the final solution: Ignition and Node-RED. Node-RED was chosen, being a lightweight, open-source and free environment that covers all the presented issues. Although Ignition was somehow closer to automation/SCADA integrators and Node-red was coming at that time from other software levels, the flexibility and openness of Node-red, respectively its flow oriented programming style would be better exploited and of greater industrial impact for the OPC UA wrapper concept.

The first step in the wrapper development is the interfacing with the local structure. The solution was prepared to be tested in real-world applications, in a case study for the water industry. Modbus TCP was the local protocol implemented at the PLC level. The Modbus TCP client node was foreseen to read an array of values from a holding register, starting from an initial address, using a poll rate of 15 seconds that was considered sufficient for the wastewater pumping stations (WWPS).

After reading from the local structure, the next step involves identifying individual variables through function blocks and applying bitwise masks. The separated tags can then support small-scale monitoring and control via dashboard packages or direct web solutions. A key feature of the wrapper is its ability to implement supplementary control algorithms without altering the existing local configuration, using function blocks and payload transfers. Moreover, variable restructuring and grouping through function blocks can reduce higher-level SCADA licensing costs (e.g., combining digital alarm and state bits into words) and optimize SCADA integration.

Database connectivity within the middleware wrapper can be achieved using packages such as SQLite, MySQL, MSSQL, or PostgreSQL. The wrapper's purpose was to enable a lightweight local database for analysis, leading to the selection of SQLite. After creating the database, data will be inserted as presented in Fig. 2.1-1. The array obtained from the local PLC is restructured to select and group tags into a final array. A timestamp is appended and converted into readable data, after which an insert function is generated and the payload transmitted to the designated SQLite database.

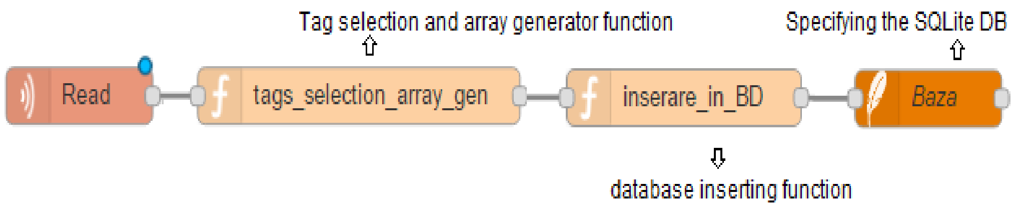


Fig. 2.1-1 Inserting into the SQLite database

The final step involved creating the OPC UA server and inserting tag values. The server was established using the OPC UA Server Node, with initial folder–tag structuring required to enable future browsing. OPC UA commands (*addFolder*, *addVariable*) are transferred successively as payloads to the OPC UA Server node. Namespace index (*ns*) and channel/tag name (*s*) are transferred as topic to the OPC UA Server node.

Following the creation of the OPC UA server and its folder–variable structure, the subsequent step was the continuous insertion of values into the defined tags (e.g. see Fig. 2.1-2). Processed values from the local automation are transferred as payloads to OPC UA items defined by namespace, tag name, and datatype. The OPC UA client then operates the write procedure using the specified server endpoint address and the designated action type.

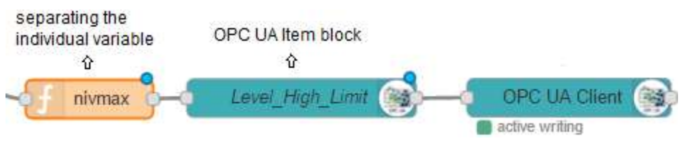


Fig. 2.1-2 Inserting tag values in the OPC UA server

After testing the Modbus TCP – OPC UA wrapper solution in the laboratory, the following step was represented by a real test scenario in the water sector. The WWPS comprised a dual-pump electro-mechanical system equipped with a level transducer, flowmeter, PAC3200 electrical parameter unit, intrusion and gas leakage sensors. Control strategies were implemented through an S7-1215 PLC. Additional local equipment included a CSM unmanaged switch for network aggregation, a Geneko 3G communication module, and a KTP 600 HMI. An automation panel of a WWPS is depicted in Fig. 2.1-3.

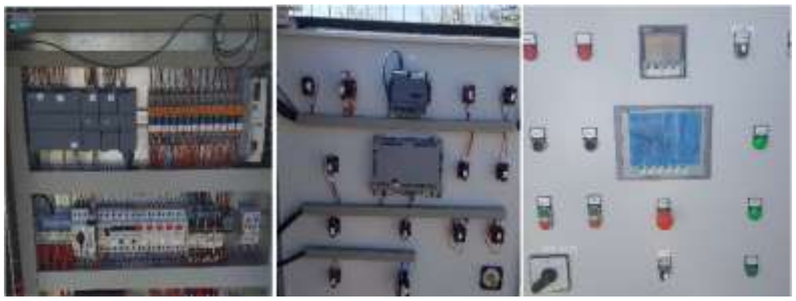
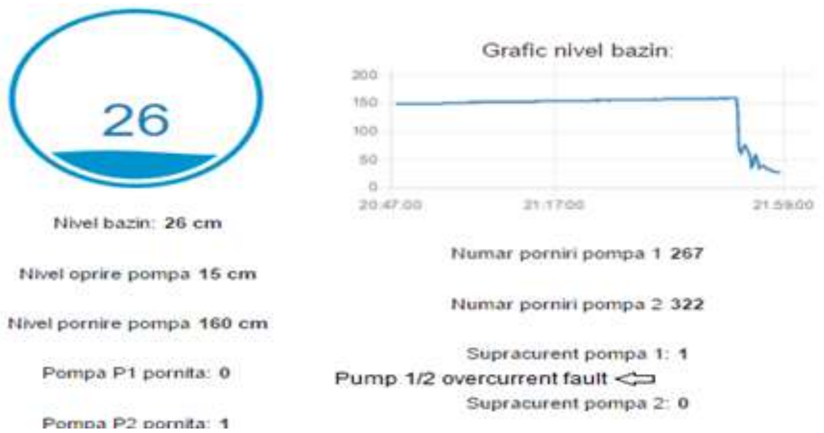


Fig. 2.1-3 The WWPS automation panel

Fig. 2.1-4 details a dashboard screenshot from the wrapper application, presenting the emptying procedure of the WWPS. Reaching the high-level limit and noticing the overcurrent fault at pump 1, the local algorithm starts pump 2. Fig. 2.1-5 displays the most recently extracted rows from the SQLite database corresponding to the exposed status.



Nivel bazin: 26 cm

Nivel oprire pompa 15 cm

Nivel pornire pompa 160 cm

Pompa P1 pornita: 0

Pompa P2 pornita: 1

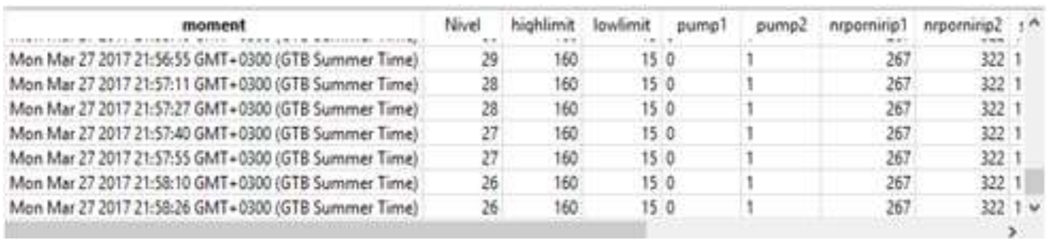
Numar porniri pompa 1 267

Numar porniri pompa 2 322

Supracurent pompa 1: 1

Supracurent pompa 2: 0

Fig. 2.1-4 Node-red dashboard – WWPS emptying



moment	Nivel	highlimit	lowlimit	pump1	pump2	nrpornirip1	nrpornirip2	s
Mon Mar 27 2017 21:56:55 GMT+0300 (GTB Summer Time)	29	160	15	0	1	267	322	1
Mon Mar 27 2017 21:57:11 GMT+0300 (GTB Summer Time)	28	160	15	0	1	267	322	1
Mon Mar 27 2017 21:57:27 GMT+0300 (GTB Summer Time)	28	160	15	0	1	267	322	1
Mon Mar 27 2017 21:57:40 GMT+0300 (GTB Summer Time)	27	160	15	0	1	267	322	1
Mon Mar 27 2017 21:57:55 GMT+0300 (GTB Summer Time)	27	160	15	0	1	267	322	1
Mon Mar 27 2017 21:58:10 GMT+0300 (GTB Summer Time)	26	160	15	0	1	267	322	1
Mon Mar 27 2017 21:58:26 GMT+0300 (GTB Summer Time)	26	160	15	0	1	267	322	1

Fig. 2.1-5 Database View

The implemented OPC UA server was accessed by a higher level IGSS SCADA application using its OPC UA client for testing. As seen in Fig. 2.1-6 the browsing procedure finds the tags within the OPC UA server, and is able to see the values of the variables (e.g. 160 for the *Level_High_Limit* as in Fig. 2.1-4 and Fig. 2.1-5), respectively to proceed to atom mapping.

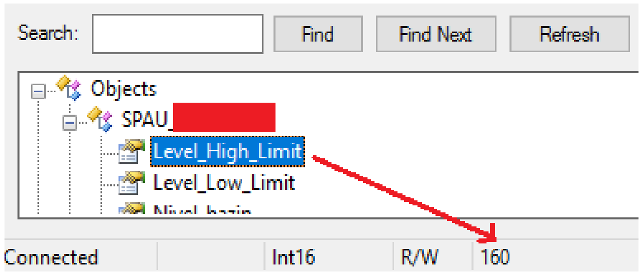


Fig. 2.1-6 Browsing the previously defined OPC UA server

After obtaining the OPC UA based middleware structure applied as a wrapper structure for Modbus TCP conversion, the following study presents a serial Modbus - OPC UA wrapper solution. The structure was designed for real-world application, using IoT-2040 hardware and Node-RED as the software environment. IoT-2040 features, 1 GB DDR3 memory, Intel Quark X1020 processor, microSD storage, dual Ethernet ports, RS-232/RS-485 interfaces, USB, 24 VDC supply, and industrial enclosure, provide high processing capacity, versatile connectivity for local automation, seamless panel integration, and industrial-grade reliability. The physical support for distance communication will be assured by the RUT240 router.

From a functional point of view, as shown in Fig. 2.1-7 the serial Modbus client transfers data to the filtering module, which analyzes, splits, and maps values to local variables. The processing module then manipulates these variables, implements diagnostic and protection structures, and extends local logic with additional algorithms. The structuring module defines the data format for OPC UA representation, while the sampling control module schedules tag injection to minimize bandwidth consumption. Finally, the node value injection module updates the specified node within the OPC UA server.

When an external OPC UA client initiates a control action on a process tag, the node value change module detects it, while the Modbus structuring module maps the information to a corresponding Modbus address. The value injection module then transmits data to the Modbus client, which acts on the external Modbus slave. Additionally, two monitoring and safety modules were implemented for both the OPC UA server and Modbus client, ensuring application status tracking and executing critical safety functions such as restarting the server, client, or the entire application.

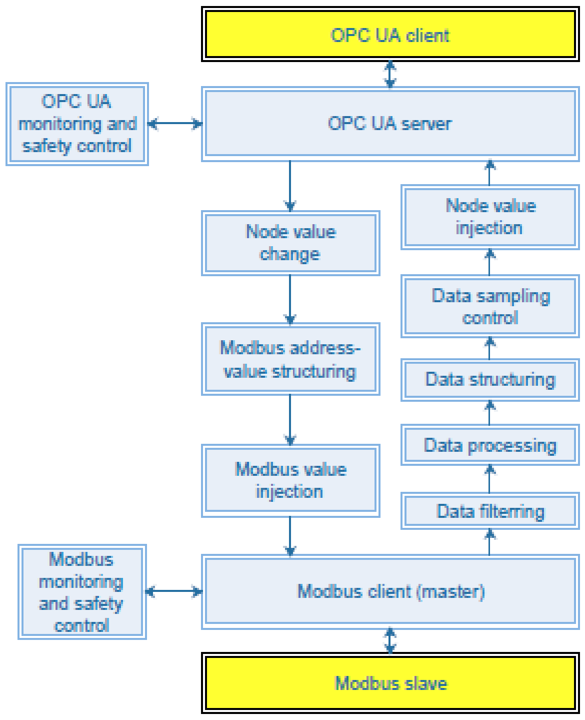


Fig. 2.1-7 Functional overview of the serial Modbus – OPC UA wrapper

The OPC UA server being ready for subscriptions, configured with folder and node setup, authentication, security mode and policy, etc., the wrapper solution was tested first in laboratory and afterwards on integrating a real WWPS. Key factors that may cause communication issues with the OPC UA client include certificate generation and exchange, hostname configuration, and proper date–time synchronization.

The laboratory setup was based on an Arduino Uno CPU, implementing a simple process with three LEDs controlled by internal variables and functions tracking operating hours and start counts. These variables, including LED and switch states, were integrated into a local serial Modbus slave structure. The Modbus protocol was tested over both RS232 and RS485 physical layers. As depicted in Fig. 2.1-8, where Modbus RTU was used, pins 0 and 1 on the Arduino board were used for serial wiring. An Ethernet link was established between the IoT-2040 and the RTU-240 router, which was configured to simulate real conditions while supporting both 4G/LTE and WiFi communications for testing.

The Modbus client was developed in Node-RED, and an OPC UA server was defined to host the taken-over Modbus variables. Additional functions were tested such as variable population scheduling, automatic redeployment on failure, protection structures, communication and server status checks, and router access control. For the laboratory setup, the OPC UA client was also implemented in Node-RED, first deployed on a Windows 10 PC and later on a Samsung Android 7 device.

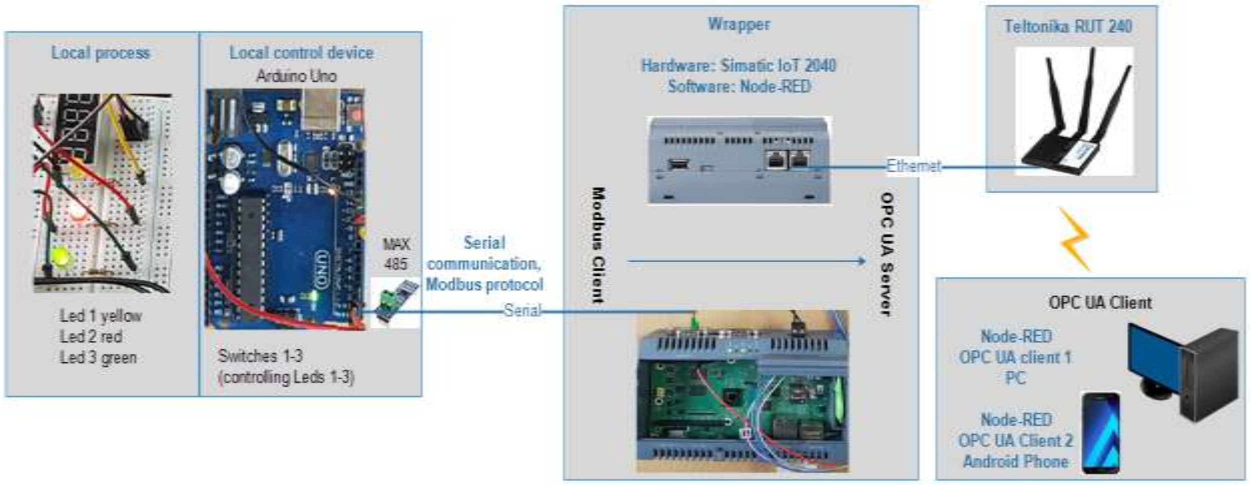
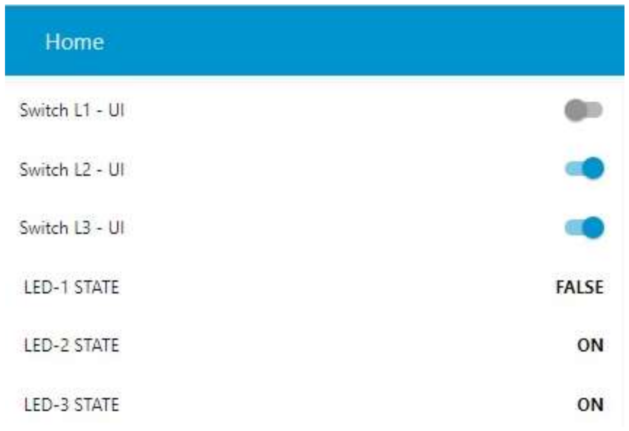


Fig. 2.1-8 Schematic view of the implemented Modbus-OPC UA wrapper for the Arduino laboratory test application

A simple dashboard GUI was created to command and monitor the three LEDs. The status from Fig. 2.1-9 illustrates the OPC UA client GUI, where activating the second and third switches triggers LED 2 and LED 3 (the state of the real process is visible in Fig. 2.1-8).


Home

Switch L1 - UI	off
Switch L2 - UI	on
Switch L3 - UI	on
LED-1 STATE	FALSE
LED-2 STATE	ON
LED-3 STATE	ON

Fig. 2.1-9 Node-RED Dashboard in OPC UA Client application for Windows (PC) and Android (Phone)

The real process consists of a WWPS with four pumps, cascaded with other stations to ensure wastewater transport across the local sewage network toward the treatment plant. Local automation is minimal, with pump operation based on level switch feedback. Each pump provides state and fault signals, complemented by a general anti-burglary signal. Electrical parameters such as voltages, currents, power, and total energy, are measured locally. The first control level is managed by a Wilo controller, while the station PLC is a Wago 750-816. The local automation panel is illustrated in Fig. 2.1-10, and Fig.

2.1-11 presents the IoT-2040 together with the Teltonika RUT240 router inside the WWPS in connection with the Wago PLC.

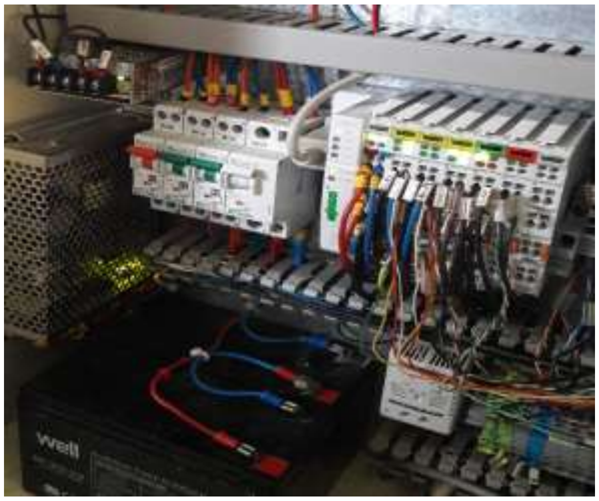


Fig. 2.1-10 A view inside the local automation panel

Fig. 2.1-11 The wrapper inside the WWPS

The architecture of the solution in the real scenario is shown in Fig. 2.1-12. The wrapper integrates into the functional system non-invasively, with the IoT-2040 connected to the Wago PLC via RS-232 using the Modbus protocol.

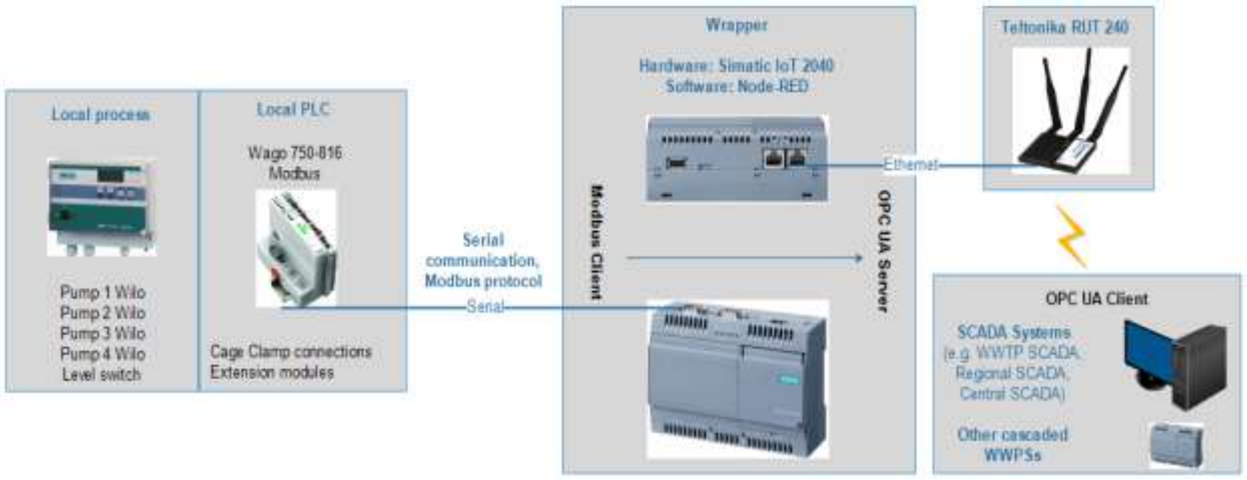


Fig. 2.1-12 Schematic view of the implemented Modbus-OPC UA wrapper inside the WWPS automation

The results obtained with an UA client from Softing company are illustrated in Fig. 2.1-13, augmented with English explanations, the primary digital tags are browsed within the local OPC UA server address space.

The Softing OPC UA client offers extensive configuration options (e.g. response times, session naming) and greater flexibility in handling inconsistencies such as hostname issues, compared to clients with fewer settings like IGSS. In IGSS, as in many SCADA systems, certain parameters are hardcoded in the interface to enhance usability and robustness.

Consequently, IGSS SCADA was employed for testing to validate solution efficiency and ensure operator accessibility through synoptic schemes, alarms, and graphics. A WWPS synoptic diagram was developed in the SCADA control room to display pump states, faults, and numerical data (e.g. operating hours, electrical parameters). All tags were tested, including faults integrated into the IGSS Alarms and Events module.

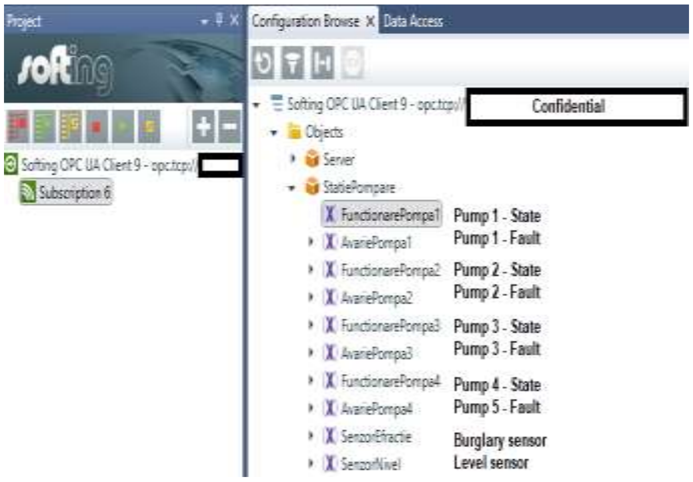

Project
Configuration Browse
Data Access
Softing OPC UA Client 9 - opc.tcp:// [redacted]
Subscription 6
Softing OPC UA Client 9 - opc.tcp://
Confidential
Objects
Server
StatiePompare
FunctionarePompa1 Pump 1 - State
AvariePompa1 Pump 1 - Fault
FunctionarePompa2 Pump 2 - State
AvariePompa2 Pump 2 - Fault
FunctionarePompa3 Pump 3 - State
AvariePompa3 Pump 3 - Fault
FunctionarePompa4 Pump 4 - State
AvariePompa4 Pump 5 - Fault
SenzorEfractie Burglary sensor
SenzorNivel Level sensor

Fig. 2.1-13 Softing OPC UA client connected to the wrapper

Two embedded graphs were placed in the diagram and mapped to digital atoms (see Fig. 2.1-14), and one independent Graph object with detailed graphical information. As pointed out in the two embedded graphs from Fig. 2.1-14, pumps 3 and 4 started four times between 9:00-13:00 on the 18th of May 2018. The first two pumps did not start in the rotation algorithm due to fault states. During testing, burglary and level overflow alarms were triggered to assess notification times at fault occurrence and resolution. Internal software faults were also induced, with the system responding as expected. When the highest-severity fault was introduced, the application automatically reinitialized, and Node-RED fully recovered the wrapper under five minutes.

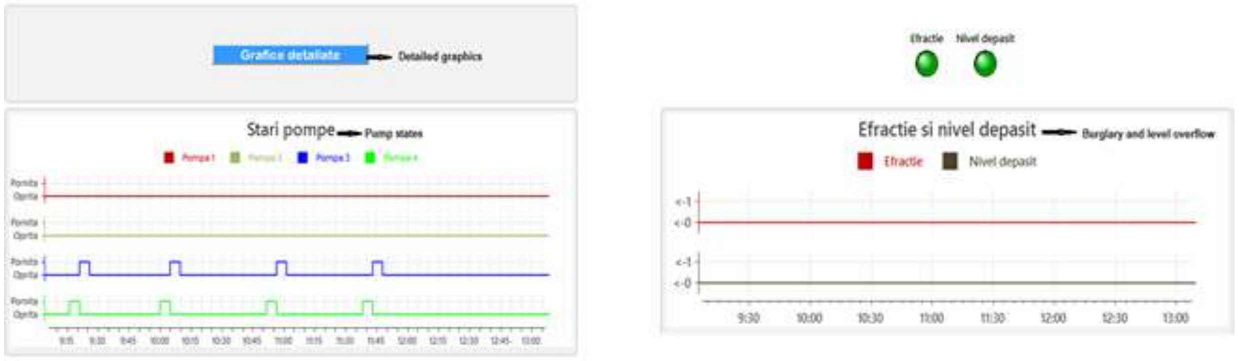


Fig. 2.1-14 Augmented screenshot from the IGSS application

2.1.2 Targeting Other Legacy Protocols and Solutions for Industry 4.0 Integration in Real Industrial Scenarios

The scenarios from 2.1.1 are covering many use cases. Other real industrial scenarios are presenting various situations that require a consistent degree of research to overcome certain obstacles, as depicted in [K-8].

2.1.2.1 M-Bus integration scenario

In the automotive manufacturing industry, the Building Management System (BMS) is an example where several approaches were necessary in order to provide Industry 4.0 integration capabilities to hardware-software solutions/equipment. A first example is referring to gas and water meters that are preconfigured with M-Bus interface and functioning in a stand-alone manner in the plant facilities. To integrate on a protocol level M-Bus based devices, few choices were available and almost all required supplementary industrial processing units within automation panels. The research approached Node-RED, as the IIoT solution for protocol integration, wrapping and conversion. But, in order to cope with the physical communication support provided by the devices, a conversion unit was adopted (IZAR Center) that was able to physically centralize and provide the M-Bus protocol under Ethernet support. Then the integration of the M-Bus data was developed within Node-RED, to be available to other levels on various ways, using OPC UA, database and visualization tools. The architecture of the solution is depicted in Fig. 2.1-15

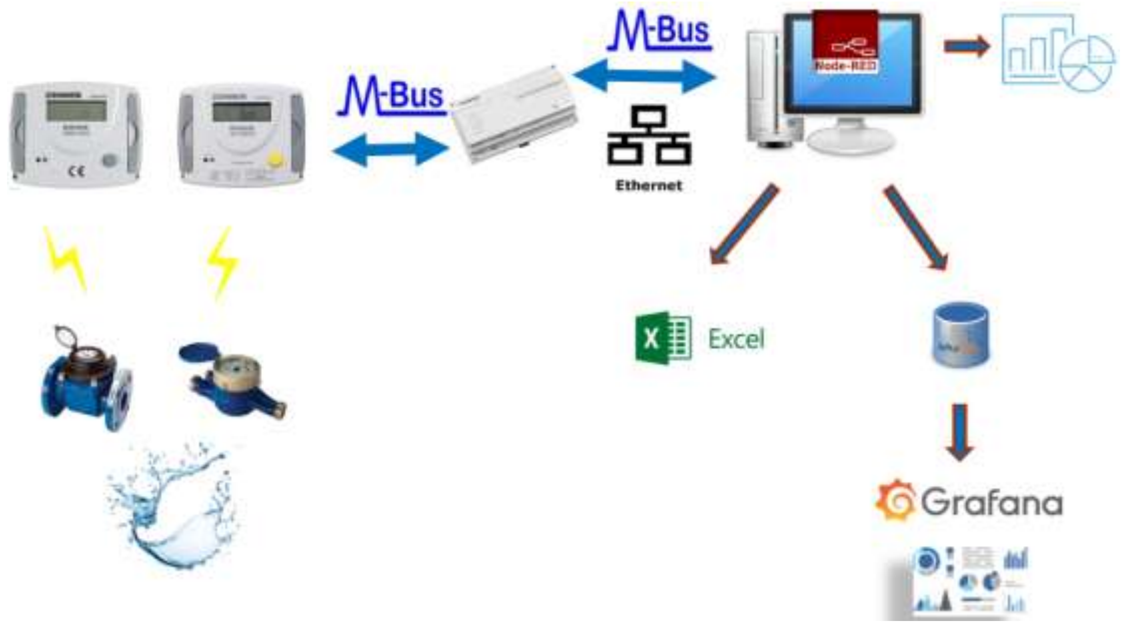


Fig. 2.1-15 M-Bus integration architecture with Node-RED

2.1.2.2 OPC DA integration scenario

The Modbus or S7 integration generally followed the principles from 2.1.1, but situations occurred where legacy supervision was the end-of-line in the integration. When it is about PLC integration then legacy SCADA has to be approached for further interoperability, but when measuring elements are supervised in proprietary environments then the difficulty is further increased. As many times legacy SCADA provides at least an OPC DA legacy server, proprietary environments are usually closed systems. The case of electrical parameters monitoring using PAC devices is a common practice. The PAC devices, depending on the version, have a native Modbus protocol implemented, either TCP or serial. The issue with the serial Modbus is that no multi-master possibility exists. Siemens proposes the Sentron Power Manager product for monitoring the electrical parameters that are measured with PAC devices. The product evolved initially towards Industry 4.0 openness but then they changed their perspective. Therefore, the Power Manager 3.5 could activate its own OPC DA server that exposed the parameters within the address space, off course being dependent of Windows operating system. But, Power Manager 4.2 application encountered in practice could not expose any protocol for further integration and the only access to maintain the functioning product was a rudimentary SQL database access.

As shown in Fig. 2.1-16, an OPC DA – UA wrapper was proposed in order to elevate the interface for Power Manager 3.5 and to extend the lifetime of the purchased and functioning software. The chosen environment was Ignition, using its gateway with licensed OPC DA interfacing. The monitoring graphical environment from Ignition was used only when implementing the protocol conversion and wrapping project that was applied to the functioning gateway. This way, a low-cost, high availability, and platform independent wrapper was obtained that exposes the data in OPC UA protocol.

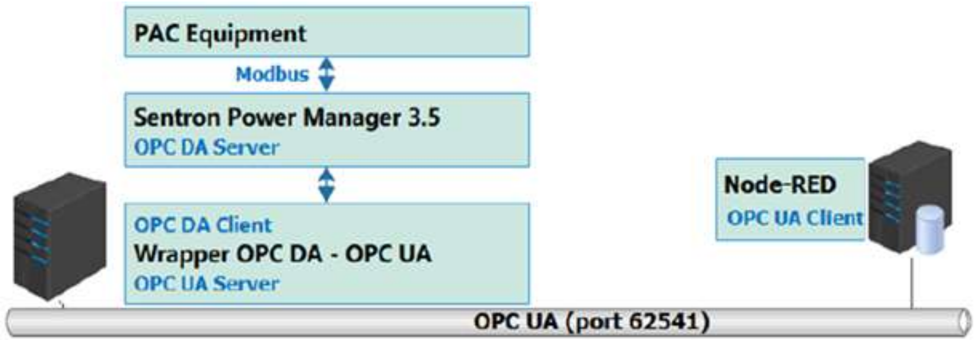


Fig. 2.1-16 OPC DA integration using OPC DA-UA wrapper

2.1.2.3 HTTP based integration scenario

Sometimes OT level protocols are not available, but as described in [K-8] web-based solutions are encountered that host data in isolated software environments that are specific to a hardware equipment, based on the HTTP protocol. This data acquisition method segregates information rather than aligning with IoT-ready systems. To integrate legacy systems, data availability must follow the IoT paradigm through reformulated acquisition, monitoring, and control functions. Technically, data exposure to specific software is limited to the HTTP protocol, with or without an API.

For the integration of a legacy application using HTTP protocol, it is mandatory to analyze the imposed requirements and limitations of the system. The real scenario where an API was available is depicted in Fig. 2.1-17 and the system consists of the OZW 775 hardware–software platform, which communicates with plant equipment via the KNX protocol, supported by a monitoring tool for periodic infrastructure checks. The supervision tool is closed and unsuitable for continuous operation. At the lower level, a web server provides an API that creates a session at each logon rather than exposing data directly in HTTP responses. Data is organized into sensor-specific data-points, each containing individual characteristics, with API responses delivering essential device information. Endpoints handle one data-point per request, but limitations include restricted concurrent requests, insufficient documentation, and heterogeneous data requiring extensive formatting.

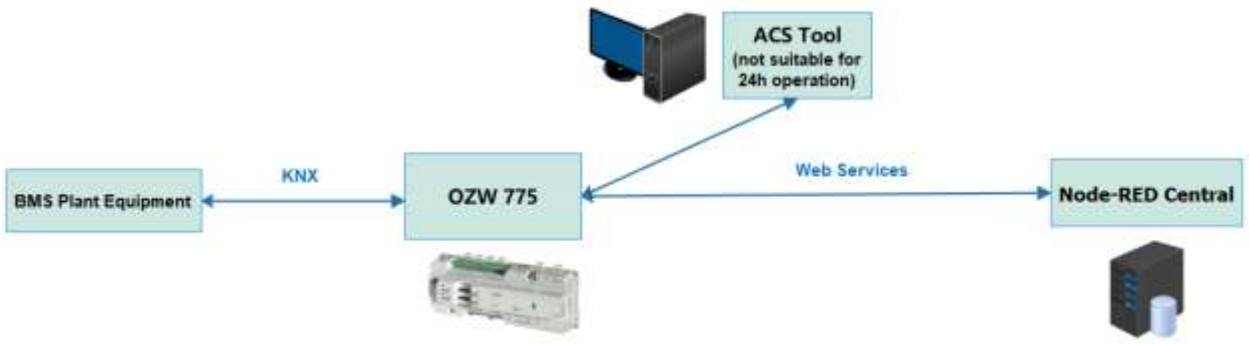


Fig. 2.1-17 Real scenario for HTTP based integration

Considering these factors, a dedicated architecture was required to manage the high volume of HTTP GET requests imposed by the legacy system. Such interoperability gaps present significant challenges for integrating legacy infrastructures. The solution was not proposed to be static, but to automate the whole callback process for the GET requests and allow seamless data processing in an efficient manner (see Fig. 2.1-18), as opposed to creating a request individually per each data-point. The dynamic attribution of configuration parameters for the HTTP request node proved to be more beneficial overall when compared to a classic approach, due to the high volume of repetitive requests.

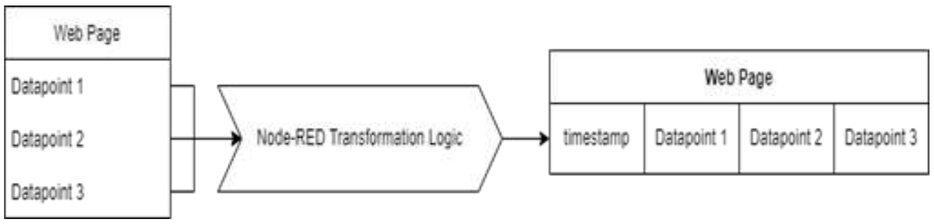


Fig. 2.1-18 Data manipulation logic

When HTTP integration is required without an API endpoint, the approach becomes rudimentary, demanding deeper investigation into alternative data acquisition methods. Tests on an Ingersoll Rand VX web server (lacking integrated API support) revealed reliance on outdated acquisition techniques that obscure software design and provide little usable information. DevTools proved essential by mapping network requests to backend calls. Initial attempts to access the server via direct hyperlinks returned HTML responses, which were unsuitable for constructing reliable logic or ensuring data quality, as their relevance for extraction was limited. Behind the web server, a certain request URL was discovered that was able to satisfy the conditions of data integrity and quality. Each page call returns in this situation a .XML response. This type of response, different from the expected response type, widely-utilized JSON, was adequate in obtaining the entailed data. The data acquisition in the exposed industrial scenario is depicted in Fig. 2.1-19.

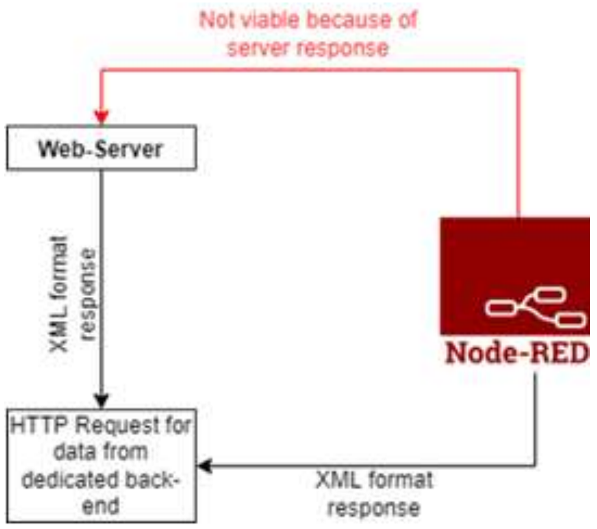

Not viable because of server response
Web-Server
XML format response
Node-RED
HTTP Request for data from dedicated back-end
XML format response

Fig. 2.1-19 Data acquisition for the case study presented

2.1.2.4 Event-based approach for data acquisition without transmission protocol

There are situations where even on the OT level, solutions are deployed as segregated and considered complete. This means that edge devices are taken over into a small proprietary software application on a separate PC that assures a rudimentary data monitoring and storage. Between the edge devices and the centralizing application it may be an industrial protocol, but over that there is no protocol for future integration. In such cases, two options arise: to get data directly from the edge devices, or to approach the centralizing application. Usually the first option mean to erase completely the centralizing application, due to the fact that usually no multi-master protocols are utilized in the corresponding communication.

The current section approaches an industrial BMS scenario where data from several temperature and humidity sensors are taken over through Modbus ASCII and Modbus RTU, and stored in an outdated manner, in text files with the specific .txt extension (see Fig. 2.1-20). This type of data storage has certain limitations and disadvantages, such as: linear searching across large levels of data due to the lack of indexing, no possible relationship between row entries, lack of relationship between file-stored data and dissimilar datatypes namely integers, floats and/or booleans. Thus, complete association with such outdated legacy mechanism comprises a dynamic approach capable of satisfying the needed requirements for data acquisition and formatting. In many real scenarios, like in the current one, the second integration option was required because the legacy structure could not be altered in any way.

Given these constraints, the implementation began by identifying a methodology to address variable file affixing times. This ruled out the classic Node-RED injection node with fixed intervals, leading instead to folder surveillance triggered by file updates, an event-driven architecture (EDA). EDA improves network efficiency by processing only changed data, though most legacy systems lack mechanisms for easy adoption. To handle this, a context-aware flow variable was introduced to detect updated files. Legacy software added complexity, as extracted data appeared only in buffer or string formats, causing datatype inaccuracies. Database insertion was designed dynamically, with timestamp formatting included, and optimized through batch operations to reduce resource consumption. This approach also enables integration of disparate tables and extension to new sensors and actuators.

2.1.2.5 OPC UA to MQTT data conversion module in Node-RED

The evolution towards Industry 4.0/5.0 determined an architectural reconsidering to adopt digital transformation strategies that can bridge the gap between the OT and IT layers. Due to the fact that according to current protocol capabilities and the general state of the art in interfacing, many conceptual approaches are defining broker based solutions to implement publish-subscribe decoupled architectures, MQTT became an important transport protocol.

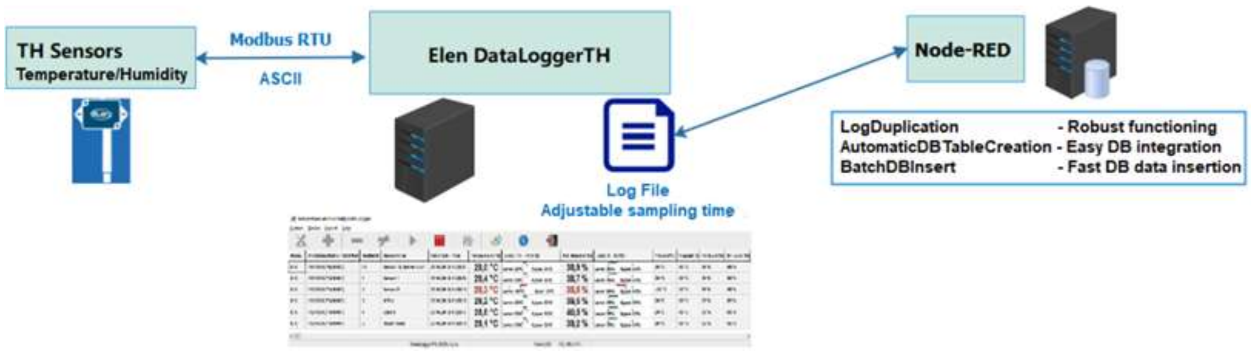

TH Sensors
Temperature/Humidity
Modbus RTU
ASCII
Elen DataLoggerTH
Log File
Adjustable sampling time
Node-RED
LogDuplication - Robust functioning
AutomaticDBTableCreation - Easy DB integration
BatchDBInsert - Fast DB data insertion

Fig. 2.1-20 Scenario for integration without protocol

The poll/response approach has disadvantages that inherently make a system slower/overwhelmed in the process of retrieving data constantly that in many cases has not changed. In digital transformation the current tools and trends in manufacturing indicate that data many times decoupled entities represent an advantage, and data should be trusted, understood and accessible at all levels. The subject will be detailed more in the following chapters, the current section presenting a Node-RED solution that allows accessing data from the OT level within OPC UA format and exposing it into MQTT topics within a broker. Obviously MQTT data has to be packed and structures, solutions will be discussed in following chapters. In the current section, as it can be observed in fig. Fig. 2.1-21, topics are initiated separately, as simple MQTT variables that can be further processed and subscribed. The address space of the OPC UA server coming from the OT level is browsed and variables are selected and put in a correct format.

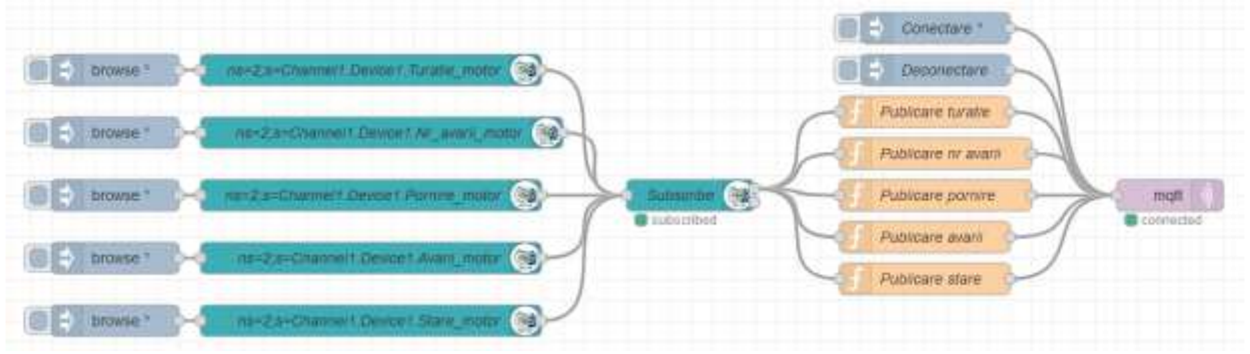

browse
ns=2;s=Channel1.Device1.Turatie_motor
browse
ns=2;s=Channel1.Device1.Nr_avarii_motor
browse
ns=2;s=Channel1.Device1.Pornire_motor
browse
ns=2;s=Channel1.Device1.Avarii_motor
browse
ns=2;s=Channel1.Device1.Stare_motor
Subscribe
subscribed
Conectare
Deconectare
Publicare turatie
Publicare nr avarii
Publicare pornire
Publicare avarii
Publicare stare
mqtt
connected

Fig. 2.1-21 OPC UA – MQTT data conversion solution

2.2 Approaching and Improving OPC UA.

OPC UA protocol is a major enabling technology and research is continuously carried out to extend and to improve its capabilities, to fulfil the growing requirements of specific industries and hierarchical levels. Consistent issues to be approached are related to the latest specifications and the real-time context that could extend the applicability of the protocol and bring significant benefits in terms of speed, data volumes, footprint, security. Section 2.2.1 synthesizes information from [K-19] that approaches first the conceptual analysis to improve the OPC UA interfacing using the Publish-Subscribe mechanism, focusing on real-time constraints and role distribution between entities. The conceptual analysis is materialized into a solution that takes OPC UA Publish-Subscribe over User Datagram Protocol (UDP) mechanism to the next level by developing a synchronization algorithm and a multithreading broker application to obtain real time responsiveness and increased efficiency by lowering the publisher and the subscriber footprint and computational effort, reducing the difficulty of sending larger volumes of data for various subscribers and the charge on the network and services in terms of polling and filtering.

Section 2.2.2 presents research from [K-18], and aims to consider higher data-volumes, approaching the multi-channel UDP-based communication, and analyzes the robustness of the developed mechanism in the context of long-term data transmission. The research extends the applicability of the OPC UA in the context of image transmission. Although highly needed, the image transmission after processing is currently beyond the reach of OPC UA or other legacy industrial protocols, being considered as a separate fraction in the industrial environment. The concept and developments are applied without special hardware constraints considering both the end-of-line industrial manufacturing process in the automotive sector and the car-to-infrastructure communication.

OPC UA has to consider the coexistence with other emerging real-time oriented protocols in the production lines. The Data Distribution Service (DDS) will be present in future architectures in areas as robots, co-bots, and compact units. Section 2.2.3, presenting the research from [K-16], proposes a solution to evaluate the real-time coexistence of OPC UA and DDS protocols, functioning in parallel and in a gateway context. The purpose is to confirm the compatibility and feasibility between the two protocols alongside a general definition of criteria and expectations from an architectural point of view, pointing out advantages and disadvantages in a neutral manner, shaping a comprehensive view of the possibilities. The solution is applied using non-ideal infrastructures to accelerate the applicability in the production lines.

2.2.1 Improving OPC UA Publish-Subscribe Mechanism over UDP with Synchronization Algorithm and Multithreading Broker Application

OPC UA, as an Industry 4.0 enabler, was initially applied at the SCADA level through software environments and centralizing servers. Its advantages soon extended to other layers, entering the PLC level via the classic client–server data acquisition component. Over time, companies demanded broader functionality, incorporating additional characteristics and services. Studies were conducted to extend application functionality down to the field-device level [27-28], or to approach the cloud integration using OPC UA [29].

New OPC UA specifications introduced the publish–subscribe mechanism, enabling real-time communication and higher data volumes [30]. Its applicability in factory automation has been evaluated [31], while further research explored integration [32] using the open62541 SDK [33], a widely maintained tool in academic and industrial contexts. Several issues remain regarding real-time performance and architectural applicability, requiring detailed analysis. Parallel studies near industrial deployment have shifted focus toward lower-level protocols such as MQTT, in relation to higher-level OPC UA [34] and Sparkplug B [35], though without implementing publish–subscribe according to the new OPC UA specifications.

The OPC UA publish–subscribe mechanism is relatively recent compared to other communication protocols widely applied in real-time systems. While diverse use cases and implementation strategies exist, its envisioned improvements were designed to incorporate established practices from industrial protocols already used in real-time applications. Two protocols that were considered in [K-19] as more mature in this direction were Some/IP and DDS. In this context, the corresponding mechanisms were studied for orienting OPC UA improvements towards previously established common good practices.

As described in [36], SOME/IP notifications inform subscribers of value changes or event occurrences, while also allowing them to request updates or verify variable status through designated methods. The standard separates responsibilities between the SOME/IP instance, which transports changed values, and the service discovery component, which manages subscription and publishing. Notifications can be cyclic, on-change, conditional, and messages include the serialized payload length, useful for end-to-end verification and filtering. At protocol level, SOME/IP employs both UDP and TCP to address congestion, message loss, bit errors, and other transmission faults.

Integrating a similar notification module into OPC UA Publish–Subscribe would reduce constant polling and improve efficiency by delivering only relevant information. Moreover, OPC UA could enhance modularity and ensure full decoupling between publishers and subscribers, as well as between system responsibilities such as information delivery, notification, security, time management, and reception.

Data Distribution Service (DDS) is applied across domains such as aeronautics, healthcare, and power industries [37, 38]. Designed for real-time control, DDS enables fault-tolerant exchanges with low latency and advanced filtering. It ensures modularity through a decoupled publish–subscribe design and supports time-based operations with multiple synchronization strategies. Integration with TSN technology enhances time determinism [39]. Several DDS practices are considered valuable for OPC UA, particularly in strengthening real-time network capabilities.

The objectives were to analyze OPC UA interfacing through the publish–subscribe paradigm, addressing real-time constraints and role distribution, while considering current developments and strategies from the automotive sector, and also to design and implement an OPC UA pub–sub solution over UDP, centered on a synchronization algorithm and a multithreading broker to achieve real-time responsiveness, higher efficiency, and extended QoS. The approach aims to minimize subscriber–publisher coupling, facilitate high-speed transmission of large data volumes, and reduce network load from polling and filtering. Availability and safety guide the design, with emphasis on fault detection, tolerance, and recovery.

2.2.1.1 The Publish-Subscribe Mechanism: Design and Architecture

In the publish–subscribe paradigm, publishers continuously disseminate information or events, while subscribers are notified upon changes. Beyond simple data exchange, responsibilities should be distributed across multiple nodes. The emphasis shifts from node-to-node links to the relationship between information and its targets, enabling efficient distribution regardless of subscriber count. As suggested in [40], middleware mechanisms can assume roles independent of payload context, decoupling publishers and subscribers. This reduces computational load, improves network efficiency, and ensures predictable message distribution. However, reliance on single event servers introduces risks in large-scale systems, as failures may cause data loss or downtime. For example, an OPC UA factory using only one publisher server for cloud interfacing is vulnerable without backup mechanisms. To mitigate such risks, responsibilities must be distributed through middleware entities that notify subscribers and manage publisher–subscriber interactions. Safety measures for fault detection, tolerance, and recovery are essential to preserve publishers/subscribers independence.

In [30], publisher and subscriber roles are described as loosely coupled, with information exchange independent of subscriber count. Their relation relies on shared understanding of *DataSets* and publishing details. Message-oriented middleware functions as a multicast address for UDP or as a broker for MQTT/AMQP. With UDP transport, publishers send data to the multicast address, while subscribers filter messages using *DataSetMetaData*, which must be transmitted beforehand. This one-to-many strategy increases complexity and may challenge time constraints. When encryption is added, a Security Key Server manages key distribution. In large systems, subscribers must decrypt and filter all incoming messages, creating computational overhead and hindering real-time performance. Dynamic publishing with frequent *DataSetMetaData* updates further risks network overload. Middleware, acting as an intermediate entity, should therefore manage publisher–subscriber relations and, if needed, distribute *DataSetMetaData* to reduce complexity and improve efficiency (see Fig. 2.2-1).

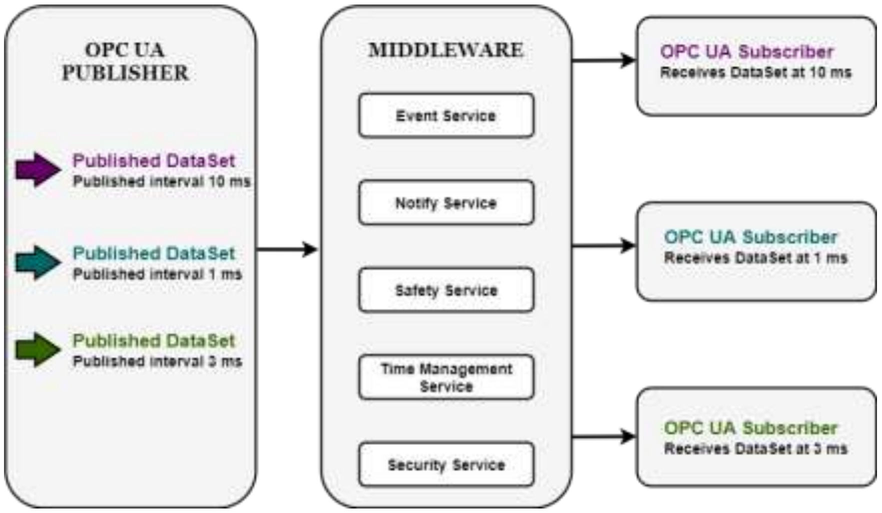


Fig. 2.2-1 Proposed OPC UA Middleware Design with Services, in real time scenarios.

Safety measures must be continuously applied, with the entity managing publisher–subscriber relations verifying the capabilities of all involved nodes. For real-time operation, guarantees on delivery, decryption, and filtering times require that these relations be persistently stored. In OPC UA publish–subscribe, such functionality could be achieved through a *PubSub Directory*.

In [40], an Event service is proposed as the entity managing publisher–subscriber relations. Publishers share information or event types, while the Event service informs them of interested subscribers, enabling direct notification. However, this reduces decoupling between publishers and subscribers. As noted, a many-to-many approach should be considered for designing robust and efficient system architectures.

In OPC UA, subscriber-side filtering involves multiple steps before payload access, often adding computational overhead and delays that hinder real-time performance. When middleware is broker-based, publishers connect to MQTT or AMQP entities, achieving decoupling but shifting communication to lower OSI layers. In this design, subscribers are MQTT/AMQP applications that bypass OPC UA's higher-layer mechanisms, meaning the publish–subscribe model is only partially implemented. Such broker architectures are mainly suited for cloud integration and heterogeneous environments, but real-time guarantees become difficult. Ensuring fixed transmission cycles, fault detection, and message loss recovery is complex. Security is limited to transport-level protection, as end-to-end OPC UA security cannot be maintained between publishers and non-OPC UA subscribers.

In [41], a publish–subscribe mechanism using ROS middleware is described, where subscriber identity and number are abstracted and managed by the middleware. This design allows publishers and subscribers to be replaced in real time. In the automotive domain, SOME/IP employs publish–subscribe for event exchange. Subscribers register to event groups via Service Discovery, which announces service availability and controls event message sequencing, ensuring only required messages are received. Similar discovery approaches exist in other protocols: DDS, for instance, discovers topics independently of applications, enabling entities to focus on data reception rather than search. This enhances decoupling and supports safety mechanisms such as lost-message detection.

2.2.1.2 *Time Synchronization in the Context of OPC UA and TSN Technology*

In the context of TSN and OPC UA publish–subscribe, IIoT is increasingly focused on real-time functionality from network to application level. OPC UA aims to support large-scale, real-time information exchange with seamless integration into existing architectures, while TSN provides data-link layer solutions for clock synchronization and time-accurate message delivery. Both evolve incrementally toward real-time requirements. For OPC UA, achieving hard real-time synchronization requires TSN, particularly its grandmaster clock standard, though current specifications may not fully meet large-scale demands.

Monitoring synchronization between publishers and subscribers is critical in complex infrastructures. Without dedicated interfaces, managing diverse time bases becomes overwhelming. Services for time management and safety against desynchronization can enhance scalability and adoption in IIoT. Additional requirements include notifications for elapsed cycles to avoid waiting states and unnecessary polling. While OPC UA allows predefined publishing intervals, current mechanisms lack dedicated services to share them, forcing subscribers to poll UDP messages. Future improvements could introduce notify services and time-triggered transmissions, reducing computational effort and enabling resynchronization. To ensure efficiency, TSN time guarantees must be clearly aligned with OPC UA layers, synchronizing operations within publishers and subscribers on a common time base (see Fig. 2.2-2). In use cases with strict time constraints, a defined time reference must be accessible. Details such as Ethernet hardware clocks or OS-specific timers can support synchronization both internally and across entities. Consequently, standardized mechanisms are required to integrate time-deterministic technologies.

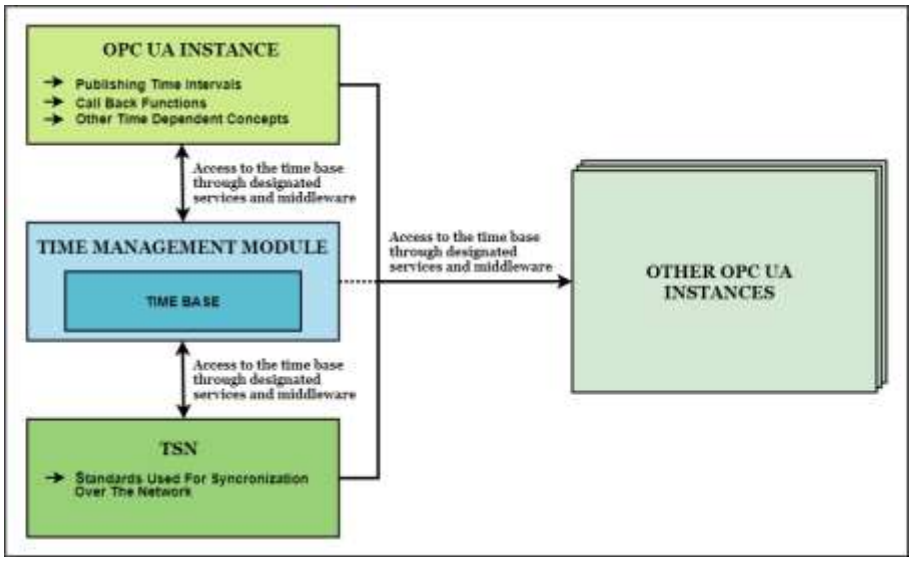

OPC UA INSTANCE
Publishing Time Intervals
Call Back Functions
Other Time Dependent Concepts
Access to the time base through designated services and middleware
TIME MANAGEMENT MODULE
TIME BASE
Access to the time base through designated services and middleware
OTHER OPC UA INSTANCES
Access to the time base through designated services and middleware
TSN
Standards Used For Syncronization Over The Network

Fig. 2.2-2 Proposed design with all entities relying to a common time base.

Both internal and external synchronization strategies must be further standardized through dedicated services and middleware, as seen in other deterministic technologies. To improve QoS in time-based operations, the notion of time should be integrated at higher OSI levels within OPC UA. Future modules must manage and share access to a common time base, enabling synchronization across applications with different clocks.

The case study presents a broker application transmitting data at fixed intervals, with subscribers synchronized to read messages at correct moment. A synchronization algorithm was developed to ensure timely reception, minimize polling, and filter only the relevant messages for each subscriber.

2.2.1.3 *Case Study*

The case study introduces a multithreading broker application within the OPC UA publish–subscribe paradigm, using UDP as the transport protocol and emphasizing real-time constraints and role distribution. With multiple entities involved, the design assigns each to a separate device operating on a Linux-based OS (see Fig. 2.2-3).

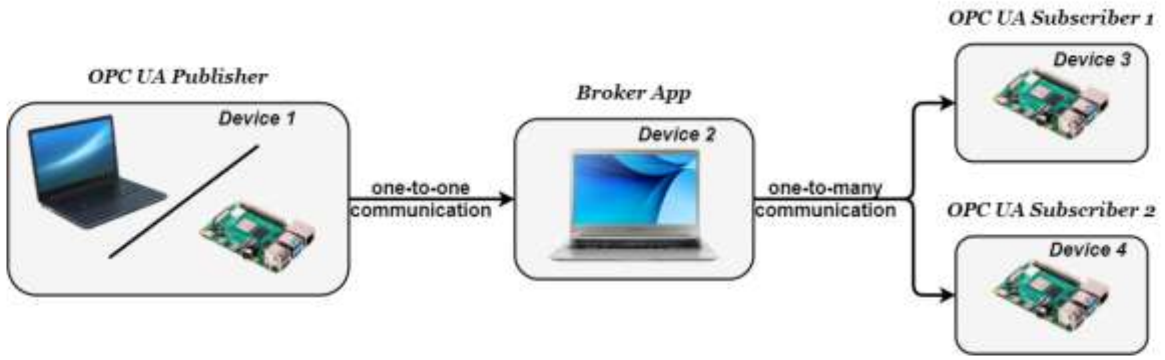


Fig. 2.2-3 General Architecture of the Case Study.

The publisher is responsible for delivering data without knowledge of the final receivers. It must first be configured and initialize the publish–subscribe connection via the *UA_Server_addPubSubConnection* method. In this case, the publisher transmits information to a single entity, the broker application. The next step is creating a *WriterGroup*, which defines parameters for network messages. For real-time constraints, the key parameter is the publishing interval in *UA_WriterGroupConfig*. The payload, represented as a hexadecimal number (e.g., 0xDC), is encapsulated in a *DataSetMessage*, independent of how data is split among subscribers. For example, each two bits may correspond to sensor values collected by a field device. While the publishing interval is set according to publisher needs, it should align with consumer requirements in practice. Finally, the publisher connects directly to the broker using its IP and port, without requiring a multicast address.

The broker application enhances efficiency by filtering and routing relevant information while synchronizing entities. It comprises two components: one receives data from the OPC UA publisher, and the other republishes extracted data at defined intervals to subscribers. Most filtering occurs at this stage of exchange. Since receiving and transmitting involve distinct time constraints and tasks, a multithreading design was adopted to improve performance and synchronization. Although the SDK [33] lacks native multithreading support, the broker's high-level architecture separates its two components into independent threads (see Fig. 2.2-4).

Future subscribers are expected to provide the broker with timing details, targeted information, and unique identifiers through a one-shot transmission based on the classic client–server paradigm. Configuration may occur via a JSON file at runtime or through periodic communication with a dedicated server holding consumer details. In the broker implementation, these parameters, timing, information targets, IDs, are hard-coded for the use case.

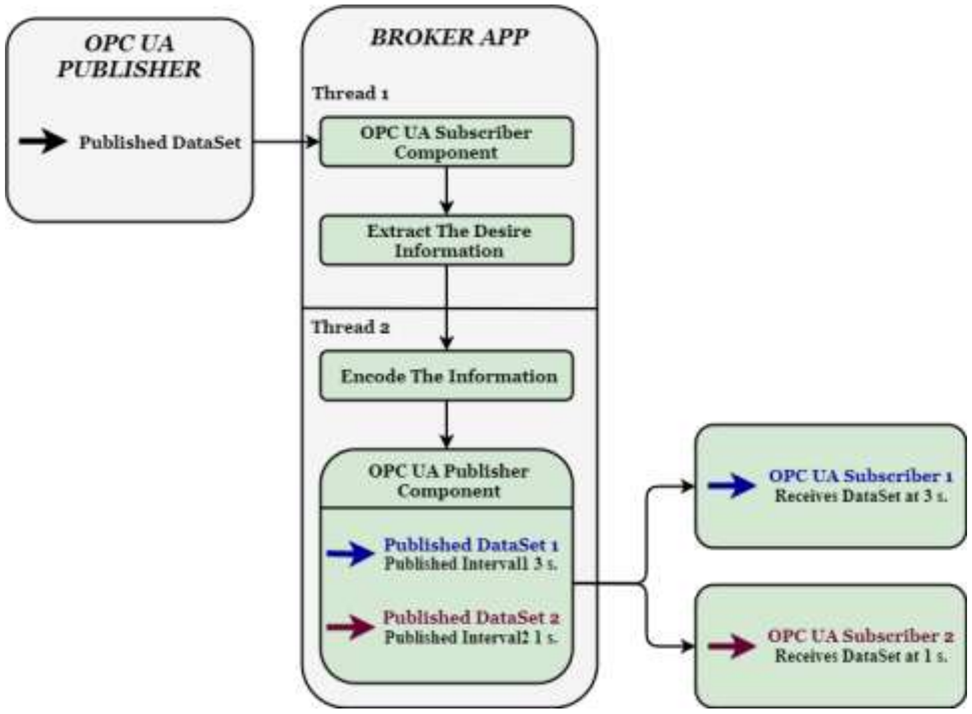


Fig. 2.2-4 Architecture and Interaction of the Broker Application.

The first thread manages the subscriber component, responsible for extracting targeted data according to subscriber preferences. Execution begins by initializing a *PubSubConnection* with the OPC UA publisher over a unicast address. Once connected, the component listens for relevant network messages. Subscription is implemented classically, interrogating incoming traffic with minimal recurrence to avoid loss of *DataSetMessages*, while parsing message fields and filtering data types.

The second thread manages the publisher component. Depending on subscriber count and required time intervals, it initializes an OPC UA publisher instance with multiple *WriterGroups*, each configured for specific publishing rates. Extracted data is buffered before transmission, enabling historical access and improving availability. Prior to assignment to a *DataSet*, an encoding step adds a unique subscriber ID. In the case study, IDs are defined as hexadecimal values (0xF for subscriber1, 0xA for subscriber2). The encoding shifts extracted data by four bits and inserts the ID into the least significant bits. Thus, the payload 0xDC becomes 0xCF for subscriber1 and

0xDA for subscriber2. Following encoding, the payloads are encapsulated in *DataSetMessages* and published at distinct intervals by the *WriterGroups*. Transmission occurs via a multicast address, enabling one-to-many communication and shared access among subscribers, as described in [30].

In case study 2, two subscriber entities were implemented to consume payloads from the OPC UA publisher, each with distinct requirements for targeted data and real-time behavior. Subscriber1 expected updates every second, while Subscriber2 required data every three seconds; the design also proved functional with intervals below 10 ms. The broker's main purpose was to prevent continuous network interrogation and reduce filtering by leveraging the publisher's predefined intervals. On the subscriber side, message reception was executed only at the expected time intervals. Additional mechanisms were introduced to handle potential desynchronization.

In some scenarios, proper timing configuration for both broker and subscribers is insufficient. Real-time synchronization becomes essential when no common time base exists and time references are not continuously exchanged. The challenge lies in dynamically aligning transmission and reception intervals across separate entities. Polling describes subscriber desynchronization from the broker and occurs in two cases: when the subscriber receives data classified as invalid (e.g., malformed or improperly executed network messages), respectively when the subscriber receives a valid message, but of a different type than expected under OPC UA protocol. In both cases, the arrival time of the desired message is unpredictable, requiring repeated execution of the receive function. Such polling states can only be avoided through synchronization between the sending and receiving moments.

From the subscriber's abstract perspective, two scenarios emerge based on message validity and recurrence. Messages encoded with the subscriber's ID are classified as valid, while all others are considered invalid regardless of subscriber count. For example, recurrence intervals were set at 100 ms for valid messages and 1000 ms for invalid ones, and vice versa. (see Fig. 2.2-5).

Scenario 1 consists of when an invalid message or polling state is detected, and consequently the delay is set to zero, allowing immediate receive operations until a synchronization event occurs (e.g. reception of a valid message). Once synchronized, the delay returns to the subscriber's predefined interval, ensuring efficient operation without unnecessary polling. Stable networks maintain synchronization, with desynchronization possible only when subscriber delivery intervals overlap.

Scenario 2 mirrors Scenario 1, but intersections between broker delivery intervals occur at each cycle. Dynamic delay adjustment during invalid messages resynchronizes broker and subscriber, maintaining recurrence and avoiding message loss.

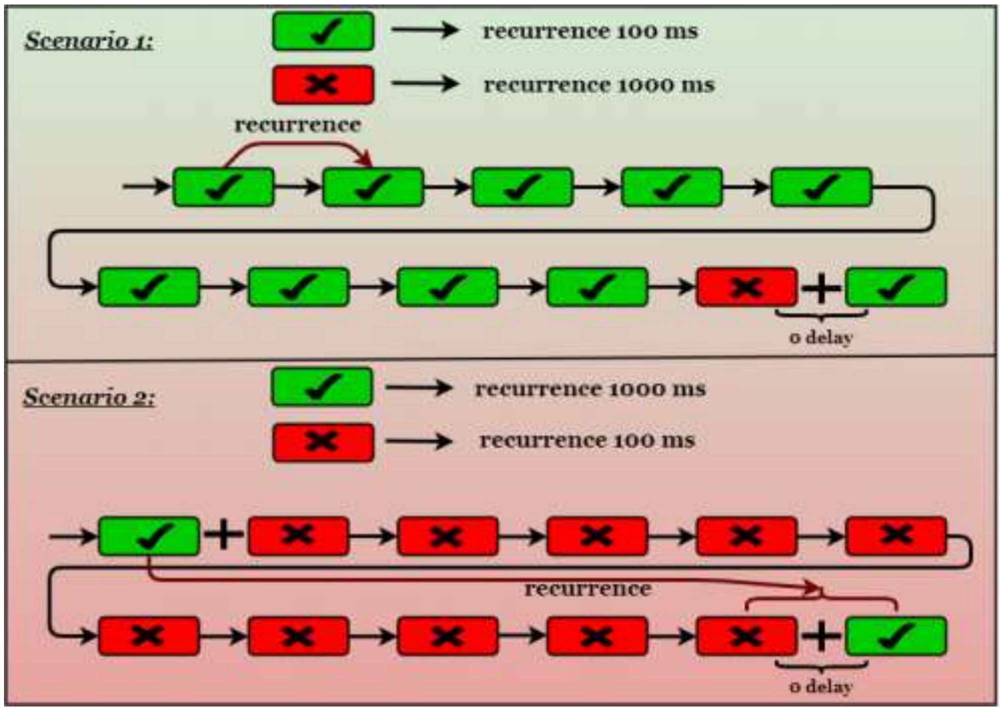


Fig. 2.2-5 The 2 scenarios from the Subscriber Perspective regarding recurrence and validity of the messages.

In both scenarios, the synchronization algorithm proved efficient, with testing confirming correct behavior in broker and subscriber operations. For universal UDP broker solutions in OPC UA, encoding of desired information is critical to synchronization. If the broker shuts down, subscribers enter polling until a valid message is received, then re-enters into normal pub-sub operation.

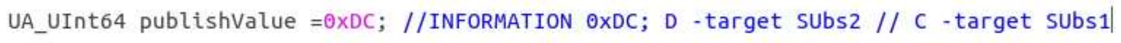
2.2.1.4 Results

During the case study, the open62541 SDK lacked a finalized Subscriber API. Considering the receiving solution compliant with [30], lower software layers may still perform network interrogations or buffer multicast traffic independently of the application layer. To address this, a polling state was defined at the application level, with the goal of ensuring subscribers process only relevant messages. This improves upon the receive concept described in [30], where subscribers must handle irrelevant or unknown messages, and aligns with the implementation in [33].

After implementation, several results were observed:

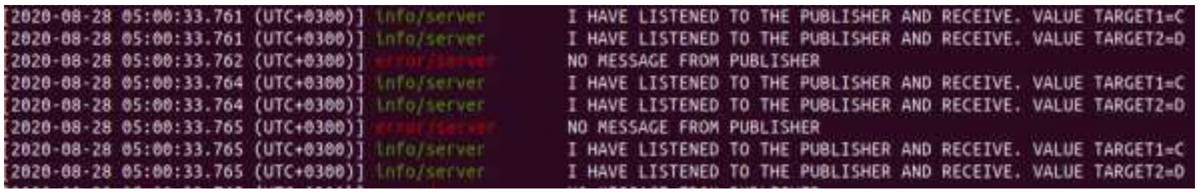
- Publisher abstraction – A single published *DataSet* transmitted information to two subscribers with different preferences and timing, simplifying configuration and enabling efficient large-scale data transfer.
- Multithreaded broker – Independent components improved speed and responsiveness.
- Data buffering – The broker stored data in transit, providing an essential service often missing in OPC UA.
- Backup functionality – The broker could republish stored data if the publisher failed, assuming publishing roles and enabling safety measures to notify consumers of malfunctions.
- Stable delivery – Encoding and synchronization ensured data was provided at consistent intervals, avoiding unnecessary transmission rates.
- Subscriber decoupling – Subscribers remained unaware of publisher details, reinforcing the loosely coupled design described in [30].
- Synchronization algorithm efficiency – Polling and excessive filtering were minimized, reducing computational effort and resource usage.
- Real-time assurance – Synchronization guaranteed timely data delivery, with resync options supporting controller-to-controller scenarios.
- OPC UA consistency.

Some results are depicted in Fig. 2.2-6 - Fig. 2.2-9, the Publisher sending and the Broker filtering and transmitting forward, followed by each Subscriber entering in stable state after a synchronizing procedure.



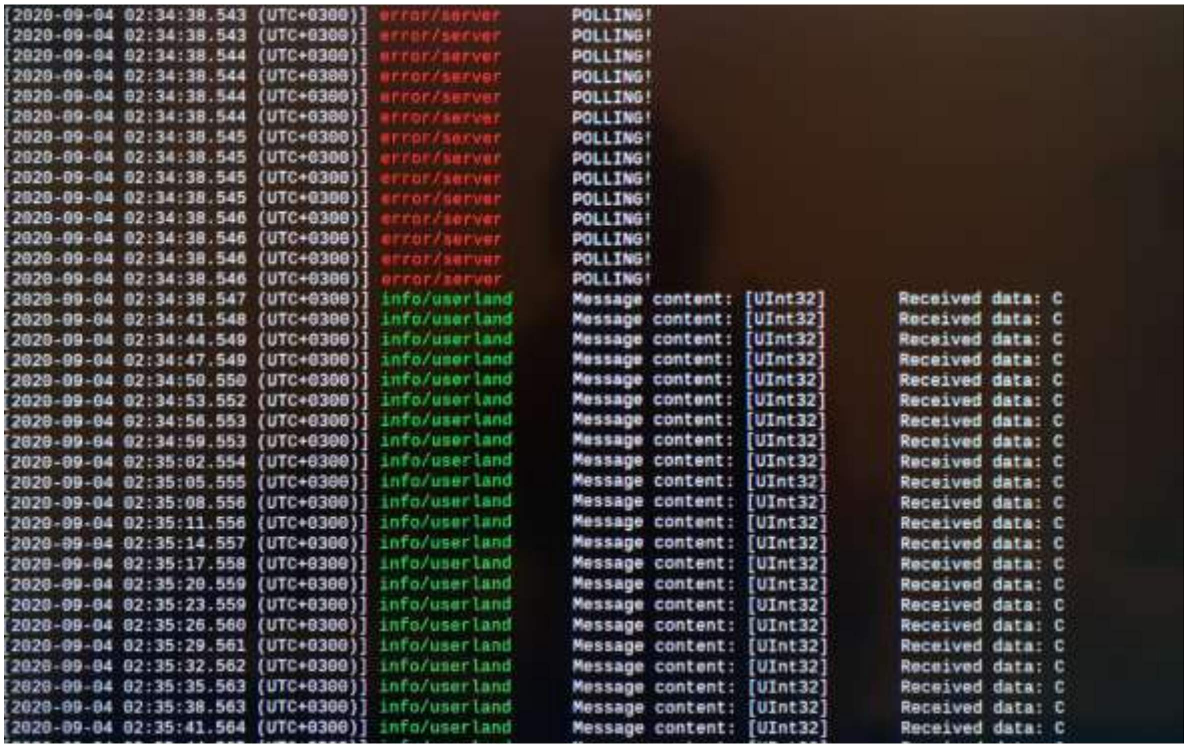
```
UA_UInt64 publishValue =0xDC; //INFORMATION 0xDC; D -target SUbs2 // C -target SUbs1
```

Fig. 2.2-6 The data sent to the broker from the terminal of the Publisher.



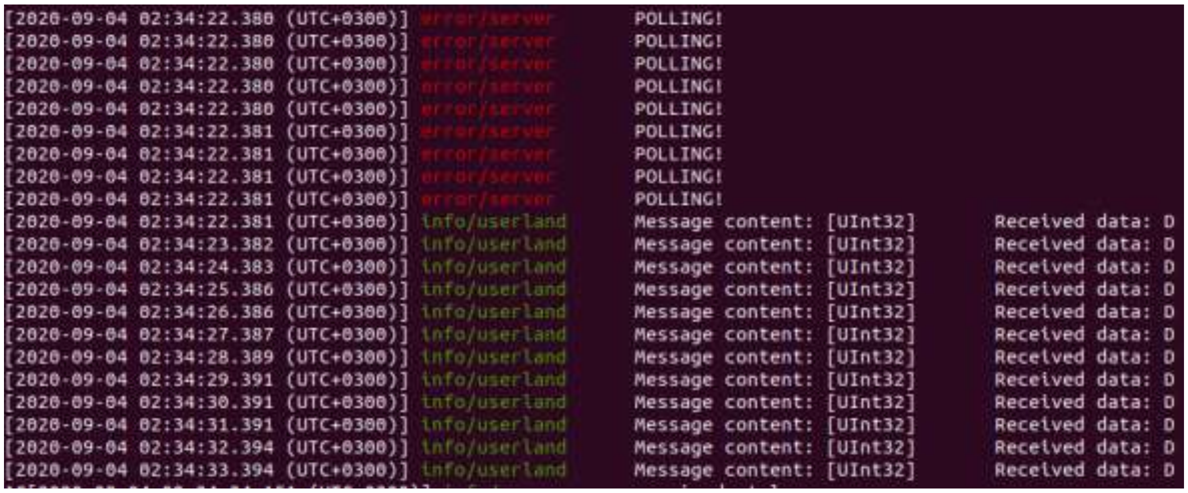
```
[2020-08-28 05:00:33.761 (UTC+0300)] info/server       I HAVE LISTENED TO THE PUBLISHER AND RECEIVE. VALUE TARGET1=C
[2020-08-28 05:00:33.761 (UTC+0300)] info/server       I HAVE LISTENED TO THE PUBLISHER AND RECEIVE. VALUE TARGET2=D
[2020-08-28 05:00:33.762 (UTC+0300)] error/server      NO MESSAGE FROM PUBLISHER
[2020-08-28 05:00:33.764 (UTC+0300)] info/server       I HAVE LISTENED TO THE PUBLISHER AND RECEIVE. VALUE TARGET1=C
[2020-08-28 05:00:33.764 (UTC+0300)] info/server       I HAVE LISTENED TO THE PUBLISHER AND RECEIVE. VALUE TARGET2=D
[2020-08-28 05:00:33.765 (UTC+0300)] error/server      NO MESSAGE FROM PUBLISHER
[2020-08-28 05:00:33.765 (UTC+0300)] info/server       I HAVE LISTENED TO THE PUBLISHER AND RECEIVE. VALUE TARGET1=C
[2020-08-28 05:00:33.765 (UTC+0300)] info/server       I HAVE LISTENED TO THE PUBLISHER AND RECEIVE. VALUE TARGET2=D
```

Fig. 2.2-7 Terminal of the Broker App receiving data from the Publisher and transmitting it to the subscribers.



```
[2020-09-04 02:34:38.543 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.543 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.544 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.544 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.544 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.544 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.545 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.545 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.545 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.545 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.546 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.546 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.546 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.546 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:38.547 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:34:41.548 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:34:44.549 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:34:47.549 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:34:50.550 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:34:53.552 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:34:56.553 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:34:59.553 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:02.554 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:05.555 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:08.556 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:11.556 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:14.557 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:17.558 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:20.559 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:23.559 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:26.560 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:29.561 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:32.562 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:35.563 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:38.563 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
[2020-09-04 02:35:41.564 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: C
```

Fig. 2.2-8 Terminal of Subscriber1 receiving the desired data at time intervals of 3 second.



```
[2020-09-04 02:34:22.380 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:22.380 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:22.380 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:22.380 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:22.380 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:22.381 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:22.381 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:22.381 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:22.381 (UTC+0300)] error/server      POLLING!
[2020-09-04 02:34:22.381 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: D
[2020-09-04 02:34:23.382 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: D
[2020-09-04 02:34:24.383 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: D
[2020-09-04 02:34:25.386 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: D
[2020-09-04 02:34:26.386 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: D
[2020-09-04 02:34:27.387 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: D
[2020-09-04 02:34:28.389 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: D
[2020-09-04 02:34:29.391 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: D
[2020-09-04 02:34:30.391 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: D
[2020-09-04 02:34:31.391 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: D
[2020-09-04 02:34:32.394 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: D
[2020-09-04 02:34:33.394 (UTC+0300)] info/userland     Message content: [UInt32]       Received data: D
```

Fig. 2.2-9 Terminal of Subscriber2 receiving the desired data at time intervals of 1 second.

An analysis of the results, highlighting advantages and disadvantages from a development perspective, is presented in Table 2-1.

Table 2-1 Case study results analysis

Entity	Advantages	Disadvantages	Achievements
OPC UA Publishers	-moderate difficulty in implementation -easy configuration for different subscribers		-easy way of sending larger amounts of data for multiple subscribers with different expectation

	with different expectation -totally decoupled from the consumers of the information		
Broker App	-multithreading capabilities -real time capabilities	-high complexity in implementation -an initial first step is needed for obtaining subscribers preferences and IDs (hard-coded information in the current implementation)	-real time behaviour and synchronization with the subscribers -data buffering -backup publisher -safety capabilities in case the publisher is shutting down
OPC UA Subscribers	-easy/moderate difficulty in implementation -totally decoupled from the provider of the information -synchronization capabilities based on the described Synchronization Algorithm	- an initial first step is needed for transmitting preferences and ID (hard-coded information in the current implementation)	-real time behaviour and synchronization with the Broker App -less polling of the network -less filtering for the desired information

2.2.2 Approaching OPC UA Publish–Subscribe in the Context of UDP-Based Multi-Channel Communication and Image Transmission

Industrial image processing emerged out of necessity but was adopted without full integration into industrial protocols or production processes. Currently, communication with other systems relies on MES binary request–approval procedures and bit-wise result storage, while image storage and transfer lack protocol standardization. Despite this, image processing remains closely tied to production lines. For example, [43] demonstrates hydraulic axial pump diagnosis by converting signals into images via continuous wavelet transform and extracting features from time–frequency representations. Similarly, [44] applies image processing and deep learning to detect deformation in pantograph contact strips of railway vehicles. In [K-20], the authors present a low-cost OpenCV-based image processing solution for detecting defects in automotive parts manufacturing, specifically faulty or missing pins, clips, and board cracks in ECUs. Most studies treat industrial processes and image transmission as separate domains. In augmented reality, [45] proposes Node-RED with MQTT for communication with mechatronic devices, though image-related tasks remain focused on the iOS mobile application.

Beyond production, the automotive industry emphasizes autonomous and enhanced driving, concepts closely tied to image processing, car-to-infrastructure and car-to-car communication, and safety procedures. Significant progress has been achieved in image processing for autonomous driving [46], while research also targets safety improvements, such as detecting infrastructure cracks [47] and accident identification from traffic images [48]. Intelligent roadside devices now process traffic data using the YOLO-CA model, forwarding results to central systems for rescue operations and signaling vehicles. Although [48] focuses on image processing, it extends into car-to-infrastructure communication. Similarly, [49] explores queue length estimation via image processing, though results remain simulation-based. Since interoperability in car-to-infrastructure communication requires industrial protocols, works such as [K-21] and [K-30] propose OPC UA as a standardized solution.

Studies such as [50] examine industry standards compliance through OPC UA. Recent specifications (e.g. [30]) enable advancement toward the publish–subscribe paradigm, further developed in [K-21] and [K-19]. Within this context, the objectives of research [K-18] are: to extend [K-21] by analyzing publish–subscribe under real-time constraints and higher data volumes, including long-term single transmissions; to investigate multi-channel UDP communication to reduce data transfer duration; to broaden OPC UA applicability to image transmission; to apply the mechanism without hardware constraints, validated through case studies in automotive end-of-line testing and car-to-infrastructure communication.

Research and industrial testing of OPC UA publish–subscribe applications employ diverse hardware, though real-time capabilities favor low-cost, resource-constrained embedded devices in controller-to-controller scenarios. For example, [51] used multiple Xilinx boards to run publisher–subscriber exchanges of UADP messages with time analysis, while [52] implemented OPC UA entities on Raspberry Pi 3B+ using open-source stacks to measure efficiency. TSN provides time guarantees through specific standards. As noted in [53] and [54], OPC UA with TSN is expected to extend to field devices, while [55] highlights real-time applicability but only via simulations. Studies confirm that synchronization, low latency, and flexibility can be achieved by adapting TSN standards to OPC UA [51], [56].

With IIoT growth, the number of networked devices and required operations increases, making data management and extraction from distributed sources critical [57]. Complex architectures must evolve to meet reliability, robustness, and efficiency demands. OPC UA publish–subscribe enhances system capabilities, opening new use cases. In this context, an image transmission application based on OPC UA was implemented in to analyze the mechanism's potential from multiple perspectives.

2.2.2.1 Image-transmission over OPC UA Publish-Subscribe concept

Image transmission and processing are increasingly present in IIoT applications. Research has explored integrating OPC UA with automotive communication protocols, enabling publish–subscribe solutions for smart infrastructures [K-30], [K-21]. In automotive manufacturing, image processing is widely applied. For example, [K-20] describes automatic optical inspection (AOI) during ECU board end-of-line testing and packaging. Companies test tens of thousands of products daily using image processing to detect defects, while packaging is also image-assisted. These solutions generate images alongside bitwise/tag-based results, which are integrated into MES communication. However, image transfer often occurs in a rudimentary manner without standardized industrial protocols. An end-of-line process flow is presented in Fig. 2.2-10, where image processing is an important part for testing and packaging in automotive parts manufacturing. After pin insertion and ECU enclosure positioning, boards are tested with AOI to detect defects. The AOI system communicates with the MES, requesting approval to start testing and transmitting bitwise results indicating pass or fail. For Industry 4.0 integration, OPC UA represents the optimal protocol for image processing, provided it meets real-time, speed, and volume requirements. Building on [K-19] and [K-21], bitwise/tag-based communication ensures real-time constraints through the publish–subscribe mechanism. The objective is to extend OPC UA publish–subscribe beyond prior work to support long-term, high-volume, and faster data transmission, enabling full integration of image processing within the production flow. Complete vertical/horizontal interoperability could be achieved using OPC UA. As depicted in Fig. 2.2-10, after testing, boards are transported and placed into packaging boxes, with ECU counting performed through image processing. Once a box reaches capacity, a final image of its contents is stored before sealing and shipment. In this case, full OPC UA interoperability is also required, as image sizes are larger while transmission intervals are longer.

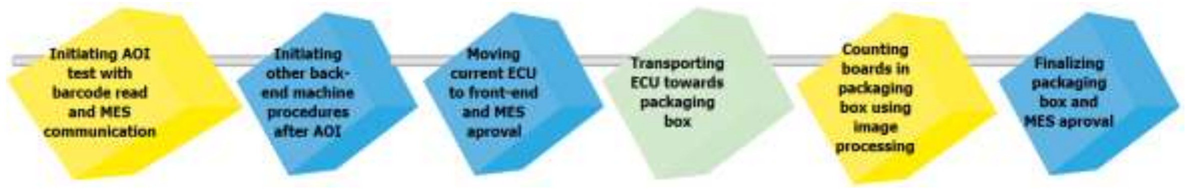

Initiating AOI test with barcode read and MES communication
Initiating other back-end machine procedures after AOI
Moving current ECU to front-end and MES aproval
Transporting ECU towards packaging box
Counting boards in packaging box using image processing
Finalizing packaging box and MES aproval

Fig. 2.2-10 End-of-line ECU testing using image processing in automotive manufacturing.

For image processing, the YOLOv3 model [58] was employed. YOLO uses a single deep convolutional network that divides the input image into a grid, with each cell predicting bounding boxes and object classes. Candidate boxes are consolidated through post-processing, and training on the COCO dataset enables detection across 80 object classes with reduced false positives. Additional operations relied on OpenCV, a cross-platform library widely used for real-time computer vision. OpenCV supports major deep learning frameworks such as PyTorch, TensorFlow, Caffe under the Apache2 license.

The image transmission application consists of two OPC UA instances, a publisher and a subscriber, designed to send images via the Pub-Sub mechanism. Each pixel is published individually and reconstructed at the subscriber side once all pixel values are received. Transmission speed and image quality serve as key indicators of reliability and efficiency. Tests were conducted with multiple images of varying resolutions and three application versions, each using different publishing intervals to evaluate how delivery speed affects image quality. Publisher and subscriber exchange *DataSetMessages*, each containing a pixel value in byte format (8-bit integer, range 0–255), generated by the Publisher's *DataSetWriter*. UDP was used as the transport protocol. The complete transmission process follows four main steps, as in Fig. 2.2-11.

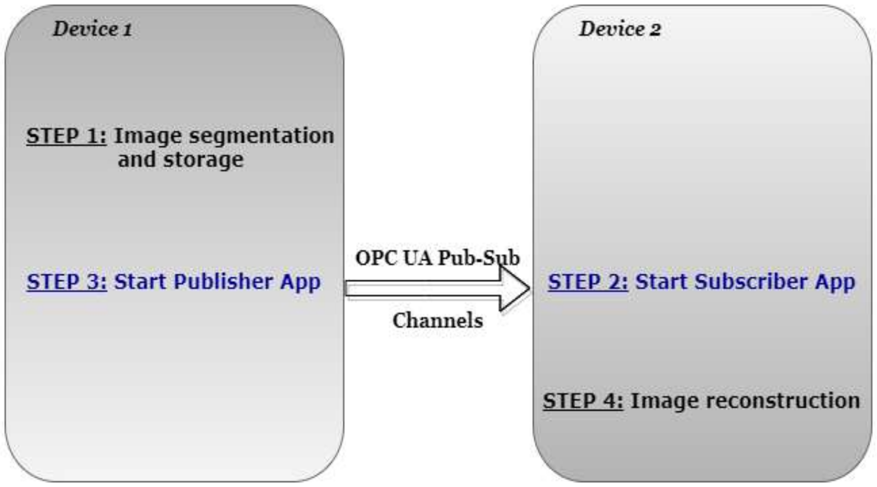

Device 1
STEP 1: Image segmentation and storage
STEP 3: Start Publisher App
OPC UA Pub-Sub
Channels
Device 2
STEP 2: Start Subscriber App
STEP 4: Image reconstruction

Fig. 2.2-11 OPC UA Publish-Subscribe image transmission steps.

Transmission durations and scenario-specific operations were measured using custom Linux timer functions. Networking tests were conducted on 2.4 GHz and 5 GHz networks. Although OPC UA Pub-Sub is real-time oriented, it cannot guarantee timing without TSN integration. Desynchronization may occur between publisher and subscriber devices. The Pub-Sub design primarily targets one-to-many communications. Yet, to meet real-time demands and avoid latency or desynchronization, multiple independent Pub-Sub channels can be deployed. This approach offers scalability for industrial operations that were previously infeasible under the traditional client–server paradigm.

2.2.2.2 Architecture and Implementation

The OPC UA image transmission application consists of two components: a publisher on the transmitting device and a subscriber on the receiving device. Additional operations, such as image segmentation before transmission and reconstruction afterward, must follow a defined sequence.

The first step in image segmentation and storage involves dividing the image into approximately equal pixel blocks. Each block is stored in a separate buffer, later accessed by the OPC UA publisher during transmission. Segmentation depends on both the number of Pub-Sub channels selected and the overall image size. Step 4 is the reverse process of the first step, the image reconstruction. The subscriber simultaneously receives image segments across multiple Pub-Sub channels and stores them in dedicated buffers. After transmission, these buffers are combined into a pixel file, with each segment placed in the correct order. When all channels transmit successfully and pixel values align properly, the image is accurately reconstructed on the receiver. The architecture is illustrated in Fig. 2.2-12.

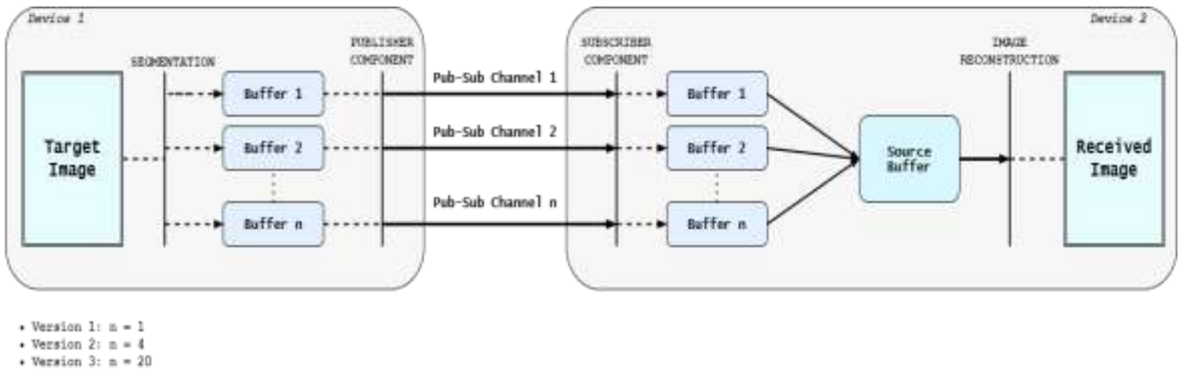


Fig. 2.2-12 General system architecture of the image transmission.

The publishing and receiving processes are synchronized to exchange pixel values at defined intervals, though each device maintains its own time base. This lack of a common reference can affect application behavior, particularly when publishing intervals fall below 10 ms. While minor pixel losses may not significantly impact image quality, a safety mechanism was developed for the subscriber side. Based on the target image resolution, each buffer corresponding to a Pub-Sub channel is tested and all received operations are counted. At transmission end, if buffer values do not match the expected pixel count, default values are inserted to complete the buffer. This ensures image reconstruction even when transmission is incomplete. By assessing reconstructed image quality, observers can identify which channels experienced desynchronization, losses, and approximate when they occurred.

2.2.2.3 Case Study 1 - Image Transmission over One and Four Pub-Sub Channels

A complete image transmission over OPC UA publish–subscribe was tested using a single channel, with a medium-large color image successfully transmitted. In the case of Fig. 2.2-13 and Fig. 2.2-14, Wifi connection was used, as a worst case physical support scenario. The quality of received images was evaluated against the originals. Publishing intervals were set between 1 ms (per pixel value) and 5 ms to ensure stability and maximize fidelity to the target image. A total of 773,490 pixel values were transmitted between publisher and subscriber at varying intervals. Results demonstrate that the system supports lengthy transmissions, approximately 12.9 minutes at 1 ms/pixel and 64.5 minutes at 5 ms/pixel, while successfully delivering complete images.

The second phase of the case study addresses a more realistic industrial scenario, using a lower-resolution black-and-white image to reduce the number of pixel values transmitted and achieve a practical transmission interval. The selected image contains 46,225 pixel values as payload. The results are in Fig. 2.2-15 and Fig. 2.2-16.

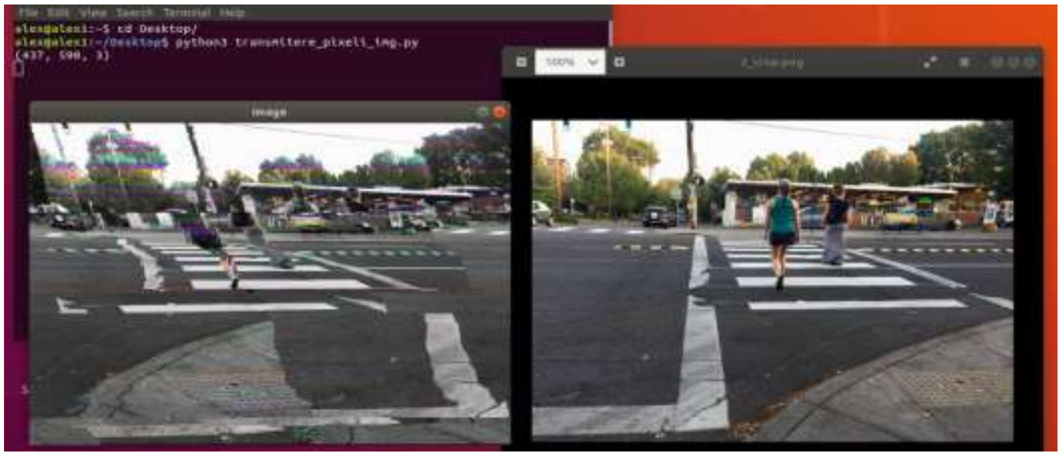
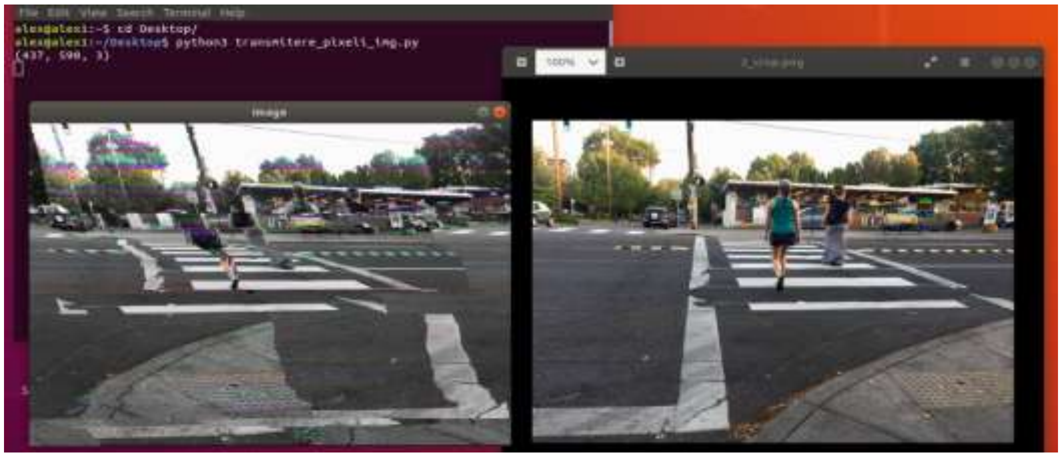
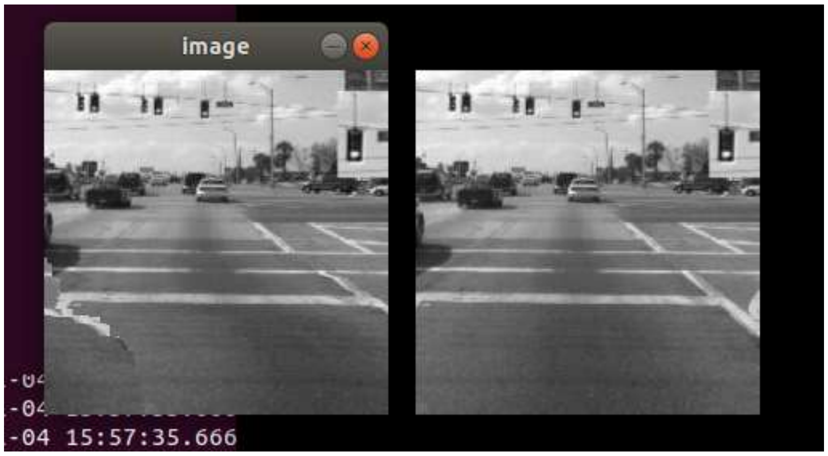

```
alex@alex1:~$ cd Desktop/
alex@alex1:~/Desktop$ python3 transmitere_pixeli_img.py
(437, 590, 3)
```

Fig. 2.2-13 The received image and the target image at 1 ms/pixel_value recurrence (phase 1).



```
alex@alex1:~$ cd Desktop/
alex@alex1:~/Desktop$ python3 transmitere_pixeli_img.py
(437, 590, 3)
```

Fig. 2.2-14 The received image and the target image at 5 ms/pixel_value recurrence (phase 1)



```
-04 15:57:35.666
```

Fig. 2.2-15 The received image and the target image at 1 ms/pixel_value recurrence (phase 2).

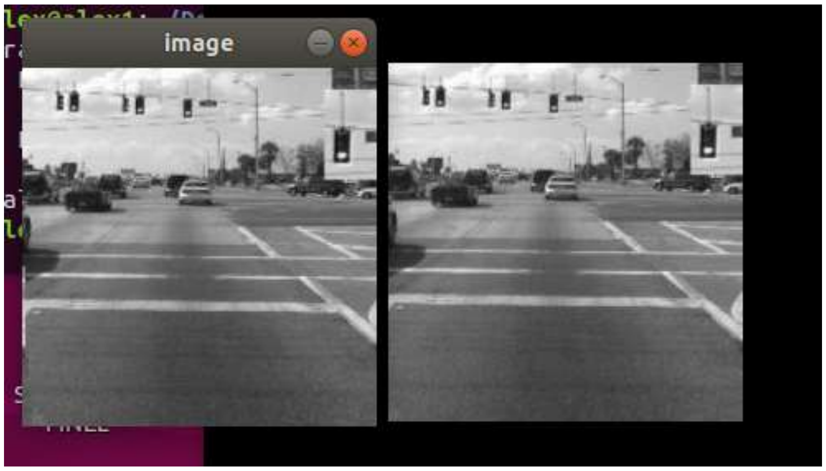


Fig. 2.2-16 The received image and the target image at 4 ms/pixel_value recurrence (phase 2)

In the second phase, the identical image was successfully received with a 4 ms publishing interval. A faster transmission was inherent, lowering the probability of desynchronization compared to phase 1, even under similar recurrence conditions. From Fig. 2.2-15, desynchronization occurred in the third quarter of the image with less degree of alterations. With the phase 2 target image, transmission time improved substantially: about 47 seconds at a 1 ms interval and 3.12 minutes at a 4 ms interval

The next step meant to increase the number of Pub-Sub channels to 4. Publishing intervals were configured identically to ensure that the duration of all four transmissions remained approximately equal. The results proved that the transmitter and received images were identical (the same as Fig. 2.2-16) at a publishing interval of 1 ms (1 millisecond/pixel value for all the 4 channels). A full transmission was obtained in approximately 12 sec.

2.2.2.4 Case Study 2 - Image Transmission over Twenty Pub-Sub Channels

The case study seeks to achieve feasible transmission times while demonstrating the scalability of multi-channel OPC UA publish–subscribe communication. Its objectives include integrating the concept into industrial scenarios using relevant process images and evaluating the impact of network capacity during implementation.

Case study 2 refers to 3 scenarios: the first consists of the previously discussed car-to-infrastructure communication, the second analyses the transmission of counted ECU boards image at the EoL in automotive manufacturing, while the third targeted ECU automatic optical inspection results in an automotive production line.

The goal was to obtain an image delivery time of under 3 sec. for the car- to-infrastructure communication, under 12 sec. for the EoL packaging boxes, and under 3 sec. the automatic optical inspection results.

For the car-to-infrastructure communication scenarios were, using 20 Pub-Sub channels with a 1 ms publishing interval achieved delivery in ~2.4 seconds, producing an image identical to the target. These results confirm the scalability of the multi-channel transmission concept, enabling flexible configurations for future applications.

For the final packaging boxes, ECUs were continuously counted, before delivering them to clients. Once the set limit is reached, the MES is notified that packaging is complete and the final image is stored locally. In the box-filling cycle, execution times are as follows: insertion of a new ECU requires at least 12 seconds, detection of a new box by the optical inspection system takes ~6 seconds, and complete filling, depending on box size and ECU count, exceeds 2.5 minutes. OPC UA multi-channel publish–subscribe image transmission results were promising. The target image was converted to grayscale and delivered fault-free to the destination, with the received image shown in Fig. 2.2-17. The complete image transmission was achieved in 7.85 sec. From a time perspective, this performance enables transmitting an image for each board inserted into the packaging box.

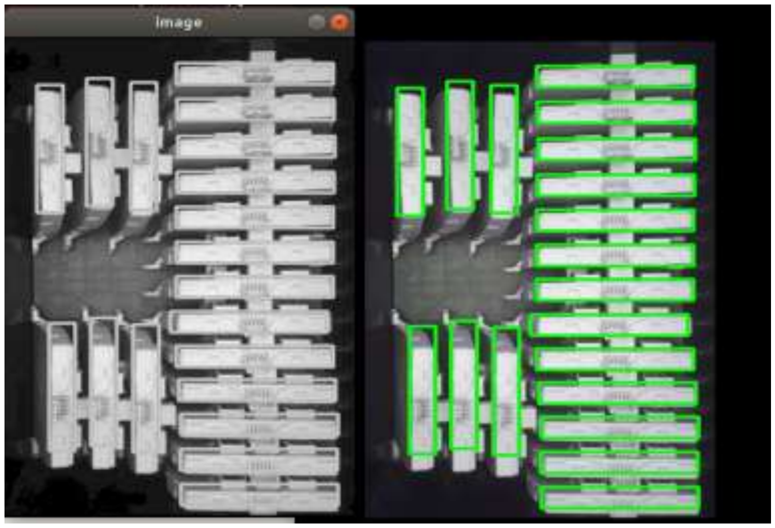


Fig. 2.2-17 OPC UA Publish-Subscribe 20 channel image transmission for packaging boxes in the automotive manufacturing - the received image in grayscale and the target image.

For the ECU EoL automatic optical inspection from [K-20], the image-based defect detection was achieved in 6.5 seconds. Considering other production line procedures, approximately 12 seconds are available to transmit inspection images within each board testing cycle. Only defect images (negative fault detection results) are stored and transmitted, typically 1–2 per cycle, since a single image may capture multiple defects in an analyzed area (e.g. a connector). With the implementation of 20-channel OPC UA publish–subscribe transmission, results were encouraging: the target image was converted to grayscale and transmitted completely to the destination. The received image in comparison with the target one is shown in Fig. 2.2-18.

A full image transmission was realized in 2.52 sec. Given the 12 sec. window in each board testing cycle, approximately 4 images can be transmitted using the 20-channel solution.

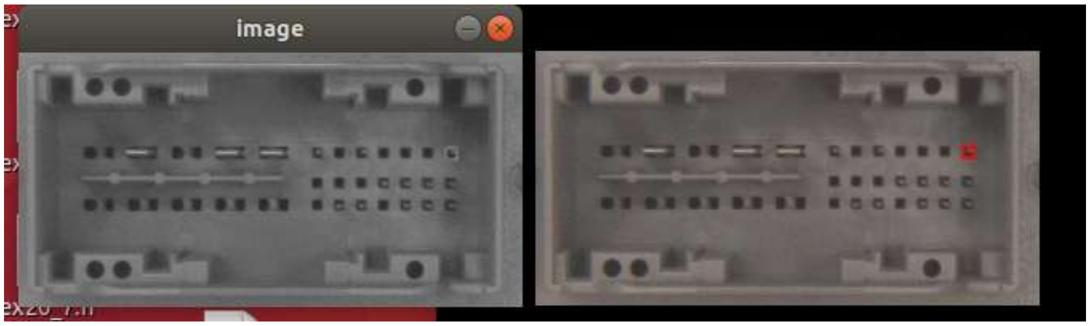


Fig. 2.2-18 OPC UA Publish-Subscribe 20 channel image transmission for ECU automatic optical inspection process in the automotive manufacturing: the received image in grayscale and the target image.

2.2.2.5 Results of the Study

Each case study provided key insights into the mechanism's stability and robustness, the factors influencing transmission in industrial contexts, and the application's performance, confirming its potential to meet objectives in a new domain of the OPC UA protocol. In Table 2-2, a comparative analysis of all case study outcomes highlights the advantages and disadvantages of each application version within its designated context.

Table 2-2 Outcome analysis for the case studies

Case study	Number of Pub-Sub Channels	Total time for a full transmission of an identical image	Factors that can produce instability	Conclusions
1-a	1	64.5 minutes for phase 1 (publishing interval of 5 ms / pixel value) 3.12 minutes for phase 2 (publishing interval of 4 ms / pixel value)	- high volume of information needed to be transmitted by 1 channel - high length of the transmission increase the probability of desynchronization between devices	- not feasible for industrial processes
1-b	4	12 seconds (publishing interval of 1 ms / pixel value)	- lower volume of information needed to be transmitted by 1 channel - not guaranteeing a low probability of desynchronization between devices	- improved performances but far from the desired outcome
2	20	2.4 seconds (publishing interval of 1 ms / pixel value)	- adequate volume of information needed to be transmitted by 1 channel - guaranteeing very low probability for desynchronization between devices	- feasible in industrial scenarios for specific processes

2.2.3 DDS and OPC UA Protocol Coexistence Solution in Real-Time and Industry 4.0 Context Using Non-Ideal Infrastructure

Following [K-19] and [K-18], the next step was to cover other important component of the OT level, the industrial robots, respectively to establish solution for interoperation. Manufacturing production lines incorporate robots, co-bots, and compact units, where future architectures will likely adopt the real-time DDS protocol defined by the Object Management Group (OMG) (see Fig. 2.2-19). In this context, analyzing the coexistence of OPC UA and DDS publish–subscribe solutions under real-time industrial requirements is essential. This chapter represents work [K-16], presenting a tool and methodology to evaluate the real-time performance of both protocols, tested in parallel and within a gateway configuration.

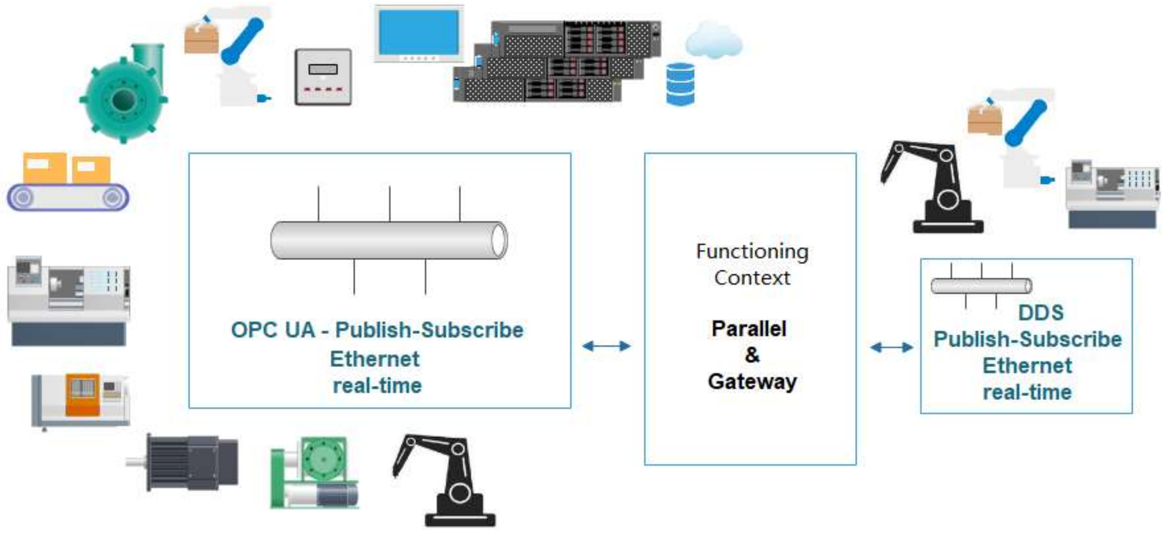


Fig. 2.2-19 Schematic view of OPC UA—DDS protocol coexistence in the Industry 4.0 context.

Due to the slow adoption of TSN in industry, faster approaches for real-time constraints on operating systems and equipment are required. Therefore, there is a need to: define criteria for evaluating DDS and OPC UA in non-ideal systems facing industrial challenges, analyze their real-time behavior, propose an architecture enabling parallel use and interaction of both protocols, implement a DDS–OPC UA gateway application.

DDS employs a data-centric model with a global data space (GDS) accessible to all entities, where information is propagated once roles are defined. Its publish–subscribe paradigm is topic-based, with subscribers expressing interest and matched to publishers. DDS also supports request/reply communication, ensuring efficiency in one-to-many, one-to-one, and many-to-many scenarios. Entities can be grouped into domains, creating isolated virtual spaces for flexible and complex data exchange.

Communication relies on the Real-Time Publish–Subscribe (RTPS) protocol, ensuring interoperability over standard networks while meeting real-time requirements. At the transport layer, RTPS operates over TCP/UDP/IP, maintaining portability and compatibility across DDS implementations. On Linux systems, DDS provides configurable blocking intervals for resource-contending functions, with prioritization mechanisms applied when limits are exceeded. The eProsima Fast DDS [59] open-source SDK was employed for DDS entity implementation. Its API is divided into two layers, one tied to the wire protocol and another abstracting DDS concept, while built-in mechanisms ensure real-time behavior. For high-volume, time-critical communication, DDS provides the Persistence Service, enabling rapid recovery after shutdown. By storing context details and the last notified data changes between components, the system can quickly restore its previous communication state, meeting hard real-time constraints.

DDS implementations span diverse domains and use cases. In [60], the advantages of the publish–subscribe paradigm and RTPS adoption for interoperability across vendors are highlighted, with applications such as underwater vehicle data exchange demonstrating feasibility in extreme environments. In [61], challenges in configuring QoS for real-time systems and the absence of DDS standards are noted, increasing complexity in understanding dynamic models. A framework-based tool is proposed to enhance software reusability across heterogeneous IoT architectures.

ROS 2, an open-source framework for complex robotic applications, is gaining traction in both research and industry [62,63]. DDS has been validated as its communication middleware [64], enabling decentralized communication and supporting hard real-time requirements. DDS in ROS 2 defaults to asynchronous publication, where data are queued and handled by background threads, suitable for non-time-critical nodes. Synchronous publication, controlled by the main thread, ensures precise timing and minimal latency for time-critical events. ROS 2 adoption in academia remains early, hindered by migration challenges from ROS 1 [65]. Nonetheless, its Industry 4.0 oriented features are expected to drive replacement. In [66], DDS is emphasized as central to ROS 2 objectives, including multi-robot cooperation, embedded systems with limited resources, real-time constraints, and communication over unstable networks.

2.2.3.1 Envisioned Architecture for the Analysis

To advance industrial solutions, a multi-node mirror architecture was defined with three DDS nodes and three OPC UA nodes, each assigned specific roles on native or virtualized Linux systems. The architecture (see Fig. 2.2-20) supports both performance comparison and gateway-based interaction scenarios, aiming to confirm compatibility and feasibility between the two protocols while establishing general criteria and expectations. Rather than relying on trial-and-error, the research community must provide neutral guidelines, performance metrics, and architectural perspectives to highlight advantages and limitations comprehensively.

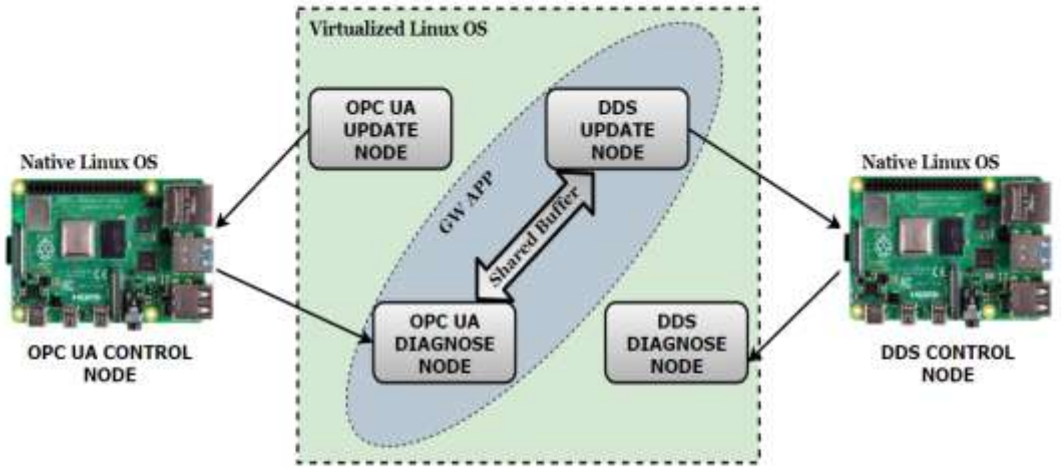


Fig. 2.2-20 The envisioned architecture for analysis.

OPC UA and DDS control nodes act as primary data consumers, running on Raspberry Pi 4 devices with native Linux systems. In real scenarios, they represent the main control segment (e.g., PLC, robot, or production line controllers). Each node integrates two communication subcomponents, the subscriber receiving updates from the update node at defined intervals, and the publisher forwarding information to diagnose nodes for safety or diagnostic operations. Update nodes serve as the main producers and distributors, accessible only to control nodes. Diagnose nodes receive data exclusively from control nodes, performing validation or acting as gateways for OPC UA–DDS interaction (e.g., transferring PLC data via OPC UA to a DDS-controlled motor). Gateway configurations rely on shared buffers between complementary entities, ensuring architectural flexibility.

Evaluation criteria, defined in [K-16], include first real-time responsiveness testing of publish/subscribe operations at device level, compared against ideal expectations. Percentage-based results estimate performance in industrial scenarios, highlighting similarities and differences between OPC UA and DDS under identical conditions, confirming their compatibility even in non-ideal setups. Also, include data buffering analysis, comparing received values with expected amounts. This ensures not only execution monitoring but also acknowledgment of data integrity, with network stability as a critical factor.

2.2.3.2 Case Studies and Results

Two case studies were conducted. The first examines how DDS and OPC UA respond under standard OSs without enhanced real-time capabilities and non-ideal networks lacking transmission guarantees, across varying time intervals. The second introduces a gateway solution for protocol interaction, switchable without architectural changes.

Case study 1 focuses on behavior analysis of both protocols. Evaluation criteria confirm comparable responses, supporting future cross-domain architectures that integrate DDS and OPC UA. In non-ideal infrastructures, real-time performance declines below 10 ms expectations, this study quantifying degradation at 10, 5, 2, and 1 ms intervals. The first criterion measures function calls executable as intervals shorten, with results obtained via a custom scheduler running independently from communication operations. A clearer view on each multithreaded node application is shown in Fig. 2.2-21.

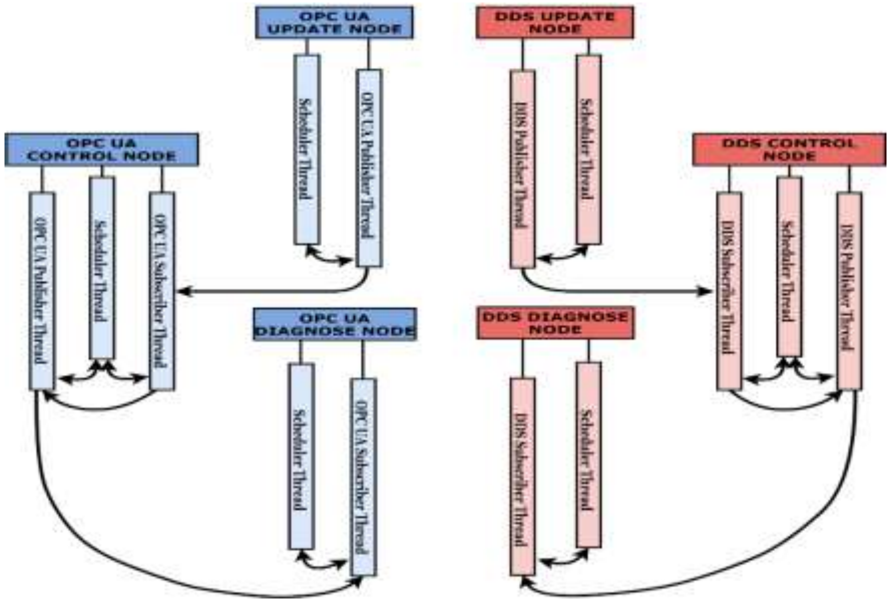


Fig. 2.2-21 Multithreading nodes from an architectural perspective.

Functions call verification process results are shown in Table 2-3 for DDS, and in Table 2-4 for OPC UA for the first criteria. The tables present success rates for each time recurrence.

The second criteria concentrates on data-buffering mechanism, to highlight OS and device desynchronization influence on receiving data (see results in Table 2-5).

Table 2-3 DDS Nodes

DDS Update Node – Virtualized Linux OS – Publish Operation Publish Operation – Recurrent Execution Check			
10 ms	5 ms	2 ms	1 ms
≈ 100 %	≈ 90 %	≈ 74 %	≈ 64 %
TOTAL Number of Tests : 2790			
DDS Control Node – Native Linux OS – Publish Operation Publish Operation – Recurrent Execution Check			
10 ms	5 ms	2 ms	1 ms
≈ 100 %	≈ 93 %	≈ 84.6 %	≈ 77 %
TOTAL Number of Tests: 2865			
DDS Control Node – Native Linux OS – Subscribe Operation Subscribe Operation – Recurrent Execution Check			
10 ms	5 ms	2 ms	1 ms
≈100 %	≈ 85 %	≈65 %	≈48.5 %
TOTAL Number of Tests: 2805			
DDS Diagnose Node – Virtualized Linux OS – Subscribe Operation Subscribe Operation – Recurrent Execution Check			
10 ms	5 ms	2 ms	1 ms
≈100 %	≈85 %	≈65 %	≈47 %
TOTAL Number of Tests: 3015			

Table 2-4 OPC UA Nodes

OPC UA Update Node – Virtualized Linux OS – Publish Operation Publish Operation – Recurrent Execution Check			
10 ms	5 ms	2 ms	1 ms
≈100 %	≈95 %	≈81.2 %	≈56 %
TOTAL Number of Tests: 2685			
OPC UA Control Node – Native Linux OS – Publish Operation Publish Operation – Recurrent Execution Check			
10 ms	5 ms	2 ms	1 ms
≈100 %	≈100 %	≈87 %	≈56 %
TOTAL Number of Tests: 2970			
OPC UA Control Node – Native Linux OS – Subscribe Operation Subscribe Operation – Recurrent Execution Check			
10 ms	5 ms	2 ms	1 ms
≈100 %	≈100 %	≈91 %	≈85 %
TOTAL Number of Tests: 3015			
OPC UA Diagnose Node – Virtualized Linux OS – Subscribe Operation Subscribe Operation – Recurrent Execution Check			
10 ms	5 ms	2 ms	1 ms
≈100 %	≈87.5 %	≈77%	≈64 %
TOTAL Number of Tests: 3015			

Table 2-5 Data buffering success rate

	OPC UA Control Node	OPC UA Diagnose Node	DDS Control Node	DDS Diagnose Node
10 ms	95%	93%	91%	91%
5 ms	95%	86%	83%	83%
2 ms	86%	76%	64%	64%
1 ms	77%	62%	43%	43%

Subscriber-side data buffering results confirm the findings from recurrent function call verification. Both criteria show proportional outcomes, with lower percentages for buffering, validating the added impact of network stability on data exchange beyond OS real-time limitations.

The second case study addresses the gateway application, which enables configurable data exchange between control nodes. Results highlight the influence of device desynchronization and network instability. Data propagation through the multi-node architecture is observable via the digital signal generated by the DDS control node from OPC UA payloads (see Fig. 2.2-22). At 100 ms recurrence, data transmission is accurate. For intervals below 10 ms, the gateway maintains delivery, but closer to 1 ms external factors, such as delayed OS responses across nodes, become significant. Since the signal depends on multiple node exchanges, any delay perturbs payload delivery, increasing the risk of inaccuracies.

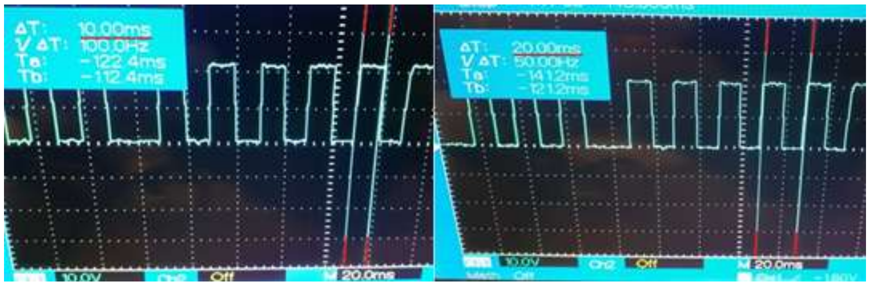


Fig. 2.2-22 Generated Digital Signal based on payload delivered by the Gateway Application at 10ms recurrence.

2.3 Modern Protocols Emerging and Coexistence in the Automotive Sector.

The current chapter continues the investigation of concepts from 2.1 and 2.2, but oriented towards the automotive sector. With the mentioned advances in the area of OPC UA interfacing and the continuously growing requirements of the industrial automation world, combined with the more and more complex configurations of ECUs inside vehicles and services associated to car to infrastructure and even car to car communications, the gap between the two domains must be analyzed and filled. This gap occurred mainly because of the rigidness and lack of transparency of the software-hardware part of the automotive sector and the new demands for car to infrastructure communications. Analyzing the VSOME/IP notify–subscribe mechanism, a VSOME/IP–OPC UA gateway can bridge protocol gaps between automotive and automation domains. Compatibility and real-time responsiveness must be assessed within diverse service-oriented architectures for automotive IoT Ethernet communication. This feasibility study is realized through a multi-protocol gateway enabling data exchange among SOME/IP, DDS, and eCAL entities for future communication scenarios. In this context, section 2.3.1 depicts the findings from works [K-15], [K-21], [K-30].

As automotive systems transition to zonal and software-defined architectures, efficient and adaptable communication protocols are increasingly critical. Case study validation is essential to assess middleware suitability for real-world integration. Section 2.3.2 presents findings from [K-2], introducing Zenoh as a lightweight, data-centric protocol built on modern networking paradigms. Zenoh was implemented in an automotive scenario with distributed zone controllers and an in-vehicle server, with DDS serving as a benchmark due to its proven performance in prior research. Experimental results highlight Zenoh's strengths in message integrity and resource efficiency, particularly under high-frequency data transmission. Unlike traditional middleware, Zenoh demonstrates strong adaptability in distributed environments with limited computational resources.

2.3.1 Some/IP, DDS, OPC UA in Automotive

Automotive developments increasingly emphasize centralized and integrated control of dynamic environments. V2X technology enables vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication via wireless networks, with control centers collecting safety data (accidents, traffic, weather) through LTE, CCTV, and GPS [69]. Premium vehicles now integrate over 100 ECUs, driving demand for inter-ECU communication. Gateways remain critical, ensuring reliable message transmission across heterogeneous networks. For example, [70] describes a gateway supporting CAN, Wi-Fi, and RS-232. Research also explores risk models for intelligent transportation systems [71] and reinforcement learning for connected vehicle control at intersections [72]. OPC UA is increasingly recommended for automotive communication, including traffic signal integration [73]. Studies highlight converting intra-car CAN systems into OPC UA, with gateway servers implemented using Unified Automation C# SDK [74]. Broader perspectives on new in-vehicle protocols are presented in [75].

Modern vehicles require high bandwidth and low latency, beyond the capabilities of CAN and FlexRay. Ethernet is expected to become the backbone of next-generation architectures, supported by high-performance gateways to centralize ECU data, while CAN, LIN, and FlexRay remain for specific applications [76]. Vehicles are evolving toward intelligent, Internet-connected systems where cyclic signal-based communication (LIN, CAN, FlexRay) coexists with service-based, event-driven IP networks. SOME/IP introduces service-oriented transmission, reducing unnecessary traffic [77]. Currently, the industry views SOME/IP, particularly VSOME/IP (Genivi's implementation), as the next-generation protocol for in-car ECU communication [78,79]. In V2I contexts, the first objective is to design a gateway enabling conversion and wrapping between SOME/IP and OPC UA. The first step in this direction was presented in [K-30], using the classic client-server paradigm for OPC UA.

The approach in [K-21] outlines steps for implementing the OPC UA publish–subscribe mechanism and testing it alongside the VSOME/IP notify–subscribe mechanism through a gateway. The study also provides insights into real-time behavior and future perspectives.

With microprocessors increasingly shaping vehicle architectures, supported by Adaptive AUTOSAR and POSIX-based operating systems, new concepts and communication technologies can be analyzed in industrial contexts, highlighting concrete advantages and limitations. Beyond SOME/IP, DDS is also investigated as a key automotive protocol [80–81]. In [K-15], another objective was achieved: offering insights into state-of-the-art communication protocols aligned with automotive demands and exploring a gateway solution based on AUTOSAR-compliant Ethernet technologies such as SOME/IP and DDS, together with the emerging middleware eCAL.

2.3.1.1 Approaches regarding Some/IP and OPC UA

The first approach from [K-30] targeted a Some/IP – OPC UA gateway. The research considered the classic OPC UA client/server based representation, and the VSOME/IP request-response mechanism.

The gateway general architecture is presented in Fig. 2.3-1, and targeted a V2I communication as depicted in the case study scheme from Fig. 2.3-2.

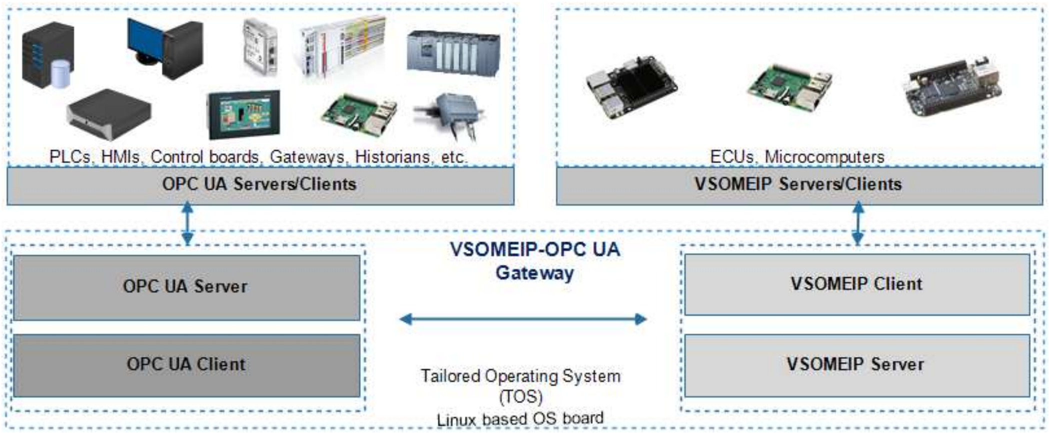


Fig. 2.3-1 General architecture of the VSOMEIP-OPC UA gateway.

Two scenarios were tested. First, elementary messages were sent from the OPC UA server (e.g. "0xAA" message as in Fig. 2.3-3), and reaching from the

OPC UA client and VSOME/IP server to the VSOME/IP client. The VSOME/IP client sends back an acknowledgement message (e.g. "0xCC in Fig. 2.3-3).

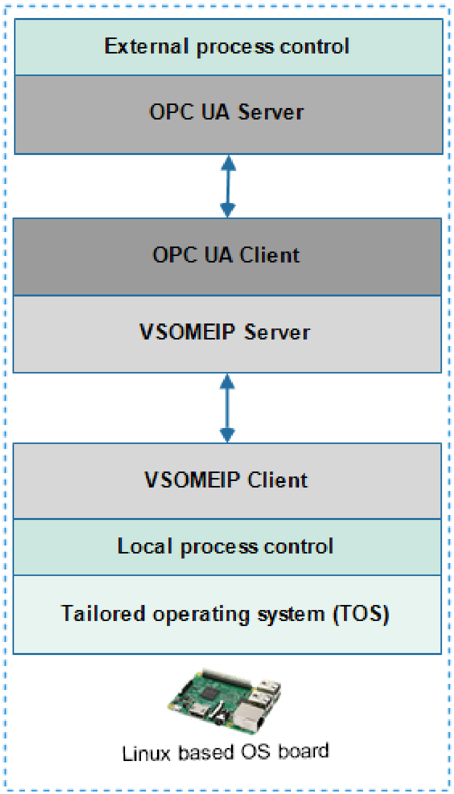
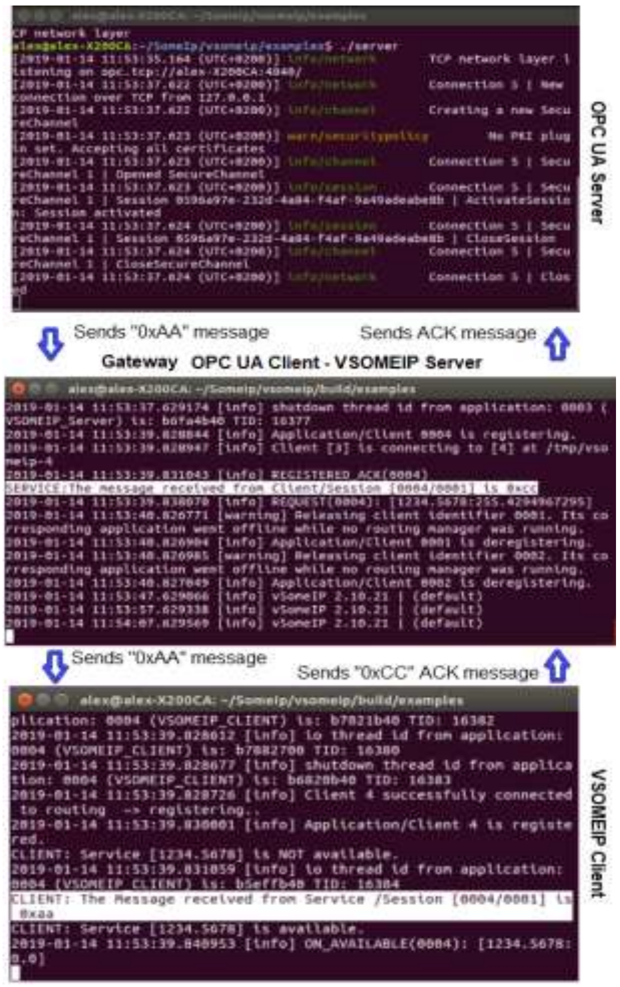


Fig. 2.3-2 Case study implemented architecture.



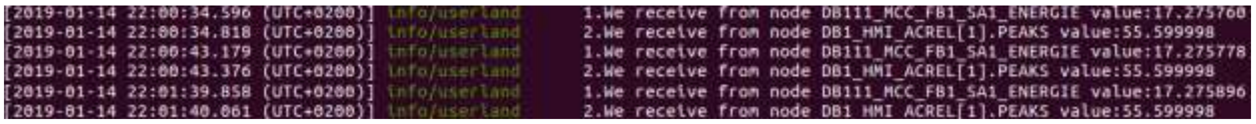
```
alex@alex-X200CA: ~/Someip/vsomeip/examples
CP network layer
alex@alex-X200CA:~/Someip/vsomeip/examples$ ./server
[2019-01-14 11:53:35.164 (UTC+0200)] info/network        TCP network layer l
istening on opc.tcp://alex-X200CA:4840/
[2019-01-14 11:53:37.622 (UTC+0200)] info/network        Connection 5 | New
connection over TCP from 127.0.0.1
[2019-01-14 11:53:37.622 (UTC+0200)] info/channel        Creating a new Secu
reChannel
[2019-01-14 11:53:37.623 (UTC+0200)] warn/securitypolicy        No PKI plug
in set. Accepting all certificates
[2019-01-14 11:53:37.623 (UTC+0200)] info/channel        Connection 5 | Secu
reChannel 1 | Opened SecureChannel
[2019-01-14 11:53:37.623 (UTC+0200)] info/session        Connection 5 | Secu
reChannel 1 | Session 6596a97e-232d-4a84-f4af-9a49edeabe8b | ActivateSessio
n: Session activated
[2019-01-14 11:53:37.624 (UTC+0200)] info/session        Connection 5 | Secu
reChannel 1 | Session 6596a97e-232d-4a84-f4af-9a49edeabe8b | CloseSession
[2019-01-14 11:53:37.624 (UTC+0200)] info/channel        Connection 5 | Secu
reChannel 1 | CloseSecureChannel
[2019-01-14 11:53:37.624 (UTC+0200)] info/network        Connection 5 | Clos
ed
```

```
alex@alex-X200CA: ~/Someip/vsomeip/build/examples
2019-01-14 11:53:37.629174 [info] shutdown thread id from application: 0003 (
VSOMEIP_Server) is: b6fa4b40 TID: 16377
2019-01-14 11:53:39.828844 [info] Application/Client 0004 is registering.
2019-01-14 11:53:39.828947 [info] Client [3] is connecting to [4] at /tmp/vso
meip-4
2019-01-14 11:53:39.831043 [info] REGISTERED_ACK(0004)
SERVICE:The message received from Client/Session [0004/0001] is 0xcc
2019-01-14 11:53:39.838070 [info] REQUEST(0004): [1234.5678:255.4294967295]
2019-01-14 11:53:40.826771 [warning] Releasing client identifier 0001. Its co
rresponding application went offline while no routing manager was running.
2019-01-14 11:53:40.826904 [info] Application/Client 0001 is deregistering.
2019-01-14 11:53:40.826985 [warning] Releasing client identifier 0002. Its co
rresponding application went offline while no routing manager was running.
2019-01-14 11:53:40.827049 [info] Application/Client 0002 is deregistering.
2019-01-14 11:53:47.629066 [info] vSomeIP 2.10.21 | (default)
2019-01-14 11:53:57.629338 [info] vSomeIP 2.10.21 | (default)
2019-01-14 11:54:07.629569 [info] vSomeIP 2.10.21 | (default)
```

```
alex@alex-X200CA: ~/Someip/vsomeip/build/examples
plication: 0004 (VSOMEIP_CLIENT) is: b7821b40 TID: 16382
2019-01-14 11:53:39.828612 [info] io thread id from application:
0004 (VSOMEIP_CLIENT) is: b7882700 TID: 16380
2019-01-14 11:53:39.828677 [info] shutdown thread id from applica
tion: 0004 (VSOMEIP_CLIENT) is: b6820b40 TID: 16383
2019-01-14 11:53:39.828726 [info] Client 4 successfully connected
 to routing  -> registering..
2019-01-14 11:53:39.830001 [info] Application/Client 4 is registe
red.
CLIENT: Service [1234.5678] is NOT available.
2019-01-14 11:53:39.831059 [info] io thread id from application:
0004 (VSOMEIP_CLIENT) is: b5effb40 TID: 16384
CLIENT: The Message received from Service /Session [0004/0001] is
 0xaa
CLIENT: Service [1234.5678] is available.
2019-01-14 11:53:39.840953 [info] ON_AVAILABLE(0004): [1234.5678:
0.0]
```

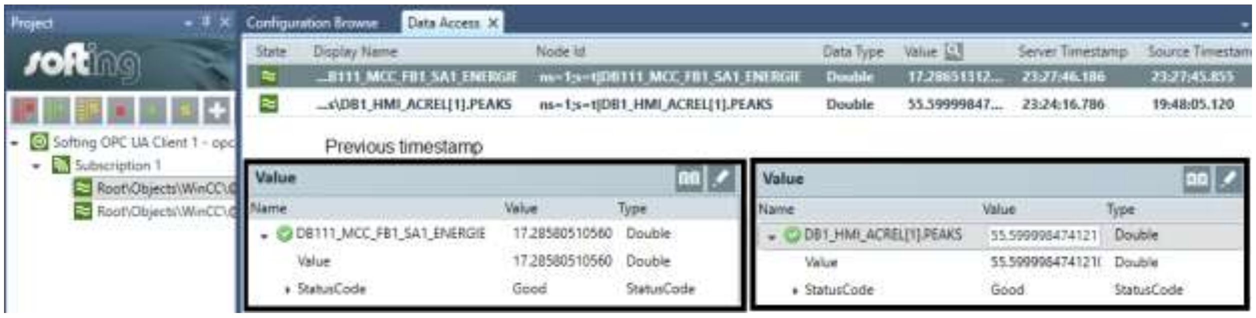
Fig. 2.3-3 Screenshots associated to the informational flow for case study 1.

A second case study targeted external OPC UA server access, to test the OPC UA client. The OPC UA server is part of a real industrial plant and linked to a WinCC Professional v13 SCADA system. Two tags, electrical energy values within the plant, were retrieved. The OPC UA client in the developed gateway reads these tag values directly from the server, as illustrated in Fig. 2.3-4, where three timestamped sets of values are received. To evaluate OPC UA interfacing efficiency, a widely used industrial OPC UA client (Softing) was employed for result comparison within a close time frame (see Fig. 2.3-5).



```
[2019-01-14 22:00:34.596 (UTC+0200)] info/userland        1.We receive from node DB111_MCC_FB1_SA1_ENERGIE value:17.275760
[2019-01-14 22:00:34.818 (UTC+0200)] info/userland        2.We receive from node DB1_HMI_ACREL[1].PEAKS value:55.599998
[2019-01-14 22:00:43.179 (UTC+0200)] info/userland        1.We receive from node DB111_MCC_FB1_SA1_ENERGIE value:17.275778
[2019-01-14 22:00:43.376 (UTC+0200)] info/userland        2.We receive from node DB1_HMI_ACREL[1].PEAKS value:55.599998
[2019-01-14 22:01:39.858 (UTC+0200)] info/userland        1.We receive from node DB111_MCC_FB1_SA1_ENERGIE value:17.275896
[2019-01-14 22:01:40.061 (UTC+0200)] info/userland        2.We receive from node DB1_HMI_ACREL[1].PEAKS value:55.599998
```

Fig. 2.3-4 OPC UA client from the gateway accessing external OPC UA server (reading two tags).


Project

Softing OPC UA Client 1 - opc

Subscription 1

Root\Objects\WinCC\

Root\Objects\WinCC\

Configuration Browse | Data Access

State	Display Name	Node Id	Data Type	Value	Server Timestamp	Source Timestam
	...B111_MCC_FB1_SA1_ENERGIE	ns=1;s=t|DB111_MCC_FB1_SA1_ENERGIE	Double	17.28651312...	23:27:46.186	23:27:45.855
	...s\DB1_HMI_ACREL[1].PEAKS	ns=1;s=t|DB1_HMI_ACREL[1].PEAKS	Double	55.59999847...	23:24:16.786	19:48:05.120

Previous timestamp

Value

Name	Value	Type
DB111_MCC_FB1_SA1_ENERGIE	17.28580510560	Double
Value	17.28580510560	Double
StatusCode	Good	StatusCode

Value

Name	Value	Type
DB1_HMI_ACREL[1].PEAKS	55.599998474121	Double
Value	55.599998474121	Double
StatusCode	Good	StatusCode

Fig. 2.3-5 Softing OPC UA client accessing OPC UA server.

The second step was to approach the OPC UA publish-subscribe mechanism according to the OPC Unified Architecture Part 14. The gateway concept was maintained in [K-21], and other features of the two protocols were analyzed. As a transport protocol, UDP is the chosen option for this case. Fig. 2.3-6 illustrates the configuration sequence for all components in the OPC UA publish–subscribe pattern, including the steps preceding and following information transmission and reception.

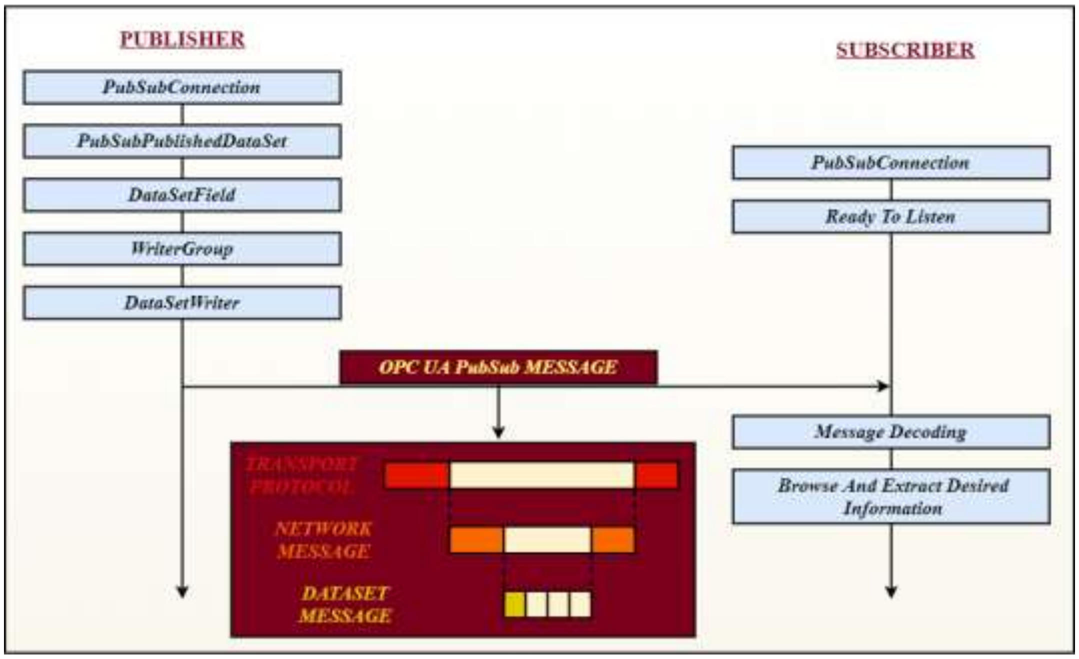


Fig. 2.3-6 OPC UA Publish-Subscribe configuration components.

In applying communication protocols with the publish–subscribe mechanism, it was necessary to establish correlations between publishing and receiving times for both protocols, together with a dependency procedure among the gateway application components (see Fig. 2.3-7). The VSOME/IP notify-subscribe pattern is an event-driven mechanism enabling publisher–subscriber communication through service discovery. Data exchange occurs via communication endpoints, which define transport protocols and configuration parameters such as port numbers, multicast addresses, and protocol details. These parameters are stored in a JSON-based VSOME/IP configuration file. The notifying (publishing) process depends on the OPC UA subscribing process: no transmission to the target client occurs until data arrives from the OPC UA publisher. Once received, the VSOME/IP message is transmitted, and the execution sequence returns to the OPC UA subscribe process within a time shorter than the OPC UA publishing interval.

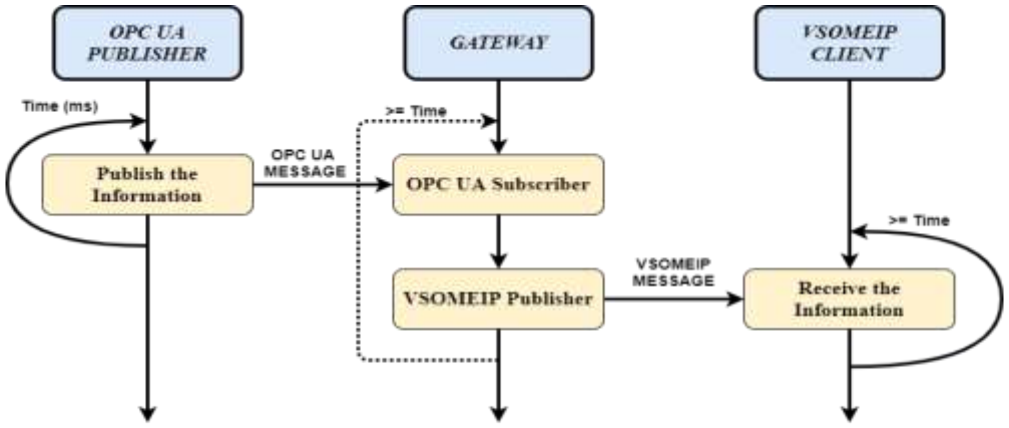


Fig. 2.3-7 Gateway application components dependency, OPC UA and VSOME/IP in the Real-Time Publish-Subscribe context.

Although [K-21] expands the gateway concept with several OPC UA developments, the current implementation focuses on VSOME/IP for intra-car communication and OPC UA for infrastructure. The scenario addresses V2I interfacing (e.g. Fig. 2.3-8). The semaphore is modeled through an OPC UA server that stores vehicle-relevant information. Three nodes hold the traffic light colors and semaphore coordinates, enabling distinction among multiple semaphores. The light color is updated by an OPC UA client running on the same machine, using Linux timers to trigger changes. Concept validation involves a third client that reads and displays server data. A detailed representation of the three devices is provided in Fig. 2.3-9.

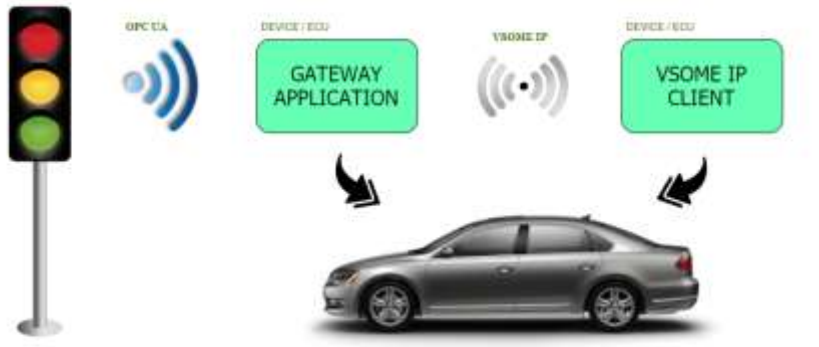


Fig. 2.3-8 Gateway case study general architecture.

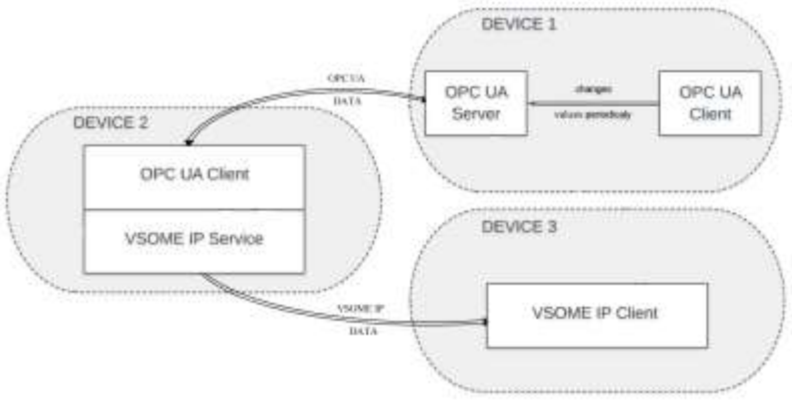


Fig. 2.3-9 The three devices used in the case study architecture.

Reading from the OPC UA server and transmitting to the VSOME/IP client are synchronized by a common timer, ensuring predictable cyclic behavior for the SOME/IP notify–subscribe mechanism. In certain scenarios, however, server reading may desynchronize from transmission. Data exchange was tested at recurrences ranging from 1 ms to 10 s using a cyclic gateway timer controlling both reading and forwarding processes. For all tested intervals, the VSOME/IP client received correct data, though transmission latency increased with larger data volumes transferred from the OPC UA server.

2.3.1.2 *Multi-Protocol Gateway in an Automotive V2X context*

Future interoperability challenges lie in interfacing diverse technologies and architectures, each with distinct hardware resources and requirements. In [K-15], a combined automotive–IoT use case defined an architecture with mixed SOME/IP, DDS, and eCAL nodes communicating via a gateway. From a hardware perspective, automotive specific technologies guided the use of microprocessors with native POSIX operating systems for most nodes. Interaction with an IoT supervisor was simulated through a virtualized Linux OS on a general-purpose computer. SOME/IP and DDS nodes represent intra-car communication infrastructure aligned with AUTOSAR compliance, while the gateway ensures data exchange between them and supports interaction with the eCAL supervisor located separately. Each entity operates on a distinct device, with transmitted data structured as cyclic heartbeat events. The hardware architecture is illustrated in Fig. 2.3-10.

In addition to developing SOME/IP, DDS, and eCAL publishers and subscribers, the gateway coordinates all participants to ensure efficient payload distribution, accounting for desynchronizations caused by network instability or operating system delays under real-time constraints. Two gateway versions were defined, depending on the heartbeat event provider. Each version integrates middleware-specific subcomponents aligned with assigned roles, with every subcomponent transmitting and receiving data on separate threads. This multithreaded design improves observation of recurrent transmissions, delays, and connectivity issues. Data delivery remains efficient, maximizing application responsiveness. The system architecture for both versions is illustrated in Fig. 2.3-11.

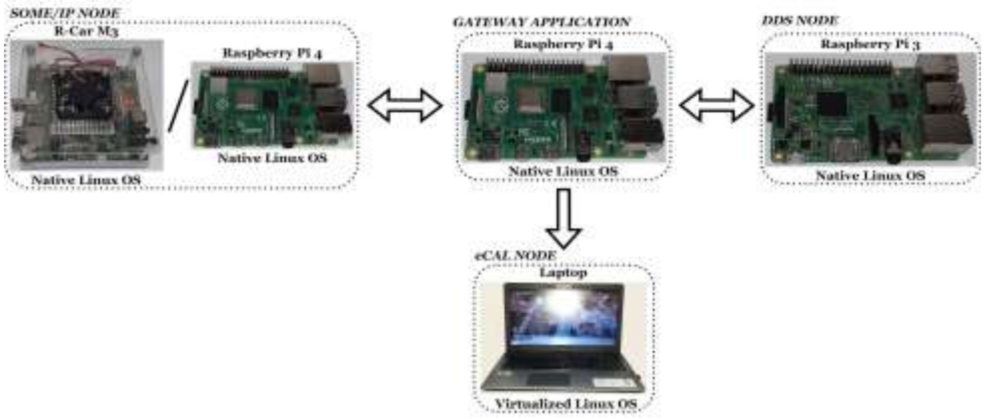


Fig. 2.3-10 Hardware architecture of the multi-protocol gateway.

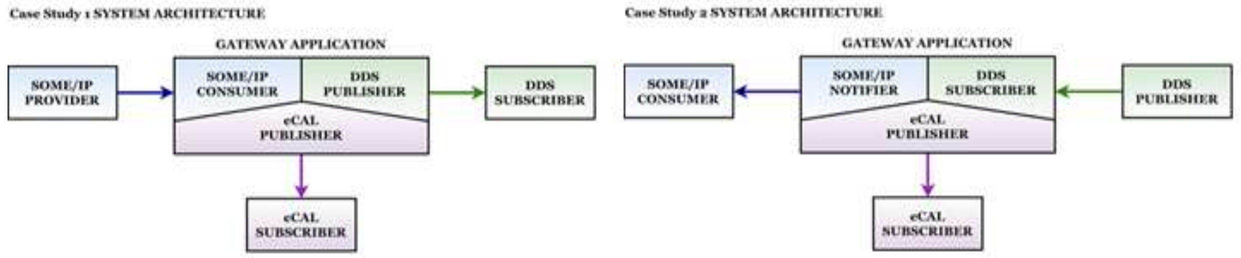


Fig. 2.3-11 System architecture showcasing both versions of the gateway.

The concept and development of the case studies enable clear observation of procedure sequences within a SOA involving nodes of diverse technological origin. Their interactions are synthesized in the proposed approach and illustrated in Fig. 2.3-12. The concept's reliability and efficiency were evaluated through a data buffering mechanism and a signal generator for the distributed heartbeat event. Both mechanisms provide clear insights into communication infrastructure behavior under real-time constraints and highlight the network's impact on message delivery at high recurrence rates. The developed buffering process is detailed architecturally in Fig. 2.3-13.

Each node's activity was tested individually, confirming high interoperability even in complex scenarios. The multithreaded design proved efficient and feasible across three publish–subscribe technologies, enabling automated cyclic data delivery. Reliability and efficiency are further demonstrated through the success rate results of the data buffering mechanism, presented in Fig. 2.3-14.

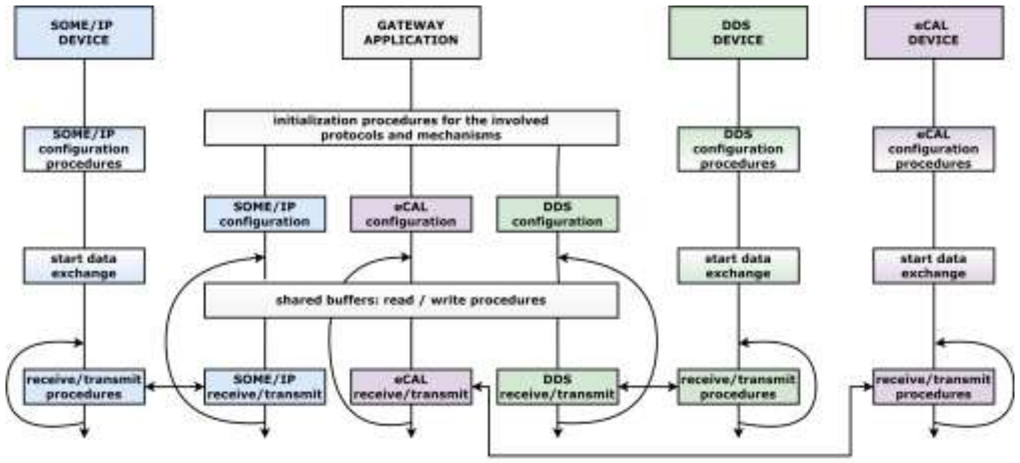

SOME/IP DEVICE
GATEWAY APPLICATION
DDS DEVICE
eCAL DEVICE
SOME/IP configuration procedures
initialization procedures for the involved protocols and mechanisms
DDS configuration procedures
eCAL configuration procedures
SOME/IP configuration
eCAL configuration
DDS configuration
start data exchange
shared buffers: read / write procedures
receive/transmit procedures
SOME/IP receive/transmit
eCAL receive/transmit
DDS receive/transmit

Fig. 2.3-12 Procedures Sequence for all nodes of the architecture.

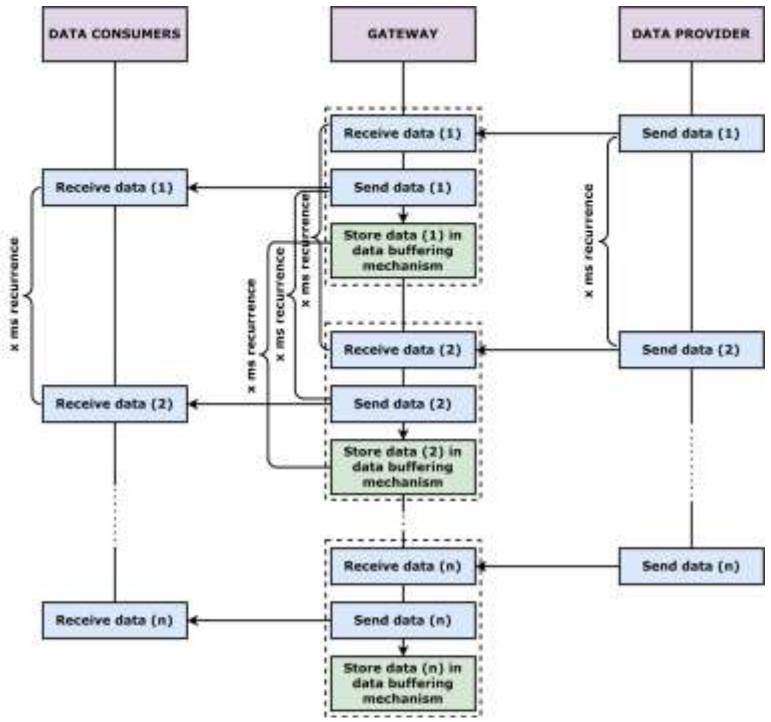


Fig. 2.3-13 Data buffering sequence.

The multi-protocol gateway successfully interfaced SOME/IP, DDS, and eCAL entities, meeting efficiency and reliability criteria while mapping the concept to a V2X communication scenario. The defined architecture supported interaction among three communication technologies across two case studies, each based on distinct gateway versions with specific characteristics. Compatibility among SOME/IP, DDS, and eCAL was confirmed, providing perspectives across industrial domains and highlighting protocol-specific advantages and limitations, as illustrated in Table 2-6.

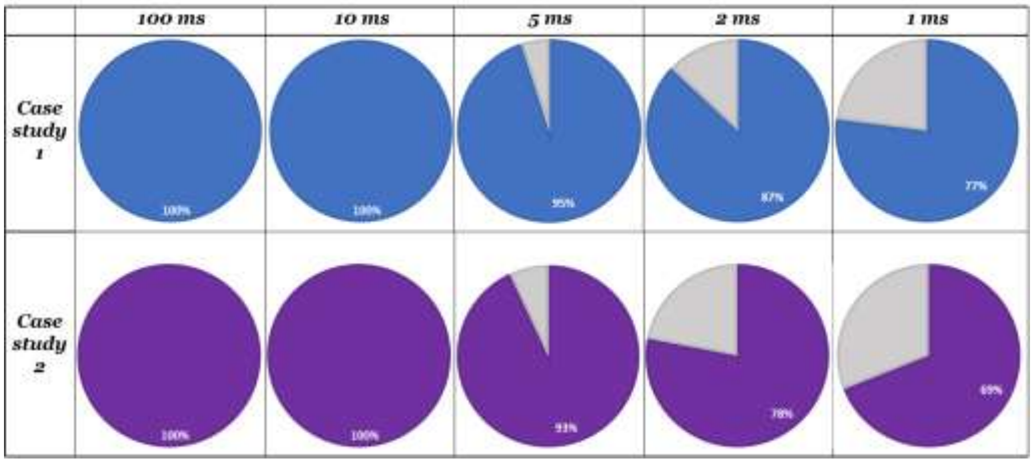


Fig. 2.3-14 Data buffering success rate results.

Table 2-6 Advantages and disadvantages related to SOME/IP, DDS and eCAL

Protocol	Advantages	Disadvantages
SOME/IP	• AUTOSAR compliant • Validated in multiple automotive use-cases • Supported on both Classic and Adaptive platforms • Reliable and efficient	• Complex configuration process
DDS	• AUTOSAR compliant • Offers multiple mechanisms that assure flexibility and scalability • Supported on Adaptive platform • Validated in multiple IoT applications and use-cases	• Not established in the automotive domain, despite being AUTOSAR compliant
eCAL	• Efficient, intuitive and easy to use for Ethernet communication scenarios • Easy configuration process • High potential for industrial and automotive related use-cases	• Not AUTOSAR compliant for now • Not very known • Not applied to full potential in explicit technical areas

2.3.2 Zenoh Approach in the Automotive Sector

Modern vehicles rely on distributed controllers that communicate in real time to ensure safety and efficiency. A key trend is the shift toward zonal architectures, where local controllers manage specific functions and report to an in-vehicle server. This reduces cabling complexity and improves integration of power and communication infrastructures, but also imposes stricter requirements on middleware and software platforms [82]. Increasing demands for real-time monitoring and cloud connectivity further highlight the need for efficient communication protocols, whose selection directly impacts system performance under automotive real-time constraints. Case-study validation has therefore become essential for assessing middleware adaptability across diverse automotive scenarios.

The transition to software-defined vehicles has accelerated the adoption of modular, scalable, and connected embedded architectures [83]. Traditional ECUs are being replaced by zonal controllers that manage sensors and actuators for specific domains (e.g., body, chassis, powertrain) and report to centralized servers [84-85]. Modern vehicles may integrate up to 150 ECUs communicating via in-vehicle networks [86]. In-vehicle servers enable advanced features such as external communication, OTA updates, and enhanced safety [87]. Reliable protocols are crucial for V2I integration, particularly in edge/cloud contexts. For example, [88] proposes a V2X enabled system architecture for accident detection and real-time data analysis.

Building on this motivation, two publish/subscribe protocols, Zenoh and DDS, are compared for real-time data delivery between in-vehicle servers and the cloud. Zenoh unifies data in motion, data at rest, and computation under a single protocol, offering location transparency, geo-distributed storage, and query-based access [89]. Its support for peer-to-peer, brokered, and routed topologies provides flexibility for complex embedded deployments [90], while its lightweight design avoids reliance on broker-based architectures [91]. Studies confirm Zenoh's suitability in automotive simulations [92], live migration of edge applications [93], SDN coordination [94], autonomous vehicle dataflows [95], and IoT surveillance [96]. Comparative analyses show Zenoh achieving lower latency and higher throughput than DDS, MQTT, and Kafka, particularly in constrained networks [97-99].

DDS, however, remains a benchmark protocol in industrial and embedded systems. Its specification includes the Data-Centric Publish-Subscribe (DCPS) layer and the optional Data Local Reconstruction Layer (DLRL), with 22 configurable QoS parameters enabling adaptability to diverse applications [100]. DDS generally outperforms alternatives such as MQTT, ZeroMQ, and AMQP in scenarios with large messages, strict real-time requirements, or many subscribers [100].

The primary aim of [K-2] was to assess whether emerging frameworks such as Zenoh can satisfy the stringent requirements of the automotive industry, particularly regarding cloud data transmission under varying time constraints.

A Zenoh-based communication architecture was implemented and benchmarked against DDS in a representative scenario, serving as an initial step toward designing adaptable, high-performance automotive middleware. The specific objectives of [K-2] were to:

- Employ distributed embedded equipment typical of automotive systems, including two zone controllers and an in-vehicle server.
- Implement a Zenoh-based prototype within a service-oriented system, a zonal control use case.
- Collect real-time control and execution data from zone controllers via the in-vehicle server and forward it to the cloud using Zenoh and DDS for comparison.
- Evaluate communication reliability at multiple publishing intervals (1000 ms, 100 ms, 10 ms, 5 ms, 1 ms) to test Zenoh under increasing speed and timing constraints.
- Highlight Zenoh's strengths over DDS in terms of data consistency, simplicity, and suitability for embedded automotive environments.
- Derive practical insights on Zenoh's configuration and adaptation for automotive-grade communication, establishing a basis for future improvements and extended testing.

2.3.2.1 Architectural Approach

The study focuses on the in-vehicle server–cloud connection, enabling real-time data monitoring and analysis. Communication is implemented with Zenoh (primary focus) and DDS, both tested under identical conditions. The system comprises two microcontrollers (XMC4500, STM32L476RG) functioning as zone-controller ECUs and a Raspberry Pi 4 serving as the in-vehicle server (see the vehicle architecture in Fig. 2.3-15).

From a hardware perspective, two potentiometers are connected to the XMC4500 zone controller: one adjusts direction (left/right), the other controls speed (high/low). This microcontroller serves as the decision-making node, coordinating subsequent operations. The STM32L476RG zone controller acts as the execution node, driving two stepper motors that simulate physical responses to control signals; in real implementations, these could be replaced by actuators or other components. The in-vehicle server processes data streams in real time, interfacing with the XMC4500 via two input channels and acquiring state information from the STM32L476RG through four channels. Beyond coordinating the controllers, the server enables V2X communication and cloud integration. External communication is handled through Zenoh and DDS, both publish/subscribe protocols with identical hardware.

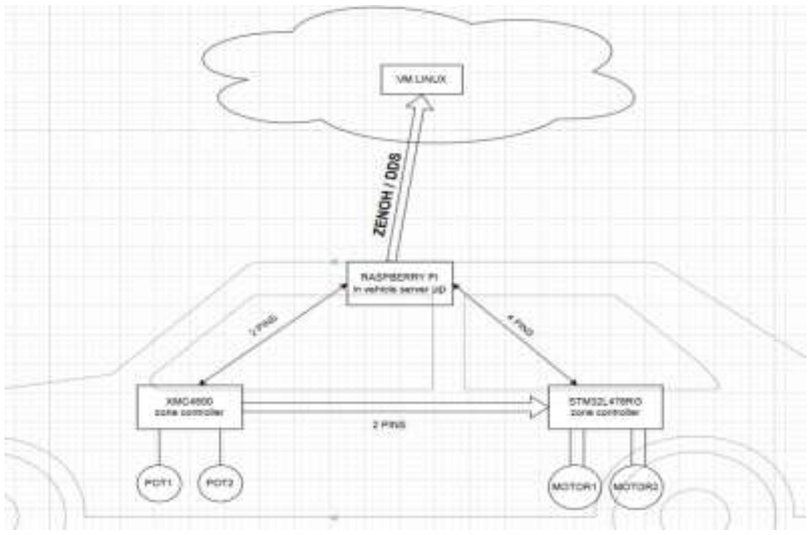


Fig. 2.3-15 General architecture for the connection between the in-vehicle server and the cloud via ZENOH/DDS.

By aggregating data from control and execution nodes, the server provides a real-time system overview, forwarding information to the cloud for monitoring and analysis. Testing under generic, non-ideal infrastructure reflects real-world conditions, exposing potential limitations and edge-case behaviors. Communication flow is evaluated by comparing messages sent by the server with those received by the cloud, revealing how performance degrades under high-frequency transmission while maintaining stable operation. This assessment highlights the robustness and flexibility of the architecture under strict timing and imperfect conditions.

2.3.2.2 Case-Study Development and Results

The testing process involved transmitting a predefined number of messages from the Raspberry Pi to the cloud server at recurrence intervals of 1000 ms, 100 ms, 10 ms, 5 ms, and 1 ms. For each interval, received messages were compared to those sent. Additional scenarios measured latency and jitter for both protocols at every interval. Two fault-injection tests were also conducted to assess protocol resilience under degraded network conditions and to analyze their impact on message delivery rates. Table 2-7 summarizes the key implementation parameters of the system, including software versions, operating system details, socket buffer configurations, and QoS settings.

Table 2-7 System configuration details

Parameter	Configuration
Protocol version	zenoh-c 1.0.0-dev-208-gaab2487 (for Zenoh) Fast-DDS 2.14.0 (Fast RTPS), Fast-CDR 2.2.1 (for DDS)

Operating System	Raspbian GNU/Linux 12 (Bookworm) Ubuntu 20.04.6 LTS
Kernel version	6.6.51+rpt-rpi-v8 (for Raspberry Pi) 5.15.0-139-generic (for Linux VM)
Socket buffer sizes	rmem_max = 212992 wmem_max = 212992
QoS for DDS	RELIABLE_RELIABILITY_QOS, TRANSIENT_LOCAL_DURABILITY_QOS (for Publisher), VOLATILE_DURABILITY_QOS (for Subscriber)
Serialization	Raw byte payload - for Zenoh, CDR (Common Data Representation) – for DDS

Both DDS and Zenoh implementations were developed. This section outlines the Zenoh-based approach. The communication layer was set to be Zenoh to evaluate real-time data transmission from the in-vehicle server (Raspberry Pi) to the cloud (Ubuntu VM), focusing on reliability and timing across varying publication intervals. Zenoh's topic-based publish/subscribe mechanism resembles DDS but differs in syntax and configuration. DDS defines topics via IDL files, while Zenoh employs lightweight key expressions (e.g. */key/topicForMonitoring*) for dynamic data access. The data structure, an unsigned long timestamp and a string message, remains unchanged but is managed through Zenoh's byte-oriented APIs. Implementation relied on the open-source zenoh-c library [101], with design references from official documentation [102]. Fig. 2.3-16 shows topic-level data flow and communication structure between *MyPublisher1* and *MySubscriber1*.

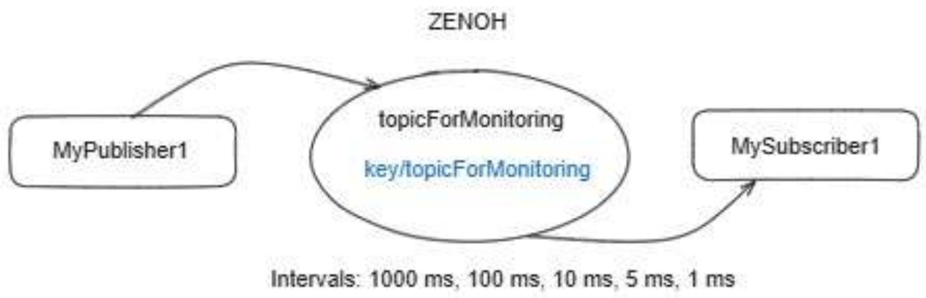


Fig. 2.3-16 Zenoh communication flow.

Although DDS and Zenoh both employ the publish/subscribe model on identical hardware, their implementations differ markedly in configuration complexity, code structure, and resource management. In DDS, topic definition requires an IDL file describing data structures, compiled with tools such as Fast DDS-Gen to generate Publisher and Subscriber code. This introduces dependencies, external tooling, and higher setup effort. DDS also demands careful configuration of participants, domain IDs, data types, *DataWriters*, and *DataReaders*. Zenoh, by contrast, offers a lightweight approach. Topics are replaced with flexible key expressions, and data is handled as byte buffers, leaving encoding/decoding to the application. Functions such as *z_put()* and *z_declare_subscriber()* simplify publishing and subscribing through callbacks, avoiding rigid participant and type registration. Memory management is streamlined via owned and loaned data types (*z_owned_bytes_t, z_loaned_sample_t*), reducing allocation overhead compared to DDS's typed system.

Integration is also easier: Zenoh relies on the zenoh-c library and a simple CMake setup, without IDL generation or multiple dependencies. Unlike DDS, it requires no domain IDs or static participant setups, making replication and modification straightforward. Both implementations used the same payload (message + timestamp), but serialization differed. DDS applied the CDR standard, ensuring type safety but adding overhead, while Zenoh transmitted raw byte arrays, improving efficiency at high publishing frequencies. Zenoh further simplified debugging and monitoring, with lightweight logic based on key expressions, particularly advantageous on resource-constrained devices like the Raspberry Pi. While DDS remains robust for large-scale, type-safe systems, Zenoh's lightweight design proved better suited for real-time experimentation, prototyping, edge-to-cloud integration in this scenario.

Performance evaluation involved testing both protocols under 5 recurrence intervals. Each 1-minute test published thousands of messages, enabling detailed statistical analysis of reliability under varying load conditions. The results are summarized in Fig. 2.3-17 and Fig. 2.3-18.

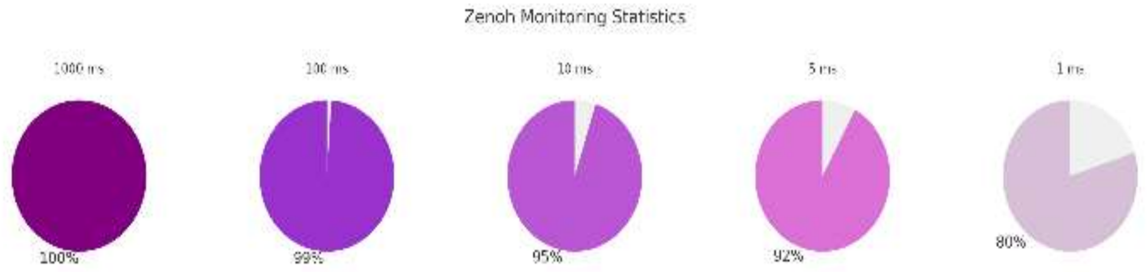


Fig. 2.3-17 Zenoh monitoring statistics.

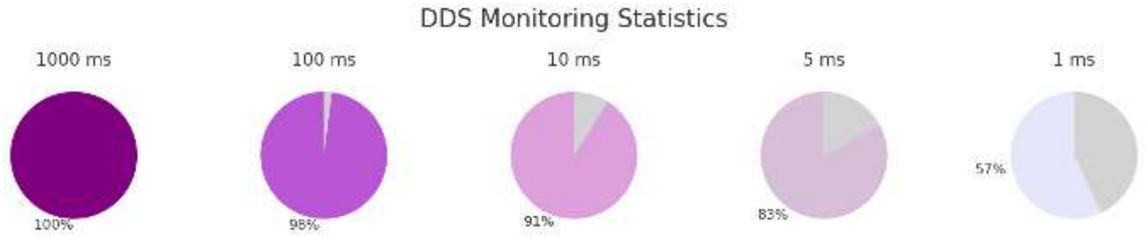


Fig. 2.3-18 DDS monitoring statistics.

At 100 ms recurrence, Zenoh achieved a 99.22% success rate, slightly outperforming DDS at 98.24%. At 10 ms, the gap widened: Zenoh delivered

95.13%, while DDS reached 90.80%. These differences reflect the influence of operating system scheduling and buffering in non-real-time environments. At 5 ms, Zenoh maintained 91.79%, compared to DDS at 83.09%. At 1 ms, DDS dropped to 57.12%, whereas Zenoh sustained 79.93%. Although both protocols decline at high publishing rates, Zenoh's resilience highlights its suitability for high-throughput, low-latency edge computing scenarios.

A dedicated test scenario further measured end-to-end latency and jitter across all intervals. Timestamped data enabled calculation of latency (publication–reception difference) and jitter (variation from expected intervals). Results, detailed in [K-2], showed Zenoh outperforming DDS. For example (see Fig. 2.3-20), at 10 ms recurrence, Zenoh achieved an average latency of 149.69 ms and jitter of 0.92 ms, indicating stable scheduling under high-frequency transmissions. DDS, by contrast, recorded 163.23 ms latency and 1.74 ms jitter, with greater fluctuation and reduced consistency.

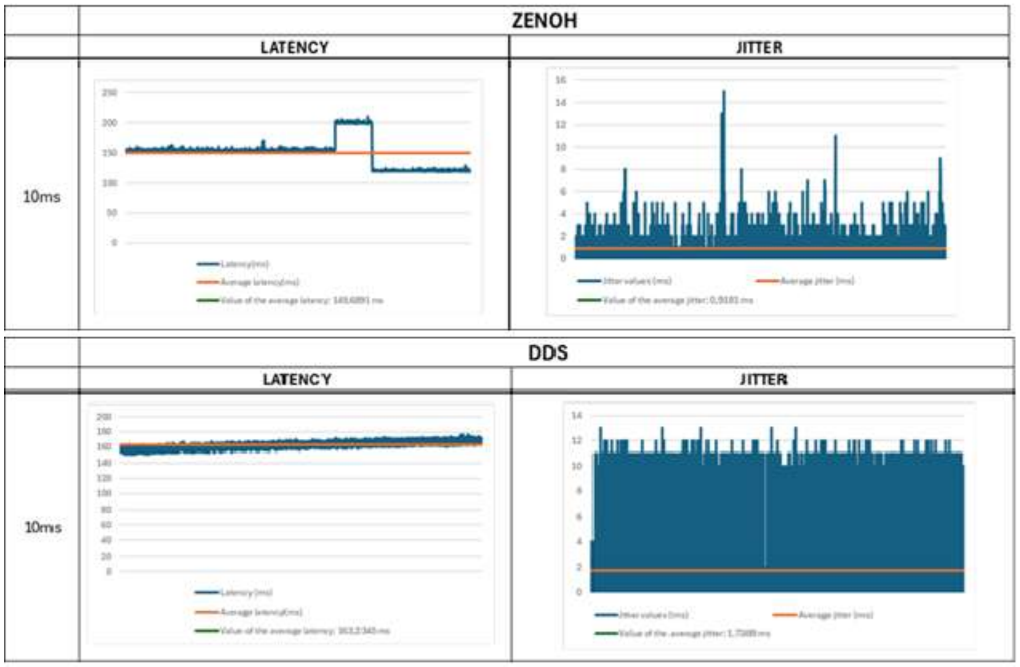


Fig. 2.3-19 Zenoh-DDS comparison in fault-injection scenarios.

To evaluate behavior under non-ideal conditions, two fault-injection scenarios were introduced: simulated packet loss and artificial network delay. These reflect common issues in wireless embedded or congested in-vehicle networks, where strict timing can be disrupted. In the delay scenario, the subscriber (Linux VM) was configured with 200 ms ± 50 ms delay, while the publisher (Raspberry Pi) had 100 ms ± 20 ms delay. In the packet-loss scenario, a 5% loss rate was applied at the publisher's interface to emulate random transmission failures. Tests results are shown in Fig. 2.3-20.

In the artificial delay scenario, Zenoh maintained: 100% delivery at 1 s, 91.51% at 100 ms, and 84.59% at 10 ms. At higher frequencies, rates declined to 80.84% (5 ms) and 52.75% (1 ms), yet Zenoh outperformed DDS. DDS remained acceptable at 1 s (100%) and 100 ms (92.64%), but dropped sharply to 47.73% at 10 ms, 32.39% at 5 ms, and 10.23% at 1 ms, indicating limited suitability under added latency for high-frequency transmissions. In the packet-loss scenario, Zenoh showed resilience, sustaining 100% at 1s, 98.20% at 100 ms, and 92.63% at 10 ms. Performance decreased to 90.42% at 5 ms and 64.02% at 1 ms. DDS was more sensitive, with 95.18% at 1000 ms and 93.33% at 100 ms, but falling to 78.04% at 10 ms, 67.53% at 5 ms, and 32.97% at 1 ms.

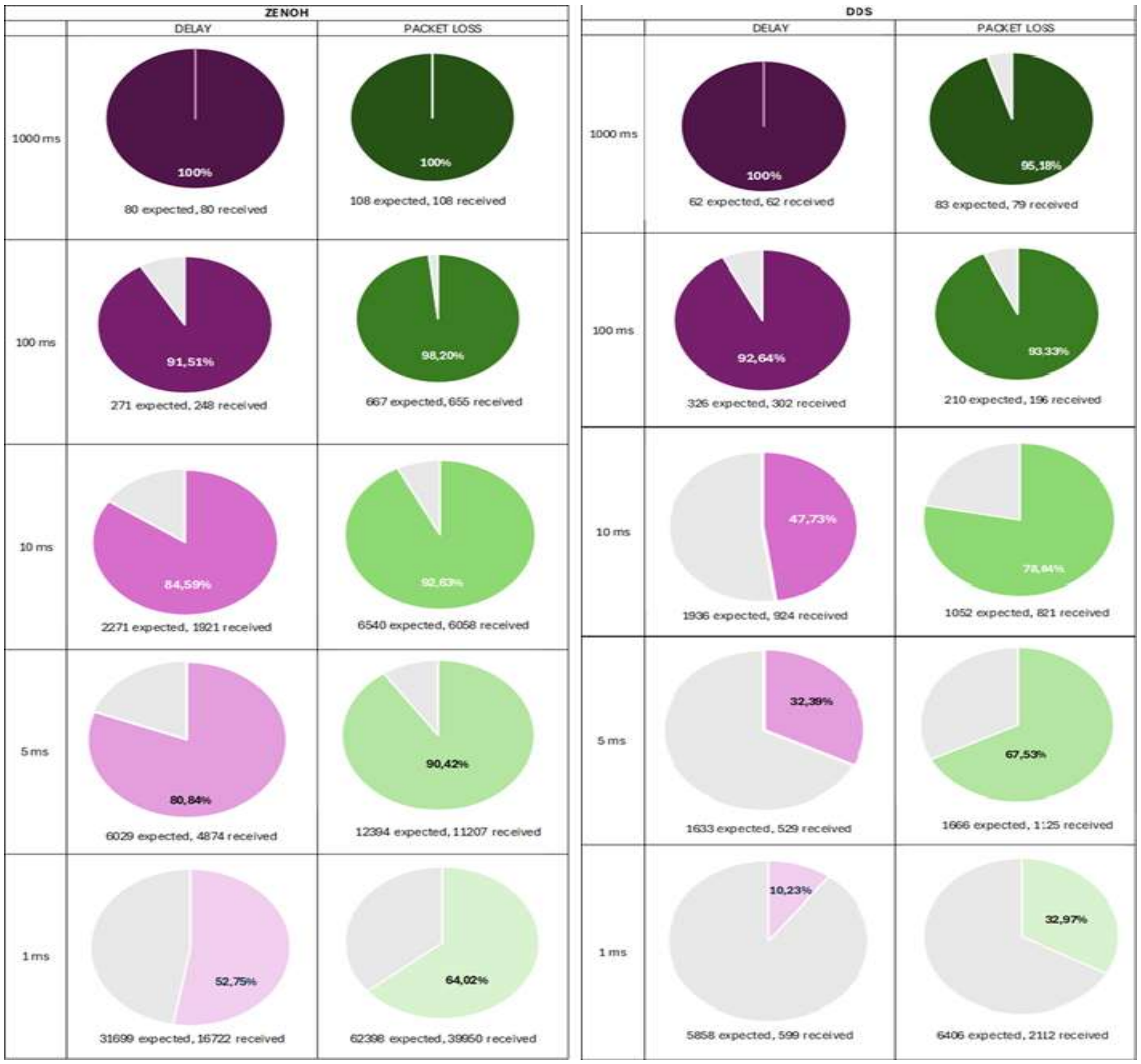


Fig. 2.3-20 Zenoh-DDS comparison in fault-injection scenarios.

3 Approaching New Technologies and Solutions in Supervisory Control and Data Acquisition

The current chapter consists of information from 6 scientific works [K-8], [K-10], [K-22], [K-34], [K-35] [K-39], all WoS indexed papers, within 3 conference proceedings and 3 Q2 journals. The current chapter presents 3 different industrial paths as follows:

- The IGSS SCADA environment that is the only environment with object-based licensing. The published papers approached two directions that were stringent in the corresponding period, namely the optimal resource allocation for IGSS [K-39], and very briefly the web access possibility [K-34].
- The mobile Android SCADA concept that is independent of a main SCADA environment, and that has OPC UA interfacing. Two papers were published regarding the Android SCADA conceived and developed solution. The first was based on OPC UA Client-Server mechanism and a basic diagram approach [K-35], and the second highly improved solution that was validated in the water industry and included modules as Alarm&Events OPC UA service, tag structuring, elaborated design and deployment services, etc. [K-10].
- The Node-RED based SCADA, relying on Node-RED open-source environment. The Node-RED SCADA research resulted in two published papers in Q2 journals, the first [K-21] as a generic solution, and the second [K-8] more complex application that was validated in a building management system real scenario.

3.1 IGSS related Advancements in Efficiency.

As the IGSS SCADA environment popularity was increasing due to the object-based licensing, reporting module, early OPC interfacing, graphics and general structuring, many industries used it in implementations both as first level and as regional/central control centers.

Work [K-39] proposed an optimization of IGSS SCADA resources for integrating wastewater and drinking water pumping stations (WWPS/PS) into higher-level SCADA systems. Local automation solutions vary widely due to differences in equipment, generations, tender specifications, and integrator practices, limiting the possibility of modifications. Thus, maximizing SCADA software capabilities is essential to reduce costs. In IGSS, licensing is object-based, with types such as analog, digital, table, and counter, each defined by preconfigured atoms. Atoms differ by role (data type, I/O, alarm) and may require templates for proper use. Traditionally, each IGSS object corresponds to a physical device (e.g., sensor, pump, mixer) and its main atom (e.g., *ActualValue* for analog, *State* for digital) represents the key operational parameter. Alarming and reporting strategies are built around these atoms, while additional atoms (e.g., *HighAlarm, LowAlarm, AlarmIn*) extend functionality. To fully exploit alarming in higher-level SCADA, optimizations are needed since local PLCs and SCADA systems already generate alarms in diverse formats, requiring template structures to access more bits from available tags.

FreeValue atoms, designed to map analog values, are underutilized in classical implementations, as they cannot trigger alarms directly or appear in standard reports. For example, a device with three alarm states (overheating, overcurrent, leakage) would require three IGSS objects, since *FreeValue* atoms are excluded from standard reporting. This approach consumes excessive resources, leading to larger license packages and higher costs when scaled to higher-level SCADA.

In the water sector, locally implemented automation solutions lack a unified classification methodology for integration. The provided tags are central to IGSS optimization, with three types identified in OPC DA/UA implementations:

- Simple tags: single values linked to local variables (e.g., pump state, level).
- Composed tags: sets of digital values represented by word variables, with bits encoding operational states or faults.
- Multiplexed tags: values varying across sample times.

Based on these structures, WWPSs were classified into four categories:

- Type 1: simple and composed tags, including pump states/faults, emergency signals, intrusion detection, and gas monitoring.
- Type 2: only simple tags.
- Type 3: simple and multiplexed tags, mainly analog.
- Type 4: simple and condensed composed tags, with pumps controlled via frequency converters and detailed fault detection.

Similarly, PSs were grouped into three categories:

- Type 1: simple and composed tags, covering pump and valve states/faults.
- Type 2: simple and multiplexed tags.
- Type 3: only simple tags.

IGSS enables optimized implementations that reduce object requirements and implicitly licensing costs. Resource optimization involves defining objects, mapping atoms, configuring alarms, and reporting systems. These strategies were aligned with operational needs of a water distribution company, ensuring operators had access to relevant tags and SCADA functionalities available.

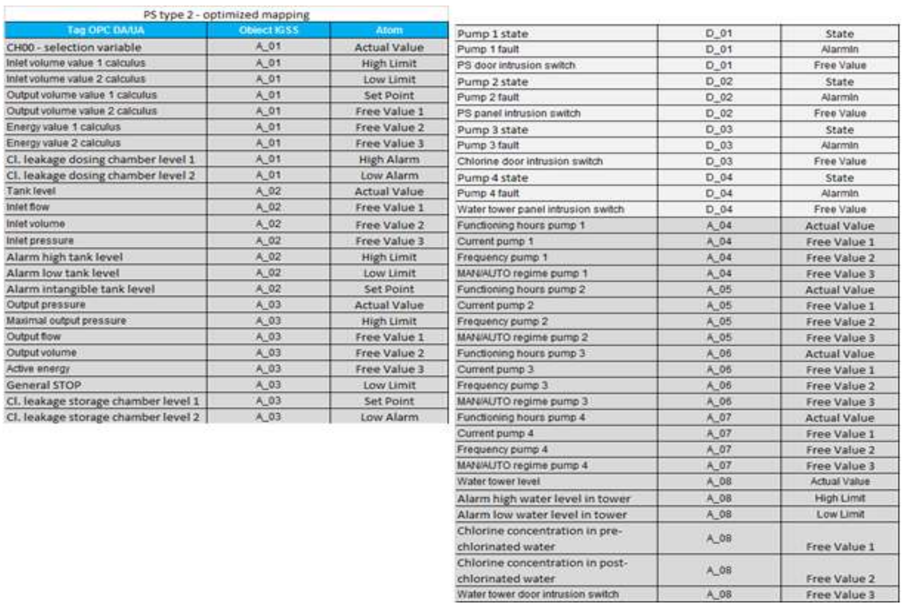
Extracting information from composed tags requires additional processing in SCADA. Two approaches were proposed:

- Templates with Calculation Module: Template structures are configured for Bit Map I/O (to expand states), Alarm In/Ack bits (to extend alarms), and State/Commands (to define states). Alarms can be mapped to the *State* atom of digital objects, with single-bit display options and individual descriptors for each state. The calculation module applies masks to identify or group bits, mainly for alarm/state delimitation. This method was used for WWPS type 4, where large digital data was in composed tags as pump/valve states/faults.
- VBA Implementations: VBA code identifies individual bits of composed tags and assigns graphical descriptors in synoptic schemes. This generic approach is effective across IGSS due to its modular design. Multiplexed tags are better suited to VBA, as their values change per sampling period. *FreeValue* atoms can be repurposed for counters, linked to alarms through scripting, enabling alarms to be triggered. Alarm texts are created and associated with objects, while conditions are managed via code. Visual descriptors ensure diagram visualization, and alarm management interfaces allow acknowledgment and clearing directly in SCADA diagrams.

For reporting, standard IGSS reports focus on base class values and lack coverage for all atoms. Optimization requires archiving and reporting of every atom. IGSS supports extended logging with MySQL or SQLite, solving storage and access issues. Custom Excel-based reports provide the only way to represent diverse atom types, executed similarly to standard reports. The mapping was designed to maximize resource utilization for each WWPS and PS type. IGSS objects were defined with a high number of atoms, ensuring proper correspondence across all IGSS modules. An illustrative example of object mapping is provided in Fig. 3.1-1 for a type 2 PS.

After IGSS resource optimization, WWPS integration required an average of 7 objects per station, while PS integration used 11–13 objects. The optimization magnitude varied by station type: classical SCADA implementations for WWPS types 1–3 typically required 22–26 objects, and type 4 up to 49. For PSs, classical solutions averaged 40 objects for type 1 and 51 for types 2–3.

All monitoring and control diagrams operated correctly, with examples shown in Fig. 3.1-2 (type 1 WWPS) and 3.1-3 (type 2 PS). Core SCADA functions, including alarming, reporting, logging, archiving, and mobile access, were also verified as fully functional. Resource optimization was not fully maximized. Objects were grouped with attention to alarm indications in diagrams, integrated graphics, predefined measurement unit associations, and mobile module functionality.



PS type 2 - optimized mapping

Tag OPC DA/UA	Object IGSS	Atom
CH00 - selection variable	A_01	Actual Value
Inlet volume value 1 calculus	A_01	High Limit
Inlet volume value 2 calculus	A_01	Low Limit
Output volume value 1 calculus	A_01	Set Point
Output volume value 2 calculus	A_01	Free Value 1
Energy value 1 calculus	A_01	Free Value 2
Energy value 2 calculus	A_01	Free Value 3
Cl. leakage dosing chamber level 1	A_01	High Alarm
Cl. leakage dosing chamber level 2	A_01	Low Alarm
Tank level	A_02	Actual Value
Inlet flow	A_02	Free Value 1
Inlet volume	A_02	Free Value 2
Inlet pressure	A_02	Free Value 3
Alarm high tank level	A_02	High Limit
Alarm low tank level	A_02	Low Limit
Alarm intangible tank level	A_02	Set Point
Output pressure	A_03	Actual Value
Maximal output pressure	A_03	High Limit
Output flow	A_03	Free Value 1
Output volume	A_03	Free Value 2
Active energy	A_03	Free Value 3
General STOP	A_03	Low Limit
Cl. leakage storage chamber level 1	A_03	Set Point
Cl. leakage storage chamber level 2	A_03	Low Alarm
Pump 1 state	D_01	State
Pump 1 fault	D_01	AlarmIn
PS door intrusion switch	D_01	Free Value
Pump 2 state	D_02	State
Pump 2 fault	D_02	AlarmIn
PS panel intrusion switch	D_02	Free Value
Pump 3 state	D_03	State
Pump 3 fault	D_03	AlarmIn
Chlorine door intrusion switch	D_03	Free Value
Pump 4 state	D_04	State
Pump 4 fault	D_04	AlarmIn
Water tower panel intrusion switch	D_04	Free Value
Functioning hours pump 1	A_04	Actual Value
Current pump 1	A_04	Free Value 1
Frequency pump 1	A_04	Free Value 2
MAN/AUTO regime pump 1	A_04	Free Value 3
Functioning hours pump 2	A_05	Actual Value
Current pump 2	A_05	Free Value 1
Frequency pump 2	A_05	Free Value 2
MAN/AUTO regime pump 2	A_05	Free Value 3
Functioning hours pump 3	A_06	Actual Value
Current pump 3	A_06	Free Value 1
Frequency pump 3	A_06	Free Value 2
MAN/AUTO regime pump 3	A_06	Free Value 3
Functioning hours pump 4	A_07	Actual Value
Current pump 4	A_07	Free Value 1
Frequency pump 4	A_07	Free Value 2
MAN/AUTO regime pump 4	A_07	Free Value 3
Water tower level	A_08	Actual Value
Alarm high water level in tower	A_08	High Limit
Alarm low water level in tower	A_08	Low Limit
Chlorine concentration in pre-chlorinated water	A_08	Free Value 1
Chlorine concentration in post-chlorinated water	A_08	Free Value 2
Water tower door intrusion switch	A_08	Free Value 3

Fig. 3.1-1 PS type 2 object mapping example.

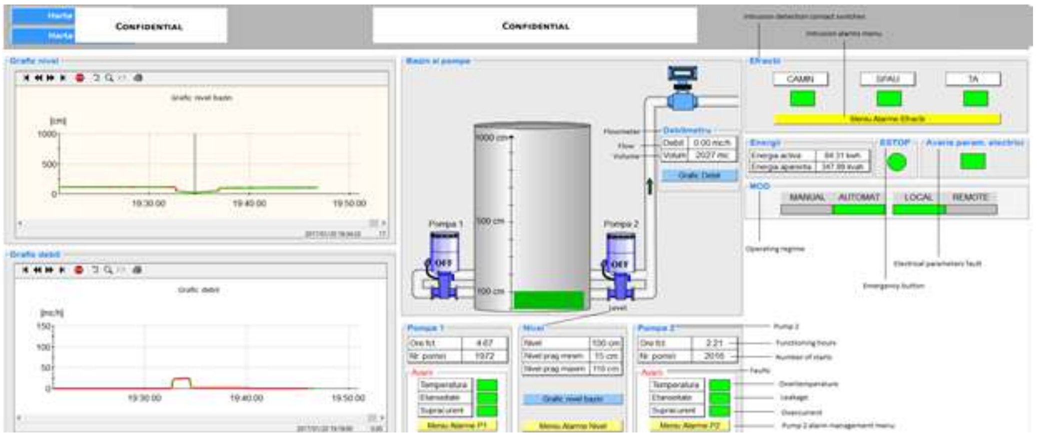
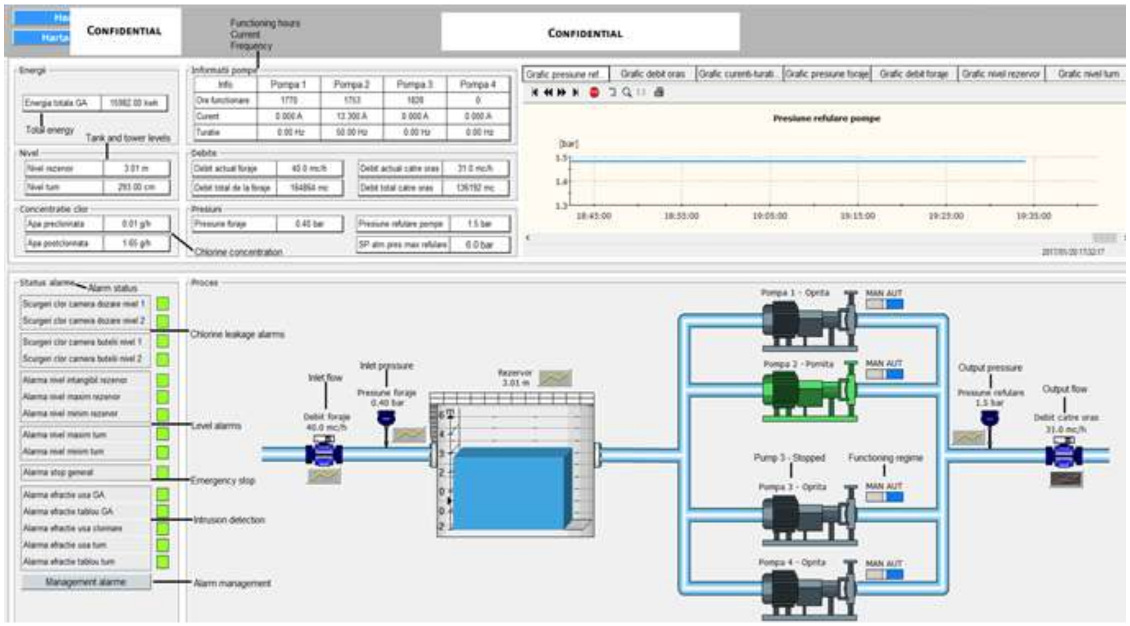


Fig. 3.1-2 WWPS type 1 IGSS diagram example using optimized resources (augmented in English).

IGSS provided limited web-based solutions for remote monitoring, including TeamViewer, LogMeIn, and an ActiveX browser client within the classical client-server setup. These approaches transferred the full graphical output of the IGSS server, requiring high-bandwidth networks. A dedicated web monitoring/control module was not prioritized, though development was possible via the ODBC server. Paper [K-34] introduced WebNavIGSS, a generalized web-based solution that leveraged existing SCADA structures in correlation with the Supervise module. Built around a webserver, WebNavIGSS enabled real-time data output from IGSS applications.



Info	Pompa 1	Pompa 2	Pompa 3	Pompa 4
Ore functionare	1770	1753	1828	0
Curent	0.000 A	13.300 A	0.000 A	0.000 A
Turatie	0.00 Hz	50.00 Hz	0.00 Hz	0.00 Hz

Energia totala GA	15982.00 kwh
Nivel rezervor	3.01 m
Nivel turn	293.00 cm
Apa preclorinata	0.01 g/h
Apa postclorinata	1.65 g/h
Debit actual foraje	40.0 mc/h
Debit total de la foraje	164864 mc
Debit actual catre oras	31.0 mc/h
Debit total catre oras	136192 mc
Presiune foraje	0.40 bar
Presiune refulare pompe	1.5 bar
SP alm pres max refulare	0.0 bar

Fig. 3.1-3 PS type 2 IGSS diagram example using optimized resources (augmented with explanations in English).

The concept relied on the IGSS ODBC server, enabling SQL-based access to configuration and process data. Through this interface, key tables as ALM (alarms), LOG (logs), and BCL (base class) were accessed. An SQL/MySQL server acted as a bridge, importing non-standard databases via ODBC and converting them into standard formats usable by higher-level applications. The web application, controlled by the webserver, processed data from the SQL/MySQL server and delivered real-time outputs in a user-friendly interface. Data transfer between the SQL server and webserver employed PHP with JavaScript, chosen over Java or C# for its speed, lower resource demands, scalability, open-source nature, and platform independence in dynamic web development. The main software components of WebNavIGSS and the proposed concept are shown in Fig. 3.1-4.

For monitoring, the webserver retrieved live data from IGSS databases—audittraildb (audit), logdb (logs), mntdb (maintenance), and hdmdb (historical). Modifying atom values via WebNavIGSS required bidirectional communication between the webserver, SQL server, and IGSS ODBC server. A Model-View-Controller (MVC) architecture was adopted, enabling modular separation of application areas and independent implementation, which proved advantageous in the IGSS context. The WebNavIGSS application is structured into modules, as shown in Fig. 3.1-5, each with defined tasks and interconnections to limit redesign or replacement impact.

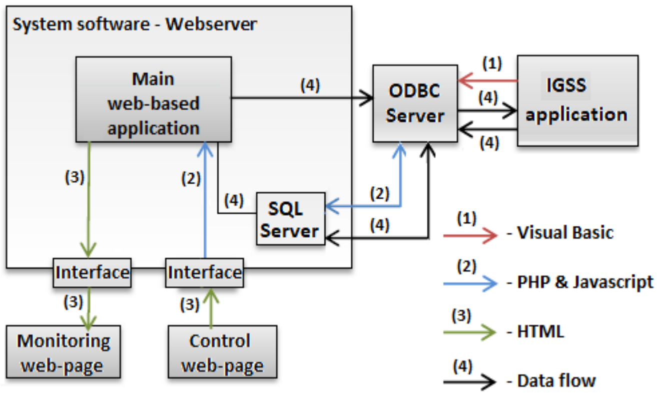


Fig. 3.1-4 Main software components of the proposed concept.

The Core Application (CoreApp) module functions as the operating system of the application, coordinating and integrating all modules. It employs priority-based task management to ensure user-critical actions (e.g., responding to queries) are processed before secondary tasks such as database imports. CoreApp also includes initialization sequences for all modules, error handling to diagnose faults and determine corrective paths, and exit routines to preserve data integrity during shutdown. The finite state machine governing CoreApp operation is shown in Fig. 3.1-6. Table 3-1 details the events, conditions and actions associated to an id number within the state machine.

The RealTimeHandler manages user communication and synchronizes values with IGSS SCADA using both asynchronous and synchronous tasks. Browser data updates occur every 100 ms, a rate imperceptible to human eye.

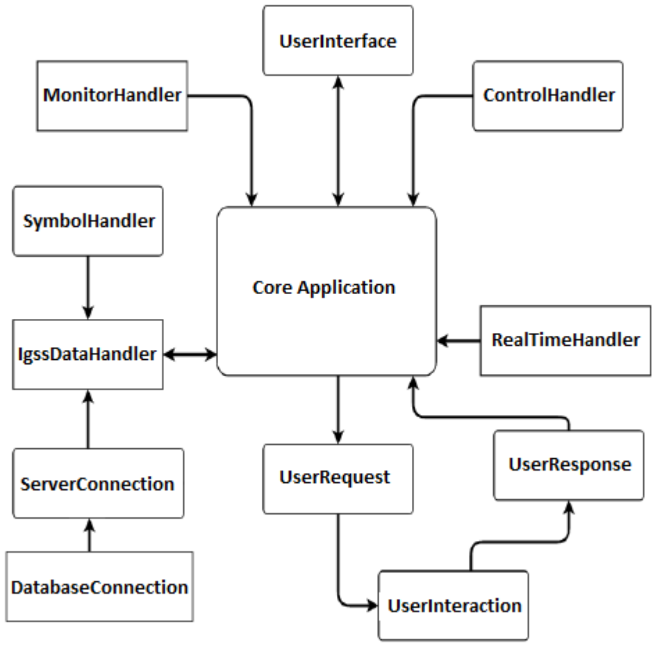


Fig. 3.1-5 WebNavIGSS application modules.

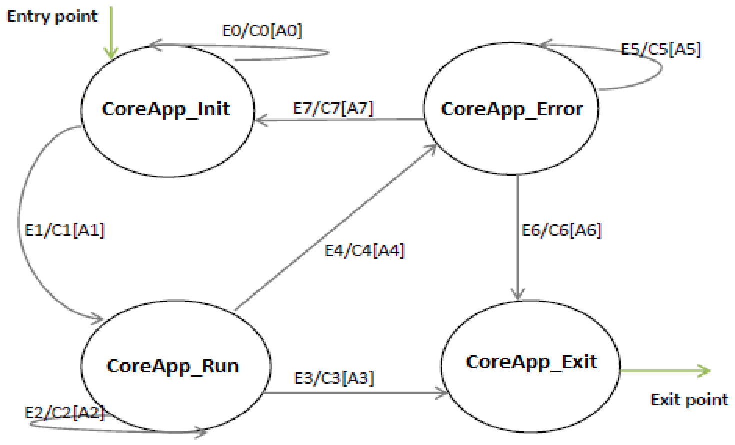


Fig. 3.1-6 Core Application module FSM.

Table 3-1 CoreApp FSM details

Id	Events (E)	Conditions (C)	Actions (A)
0	CoreApp not connected	Connection with other modules established? No.	CoreApp remains in same state
1	CoreApp initialized with other modules	All modules initialized? Yes.	CoreApp moves to Run state
2	CoreApp periodically checks application status	An error occurred? No.	CoreApp remains in same state
3	CoreApp received a request to exit the browser	A request was received to close the session? Yes.	CoreApp moves to Exit state
4	An error occurs while CoreApp is in run mode	An error was reported? Yes.	CoreApp moves to Error state
5	CoreApp is in error state and diagnosis is initiated.	Error diagnosis ready? No.	CoreApp remains in same state
6	CoreApp could not solve the problem.	CoreApp can solve the problem? No.	CoreApp moves to Exit state
7	CoreApp can solve the problem by reinitializing the module.	CoreApp can solve the problem? Yes.	CoreApp moves to Init state to initialize all modules

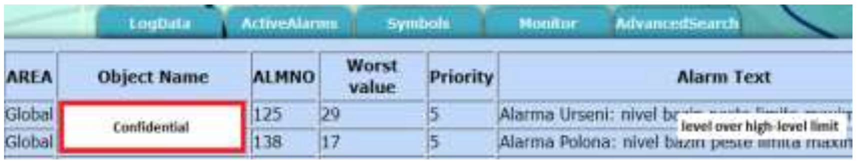
The IGSSDataHandler manages data exchange with IGSS, storing information in SQL/MySQL servers, handling databases, and executing queries. A ServerConnection ensures a unique, active SQL session, directing the DatabaseConnection module. Databases contain tables processed through implemented SQL and query scripts, including customized user views.

In the WebNavIGSS GUI, IGSS symbols, graphical descriptors of defined objects and their states, are imported via the SymbolHandler module. User management structures are also implemented through IGSSDataHandler, linking SCADA credentials with the web application.

A small auditing historian was implemented to record user activity within WebNavIGSS. In case of connection errors, the IGSSDataHandler attempts reconnection for 5 min. at 100 ms intervals. If unsuccessful, it halts the procedure and reports an error to CoreApp. The UserInterface module translates actions into graphical structures accessible via the webserver, using

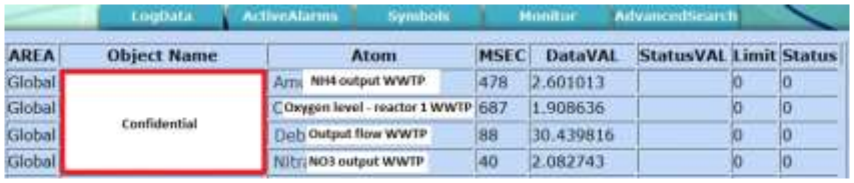
HTML and CSS. To ensure synchronization with CoreApp, communications are uniquely delimited, isolating the module from faults in others. The MonitorHandler establishes software connections for monitoring, similar to IGSS's Supervise module but with simplified data views. Users can filter information by area, diagram, object name, alarm status, or atom conditions, with search functions available. The ControlHandler manages control functionality, enabling modification of atom values within objects.

WebNavIGSS was validated both in laboratory conditions and in a real deployment at a regional IGSS SCADA control center of a water distribution company. The IGSS application encompassed water and wastewater facilities. Two screenshots illustrate the web navigator in operation, supplemented with English annotations. Fig. 3.1-7 illustrates active IGSS alarms in WebNavIGSS, highlighting two WWPS where levels exceeded the high limit. Fig. 3.1-8 presents a filtered view from WWTP objects, two nitrogen output values, one oxygen value from the biological reactor, and the output flow.



AREA	Object Name	ALMNO	Worst value	Priority	Alarm Text
Global	Confidential	125	29	5	Alarma Urseni: nivel b[illegible] level over high-level limit
Global		138	17	5	Alarma Polona: nivel bazin peste limita maxi

Fig. 3.1-7 WebNavIGSS screenshot of some active alarms.



AREA	Object Name	Atom	MSEC	DataVAL	StatusVAL	Limit	Status
Global	Confidential	Am NH4 output WWTP	478	2.601013		0	0
Global		C Oxygen level - reactor 1 WWTP	687	1.908636		0	0
Global		Deb Output flow WWTP	88	30.439816		0	0
Global		Nitr NO3 output WWTP	40	2.082743		0	0

Fig. 3.1-8 WebNavIGSS screenshot of some live values.

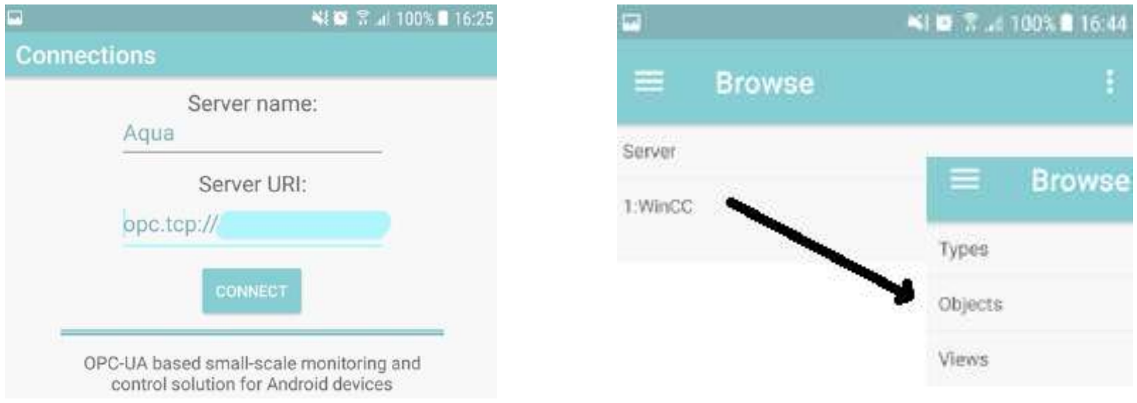
3.2 Android and OPC UA based Mobile SCADA Solution.

Accessing SCADA applications from mobile devices became a necessity in several industries. Most of the mobile solutions are based on traditional SCADA environments extensions, needing SCADA server applications in a control room. Independent SCADA solution was necessary, that is able to make use of Industry 4.0 improvements in interfacing and to be based on OPC UA protocol. First, in [K-35], a basic Android SCADA was conceived and developed solution, based on OPC UA Client-Server mechanism. The application was validated through a case study at a water treatment facility comprising a treatment plant, distant wells, pumping stations, reservoirs, and a water tower. Local automation/SCADA involved two separate solutions from different manufacturers under distinct contracts, with OPC serving as the interface. The treatment plant control room operated on two redundant SCADA servers using WinCC 7.2 with Connectivity packs exposing OPC UA servers.

The study focused on facility operation and maintenance, while the second contractor's performance was observed during implementation of wells and distribution. An Android UA client could have reduced implementation time and costs by addressing inefficiencies such as personnel confinement to monitoring sites, delays from failed tests, limited process visibility, and restricted SCADA control room access. Additionally, the treatment plant contractor incurred significant effort traveling between the control room and equipment during testing.

The following figures display the initial Android SCADA version tested in the case study. Fig. 3.2-1 depicts the OPC UA connection and a folder browsing. Fig. 3.2-2 presents a tag browsing and selection, respectively a subscription generation. In the subscription module, operators view basic variable information, including the UA server node identifier (ns, s) and tag value. As shown in Fig. 3.2-3, variables can be selected and customized via a popup window, allowing assignment of a title (e.g., output flow, well level, pump state, pressure, current, valve opening, with units) and an image for easier device identification. An updatable picture list was introduced for common equipment in the water facility. The subscription interface displays variables in full setup. For example, a WTP flow with a value of 34.53111 m³/h.

Several protection structures were implemented to guide operators when incorrect commands are issued (e.g., invalid subscriptions) and to maintain continuous system status awareness. The objective was to create a user-friendly application requiring minimal operator expertise.


Connections

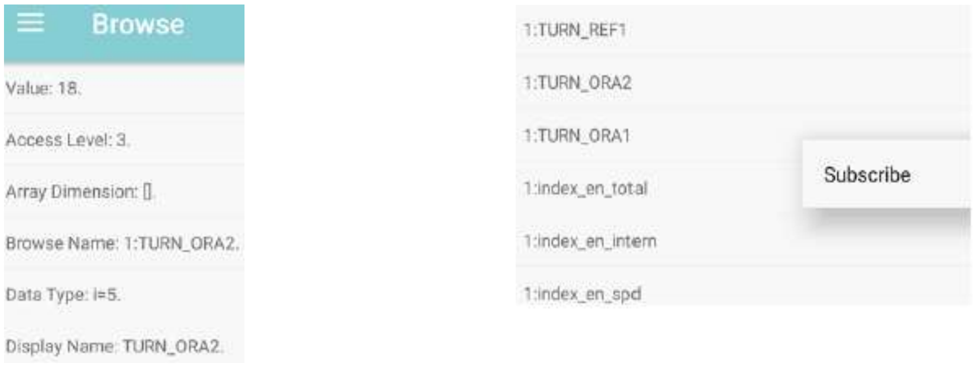
Server name:
Aqua

Server URI:
opc.tcp://[redacted]

OPC-UA based small-scale monitoring and control solution for Android devices

Fig. 3.2-1 Connecting to the OPC UA Server and folder browsing.

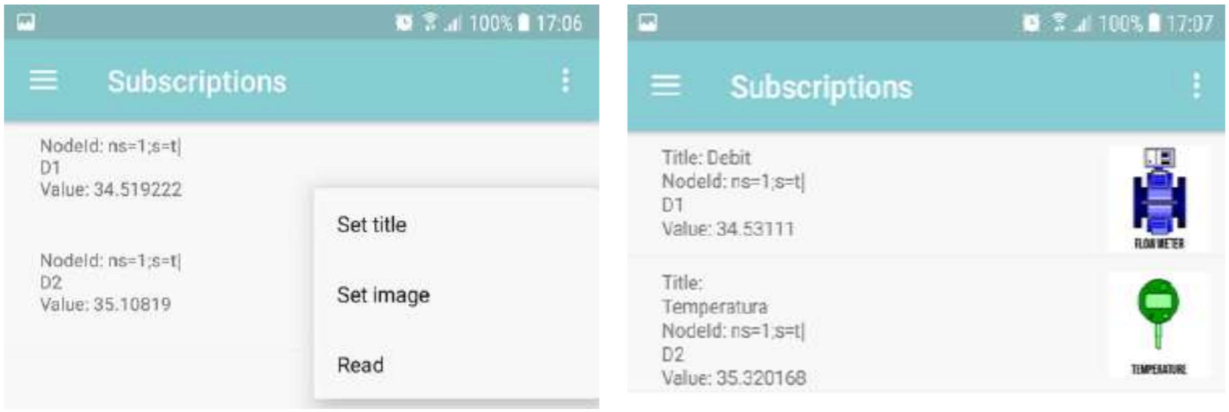
Following the initial small-scale application, research advanced toward a more complex prototype system for the water industry, as reported in [K-10]. Starting the application initiates a new Linux process with a Main thread responsible for all user interface (UI) interactions. In the first version, this thread also managed OPC UA server connections. However, newer SDKs prohibit networking on the main thread, generating a NetworkOnMainThreadException. The updated application therefore integrated multiple Background threads to handle network operations, including OPC UA server connections, Node ID subscriptions, and client disconnections, as illustrated in Fig. 3.2-4.



Value: 18.
Access Level: 3.
Array Dimension: [].
Browse Name: 1:TURN_ORA2.
Data Type: i=5.
Display Name: TURN_ORA2.

1:TURN_REF1
1:TURN_ORA2
1:TURN_ORA1
1:index_en_total
1:index_en_intern
1:index_en_spd

Fig. 3.2-2 Variable browsing and initiating a subscription.



NodeId: ns=1;s=t|
D1
Value: 34.519222

NodeId: ns=1;s=t|
D2
Value: 35.10819

Title: Debit
NodeId: ns=1;s=t|
D1
Value: 34.53111

Title:
Temperatura
NodeId: ns=1;s=t|
D2
Value: 35.320168

Fig. 3.2-3 Variable browsing and initiating a subscription.

The general architecture of the application is shown in Fig. 3.2-5, comprising two Android Activities: Connect Activity and After Connect Activity. In the connection module, users can initiate new OPC UA sessions, discover server endpoints via URI, or reconnect to existing sessions. All operations run asynchronously, separate from the main UI thread. The Discover Endpoints feature provides a list of *EndpointDescription* objects containing connection details such as URL, security mode, policy, and certificate. Selecting an endpoint creates a new client, and successful connections are stored with the session name in *SharedPreferences*. On reconnection, the stored *EndpointDescription* is retrieved, and all diagrams, objects, and subscriptions are automatically rebuilt for the client.

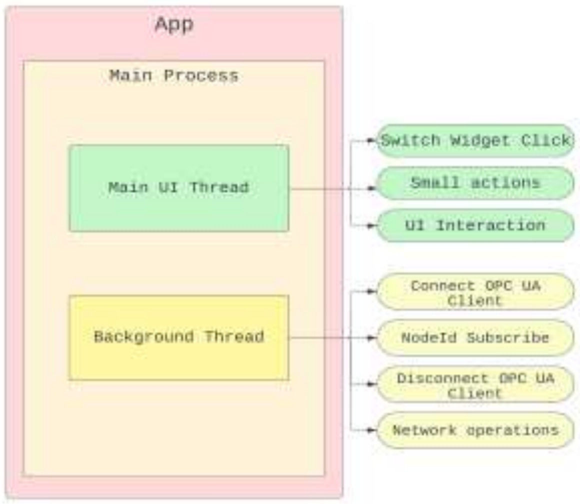


Fig. 3.2-4 Main processes of the application.

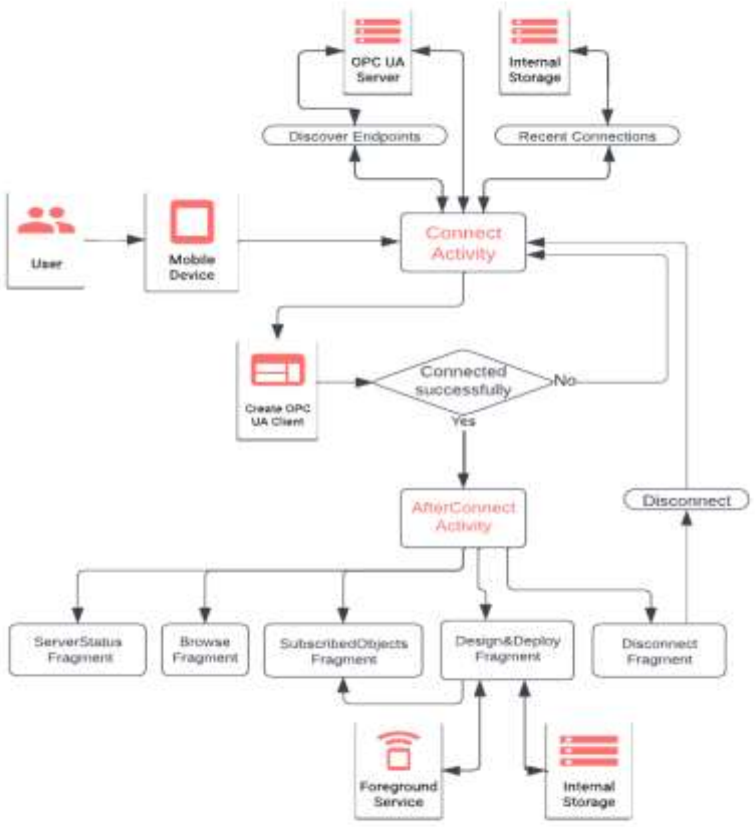


Fig. 3.2-5 General architecture of the application

Later, the MQTT protocol was also added to the application, as alternative to OPC UA. Fig. 3.2-6 depicts the choice to be taken at the initial connection.

Three object types were defined in the application: digital, analog, and alarm. Digital and analog objects serve as structured items for storing and representing data, and can be used to populate diagrams. Each may be enhanced with representative images, titles, and a *NodeId* linking the object to an OPC UA server variable. The custom object types with distinct characteristics are illustrated in Fig. 3.2-7.

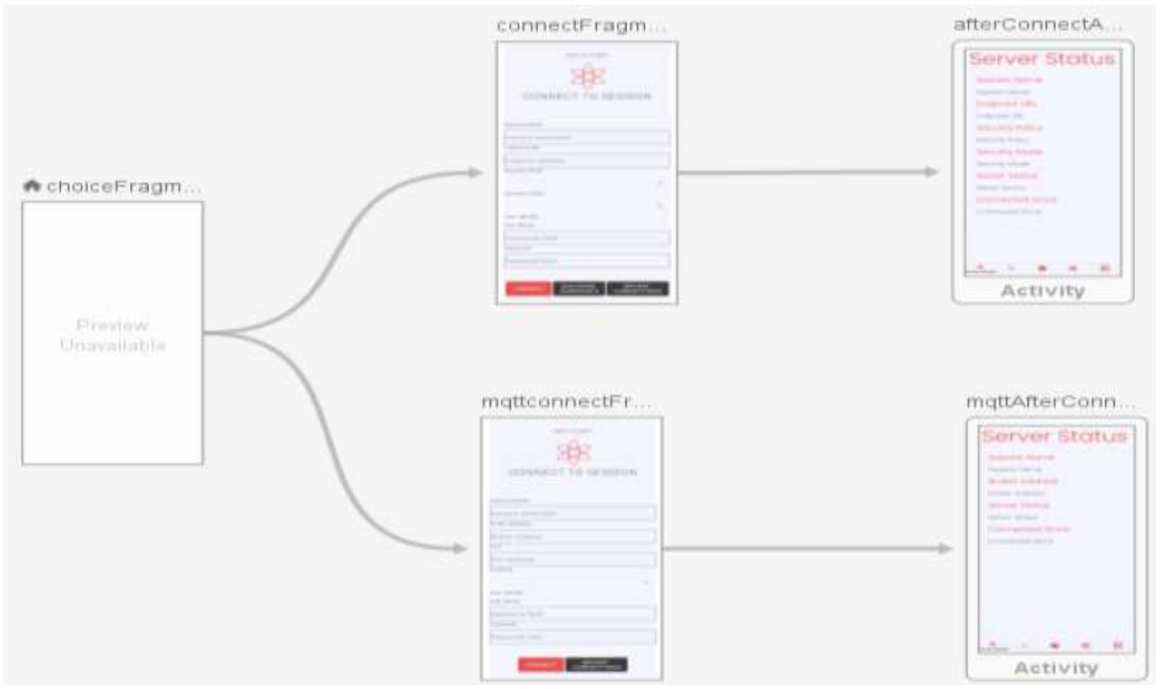


Fig. 3.2-6 Choosing the communication protocol OPC UA - MQTT

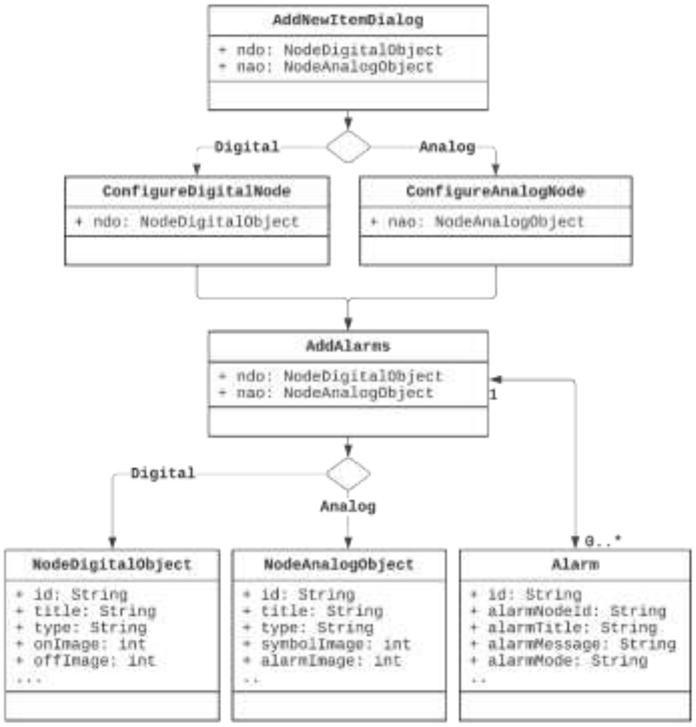


Fig. 3.2-7 Software architecture of creating a Digital or Analog Object and Alarms.

Reliable OPC UA client–server communication was ensured through a reconnection strategy, improved error handling, and optimized disconnection processes. A listener monitors server status, triggering automatic reconnection when issues occur. Disconnection is managed both manually via the UI and automatically when the application closes, preventing crashes and time-outs caused by limited concurrent server connections. Upon successful connection, the second activity serves as the application hub, containing a menu with Server Status, Browse, Subscribed Objects, Design and Deploy, and Disconnect Fragments. Navigation between fragments is handled by a dedicated component. The Design and Deploy Fragment is central: in Design mode, users can create, configure, and edit digital/analog objects and alarms through three dialogs (object creation, NodeId selection, alarm setup). In Deploy mode, a Foreground Service subscribes asynchronously to all *NodeId*s, ensuring data updates and alarm notifications remain visible even when the app is minimized. Data persistence uses *SharedPreferences* for sessions and simple objects, with Gson converting complex objects to JSON strings for storage and retrieval. Alarm objects, linked to *NodeId*s, support the Alarms and Events (A&E) service. Conditions trigger alarms, changing object color to red and logging events with time and date. With servers supporting the Alarm and Condition (A&C) service, alarms are transmitted separately from data access, enabling acknowledgment across the system and consistent messaging for all participants.

Both monitoring and control are implemented. Control tasks use asynchronous operations: each *ImageView* registers context menus, showing edit options in Design mode and control options in Deploy mode (see Fig. 3.2-8). Three control cases were developed:

- Digital object with numerical ON value: *digitalObject.isValueOn()* verifies configuration, and *digitalObject.getOnValue()* retrieves the ON state. The OPC UA Node data type is checked to avoid *Bad_TypeMismatch*.
- Digital object with bitwise ON state: *digitalObject.isBitwiseON()* confirms configuration, and *digitalObject.getOnBit()* retrieves the bit position. Both ON and OFF values can be assigned to a bit.
- Analog object: a dialog prompts the user to enter a new value in an *EditText* field. The linked Node data type is validated, and the value is written using *client.writeAttribute* for the *NodeId*.

Adding a new object to the layout involves three dialogs. The first, opened via the + button, is the Configure Object dialog where object attributes are defined. The primary attribute is the type (Analog or Digital). Analog objects require images for Symbol and Alarm states, while digital objects require images for ON, OFF, and Alarm states. For testing, an analog object named

"P2_Putere", representing a pump power meter, was created with Symbol and Alarm images as shown in Fig. 3.2-9.

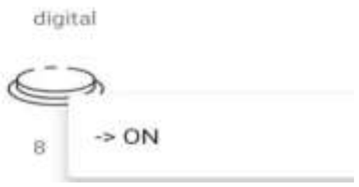


Fig. 3.2-8 Control menu displayed in Deploy mode

The second dialog links the object to its corresponding *NodeId* from the OPC UA Address Space. Node selection is performed through a browsing dialog, where the operator taps and holds the desired node, as shown in Fig. 3.2-10.

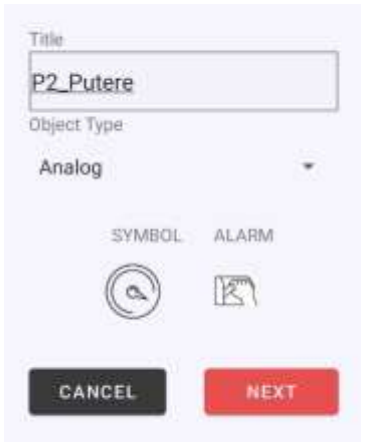

Title: P2_Putere
Object Type: Analog

Fig. 3.2-9 Configuring a new analog object named "P2_Putere"

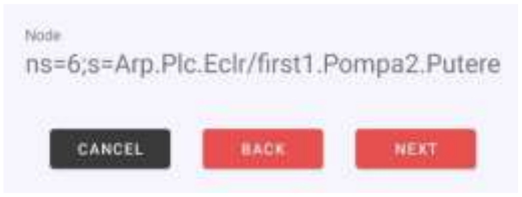

Node: ns=6;s=Arp.Plc.Eclr/first1.Pompa2.Putere

Fig. 3.2-10 Selecting the NodeId for the analog object

The final step involves adding an alarm. In the previous example, an Alarm object was created using the same NodeId as the analog object, with a unique name, message, mode, and setpoint. (Fig. 3.2-11). Alarm modes include: equal, not equal, above setpoint, below setpoint, between setpoints, outside setpoints, ON value, or bit state. Depending on the mode, one or two setpoints are required. In the example, the mode is above setpoint; thus, if the server value exceeds the setpoint "3," an alarm is triggered.

Switching to Deploy mode triggers the Foreground Service. An icon appears in the device status bar, and a notification is shown in the drawer (Fig. 3.2-12), initiating monitoring. The alarm is appearing as notification (see Fig. 3.2-13), respectively marked on the object within the view (Fig. 3.2-14).

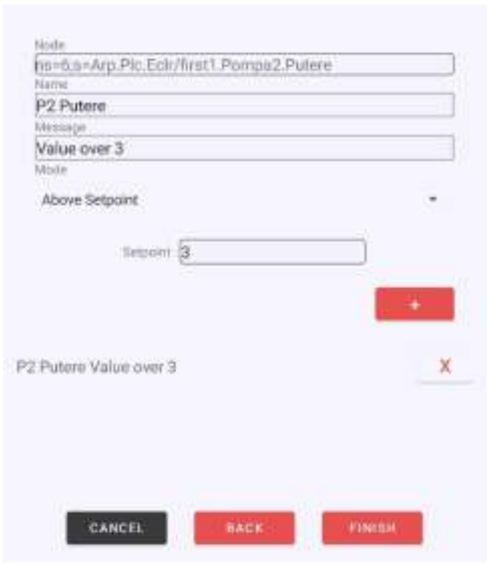
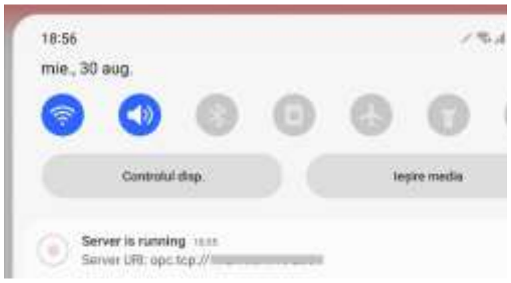

Node
ns=6;s=Arp.Plc.Eclr/first1.Pompa2.Putere
Name
P2 Putere
Message
Value over 3
Mode
Above Setpoint
Setpoint 3
P2 Putere Value over 3

Fig. 3.2-11 Adding an alarm to the analog object


18:56
mie., 30 aug.

Server is running
Server URI: opc.tcp://[redacted]

Fig. 3.2-12 Notification when the server starts running

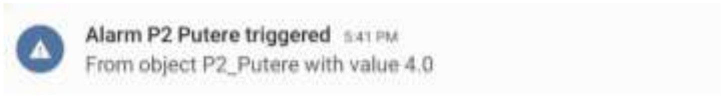
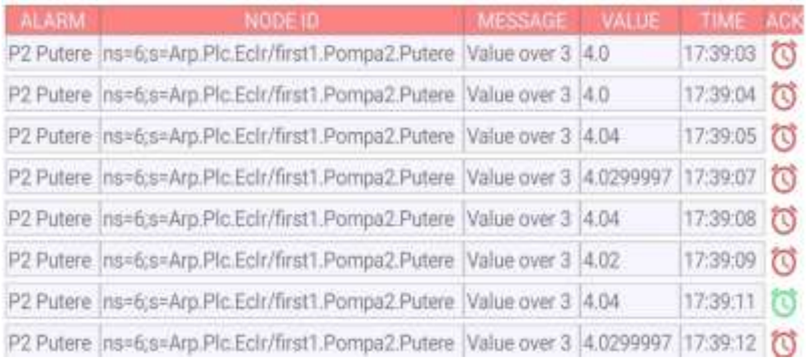

Alarm P2 Putere triggered 5:41 PM
From object P2_Putere with value 4.0

Fig. 3.2-13 Alarm notification for object "P2_Putere" when value is over set-point 3

During Deploy mode, triggered alarms are displayed in an alarm list. Activating the Alarm List button opens a *DialogFragment* with a scrollable *TableLayout*. Each alarm is represented by a new row containing its title, NodeId, message, value, trigger time, and a red alarm image for acknowledgement, as shown in Fig. 3.2-15. The user can acknowledge an alarm by tapping on it and confirming the action. This action changes the acknowledge alarm image color. The list is limited to 50 alarms that are chronologically kept within the list to avoid saving unnecessary data which can diminish performance.

For detailed time-based monitoring, trend graphs were implemented using *SurfaceView* and *Canvas*. Users can select multiple NodeIds for simultaneous monitoring, and graphs can be scaled within defined limits. As an example, the evolution of pump speed (rotations/min) is graphically illustrated in Fig.3.2-16 using a trend graph.

Fig. 3.2-14 Value above set-point 3, showing an alarm within the diagram.



ALARM	NODE ID	MESSAGE	VALUE	TIME	ACK
P2 Putere	ns=6;s=Arp.Plc.Eclr/first1.Pompa2.Putere	Value over 3	4.0	17:39:03	⏰
P2 Putere	ns=6;s=Arp.Plc.Eclr/first1.Pompa2.Putere	Value over 3	4.0	17:39:04	⏰
P2 Putere	ns=6;s=Arp.Plc.Eclr/first1.Pompa2.Putere	Value over 3	4.04	17:39:05	⏰
P2 Putere	ns=6;s=Arp.Plc.Eclr/first1.Pompa2.Putere	Value over 3	4.0299997	17:39:07	⏰
P2 Putere	ns=6;s=Arp.Plc.Eclr/first1.Pompa2.Putere	Value over 3	4.04	17:39:08	⏰
P2 Putere	ns=6;s=Arp.Plc.Eclr/first1.Pompa2.Putere	Value over 3	4.02	17:39:09	⏰
P2 Putere	ns=6;s=Arp.Plc.Eclr/first1.Pompa2.Putere	Value over 3	4.04	17:39:11	⏰
P2 Putere	ns=6;s=Arp.Plc.Eclr/first1.Pompa2.Putere	Value over 3	4.0299997	17:39:12	⏰

Fig. 3.2-15 A list of alarms triggered for analog object "P2_Putere"..

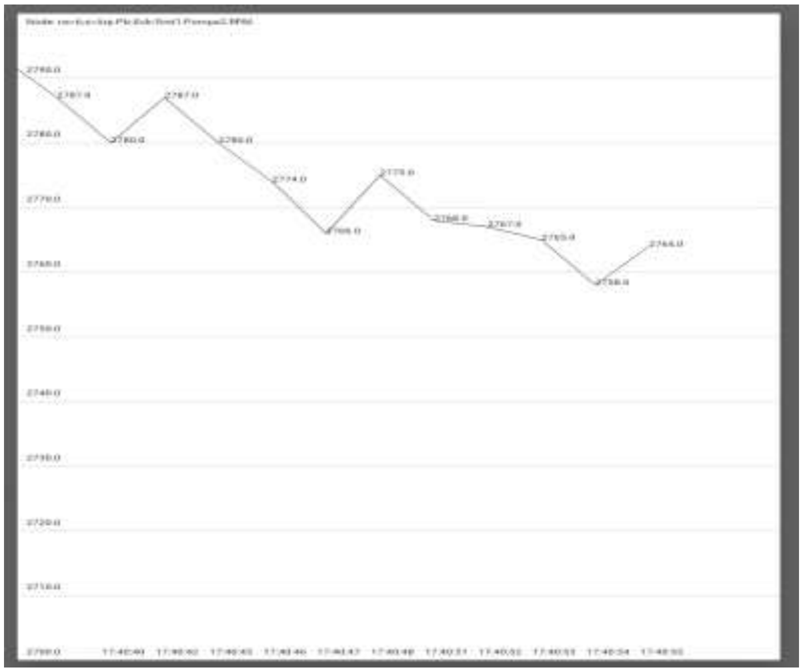


Fig. 3.2-16 Tendency graph for a node with values from 2700 to 2800.

3.3 Approaching Node-RED SCADA while Acknowledging Industry 5.0 Requirements.

The industry is initiating more-and-more transition towards solutions that are assuring as much as possible the three Industry 5.0 pillars. In this sense the essence is to be able to adapt and to be extended rapidly and efficiently to any requirement of the operator, to assure proper and quick maintenance, to shorten and fasten supply lines. One direction would require to initiate more basic level development within open-source environments. However, IIoT and digital transformation focus, popularity and new technologies would be essential. Therefore, another SCADA approach that was undertaken is referring open-source environment based SCADA. The Node-RED environment was in the center of this approach, as being more and more present in industrial environments and satisfying all criteria needed to successfully deploy and maintain a new type of SCADA system.

The steps towards researching and developing a Node-RED based SCADA were to initially obtain a generic solution that cover basic SCADA modules and then to refine, improve, and grow the solution for industrial applicability. Therefore, the current section exposes parts of studies from [K-22] and [K-8]

OPC UA represents a modern application-layer protocol designed to support increasingly demanding requirements for high-volume and high-speed industrial data exchange. In order to complement its capabilities, additional transport-level protocols are being adopted, such as UDP and more recent solutions within the OSI model, including Message Queuing Telemetry Transport (MQTT). Both MQTT and AMQP provide efficient mechanisms for cloud integration, relying on a robust publish–subscribe paradigm that is well-suited for handling large-scale data streams. Current research highlights MQTT as a central focus of academic inquiry, while industrial practice has also embraced it, with implementations appearing in PLCs. SCADA systems are gradually adopting MQTT as well. For instance, Ignition, one of the most competitive SCADA platforms, offers MQTT functionality through third-party modules, with Sparkplug serving as the associated application protocol. Despite its potential, Sparkplug has yet to achieve widespread adoption, particularly within European markets.

Traditional SCADA systems, however, continue to evolve at a slower pace. Their high cost and the necessity of maintaining backward compatibility with earlier versions pose significant challenges. Moreover, many SCADA environments historically relied on domain-specific legacy protocols, such as IEC 60870-5-104 in the electrical sector. Initially, the primary challenge for SCADA was the integration of heterogeneous data sources, especially interfacing with lower-level devices like PLCs. This led to the development of numerous proprietary drivers, which often distinguished one SCADA solution from another. With the emergence of centralized OPC and later OPC UA servers, SCADA platforms gradually transitioned to these standards. The introduction of OPC clients further accelerated adoption, enabling interoperability across higher system layers. Some environments, such as Citect, were designed natively around OPC, while others like Ignition, which emphasizes customization and software-centric development, support OPC/OPC UA alongside legacy protocols including Modbus TCP, S7, and Ethernet/IP. Nevertheless, only a limited number of SCADA systems currently integrate MQTT as a native communication option.

Beyond communication, SCADA platforms encompass a wide range of functionalities, including data visualization, graphical interface development, logging, archiving, alarm management, reporting, and mobile access. Conventional SCADA systems typically represent data through tag-based or graphical models. To reduce development and maintenance costs, alternative paradigms have been introduced. For example, the IGSS employs an object-oriented methodology, wherein physical assets are represented by digital objects composed of atomic elements linked to PLC tags. Even licensing in IGSS is object-based. Other object-oriented approaches define core entities with attributes and methods, deferring graphical representation to later stages, thereby enhancing development efficiency.

Data storage and archiving are increasingly supported by SQL-based databases, with platforms such as WinCC, Ignition, and Indusoft adopting this model. IGSS has also transitioned to SQLite as its default database engine. Mobile capabilities are another area of expansion, ranging from basic remote desktop modules to dedicated mobile applications (e.g. IGSS) and advanced frameworks such as Ignition Perspective, which enable custom mobile solutions. Modern SCADA development emphasizes rapid, concurrent deployment, often leveraging web technologies to deliver flexible and scalable solutions, as seen in Indusoft and Ignition.

The [K-22] study introduced a cost-effective, modular, platform-independent SCADA architecture built upon the Node-RED IoT framework. The proposed system establishes an IoT network that facilitates seamless communication between physical and digital components while implementing essential SCADA functionalities for process monitoring. The solution integrates Modbus TCP and MQTT protocols, employs InfluxDB for time-series data management, and utilizes Grafana to enhance visualization and database interaction. Experimental validation demonstrates the effectiveness of this approach, confirming its suitability for efficient and customizable process supervision.

Node-RED incorporates a wide range of communication protocols spanning multiple layers of the OSI model. Among these, the OPC UA has long been established, continuously evolving to expand its interoperability and functional scope. Within this framework, OPC UA clients enable direct connectivity to field devices for data acquisition, while OPC UA servers facilitate integration at higher system levels or enable peer-to-peer communication by exposing structured datasets. Sparkplug B has been introduced into Node-RED, providing standardized connectivity between Sparkplug-enabled devices through the MQTT transport protocol. MQTT itself has emerged as a universal messaging protocol, valued for its simplicity, lightweight implementation, and minimal resource requirements. Node-RED natively supports MQTT through dedicated nodes, enabling integration with brokers such as Mosquitto. In parallel, AMQP nodes are also available, complementing MQTT in providing reliable cloud-oriented communication.

Beyond these, Node-RED offers connectivity to major cloud ecosystems, including Microsoft Azure and Amazon Web Services (AWS). Furthermore, Node-RED has developed its own hosted cloud platform, Front End Node-RED (FRED), which provides a managed environment for deploying IoT solutions. The adoption of cloud services in IoT architectures is critical for centralized data management, ensuring secure, efficient, rapid delivery of information.

In IoT platforms, historical data archiving is essential for analytics, requiring robust storage methodologies within system infrastructures. Selecting an appropriate database is critical, with factors such as scalability, portability, efficiency in writing and accessing data, compression, security, and implementation costs playing decisive roles. Node-RED supports integration with multiple databases, including MSSQL, MySQL, SQLite, PostgreSQL, and Oracle, as well as modern time-series solutions like InfluxDB. While relational SQL databases transformation remain viable, time-series systems provide performance for rapid logging and retrieval of continuously generated data. Due to its modular and customizable design, Node-RED enables SCADA-specific functions such as logging, archiving, and reporting, even at the integrator level. Visualization tools like Grafana further enhance database manipulation and graphical representation, supporting the creation of dashboards that emulate SCADA monitoring. Compared to traditional SCADA systems, Node-RED offers broader connectivity, flexible scripting, and advanced data analysis capabilities, yielding a favorable cost–benefit balance. Its flow-based architecture, exportable as JSON files, ensures high technological readiness and facilitates straightforward identification of tags, objects, and data streams. Consequently, Node-RED demonstrates significant potential for IoT/IIoT applications and is increasingly positioned as a competitive environment for next-generation SCADA solutions.

The architecture of the case-study SCADA system implemented in Node-RED is illustrated in Fig. 3.3-1. The communication protocol selected for PLC communication was Modbus TCP, though other protocols could be applied. In this configuration, data generated by a simulated PLC is acquired and processed within Node-RED, which inserts the resulting time-series values into the designated database. Grafana subsequently performs scheduled queries on this data source, enabling visualization of key metrics. The resulting dashboard panels are integrated into the Node-RED interface, providing a unified monitoring environment. An MQTT broker manages message distribution, ensuring that all published data are delivered to subscribed clients. Finally, two client applications were employed to validate communication across the network.

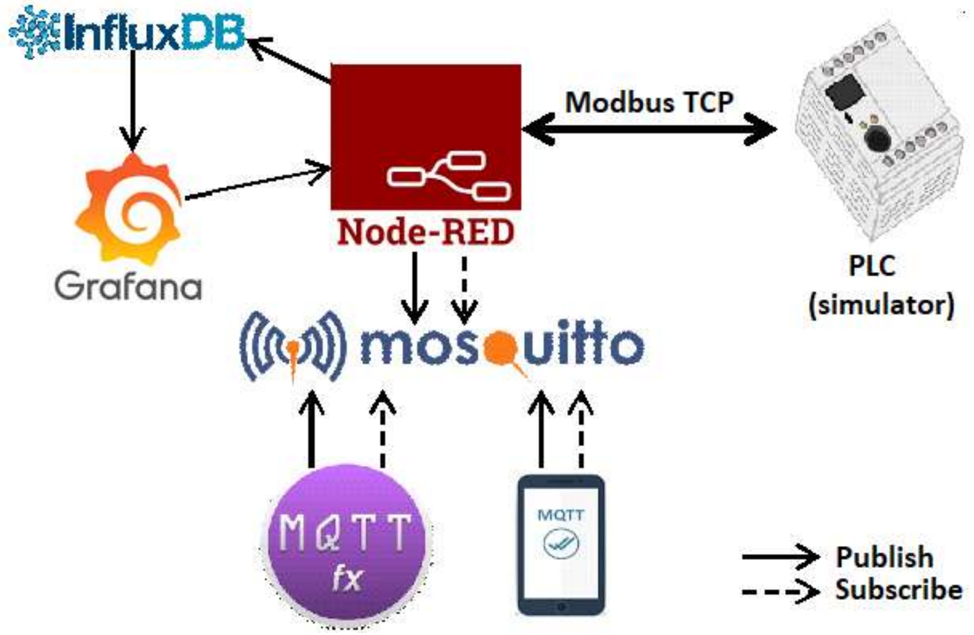


Fig. 3.3-1 The architecture of the proposed case study.

To ensure proper data storage within measurements, a dictionary must be defined using JavaScript. Specific values can then be accessed or published to designated topics by declaring global variables through the *global.set()* function, while retrieval is enabled via *global.get()*, callable from any flow or sub-flow node. Following data processing, the formatted information is inserted into the InfluxDB time-series database (see Fig. 3.3-2).

The second stage in completing the main SCADA component involves embedding Grafana panels into a unified Node-RED dashboard (see Fig. 3.3-3). This integration is achieved through the layout tab's grouping system, where each panel is assigned a tab and positioned at the desired interface location. Embedding is performed via a function node, whose payload specifies the Grafana related characteristics. This configuration ensures seamless visualization within a single monitoring environment.

The sub-flows are integrated into the main application flow, which defines the overall system logic. The dashboard elements, comprising a slider node, two switch nodes, and a non-editable text node, were configured and grouped within the interface to link graphical components with the underlying logic. To enhance monitoring, an SVG node was employed, enabling animated visualization of motor states through customized properties. Message payloads were structured in a function node to update selectors with specific attributes, such as the fill property, ensuring accurate graphical representation. User inputs are processed by function nodes, which publish results to an MQTT topic. Finally, the processed messages are directed to a “Modbus-Write” node, allowing modification of values at designated holding register addresses, thereby completing the control of the system.

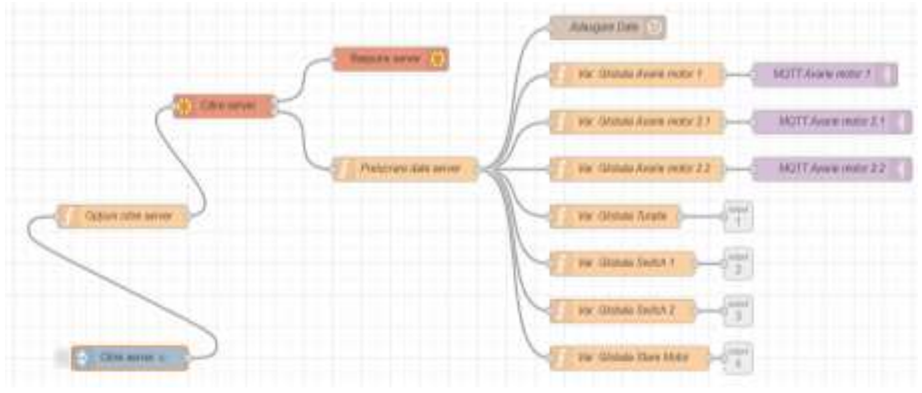


Fig. 3.3-2 Sub-flow for reading, processing and inserting data into the database.

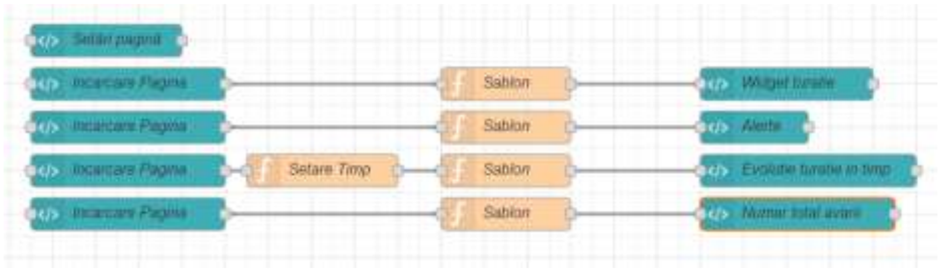


Fig. 3.3-3 Sub-flow for importing Grafana panels.

The solution was tested and validated using various scenarios, using standard computer and mobile device as a SCADA running entity. A screenshot depicting dashboard status as a result from a tested scenario is presented in Fig. 3.3-4. As observed, all SCADA related modules are functioning properly.

Following [K-22], the Node-RED SCADA solution was extended and applied in a real industrial scenario, within the building management system of an automotive company. Research contracts were improving the solution and the final application is running for several years now in the industrial environment.

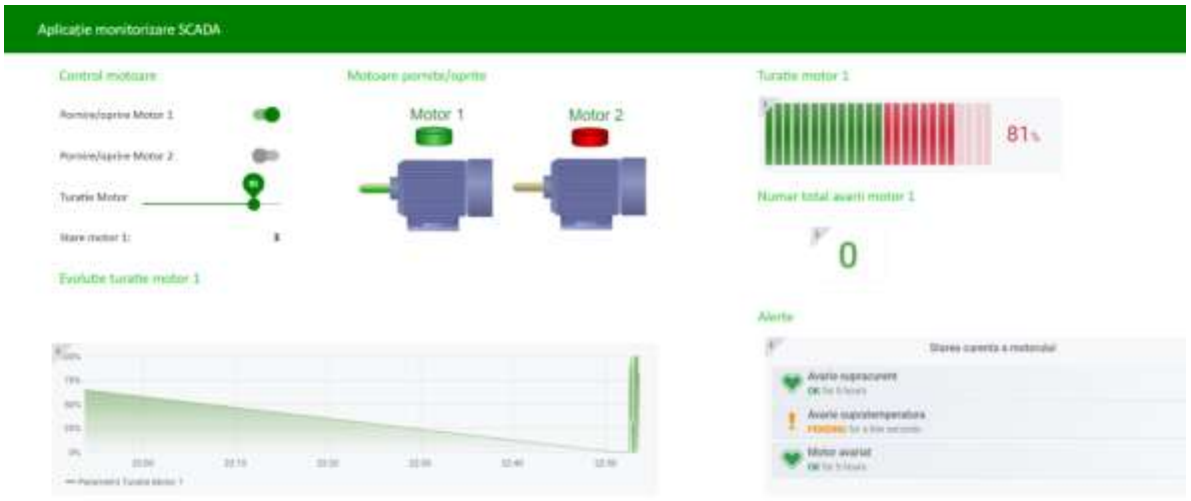
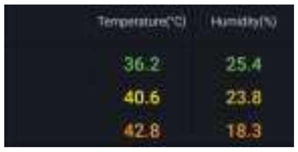


Fig. 3.3-4 One status of the SCADA dashboard in a tested scenario.

Work [K-8] presents some main outcomes of the research activity, the Node-RED SCADA solution being a main contributor. The application was tested for correct data acquisition, data representation, complete alarming and notifying module, reporting module, and running on premise and on the cloud. Figures 3.3-5, 3.3-6, 3.3-7 are presenting screenshots regarding the functioning of some modules.



Temperature(°C)	Humidity(%)
36.2	25.4
40.6	23.8
42.8	18.3

Fig. 3.3-5 Grafana stat view for event-based sensor data.

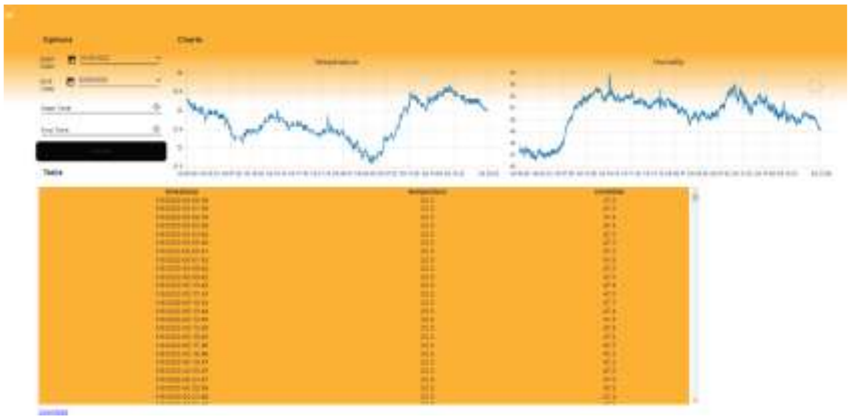


Fig. 3.3-6 Grafana stat view for Datalogger acquired data.

The research resulted in a fully operational solution capable of integrating diverse legacy systems commonly used in industry. Visual outputs confirmed both the accuracy of acquired data and the effectiveness of the acquisition process. Statistical dashboards were configured to display final values stored in the designated database tables. Upon successful insertion, the data are retrieved from the database and presented on the dashboard through the appropriate graphical components, ensuring reliable monitoring and validation of system performance. Fig. 3.3-8 presents some example of processing logic usage of CPU and Memory.

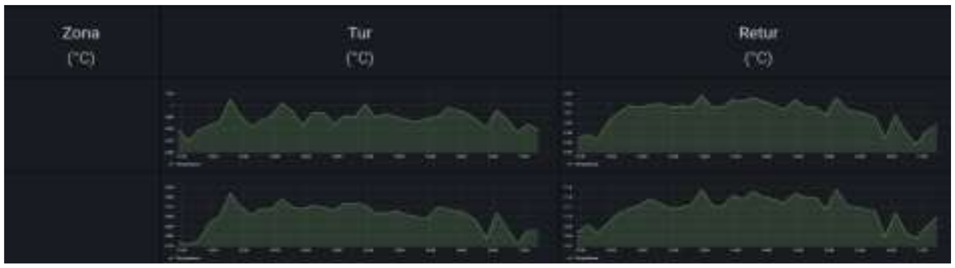


Fig. 3.3-7 Grafana trend chart.

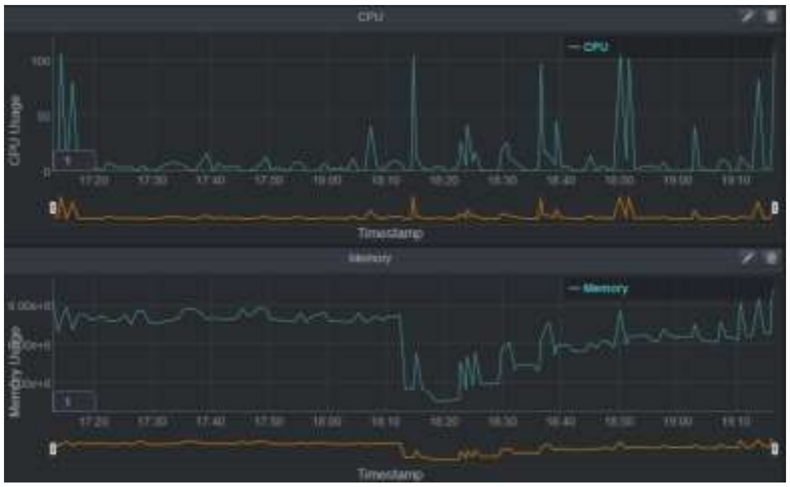


Fig. 3.3-8 Example of Processing logic usage of CPU and Memory.

4 Increasing Efficiency in an IIoT Guided Industrial Evolution

The current chapter consists of information from 11 scientific works [K-7], [K-12], [K-14], [K-17], [K-20], [K-23], [K-24], [K-28], [K-32], [K-33], [K-36]. The goal was to increase efficiency in the industry, following IIoT and Industry 4.0 principles. The research works had in mind Industry 4.0 targets, but some influenced positively future Industry 5.0 pillars through various results and conceptual approaches (e.g. decentralized and local processing, energy consumption reduction, wastewater overflow prevention, non-invasive control augmentation to improve and extend the lifetime of systems, defect and quality indicator forecasting, human-centric applications both in water and automotive scenarios). The industrial domains of application are: the water sector and the automotive manufacturing.

Industrial sectors differ in their capacity for reconfiguration. While industries such as automotive manufacturing frequently adapt production lines to client demands, the water sector remains resistant to invasive changes, relying heavily on legacy systems. This results in heterogeneous, chronologically dispersed solutions that require efficiency improvements through non-invasive and sustainable strategies. Consequently, water treatment and distribution facilities face persistent challenges including high energy use, equipment failures, excessive chemical consumption, maintenance demands, and variable source quality [103]. Many technical implementations, once viable, are now outdated, requiring renewed academic and industrial efforts to unlock the sector’s potential for human health and environmental protection [104].

Under Industry 4.0, competition has focused on system connectivity and interoperability [105], as well as safeguarding critical infrastructure through automation, SCADA, and communication technologies [106]. However, research must be industry-oriented, as many studies remain theoretical without practical applicability. Industry 4.0 connectivity has introduced the concept of data accumulation, typically implemented via historian applications [107]. Current approaches emphasize traditional storage and reporting rather than process orientation. For example, [108] adapts historian solutions for the electrical domain, extending support for IEC 61850. In the water sector, decentralized historian solutions are needed [109]. Despite large volumes of collected data, much remains unused, highlighting the need for proactive historian applications, advanced analytics, and optimization strategies. Autonomous optimization loops require model-based analysis, decision procedures, and non-invasive control. Studies such as [113] demonstrate the potential of data-driven analysis of latent alarms and events in IIoT contexts.

Emerging edge/fog computing concepts further support Industry 4.0 by enabling local automation. Research in [114] evaluates middleware platforms for IoT solutions in fog and cloud configurations, applied to irrigation scenarios. Fog computing is also proposed in [115, 116] and extended to hybrid wind farm control [117]. Conversely, cloud computing remains advantageous for large-scale distributed processes requiring less granular optimization, such as supply chain integration [118].

As drinking water facilities increasingly adopt Industry 4.0, the integration of physical and digital systems introduces challenges such as high energy consumption, maintenance demands, source quality variations [119], pump failures [120], and excessive chemical use [121]. Studies address these issues from multiple perspectives. For example, [122] analyzes energy requirements and carbon footprint in desalination for swimming pools, though without optimization steps. Improvements are proposed in [123] through real-time SCADA monitoring to reduce water loss, while [124] develops a non-linear multi-year model for sustainable groundwater distribution in irrigation. Water demand calibration impacts are studied in [125]. Optimization of turbidity treatment using natural coagulants is presented in [126], and water quality monitoring strategies are discussed in [127]. The influence of climate change on drinking water systems is highlighted in [128]. Optimization efforts often focus on costs, linked to chemical usage, energy, and maintenance. [129] examines proportional effects of chemical consumption on quality indicators, while [130] considers costs at a general level. Automation strategies also reduce costs, as shown in [131] with frequency-converter pumps in small facilities. A broader cost perception study of distribution networks is presented in [132]. Further research explores reservoir operation optimization to minimize pollution losses [133], energy reduction in desalination [134], and raw source water impacts on treatment [135]. Long-term degradation of water sources is documented in [136], emphasizing the need for systematic data collection and learning. Predictive approaches include turbidity forecasting with early warning systems [137], anomaly detection in distribution via supervised learning [138], SCADA-based anomaly identification [139], and missing data compensation [140]. Integration of weather data [141] demonstrates how IIoT concepts, particularly proactive historian applications, can enhance efficiency in water systems.

In wastewater treatment, Sandu et al. [142] conducted a numerical study introducing wall structures to prevent low-velocity flows that promote sedimentation and disrupt treatment. Similarly, [143] proposed predictive control schemes to enhance stability and efficiency. Neural networks are widely applied in industry for improved decision-making with accumulated data. Examples include defect detection via deep learning [144], CNNs for sheet-metal fixture layouts [145], and IIoT deep learning syntheses [146]. Prediction of faults and uncertainties is increasingly important, with LSTM recurrent NN models used for data-driven approaches. For instance, [147] predicts faults by generating expected images one second ahead, while [148] forecasts production progress. Other solution includes probabilistic temperature prediction in additive manufacturing [149]. Research must remain process and data driven, grounded in real industrial systems. Optimized LSTM strategies for chemical fault diagnosis [150] and short-term voltage stability assessment [151] demonstrate superior performance compared to traditional methods, though further industrial deployment is needed. In the water domain, improvements are largely data-driven and NN-based. CNNs are applied to pipe leakage detection [152], CNN/LSTM combinations to underground drainage sensing [153], and Raspberry Pi-based CNN/LSTM systems to mechanical water meters for leakage detection [154]. Prediction studies remain limited: [155] forecasts water quality extremes but lacks local adaptability; [156] addresses pipe failure prediction without clear timeframe applicability; and [157] focuses on company needs but fails to achieve the required prediction horizon for maintenance/control adjustments.

Predictive maintenance [158] represents a key transformation area under Industry 4.0, reducing downtime, operating costs, and enhancing efficiency, productivity, and profitability [159]. Implemented solutions, such as [160], already demonstrate significant improvements in manufacturing. Furthermore, work [K-14] details predictive maintenance by forecasting cylinder defects in the automotive industry.

Sections 4.1 is presenting the initial phase development of the low-cost decentralized historian from [K-33].

Section 4.2 introduces the concept of a Proactive historian and details the evolution to obtain a non-invasive automatic solution that reduces energy consumption in the drinking water facilities. Also, the progress towards long-term testing of the solution is presented. The information is from papers [K-28], [K-23], [K-17], [K-12].

Section 4.3 presents the solution from [K-7], based on an LSTM decentralized edge AI technique within the proactive historian that was able to predict faults and indicator values in the water sector.

Section 4.4 extends the non-invasive analysis and correction idea to wastewater treatment plants (WWTP) with the information from [K-36], where an energy reduction technique is developed, and the information from [K-24] where weather-based prediction is included in the historian.

Section 4.5 details the findings from [K-32], where improvements were realized in the functioning of groups of WWPSa using a non-invasive solution developed within a higher-level wrapping structure.

Section 4.6 presents an efficiency increase solution in the automotive manufacturing from [K-20], where image processing hardware-software structure was researched to detect ECU defects at the EoL production.

4.1 Decentralized Low-Cost Proactive Historian.

The section describes briefly the lightweight and low-cost historian developed in a first phase in [K-33] that is based on OPC UA interfacing, but extensible also for other protocols. A platform-independent historian was developed for edge deployment, suitable even for automation panel integration without SCADA supervision. The Java-based application embeds the Node-Red platform for SCADA interfacing and uses a SQLite database. Designed for high TRL, it supports long-term communication monitoring and reconfiguration, enabling rapid applicability in the water industry. It also serves as a foundation for further research on automatic stored-data analysis, generating conclusions for diverse water objectives and transmitting them in appropriate formats for control adjustments or alarms. Commercial SCADA software typically offers limited logging/archiving functions, with data manipulation constrained by licensing, file formats, or restricted export options. Historian software is often sold separately at high cost. Local SCADA control rooms, covering WTPs, WWTPs, chlorination stations, pumping stations, wells, reservoirs, and measurement points, hold the largest data volumes. Effective local historians must balance interoperability with cost-benefit considerations.

Practical experience shows that only ~5% of SCADA control rooms use separate historians, mostly in large treatment facilities. These are expensive, platform-dependent, LAN-connected products, rarely used by operators beyond data export or archiving. Low-competency operators often struggle with complex interfaces, while skilled operators use them sparingly. Downtime of SCADA systems frequently disables historians until external maintenance intervenes, as their structure is tightly coupled to vendor software.

At the automation panel level (WWPSs, wells, pumping stations, chlorination stations, small WTPs), historians are absent, with only short-term HMI logging available. Thus, a low-cost, lightweight historian is needed, adapted for rapid panel integration and accessible to operators with limited IT skills.

At the central/regional SCADA level, historians are typically connected to SCADA servers or, where permitted, to SCADA gateways. Operators use them mainly for archiving, exporting, and offline analysis, with historical data later supporting higher-level processing. Several issues arise at this level:

- Data collected in regional control centers is limited to generic information.
- Water distribution companies cover large areas (one or several counties), so local processes are not monitored in detail. Attempts to centralize large volumes of data from local automation panels (via OPC/OPC UA servers) often result in misrepresentation, leaving historians as mere storage tools with numerous unhandled alarms.
- Instead of transferring large amounts of live data for costly central analysis, conclusions should be derived locally and then processed at the central historian for clarity.
- Communication failures with local systems sometimes lead to data loss.

The overall historian architecture and its integration with other OPC UA interfacing structures is illustrated in Fig. 4.1-1. The historian structure is defined by several key characteristics: OPC UA interfacing with full security compliance, extensible to legacy protocols (e.g., Modbus, S7); Integration with control structures, transmitting processed conclusions to local automation for algorithm adjustments; Address space browsing and variable definition for historian inclusion; Database management, storing selected tags with timestamps, adaptable to variable or table changes; Querying and exporting data in online/offline modes across chosen intervals, supporting .xls, .pdf, and .csv formats; Operator usability through a simple GUI; Continuous monitoring of the Java application, Node-Red, and SQLite, with log file generation for analysis while maintaining lightweight operation; Fault tolerance, ensuring uninterrupted historian operation despite server timeouts, discarded sockets, exceptions, or user errors; Modularity, enabling future development of automatic data analysis and local algorithm adjustments for water industry objectives.

The historian was validated in real scenarios, specifically at a WTP serving ~8000 inhabitants. This facility operates with an older WinCC 7.2 SCADA system and a Connectivity Pack extension exposing tags via OPC UA servers. The application supports self-monitoring, fault handling, and auto-diagnosis across all three levels: the Java core, Node-Red interface, and SQLite database. The GUI includes a Status section providing operators with essential system information (see Fig. 4.1-2). A status log file is maintained for engineering, recording operational details, encountered faults. Real-system testing revealed periodic timeouts indicating connection loss to the local OPC UA server. These faults required automatic Node-Red interventions to reinitialize the connection. The number of restarts is tracked, with the Overall section displaying information linked to the most recent connection start.

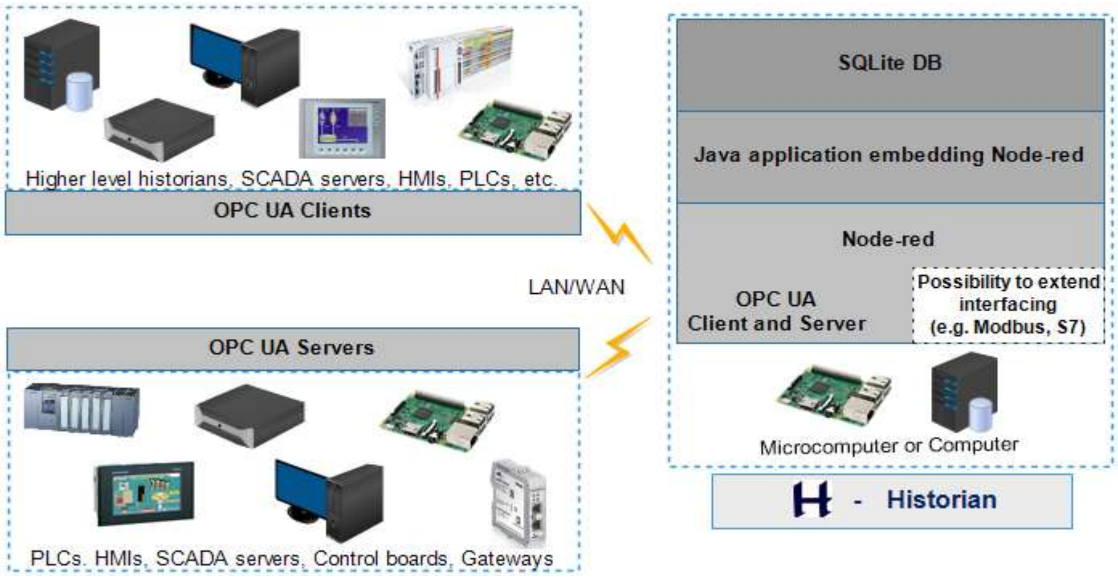
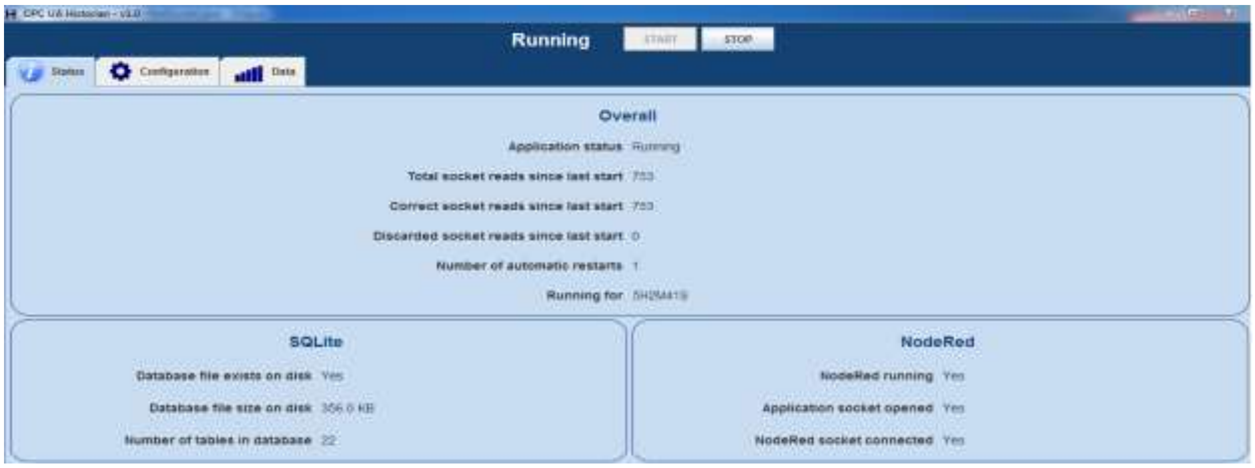

Higher level historians, SCADA servers, HMIs, PLCs, etc.
OPC UA Clients
LAN/WAN
OPC UA Servers
PLCs, HMIs, SCADA servers, Control boards, Gateways
SQLite DB
Java application embedding Node-red
Node-red
OPC UA
Client and Server
Possibility to extend interfacing (e.g. Modbus, S7)
Microcomputer or Computer
H - Historian

Fig. 4.1-1 General architecture of the developed historian and the relation with OPC UA structures.

The GUI Configuration section is presented in Fig. 4.1-3. Configuration is performed entirely within the application. Database tables are indexed by timestamp and server ID, while the configuration file remains encrypted on the local operating system. OPC UA interfacing supports five security policies (None, Basic128, Basic128Rsa15, Basic256, Basic256Sha256) and three modes (None, Sign, Sign&Encrypt), with user credential management.

The Data section enables stored information manipulation. Export and chart modules—critical for water operators—allow table selection, each defined by non-overlapping timestamp ranges. Both modules operate in Running and Not Running states. Data can be exported for analysis in .pdf, .xls, and .csv formats, while variable evolution across selected tables and time ranges can be visualized in charts, as shown in Fig. 4.1-4.



Overall

Application status: Running

Total socket reads since last start: 753

Correct socket reads since last start: 753

Discarded socket reads since last start: 0

Number of automatic restarts: 1

Running for: 5H25M41S

SQLite

Database file exists on disk: Yes

Database file size on disk: 356.0 KB

Number of tables in database: 22

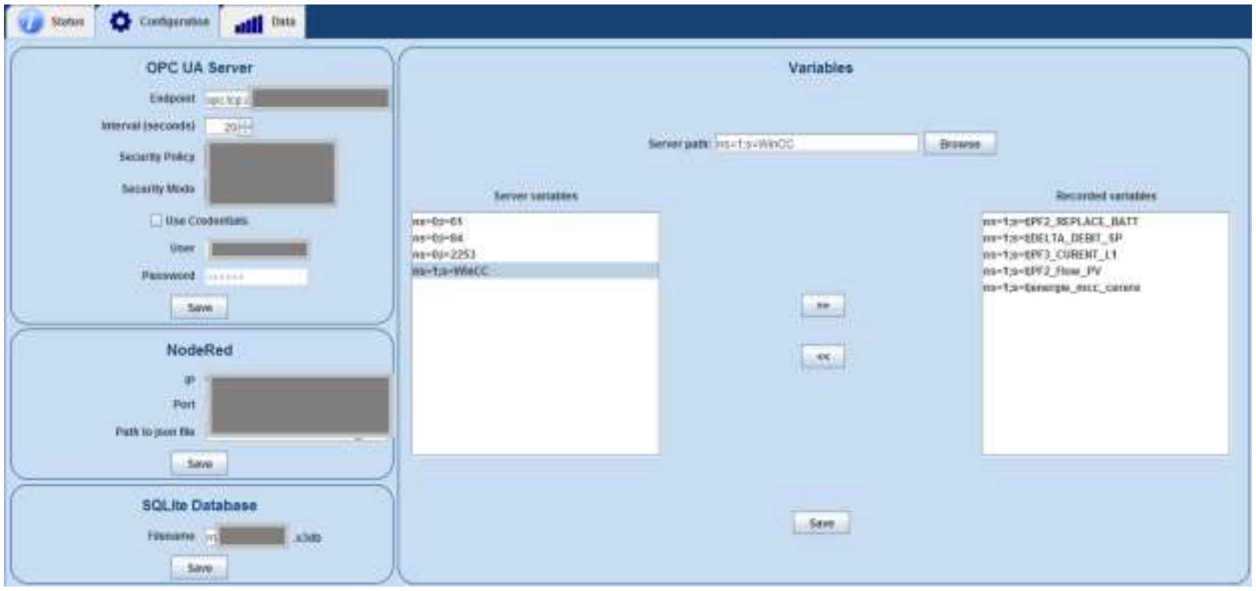
NodeRed

NodeRed running: Yes

Application socket opened: Yes

NodeRed socket connected: Yes

Fig. 4.1-2 Status section of the GUI.



OPC UA Server

Endpoint: opc.tcp:// [redacted]
Interval (seconds): 20
Security Policy: [redacted]
Security Mode
☐ Use Credentials
User: [redacted]
Password: ••••••

NodeRed

IP: [redacted]
Port
Path to json file

SQLite Database

Filename: m [redacted] .s3db

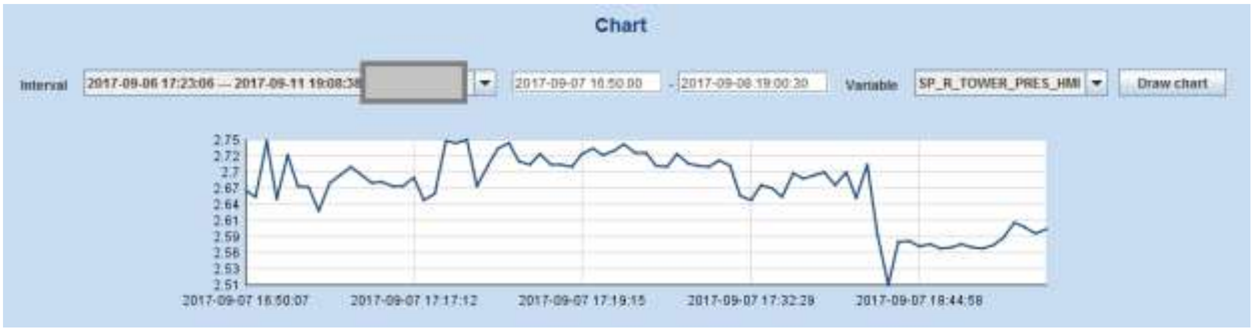
Variables

Server path: ns=1;s=WinCC

Server variables:
ns=0;i=61
ns=0;i=84
ns=0;i=2253
ns=1;s=WinCC

Recorded variables:
ns=1;s=tPF2_REPLACE_BATT
ns=1;s=tDELTA_DEBIT_SP
ns=1;s=tPF3_CURENT_L1
ns=1;s=tPF2_flow_PV
ns=1;s=tenergie_mcc_curent

Fig. 4.1-3 Configuration section of the GUI.



Chart

Interval: 2017-09-06 17:23:06 — 2017-09-11 19:08:38 [redacted] 2017-09-07 16:50:00 - 2017-09-08 19:00:30 Variable: SP_R_TOWER_PRES_HMI

Fig. 4.1-4 Output water pressure evolution inside a selected time interval represented within the chart module.

4.2 Non-Invasive IIoT Solution within the Proactive Historian to Reduce Energy Consumption for Drinking Water Facilities.

Introducing the concept of decentralized Proactive historian means to gain better knowledge about the process on the edge, to process data, to conclude an improvement scenario and to be able to execute the efficiency increase solution. This was a four step strategy for a significant accomplishment within drinking water facilities. First, [K-28] established data dependencies between the water source selection and the energy consumption, followed by [K-23] that processed data, developed the strategy to improve and to action. Work [K-17] completely automated the strategy, in order to continuously identify and update the quality of the water sources, respectively [K-12] took the research to the highest-level and apply the strategy after long-term usage, making a process aware historian.

4.2.1 Proactive historian identifying and applying the energy reduction strategy

Works [K-28], [K-23] applied the decentralized historian on drinking water facilities (DWF), and transformed it in a proactive edge solution. The current section focuses on a typical DWF that is presented in Fig. 4.2-1 (a functional real process) and consists of water sources, WTP, and water distribution facility (WDF). The water wells (WW) have two main local control loops in the automatic regime that guide the water pumping. The primary local control loop is flow-based, with a secondary level-based loop serving as redundancy. Operator-defined set-points are fixed. The analyzed scenario corresponds to a developed DWF. The DWF includes six WWs, though only four operated in automatic mode during the initial analysis period.

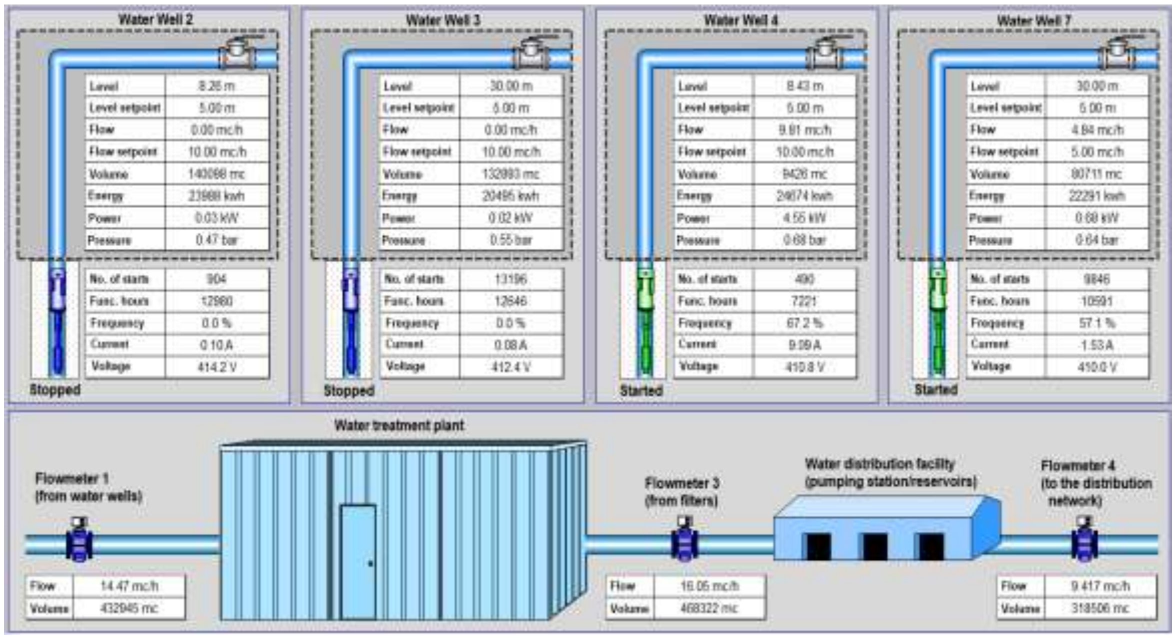
The water from the WWs flows into the WTP as presented in Fig. 4.2-2. Water treatment involves aeration, sand and charcoal filtration (4 sand and 2 charcoal filters), disinfection via chlorine stations, and sludge treatment. Sand filters reduce turbidity, while aeration and charcoal filtration regulate pH and conductivity. Maintaining legal limits requires significant energy and chlorine consumption (e.g., blowers, filter maintenance, chlorine injection). Filters are frequently cleaned with air and water to prevent clogging, leading to high energy use and water losses. Chlorine dosing employs a flow-based control strategy, supplemented by a closed-loop residual chlorine feedback system. This secondary loop requires continuous water flow from WWs to the WTP and approximately 30 minutes to achieve efficiency.

After the filters the water is taken over by the WDF (see Fig. 4.2-1). A typical WDF includes a pumping station (PS3), electric valves, and reservoirs. In this study, PS3 operates with three pumps equipped with frequency converters (FCs). Water distribution and request are managed by 3 control algorithms:

- Pressure-based loop: regulates distribution and rotates pumps based on operating hours.
- Primary level-based loop: maintains reservoir levels within hysteresis limits. When levels drop, water is requested from WWs. Variations in consumption and reserve issues can prevent strict hysteresis control, leading to higher energy use and treatment disturbances.
- Secondary flow-based loop: anticipates peak demand by comparing Flowmeter 4 and Flowmeter 1 values. If the difference exceeds a threshold, water is requested from WWs. This loop responds faster than the primary one. Both water-requesting loops select WWs based on operating hours and must account for water and time losses within the WTP.

At night, reduced demand allows the level control algorithm to stop water sources. However, due to network losses and fixed WW flow set-points, sources may start and stop repeatedly, causing pump wear and process disturbances. Short activations prevent the WTP from reaching stable operating parameters (e.g., chlorine reaction, aeration, filtration).

WWs differ in flow capacity and water quality, which vary over time. Analysis of more than 50 DWFs revealed that automation solutions rarely consider WW quality indicators. By monitoring parameters such as residual chlorine, blower hours, filter cycles, WW states, flows, and operating times, quality indicators can be adapted. This enables variable flow set-point distribution, reducing energy, chemical consumption, and equipment costs, since frequent starts increase maintenance and replacement needs. SCADA architectures typically equip WWs with PLCs, either directly connected to WTP control rooms or integrated into WDF PLCs. In older systems, WWs are activated by aeration tank levels or local pressure changes, without reservoir-based requests. These legacy solutions lack advanced control strategies, though they could yield higher energy savings. Modern WTP automation employs redundant PLCs, with WDF PLCs integrated into SCADA systems centered on redundant servers. Electrical parameters are monitored in real time, supporting more reliable and efficient operation. Fig. 4.2-3 illustrates key electrical for both (redundant) power lines. Energy data from the WTP automation (MCC), WDF (PS3), internal services panel, and total WTP+WDF consumption are central to the proposed solution. A DWF is critical infrastructure and research requires monitoring, with detailed justification, strategy, approvals. Interventions on legacy systems must remain non-invasive, while even newer WTP automation may face owner-imposed constraints, limiting the capacity of tested strategies.


Water treatment plant
Flowmeter 1 (from water wells)
Flowmeter 3 (from filters)
Water distribution facility (pumping station/reservoirs)
Flowmeter 4 (to the distribution network)

	Water Well 2	Water Well 3	Water Well 4	Water Well 7
Level	8.26 m	30.00 m	8.43 m	30.00 m
Level setpoint	5.00 m	5.00 m	5.00 m	5.00 m
Flow	0.00 mc/h	0.00 mc/h	9.81 mc/h	4.84 mc/h
Flow setpoint	10.00 mc/h	10.00 mc/h	10.00 mc/h	5.00 mc/h
Volume	140088 mc	132883 mc	9426 mc	80711 mc
Energy	23988 kwh	20495 kwh	24674 kwh	22291 kwh
Power	0.03 kW	0.02 kW	4.55 kW	0.68 kW
Pressure	0.47 bar	0.55 bar	0.68 bar	0.64 bar
No. of starts	904	13196	490	9846
Func. hours	12980	12646	7221	10591
Frequency	0.0 %	0.0 %	67.2 %	57.1 %
Current	0.10 A	0.08 A	9.09 A	1.53 A
Voltage	414.2 V	412.4 V	410.8 V	410.0 V
	Stopped	Stopped	Started	Started

	Flowmeter 1	Flowmeter 3	Flowmeter 4
Flow	14.47 mc/h	16.05 mc/h	9.417 mc/h
Volume	432945 mc	468322 mc	318506 mc

Fig. 4.2-1 A drinking water facility (DWF).

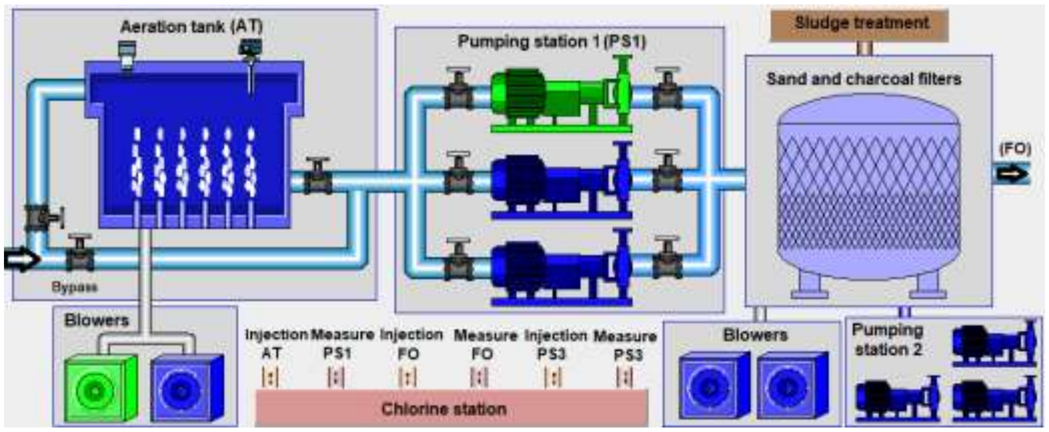
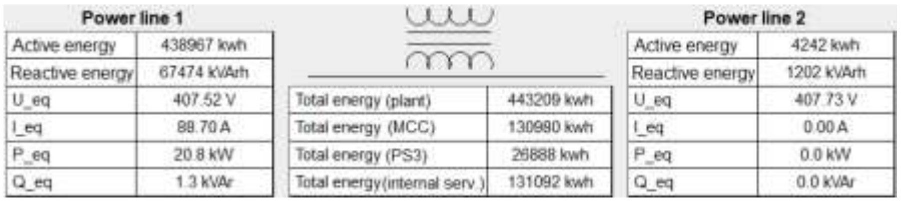


Fig. 4.2-2 A water treatment plant (WTP).



Power line 1	
Active energy	438967 kwh
Reactive energy	67474 kVArh
U_eq	407.52 V
I_eq	88.70 A
P_eq	20.8 kW
Q_eq	1.3 kVAr

Total energy (plant)	443209 kwh
Total energy (MCC)	130980 kwh
Total energy (PS3)	26888 kwh
Total energy(internal serv.)	131092 kwh

Power line 2	
Active energy	4242 kwh
Reactive energy	1202 kVArh
U_eq	407.73 V
I_eq	0.00 A
P_eq	0.0 kW
Q_eq	0.0 kVAr

Fig. 4.2-3 Main electrical parameters monitoring of the WTP and water distribution facility.

Improving DWF efficiency requires reducing energy and substance consumption while increasing productivity and availability. Long-term data analysis of thousands of process tags is used to derive an optimal cost-oriented recipe, which is first tested on process models and then non-invasively implemented at the edge/fog level of real systems. The solution interfaces with local systems to exchange data and apply the identified recipe without disrupting automation. Communication with WTP SCADA typically uses OPC

UA. When WDF and WWs are fully integrated, redundant SCADA servers handle interactions. The historian also supports PLC interoperation via legacy protocols, with direct PLC communication serving as backup during SCADA maintenance or OPC UA failures.

Data gathering and dependency analysis enable identification of WW quality indicators, linked to total energy consumption. These indicators guide the establishment of priorities and flow set-point references for each WW. The strategy achieves energy efficiency when system reactions are based simultaneously on WW priority indicators and flow set-points.

After analyzing the local process, a priority indicator is established for each WW. The priority will be a selection tool based on water quality (PQ_f) and functioning hours (PH_f), and it will influence the flow set-point (F_{W_f}) of the well's local flow-based control loop. Variable flow set-points can replace fixed values, with formulas developed for priority indicators and flow references (see [K-23]). These account for pump protection, well capacity limits, and reservoir hysteresis levels. The efficiency improvement solution within the proactive historian was validated on a calibrated DWF model using real input data and later on a real system. Direct experimentation on critical infrastructure is not feasible, so testing required long-term procedures. Two scenarios were examined. In Scenario 1, initial results from data accumulation, analysis, and conclusion phases, were tested on models and short-term real systems under strict operator supervision, demonstrating energy efficiency gains. In Scenario 2, two-week continuous testing was involving the real plant, with operator-imposed constraints. Flow set-points for WWs remained fixed, no additional wells were activated, and WW4 was replaced with WW1 within selected wells. The solution was thus tested without all modules (variable flow set-points and full well activation). Supplementary proactive historian analysis extended over an additional year of data.

The scenarios focus on the specified DWF, where local PLCs from the WDF and WWs communicate via the S7 protocol. The WTP automation employs two redundant S7-400H PLCs, while the SCADA system is WinCC 7.2 with Connectivity Pack, operating on two redundant servers. In the first test scenario, Fig. 4.2-4 shows the flow evolution of four WWs under fixed flow set-point operation, without proactive historian intervention. Water demand activates WWs without accounting for source quality or quantity.

The first scenario (Fig. 4.2-5–4.2-8) applies priority setting results (Fig. 4.2-6) for the four WWs considering functioning hours (Fig. 4.2-5) and water quality indicators. The evolutions of WWs flow references, the reservoir level, and the total flow set-point for the WWs after applying the solution, are presented in Fig. 4.2-7. The impact on this short-term test on reducing energy consumption would be, from Fig. 4.2-8 (percentage power differences between the system with and without the solution), about 9%.

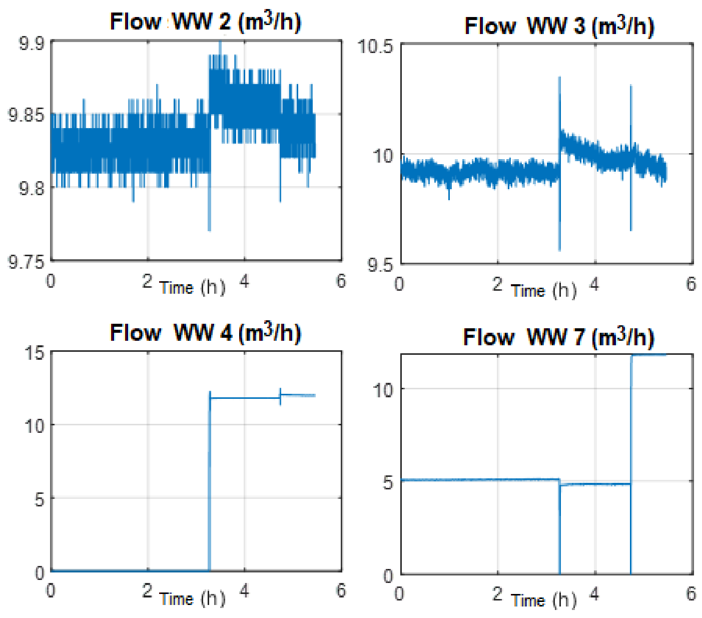


Fig. 4.2-4 Example of flow evolution from the water wells with fixed set-points.

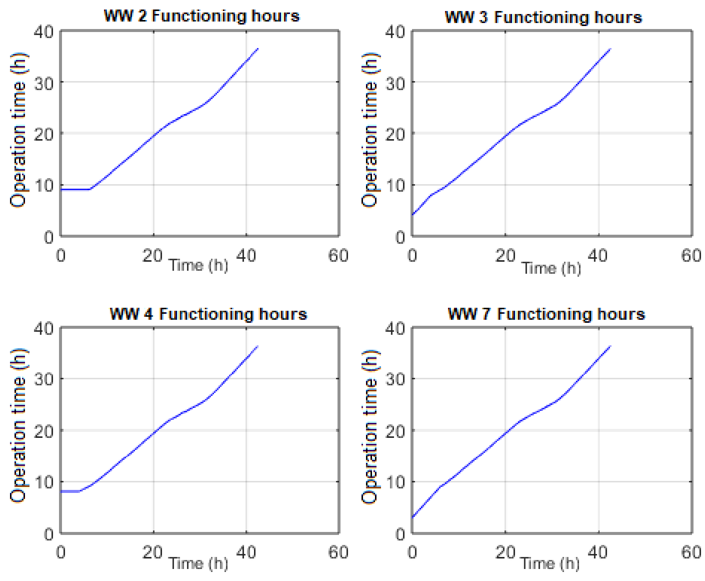


Fig. 4.2-5 Water wells functioning hours in the test scenario.

The second scenario involved two-week supervised test on the real system under constrained conditions, with fixed flow set-points for WWs. During extended data analysis, changes in local operation were observed, manual WW activation/deactivation driven by operator preferences or equipment faults. Over 1.5 years, operators typically switched WWs every 4–7 months.

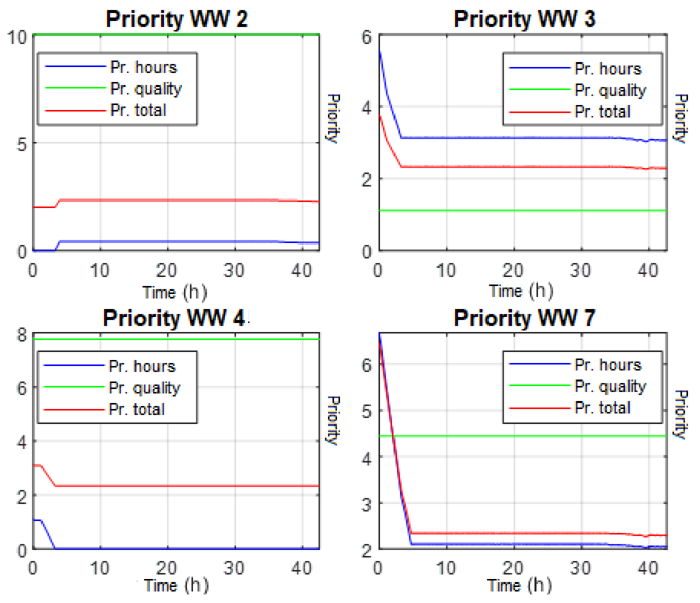


Fig. 4.2-6 Water wells priority indicators in the test scenario.

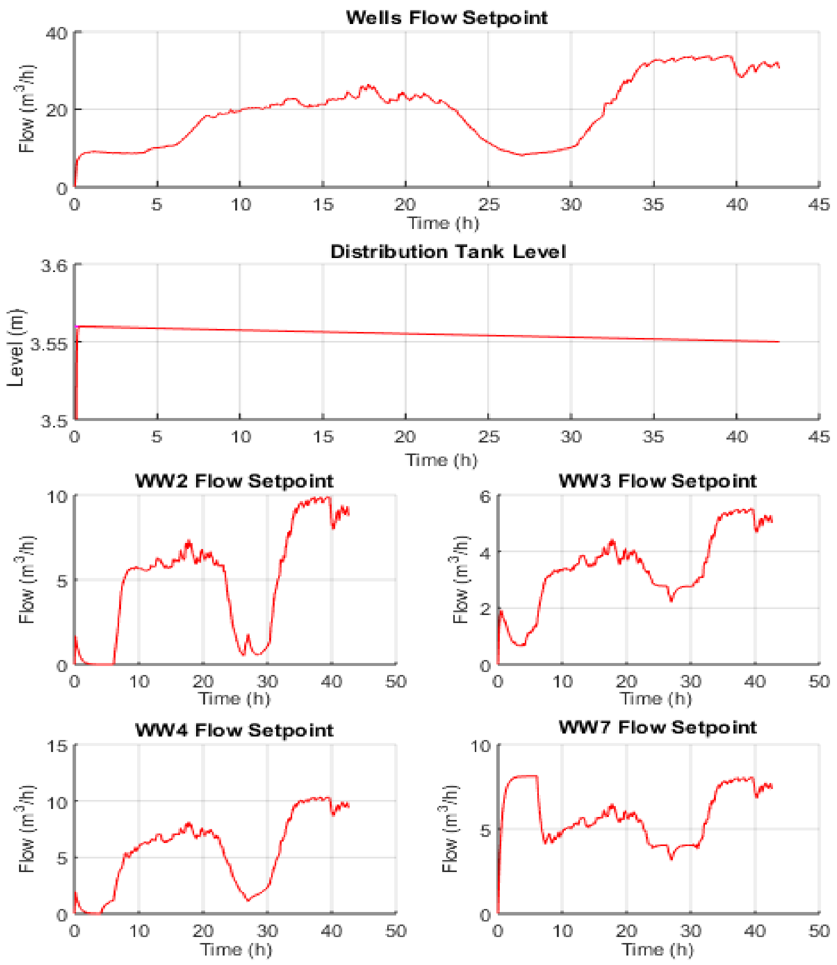


Fig. 4.2-7 The resulting evolution of: the total flow requested from the water wells, the level in the distribution tank, and the flow set-point for each water well.

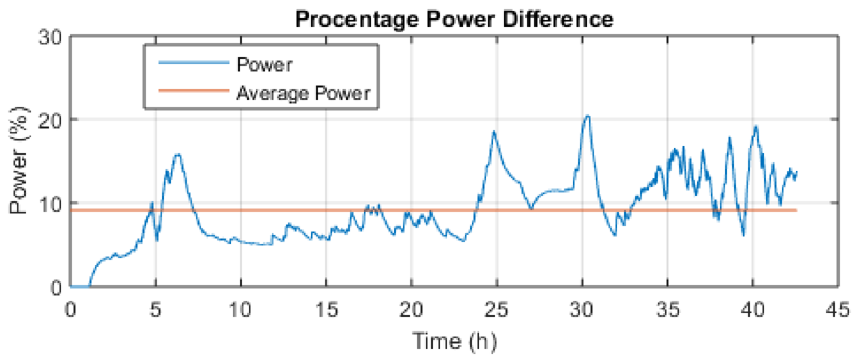


Fig. 4.2-8 Percentage power difference after using the solution.

It can be concluded that integrating a new water source requires at least four months of consistent data analysis before the proactive historian can correctly incorporate it into decision and control algorithms.

During the two-week period (23 November–07 December 2019), the flow evolution of four WWs (WW1, WW2, WW3, WW7) and the corresponding total energy consumption were recorded and stored. Fig. 4.2-9 shows the 1st week data. The flow evolution of four WWs and the corresponding total energy consumption were recorded over four weeks (11 January–08 February 2020) without applying the efficiency solution. For accurate comparison, these tests were conducted during a period with similar water demand and consumption as the initial two-week trial, excluding the winter holiday interval. Fig. 4.2-10 presents an example of the stored data for the 4th week.

Table 4-1 presents the weekly initial and final energy index values, weekly energy consumption, and average consumption for both the two-week period (2.7 MWh) and the four-week period (3.5 MWh). The percentage difference demonstrates the efficiency of the proposed solution, with the study identifying a consistent ~30% increase in energy consumption.

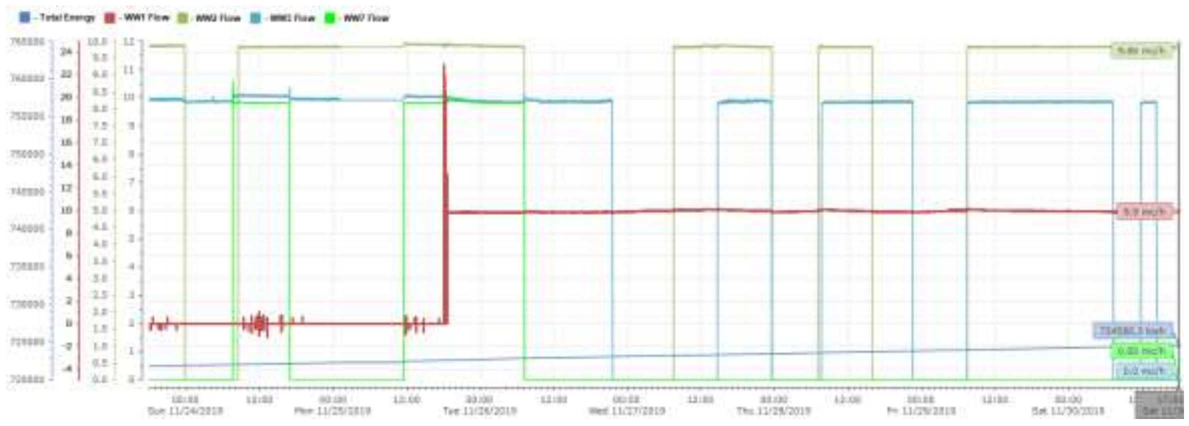


Fig. 4.2-9 Results with constrained solution. Well flows (mc/h) and energy consumption (kWh) in Week 1 of tests.

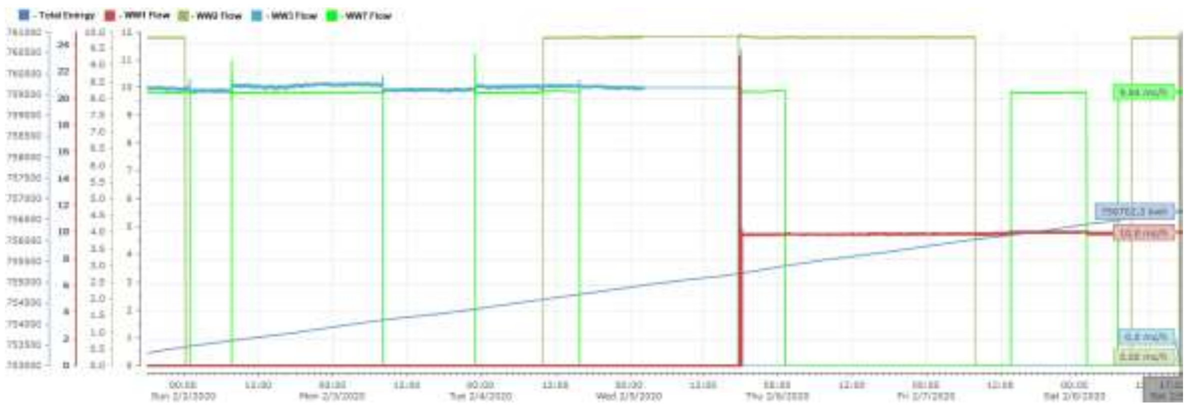


Fig. 4.2-10 Results without the solution. Well flows (mc/h) and energy consumption (kWh) in Week 4 of tests.

Table 4-1 Total energy consumption (MWh) of the drinking water facility (DWF).

	Two weeks with constrained FDC solution (23 Novembe 2019–07 December 2019)		**Four weeks without FDC solution (11 January 2020–08 February 2020)**			
	Week 1	**Week 2**	**Week 1**	**Week 2**	**Week 3**	**Week 4**
Init. val. (MWh)	722	724.6	742.7	746.3	749.8	753.3
Final val. (MWh)	724.6	727.4	746.3	749.8	753.3	756.7
Consumption (MWh)]	2.6	2.8	3.6	3.5	3.5	3.4
Average (MWh)	2.7		3.5			
Difference (%)	+30%					

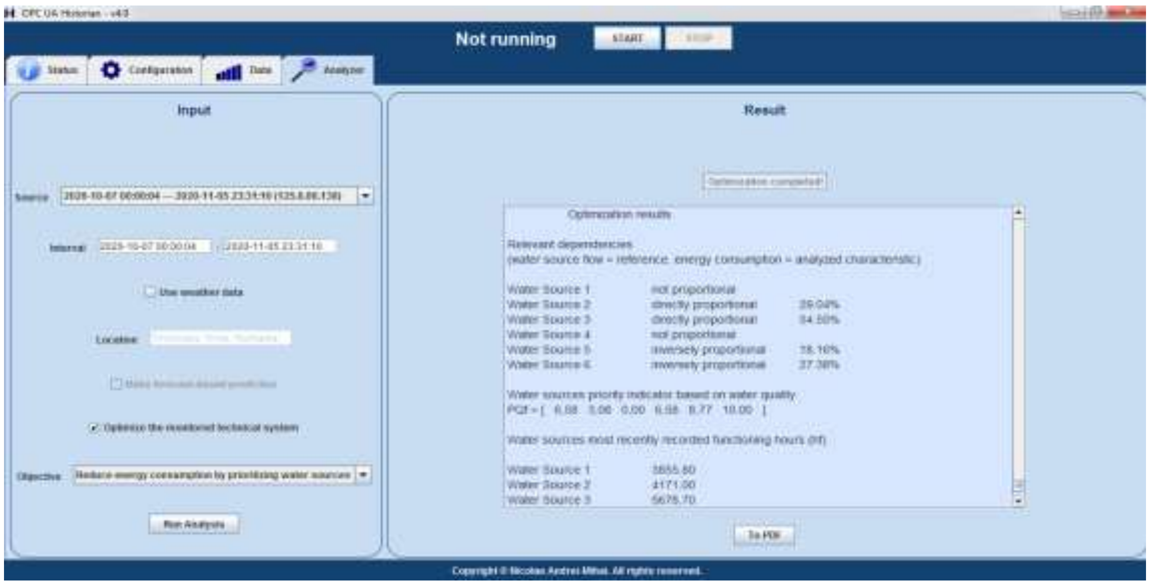
4.2.2 Proactive historian in complete solution and long-term testing for the energy reduction

Works [K-17], [K-12] were continuing the previous advances. The research contributes to the effort towards obtaining a proactive historian able to increase the efficiency of the supervised industrial system. The key contribution in [K-24] is the automatic identification and adaptation of water well quality indicators through continuous long-term analysis within the proactive historian. Further adjustments and testing of concepts were performed to advance the technological readiness level.

In practice, water quality varies across DWTP sources and changes over time. When a new source is commissioned, a technical datasheet is created based on laboratory analysis of sampled water. However, as quality evolves due to factors such as pollution or overuse, no sensor devices exist to update a general quality indicator. Operators often rely on empirical observations (e.g. noticing reduced equipment strain when requesting water from certain sources) but these are subjective and non-scientific. Thus, the proactive historian must continuously analyze operational data, derive reliable quality indicators, and react autonomously. By integrating flow distribution, operating times, water sufficiency, and maintenance factors, the historian can establish strategies that reduce energy consumption while optimizing source utilization.

The proactive historian must be flexible and adaptive, evolving with system requirements. An objective function (e.g. Fig. 4.2-11) should be defined, with appropriate constraints applied to guide optimization.

As detailed in [K-12], the proactive historian is designed to be process-aware. Local data is contextualized, process components are understood, causal relationships are identified, and the impact of actions is recognized. Fig. 4.2-12 illustrates the process-aware interface, with components, constraints, and the objective function. A key characteristic of the historian is flexibility, enabling modification of process components and constraints.



Source: 2020-10-07 00:00:04 — 2020-11-05 23:31:10 (125.8.86.138)

Interval: 2020-10-07 00:00:04 2020-11-05 23:31:10

☐ Use weather data

Location:

☐ Make forecast-based prediction

☑ Optimize the monitored technical system

Objective: Reduce energy consumption by prioritizing water sources

Optimization completed!

```
Optimization results

Relevant dependencies
(water source flow = reference, energy consumption = analyzed characteristic)

Water Source 1        not proportional
Water Source 2        directly proportional      29.04%
Water Source 3        directly proportional      34.50%
Water Source 4        not proportional
Water Source 5        inversely proportional     18.16%
Water Source 6        inversely proportional     27.38%

Water sources priority indicator based on water quality
PQI = [  6.88  3.06  0.00  6.88  8.77  10.00  ]

Water sources most recently recorded functioning hours (hf)

Water Source 1        3655.80
Water Source 2        4171.00
Water Source 3        5676.70
```

Fig. 4.2-11 Optimizing objectives choice inside the proactive Historian application.

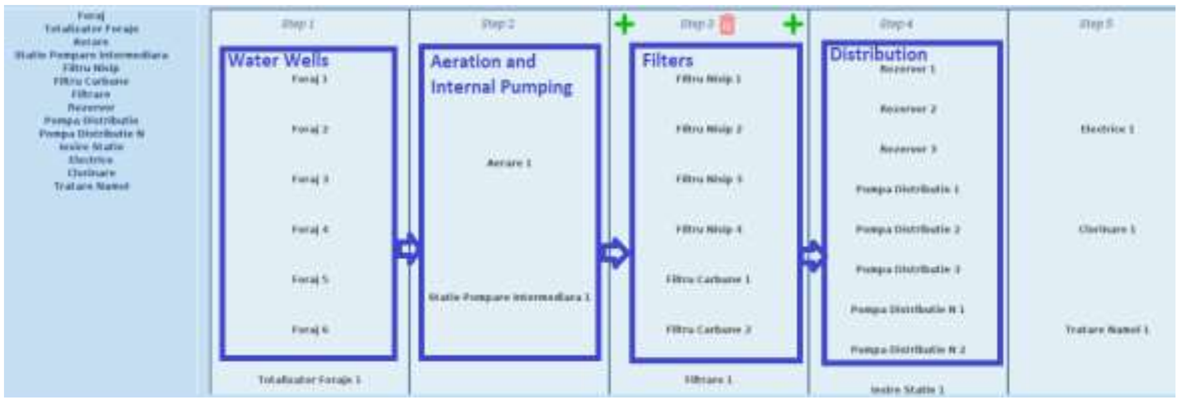


Fig. 4.2-12 Optimizing objectives choice inside the proactive Historian application.

Building on earlier work with the proactive decentralized historian, [K-12] examined a long-operated water treatment and distribution facility. Operators had established a local regime based on observed process changes, restrictions, and response strategies. For this case study, the historian was tailored and tested under a suboptimal scenario, where water sources were manually selected to balance availability and energy efficiency, without accounting for failures or demand variations. This scenario was chosen as a challenge: energy use was near minimal, and daily demand could be met by two wells operating close to optimal points.

The proposed low-cost historian aimed to improve facility operation by enhancing energy efficiency and addressing issues such as extended personnel hours, inability to meet rapid demand changes, and equipment faults from heavy use. Its design ensured non-invasive integration with legacy systems. The goal was to demonstrate the historian's ability to adapt to suboptimal industrial scenarios, generate process-aware recipes, and interoperate with legacy systems to apply improvements.

Over time, facility practices shifted. To minimize energy consumption, operators discontinued automatic activation and source selection based on reservoir levels and operating hours, due to difficulties in establishing flow set-points. This created new research challenges:

- Minimal room for further energy reduction: Operators already selected sources based on prior research, with some scenarios requiring only two wells to meet demand at near-optimal drive frequencies.
- Fixed flow set-points: Wells operated continuously at constant flows, benefiting from lower night tariffs but failing to adapt to demand variation or equipment faults, leading to wasted water or shortages.
- Uneven operating time distribution: Manual regimes caused wear and defects. One source could no longer sustain flow-based control, while another showed large fluctuations, further limiting efficiency gains and underscoring the need for full automation.

A further challenge was to extend historian testing over longer autonomous periods while maintaining non-invasive interoperability with legacy system.

The identified challenges were translated into historian tailoring and testing tasks, summarized as: Automatic well selection based on accumulated data, prioritizing energy efficiency while considering operating hours; Automatic activation of wells according to varying demand, peak-hour accumulation, variable flow set-points, and pump frequency constraints (upper, lower, optimal); Legacy system interoperability to set flow references and control pumps, with multiple checks for manual regimes, level-based operation, fault detection, and proper sampling periods; Safety procedures to deactivate historian-based automation in case of malfunctions or operator request, restoring previous local settings before decoupling; Performance evaluation under suboptimal regimes and extended operation periods.

The optimizing algorithm was refined to include a hysteresis factor (h), expressed as a percentage of the minimum source flow.

E.g. $h = \frac{1}{2} \cdot minimum\_next\_source\_flow\_setpoint$

If the difference between the total distributed flow and the sum of flows from active sources is less than ½ of the minimum flow of the next idle source (by priority), that source is not started, preventing pump wear. The same hysteresis rule applies when stopping a source. A fixed hysteresis value proved more efficient than variable values, which require frequent updates and correlation with source evolution but showed no benefit. The historian was updated to read minimum and maximum flows from a configuration file, allowing operators to adjust limits as they change over time.

Automation at the WTP uses OPC UA tags for pump start/stop commands rather than a 0-based flow reference convention, requiring algorithm adjustments to set these tags correctly. Manual well selection toward automation demanded additional condition checks and reactions to legacy system behavior. Specific OPC UA tags for start/stop commands, reference flows, and total delivered flow had to be defined and verified, significantly increasing the algorithmic and protection structures for system interoperation.

Finally, in the targeted WTP, filter washing occurs every 24 hours, consuming ~50 m³ of treated water from a tank with 400 m³ capacity, completed in ~30 minutes. To compensate for this rapid level drop, the historian was adjusted to compute the target water flow not as equal to the distribution flow, but as an augmented value accounting for filter washing operations.

$$flow\_required\_from\_sources = \boldsymbol{p\%} * DWTP\_output\_flow.$$

(where $p\%$ was fixed at 120%, but may vary between 110–130% depending on the treated daily volumes in the WTP, which fluctuate seasonally).

The tank's water level was maintained by a gradual rise compensating filter-cleaning drops, while experiments monitored washing cycle frequency and clogging status. The solution interfaced with the legacy system through two loops: Monitoring loop, where the historian collected OPC UA tag values every 20 s from the WTP server; Optimizing loop, where the historian retrieved the latest output flow, applied the optimization algorithm, and wrote updated reference flows and start/stop commands through OPC UA at 60 s intervals.

The historian was deployed on a Raspberry Pi 4 Model B in the WTP command room, connected to a UPS and accessible remotely via SSH tunneling. It autonomously analyzed stored data, generated optimization recipes, and applied them non-invasively. For testing, the historian operated in monitoring mode from 30 August 2022, with optimization tested during a 50-hour interval (27 February–01 March 2023). During this period, operators made no adjustments, leaving water source control entirely to the historian. The first evaluation focused on daily energy consumption, using three months of prior data. Monthly energy indexes were converted to daily averages, with results summarized in in Table 4-2, demonstrating energy reduction.

Table 4-2 Total energy consumption per day comparison during test with previous months

	December 2022	**January 2023**	**February 2023 01.02 - 27.02 (before test)**
Energy index start (kWh)	1252010,25	1266546,5	1281298,75
Energy index end (kWh)	1266546,25	1281298,625	1293673,25
Total energy consumed (kWh)	14536	14752,125	12374,5
Energy per day (kWh)	468,90	475,875	475,942
Energy/day (kWh) during test		454,38	
Comparison	- 14,52 kWh/day **- 3,1%**	- 21,495 kWh/day **- 4,51%**	- 21,562 kWh/day **- 4,53%**

The second analysis compared total energy consumption during the test interval with similar periods. Specifically, the same Monday 13:30–Wednesday 15:30 interval was examined across three of the four weeks preceding the test. In addition, comparable 50-hour intervals (Wednesday–Friday) were analyzed in both the week before and the week of the test. Across all five reference intervals, results consistently showed energy consumption reductions when applying the optimizing strategy, as illustrated in Table 4-3.

Table 4-3 Total energy consumption comparison during test with other similar 50-hours long intervals

	30.01.2023 13:30 - 01.02.2023 15:30 (Monday – Wednesday)	**06.02.2023 13:30 - 08.02.2023 15:30 (Monday – Wednesday)**	**13.02.2023 13:30 - 15.02.2023 15:30 (Monday – Wednesday)**	**22.02.2023 13:30 - 24.02.2023 15:30 (Wednesday – Friday)**	**01.03.2023 15:30 - 03.03.2023 17:30 (Wednesday – Friday)**
Energy index start (kWh)	1280569,5	1283990,5	1287399	1291612,875	1294833
Energy index end (kWh)	1281572,5	1284977	1288386,5	1292580,5	1295789,5
Total energy consumed (kWh)	1003	986,5	987,5	967,625	956,5
During test (**27.02.2023 13:30 - 01.03.2023 15:30 Monday - Wednesday**)					
Energy index start (kWh)			1293886,25		
Energy index end (kWh)			1294832,875		
Total energy consumed (kWh)			946,625		
Comparison	- 56,375 kWh **- 5,95%**	- 39,875 kWh **- 4,21%**	- 40,875 kWh **- 4,31%**	- 21 kWh **- 2,22%**	- 9,875 kWh **- 1,03%**

The third analysis evaluated the water volume entering the WTP, using energy consumption/water volume as the metric. Two comparison intervals were considered: the entire week preceding the test and a similar 50-hour interval. Results, summarized in Table 4-4, demonstrate effective energy consumption optimizations during the test period.

Table 4-4 Total energy consumption per m3 of water entering WTP comparison during test with previous intervals

	20.02.2023 00:00:00 – 27.02.2023 00:00:00 (the week before test)	**22.02.2023 13:30 – 24.02.2023 15:30**
Total energy consumed (kWh)	3156,125	967,625
Total water volume entering DWTP (m^3)	2813,8	816
Total energy / water volume (kWh / m^3)	1,121	1,186
During test (**27.02.2023 13:30 - 01.03.2023 15:30**)		
Total energy consumed (kWh)	946,625	
Total water volume entering DWTP (m^3)	882,9	
Total energy / water volume (kWh / m^3)	1,072	
Comparison	- 0,049 kWh/m^3 **- 4,37%**	- 0,114 kWh/m^3 **- 9,61%**

Under these conditions, the results were consistent. No stability issues were observed during the 50-hour test: the historian software ran without errors or interruptions, interoperability with the monitored system was seamless, and the WTP operated under normal quality parameters. The distribution tank remained safely above risk limits, even after filter washing, and no operator intervention was required. Consequently, the historian demonstrated a high TRL level in this approach.

Replacing manual source selection with the proactive historian enables system adaptation to diverse water needs, including population demand, equipment failures, poor source quality, clogged filters, or pipe breaks. The historian adjusts water source flows within 60 seconds of sudden changes, ensuring responsive and efficient operation. Another observation concerns equipment wear. Overuse of individual sources is mitigated by considering their operating hours, yet this hypothesis requires long-term validation of the solution.

4.3 Long Short-Term Memory-Based Prediction Solution Inside a Decentralized Proactive Historian for Water Industry 4.0.

The current section is showcasing the study from [K-7]. Local systems in the water sector are typically centralized within SCADA control centers at local, regional, and central levels. However, as information ascends the hierarchical pyramid, it becomes filtered and generalized, reducing knowledge of local processes and limiting the ability to react automatically at the operational level. Current digital transformation research, aligned with Industry 5.0 principles, emphasizes decentralization and increased edge-level processing. Consequently, solutions that exchange process data and implement LSTM neural network strategies should be deployed at the edge/fog level, close to SCADA systems or automation panels.

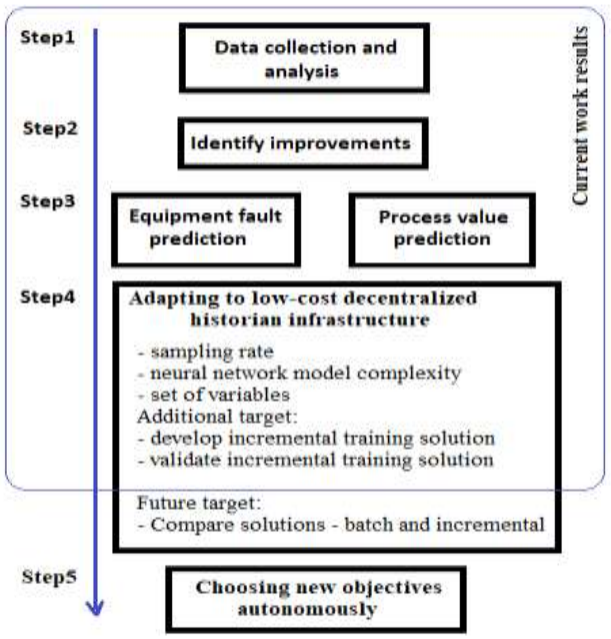
The proactive historian is designed as a low-cost decentralized solution capable of interfacing with legacy systems, collecting and analyzing data, identifying dependencies, and achieving objective functions under constraints in a process-aware manner. It can act directly on local automation to implement optimized recipes. This approach opens significant research opportunities in the water sector, particularly when industrial data is applied to specific scenarios and process structures, requiring tailoring and long-term testing. An intensive project to develop and validate the proactive historian in real scenarios began in July 2022, with research steps and current limitations outlined in the study (see Fig. 4.3-1). Pilot structures were initiated within operational drinking water and wastewater legacy systems, with proactive historians tailored for specific scenarios.

This section addresses prediction, a critical element for meaningful improvements that must be grounded in real systems. Two prediction targets were defined: Primary target - equipment fault prediction, a key industrial research requirement; Secondary target - prediction of analog process values to enable non-invasive control adjustments in existing automation. This need arises from the high time constants in certain treatment processes, where immediate control changes cannot achieve desired effects. Such situations are common in the biological phase of wastewater treatment and in drinking water treatment (e.g. chlorine correction).

In Fig. 4.3-1, the third step of the research divides into two branches of the prediction algorithm, followed by a fourth step adapting the strategy to the low-cost historian infrastructure:

- Step 3 – Branch 1: Fault prediction. Fault prediction required datasets containing equipment faults. All process equipment was monitored, with experts identifying critical elements and time objectives. The chosen method was an LSTM recurrent neural network (NN), applied through: training on the initial dataset, validation with a second dataset, further validation using newly acquired data.

- Step 3 – Branch 2: Process value prediction. Similar actions were applied to stateless variables, with the algorithm tailored for analog value prediction to support non-invasive process control.



Step1
Data collection and analysis
Step2
Identify improvements
Step3
Equipment fault prediction
Process value prediction
Current work results
Step4
Adapting to low-cost decentralized historian infrastructure
- sampling rate
- neural network model complexity
- set of variables
Additional target:
- develop incremental training solution
- validate incremental training solution
Future target:
- Compare solutions - batch and incremental
Step5
Choosing new objectives autonomously

Fig. 4.3-1 Optimizing objectives choice inside the proactive Historian application.

Step 4 is the Adaptation to low-cost decentralized historian infrastructure. The prediction strategy was adjusted to account for: Reduced sampling rate, limiting data volume and processing time; Simplified NN model complexity; Smaller variable sets. An additional research goal was to enable independent, automatic algorithm improvement by developing and validating incremental training within the historian. Future work will compare on-condition batch training with incremental approaches.

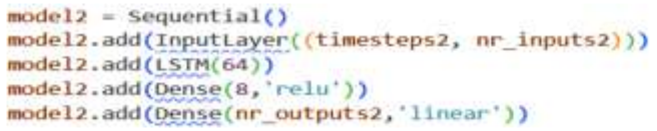
Step 5 is the Autonomous objective selection. Further research will integrate additional AI techniques to: Define prerequisites for fault prediction (e.g. minimum fault counts) and process value prediction (e.g. values improving time-constrained processes); Identify appropriate variable sets for chosen tag values; Autonomously validate incremental training; Establish conditions and constraints for the historian to evaluate, validate, and deploy new algorithms alongside legacy systems, with safety, ethical analysis, upgrade procedures.

4.3.1 Prediction Solution in the Proactive Historian

For implementation, Microsoft Visual Studio Code was used with Python 3. Data structuring and visualization employed Pandas, NumPy, Matplotlib, Scikit-learn (train_test_split), and Beautiful Soup. The AI model was developed using TensorFlow (Sequential, Layers, MeanSquared, Adam) and

Scikit-learn (MinMaxScaler, Normalize, Accuracy_score). The next step was to select the most suitable neural network model for predicting future values. Since the dataset consists of time-ordered sequences, the problem was framed as time series forecasting. An LSTM recurrent neural network was chosen for its ability to capture long-term dependencies, selectively retain or discard information via input, output, and forget gates, and handle non-linear, non-stationary data. Sequential processing preserves event order, making LSTM particularly appropriate for this case.

Two generic LSTM models were developed: a complex model (~100,000 parameters) and a simpler model (~19,000 parameters). Subsequent prediction tests showed that the simpler model (see Fig. 4.3-2) achieved equal or superior performance compared to the complex one.



```
model2 = Sequential()
model2.add(InputLayer((timesteps2, nr_inputs2)))
model2.add(LSTM(64))
model2.add(Dense(8,'relu'))
model2.add(Dense(nr_outputs2,'linear'))
```

Fig. 4.3-2 The simpler model details.

The simpler sequential model comprises the following layers:

- *InputLayer* – defines input format, with size determined by *timesteps* (sequence length) and *nr_inputs* (features/OPC UA tags).
- *LSTM* – 64 units capture temporal dependencies and complex sequential relations in the data.
- *Dense* – 8 units with ReLU activation, introducing non-linearity by zeroing negative values and retaining positives, aiding non-linear feature learning.
- *Dense* – *nr_outputs* units with linear activation, generating unrestricted continuous predictions.

An important perspective is the integration of Incremental Training to enable automated learning and improve model accuracy after real-world deployment. This approach allows the AI to adapt to evolving data and adjust predictions accordingly, but mechanisms must be enforced to prevent negative impacts on accuracy. Hardware capabilities must also be evaluated to ensure feasibility compared to models without incremental updates.

The architecture is structured as a loop managed by the historian's Java application: Data extraction from the SQLite database into a CSV file, covering the most recent 15 hours at 1-minute intervals; Execution of the LSTM model via Python, loading both the input data and the TensorFlow model. Predictions generated here also represent the incremental learning step; Processing and storage of predictions back into the SQLite database, forming the basis for subsequent optimization actions.

Adaptive mechanisms include data preprocessing, rolling averages, moving medians, real-time anomaly detection, retraining (incremental and periodic full training), and continuous monitoring with KPIs such as accuracy and precision. Incremental Training minimizes latency, requires fewer resources per update, and scales better than batch training with large datasets. However, it demands constant computational resources and lacks advanced optimization techniques available in batch training. A hybrid approach, batch training during off-peak times combined with incremental updates, ensures scalability, efficiency, and responsiveness. Additional strategies include dynamic learning rate adjustment, sliding window methods, prioritized memory updates, and regular mini-batch updates. Concept drift can be detected using statistical tests such as the Page-Hinkley test or ADWIN.

4.3.2 Prediction Case Study and Results

The WWTP case study serves ~6,000 inhabitants and follows a classical sequential process. Two treatment lines are implemented: after mechanical inlet treatment, biological treatment occurs in two sequential batch reactors with time-based aerobic and anoxic phases. The plant also includes sludge and bypass lines. Control is managed by nine S7-1200 PLCs connected to a redundant SCADA WinCC V13 system via fiber optic ring. The pilot structure integrated the proactive historian through two OPC UA servers from SCADA.

Deployment revealed two issues: Frequent sludge pump failures in biological reactors, disrupting continuity and maintenance; Inadequate response of biological treatment to sudden CODcr variations, requiring adaptation time.

Predicting pump faults at least 3 hours ahead and CODcr fluctuations at least 1 hour ahead would enable timely corrective actions. For implementation, data preprocessing was required. The historian, storing data since July 2022, monitored 460 OPC UA tags at ~20 sec. intervals. After filtering, data was structured for LSTM modeling. Using Scikit-learn's *train_test_split*, datasets were divided into 63% training, 30% testing, and 7% validation.

The first practical application focused on predicting sludge pump status at the WWTP, aligned with the predictive maintenance objective of the research. The studied pump, one of two handling sludge disposal from the first biological reactor, had its status recorded by the historian (see Table 4-5). The input data for training the generic LSTM neural network consisted of the selected OPC UA tags listed in Table 4-6. These characteristics were essential, as adding or removing tags affected model optimality. The pump status served as the output variable. The sample size was set to 30 time steps, enabling predictions over 5 future hours, with 10 epochs applied to the database.

Table 4-5 Mapping of sludge pump values to their significance

Value	Significance
0	Unknown status
1	Fault
2	Not running
3 and 4	Normal running
9	Warning

Table 4-6 List of characteristics for sludge pump failure prediction

Characteristic significance	Measurement unit	Type of value
Pump status	See Table 1	Numerical integer
Pump automatic mode	-	Boolean
Pump remote command	-	Boolean
Pump frequency	Hz	Numerical float
Pump power	kW	Numerical float
Pump current	A	Numerical float
Pump energy meter	kWh	Numerical integer
Biological reactor sludge stabilization flow	m^3/h	Numerical float
Biological reactor sludge stabilization volume	m^3	Numerical float

The LSTM configuration used a learning rate of 0.001, batch size of 32, 100 epochs, and Mean Squared Error (MSE) as the loss function. Training and testing were conducted on two scenarios, differentiated by the time intervals selected for the datasets. Graphical results illustrating the sludge pump status prediction for one of the tested scenarios are presented in Fig. 4.3-3 (with the actual state of the tags) and Fig. 4.3-4 (only the prediction).

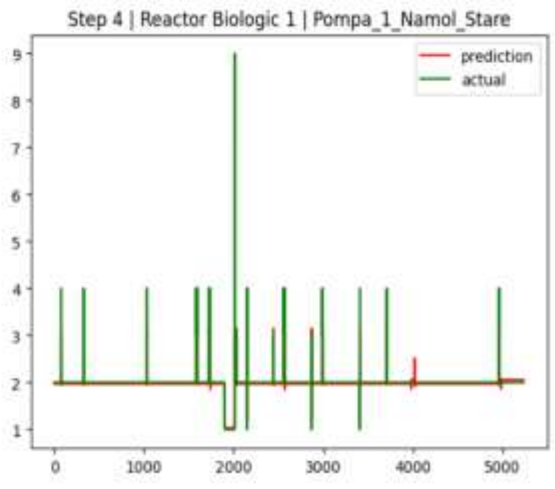


Fig. 4.3-3 Pump state and prediction.

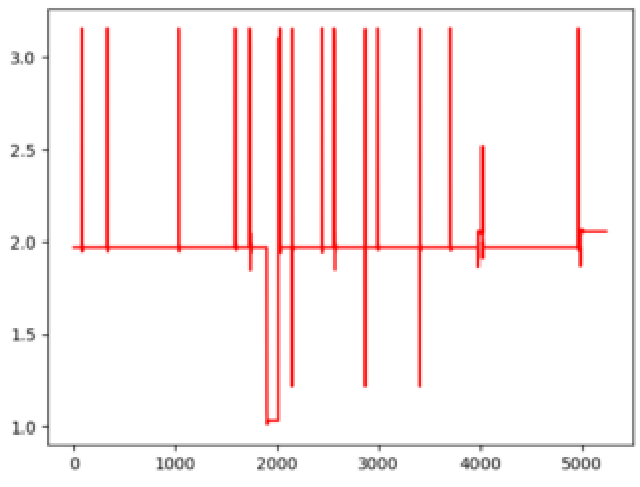


Fig. 4.3-4 Prediction only.

Model accuracy was evaluated using four approaches: *r2_score* (Scikit-learn) to compute the coefficient of determination with zero error margin; MSE and

RMSE as standard error metrics; A final method applying a 0.5 error margin for integer pump status values, allowing extrapolation to compensate prediction deviations. All approaches were applied across both scenarios, with results summarized in Table 4-7, confirming strong model accuracy.

Table 4-7 Accuracy of trained models – sludge pump prediction

Evaluation Approach	Scenario 1	Scenario 2
Coefficient of Determination (R^2)	0.546	0.546
Mean Squared Error (MSE)	1.056	1.056
Root Mean Squared Error (RMSE)	1.027	1.027
Accuracy (correct/total with 0.5 error margin)	98.625 %	96 %

The second practical application focused on predicting the *CODcr* quality indicator, using the same generic LSTM neural network as the starting point. This application pursued a secondary objective: optimizing legacy process control. The selected input characteristics from the available data, used to train the LSTM model, are listed in Table 4-8.

Table 4-8 List of characteristics used as input data for neural network training – CODcr prediction at WWTP inlet

Characteristic significance	Measurement unit	Type of value
CODcr	mg/l	Numerical float
Phosphate (PO4)	mg/l	Numerical float
Ammonium (NH4)	mg/l	Numerical float
Acidity (pH)	-	Numerical float [0-14] interval
Temperature	Celsius degrees	Numerical float
Flow from septic trucks	m3/h	Numerical float
Volume from septic trucks	m3	Numerical float

The LSTM configuration employed a learning rate of 0.01, batch size of 32, 50 epochs, and MSE as the loss function. Results are presented in Fig. 4.3-5 (predicted vs. actual values) and Fig. 4.3-6 (predictions only). Following the accuracy evaluation, the model results were presented in Table 4-9.

Table 4-9 Accuracy of the trained model – CODcr prediction at the WWTP inlet

Evaluation Approach	Result
Coefficient of Determination (R^2)	0.422
Mean Squared Error (MSE)	173.535
Root Mean Squared Error (RMSE)	13.173
Accuracy (with Scikit-learn library)	97.237 %

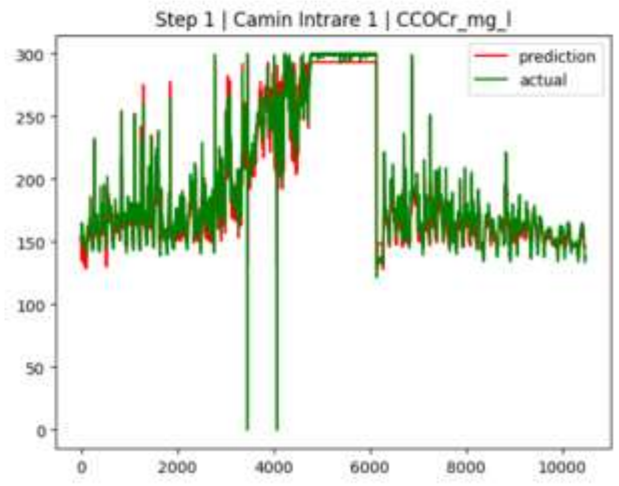


Fig. 4.3-5 Actual value and prediction.

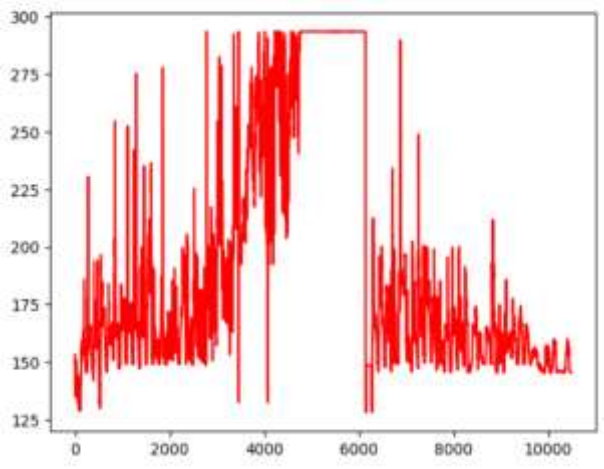


Fig. 4.3-6 Value prediction only.

4.4 Non-Invasive Control Solution for Energy Efficiency in Wastewater Treatment Plants.

Study [K-24] highlighted the interoperability of the historian, demonstrating its ability to augment functional systems with external data. Weather information was integrated and analyzed through graph-based dependency methods to predict wastewater treatment values, showing how systems can connect and interoperate.

Work [K-36] addressed energy efficiency in WWTPs, noting that most automation follows a standard oxygen regulation pattern for nitrification/denitrification, supported by two-positional air pressure control.

Improving energy efficiency remains a major concern in wastewater treatment [161]. Companies monitor specific consumptions (energy and substances per m^3 of treated water/sludge). Blowers are the largest energy consumers. Based on data analysis, and recognizing oxygen's role in reducing ammonium while limiting nitrate increase, a Model Predictive Control (MPC) strategy was applied to optimize oxygen use. MPC has a strong theoretical background in the water industry [162]. Within IIoT and Industry 4.0, invasive interventions in existing plants are avoided due to warranties, documentation gaps, and infrastructure risks. Interoperability enables higher-level control structures without altering local automation, making it a cornerstone of smart water management [163]. With OPC UA servers widely deployed, the proposed MPC strategy was designed as noninvasive.

The research in [K-36] proceeded in two phases: First - plant data analysis, model design and calibration, MPC algorithm development, simulation with real inputs, OPC UA interfacing, and Node-RED structuring; Second (planned) - detailed implementation, parameter refinement, long-term real-time testing, results analysis, and validation of the developed structure.

Without detailing the benchmark simulation model, the higher-level control structure was implemented as illustrated in Fig. 4.4-1.

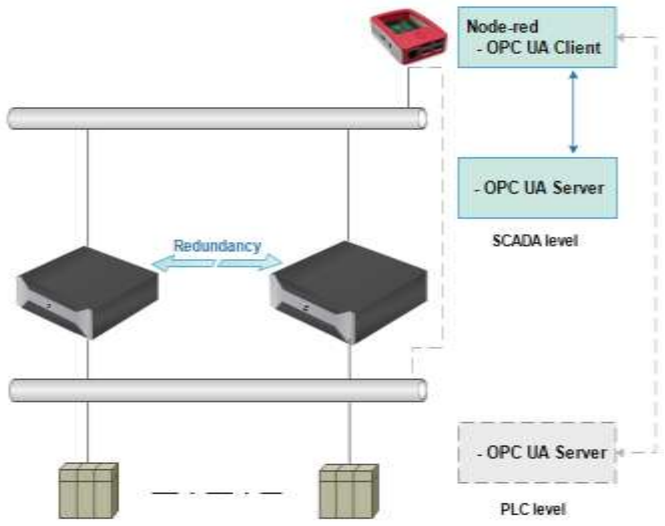


Fig. 4.4-1 The simpler model details.

The MPC higher-level control scheme for the WWTP is shown in Fig. 4.4-2. To enhance performance, two measurable disturbances were considered: influent flow rate (Q_0) and influent ammonium concentration (NH_4). A linear process model incorporating these disturbances was required to implement the feedforward component of the MPC strategy. The manipulated variable was the set-point of the in-plant closed-loop dissolved oxygen controller.

All simulations were conducted in Matlab Simulink using real BSM1 plant parameters (tank dimensions, external recycle flow, sludge wastage flow). The full plant's differential equations were solved with a 4th-order Runge–Kutta routine, employing a fixed integration step of 0.005 hours.

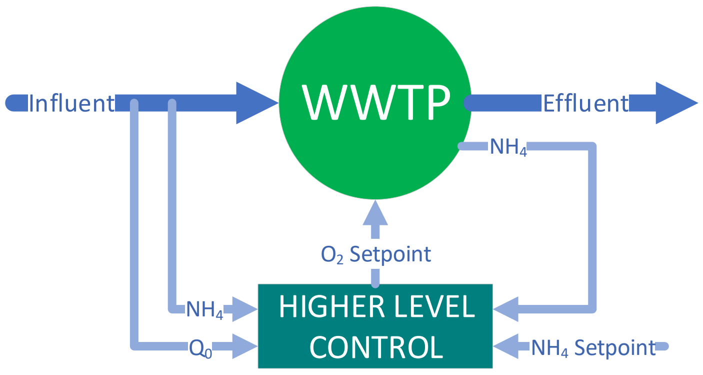


Fig. 4.4-2 Inputs and outputs of the application.

The algorithm implementation comprised two components: OPC UA interfacing and the MPC algorithm. Through the OPC UA Client, once connected to the server, the application retrieved WWTP variables including inlet/outlet flow, inlet/outlet ammonium, aeration basin oxygen and ammonium, and the high/low oxygen limits of the two-positional controllers. In addition, failure tags from flowmeters and ammonium/oxygen sensors were monitored, as they could impact system performance. After subscribing to the OPC UA tags, Fig. 4.4-3 illustrates the inlet flow of the WWTP over 8 hours on 27 Oct. 2017.

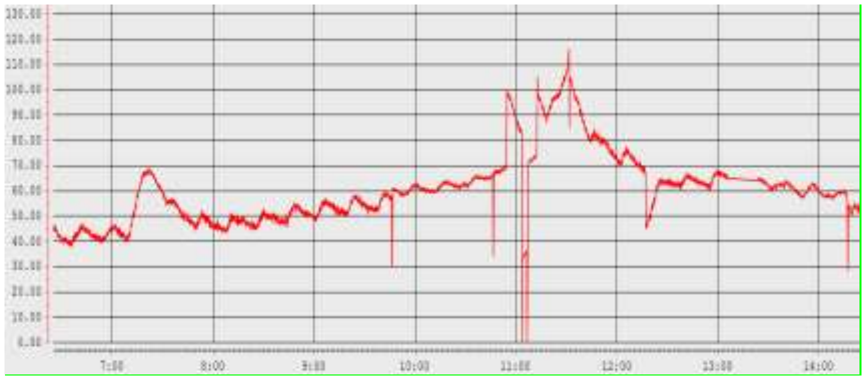


Fig. 4.4-3 WWTP inlet flow [m3/h] for 8 hours on the 27th of October 2017.

After processing and validating the subscribed variables at each sampling period, the MPC algorithm outputs a two-element array that adjusts the high and low limits of the two-positional oxygen controller with hysteresis. Fig. 4.4-4 illustrates the extraction of these oxygen limit values from the MPC decision, the configuration of namespace and tag names, the value insertion into the OPC UA server, and the hysteresis display in the dashboard.

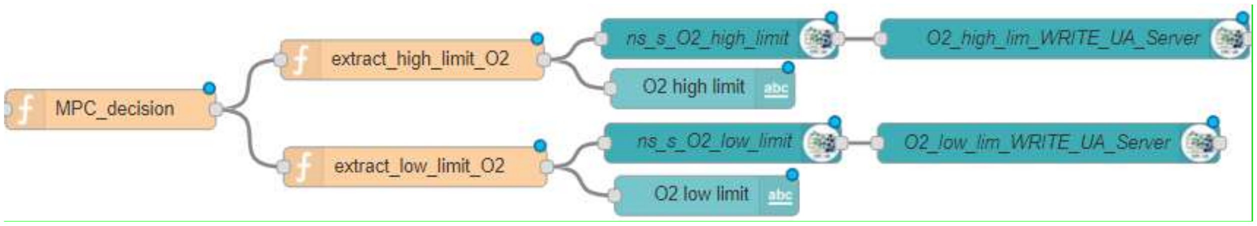


Fig. 4.4-4 Oxygen high and low limit value insertion to the OPC UA Server.

To apply linear MPC in wastewater treatment control, the BSM1-based model required calibration and validation. The system's dynamic response was obtained after running the benchmark for 100 days under normal conditions, ensuring steady-state values. Simulations combined BSM1 operational data with real plant data, including influent/effluent concentrations (NH_4, NO_3, P), pH, and flow rates (influent, effluent, recycle, wastage) over 30 days, of which 28 days were used in benchmark runs. BSM1 data corresponded to a dry-weather scenario with a 15-minute sampling period.

Initial open-loop tests showed that 14 days were sufficient to capture system dynamics. Consequently, 50% of the data was used for model calibration, and the remaining 50% for validation. Figure 4.4-5 presents the real data inputs employed in the validation benchmark (14 days).

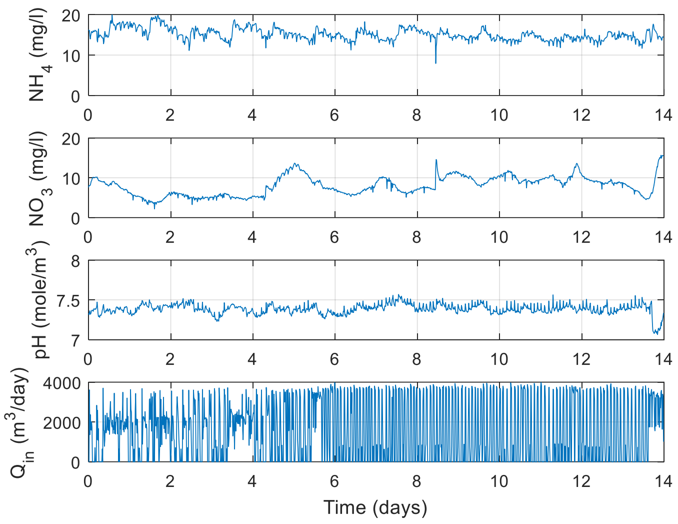


Fig. 4.4-5 Model validation data input (wastewater treatment plant data)

A comparison of process and model dissolved oxygen concentrations in the bio-reactor tank for days 8 and 9 is shown in Fig. 4.4-6.

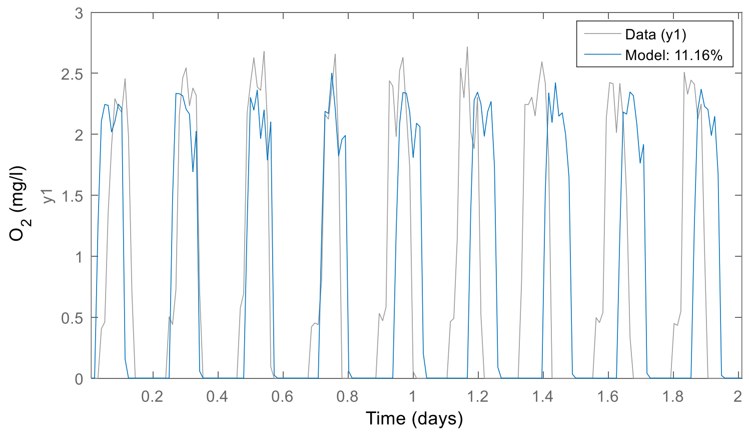


Fig. 4.4-6 Comparison between data and model oxygen concentration in bio-reactor tank.

The fit (F) is calculated, where y/ $\hat{y}$ is the validation data/model output.

$$F = 100\left(1 - \frac{\|y - \hat{y}\|}{\|y - mean(y)\|}\right)$$

The model achieved a fit of 11.16%, yet given the dynamics of biological nitrification and denitrification, the calibration is considered an acceptable representation of process behavior. Using real WWTP data, results indicate that aeration energy savings are feasible without compromising purified water quality. Fig. 4.4-7 presents mean daily influent and effluent ammonium concentrations over 30 days. Legal effluent limits are 4 mg/L ammonium and 18 mg/L total nitrogen (sum of nitrate/nitrite and Kjeldahl nitrogen). A clear relationship emerges: higher effluent ammonium concentrations correspond to lower aeration energy consumption per m³ of treated wastewater. The same applies to total nitrogen. Thus, oxygen supply can be reduced, achieving energy savings while maintaining compliance with regulatory thresholds.

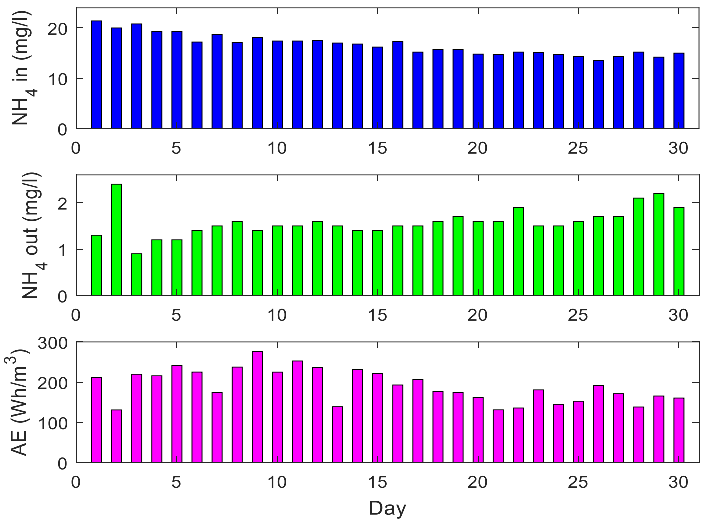


Fig. 4.4-7 Influent/effluent mean day ammonia nitrogen concentration and Aeration Energy (AE) consumed for ammonia nitrogen removal from wastewater.

BSM1 simulations confirmed the concept. Fig. 4.4-8 presents results for two oxygen set-points, showing that lower O_2 during nitrification increases effluent ammonium concentration, yet it remains below the regulatory limit.

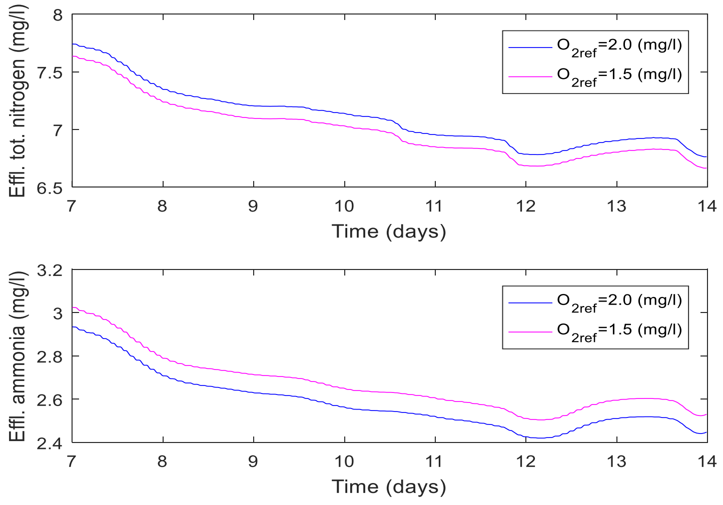


Fig. 4.4-8 Effluent ammonia and total nitrogen concentrations for two O2 concentration set-points.

At an oxygen set-point of 2 mg/L, aeration energy consumption was 253.66 kWh/day, decreasing to 236.36 kWh/day at 1.5 mg/L, showing a 7% reduction. Employing a variable reference, adjusted according to effluent ammonium concentration, enables significant aeration energy savings.

4.5 Non-Invasive Control Solution inside Higher-Level OPC UA based Wrapper for Optimizing Groups of Wastewater Systems.

The current chapter presents the research from [K-32]. Water distribution companies continuously develop local automation and SCADA systems for new objectives or refurbishments. These projects are executed by entrepreneurs under specific contracts: WWPS contracts typically follow the FIDIC Red Book, while WWTP contracts follow the FIDIC Yellow Book. Thus, WWPSs are implemented according to tendered technical designs with limited execution flexibility, whereas WWTPs allow entrepreneurs to propose their own technical concepts. Local automation/SCADA solutions result from diverse equipment and vendor-specific implementations, while legacy systems often rely on phased-out or proprietary technologies. According to [164], water control structures face interoperability issues due to non-standard SCADA interfaces, leading to integration problems in consumption, distribution, identification, and maintenance. As noted in [165], water and wastewater networks require retrofitting, extension, and maintenance for functional optimization, with flexibility and interoperability being critical in industrial environments.

Interoperability does not guarantee interoperation. WWPSs are typically cascaded, with outputs feeding WWTPs, creating process interdependencies. However, analysis shows that entities generally operate independently due to differing contracts and requirements. Two major problems were identified:

- Storm water flows disrupt WWTP processes, causing bypass use, financial losses, energy costs, and potential pollution.
- WWPS blockages from foreign materials or electrical faults can cascade upstream, leading to public wastewater overflows.

Optimization is possible if WWPSs and WWTPs are treated as groups with interoperation. Given warranties, maintenance contracts, missing documentation, proprietary constraints, invasive changes to local systems should be avoided. A non-invasive control structure, capable of manipulating local variables and enhancing existing systems, reduces implementation time, cost, and downtime. The proposed solution is modularly implemented within an OPC UA-based wrapper, integrated into the decentralized historian, and designed to optimize WWPS and WWTP behavior collectively.

4.5.1 Group Control Solution for WWPS-WWTP

The proposed solutions address the two identified problems by considering the in-series sewage system structure (WWPSs–WWTP) shown in Fig. 4.5-1. The architecture consists of a WWPS network linked to the WWTP, where each station collects wastewater and pumps it to the next, with WWPS 1 serving as the feeder to the treatment plant.

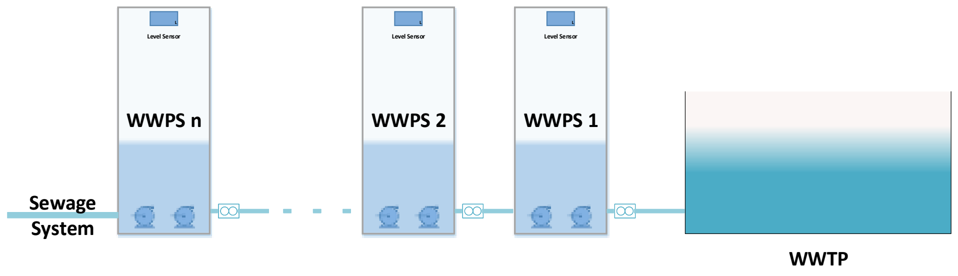


Fig. 4.5-1 Sewage system –WWPSs – WWTP architecture.

Optimizing WWPS–WWTP group control requires designing two higher-level control strategies (HLCS) that address the identified problems without altering local systems. These strategies are tailored to the specific WWPS type (frequency converter or direct/soft starter). In both approaches, WWPSs act as buffer tanks, with two objectives: Reducing WWTP influent flow during excess wastewater from pluvial water; Limiting discharge rates toward a blocked WWPS to prevent overload.

When a fault condition is detected, the application shifts the system into a fault state, activating higher-level supervisory control. The proposed solution modifies the start/stop set-points of the WWPS group, achieving a non-invasive control strategy (see Fig. 4.5-2). The *WWPS$_i$* manipulated variables are pumps start level (*Li_ON*), pumps stop level (*Li_OFF*) and pumps frequency (*F_FCi*) for WWPSs equipped with FCs (Frequency Converter). The controlled variables are the level or the flow rate. The HLCS monitors also the actual wastewater level (*L$_i$*), the flow if available (*F$_i$*) and two fault generator parameters (*FLT$_i$*) - electrical powering fault and emergency button status.

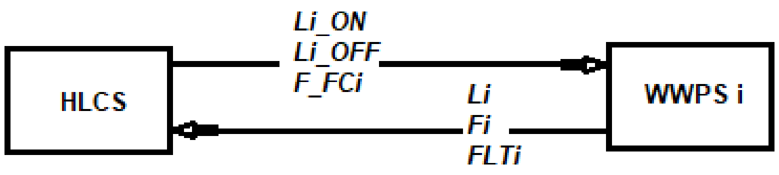


Fig. 4.5-2 HLCS interaction with WWPS from a WWTP–WWPSs network.

The optimization concepts for WWPS–WWTP groups are presented separately, reflecting the distinct nature of the two identified problems. Key variables are defined for clarity: Li_ON_high / Li_OFF_high – elevated pump start/stop levels for WWPSi; Li_ON_normal / Li_OFF_normal – normal pump start/stop levels for WWPSi; WWTP_inlet_max_flow / WWTP_inlet_min_flow – maximum/ minimum influent volumes over a fixed period; WWTP_inlet_actual_flow – current influent flowmeter reading; WWPSi_power_failure / WWPSi_emg – electrical failure and emergency button states; WWPSi_faulty_state – faulty condition of WWPSi (see Fig. 4.5-1).

Supplementary wastewater flow volume scenario consists of: A fault is detected when WWTP_inlet_actual_flow > WWTP_inlet_max_flow and L1 > L1_ON, caused by storm water entering the WWPS network. This activates WWPS1_faulty_state. If any upstream WWPS(i-1) is faulty and Li-1 > Li-1_OFF_high, then WWPSi_faulty_state is also triggered.

Higher-level control steps are described in the following lines. For WWPS 1 with frequency converters (FCs), a flow-based closed-loop PI control maintains influent flow at the WWTP nominal design rate, with FC frequency as the control signal. Once pumps are active, L1_ON shifts to L1_ON_high. If supplementary inflow forces pump frequency below 30 Hz, L1_OFF is raised to L1_OFF_high, stopping pumps. If L1 > L1_OFF_high, low-frequency protection reduces operating frequency to zero. If L1 > L1_ON_high, pumps run at maximum frequency until L1 < L1_OFF_high. When pumps stop and WWTP_inlet_min_flow is reached, L1_ON and L1_OFF revert to normal levels under closed-loop control.

For WWPS 1 with direct or soft start, L1_ON and L1_OFF are first shifted to L1_ON_high and L1_OFF_high, stopping the pumps. When pumps are off and WWTP_inlet_min_flow is reached, the levels revert to L1_ON_normal and L1_OFF_normal.

In upstream WWPSs (i > 1) entering a faulty state, level control maximizes storage capacity: Li_ON and Li_OFF move to high values, stopping pumps. For WWPSs with frequency converters (FCs), a closed-loop level control maintains the maximum possible level. At the farthest upstream station (WWPS n), if wastewater reaches Ln_OFF_high, no buffer remains and flooding must be avoided. Operation then follows the hysteresis band [Ln_ON_high, Ln_OFF_high], with FCs maintaining Ln_ON_high as the setpoint. All other WWPSs (i > 1) follow the same procedure.

Fault control deactivation initiates when WWPSi_faulty_state clears, Li_ON and Li_OFF return to normal values, respectively FCs reset to initial frequency. Full deactivation requires sequential recovery from the farthest upstream station toward WWPS 1. If influent volume falls below WWTP_inlet_min_flow and L1 < L1_OFF_normal (WWPS 1 empty), the farthest faulty WWPS is deactivated. Subsequently, each WWPSi_faulty_state clears when L_{i+1} < Li+1_OFF_normal, WWPS(i+1) is deactivated, and WWTP_inlet_min_flow is reached.

The Blocked WWPS scenario is further described. Fault detection conditions include: a) WWPS level not reaching Li_OFF with near-zero discharge (for calibrated flowmeters). b) WWPS level (Li > Li_ON) not decreasing over time (widely applicable and automatically identifiable). c) Fault signals such as WWPSi_power_failure or WWPSi_emg.

Higher-level control procedure begins with WWPS1_faulty_state, treating WWPS 1 as blocked. Setpoints L1_ON and L1_OFF are raised to L1_ON_high and L1_OFF_high, stopping pumps.

If L1 > L1_max = L1_ON_high – dL1 (≈ Li_OFF_high), WWPS2_faulty_state is triggered, extending buffering to WWPS 2. This process cascades upstream through all WWPSs. If the fault persists at the last station (WWPS n), pumps operate under faulty setpoints (Li_ON_high, Li_OFF_high). For FC-equipped pumps, Li_ON_high serves as the PI control reference.

Fault control deactivation for blocked WWPS scenarios (e.g., clogged pipes) require maintenance intervention and manual reset via digital tag switch.

Deactivation begins by restoring L1_ON/L1_OFF to normal values. If L1 < L1_OFF_normal, the farthest faulty WWPS (WWPSl) is cleared. Sequential recovery proceeds upstream to WWPS 1, with each WWPSi_faulty_state deactivated once Li+1 < Li+1_OFF_normal and the downstream station is cleared.

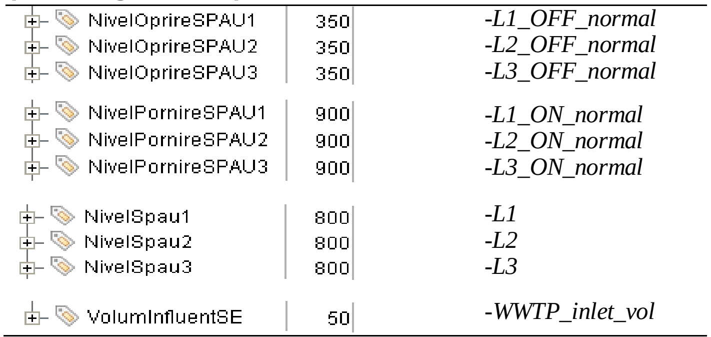
As local automation generalities, each WWPS operates autonomously with: Two-positional hysteresis level control and analogue measurement; At least two pumps, with local fault detection (overcurrent, overheating, leakage); Manual/automatic selectors, emergency button, and electrical fault detection (UPS/generator); Flowmeters measuring discharge flow and volume.

HLCS monitors available data (levels, hysteresis limits, emergency/fault states, flow, pump frequency/state). Fault detection and control are achieved by adjusting high/low level limits and, for FCs, pump frequency. Direct start/stop commands are generally overridden by local controllers, but frequency setting is supported. By monitoring a limited set of tags and adjusting only level limits and pump frequency, HLCS achieves non-invasive fault detection and control for WWPS–WWTP systems.

4.5.2 Results

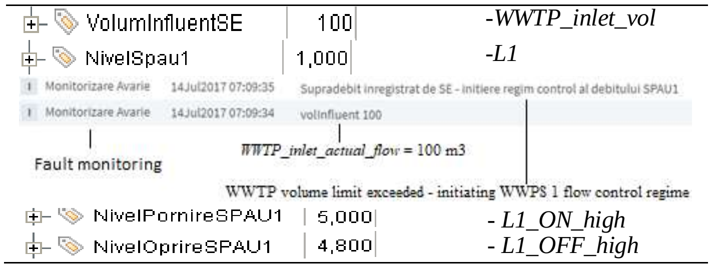
The solution was implemented in the local historian (Node-RED) and in Ignition, with two case studies addressing both simulation and real scenarios. Simulation was required to test conditions difficult to reproduce simultaneously in practice and to explore application limits without stressing critical infrastructure. The simulation scenario examines the WWPS–WWTP network under supplementary wastewater inflows at the WWTP inlet caused by storm water, reflecting real facility characteristics. The WWTP serves a city of ~13,000 inhabitants, with a maximum influent flow (30 min) of 98 m³. When this threshold is exceeded and the equalization basin is full, wastewater is diverted to the bypass channel untreated. The minimum influent volume (30 min) required for treatment is 17.5 m³, thus WWTP_inlet_min_flow was set to 49 m³. The simulated network includes three WWPSs without frequency converters. During testing, tag values were manually adjusted to trigger fault control, progressively extending from WWPS 1 to WWPS 3, followed by verification of gradual fault deactivation.

The experiment begins as shown in Fig. 4.5-3, with levels expressed in mm and WWTP_inlet_vol in m³. All events are logged via the Ignition Gateway to track system status. When WWTP_inlet_vol > WWTP_inlet_max_flow (Fig. 4.5-4), the fault control procedure is triggered, automatically raising the hysteresis limits of WWPS 1. Subsequently, the wastewater level in WWPS 1 is manually increased so that L1 > L1_OFF_high. The algorithm then shifts the upstream station (WWPS 2) into a faulty state, likewise increasing its hysteresis limits (see Fig. 4.5-5).



NivelOprireSPAU1	350	*-L1_OFF_normal*
NivelOprireSPAU2	350	*-L2_OFF_normal*
NivelOprireSPAU3	350	*-L3_OFF_normal*
NivelPornireSPAU1	900	*-L1_ON_normal*
NivelPornireSPAU2	900	*-L2_ON_normal*
NivelPornireSPAU3	900	*-L3_ON_normal*
NivelSpau1	800	*-L1*
NivelSpau2	800	*-L2*
NivelSpau3	800	*-L3*
VolumInfluentSE	50	*-WWTP_inlet_vol*

Fig. 4.5-3 Screenshots presenting the starting point of the experiment.



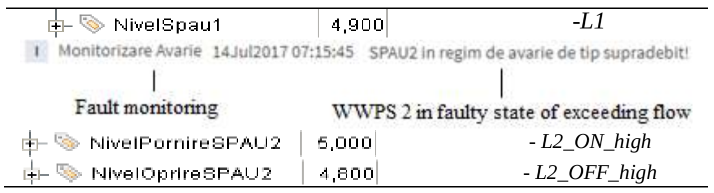
VolumInfluentSE	100	*-WWTP_inlet_vol*
NivelSpau1	1,000	*-L1*

```
Monitorizare Avarie   14Jul2017 07:09:35   Supradebit inregistrat de SE - initiere regim control al debitului SPAU1
Monitorizare Avarie   14Jul2017 07:09:34   volInfluent 100
```

Fault monitoring
WWTP_inlet_actual_flow = 100 m3
WWTP volume limit exceeded - initiating WWPS 1 flow control regime

NivelPornireSPAU1	5,000	*- L1_ON_high*
NivelOprireSPAU1	4,800	*- L1_OFF_high*

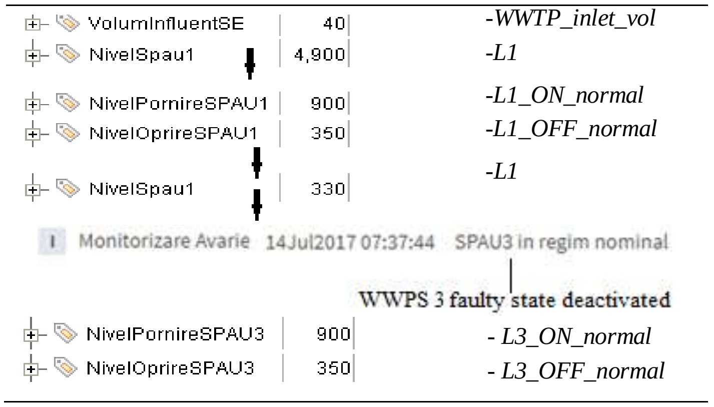
Fig. 4.5-4 Entering fault control procedure and faulty state for WWPS 1.



```
NivelSpau1            4,900      -L1
Monitorizare Avarie  14Jul2017 07:15:45  SPAU2 in regim de avarie de tip supradebit!
Fault monitoring      WWPS 2 in faulty state of exceeding flow
NivelPornireSPAU2     5,000      - L2_ON_high
NivelOprireSPAU2      4,800      - L2_OFF_high
```

Fig. 4.5-5 Extending the fault control procedure to WWPS 2.

The procedure continues until WWPS 3 enters a faulty state, resulting in all WWPSs being blocked. The WWTP inlet volume (30 min) then decreases below WWTP_inlet_min_flow. With L1 > L1_OFF_high, the hysteresis limits of WWPS 1 are reduced to initiate pumping. As shown in Figure 4.5-6, when L1 < L1_OFF_normal and WWTP_inlet_vol ≤ WWTP_inlet_max_flow, the faulty state of WWPS 3 is deactivated, and its hysteresis limits return to normal operating values.



```
VolumInfluentSE         40      -WWTP_inlet_vol
NivelSpau1           4,900      -L1
NivelPornireSPAU1      900      -L1_ON_normal
NivelOprireSPAU1       350      -L1_OFF_normal
                                -L1
NivelSpau1             330
Monitorizare Avarie  14Jul2017 07:37:44   SPAU3 in regim nominal
                  WWPS 3 faulty state deactivated
NivelPornireSPAU3      900      - L3_ON_normal
NivelOprireSPAU3       350      - L3_OFF_normal
```

Fig. 4.5-6 Deactivating the faulty state for WWPS 3.

By lowering wastewater levels and keeping WWTP_inlet_vol below the maximum threshold, all WWPSs progressively exit their faulty states.

Two real WWPS experiments were conducted to demonstrate the capabilities of the non-invasive HLCS and the benefits of interoperation. The tests involved two fault scenarios within the WWPS network (Fig. 4.5-7). Wastewater collected by WWPSs 3 and 6 was pumped to WWPS 7 and then to WWPS 8. The fault conditions were artificially induced, and all actions remained non-invasive with respect to local system operation.

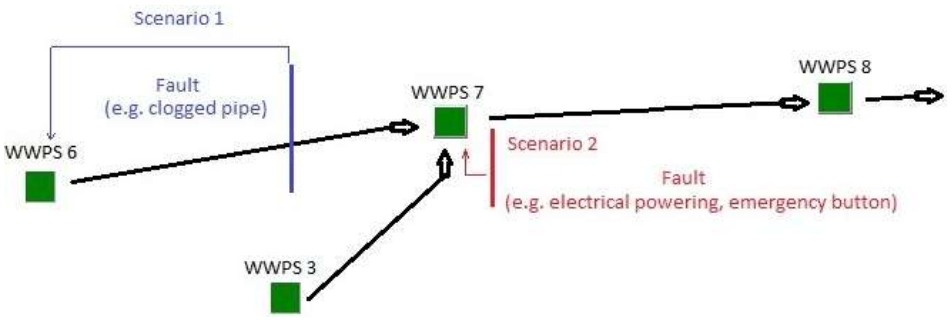


Fig. 4.5-7 WWPS network configuration.

Scenario 1 examines a clogged pipe fault at WWPS 6, detected by a high wastewater level. Prior to the experiment, WWPS 6 operated normally within the two-positional hysteresis band (pump stop/start limits: 350–900 mm). The level evolution over a 4-hour period (09:45–13:50) is shown in Fig. 4.5-8.

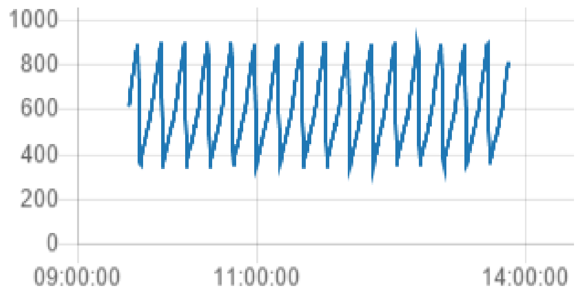


Fig. 4.5-8 WWPS 6 functioning on 17.06.2017 for 4 hours.

The Scenario 1 tests were conducted on 17.06.2017 (Saturday), when water consumption remained relatively constant, with minor storm water inflows from light rain. Under these conditions, about four filling/emptying cycles per hour were observed (see Fig. 4.5-8). WWPS 6, a first-level wastewater collector, operates independently of upstream pumping failures. The fault was induced by lowering the pump start level limit in the HLCS (not locally). At 16:30, the fault was detected, and the pump start/stop limits were raised to 5000 mm and 4800 mm (see Table 4-10). The level limit set to 5000 mm is roughly the half of the station storing capacity.

Table 4-10 Database view between 16:29:53 and 16:30:38

Moment	Level	HighLimit	LowLimit
Sat Jun 17 2017 16:29:53	863	900	350
Sat Jun 17 2017 16:30:08	878	900	350
Sat Jun 17 2017 16:30:23	892	5000	4800
Sat Jun 17 2017 16:30:38	908	5000	4800

WWPS 6 remained in the fault control procedure for 2 hours 15 min., with the level reaching 3573 mm. An initial measurement error occurred, as the level rose abruptly from 950 mm to 3495 mm in 2 min. Thereafter, the increase was gradual, totaling only 70–80 mm. The level evolution during fault control is shown in Fig. 4.5-9. At 18:46, the fault control procedure was deactivated, and the pump start/stop levels were restored to normal operating values. Table 4-11 presents database values recorded at the moment of deactivation.

After wastewater accumulation in the faulty scenario, WWPS 6 was emptied within 2 minutes. The experiment continued until 19:35, confirming that WWPS 6 returned to normal operation. The HLCS effectively detects and manages a clogged pipe failure, providing operators with approximately

48 hours for repair (under the 5000 mm level limit) without introducing additional issues in WWPS performance or unnecessary energy consumption.

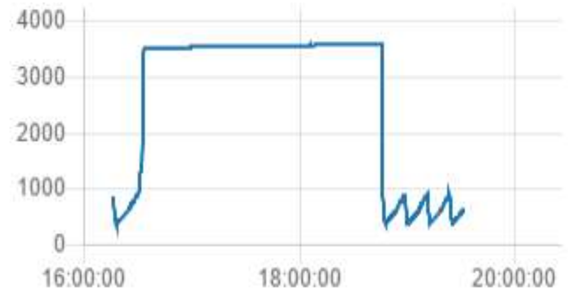


Fig. 4.5-9 WWPS 6 functioning on 17.06.2017 after failure detection.

Table 4-11 Database view between 18:46:23 and 18:48:24

Moment	Level [mm]	HighLimit [mm]	LowLimit [mm]
Sat Jun 17 2017 18:46:23	3573	5000	4800
Sat Jun 17 2017 18:46:39	1190	900	350
…			
Sat Jun 17 2017 18:48:24	351	900	350

Scenario 2 examined a failure at WWPS 7 (e.g. emergency button or electrical fault). The station’s behavior is analyzed globally alongside other WWPSs (Fig. 4.5-7). As a critical network point, WWPS 7 was operating normally, but its emergency button tag was artificially activated in the HLCS to trigger fault control.

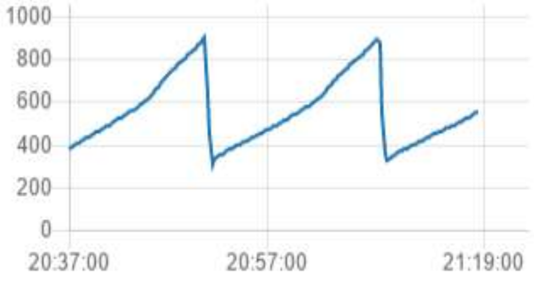
Prior to fault activation, all WWPSs functioned in normal regime for 23 minutes (20:37–21:00), with level evolutions shown in Fig. 4.5-10. WWPS 7 averaged 12 filling/emptying cycles per hour. The tests were conducted on 19.06.2017 (Monday evening), during high water consumption. At 21:00, the failure was detected, and pump start/stop limits for WWPS 7 were raised to 5000 mm and 4800 mm. The level evolution until 21:20 is presented in Fig. 4.5-10.

The failure at WWPS 7 was maintained between 21:00 and 22:00, during which the level rose to 1795 mm. Beyond 900 mm, the measurement error was smaller than in Scenario 1 (WWPS 6), though a brief, unjustified spike in level indication was still observed (Fig. 4.5-10c). Throughout the fault control procedure, WWPSs 3 and 6 remained unaffected (Fig. 4.5-11).

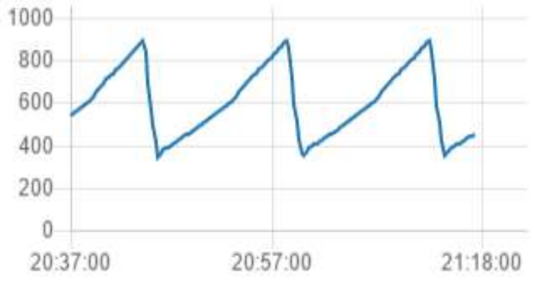
Fig. 4.5-12 illustrates the level evolution of WWPSs 7 and 8 before and after fault control deactivation. As shown in Fig. 4.5-12a, approximately 10 minutes were required for WWPS 7 to resume a normal filling/emptying cycle following wastewater accumulation during fault control.

During the fault control procedure at WWPS 7, with WWPSs 3 and 6 operating normally, the level rise rate stabilized after 2.5 minutes at approximately

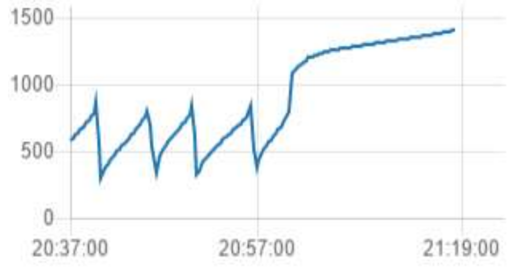
600 mm/hour. Under these conditions, the non-invasive solution provides the operator with about 6.3 hours to resolve the issue (based on a 5000 mm starting set-point) without creating additional problems at WWPS 7 or adversely affecting other stations in the network.



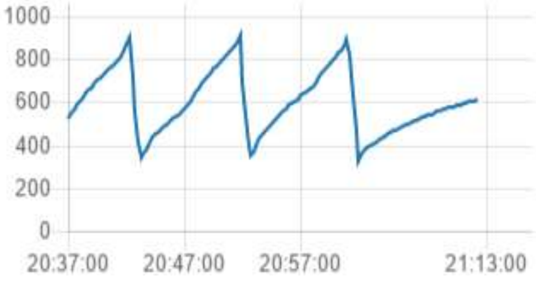
a) WWPS 3 level before and after the failure



b) WWPS 6 level before and after the failure

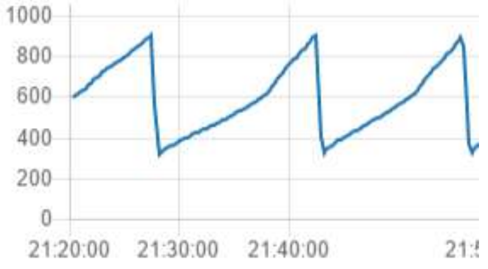


c) WWPS 7 level before and after the failure

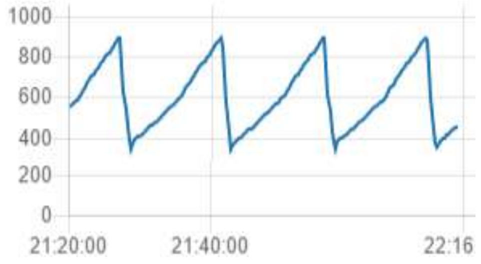


d) WWPS 8 level before and after the failure

Fig. 4.5-10 WWPS 3, 6, 7, 8 before and after the failure.

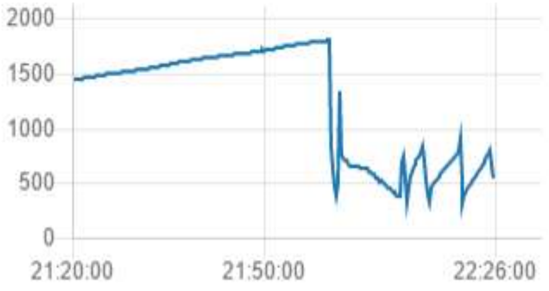


a) WWPS 3 level until failure deactivation

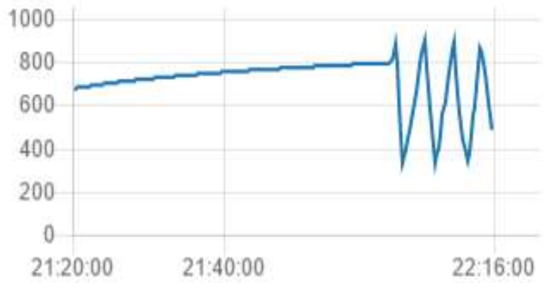


b) WWPS 6 level until failure deactivation

Fig. 4.5-11 WWPS 3, 6 until failure deactivation.



a) WWPS7 level until and after failure deactivation



b) WWPS8 level until and after failure deactivation

Fig. 4.5-12 WWPS 7, 8 until and after failure deactivation.

4.6 Image-Processing-Based Low-Cost Fault Detection Solution for End-of-Line ECUs in Automotive Manufacturing.

The current chapter presents data from [K-20]. In the automotive industry, Electronic Control Unit (ECU) manufacturing concludes with complex end-of-line (EoL) testing before delivery to clients. Within the Industry 4.0 framework, efficiency gains require automatic visual inspection, integrated seamlessly into legacy production lines without interruptions. To align with Industry 5.0 principles, human involvement in decision-making remains essential. This chapter introduces an image-processing low-cost fault detection (IP-LC-FD) solution for EoL ECUs, designed for high-quality, rapid detection. The system targets defects such as incorrect pin mounting, missing or extra pins, damaged clips, and surface cracks. The IP-LC-FD system combines hardware and software: Raspberry Pi microcomputers, Pi cameras, and Python/OpenCV environments. The research progressed through two stages: development of an experimental model followed by a prototype.

4.6.1 The Experimental Model

The experimental model aimed to achieve a ≥95% fault detection rate with processing times under 7 s, ensuring integration into the production line. Cost reduction was prioritized, focusing first on hardware, then on software and maintenance. The major hardware expense was tele-centric cameras, whose cost increases with object size. The first ECU analyzed measured 21 cm × 18 cm, requiring the IP-LC-FD system to be positioned for easy operator handling. The hardware architecture (see Fig. 4.6-1) was built around Raspberry Pi 3 boards and Pi cameras. A board-camera ensemble costs over 100×less than a tele-centric camera, though Pi cameras introduce perspective distortion and Raspberry Pi boards have limited processing power compared to PCs with GPUs. The architecture comprised: 4 Raspberry Pi 3 boards (1 master, 3 slaves); 4 V2 Pi cameras with 2× telephoto lenses, mounted 31 cm above the ECU (±0.5 cm error on vertical axis, I²C communication); Ethernet switch; Barcode scanner.

Fig. 4.6-2a shows the final positioning of the enclosures in the experimental model. The complete structure (Fig. 4.6-2b) incorporates a uniform lighting system installed adjacent to the enclosures. From an image-processing perspective, each ECU module was divided into four zones, each photographed by a camera–Raspberry Pi assembly. Due to the board’s dimensions, this division was necessary to properly manage perspective distortions and shadow effects. A switch integrated into the hardware architecture enabled parallel processing of the four image areas, increasing speed through both parallelization and multithreading in the Raspberry Pi software. For software and maintenance costs, the system emphasized flexibility, adaptability, and modularity: Easy module replacement; Expandability to different ECU types; Capability to learn new configurations for existing ECUs; GUI-based parameter configuration (lighting, area, search position, etc.); Operation without moving parts, reducing maintenance needs. Thus, the low-cost image-processing fault detection solution overcame hardware limitations while delivering performance comparable to, or better than, more expensive systems.

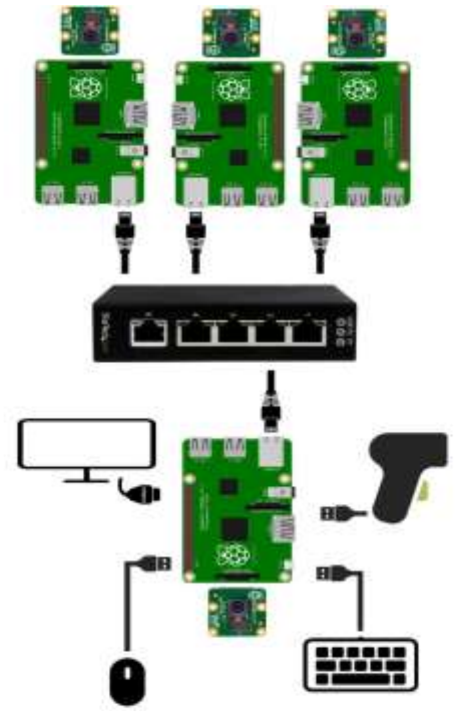


Fig. 4.6-1 The hardware architecture of the IP-LC-FD experimental model.

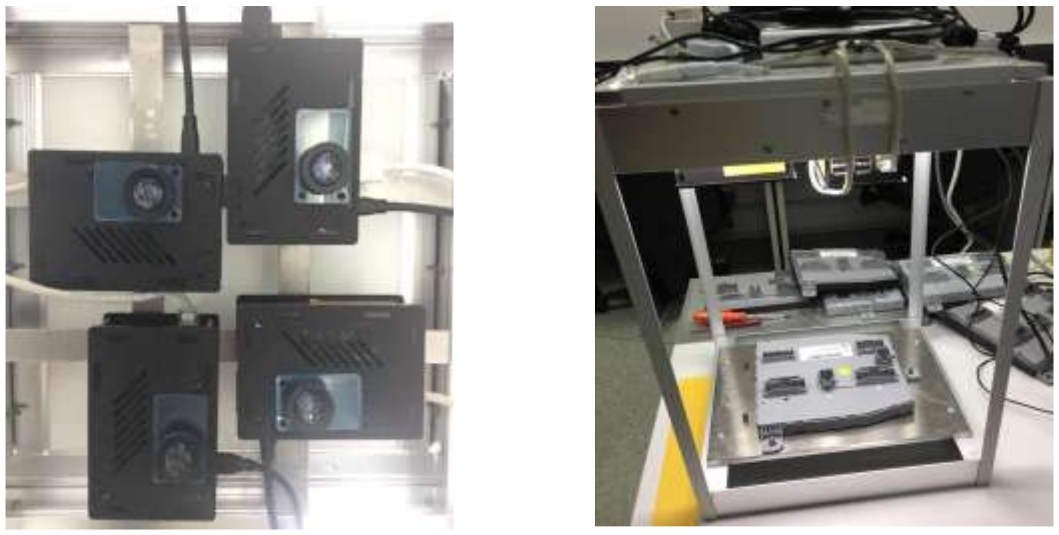


Fig. 4.6-2 The experimental model hardware of the IP-LC-FD. a) The disposal of the four Raspberry Pi - Pi camera ensembles. b) The final experimental model.

The experimental model enables fault detection on ECU boards at the EoL stage. Fig. 4.6-3 illustrates examples of pins, connectors, clips, and cracks. The system tasks include: Detecting crooked, missing, or extra pins; Identifying misplaced or damaged clips; Detecting board cracks; Reporting and logging the inspection process; Collecting/marking faults and aggregating data on the master unit; Supporting user and board selection, debug procedures, and GUI-based interaction; Managing existing configurations and learning new ECU configurations; Barcode reading; Data exchange among the four micro-computers and communication with higher-level traceability.

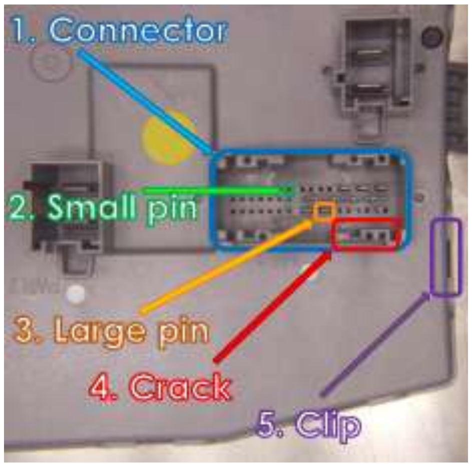


Fig. 4.6-3 Some analyzed components on an ECU.

The master node controls each slave, dictating tasks such as image capture, processing, and download preparation. Each slave serves only the master, with communication handled via Ethernet. Slave nodes implement servers, and data exchange uses Remote Procedure Call (RPC), a request–response protocol for process synchronization. RPC methods, registered on the server, may accept parameters in XML or JSON formats. During execution, the client remains blocked until completion, with an error-catching mechanism for networking failures. RPC functions as a form of inter-process communication (IPC) across distributed systems. Importantly, system modifications (e.g. adding modules or cameras) do not affect the server or RPC requests.

The architecture (Fig. 4.6-4) shows dark green RPC requests at the master, bidirectionally linked to slave servers. The Poller module manages incoming server requests. Color coding indicates: Beige-white blocks: specialized modules unique to each node; Yellow blocks: distributed modules with similar functions; Green blocks: communication modules handling requests, prioritization, interpretation, keep-alive connections, and command execution. The master node is the central element, providing command and control and serving as the sole communicator with other nodes. Its software components are shown in Fig. 4.6-5, and the functional flowchart in Fig. 4.6-6.

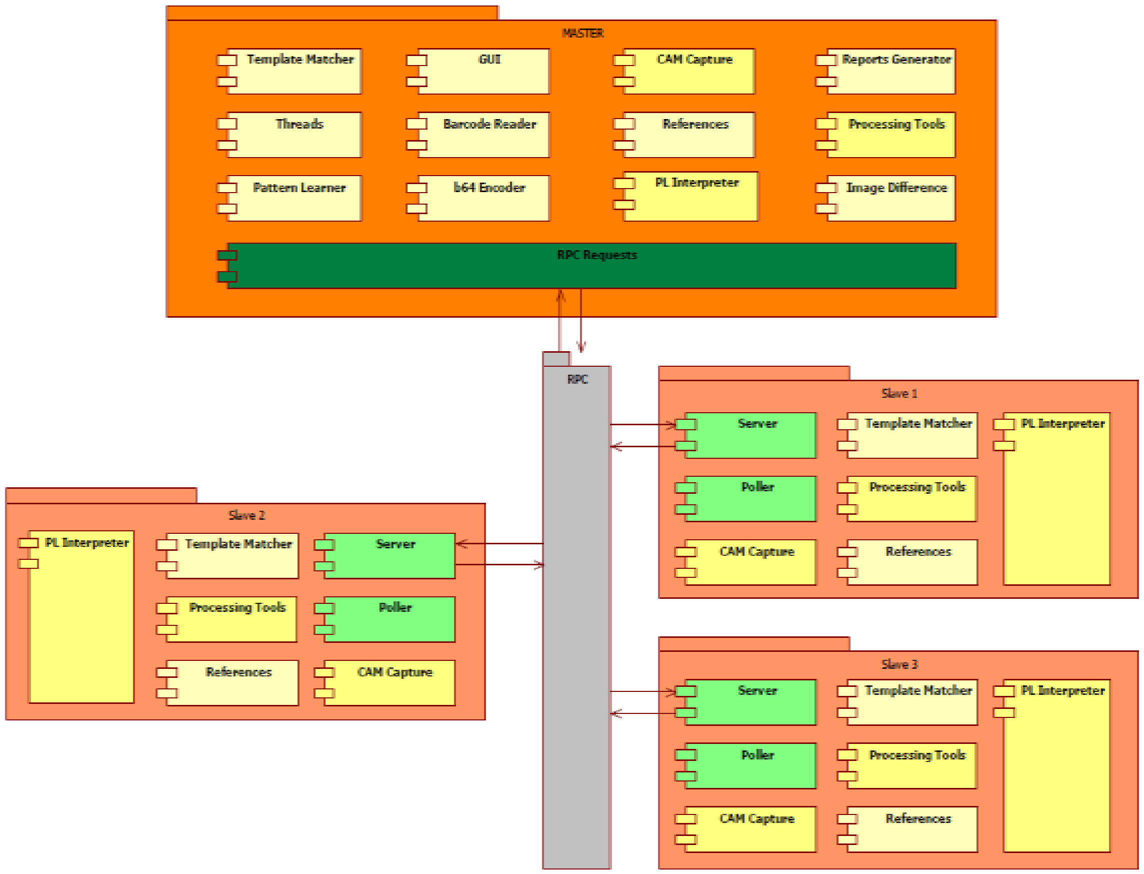


Fig. 4.6-4 The general architecture of the IP-LC-FD experimental model.

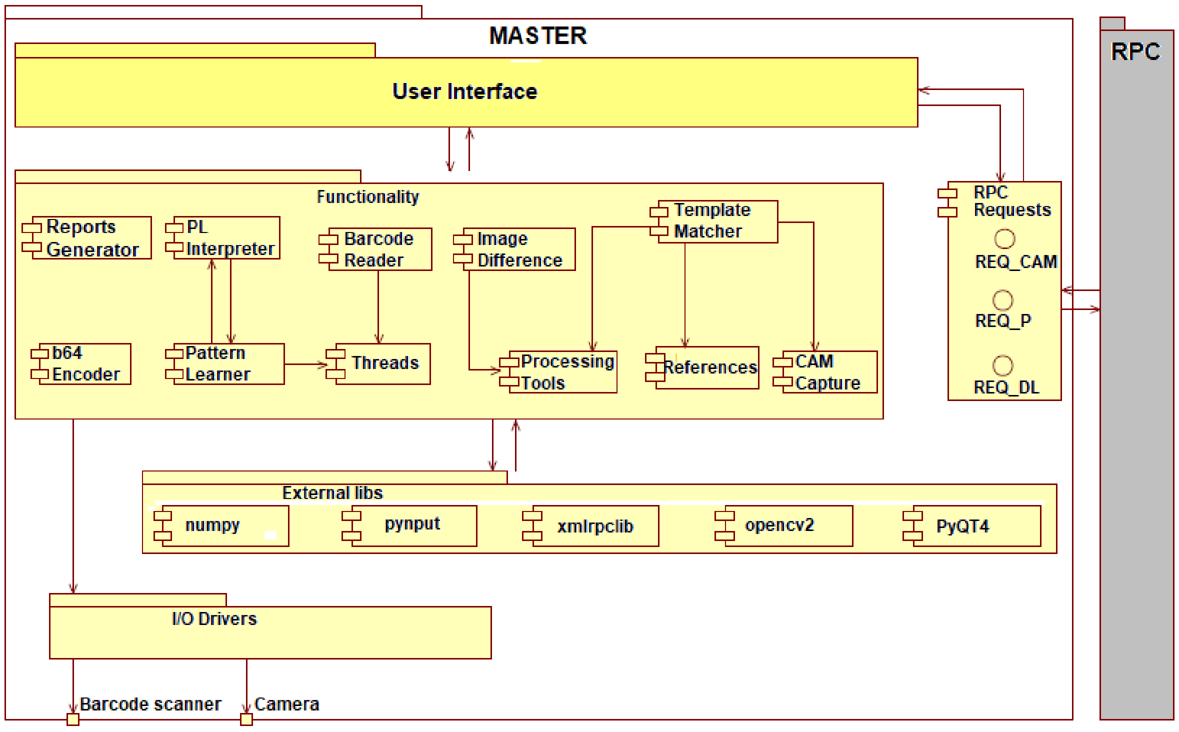


Fig. 4.6-5 Software architecture of the Master node.

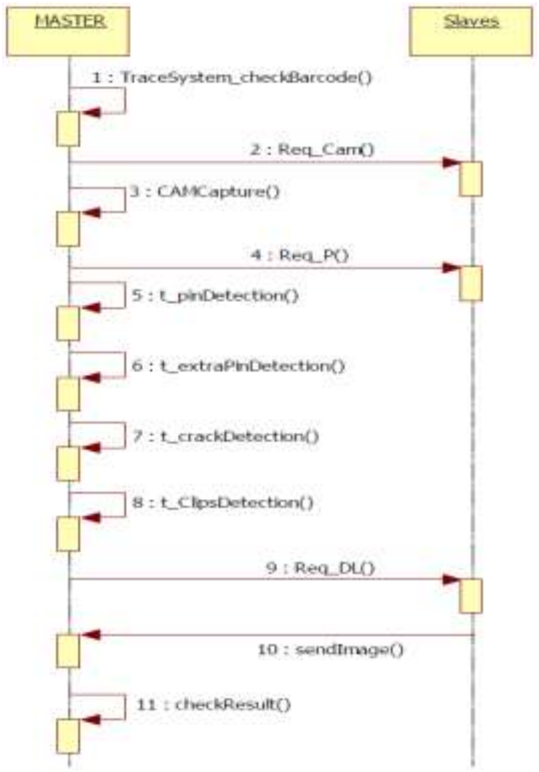


Fig. 4.6-6 The processing flowchart.

In the software architecture, each slave node is organized into components based on functionality and interaction with other modules (see Fig. 4.6-7). The foundation includes I/O drivers controlling peripherals and external libraries (.dll files for xmlrpclib, OpenCV2, and NumPy) that support higher-level components. The two primary low-level modules dependent on these libraries are Functionality and Server. The system is modular, with each of the 15 main modules having a defined scope and simple interconnections. Some modules include configuration files for calibration. Due to system scale and confidentiality, detailed descriptions remain limited.

A key module is the Template Matcher, the functional core of the application. It employs OpenCV primitives and low-level image processing techniques, operating directly on pixel color values to analyze input images.

One of the module’s most complex methods addresses detection of the smallest ECU pins. It uses three input parameters: templ – a template image from the project file structure, used to locate matching zones in the source image; connector – the cropped image of a specific ECU connector; refs – a list of coordinates, previously processed by the Pattern Learning module, which selects positions via the user interface and returns them for optimization. The refs list ensures pins are detected within expected positions, allowing validation against a threshold zone. Invalid pins are added to a fault list, later reported to higher application levels and visually marked with a red square in the interface. The module also includes a method for connector identification and a compare function, which extends the correct set of pin coordinates provided by the Pattern Learner. The compare method checks: Missing pins – when no detection occurs at expected coordinates; Extra pins – when detection occurs outside the defined set. Extra pin detection follows an inverse logic to basic detection, introducing significant complexity.

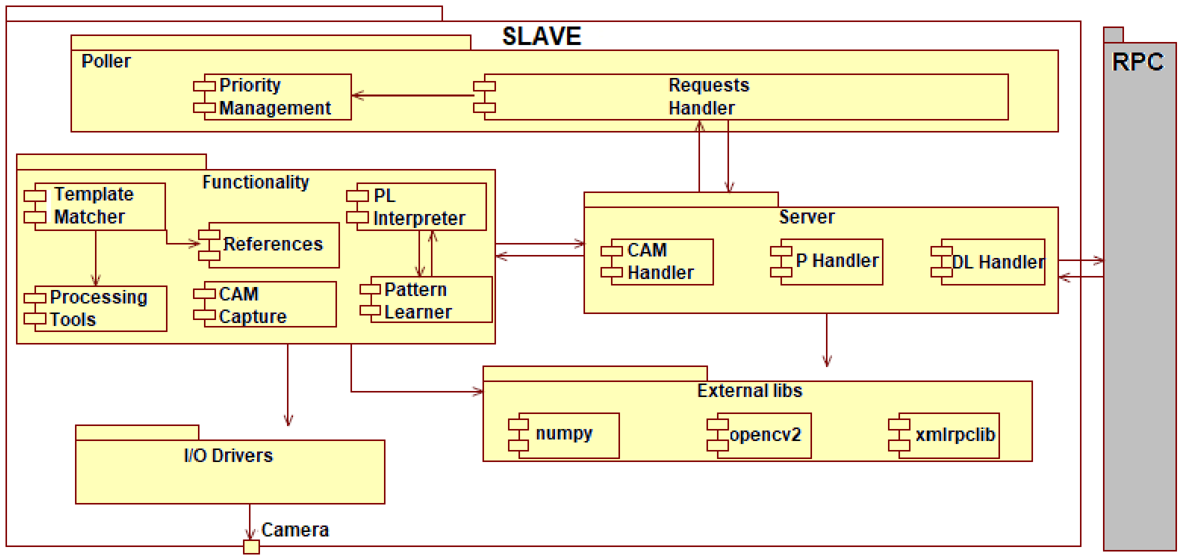


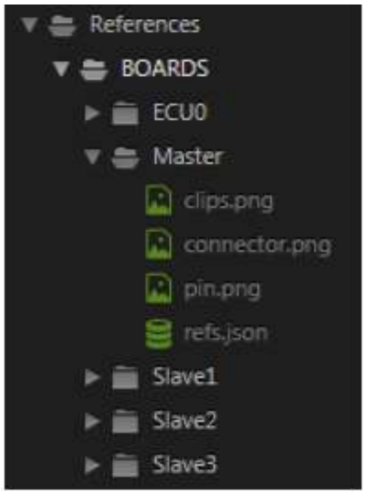
Fig. 4.6-7 The processing flowchart.

The Threads module manages all application threads, continuously monitoring resources during runtime to avoid delays. One key resource is memory availability for report storage. The system periodically diagnoses the report location, calculates byte usage, and displays the result as a status bar (0–100%) on the user interface.

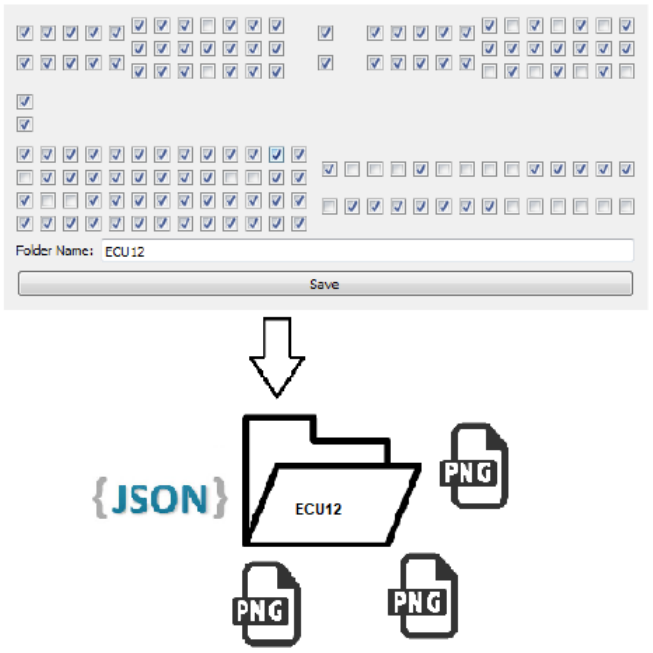
The Pattern Learner module enables analysis of multiple ECU board types sharing the same mechanical structure. While the boards tested in the experimental phase had identical structures, their pin configurations defined distinct functional characteristics and circuits. In industry, reusing mechanical structures across ECU variants is common to reduce costs. Accordingly, the Pattern Learner exposes all connector pin configurations in a user interface window, allowing operators to add or remove pins as needed.

The Base64 module encodes images into character arrays to simplify network transfer. Direct binary transmission may cause compatibility issues across operating systems or misinterpretation by certain protocols; encoding ensures all data is represented as ASCII text. The References module generates reference files for specific ECU types, supporting image processing by storing data on board elements and baseline captures for defect identification. Each reference set (Fig. 4.6-8a) is saved in a folder containing a template (.png) and a pins layout (.json) (Fig. 4.6-8b).

The PL Interpreter module translates Boolean values from the GUI into pin coordinates. Marked pins are validated (correct position, intact, not bent), while unmarked pins are checked to confirm no extras exist. Each connector's pins are processed sequentially, producing a dictionary structure with connector names as keys and Boolean lists as values. This dictionary is stored in a JSON file as a reference resource (Fig. 4.6-9).



a)



b)

Fig. 4.6-8 References - a) file structure; b) saving procedure

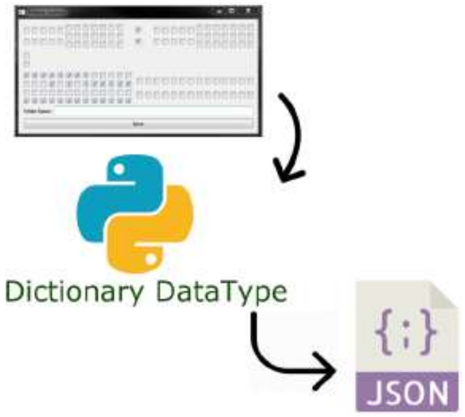


Fig. 4.6-9 Data saving model.

From high-level objects, values are extracted, grouped into a dictionary, and transmitted by the master node to the slaves. Once loaded into RAM, two functions apply filters to obtain pin coordinates (Fig. 4.6-10). These coordinates, stored as lists of tuples, serve as inputs for detection methods in the TemplateMatcher module. The Processing Tools module supports other detection modules with auxiliary methods. Though indirect, its impact on overall performance is significant. For example, rotation procedures are continuously applied, using straight angles to align camera positions or small angles (≤10°) to compensate for mechanical placement tolerances. Improper rotation can compromise cropping and lead to data loss. Cropping procedures are essential both for accurate detection and for reducing execution time.

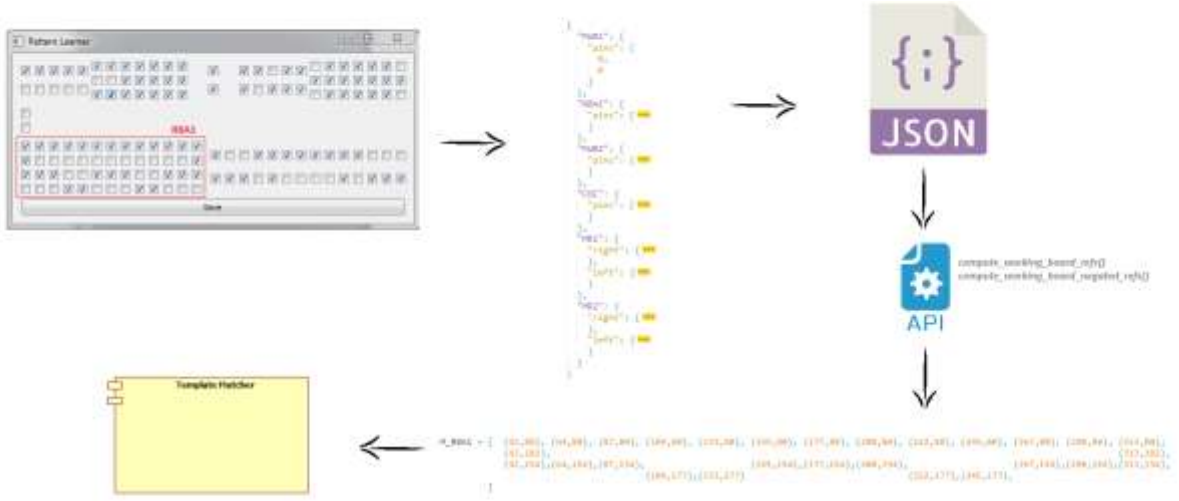


Fig. 4.6-10 Coordinates conversion process.

4.6.2 The Prototype

A new mechanical-hardware structure was required to meet production line demands and ensure performance for basic ECU types. A prototype was designed, implemented, integrated, tested, and validated in the production line. Building on the experimental model, the prototype employed six Raspberry Pis and six physically separated cameras, improving inspection quality for connectors affected by perspective issues. The separation also enhanced surface luminosity during analysis. The first prototype used a circular mechanical surface with cameras magnetically attached (Fig. 4.6-11). However, during initial experiments, operators frequently displaced cameras, disrupting detection and necessitating recalibration. To address this, the final design adopted a rectangular mechanical structure (Fig. 4.6-12), allowing easily adjustable yet stable camera positioning.

The system's data transmission capability was expanded to support 1 master and up to 5 slaves, covering request/response exchanges, data aggregation, reporting, and concluding procedures. A generic platform was designed for n processing modules. In practice, the prototype includes two branches, one master and one slave, with generic slave software applicable to any Raspberry Pi in the scheme. This modularity simplifies maintenance and replacement.

The prototype was extended to operate across the main ECU classes in the production line. This required a new software module concept to accommodate differences in hardware-mechanical structures and ensure adaptability. Modules were developed to incorporate layouts from all ECU classes and their specific board sets. The pin search module was further optimized through cropping techniques, reducing both the search area and processing time.

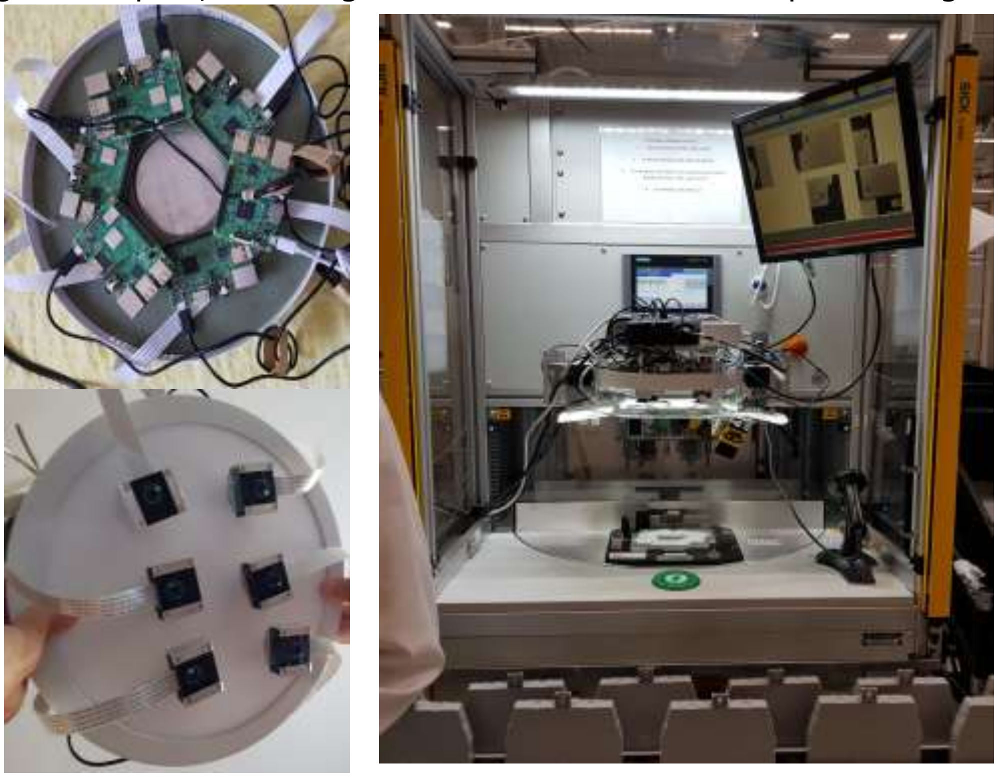


Fig. 4.6-11 First phase of the IP-LC-FD prototype.

New detection modules were developed based on “islands” identification, with islands separated or grouped to improve accuracy. A dynamic illumination threshold was introduced for each pin, addressing light and shadow variations in open environments. Since pins and pinholes are small, variable thresholds were essential for reliable detection.

Layout management (saving, storing, loading) was optimized to handle the large number of layouts in production. Detection task optimization was achieved by having the master equipment extract and distribute connector lists to slaves, eliminating fixed assignments and allowing easy module replacement. Processing time was reduced by removing the need for connector rotation in slave software. Finally, a new layout learning module was implemented exclusively on the master equipment, avoiding the need for ssh/vnc connections to individual slaves.

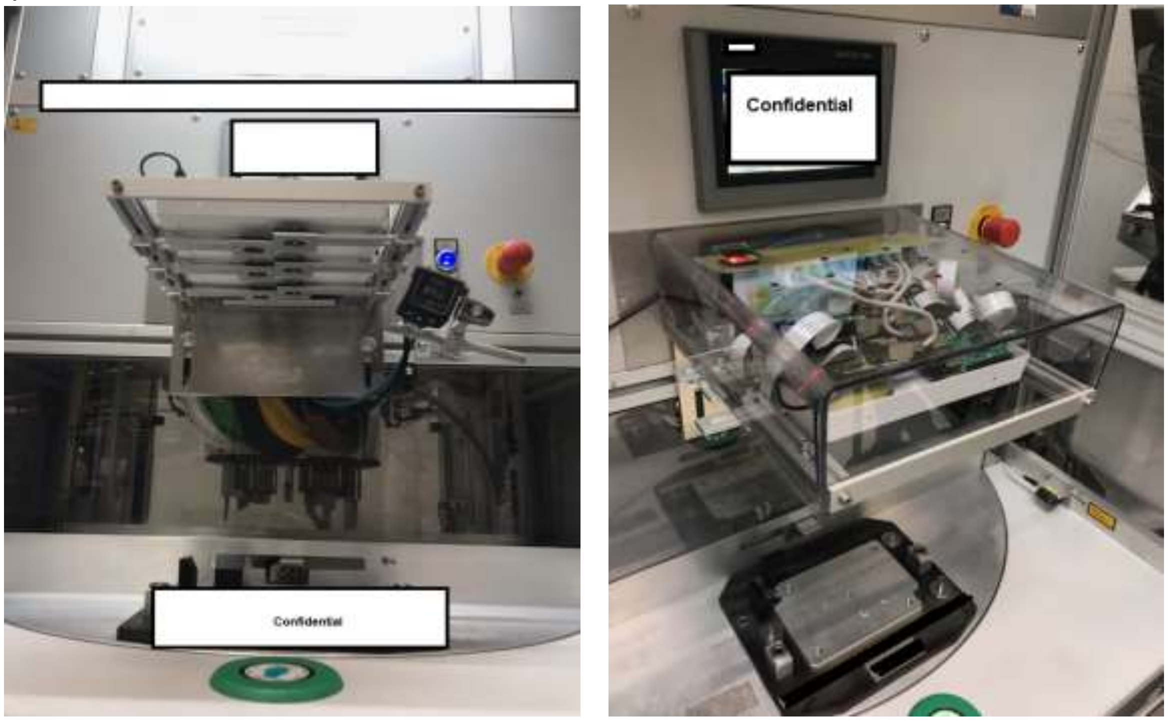


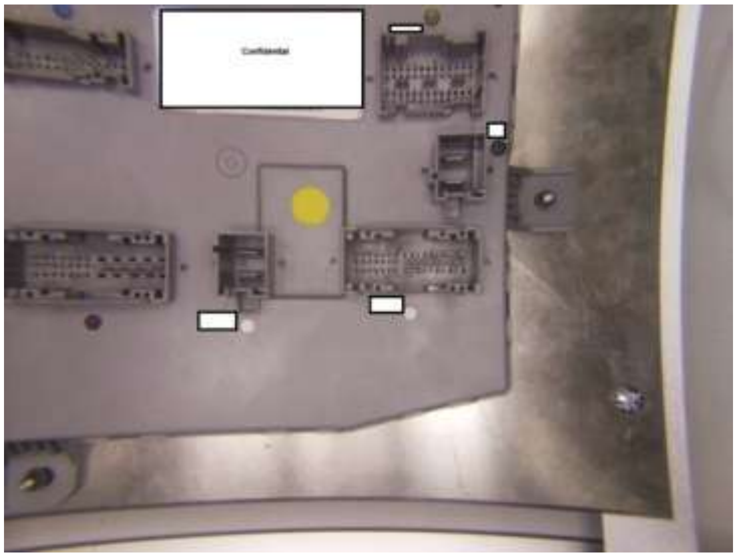
Fig. 4.6-12 Final IP-LC-FD prototype.

A new offset separation was implemented, assigning each pin its own search area. Additionally, a dedicated module calculates the filling factor for each pin's offset. The prototype supports full configuration changes via the GUI and operates in complete integration with the company's traceability software.

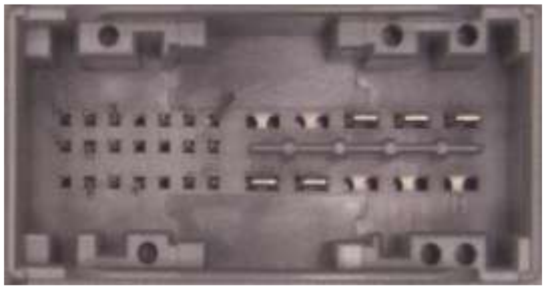
4.6.3 Results

Both the experimental model and the prototype were rigorously tested and validated according to their technological readiness level. For the experimental model, testing was conducted in the laboratory using the IP-LC-FD stand with ~40 ECU boards from a single class. Different pin layouts allowed evaluation of multiple configurations. To simulate diverse scenarios, boards were deliberately modified (bent/broken pins, induced cracks, damaged clips). The stand itself was optimized to meet production line requirements, concluding that a 30 cm × 30 cm diffuse light surface should be mounted above, with Raspberry Pi and camera enclosures positioned ~5 cm below, and ECUs placed 30.5 cm beneath the enclosures.

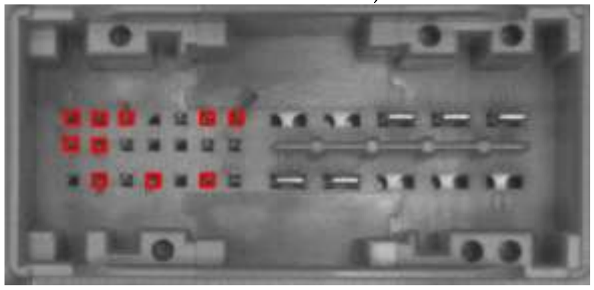
The prototype was tested over 3 months in production, using more than 1000 boards across four ECU classes with varied pin configurations. In the first scenario, the fault detection mechanism for bent pins was analyzed. The algorithm consisted of capture the source image (Fig. 4.6-13a), crop the connector region (Fig. 4.6-13b), detect pin holes and pin positions, mark bent pins with red squares, correctly placed pins remained unmarked (Fig. 4.6-13c).



a)



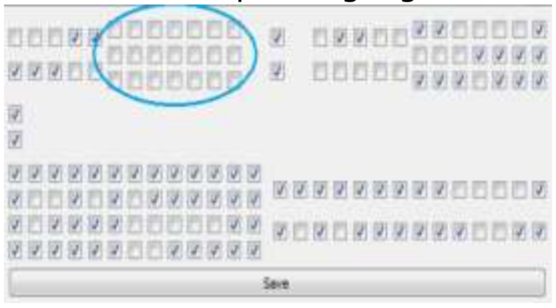
b)



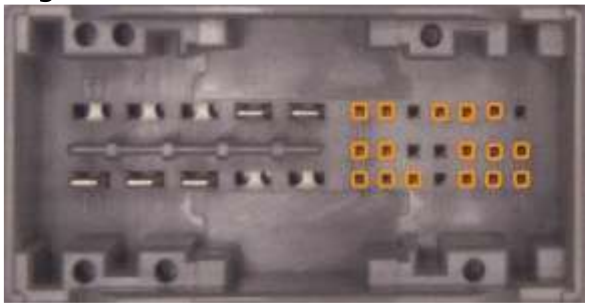
c)

Fig. 4.6-13 Results obtained with the IP-LC-FD system: a) the source image; b) the extracted connector; c) the faulted pins detection.

The second scenario illustrates extra pin detection. Within the Pattern Learner Interpreter, all pins linked to a connector are unchecked (see Fig. 4.6-14a). In this configuration, the algorithm identifies pins located in unmarked positions, while leaving empty holes unmarked. The results (see Fig. 4.6-14b) show all detected extra pins highlighted in orange.



a)



b)

Fig. 4.6-14 Results obtained with the IP-LC-FD system: a) the unmarked pins in the Pattern Learner module for a connector; b) the detected extra pins in the specified connector.

Table 4-12 summarizes additional testing results, while Fig. 4.6-15 illustrates the user interface during a test in which an ECU board showed no faults and was declared Passed.

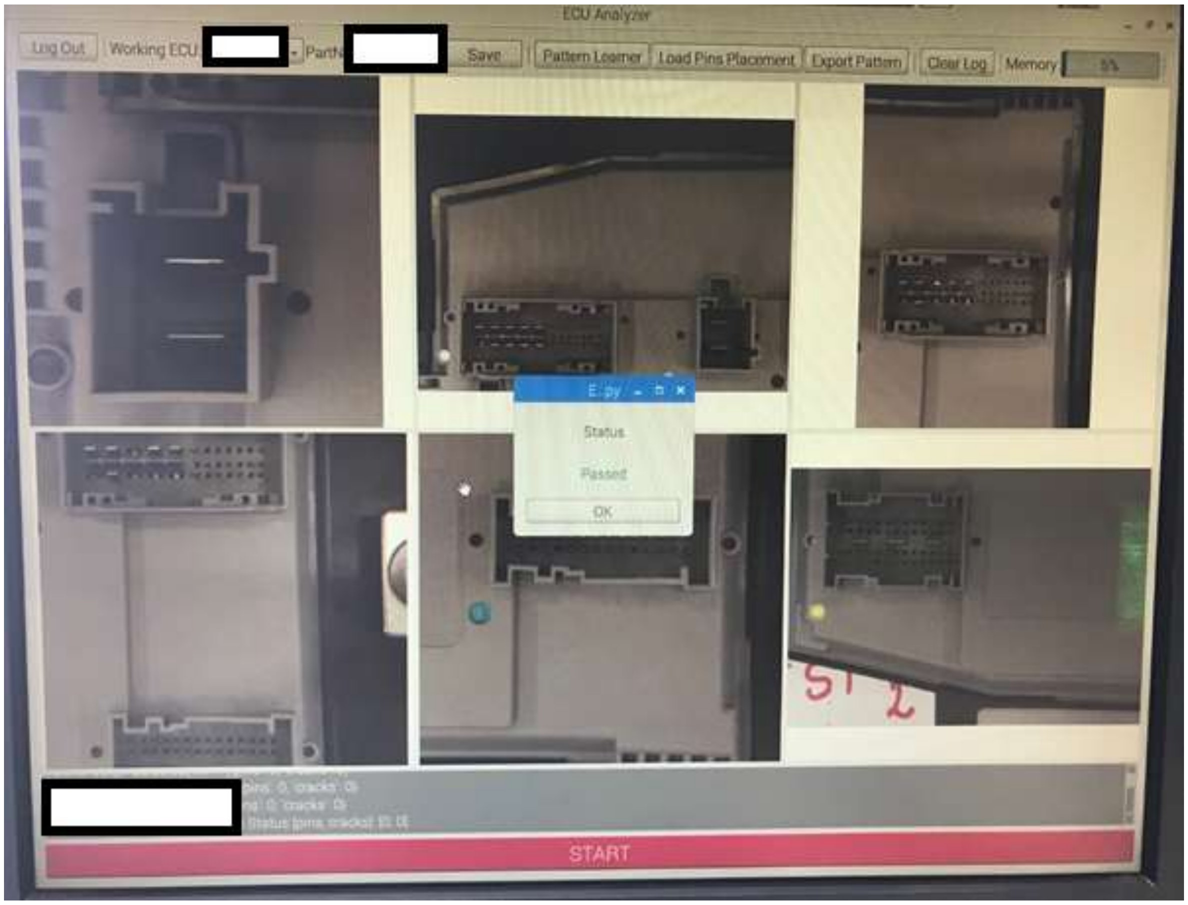


Fig. 4.6-15 An overview of the user interface and a test result for no detected faults.

Table 4-12 Test procedure implementation results

Test procedure	Success rate
Testing the informational flow at the algorithm level ("fairness" concept). a) All module/function activation signals producing the expected outcome (e.g. image capturing on the master/slaves generates the image file, the EdgeDetection function always determines the corresponding execution, the ImageDifference executes always the correct code, etc.); b) correct transition between the states, no unknown state; c) the application is providing outputs and allows the transition to e new cycle both in normal and debug functioning regimes.	100%
Verifying the communications and the threads. a) The communication between the master and the slaves; b) The communication with the traceability application; c) Correct functioning of the local threads.	100%
Testing the local reporting module.	100%
Testing the ability to learn new types of boards and the management of saved ECUs.	100%

Verifying the data aggregation and integration from all processing equipment, and the concluding manner.	100%
Testing the referencing procedure (references addition, adjustments, etc.)	100%
Testing the correct connector detection and fault detection for small and large pins, clips, cracks.	98%
Testing the missing pins detection.	98%
Testing the extra pins detection.	98%
Capability Pass repeatability test – An good ECU is tested successfully 50 times in the industrial environment and 50 times the system provides the same result, Passed.	100%

5 Efficient and Human Centered Industry 5.0 Data Propagation and Representation in the context of Technology Oriented Digital Transformation Interfacing

The current chapter consists of information from three scientific works [K-1], [K-3], [K-4]. The goal was to increase efficiency of data propagation and representation in the context of digital transformation and Industry 5.0 requirements. The solution had to target a high TRL, and to be technology driven and applicable in the current industrial context. Another objective was to bring the academic perspective closer to the industrial technology, to reduce the gap between the entities. Currently the academia is not really involved in the technology validation and fast spreading, this issue causing lots of unverified opinions and guidance to arise.

Efficiency increase is the first objective that requires approaching digital transformation and applying IIoT and Industry 4.0 concepts. Digital transformation fundamentally relies on interfacing and connectivity. The challenges posed by legacy protocols and solutions at the OT level are now extending to the IT domain, where outdated legacy systems introduce an entirely different set of complexities. Industry 5.0 extends previous advancements by integrating a societal vision centered on human-centric values, sustainability, and resilience, while maintaining the efficiency goals of Industry 4.0. At the OT level, data often lacks structure and context, which are typically provided by middleware bridging IT systems and higher SCADA layers. As hierarchy increases, technological expertise declines, leading to suboptimal process representations. Moreover, the existence of multiple SCADA solutions for the same industrial process generates inconsistent interpretations, higher costs, longer development timelines, and greater maintenance complexity.

Section 5.1 is based on [K-3], and it is focusing on establishing a foundation for a single-source-of-truth (SSoT) human-oriented data representation in a context of a virtual unified space for digital transformation. The chosen transport protocol is Message Queue Telemetry Transport (MQTT) and data is transmitted in a structured and contextualized form using JSON and Sparkplug B. The physical objects from industrial processes contain graphic descriptors and defined templates that increase human perception, creative initiative and guiding capabilities, making him able to be in the center of decision making. The other two pillars of Industry 5.0, sustainability and resilience are also positively influenced by the improved representation and perception on all levels, including possibilities of decentralized decision making and AI guiding. The applicability of the solution can be both on the IT and OT levels, including the cloud, and in process control independent applications. The section approaches the Ignition towards Node-RED data propagation. The Ignition environment is set to function in the MQTT and Sparkplug configuration, acting as a SCADA node. An external software application that can be placed on any hierarchical level is set to interface using MQTT, integrating a complete user-defined data type template.

Section 5.2 is based on [K-4] and it approaches the research regarding the OPC UA protocol that can proliferate structured data including graphical representation from the OT level towards other higher supervision levels. The approach considers both legacy systems and technological progress, assuring efficient and human centered structured, contextualized, and graphically sustained data propagation on the OT level. The builds upon industry adopted environments and devices placing Node-RED on the lower level, assumedly PLC level, respectively Ignition on the higher level as SCADA environment.

Section 5.3 is based on [K-1] and it is closing the three-step research focusing on technology-driven data propagation and representation in the Industry 5.0 context. The study provides a bidirectional and flexible propagation of structured and graphically represented data using Sparkplug B protocol, considering Node-RED and Ignition environments. The solution is applied and evaluated qualitatively and quantitatively, proving its efficiency.

5.1 Targeting Broker Based Solution in the Context of Technology Driven Digital Transformation, from Ignition Sparkplug B to Node-RED.

Industrial efficiency has advanced through systems adopting a common OT-level language. Enabled by IIoT and Industry 4.0, many industries achieved open, interoperable, flexible, scalable, and high-performance OT process control, monitoring, and supervision. However, the persistence of legacy systems with diverse protocols limits data quality and consistency, while multiple SCADA solutions across hierarchical levels hinder unified and reliable operation and maintenance.

At the IT level, efforts focused on digital transformation, primarily through ERP implementations (financial modules, asset management). Yet interoperability remains constrained: systems rely on REST architectures, local relational databases without timestamps, and even isolated Excel files. The most impactful improvement came from cloud technologies, which facilitated adoption of lower-level OSI protocols such as MQTT and AMQP, positively influencing IT integration.

The envisioned digital transformation that currently represents one of the most important topics associated with industrial revolution today [166-167] assumes first of all interoperability and OT-IT data integration [168]. For digital transformation, data must be structured, modeled, normalized, contextualized, and trusted. Key debates persist around protocol foundations, architectures, conversion/wrapping modules, truth sources, security, and implementation techniques. Current directions converge on publish–subscribe protocols and event-driven architectures. The state of research, reflected in off-the-shelf solutions, will guide urgent trends. However, industrial legacy systems and the impact of existing products must be considered when defining architectures. Academic research, validation should be significantly expanded to position technologies impartially by value, impact, and future necessity.

The primary objective of Industry 4.0 is to enhance efficiency, as highlighted in reviews of automation and supervision trends [169]. Efficiency improvement was regarded from various perspectives, as energy consumption reduction [170], as increasing production volumes and speed [171], as faster and better fault prediction [K-7], etc. Certain directions, such as enhancing functional safety, reducing the environmental footprint, or improving the quality of human work, are not direct objectives of Industry 4.0. These aspects may appear as secondary accomplishments but are not foreseen outputs of Industry 4.0. According to [172], European industry should reinterpret Industry 4.0 to include a societal perspective and transition toward Industry 5.0. Industry 5.0 builds on Industry 4.0 through three pillars [173]. [174] details a human-centric resilient transition, defining three functioning modes: autonomous, parallel, and expert/emergency. Study [175] anticipates a greater share of virtual workers than humans and field robots, while arguing that humans must remain at the top of decision-making.

This chapter aims to address key objectives in light of the current industrial context, research status, and rapidly evolving scenarios:

Goal 1: Establish a unitary view of process objects across all hierarchical levels, enhancing human vision and perception within the company and positioning them as central actors in decision-making.

Goal 2: Consolidate a Single Source of Truth (SSoT) through contextualized and distributed data, thereby increasing resiliency via improved access to structured and understandable information for operations, and supporting sustainability through human-guided decentralized AI algorithms.

Goal 3: Promote academic analysis of industrial developments, ensuring the establishment, evolution, and consolidation of chronologically validated technologies and solutions.

5.1.1 Industrial and Scientific Context

OT systems continued to evolve, with increasingly decoupled solutions emerging. OPC UA, as the most representative Industry 4.0 protocol, introduced new specifications for Publish–Subscribe requirements, with scientific contributions such as paper [K-19], which proposed a UDP broker-based solution. However, companies have not yet mass-produced equipment implementing the latest OPC UA specifications, with many devices still relying on the Client–Server paradigm. This slow adoption is also noted in [176]. A significant perspective is set by Siemens, which adopted MQTT as a transport protocol for its OPC UA devices (e.g. S7-1500) to enable Publish–Subscribe functionality. Furthermore, work [177] presents an OPC UA Publish–Subscribe solution using MQTT for sensor networks, achieving high-frequency, low-latency message transmission, developed with the open62541 stack.

Paper [178] addresses low-cost, long-range wireless sensor interfacing, showing that Sparkplug B with MQTT offers advantages over the legacy Modbus protocol. Work [179] proposes an architecture based on Sparkplug B and MQTT for a unified namespace, with data pushed directly from OT without applying the Purdue hierarchical model. Sparkplug B is only gradually entering the scientific literature, promoted mainly through successful off-the-shelf products from Inductive Automation and Cirrus Link. Sparkplug B provides several benefits, easy manipulation, plug-and-play, auto-discovery, and low overhead, and is strongly supported by influential products such as Ignition, which offers versions for SCADA, Edge, and Cloud, and brokers like HiveMQ, enabling unified namespace setups in the context of MQTT and Sparkplug B. The protocol's adoption is highly dependent on the success of the Ignition environment, and digital transformation based on Sparkplug B is difficult to envision without it. Ignition is among the most flexible and open SCADA environments, though additional solutions are needed to fully exploit its potential. Compared with IGSS SCADA from Schneider Electric, Ignition does not natively structure data into objects and atoms, and templates are less easily conceived. However, with further software development, user-defined data types (UDTs) can diversify and extend perspectives.

A major advantage of a human-centric Industry 5.0 system would be to provide unitary graphical representations across all user levels. Moreover, Sparkplug B could be further exploited to achieve capabilities equivalent to OPC UA Alarms & Conditions (A&C), as implemented in Siemens products. The overarching goals are to research and develop bridging solutions between industry-driven environments and protocols, highlighting the strengths of widespread technologies, and to provide additional capabilities that enhance independence, flexibility, scalability, and operational efficiency.

Digital transformation introduces the Smart SCADA concept (e.g., Xylem Vue), positioned above legacy SCADA systems to manage OT-level supervision while enabling OT–IT convergence. Other companies employ the Node-RED environment to integrate lower-level SCADA and provide higher-level supervision and intelligence (e.g., [K-8] in automotive manufacturing building management). At the IT level, which plays a critical role in digital transformation [180], interfacing differs substantially from OT. Common practices include REST and SOAP APIs, though many IT solutions remain isolated silos, with direct database access and Excel-based imports still prevalent. Some technologies now provide MQTT links. Meanwhile, ERP systems, increasingly complex due to digitalization requirements [181], face challenges in integration with OT, often requiring data lakes, warehouses, and business intelligence dashboards [182].

The OT–IT convergence is regarded as the most essential step in digital transformation. An SSoT ensuring common knowledge and structured context across all data remains insufficiently addressed in academic literature, as noted in [183] regarding the unified namespace. Practice-oriented researchers, however, present the unified namespace as a key enabler. In [183], authors propose integrating the unified namespace with ISA-95 architecture via asset-administration shells for data interpretation and wrapping. Work [176] provides an overview of a unified namespace-based event-driven architecture for smart shop floors, discussing protocols and open-source tools. The unified namespace using MQTT is further explored in [184], applied to printed-circuit-board surface-mount technology systems within the Industry 4.0 context.

5.1.2 General Architecture and Solution Development

The proposed solution employs a two-step approach: first, SCADA-level data processing prior to publication; second, subscriber/client-level processing to ensure integration and visualization. It defines a publish−subscribe event-driven architecture (EDA) with decoupled entities, aligned with the unified virtual space/unified namespace concept, and extends the SSoT principle through graphical representation of process objects and associated data values. In industrial practice, particularly within the Purdue model, the upper OT level is represented by SCADA control centers. Modern SCADA systems increasingly employ user-defined data types (UDTs), with some environments integrating UDTs into graphical templates. The Ignition environment enables the creation of structured instances combining UDT-based data with graphical descriptors, supporting advanced contextualization and visualization.

Data from structured instances are published via the Sparkplug B and MQTT protocol ensemble, transmitted through an MQTT broker within a unified virtual space. In this architecture, subscribers can access the published data according to the framework presented in Fig. 5.1-1. Within the Node-RED, subscribers can access data via an EDA without polling, providing human operators with enhanced graphical representations of technological process values. The resulting template instances are perceived in a unitary manner across the enterprise, ensuring consistency in visualization and interpretation.

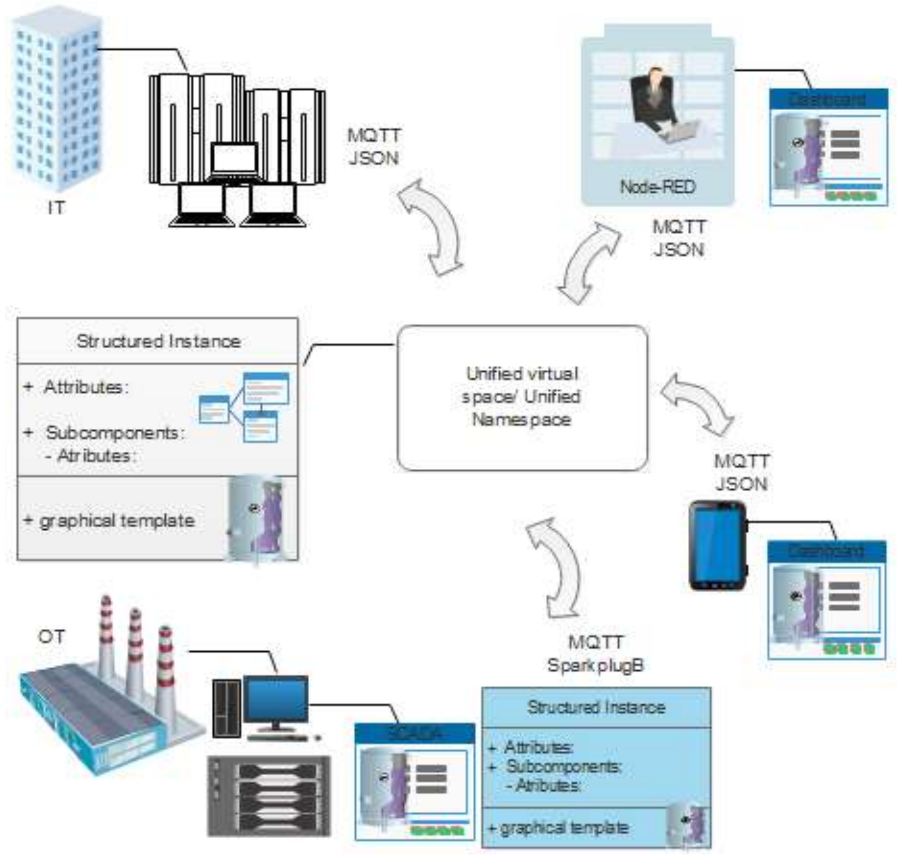


Fig. 5.1-1 Solution architecture considering MQTT Sparkplug B as UDT generator

The two-step methodology outlined above is designed to implement the workflow of data management as described within Fig. 5.1-2. The workflow integrates Ignition for structured data modeling and processing, preparing information for publication via an MQTT broker, and Node-RED for real-time processing and visualization. This approach ensures efficient data acquisition, transformation, and dissemination within an EDA based on Sparkplug B with decoupled entities. In the initial phase, Ignition builds the data structure by defining and instantiating a UDT, subsequently linked to a graphical template. The UDT encapsulates key process attributes along with an additional variable that stores the template, combining structural elements (layout, dimensions, hierarchy) and behavioral elements (tag bindings, real-time values, interactivity). This variable enables external processing and visualization.

The next phase involves data extraction and serialization into JSON, chosen for its flexibility and ease of transmission via MQTT. Extraction is performed through event-based scripts, ensuring updates propagate only when changes occur. Scripts systematically traverse the template hierarchy, retrieving all relevant properties while preserving relationships among nested elements. A recursive approach guarantees proper representation of group structures in the final dataset. Additional steps include recalculating bounding boxes for ShapeGroup elements, since aggregate dimensions are not inherently stored. By analyzing subcomponent coordinates, the recalculation preserves the visual hierarchy. Specialized mechanisms address component-specific properties, such as multi-state behaviors in toggles or buttons, and color gradient interpolation for shape-based graphics.

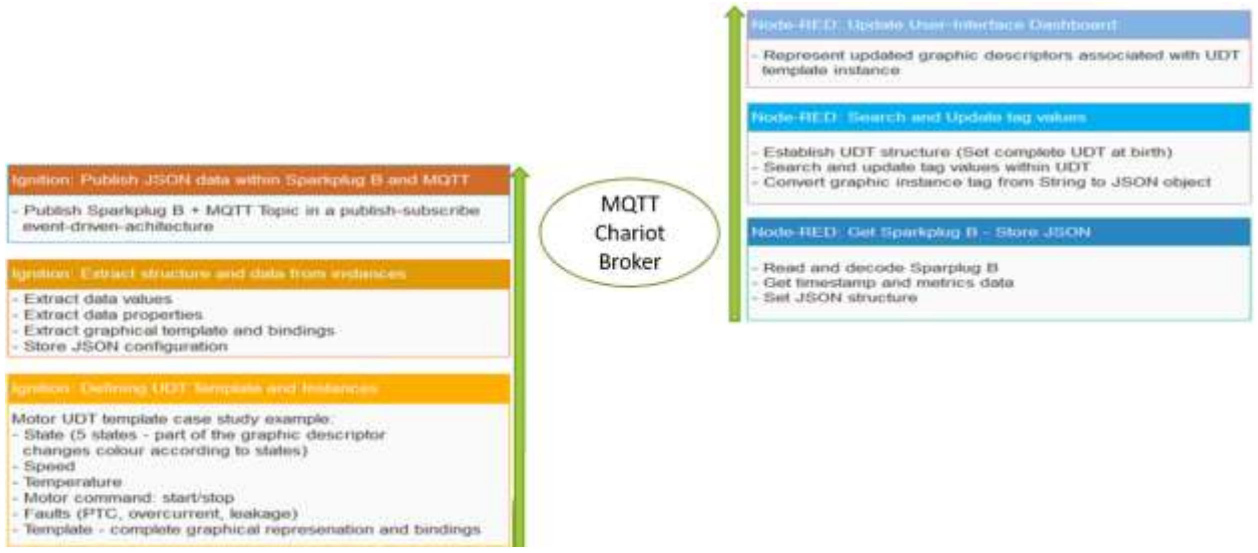
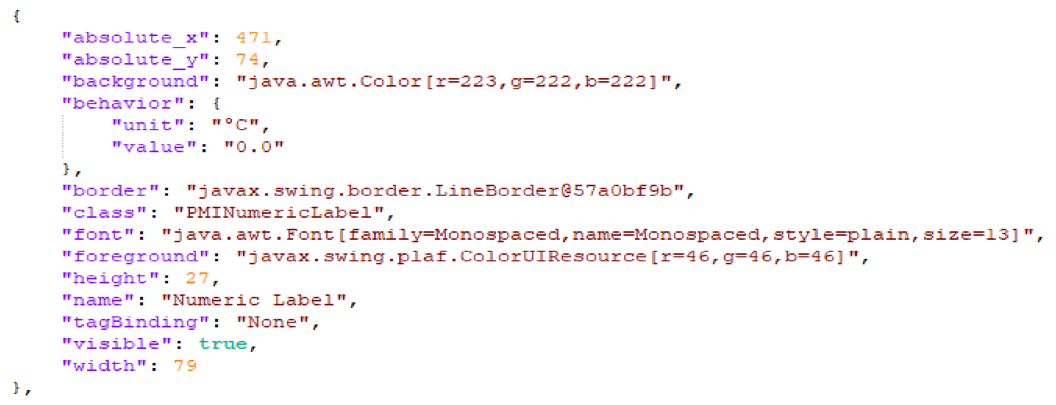

Ignition: Publish JSON data within Sparkplug B and MQTT
- Publish Sparkplug B + MQTT Topic in a publish-subscribe event-driven-achitecture
Ignition: Extract structure and data from instances
- Extract data values
- Extract data properties
- Extract graphical template and bindings
- Store JSON configuration
Ignition: Defining UDT Template and Instances
Motor UDT template case study example:
- State (5 states - part of the graphic descriptor changes colour according to states)
- Speed
- Temperature
- Motor command: start/stop
- Faults (PTC, overcurrent, leakage)
- Template - complete graphical represenation and bindings
MQTT Chariot Broker
Node-RED: Update User-Interface Dashboard
- Represent updated graphic descriptors associated with UDT template instance
Node-RED: Search and Update tag values
- Establish UDT structure (Set complete UDT at birth)
- Search and update tag values within UDT
- Convert graphic instance tag from String to JSON object
Node-RED: Get Sparkplug B - Store JSON
- Read and decode Sparplug B
- Get timestamp and metrics data
- Set JSON structure

Fig. 5.1-2 Functional diagram

The final extracted dataset comprises both static and dynamic properties. Static attributes include absolute and relative positions, dimensions, background and foreground colors, and font properties. These define the structural characteristics of elements and are essential for accurate visual representation. Dynamic attributes capture data-driven elements such as real-time values from tag bindings, component states, and specific object behaviors. Together, they establish a direct link between visualization and the functional layer of the monitored process, ensuring operational accuracy.

The extracted data is serialized into a hierarchical JSON format, fully compliant with Sparkplug B specifications. This compliance guarantees semantic compatibility with industrial automation frameworks and enables seamless integration into subsequent processing workflows. An element resulting from serialization, containing both visual and behavioral properties, is highlighted in Fig. 5.1-3. Structured data include both the static properties together with the behavioral ones.



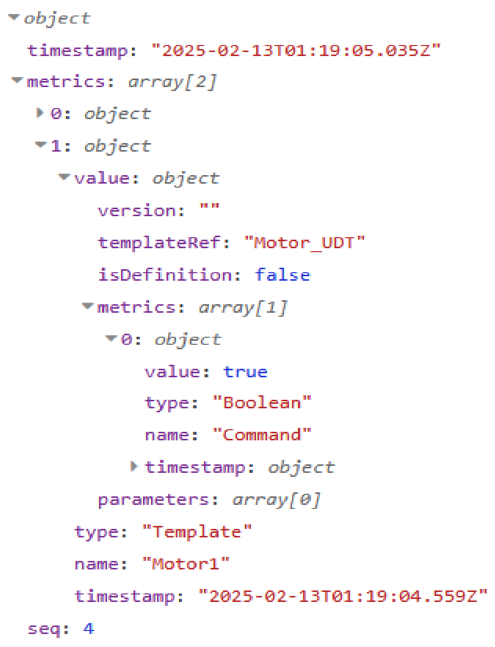
```
{
    "absolute_x": 471,
    "absolute_y": 74,
    "background": "java.awt.Color[r=223,g=222,b=222]",
    "behavior": {
        "unit": "°C",
        "value": "0.0"
    },
    "border": "javax.swing.border.LineBorder@57a0bf9b",
    "class": "PMINumericLabel",
    "font": "java.awt.Font[family=Monospaced,name=Monospaced,style=plain,size=13]",
    "foreground": "javax.swing.plaf.ColorUIResource[r=46,g=46,b=46]",
    "height": 27,
    "name": "Numeric Label",
    "tagBinding": "None",
    "visible": true,
    "width": 79
},
```

Fig. 5.1-3 JSON structure containing behavior properties

At the next stage, the JSON object is published into an MQTT tag within Ignition. During data distribution, the MQTT broker serves as the intermediary between Ignition and the final consumer, represented by Node-RED. Within the MQTT transmitter, correct metric encoding is essential. Ignition Sparkplug B introduces a birth certificate message upon initial publication, registering the UDT template instance as the active data source. The Node-RED consumer detects the availability of new data. Messages are structured into distinct metrics, including timestamps, element names and types, real-time tag values, and metadata, enabling the consumer to reconstruct historical data, track system changes, and maintain updates. In cases of failure or removal, a death certificate message is transmitted, signaling the consumer that the data source is no longer valid. This mechanism ensures continuous awareness of data validity, preventing reliance on outdated information.

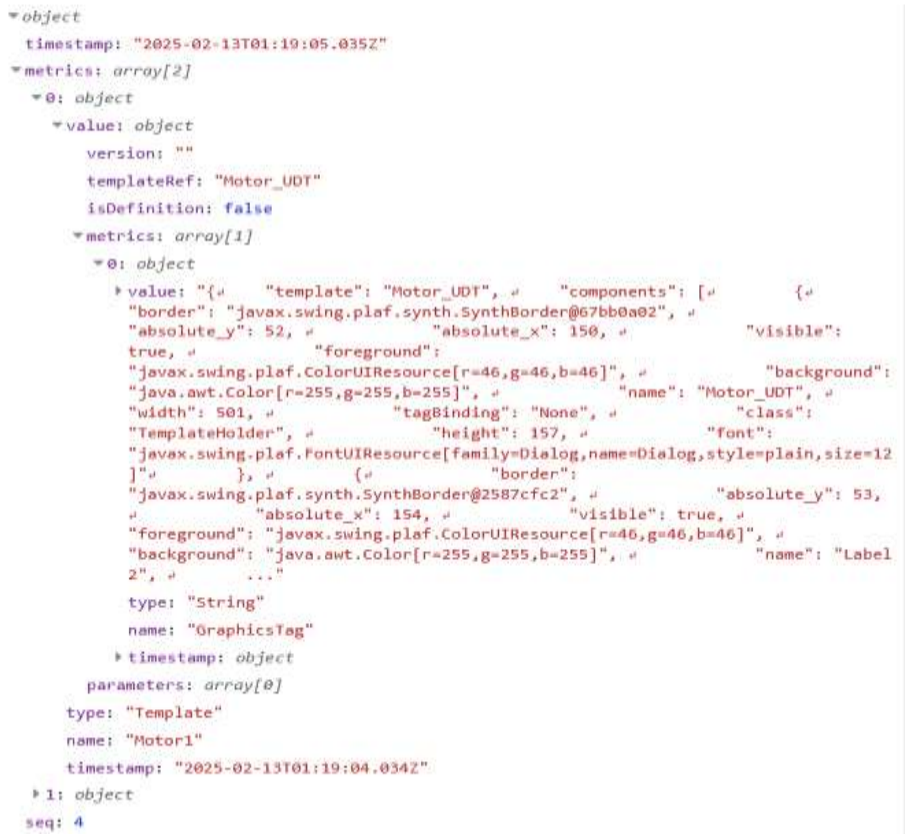
The second step involves the Node-RED client as the final consumer. Once the MQTT broker distributes the JSON payload, Node-RED parses the data, extracts relevant metrics, and prepares them for real-time visualization. Message structures received contain both visualization data and updated tag values, highlighted within Fig. 5.1-4 and Fig. 5.1-5. Each metric corresponds to a specific attribute of the UDT instance.

At this stage, Node-RED searches for updated tags within the UDT instance to synchronize with real-time data and apply graphical representation updates.



```
▾object
  timestamp: "2025-02-13T01:19:05.035Z"
 ▾metrics: array[2]
   ▸0: object
   ▾1: object
     ▾value: object
        version: ""
        templateRef: "Motor_UDT"
        isDefinition: false
       ▾metrics: array[1]
         ▾0: object
            value: true
            type: "Boolean"
            name: "Command"
           ▸timestamp: object
        parameters: array[0]
      type: "Template"
      name: "Motor1"
      timestamp: "2025-02-13T01:19:04.559Z"
  seq: 4
```

Fig. 5.1-4 JSON structure in Node-RED for key attribute



```
▾object
  timestamp: "2025-02-13T01:19:05.035Z"
 ▾metrics: array[2]
   ▾0: object
     ▾value: object
        version: ""
        templateRef: "Motor_UDT"
        isDefinition: false
       ▾metrics: array[1]
         ▾0: object
           ▸value: "{↵    "template": "Motor_UDT", ↵    "components": [↵        {↵
            "border": "javax.swing.plaf.synth.SynthBorder@67bb0a02", ↵
            "absolute_y": 52, ↵            "absolute_x": 150, ↵            "visible":
            true, ↵            "foreground":
            "javax.swing.plaf.ColorUIResource[r=46,g=46,b=46]", ↵            "background":
            "java.awt.Color[r=255,g=255,b=255]", ↵            "name": "Motor_UDT", ↵
            "width": 501, ↵            "tagBinding": "None", ↵            "class":
            "TemplateHolder", ↵            "height": 157, ↵            "font":
            "javax.swing.plaf.FontUIResource[family=Dialog,name=Dialog,style=plain,size=12
            ]"↵        }, ↵        {↵            "border":
            "javax.swing.plaf.synth.SynthBorder@2587cfc2", ↵            "absolute_y": 53,
            ↵            "absolute_x": 154, ↵            "visible": true, ↵
            "foreground": "javax.swing.plaf.ColorUIResource[r=46,g=46,b=46]", ↵
            "background": "java.awt.Color[r=255,g=255,b=255]", ↵            "name": "Label
            2", ↵        ..."
            type: "String"
            name: "GraphicsTag"
           ▸timestamp: object
        parameters: array[0]
      type: "Template"
      name: "Motor1"
      timestamp: "2025-02-13T01:19:04.034Z"
   ▸1: object
  seq: 4
```

Fig. 5.1-5 JSON structure in Node-RED for visualization data

The consumer identifies the relevant UDT instance via its unique identifier and modifies the associated tag values. Once synchronization is complete, the dataset is restructured into a new JSON payload, which serves as the foundation for interface construction. The interface is constructed using an SVG-based approach, dynamically generating UI components from the JSON payload. Each visual property is mapped onto the SVG canvas with precise positioning to ensure consistency. Path-based shapes are grouped within their parent containers, preserving hierarchical structure. All components are continuously updated in response to real-time data changes, maintaining a live connection to the underlying system state.

5.1.3 Case Study and Results

The solution was validated through a case study involving two main entities: the publisher and the subscriber, connected via an MQTT broker. The publisher is an Ignition SCADA (Vision), which communicates with the lower PLC layer using OPC UA. For testing, the environment employed a KepserverEx OPC UA server. Ignition transmits structured data to the Chariot MQTT broker using Sparkplug B over MQTT. The subscriber is a Node-RED application, which receives and processes the data, converting it into JSON format. The overall case study architecture is represented in Fig. 5.1-6.

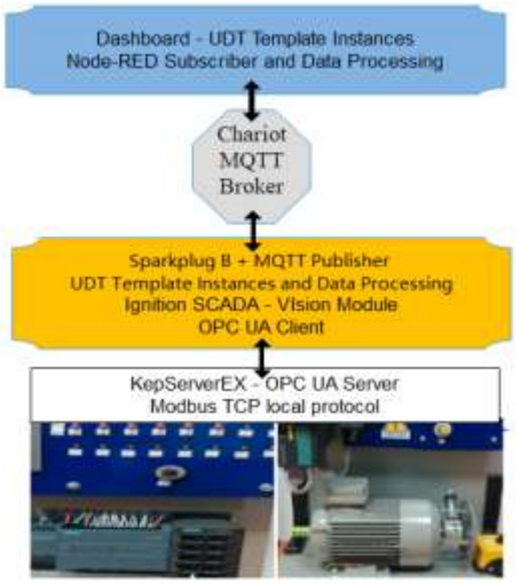

Dashboard - UDT Template Instances
Node-RED Subscriber and Data Processing
Chariot
MQTT
Broker
Sparkplug B + MQTT Publisher
UDT Template Instances and Data Processing
Ignition SCADA - Vision Module
OPC UA Client
KepServerEX - OPC UA Server
Modbus TCP local protocol

Fig. 5.1-6 Case study architecture

The scenario illustrates motor supervision. Unstructured data reaches the SCADA level, where it is structured in Ignition using a UDT that defines attributes such as temperature, speed (with units), motor state, motor control, and faults. Motor states are represented by five values (word 0–4), set according to the PLC algorithm and varying with motor speed. Fault detection considers three conditions (over-temperature, over-current, and leakage) identified through bit-wise analysis of the three least significant bits of the incoming word tag. Template graphics are linked to UDT in Ignition. A motor descriptor from the Symbol Factory is configured to bind with the motor state tag, changing color according to speed (e.g. yellow for 1500–2500 rpm, orange for 2500–300 rpm). Two LED displays are bound to the UDT's temperature and speed properties, while a start-stop button (linked to the least significant bit of the OPC UA tag) and a label are added.

UDT template instances were created, and a Jython scripting solution was applied to publish structured data, including template graphics and bindings, stored as a string attribute of the UDT. Data was transmitted via the broker, then extracted, processed, and represented in Node-RED, with updates reflected in the Node-RED Dashboard, closely mirroring the Ignition Window. Multiple tests across various local tag values validated the approach. The paper presents two scenarios, with the initial configuration showing three motor instances, each with distinct data values, as depicted in Fig. 5.1-7 and Fig. 5.1-8. Fig. 5.1-9 and Fig. 5.1-10 illustrate the resulted dashboard in Node-RED, showing correct, structured and complete data representation (e.g. the state colors of motor 1 is yellow while the speed is 2000 rpm).

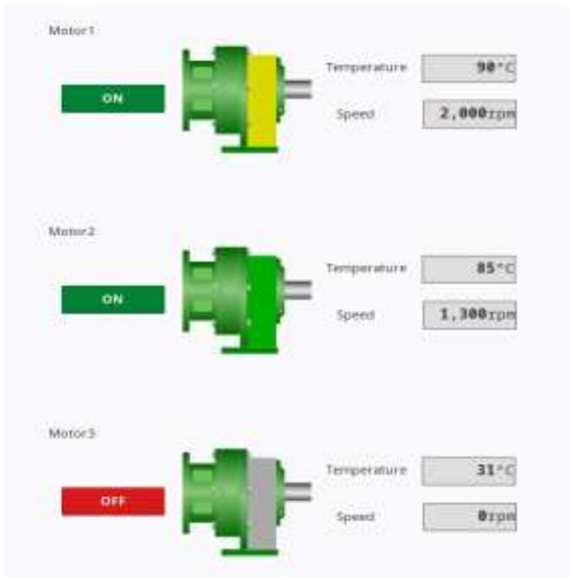


Fig. 5.1-7 Three motor UDT template instances in function in Ignition Window – scenario 1

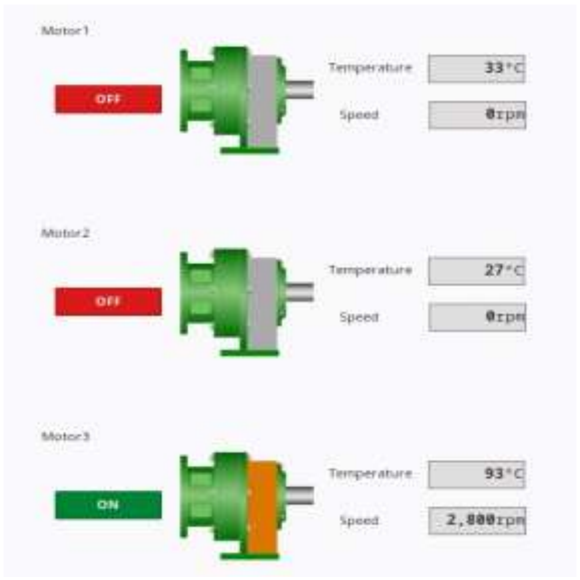


Fig. 5.1-8 Three motor UDT template instances in function in Ignition Window – scenario 2

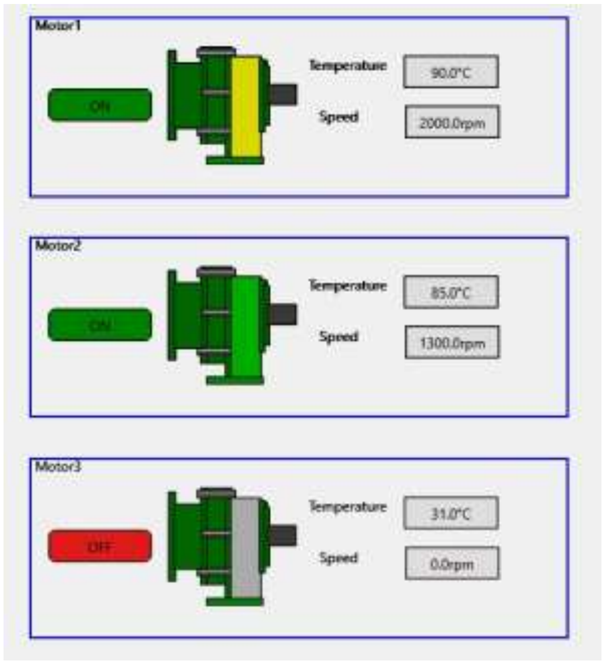


Fig. 5.1-9 The three UDT template instances shown automatically in Node-RED – scenario 1

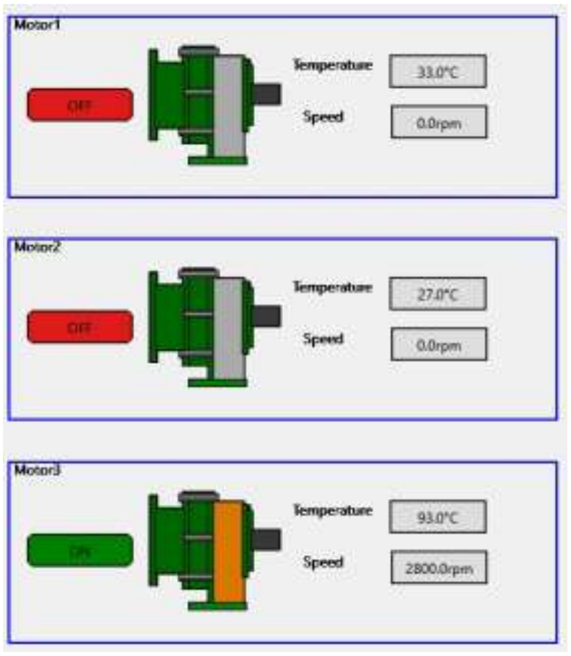


Fig. 5.1-10 The three UDT template instances shown automatically in Node-RED – scenario 2

5.2 Data Propagation on the Operational Technology Level Based on OPC UA Interfacing, within a Case Study Using Node-RED and Ignition.

Considering both legacy systems and technological progress, the section proposes a solution that assures efficient and human centered structured, contextualized, and graphically sustained data propagation on the OT level. This work builds upon industry-adopted environments and protocols to ensure rapid adoption and broad impact. A Node-RED and Ignition case study demonstrates PLC–SCADA data integration, with structured and graphically represented data transfer via OPC UA, yielding positive results without requiring additional SCADA-level developments.

With the Purdue model dominating the OT level, newer architectures sometimes side-push SCADA systems [185] through direct PLC data transfers. Security concerns and protocol-related advantages often position SCADA as a pass-through layer for vertical data movement. Emerging unified virtual space/unified namespace architectures [186] also incorporate SCADA levels, typically local HMIs and supervisory systems, but rely on decoupled entities and middleware. In some cases, isolating the PLC level leads to local HMIs functioning as pass-through SCADA structures. Across architectures, data structuring, quality, adaptability, and human-centricity must be prioritized. A bottom-up approach to structuring data enhances higher-layer integration, enabling improved usage, comprehension, and sustainable AI deployments at edge/fog levels. When data structures include graphical representations of protocol-interfaced entities, all consumers, including human operators, gain a unitary perception of process entities. Since control strategies and representations developed at the PLC level are closest to the technological process, they provide the most meaningful insights.

Cost considerations remain critical. Implementing multiple SCADA solutions across hierarchical levels often duplicates process and control structures, leading not only to development costs but also to licensing and maintenance expenses, particularly for local automation panel HMIs. Additional software layers within HMIs may reduce reliability, availability, and security, especially when upgrades are required. An essential aspect of the current digital transformation trend is the choice between off-the-shelf products proven competitive and flexible [187], or high-TRL solutions capable of rapid implementation. Legacy OT systems have long lifecycles, and changes involve significant costs following detailed cost–benefit analyses. Conversely, the cost of waiting can also be substantial in the context of Industry 4.0 and 5.0 requirements, which limit protocol improvements or redefinition of software environments [188]. Academic research must therefore evaluate whether full reinvention of protocols and environments is necessary, or if add-on improvements to existing developments are sufficient. Within this context, a foreseen human-centered Industry 5.0 solution aims to ensure: Structured, contextualized, and graphically represented data propagation at the OT level; Fast deployment by leveraging existing industrial technologies and equipment; Large impact through reliance on key industrial protocol concepts; Efficiency in time, cost, and processing for SCADA development, maintenance, and data manipulation.

5.2.1 Actual Status of Literature and Industry

The OPC UA protocol was the key enabler of Industry 4.0 and IIoT [189–190], succeeding the influential OPC Classic, which, through centralized servers, opened industrial connectivity for major producers and new enterprises. Transitioning to OPC UA at the OT level has faced persistent challenges due to the inertia of legacy systems and older products [188]. While Modbus TCP, Profinet, Ethernet/IP, and other Ethernet-based protocols remain consolidated at PLC levels, lower automation layers still rely on serial Modbus, Profibus, Mewtocol, Canopen, etc. Despite this, OPC UA has become the widespread IIoT interface in manufacturing and other industries, with numerous wrappers developed. It is now present on most PLCs, where equipment typically follows basic client–server specifications with fixed structuring.

At the OT–IT convergence, the Unified Namespace concept [183] emerged, often broker-based with MQTT and Sparkplug B packing, promoted by industrial products such as Ignition from Inductive Automation. Ignition gained broad adoption in industries using OPC UA, valued for its namespace, address space, stability, and security, while Sparkplug B achieved popularity in specific domains. In [191], process data is transmitted via MQTT, packed in Sparkplug B, and visualized through Node-RED and Ignition. Although OPC UA has advanced with publish–subscribe specifications, few industrial solutions implement full end-to-end OPC UA publish–subscribe systems. Work [192] demonstrates such an approach, using MQTT for wireless sensor networks within the industrial Internet.

Sparkplug B is not a widely adopted protocol. Some issues were discussed and need to be solved: Birth-storms occurrence when the main client/subscriber disconnects, forcing all publishers to resend birth certificates, which can overwhelm the network; Data loss risks if the primary client crashes, as publishers stop transmitting under the store-and-forward mechanism, with quality of service remaining limited; Integration challenges exist for products not specifically designed for the protocol, and auto-discovery of complex data structures is difficult; UDTs are transferred entirely from publisher to subscriber, with no intrinsic selection capability; the rigid topic structure lacks the flexibility and address space of OPC UA; Data contextualization is constrained, limited to a few levels and largely dictated by the publisher.

Interfacing research and development requirements have shifted the PLC-level perspective. Phoenix Contact integrated MQTT, Sparkplug B, and OPC UA into PLCnext technology, while current PLC equipment benefits from enhanced interfacing through Node-RED and Codesys. Companies such as Siemens and Kunbus provide solutions (via IoT2000 series gateways for protocol conversion and wrapping [K-31], and Revolution Pi PLCs) capable of complex interfacing, data structuring, and representation. These newer devices, whether PLCs or gateways for legacy systems, often embed Node-RED, which research highlights as a PLC interfacing solution [193–194]. Beyond interfacing, Node-RED can support complete control strategies, including data structuring and contextualization within the PLC. Node-RED also enables graphical representation of processes and components [195], [K-1], a capability critical for Industry 5.0, emphasizing efficiency, human-centricity, sustainability, and resilience. Traditionally, automation/SCADA setups involve local panels with a PLC and HMI, requiring licensing, proprietary development environments, and additional tasks. Higher SCADA levels in OT introduce further HMI requirements and OPC UA interfacing, often with multiple HMI/SCADA systems (e.g. [196] shows additional equipment efficiency supervision in Ignition SCADA). These systems typically lack unitary structuring and contextualization, while incurring separate licensing, development, and maintenance costs. Industrial practice and research trends highlight the need for impactful solutions. Leveraging Ignition SCADA's effectiveness and Node-RED's PLC-level capabilities offer benefits: implementing process control, data structuring/contextualization, supervision strategies directly at the PLC level, while propagating structured data and graphical representations to SCADA via OPC UA, without requiring supplementary SCADA development.

5.2.2 Proposed Solution

The proposed solution primarily targets the OT level, while ensuring that structured and contextualized data also reaches the IT level. Structured instances and attributes are generated at the PLC level in Node-RED and encapsulated within the OPC UA protocol. On the OPC UA client side, represented by Ignition SCADA, the data is accessed and visualized through the corresponding processing module. The targeted OT architecture (Fig. 5.2-1) assures control the local process at the PLC level, which is approached in two ways (modern PLCs supporting IIoT environments such as Node-RED with data packed in OPC UA, or legacy PLCs with gateways/wrappers).

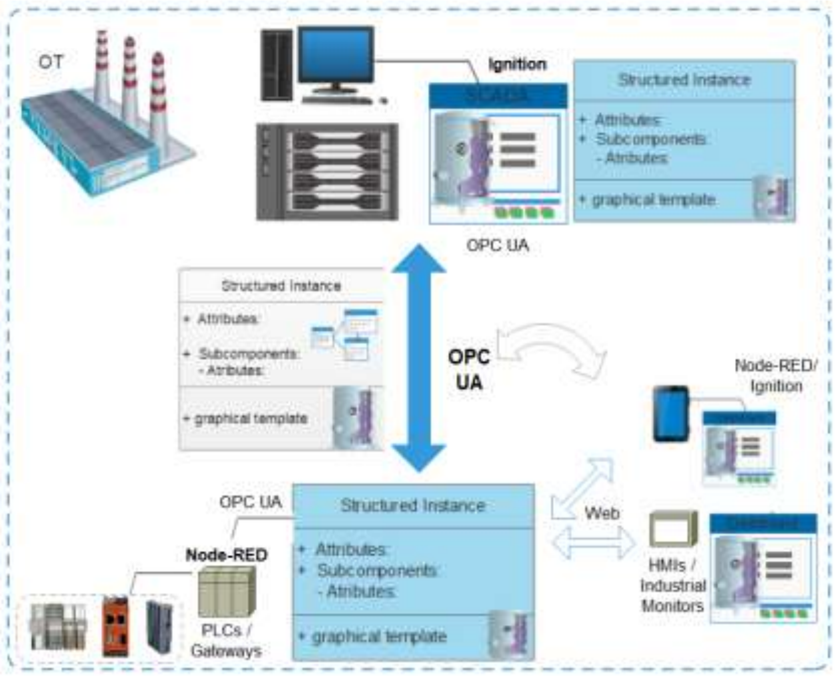


Fig. 5.2-1 System architecture on the OT level

At the SCADA level, the OPC UA server provides access to structured instances within the address space. Processing modules in Ignition retrieve these instances and depict the complete structure, including graphical representations, without requiring additional synoptic scheme development. Other HMIs, whether based on Ignition or Node-RED, can interface and subscribe to the exposed instances via OPC UA with minimal effort, or alternatively access the full Node-RED dashboard through a local browser.

The solution follows a two-step approach, with the implemented data workflow presented in Fig. 5.2-2, integrating Node-RED and Ignition through OPC UA.

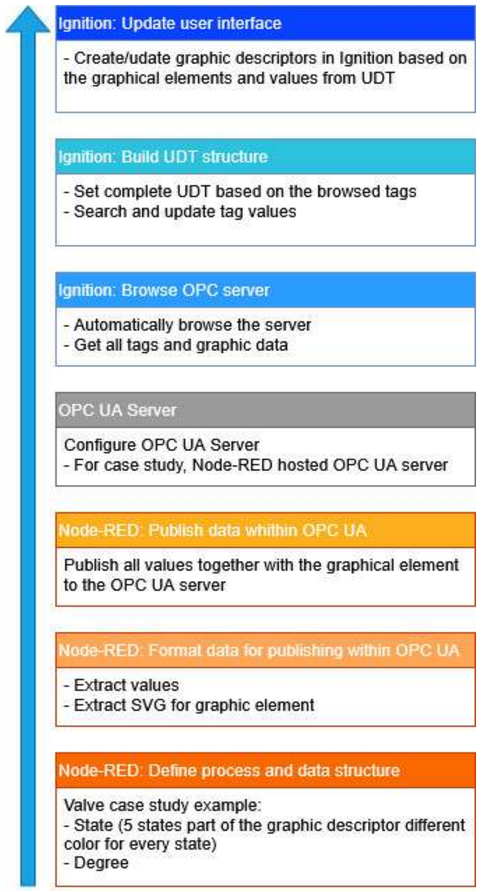

Ignition: Update user interface
- Create/udate graphic descriptors in Ignition based on the graphical elements and values from UDT
Ignition: Build UDT structure
- Set complete UDT based on the browsed tags
- Search and update tag values
Ignition: Browse OPC server
- Automatically browse the server
- Get all tags and graphic data
OPC UA Server
Configure OPC UA Server
- For case study, Node-RED hosted OPC UA server
Node-RED: Publish data whithin OPC UA
Publish all values together with the graphical element to the OPC UA server
Node-RED: Format data for publishing within OPC UA
- Extract values
- Extract SVG for graphic element
Node-RED: Define process and data structure
Valve case study example:
- State (5 states part of the graphic descriptor different color for every state)
- Degree

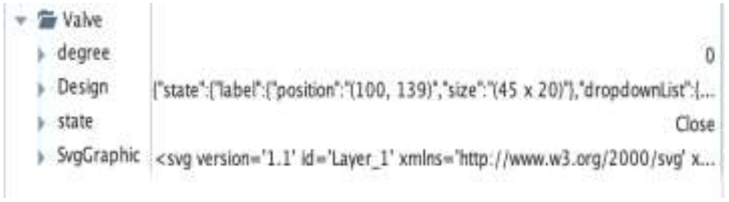
Fig. 5.2-2 System architecture on the OT level

The first step involves defining the data structure in Node-RED, which communicates directly with field devices, polling data and publishing it to the OPC UA server. Node-RED also functions as an independent dashboard, offering live monitoring of process variables through built-in web visualization tools without additional software. In the initial phase, data is organized into logical groups before publication. The processed dataset encapsulates both behavioral parameters (real-time values) and structural parameters (graphical representations, location, dimensions), categorized numerically for standardized OPC UA integration. Visualization employs SVG elements to represent monitored components. This approach enables dynamic updates by embedding color-coded attributes linked to behavioral data, reflecting real-time status. SVG’s XML foundation ensures flexible manipulation while preserving hierarchical structure, position, size, color attributes, eliminating the need to redraw components during updates. The final dataset is integrated into the OPC UA framework, which offers features absent in Sparkplug B, such as address-space subscriptions, flexible structures, methods, robustness, and protocol-level security (encryption and authentication).

The second step is represented by the Ignition client, acting as consumer and visualization engine. Structured data received via OPC UA is transformed into a dynamic user interface through two complementary functions:

- Automatic creation of UDTs, organizing process parameters in a standardized, scalable manner.
- Independent UI rendering, querying structured data to generate components, allowing modifications without altering the structure.

UDTs are created dynamically by detecting available server nodes and retrieving their values. Each UDT represents a specific component, grouping tags rather than treating them individually. This automatic process ensures continuous updates when new data is added, providing a scalable and adaptive system. Once instantiated, a UDT structure is highlighted in Fig. 5.2-3.



Valve	
degree	0
Design	{"state":{"label":{"position":"(100, 139)","size":"(45 x 20)"},"dropdownList":{...
state	Close
SvgGraphic	<svg version='1.1' id='Layer_1' xmlns='http://www.w3.org/2000/svg' x...

Fig. 5.2-3 UDT Instance

At this stage, the user interface is dynamically generated once a window component is provided. The automatic process follows two steps: first, the complete UDT structure is retrieved and its components separated into numeric values, operational states, and graphical elements; second, these components are created and placed within the UI. Existing elements are only updated rather than rebuilt. During UI creation, parameters are iterated and type-verified. Numeric values trigger automatic generation of text fields with real-time display and measurement units. States are represented by multi-state display elements, with text and color linked to the corresponding state. Labels and descriptors are also dynamically created to provide contextual information. The use of SVGs enables dynamic generation of graphical elements representing physical components. Their hierarchical, self-contained structure eliminates the need for additional rendering techniques, while properties such as position, size, color can be modified.

5.2.3 Case Study and Results for Node-RED to Ignition Data Propagation using OPC UA

The case study is based on a PLC-level Node-RED development that exposes structured data through the OPC UA server, and a SCADA-level Ignition implementation that accesses these structured instances via its embedded OPC UA client. Ignition then automatically deploys the selected data and its graphical representation within the SCADA diagram. The Node-RED to Ignition data flow architecture is illustrated in Fig. 5.2-4. Both PLC level approaches can be observed, with the newer PLCs embodying Node-RED, and also legacy PLCs with legacy protocols augmented with the gateway/wrapper equipment.

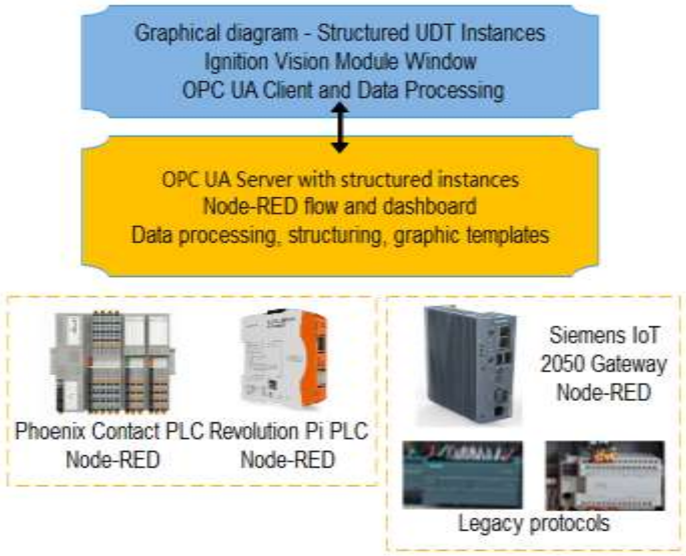

Graphical diagram - Structured UDT Instances
Ignition Vision Module Window
OPC UA Client and Data Processing
OPC UA Server with structured instances
Node-RED flow and dashboard
Data processing, structuring, graphic templates
Phoenix Contact PLC
Node-RED
Revolution Pi PLC
Node-RED
Siemens IoT 2050 Gateway
Node-RED
Legacy protocols

Fig. 5.2-4 Case study Node-RED to Ignition OPC UA data flow architecture

The case study examines a control valve within a technological process. The valve operates in five states: open, closed, opening, closing, and faulted. It provides also the degree of opening, expressed in percentage units (%). These attributes are organized into a structured dataset. In addition to values and contextual data, Node-RED integrates a graphical template for the valve. The template displays the valve's current state name and degree of opening with units, alongside an associated image. Color coding indicates the state (e.g. green = open, grey = closed).

Tests confirm that valve instances generated at the PLC level are successfully integrated and represented within the SCADA application. Fig. 5.2-5 illustrates the local Node-RED scenario, where the valve transitions from the open state to the closed state, subsequently entering the fault state. Fig.

5.2-5 a)-d) are showing the template instance on the Node-RED dashboard in all states.

The instance is integrated into SCADA through a drag-and-drop process after accessing the OPC UA server. Data is then automatically propagated and graphically represented, flowing from the Node-RED PLC level to the Ignition SCADA level. The results associated to the scenario from Fig. 5.2-5 are presented in the Ignition window in Fig. 5.2-6. Fig. 5.2-6 a)-d) illustrate the instance template propagation within the Ignition window, depicting the valve's transition from open to closed, and subsequently to the fault state

From Fig. 5.2-5 and Fig. 5.2-6, the structured data and its graphical template are correctly imported from Node-RED into Ignition, updating dynamically according to the local valve behavior.

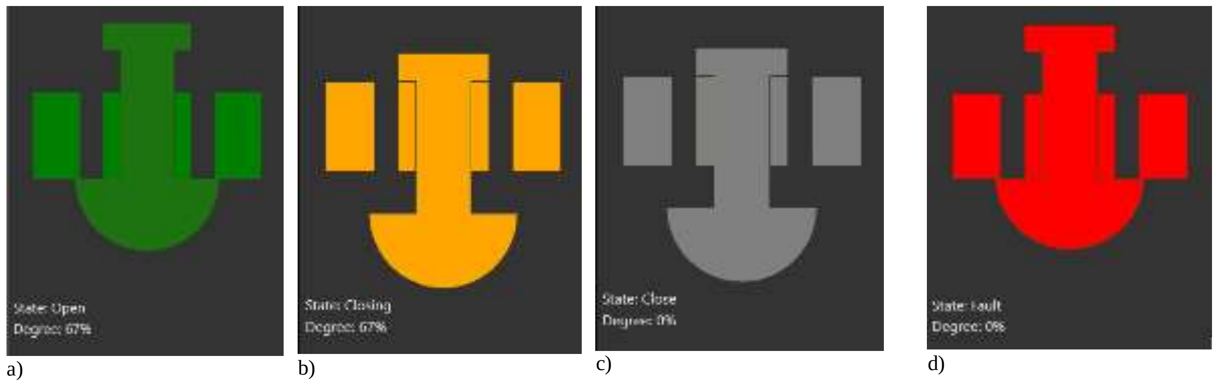
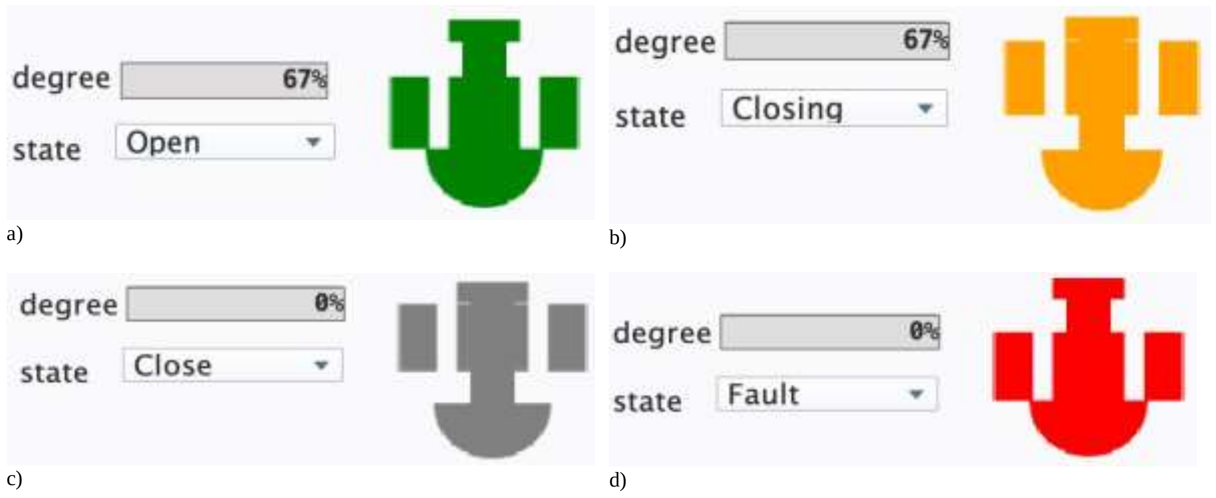


Fig. 5.2-5 Valve template instance representation in the Node-RED dashboard - a) open, b) closing, c) close, d) faulted.


a) degree 67% state Open

b) degree 67% state Closing

c) degree 0% state Close

d) degree 0% state Fault

Fig. 5.2-6 Valve template instance representation in the Ignition window - a) open, b) closing, c) close, d) faulted.

5.3 Solving and Completing Structured Bidirectional Data Propagation and Representation in the Sparkplug B context, using Ignition and Node-RED.

Digital transformation, like other strategic concepts, advances through incremental steps toward defined objectives, though the extent of achievement and precision of vision remain variable. To align with Industry 4.0 and 5.0 paradigms [197], research must emphasize efficiency gains alongside human-centered, sustainable, and resilient solutions. Since data constitutes the foundation of digital transformation, its propagation across hierarchical layers, spanning OT and IT, is critical. Each level presents distinct requirements in terms of protocols and information availability, with OT systems often subject to stricter security measures.

Rapid deployment and adoption in this context call for high-TRL, off-the-shelf environments as research infrastructure. Designing robust architectures and enhancing data interfacing and representation provide significant advantages. Within this framework, Ignition [191] and Node-RED [196] exemplify flexible platforms that support IIoT/IoT-driven digital transformation. Both environments integrate modern standards such as Sparkplug B and OPC UA, enabling scalable and interoperable solutions.

Broker-based EDA, emphasizing decoupled entities, increasingly dominate strategies for single-source-of-truth (SSoT) approaches within Unified Virtual Space (UVS) or Unified Namespace (UNS). With limited academic involvement, most architectural evolution is driven by industry, often leading to flawed comparisons between protocols such as MQTT and OPC UA. Industrial practice highlights that MQTT payloads require higher-level structuring through protocols like Sparkplug B or OPC UA [198], and certain sectors demand distributed architectures with multiple UNSs.

The research addresses efficiency and Industry 5.0 objectives by ensuring data is structured, contextualized, trusted, visually enriched, and propagated across hierarchical levels. Template-based instances provide unified perspectives of technological processes, enhancing human involvement in decision-making and enabling decentralized AI under human guidance. Structured SSoT data availability also strengthens resiliency and sustainability by improving operational insight and maintenance interventions. Academic validation remains essential to consolidate technologies, establish proper timelines, and mitigate risks of proprietary dependence. The current third stage of the research delivers a comprehensive Sparkplug B/MQTT solution, enabling bidirectional, flexible, and contextualized data propagation across all levels. Templates generated in Node-RED or Ignition are distributed to consumers, who may also act as publishers, ensuring updated structures are re-propagated. With built-in monitoring for traceability and security, the solution supports transition toward digital passports for process components while reducing SCADA development, maintenance, and HMI-related costs.

Legacy systems remain a significant challenge at the OT level. However, over time, research and industrial development have established OPC UA as the dominant protocol. This transition was neither smooth nor spontaneous, requiring strong industry support alongside academic studies and endorsements. Once its advantages were recognized, adoption accelerated rapidly, resulting in widespread deployment of client–server OPC UA systems. Although newer OPC UA publish–subscribe specifications have been developed and studied, their integration into industrial devices has been slow. Instead, industry trends increasingly favor Sparkplug B, driven by SCADA-level implementations. OPC UA traditionally addressed higher OSI layers, extending its publish–subscribe mechanisms to the transport layer via MQTT or UDP. In contrast, Sparkplug B was designed from the outset around MQTT, promoting a decoupled entity architecture. MQTT itself has gained dominance across IT-level applications—both cloud-based and on-premises—despite persistent legacy challenges [199]. It has also penetrated lower-level OT equipment such as meters and transducers, and is strongly endorsed by leading companies as a foundation for digital transformation. Literature consistently identifies MQTT as one of the most advantageous technologies due to its simplicity, lightweight design, and efficiency in constrained environments [200].

While protocol specifications continue to evolve, their practical impact depends on full adoption within industrial products. Theoretical standards alone, even when supported by SDKs such as open62541, remain insufficient without commercial implementation. Current digital transformation efforts rely heavily on off-the-shelf solutions, yet without academic validation and improvement, progress is largely steered by industry influencers and existing systems. The scarcity of research addressing a common language between OT and IT exacerbates this issue, leaving product-owning companies to dictate directions for manufacturing and related sectors. Consequently, the landscape remains fragmented, with urgent demand for rapid deployment of digital transformation modules.

Although Sparkplug B requires integration with Ignition to enable straightforward UDT manipulation, Node-RED has emerged as a powerful IoT environment bridging both OT and IT domains. With extensive interfacing and data-handling capabilities, Node-RED plays a central role in digital transformation. It is increasingly embedded in gateway products and PLCs, while industries and developers rapidly expand its applications. By promoting openness and avoiding proprietary lock-ins, Node-RED enhances independence, resilience, sustainability, and human-centered system design.

Within this context, UNS/UVS concept is critical to SSoT architectures. Positioned at the OT–IT convergence, UNS/UVS provides structured and contextualized data for all consumers. High-TRL implementations typically rely on MQTT brokers, requiring higher-level protocols to enrich raw payloads. Sparkplug B combined with MQTT represents one of the dominant UNS architectures, transmitting OT data independently of Purdue hierarchies. Proper structuring of MQTT remains essential for interoperability, with Sparkplug B offering a technology-oriented solution.

Beyond OT–IT convergence, UNS/UVS supports Industry 5.0 pillars of human centricity, sustainability, and resilience. Graphical representations linked to contextualized data enhance human centricity, providing unified process knowledge and empowering personnel without requiring direct manipulation of low-level infrastructure. Efficiency is achieved through reduced hardware, licensing, and development costs, coupled with instant access to complete process data. Openness and flexibility, ensured by environments such as Node-RED, allow continuous evolution and bidirectional updates of tag values and graphical interfaces. Such capabilities, whether implemented in Node-RED or Ignition, represent novel contributions not yet fully explored in the literature.

5.3.1 Solution Overview

The proposed solution employs an event-driven architecture built on decoupled entities. At its core lies a UVS/UNS implemented through MQTT and Sparkplug B, functioning as the SSoT. Within this framework, data are structured and contextualized, ensuring consistency across all system components. The shared information encompasses both functional attributes and graphical representations of technological process objects, enabling comprehensive interoperability and visualization.

As illustrated in Fig. 5.3-1, the architecture relies on a UVS/UNS core, where standardized data exchange is enabled through the Sparkplug B protocol. Edge-generated UDT instances, structured, contextualized, and graphically represented, can be modified by authorized data producers and consumers, with auditing procedures serving as initial steps toward digital passport implementation. This design supports lightweight, event-driven, flexible, and scalable information exchange across heterogeneous components.

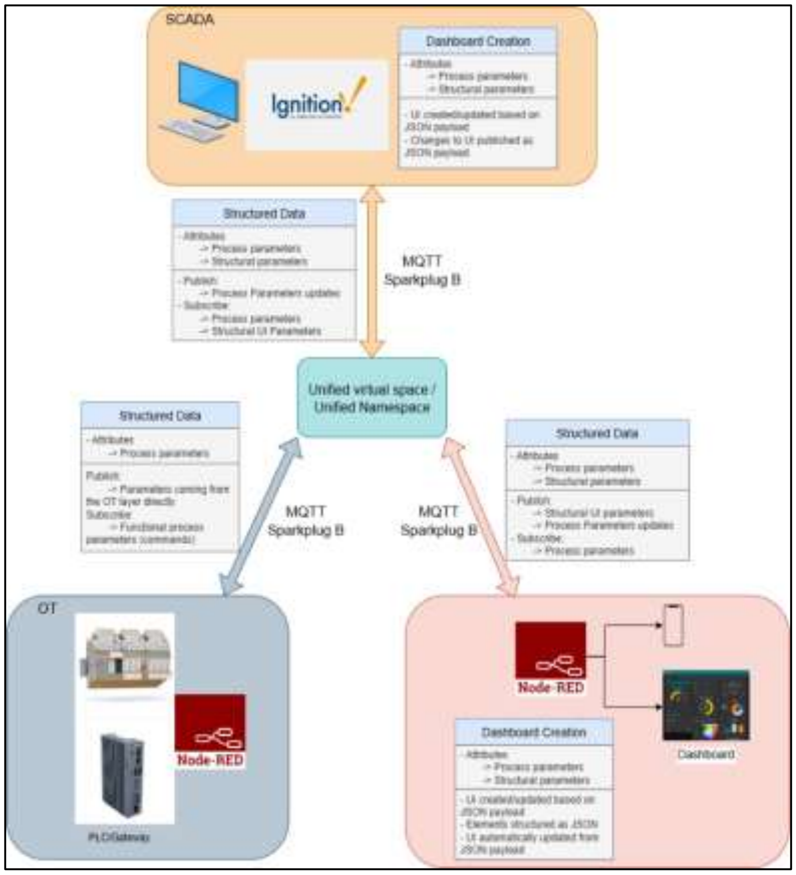


Fig. 5.3-1 System architecture

At the OT layer, programmable logic controllers (PLCs) serve as the primary components, either operating with Node-RED as middleware or functioning directly as programmable environments (e.g. Phoenix Contact PLC, Revolution Pi). This layer generates event-driven structured data and attributes, which are routed into the UVS/UNS. Through Sparkplug B abstraction, contextual information and real-time values are standardized regardless of source. Node-RED thus acts as an OT-level force for UDT structuring and graphical dashboard generation, while also functioning as an IT-level data consumer or an OT-level SCADA entity, with capabilities to publish or modify data when required.

Ignition is typically regarded as an OT SCADA-level component, though it is increasingly deployed at the PLC level (e.g., Groov Epic) and within IT environments [196]. It interacts with the UVS/UNS to read and write data, supporting both process control and dashboard construction. Functional information (e.g. dynamic process variables) and structural information (e.g. UI definitions) are integrated as JSON objects, enabling dynamic modification or creation of payloads for flexible dashboard reconfiguration without disrupting process integrity. In parallel, Node-RED deployments build dashboards and manage control tasks. Both Ignition and Node-RED operate concurrently, replicating functionalities and subscribing to a unified namespace. As a result, updates to process parameters and user interfaces are consistently propagated, ensuring synchronized and current system views across all components.

The implementation of the architecture was realized in two stages, with two self-descripting entities, one in Node-RED and one in Ignition Perspective. The implementation centers on bidirectional synchronization between Node-RED and Ignition, both capable of generating template instances. Instance definitions, comprising graphical components, structural values, and metadata, are serialized into JSON payloads and propagated to the UNS. Consumer environments subscribe to these payloads, reconstruct dashboards upon changes, and act as secondary visualization layers. Once dashboards are registered in the UNS, any modification is automatically serialized, published, and synchronized across platforms, eliminating dependency on a single tool and ensuring independent interoperability.

To move beyond static interfaces, dashboards are treated as structured data objects. Each element, symbols, buttons, or indicators, is assigned metadata describing type, tag, position, and dimension. Attributes are extracted at runtime via the Document Object Model (DOM), with coordinates, bounding boxes, and process parameter values systematically encoded into hierarchical JSON objects. This serialization produces a digital twin of the dashboard, transmitted as a Sparkplug B payload to the UNS, where it becomes accessible to all subscribing entities capable of interpretation.

To strengthen industrial compliance and system robustness, a logging and traceability mechanism was integrated for all components. Each structural modification to a dashboard definition is recorded with its timestamp and published to the UNS on a dedicated topic, ensuring auditable evidence of changes. This functionality is particularly relevant in regulated industrial contexts and represents an initial step toward the digital passport concept [201]. The mechanism functions symmetrically across environments, capturing both user-triggered and script-based events. These are serialized into JSON entries and disseminated via MQTT, making traceability data instantly available to consumers. Simultaneously, all records are stored in a database, creating a historical archive for long-term auditing and analysis. Each entry includes contextual details such as the originating platform (Node-RED or Ignition) and precise timestamps, thereby clarifying authorship, reducing ambiguity, and supporting troubleshooting.

From an operational perspective, the mechanism ensures compliance with industry standards requiring auditable change records. By embedding traceability data into the UNS alongside process and structural information, the solution elevates its importance within the digital transformation framework. The overall solution implementation is illustrated in Fig. 5.3-2.

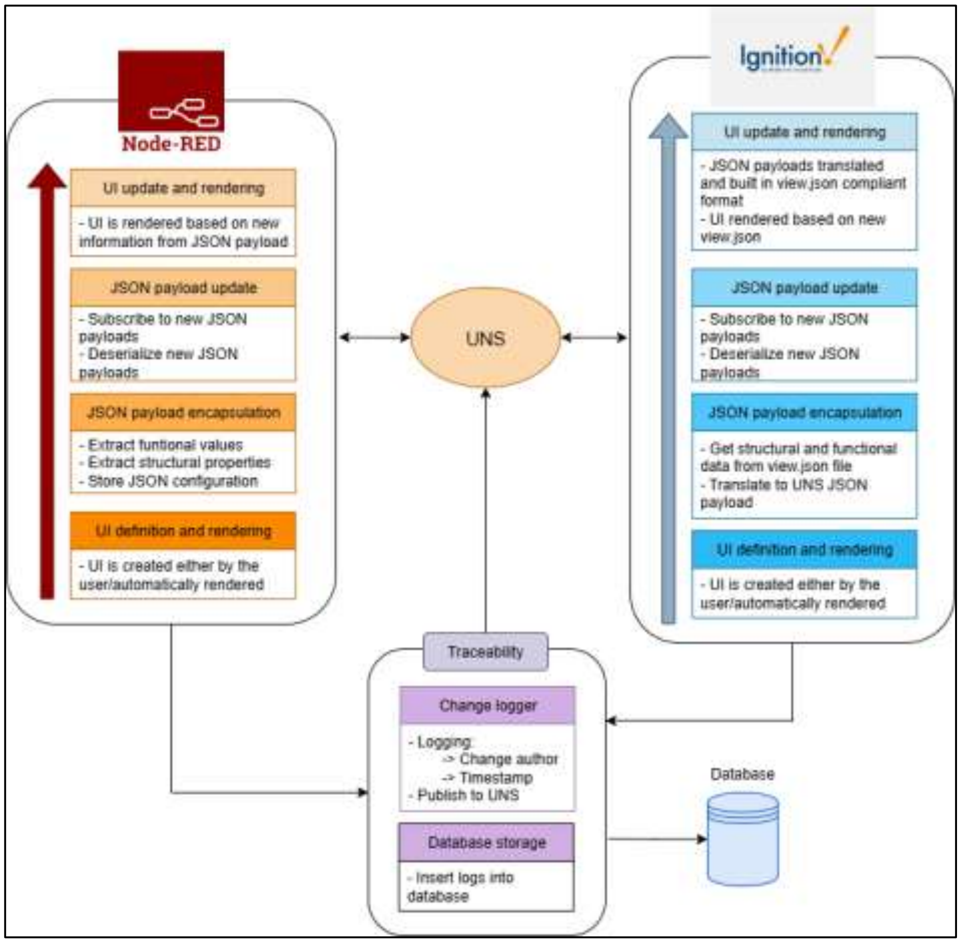

Node-RED
UI update and rendering
- UI is rendered based on new information from JSON payload
JSON payload update
- Subscribe to new JSON payloads
- Deserialize new JSON payloads
JSON payload encapsulation
- Extract funtional values
- Extract structural properties
- Store JSON configuration
UI definition and rendering
- UI is created either by the user/automatically rendered
UNS
Ignition
UI update and rendering
- JSON payloads translated and built in view.json compliant format
- UI rendered based on new view.json
JSON payload update
- Subscribe to new JSON payloads
- Deserialize new JSON payloads
JSON payload encapsulation
- Get structural and functional data from view.json file
- Translate to UNS JSON payload
UI definition and rendering
- UI is created either by the user/automatically rendered
Traceability
Change logger
- Logging:
-> Change author
-> Timestamp
- Publish to UNS
Database storage
- Insert logs into database
Database

Fig. 5.3-2 Functional diagram

5.3.2 Case Study and Results

The first case study demonstrates a PLC-level Node-RED development that exposes structured and contextualized data into an MQTT-based UNS using the Sparkplug B standard. An Ignition layer, acting as a Sparkplug B client, consumes this data. Node-RED generates the datatype and initial instances, while subscribers automatically deploy the structured payload, including graphical diagram representations, into Perspective views. Data propagation occurs only upon publisher-side changes, and any consumer may also act as a publisher. Thus, Node-RED and Ignition function as both subscribers and publishers, achieving bidirectional synchronization after initialization, independent of the original data source.

The case study focuses on pump-related process data, including attributes such as speed, pumped flow rate, measurement units, states, controls, and faults. Four operational states (Word-type values 0–3) were defined, reflecting common automation practices, with transitions linked to speed levels. Three faults, over-temperature, over-current, and leakage, were encoded bit-wise in Word tags. All data were associated with graphical representations and published within the UNS as complete dashboard instances.

The scenario validated correct data generation and synchronization between Node-RED and Ignition. As shown in Fig. 5.3-3, all elements, including the graphical pump symbol, control button, and analog displays for speed and flow, were consistently reproduced in Ignition. Real-time process values were directly bound to Ignition tags, ensuring system-wide data consistency.

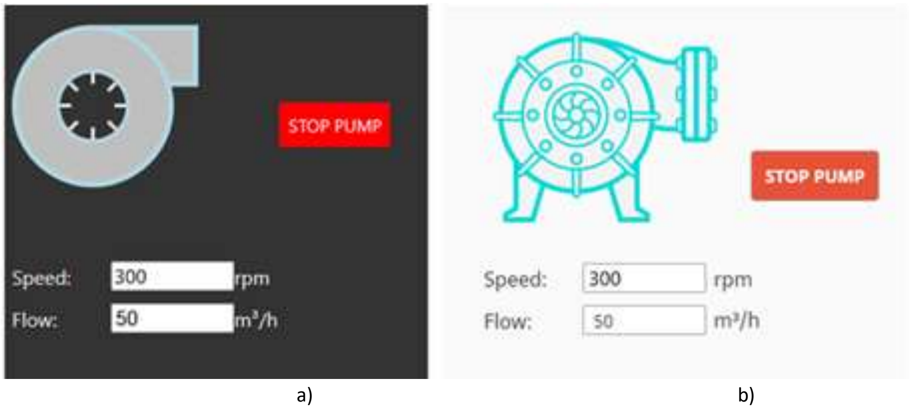


Fig. 5.3-3 Pump UI representation in: a) Node-RED dashboard, b) Ignition Perspective view

The second scenario evaluated the system's ability to propagate and respond to structural modifications. In this demonstration, graphical elements in the Node-RED dashboard, specifically the flow label and text field, were repositioned downward. The updated JSON payload was then published by Node-RED and, upon receipt, Ignition reprocessed the *view.json* file to update its interface. As shown in Fig. 5.3-4, the flow display elements were correctly repositioned within Ignition to match the Node-RED layout. No inconsistencies were detected, confirming the synchronization mechanism's effectiveness in handling real-time structural changes.

The third scenario examined resizing operations, beginning with the dimensions shown in Fig. 5.3-5. In this case, the speed label component was resized within the Node-RED dashboard. The modification triggered publication of a new JSON payload, which was subsequently ingested by

Ignition. The Ignition view accurately reflected the resized element, preserving readability and structural coherence.

Across all tested scenarios, dynamic bindings to process data tags remained intact and unaffected by structural changes, thereby ensuring consistent functionality and maintaining data integrity throughout interface adaptations.

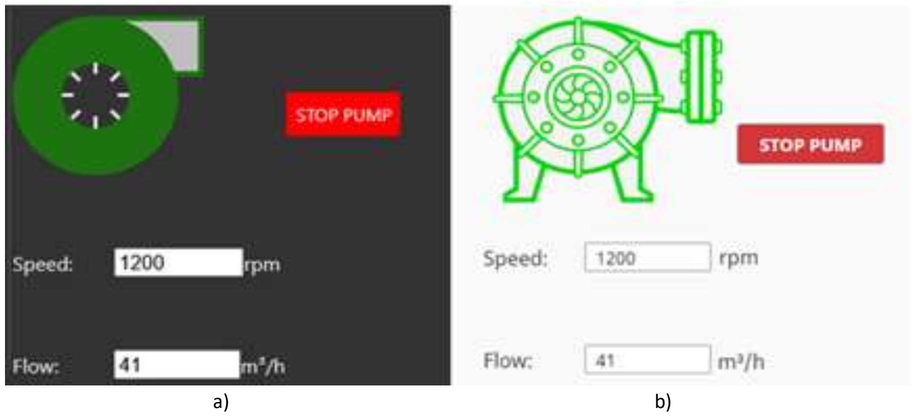


Fig. 5.3-4 Pump UI representation after increased spacing in: a) Node-RED dashboard, b) Ignition Perspective view

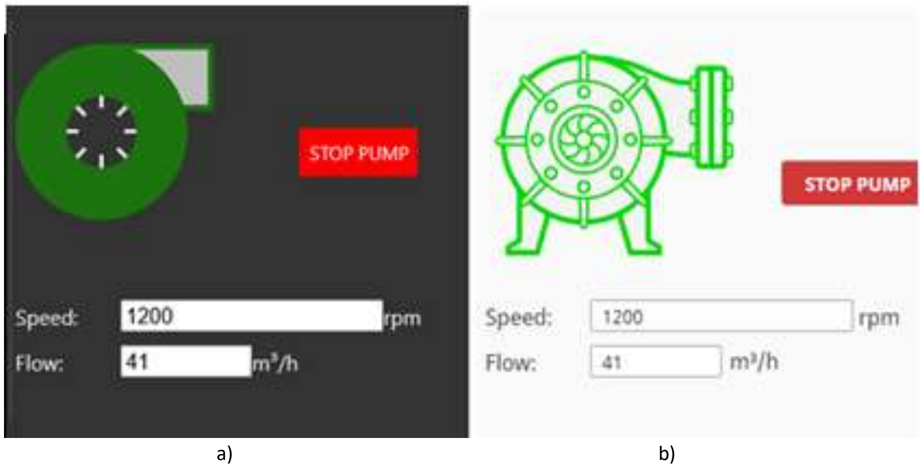


Fig. 5.3-5 Pump UI representation after item resize in: a) Node-RED dashboard, b) Ignition Perspective view

The final scenario proves bidirectional modification of the generating instance. In earlier experiments, Node-RED acted as the template generator, with changes propagated to Ignition. In this case, the instance was altered within Ignition, which functioned as the publisher, while Node-RED consumed the updated data. As shown in Fig. 5.3-6, the resized pump graphical descriptor was successfully propagated back from Ignition to Node-RED. This confirms that, regardless of the original data generator, both functional and graphical representations can be modified by any subscriber with publishing rights.

The second case study is where Ignition serves as the initial instance generator, exposing structured data to the UVS/UNS through its custom view manipulation mechanism. To maintain consistency with earlier evaluations, the first scenario verifies the correct generation of dashboards published by Ignition Perspective and consumed in Node-RED. As shown in Fig. 5.3-7, all elements were faithfully reproduced within Node-RED. Real-time process values remained directly bound to Node-RED process tags, ensuring system-wide data consistency.

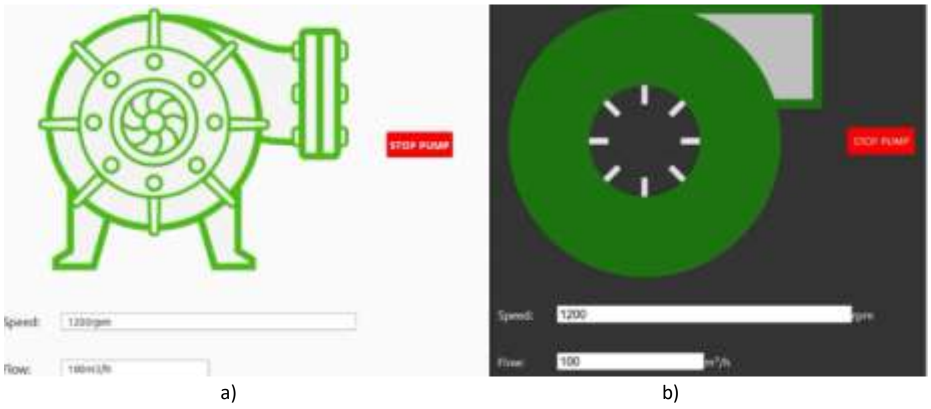
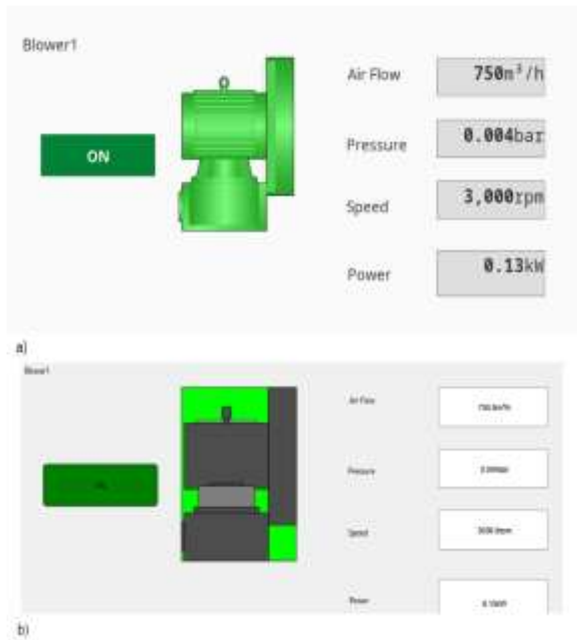


Fig. 5.3-6 Representation after pump graphical descriptor increase in size: a) Ignition – now as publisher, b) Node-RED - consumer



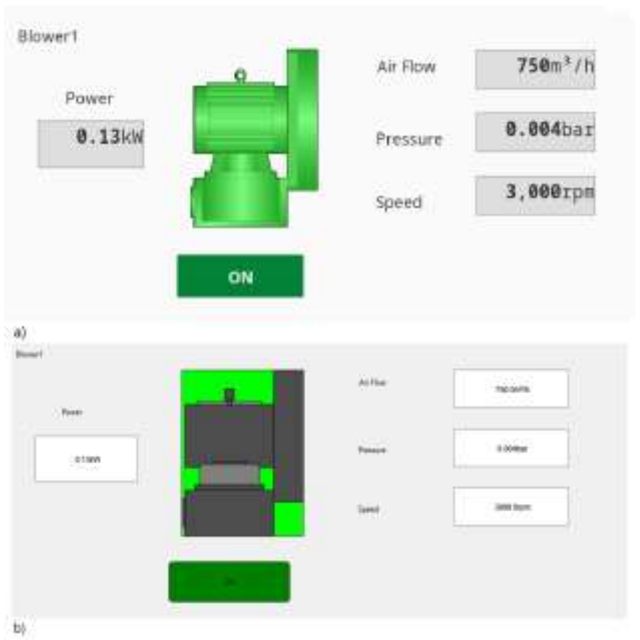


Fig. 5.3-7 UI representation in: a) Ignition Perspective view, b) Node-RED dashboard

Fig. 5.3-8 UI representation after element reposition in: a) Ignition Perspective view, b) Node-RED dashboard

To further validate the solution, structural modifications were applied within the Ignition view by repositioning two template elements, the "Power" indicator shifted from right to left, and the command button placed beneath the pump descriptor. As shown in Fig. 5.3-8, the Node-RED dashboard accurately mirrored these changes, confirming correct synchronization of structural updates between environments.

The resizing experiment began with the user interface shown in Fig. 5.3-9, where the dimensions of the right-side textboxes were modified. These changes were captured by the synchronization mechanism, serialized, and transmitted to the UNS. Node-RED successfully consumed the updated payload and accurately rendered the resized elements.

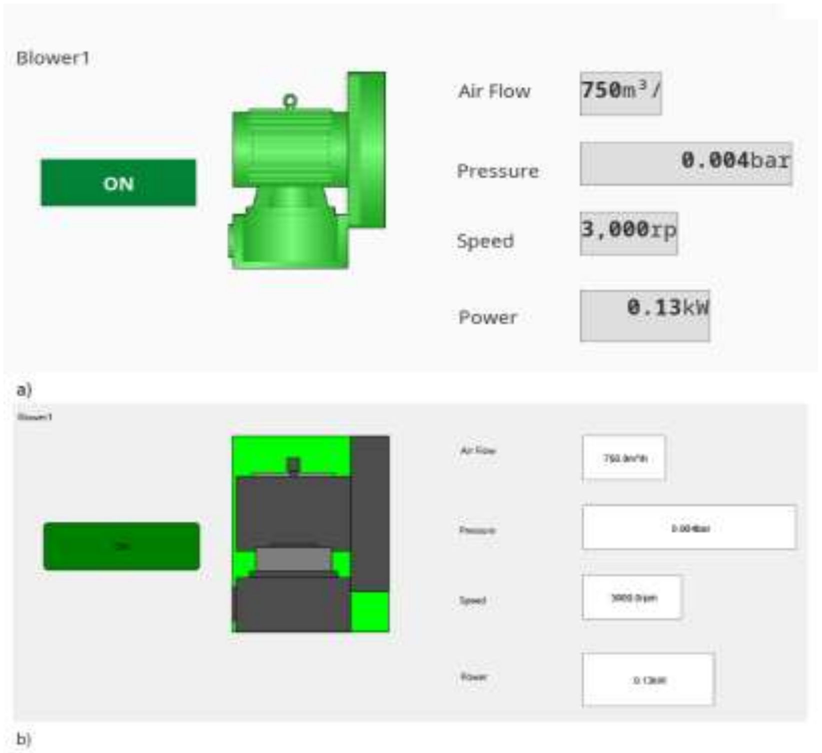
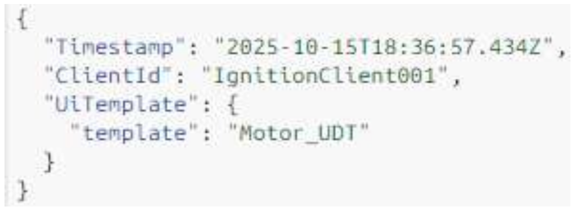


Fig. 5.3-9 UI representation after resizing in: a) Ignition Perspective view, b) Node-RED dashboard

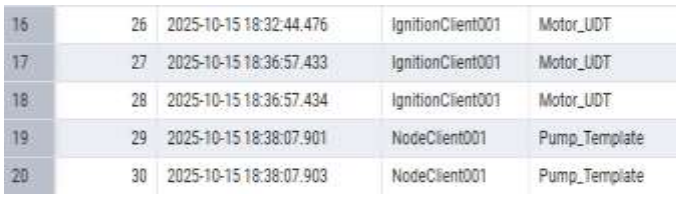
Beyond synchronization, a key requirement of the implementation is the traceability mechanism, serving as a foundation for a software-based digital passport within the UVS/UNS. In this scenario, every template modification triggers a log entry identifying the authoring platform (Node-RED or Ignition) and its timestamp. These traces are published to the UNS, with examples shown in Fig. 5.3-10. For long-term auditing, all records are persisted in a dedicated database, with PostgreSQL selected for this case.



```
{
  "Timestamp": "2025-10-15T18:36:57.434Z",
  "ClientId": "IgnitionClient001",
  "UiTemplate": {
    "template": "Motor_UDT"
  }
}
```

Fig. 5.3-10 JSON payload containing traceability data

The traceability mechanism was evaluated for the tested scenarios. Modification within instances descriptors were logged and reflected into the database. Traces are shown in Fig. 5.3-11 for auditing as database entries.



16	26	2025-10-15 18:32:44.476	IgnitionClient001	Motor_UDT
17	27	2025-10-15 18:36:57.433	IgnitionClient001	Motor_UDT
18	28	2025-10-15 18:36:57.434	IgnitionClient001	Motor_UDT
19	29	2025-10-15 18:38:07.901	NodeClient001	Pump_Template
20	30	2025-10-15 18:38:07.903	NodeClient001	Pump_Template

Fig. 5.3-11 Database table containing historical traceability data

When addressing solutions involving graphical data transmission, latency considerations must be evaluated. The proposed approach employs Sparkplug B to transmit structured, contextualized, and graphically enriched data, with time-based measurements used to assess viability. Two aspects are central: the impact of rendering and the overhead introduced by graphical descriptors in transmission. Latency measurements across all experiments showed that data transmission times were consistent in both case studies, regardless of the publisher. However, rendering durations were consistently higher in Node-RED compared to Ignition. Accordingly, results are reported in the worst-case scenario, with rendering performed in Node-RED. Figures 5.3-12 and 5.3-13 illustrate comparative cases of data transmission with and without rendering, confirming that rendering contributes approximately 5–6ms to overall latency.

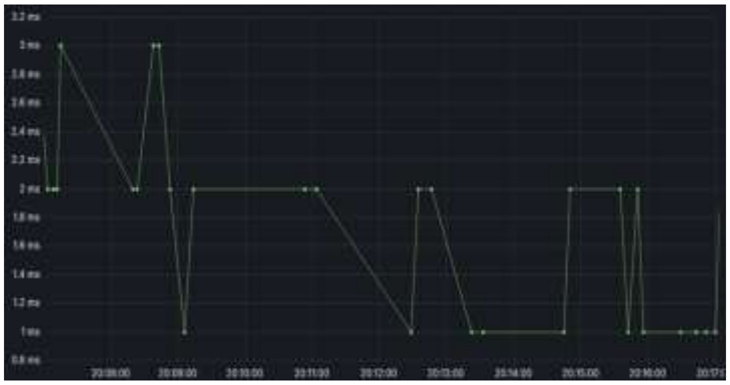


Fig. 5.3-12 UDT transmission without Node-RED rendering

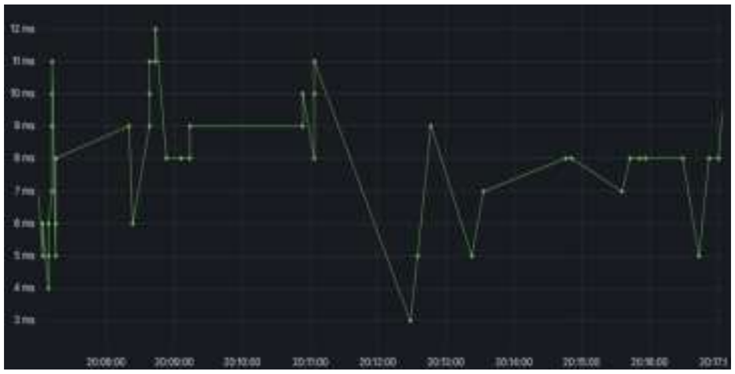


Fig. 5.3-13 UDT transmission with Node-RED rendering

Figures 5.3-14 and 5.3-15 are depicting comparatively situations where data instances are transmitted with and without graphical descriptors. It can be observed that the graphical representation weights under 1 ms.

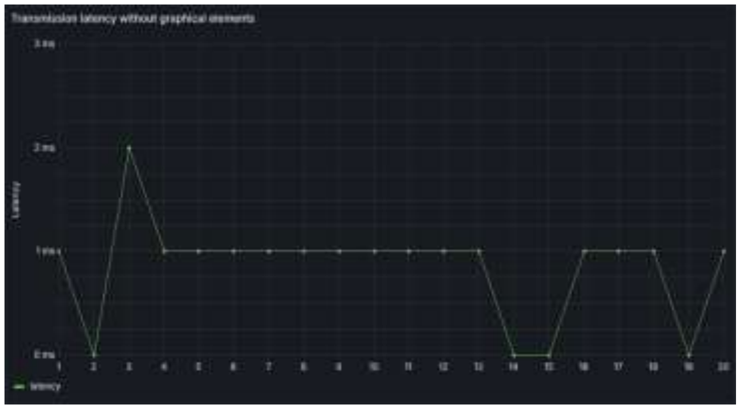


Fig. 5.3-14 Transmission latency without graphical representation

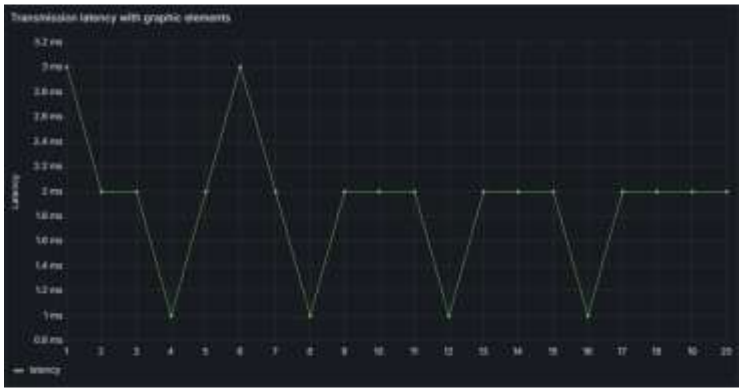


Fig. 5.3-15 Transmission latency with graphical representation

6 Scientific and academic development directions

The current chapter presents briefly the scientific and teaching plan within the Automation and Applied Informatics Department. The subsequent progression of the academic career encompasses both the didactic and the scientific research components, with the two elements being intrinsically correlated.

6.1.1 Teaching Development Plan

The development of the teaching career aims at to continuously enhance specialty competencies, as well as psycho-pedagogical and managerial skills.

The didactic activity will be underpinned by the following components:

- Continuous improvement and updating of course and laboratory support materials for the subjects that are taught.
- Diversification and continuous improvement of teaching methodologies, including interactive and collaborative methods, team-based learning, and group learning.
- Creation of new laboratory works and systems.
- Enhancement of laboratory infrastructure for the disciplines: "Automation Elements," "Industrial Internet of Things (IIoT),", "SCADA - Industrial Solutions for Data Acquisition and Supervisory Control.", etc. This will be achieved through involvement in drafting financing applications for non-reimbursable structural funds and securing sponsorships from relevant industry companies.
  Actions have already been undertaken to improve the teaching infrastructure, being an essential aspect of the taught domains. The infrastructure has to continuously improve to be able to cope with industrial evolution and new perspectives.
- Drafting of two academic books in the fields of Industrial Internet of Things and SCADA.
- Drafting a tutorial regarding conceptual and practical approach of Digital Transformation.
- Proposing new disciplines in a close connection with the industry, especially for graduate level studies, and creating corresponding course and laboratory materials.
- Continuation of existing collaborations and establishment of new partnerships with specialized companies and organizations to support didactic activities, organize meetings and workshops involving students,

and facilitate practical training (internships) and the supervision of bachelor's and master's theses.

- Supervision of Bachelor and Master Theses within the approached research domains.
- Coordinating Doctoral students.
- Guidance of master students within the research programs related to the corresponding programmes.
- Mentoring students in drafting research papers and participating in national and international scientific events.
- Participation in continuous professional development courses and knowledge enhancement programs, organized by both higher education institutions and in collaboration with specialized companies.
- Participation in teaching internships with universities abroad.

6.1.2 Research Development Plan

The research directions in the following period will be in close connection with Industry 4.0, 5.0, Digital Transformation, Artificial Intelligence, Digital Twin and IIoT domains. Contributions are foreseen on the Operational Technology level, and on the Operational Technology connection with Information Technology level. Efficiency increase related studies will be made in close correlation with achieving Industry 5.0 pillars. Studies will be made to increase the edge level processing and the corresponding security level. The approached industries will be the water sector and the manufacturing.

The scientific activity will be based on the following components:

- Dissemination of research outcomes through the publication of scientific papers in specialized journals and conference proceedings, and participation in scientific events such as international conferences and exhibitions. Specifically, I will publish a minimum of two articles per year in ISI Web of Science indexed scientific journals, preferably those with a minimum impact factor of 0.5, and a minimum of one article per year in the proceedings of international conferences indexed in recognized international databases (e.g., IEEE, Scopus, Dblp).
  Furthermore, by attending international conferences annually, I aim to disseminate research results, exchange experiences, and share information with other researchers in the field. This scientific activity will thus contribute to enhancing the prestige of the Department of

Automation and Applied Informatics, the Faculty of Automation and Computer Engineering, and the Politehnica University of Timișoara.

- Participation in the editorial boards of scientific journals and in the Scientific and Organizing Committees of international conferences.
- Realizing reviewing activities for Web-of-Science indexed international journals and for conferences.
- Participating in the evaluation activity for European research and development projects.
- Writing project proposals and participating in working groups for drafting national and international research project proposals. Scientific research will be sustained and continued by obtaining funding following the submission of project proposals in national funding competitions, as well as by participating with foreign partners in drafting and submitting project proposals in international competitions.
  Undertaking research and development projects for the industry, especially in the digital transformation, IIoT and AI domains.
- Accomplishing research objectives within work packages of research projects (e.g. HRIA – Romanian Hub for Artificial Intelligence).

The two components (teaching and researching) of subsequent career development are intrinsically correlated, as the results obtained from research activities (including studies, methods, models, and technologies), especially those in close connection with the industry, will contribute to the continuous updating of curricula, course materials, and other didactic support resources for the disciplines. One of the objective in the research and development projects is to involve graduate and undergraduate students in the activities, to be able to elevate the quality of their theses, and to provide them an approach towards understanding and elaboration of scientific works.

The development of my academic career is based on the following set of core values: openness to novelty, orientation towards applied research and learning, collaborative study, collegiality, and respect. My academic career development plan aims at raising the standards of academic and professional excellence, as well as fostering collaboration with colleagues and researchers in the field of systems engineering or related disciplines.

References

[K-1] Korodi A., Vesa V.C., Solving and Completing Structured Bidirectional Data Propagation and Representation in the Sparkplug B context, using Ignition and Node-RED, submitted to Journal, 2025

[K-2] Chisalita A.I., Korodi A. Stepping towards Zenoh Protocol in Automotive Scenarios. IEEE Access. Sept. 2025

[K-3] Korodi A., Vesa V.C., Dontu R.A., Human Centered Industry 5.0 Data Representation in the context of Technology Driven Digital Transformation Interfacing, Proc. of the 23rd IEEE Int. Conference on Industrial Informatics (INDIN), Kunming, China, 2025

[K-4] Korodi A., Vesa V.C., Dontu R.A., Efficient and Human Centered Industry 5.0 Data Propagation on the Operational Technology Level. Case Study with OPC UA Interfacing, Node-RED and Ignition., Proc. of the 21st International Conference on Automation Science and Engineering (IEEE CASE), Los-Angeles, USA, 2025

[K-5] Nitulescu I.V., Korodi A., Curiac D.I., Contribution Study for Digital Transformation in the Context of Legacy IT Level Asset Management Systems, Proc. of the 25th Int. Conf. on Control Systems and Computer Science (CSCS), Bucharest, Romania, 2025

[K-6] Dontu R.A, Korodi A., Integrated IoT Solution Targeting Cloud-based Infrastructures for Digital Transformation using MQTT over JSON Protocol, Proc. of the 25th Int. Conf. on Control Systems and Computer Science (CSCS), Bucharest, Romania, 2025

[K-7] A. Korodi, A. Nicolae, D. Brisc, I. Drăghici and A. Corui, "Long Short-Term Memory-Based Prediction Solution Inside a Decentralized Proactive Historian for Water Industry 4.0," IEEE Access, vol. 12, pp. 99526-99536, 2024, doi: 10.1109/ACCESS.2024.3428866.

[K-8] Korodi, A.; Nițulescu, I.-V.; Fülöp, A.-A.; Vesa, V.-C.; Demian, P.; Braneci, R.-A.; Popescu, D. Integration of Legacy Industrial Equipment in a Building-Management System Industry 5.0 Scenario. Electronics 2024, 13, 3229. https://doi.org/10.3390/electronics13163229

[K-9] Ungureanu, V.-I.; Negirla, P.; Korodi, A. Image-Compression Techniques: Classical and "Region-of-Interest-Based" Approaches Presented in Recent Papers. Sensors 2024, 24, 791. https://doi.org/10.3390/s24030791

[K-10] Mateoiu, A.-M.; Korodi, A.; Stoianovici, A.; Tira, R. Supervisory Monitoring and Control Solution on Android Mobile Devices for the Water Industry 4.0. Sustainability 2023, 15, 16022. https://doi.org/10.3390/su152216022

[K-11] Tidea, A; Korodi, A.; ECC Implementation and Performance Evaluation for Securing OPC UA Communication, Proc. of the 22nd IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom-2023), pp. 1712-1719, Exeter, UK, 2023

[K-12] Korodi, A.; Nicolae, A.; Drăghici, I.A. Proactive Decentralized Historian-Improving Legacy System in the Water Industry 4.0 Context. Sustainability 2023, 15, 11487. https://doi.org/10.3390/su151511487

[K-13] Tidrea, A.; Korodi, A.; Silea, I. Elliptic Curve Cryptography Considerations for Securing Automation and SCADA Systems. Sensors 2023, 23, 2686. https://doi.org/10.3390/s23052686

[K-14] A. Nicolae, C. Burlacu, A. Stanciu, A. Korodi, B. Poşa and F. Mihaila, "An Industry 4.0 Oriented Predictive Maintenance Solution Deployed in Real-World Automotive Manufacturing Facilities," 2023 24th International Conference on Control Systems and Computer Science (CSCS), Bucharest, Romania, 2023, pp. 99-104, doi: 10.1109/CSCS59211.2023.00025.

[K-15] Ioana, A.; Korodi, A.; Silea, I. Automotive IoT Ethernet-Based Communication Technologies Applied in a V2X Context via a Multi-Protocol Gateway. Sensors 2022, 22, 6382. https://doi.org/10.3390/s22176382

[K-16] Ioana A, Korodi A. DDS and OPC UA Protocol Coexistence Solution in Real-Time and Industry 4.0 Context Using Non-Ideal Infrastructure. Sensors 2021, 21, 7760.

[K-17] Nicolae A, Korodi A, Silea I. Complete Automation of an Energy Consumption Reduction Strategy from a Water Treatment and Distribution Facility, Inside an Industrial Internet of Things-Compliant Proactive Historian Application. Sensors. 2021; 21(7):2569.

[K-18] Ioana A, Burlacu C, Korodi A. Approaching OPC UA Publish–Subscribe in the Context of UDP-Based Multi-Channel Communication and Image Transmission. Sensors. 2021; 21(4):1296.

[K-19] Ioana A, Korodi A. Improving OPC UA Publish-Subscribe Mechanism over UDP with Synchronization Algorithm and Multithreading Broker Application. Sensors. 2020; 20(19):5591.

[K-20] Korodi A, Anitei D, Boitor A, Silea I. Image-Processing-Based Low-Cost Fault Detection Solution for End-of-Line ECUs in Automotive Manufacturing. Sensors. 2020; 20(12):3520.

[K-21] Ioana A, Korodi A. OPC UA Publish-Subscribe and VSOME/IP Notify-Subscribe Based Gateway Application in the Context of Car to Infrastructure Communication. Sensors. 2020; 20(16):4624.

[K-22] Nițulescu I-V, Korodi A. Supervisory Control and Data Acquisition Approach in Node-RED: Application and Discussions. IoT. 2020; 1(1): 76-91.

[K-23] Korodi A., Crisan R., Nicolae A., Silea I. ”Industrial Internet of Things and Fog Computing to Reduce Energy Consumption in Drinking Water Facilities”. Processes 2020, 8, 282.

[K-24] Nicolae, A.; Korodi A.; Silea, I. Weather-Based Prediction Strategy inside the Proactive Historian with Application in Wastewater Treatment Plants. Appl. Sci. 2020, 10, 3015.

[K-25] Ungureanu, V.-I.; Miclea, R.-C.; Korodi, A.; Silea, I. A Novel Approach against Sun Glare to Enhance Driver Safety. Appl. Sci. 2020, 10, 3032.

[K-26] Nicolae A., Korodi A., Silea I., „An Overview of Industry 4.0 Development Directions in the Industrial Internet of Things Context”, Romanian Journal of Information Science and Technology (ROMJIST), vol. 22, issue 3-4, pp. 183–201, 2019

[K-27] Tidrea A., Korodi A., Silea, „Cryptographic Considerations for Automation and SCADA Systems using Trusted Platform Modules”, Sensors, vol. 19, issue 19, 2019

[K-28] Nicolae A., Korodi A., Silea I., „Identifying Data Dependencies as First Step to Obtain a Proactive Historian: Test Scenario in the Water Industry 4.0”, Water, 11/1144, 2019.

[K-29] Petre C.A., Korodi A., "Honeypot Inside an OPC UA Wrapper for Water Pumping Stations", Proceedings of the 22nd International Conference on Control Systems and Computer Science (CSCS), Bucharest, Romania, 28-30 May 2019

[K-30] Ioana A., Korodi A., "VSOMEIP - OPC UA Gateway Solution for the Automotive Industry", Proc. of the International Conference on Engineering, Technology and Innovation, Sophia Antipolis, France, 17-19 June, 2019

[K-31] Toc S.I., Korodi A., ”Modbus-OPC UA Wrapper using Node-RED and IoT-2040 with application in the water industry”, Proc. of the 16th IEEE International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, September 13-15, 2018

[K-32] Korodi A., Radu M.A., Crisan R., ”Non-Invasive Control Solution inside Higher-Level OPC UA based Wrapper for Optimizing Groups of Wastewater Systems”, Proceedings of the IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Torino, Italy, September 4-7, 2018.

[K-33] Nicolae A., Korodi A., „Node-Red and OPC UA Based Lightweight and Low-Cost Historian with Application in the Water Industry", Proc. of the 16th IEEE International Conference on Industrial Informatics (INDIN), Porto, Portugal, July 18-20, 2018.

[K-34] Tidrea A., Korodi A., „WebNavIGSS Web-Based Software Solution for IGSS SCADA Applications", Proceedings of the 26th Mediterranean Conference on Control and Automation (MED), Zadar, Croatia, June 19-22, 2018

[K-35] Mateoiu A., Korodi A., OPC-UA based small-scale monitoring and control solution for Android devices. Case study for water treatment plants.", Proceedings of the 4th International Conference on Control, Automation and Robotics (ICCAR 2018), Auckland, New Zealand, April 20-23, 2018

[K-36] Crisan R., Korodi A., „Noninvasive Control Solution for Energy Efficiency in Wastewater Treatment Plants", Proceedings of the 19th International Conference on Industrial Technology (ICIT), Lyon, France, pp. 1604-1609, February 20-22, 2018

[K-37] Nicolae A., Korodi A., Silea I., „Modular and model-driven configurable approach for a centralized home-security system", Proceedings IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI 2018), Timisoara, Romania, May 17-19, 2018

[K-38] Korodi A., Silea I., "Achieving Interoperability Using Low-Cost Middleware OPC UA Wrapping Structure. Case Study in the Water Industry", Proc. of the 15th IEEE International Conference on Industrial Informatics (INDIN), Emden, Germany, pp. 1223-1228, 24-26 July, 2017

[K-39] Korodi A., Huple T., Silea I., Stefan O., "IGSS Higher-Level SCADA Optimal Resource Allocation to Integrate Water and Waste Water Pumping Stations", Proc. of the 21th International Conference on Control Systems and Computer Science, pp. 93-98, Bucharest, Romania, 29-31 May 2017

[K-40] Korodi A., Silea I., „Specifying and Tendering of Automation and SCADA Systems: Case Study for Waste Water Treatment Plants", Proc. of the IEEE Conference on Control Applications (CCA), part of the IEEE Multi-Conference on Systems and Control (MSC 2014), Antibes, France, October, 2014

[K-41] Korodi A., „Building a Knowledge Base to Obtain the Maximum Power Point for a PV panel", Proceedings of the IEEE Conference on Control Applications (CCA), part of the IEEE Multi-Conference on Systems and Control (MSC 2012), ISBN 978-1-4673-4504-0, pp. 1098-1103, Dubrovnik, Croatia, October, 2012

[K-42] Dragomir T. L., Petcuţ F. M., Korodi A., "Reference Value Generator of Maximum Power Point Coordinates of the Photovoltaic Panel External Characteristic", New Concepts and Applications in Soft Computing - Studies in Computational Intelligence, vol. 417, pp. 71-96, Springer, 2012

[K-43] Korodi A., Petcut F.M., Dragomir T.L., „Interpolative Based Implementation of a Photovoltaic Panel", Proceedings of the 19th Mediterranean Conference on Control and Automation, ISBN 978-1-4577-0124-5, pp. 345-350, Corfu, Greece, June 20-23, 2011

[K-44] Stanciu M.C., Korodi, A., „The Social Marginalization of the Elderly - Probabilistic Analysis through Markov Models", Procedia - Social and Behavioral Sciences, Elsevier, vol. 46, pp. 504-508, 2012

[K-45] Korodi A., Codrean A., Timofte A., Dragomir T.L., „Cardiovascular Model with Human Elastance Function and Valve Dynamics", Proceedings of the 20th Mediterranean Conference on Control and Automation, MED 2012, ISBN 978-1-4673-2529-5, pp. 1421-1427, Barcelona, Spain, July, 2012

[K-46] Codrean A., Korodi A., Dragomir T.L., Kovacs L., "Up to Date Issues on Modeling the Nervous Control of the Cardiovascular System on Short-Term", Proceedings of the

18th IFAC World Congress, ISBN 978-3-902661-93-7, pp. 3747-3752, Milan, Italy, 28 Aug. - 2 Sept., 2011

[K-47] Codrean A., Korodi A., Jiveț I., T.L. Dragomir, "Developing a Lumped Model for the Vestibular Receptors", ISBN 978-3-642-22586-4, Proceedings of the Advancements of Medicine and Health Care through Technology, pp. 260-265, Cluj-Napoca, Romania, 29 Aug. – 2 Sept., 2011

[K-48] Codrean, A., Korodi A., Dragomir, T.L., Ceregan V., "Modeling the Vestibular Nucleus", Lecture Notes in Electrical Engineering - Journal of Intelligent Control and Computer Engineering, Springer, vol. 70, pp. 293-306, 2011

[K-49] Codrean A., Ceregan V., Dragomir, T.L., Korodi A., "Interpolative frequency characteristics generators for the Vestibular Nucleus Activity", Proc. of the IAENG Int. Conf. on Bioinformatics, Hong Kong, vol. 1, pp. 195-199, ISBN 978-988-17012-8-2, ISSN 2078-0958 (print), 18-20 March, 2010

[K-50] Korodi A., Ceregan V., Dragomir, T.L., Codrean A. "A Continuous-Time Dynamical Model for the Vestibular Nucleus", Proc. of the 12th Mediterranean Conference on Medical and Biological Engineering and Computing - IFMBE MEDICON 2010, pp. 627-630, Publisher Springer Berlin Heidelberg, ISSN1680-0737 (Print), 1433-9277 (Online), Chalkidiki, Greece, 27-30 May, 2010

[K-51] Ceregan V., Korodi A., Dragomir, T.L., Codrean A. "An Adaptive Interpolative Based Model for the Vestibular Nucleus", Proc. of the IEEE Int. Joint Conf. on Computational Cybernetics and Technical Informatics – ICCC-CONTI 2010, pp. 31-36, Timisoara, Romania, 27-29 May, 2010

[K-52] Korol T., Korodi A., "An Evaluation of Effectiveness of Fuzzy Logic Model in Predicting the Business Bankruptcy", Romanian Journal of Economic Forecasting, vol. 14, issue 3, pp. 92-107, 2011

[K-53] Korol, T., Korodi A., "Predicting bankruptcy with the use of macroeconomic variables", Journal of Economic Computation and Economic Cybernetics Studies and Research, vol. 44/1, pp. 201-220, 2010

[K-54] Korodi A., Corman I. M., „Wheeled Mobile Robot Model and Cooperative Formation Control" WSEAS Transactions on Systems - Special Issue on Collaborative Systems, vol. 11, 2012

[K-55] Korodi A., Dragomir, T.L., "Correcting Odometry Errors for Mobile Robots Using Image Processing", Proceedings of the IAENG International Conference on Control and Automation, Hong Kong, vol. 2, pp. 1040-1045, 18-20 March, 2010 – Awarded with the Certificate of Merit

[K-56] Korodi A., Dragomir T.L., „Availability Studies and Solutions for Wheeled Mobile Robots", Lecture Notes in Electrical Engineering - Intelligent Control and Computer Engineering, Springer, vol. 70, pp. 47-58, 2011

[K-57] Korodi A., Codrean A., Banita L., „Aspects Regarding the Object Following Control Procedure for Wheeled Mobile Robots", WSEAS Trans. on Systems and Control, Issue 6, vol. 3, pp. 537-546, ISSN 1991-8763, June, 2008

[K-58] Korodi A., Codrean A., Banita L., Butaru A., Carnaru R., „Object Following Control for Wheeled Mobile Robots", Proc. of the 9th WSEAS Int. Conf. on Automation and Technical Information ICAI, pp. 338-343, ISBN 978-960-6766-77-0, ISSN 1790-5117, Bucharest, Romania, June 24-26, 2008

[K-59] Korodi, A., Huple T., Automation Elements – Applications 1, Programming Panasonic PLC and HMI – IGSS and Ignition SCADA Development, Edit. Politehnica, 206 pages, 978-606-554-996-8, Timisoara, 2015

[K-60] Korodi, A., Robu, R., Pintea, R., Computer Programming – Applications, Edit. Politehnica, 133 pages, ISBN 978-973-625-649-3, Timisoara, 2008

[1] E. Moraes, H. Lepikson, S. Konstantinov, "Improving connectivity for runtime simulation of automation systems via OPC UA", Proc. of the 13th IEEE International Conf. on Industrial Informatics (INDIN), 2015

[2] S. Wredea, O. Beyerb, C. Dreyera, "Vertical Integration and Service Orchestration for Modular Production Systems using Business Process Models", Procedia Technology 26 - The 3rd Int. Conf. on System-integrated Intelligence, SysInt, pp. 259-266, 2016.

[3] C. Bergera, A. Heesa, S. Braunreuthera, G. Reinhart, "Characterization of Cyber-Physical Sensor Systems", Proc. CIRP 41, pp. 638 – 643, 2016

[4] I. Seilonen, T. Tuovinen, et.al., "Aggregating OPC UA servers for monitoring manufacturing systems and mobile work machines", Proc. of IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), 2016

[5] M.V. García, E. Irisarri, et.al., "Plant floor communications integration using a low cost CPPS architecture", Proc. of the IEEE 21st Int. Conf. on Emerging Technologies and Factory Automation (ETFA), 2016

[6] V.H. Nguyen, Q.T. Tran, Y. Besanger, "SCADA as a service approach for interoperability of micro-grid platforms", Sustainable energy grids & networks, vol. 8, pp. 26-36, 2016

[7] P.A. Nyen, E. Polanscak, O. Roulet-Dubonnet, et.al., "Distributed, autonomous control in production of jet turbine parts", Proc. of the 6th CIRP Conf. on Learning Factories, vol. 54, pp. 191-196, Norway, 2016

[8] N.C. Gaitan, "MCIP Client Application for SCADA in Iiot Environment", International Journal of Advanced Computer Science and Applications, 6/ 9, pp. 158-163, 2015

[9] P. Drahos, E. Kucera, O. Haffner, I. Klimo, "Trends in industrial communication and OPC UA", Proceedings of the International Conference on Cybernetics & Informatics (K&I), pp. 1-5, Lazy pod Makytou, Slovakia, 28 Ian.-03 Feb., 2018

[10] M. Graube, S. Hensel, C. Iatrou, L. Urbas , "Information models in OPC UA and their advantages and disadvantages", Proceedings of the 22nd IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 12-15 Sept. 2017

[11] H. Haskamp, M. Meyer, R. Mollmann, F. Orth, A. Colombo, "Benchmarking of existing OPC UA implementations for Industrie 4.0-compliant digitalization solutions", Proc. of the 15th IEEE Intern. Conf. on Ind. Info. (INDIN), Emden, Germany, pp. 589-594, 24-26 July, 2017

[12] S. Cavalieri, A. Regalbuto, "Integration of IEC 61850 SCL and OPC UA to improve interoperability in Smart Grid environment", Computer Standards & Interfaces, vol. 47, pp. 77-99, Aug. 2016

[13] M. Muller, E. Wings, L. Bergmann, "Developing Open Source Cyber-Physical Systems for Service-Oriented Architectures Using OPC UA", Proc. of the 15th IEEE Intern. Conf. on Ind. Info. (INDIN), Emden, Germany, pp. 83-88, 24-26 July, 2017

[14] A. Veichtlbauer, M. Ortmayer, T. Heistracher, "OPC UA Integration for Field Devices", Proc. of the 15th IEEE Intern. Conf. on Ind. Info. (INDIN), Emden, Germany, pp. 419-424, 24-26 July, 2017

[15] H. Derhamy, J. Ronnholm, J. Delsing, J. Eliasson, J. van Deventer, "Protocol interoperability of OPC UA in Service Oriented Architectures", Proc. of the 15th IEEE Intern. Conf. on Ind. Info. (INDIN), Emden, Germany, pp. 44-50, 24-26 July, 2017

[16] A. Ismail, W. Kastner, "Throttled service calls in OPC UA", Proceedings of the IEEE Int. Conf. on Industrial Technology (ICIT), pp. 1658 - 1663, Lyon France, Feb. 2018

[17] S. Cavalieri, D. Di Stefano, M.G. Salafia, M.S. Scroppo, "A web-based platform for OPC UA integration in IIoT environment", Proceedings of the 22nd IEEE Int. Conf. on Emerging Tech. and Factory Automation (ETFA), Limassol, Cyprus, 12-15 Sept. 2017

[18] M. Zarte, A. Pechmann, J. Wermann, F. Gosewehr, A.W. Colombo, "Building an Industry 4.0-compliant lab environment to demonstrate connectivity between shop floor and IT levels of an enterprise", Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON 2016), Florence, Italy, 23-26 Oct. 2016

[19] A. Balador, N. Ericsson, Z. Bakhshi, "Communication middleware technologies for industrial distributed control systems: A literature review", Proceedings of the 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 12-15 Sept. 2017

[20] F. S. Costa et al., "FASTEN IIoT: An Open Real-Time Platform for Vertical, Horizontal and End-To-End Integration," Sen-sors, vol. 20, no. 19, p. 5499, Sep. 2020, doi: 10.3390/s20195499.

[21] D. Baudouin, et.al.. (2021). The challenges, approaches, and used techniques of CPS for manufacturing in Industry 4.0: a literature review. Int. J. of Adv. M.. Tec. 1-18. 10.1007/s00170-020-06572-4.

[22] Shakib, Kazi & Neha, Farhin. (2021). A Study for taking an approach in Industrial IoT based Solution. Journal of Physics: Conference Series. 1831. 012007. 10.1088/1742-6596/1831/1/012007.

[23] D. Simić, N. Saulic. "Logistics Industry 4.0: Challenges and Opportunities". Business, Computer Science, Engineering (2019).

[24] Orellana, Felipe & Torres, Romina. (2019). From legacy-based factories to smart factories level 2 according to the industry 4.0. International Journal of Computer Integrated Manufacturing. 32. 1-11. 10.1080/0951192X.2019.1609702.

[25] Ferrari, P. et.al. (2017). "Evaluation of communication latency in industrial IoT applications." IEEE Int. Workshop on Meas-urements and Networking Proceedings (M&N), 1-6. 10.1109/IWMN.2017.8078359.

[26] Govindarajan, Naveen & Ramis, Borja & Xu, Xiangbin & Nieto, Norma & Martinez Lastra, Jose Luis. (2016). An approach for integrating legacy systems in the manufacturing industry. 683-688. 10.1109/INDIN.2016.7819247.

[27] Gogolev, A.; Mendoza, F.; Braun, R. TSN-Enabled OPC UA in Field Devices. IEEE 23rd Int. Conf. on Emerging Technologies and Factory Automation (ETFA), Torino, Italy, 4–7 September 2018; pp. 297–303.

[28] Gogolev, A.; Braun, R.; Bauer, P. TSN Traffic Shaping for OPC UA Field Devices. 2019 IEEE 17th Int. Conf. on Industrial Informatics (INDIN), Helsinki, Finland, 2019, pp. 951-956, doi: 10.1109/INDIN41052.2019.8972252.

[29] Haskamp, H.; Orth, F.; Wermann, J.; Colombo A.W. Implementing an OPC UA interface for legacy PLC-based automation systems using the Azure cloud: An ICPS-architecture with a retrofitted RFID system. 2018 IEEE Industrial Cyber-Physical Systems (ICPS), St. Petersburg, 2018, pp. 115-121, doi: 10.1109/ICPHYS.2018.8387646.

[30] OPC. 10000-14-UA Specification Part 14 PubSub; OPC Foundation: Scottsdale, AR, USA, 1 April 2018

[31] Eckhardt, A.; Müller, S.; Leurs, L. An evaluation of the applicability of OPC UA Publish Subscribe on factory automation use cases. In Proceedings of the IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Torino, Italy, 4–7 September 2018; pp. 1071–1074.

[32] Pfrommer, J.; Ebner, A.; Ravikumar, S.; Karunakaran, B. Open Source OPC UA PubSub Over TSN for Realtime Industrial Communication. In Proceedings of the IEEE 23rd

International Conference on Emerging Technologies and Factory Automation (ETFA), Torino, Italy, 4–7 September 2018; pp. 1087–1090, doi:10.1109/ETFA.2018.8502479.

[33] Available online: https://github.com/open62541/open62541.

[34] Peniak P.; Bubenikova E.; Spalek J. Model of Integration Gateway for Communication of OPC/MQTT Devices. 2020 Cybernetics & Informatics (K&I), Velke Karlovice, Czech Republic, 2020, pp. 1-5, doi: 10.1109/KI48306.2020.9039852.

[35] Available online: https://www.cirrus-link.com/mqtt-sparkplug-tahu (accessed on 2nd of September 2020)

[36] AUTOSAR. SOME/IP Protocol Specification; AUTOSAR: Munich, Germany, 2016

[37] Vidal, I.; Bellavista, P.; Sanchez-Aguero, V.; Garcia-Reinoso, J.; Valera, F.; Nogales, B.; Azcorra, A. Enabling Multi-Mission Interoperable UAS Using Data-Centric Communications. Sensors 2018, 18, 3421.

[38] Youssef, T.A.; Esfahani, M.M.; Mohammed, O. Data-Centric Communication Framework for Multicast IEC 61850 Routable GOOSE Messages over the WAN in Modern Power Systems. Appl. Sci. 2020, 10, 848.

[39] L. Li, H. Zhu, et.al., TSN-Based Scheduling of Task-Network Co-Scheduling on In-Vehicle Multi-Core System, Proc. of the 23rd IEEE International Conference on Industrial Informatics (INDIN), Kunming, China, 2025

[40] Tarkoma, S. Publish/Subscribe Systems: Design and Principles, Wiley 1st Edition, 2012.

[41] Newman, W.S. A Systematic Approach to Learning Robot Programming with ROS. Chapman and Hall/CRC, 2017.

[42] AUTOSAR. Requirements on Time Synchronization; AUTOSAR: Munich, Germany, 2019.

[43] Tang, S.; Zhu, Y.; Yuan, S.; Li, G. Intelligent Diagnosis towards Hydraulic Axial Piston Pump Using a Novel Integrated CNN Model. Sensors 2020, 20, 7152.

[44] Na, K.-M.; Lee, K.; Shin, S.-K.; Kim, H. Detecting Deformation on Pantograph Contact Strip of Railway Vehicle on Image Processing and Deep Learning. Applied Sciences 2020, 10, 8509.

[45] Stark, E.; Kučera, E.; Haffner, O.; Drahoš, P.; Leskovský, R. Using Augmented Reality and Internet of Things for Control and Monitoring of Mechatronic Devices. Electronics 2020, 9, 1272.

[46] Yahiaoui, L.; Horgan, J.; Deegan, B.; Yogamani, S.; Hughes, C.; Denny, P. Overview and Empirical Analysis of ISP Parameter Tuning for Visual Perception in Autonomous Driving. J. Imaging 2019, 5, 78.

[47] Kang, S.; Chun, C.; Shim, S.;Ryu, S.; Baek, J. Real Time Image Processing System for Detecting Infrastructure Damage: Crack. 2019 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2019, pp. 1-3, doi: 10.1109/ICCE.2019.8661830.

[48] Tian, D.; Zhang, C.; Duan, X.; Wang, X. An Automatic Car Accident Detection Method Based on Cooperative Vehicle Infrastructure Systems. IEEE Access, vol. 7, pp. 127453-127463, 2019.

[49] Ahn, S.; Choi, J. Utilization of V2X Communications for Vehicle Queue Length Estimation. 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, 2018, pp. 645-648, doi: 10.1109/ICTC.2018.8539574.

[50] Mathias, S. G.; Schmied, S.; Grossmann, D.; Müller, R. K.; Mroß, B. A Compliance Testing Structure for Implementation of Industry Standards through OPC UA. 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 2020, pp. 1091-1094.

[51] Eckhardt A.; Müller S. Analysis of the Round Trip Time of OPC UA and TSN based Peer-to-Peer Communication. 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 2019, pp. 161-167.

[52] Cenedese A.; Frodella M.; Tramarin F.; Vitturi A. Comparative assessment of different OPC UA open–source stacks for embedded systems. 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 2019, pp. 1127-1134.

[53] Initiative: Field Level Communications (FLC) OPC Foundation Extends OPC UA Including TSN Down To Field Level. OPC Foundation Tech. Rep., February 2019.

[54] "OPC UA for Programmable Logic Controllers based on IEC61131-3", OPC Foundation Tech. Rep., March 2010.

[55] Panda S. K.; Majumder M.; Wisniewski L.; Jasperneite J. Real-time Industrial Communication by using OPC UA Field Level Communication. 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 2020, pp. 1143-1146

[56] Li Y.; Jiang J.; Lee C.; Hong S. H. Practical Implementation of an OPC UA TSN Communication Architecture for a Manufacturing System. IEEE Access, vol. 8, pp. 200100-200111, 2020.

[57] Iatrou C. P.; Ketzel L.; Graube M.; et.al. Design classification of aggregating systems in intelligent information system architectures. 2020 25th IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 2020, pp. 745-752.

[58] Redmon J.; Divvala S.; Girshick R.; Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788.

[59] Available online: https://raw.githubusercontent.com/eProsima/Fast-DDS/master/fastrtps.repos.

[60] Rodríguez-Molina, J.; Bilbao, S.; Martínez, B.; Frasheri, M.; Cürüklü, B. An Optimized, Data Distribution Service-Based Solution for Reliable Data Exchange Among Autonomous Underwater Vehicles. Sensors 2017, 17, 1802.

[61] Kumar, M.; Singh, A.K. FDDS: An Integrated Conceptual FDDS Framework for DDS Based Middleware. In Proceedings of the 2019 Int. Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 17–19 July 2019; pp. 1952–1956.

[62] Barciś, M.; Barciś, A.; Hellwagner, H. Information Distribution in Multi-Robot Systems: Utility-Based Evaluation Model. Sen-sors 2020, 20, 710.

[63] Thulasiraman, P.; Chen, Z.; Allen, B.; Bingham, B. Evaluation of the Robot Operating System 2 in Lossy Unmanned Net-works. In Proceedings of the 2020 IEEE International Systems Conference (SysCon), Online, 24–27 August 2020; pp. 1–8.

[64] Fernandez, J.; Allen, B.; Thulasiraman, P.; Bingham, B. Performance Study of the Robot Operating System 2 with QoS and Cyber Security Settings. In Proceedings of the 2020 IEEE International Systems Conference (SysCon), Online, 24–27 August 2020; pp. 1–6.

[65] Coronado, E.; Venture, G. Towards IoT-Aided Human–Robot Interaction Using NEP and ROS: A Platform-Independent, Accessible and Distributed Approach. Sensors 2020.

[66] Morita, R.; Matsubara, K. Dynamic Binding a Proper DDS Implementation for Optimizing Inter-Node Communication in ROS2. In Proceedings of the 2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Hakodate, Japan, 28–31 August 2018; pp. 246–247.

[67] Profanter, S.; Tekat, A.; Dorofeev, K.; Rickert, M.; Knoll, A. OPC UA versus ROS, DDS, and MQTT: Performance Evaluation of Industry 4.0 Protocols. In Proceedings of the 2019

IEEE International Conference on Industrial Technology (ICIT), Mel-bourne, VIC, Australia, 13–15 February 2019, pp. 955–962.

[68] Sim, W.; Song, B.; Shin, J.; Kim, T. Data Distribution Service Converter Based on the Open Platform Communications Unified Architecture Publish–Subscribe Protocol. Electronics 2021, 10, 2524, https://doi.org/10.3390/electronics10202524.

[69] Sung K.; Lee, J.; Shin, J. Study of CAN-to-3GPP LTE gateway architecture for automotive safety in V21 environment. 2015 IEEE International Conference on Advanced Communications Technology.

[70] H. Li, L. Chen, W. Chang, J. Tang and K. S. Li. Design and development of an extensible multi-protocol automotive gateway. 2016 IEEE Int. Conf. on Consumer Electronics-Taiwan (ICCE-TW), Nantou, 2016, pp. 1-2, doi: 10.1109/ICCE-TW.2016.7521011.

[71] Inkoom, S.; Sobanjo, J.; Chicken, E. Competing Risks Models for the Assessment of Intelligent Transportation Systems Devices: A Case Study for Connected and Autonomous Vehicle Applications. Infrastructures 2020, 5, 30.

[72] Chen, J.; Xue, Z.; Fan, D. Deep Reinforcement Learning Based Left-Turn Connected and Automated Vehicle Control at Signalized Intersection in Vehicle-to-Infrastructure Environment. Information 2020, 11, 77.

[73] Swamy N., "Evaluation of OPC-UA Technology in a Car-2x Communication towards an Industry 4.0 Driven Automotive Domain", Master Thesis, Dept. of Computer Science, Technische Universitat Chemnitz, 2017

[74] R. Cupek, A. Ziebinski and M. Drewniak. An OPC UA server as a gateway that shares CAN network data and engineering knowledge. 2017 IEEE International Conference on Industrial Technology (ICIT), Toronto, ON, 2017, pp. 1424-1429, doi: 10.1109/ICIT.2017.7915574.

[75] Neumann A., Mytych M.J., Wesemann D., Wisniewski L., Jasperneite J. Approaches for In-vehicle Communication – An Analysis and Outlook. Communications in Computer and Information Science, vol 718. Springer, 2017

[76] S. Shreejith et al. VEGa: A High Performance Vehicular Ethernet Gateway on Hybrid FPGA. IEEE Transactions on Computers, vol. 66, no. 10, pp. 1790-1803, 1 Oct. 2017, doi: 10.1109/TC.2017.2700277.

[77] L. L. Bello, R. Mariani, S. Mubeen and S. Saponara. Recent Advances and Trends in On-Board Embedded and Networked Automotive Systems. IEEE Transactions on Industrial Informatics, vol. 15, no. 2, pp. 1038-1051, Feb. 2019, doi: 10.1109/TII.2018.2879544.

[78] M. Iorio, M. Reineri, et.al., "Securing SOME/IP for In-Vehicle Service Protection," IEEE T. on Vehicular Tech., 69/11, pp. 13450-13466, 2020.

[79] Ma, B.; Yang, S.; Zuo, Z.; Zou, B.; Cao, Y.; Yan, X.; Zhou, S.; Li, J. An Authentication and Secure Communication Scheme for In-Vehicle Networks Based on SOME/IP. Sensors 2022, 22, 647.

[80] D. -S. Cho, S. Yun, et.al., "Autonomous Driving System Verification Framework with FMI Co-Simulation based on OMG DDS," 2020 IEEE Int. C. on Consumer Electronics (ICCE), 2020, pp. 1-6

[81] T. Bijlsma et al., "A Distributed Safety Mechanism using Middleware and Hypervisors for Autonomous Vehicles," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2020, pp. 1175-1180

[82] H. Zhu, W. Zhou, Z. Li, L. Li, and T. Huang, "Requirements-Driven Automotive Electrical/Electronic Architecture: A Survey and Prospective Trends", IEEE Access, vol. 9, pp. 100096–100112, 2021, doi: 10.1109/ACCESS.2021.3093077.

[83] Z. Liu, W. Zhang, and F. Zhao, "Impact, Challenges and Prospect of Software-Defined Vehicles", Automot. Innov., vol. 5, no. 2, pp. 180–194, Apr. 2022, doi: 10.1007/s42154-022-00179-z.
[84] "Zone Control Units", Continental Automotive. Accessed on: 26.03.2025. [Online]. Available at: https://www.continental-automotive.com/en/solutions/server-zone-architecture/zone-control-units.html
[85] "Zonal Architecture vs. Domain Architecture", Molex. Accessed on: 26.03.2025. [Online]. Available at: https://www.molex.com/en-us/blog/zonal-architecture-vs-domain-architecture-modular-automotive-infrastructure-face-off .
[86] R. Mader, G. Winkler, T. Reindl and N. Pandya, "The Car's Electronic Architecture in Motion: The Coming Transformation", Vitesco Technologies, 2021. Accessed on: 26.03.2025 [Online]. Available at: https://www.vitesco-technologies.com/getmedia/4226dcd2-8753-48218c7a-7644ba070b82/WMS2021_Mader_The-Cars-Electronic-Architecture-in-Motion.pdf
[87] R. S. Rathore, C. Hewage, O. Kaiwartya, and J. Lloret, "In-Vehicle Communication Cyber Security: Challenges and Solutions", Sensors, vol. 22, no. 17, p. 6679, Sep. 2022, doi: 10.3390/s22176679.
[88] K. A. Khaliq, O. Chughtai, A. Shahwani, A. Qayyum, and J. Pannek, "Road Accidents Detection, Data Collection and Data Analysis Using V2X Communication and Edge/Cloud Computing", Electronics, vol. 8, no. 8, p. 896, Aug. 2019, doi: 10.3390/electronics8080896.
[89] "Zenoh", Eclipse Foundation, 2022. [Online]. Available at: https://zenoh.io/
[90] M. Barón, L. Diez, M. Zverev, J. R. Juárez, and R. Agüero, "On the performance of Zenoh in Industrial IoT Scenarios", Ad Hoc Networks, vol. 170, p. 103784, Apr. 2025, doi: 10.1016/j.adhoc.2025.103784.
[91] A. Saleh, S. Tarkoma, S. Pirttikangas, and L. Lovén, "Publish/Subscribe for Edge Intelligence: Systematic Review and Future Prospects", 2024. doi: 10.2139/ssrn.4872730.
[92] L. Lusvarghi, B. Coll-Perales, J. Gozalvez, K. Aghababaiyan, M. Almela, and M. Sepulcre, "Characterization of In-Vehicle Network Sensor Data Traffic in Autonomous Vehicles", in 2024 IEEE Vehicular Networking Conference (VNC), May 2024, pp. 211–214. doi: 10.1109/VNC61989.2024.10575960.
[93] M. Kreutzer, M. Seidler, K. Dudzik, V. P. Betancourt, and J. Becker, "Migration of Isolated Application Across Heterogeneous Edge Systems", in 2024 IEEE 8th International Conference on Fog and Edge Computing (ICFEC), May 2024, pp. 64–70. doi: 10.1109/ICFEC61590.2024.00014.
[94] F. Giarre, L. Cominardi, and P. Casari, "Realizing Flat Multi-Zone Multi-Controller Software-Defined Networks using Zenoh", in 2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Nov. 2022, pp. 45–51. doi: 10.1109/NFV-SDN56302.2022.9974876.
[95] G. Baldoni, J. Loudet, L. Cominardi, A. Corsaro, and Y. He, "Zenoh-based Dataflow Framework for Autonomous Vehicles", in 2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C), Dec. 2021, pp. 555–560. doi: 10.1109/QRS-C55045.2021.00085.
[96] L. Khorkheli, D. Bourne, G. B. Satrya, S. Elnaffar, and S. E. Choutri, "Zenoh-powered post-quantum security: protecting IoT-based smart surveillance systems", IET Conf. Proc., vol. 2023, no. 39, pp. 168–172, Jan. 2024, doi: 10.1049/icp.2024.0481.

[97] A. Corsaro and colab., "Zenoh: Unifying Communication, Storage and Computation from the Cloud to the Microcontroller", in 2023 26th Euromicro Conference on Digital System Design (DSD), Sep. 2023, pp. 422–428. doi: 10.1109/DSD60849.2023.00065.

[98] J. Zhang, X. Yu, S. Ha, J. Peña Queralta, and T. Westerlund, "Comparison of Middlewares in Edge-to-Edge and Edge-to-Cloud Communication for Distributed ROS 2 Systems", J Intell Robot Syst, v. 110, no. 4, p. 162, Nov. 2024, doi: 10.1007/s10846-024-02187-z.

[99] W.-Y. Liang, Y. Yuan, and H.-J. Lin, "A Performance Study on the Throughput and Latency of Zenoh, MQTT, Kafka, and DDS," Zenoh Taiwan Team, Zettascale Technology and National Taiwan University, Mar. 2023. [Online]. Available at: https://arxiv.org/pdf/2303.09419.pdf .

[100] K. Peeroo, P. Popov, and V. Stankovic, "A Survey on Experimental Performance Evaluation of Data Distribution Service (DDS) Implementations," Oct. 2023. Accessed on: 01.04.2025 [Online]. Available at: https://arxiv.org/pdf/2310.16630 .

[101] Eclipse Foundation, "zenoh-c," GitHub Repository, [Online]. Available at: https://github.com/eclipse-zenoh/zenoh-c .

[102] Eclipse Foundation, "zenoh-c Documentation," [Online]. Available at: https://zenoh-c.readthedocs.io/en/stable/examples.html

[103] Goh, K. H.; See, K. F. Twenty years of water utility benchmarking: A bibliometric analysis of emerging interest in water research and collaboration. Journal of Cleaner Production. 2021, 284, 124711.

[104] Kesari, K. K.; Soni, R.; Jamal, Q. M. S.; Tripathi, P.; Lal, J. A.; Jha, N. K.; Siddiqui, M. H.; Kumar, P.; Tripathi, V.; Ruoko-lainen, J. Wastewater Treatment and Reuse: a Review of its Applications and Health Implications. Water, Air, & Soil Pol-lution. 2021, 232, 208.

[105] SC5-11-2018 Horizon 2020 CE – "Digital solutions for water: linking the physical and digital world for water solutions".

[106] DS-3-2015: Horizon 2020 CE – "The role of ICT in Critical Infrastructure Protection"

[107] Mazur, D.C.; Entzminger, R.A.; Kay, J.A. Enhancing Traditional Process SCADA and Historians for Industrial and Commercial Power Systems with Energy (Via IEC 61850). IEEE Trans. Ind. Appl. 2016, 52, 76–82.

[108] Gray, A.D.L.; Pisica, I.; Taylor, G. A.; Whitehurst, L. A Standardised Modular Approach for Site SCADA, Applications Within a Water Utility. IEEE Access 2017, Volume 5, pp. 17177-17187.

[109] Wang, D. Building Value in a World of Technological Change: Data Analytics and Industry 4.0. IEEE Eng. Manag. Rev. 2018, 46, 32–33.

[110] Savastano, M.; Amendola, C.; Bellini, F.; D'Ascenzo, F. Contextual Impacts on Industrial Processes Brought by the Digital Transformation of Manufacturing: A Systematic Review. Sustainability 2019, 11, 891.

[111] Maskuriy, R.; Selamat, A.; Ali, K.N.; Maresova, P.; Krejcar, O. Industry 4.0 for the Construction Industry—How Ready Is the Industry? Appl. Sci. 2019, 9, 2819.

[112] Johansson, N.; Roth, E.; Reim, W. Smart and Sustainable eMaintenance: Capabilities for Digitalization of Maintenance. Sustainability 2019, 11, 3553.

[113] Bezerra, A.; Silva, I.; Guedes, L.A.; Silva, D.; Leitão, G.; Saito, K. Extracting Value from Industrial Alarms and Events: A Data-Driven Approach Based on Exploratory Data Analysis. Sensors 2019, 19, 2772.

[114] Zyrianoff, I.; Heideker, A.; Silva, D.; Kamienski, C. Scalability of an Internet of Things Platform for Smart Water Management for Agriculture. In Proceedings of the 23rd Conference of Open Innovations Association (FRUCT), Bologna, Italy, 13–16 Nov. 2018.

[115] Ma, K.; Bagula, A.; Nyirenda, C.; Ajayi, O. An IoT-Based Fog Computing Model. Sensors 2019, 19, 2783.

[116] Chekired, D.A.; Khoukhi, L.; Mouftah, H.T. Industrial IoT Data Scheduling Based on Hierarchical Fog Computing: A Key for Enabling Smart Factory. IEEE Trans. Ind. Inform. 2018, 14, 4590–4602.

[117] Dhiman, H.; Deb, D. Studies in Systems, Decision and Control. In Decision and Control in Hybrid Wind Farms; Springer Nature: 2020; ISBN 978-981-15-0274-3.

[118] Lee, H. Effective Dynamic Control Strategy of a Key Supplier with Multiple Downstream Manufacturers Using Industrial Internet of Things and Cloud System. Processes 2019, 7, 172.

[119] Mohammed, H.; Longva, A.; Seidu, R. Impact of Climate Forecasts on the Microbial Quality of a Drinking Water Source in Norway Using Hydrodynamic Modeling. Water J. 2019, 11, 527.

[120] Patel, R.; Gojiya, A.; Deb, D. Failure Reconfiguration of Pumps in Two Reservoirs Connected to Overhead Tank. In Innovations in Infrastructure. Advances in Intelligent Systems and Computing; Deb, D., Balas, V., Dey, R., Eds.; Springer: Singapore, 2019; Vol. 757.

[121] Fischer, J.; Freudenthaler, P.J.; Lang, R.W.; Buchberger, W.; Mantell, S.C. Chlorinated Water Induced Aging of Pipe Grade Polypropylene Random Copolymers. Polymers 2019, 11, 996

[122] Pérez, F.J.; Otín, M.R.; Mouhaffel, A.G.; Martín, R.D.; Chinarro, D. Energy and Water Consumption and Carbon Footprint in Tourist Pools Supplied by Desalination Plants: Case Study, the Canary Islands. IEEE Access 2018, 6, 11727–11737.

[123] Babunski, D.; Zaev, E.; Tuneski, A.; Bozovic, D. Optimization methods for water supply SCADA system. In Proceedings of the 7th Mediterranean Conference on Embedded Computing, Budva, Montenegro, 10–14 June 2018.

[124] Huang, Q.; Du, Z.; Lu, N.; Yu, X. Application of non-linear optimization model of groundwater in well irrigation district of northern China. In Proceedings of the Int. Symp. on Water Resource and Environmental Protection, Xi'an, China, 20–22 May 2011.

[125] Bartkiewicz, E.; Zimoch, I. Impact of Water Demand Pattern on Calibration Process. Proceedings 2018, 2, 191.

[126] Ali, E.N.; Muyibi, S.A.; Alam, M.Z.; Salleh, H.M. Optimization of water treatment parameters using processed Moringa oleifera as a natural coagulant for low turbidity water. In Proceedings of the Int. Conference on Statistics in Science, Business and Engineering, Langkawi, Malaysia, 10–12 September 2012.

[127] Whitfield, P. Monitoring water quality through data collection and analysis. In Proceedings of the GEOSS Workshop XLI, Vancouver, BC, Canada, 24 July 2011.

[128] Stang, S.; Wang, H.Y.; et.al. Influences of water quality and climate on the water-energy nexus: A spatial comparison of two water systems. J. of Environ. Manag. 2018, vol. 218, pp. 613-621.

[129] Rosinska, A.; Dabrowska, L. Selection of Coagulants for the Removal of Chosen PAH from Drinking Water. WATER Journal 2018, Volume 10, Issue 7.

[130] Tsagarakis, K.P. Operating Cost Coverage vs. Water Utility Complaints. WATER Journal 2018, 10/1.

[131] Wu, W.; et.al. Incorporation of variable-speed pumping in multiobjective genetic algorithm optimization of the design of water transmission systems. J. Water Resour. Plan. Manag. ASCE 2012.

[132] Mala-Jetmarova, H.; Sultanova, N.; Savic, D. Lost in Optimisation of Water Distribution Systems? A Literature Review of System Design. WATER Journal 2018, 10/3.

[133] Ciolofan, S.; Militaru, G.; Draghia, A.; Drobot, R.; Dragoicea, M. Optimization of Water Reservoir Operation to Minimize the Economic Losses Caused by Pollution. IEEE Access 2018, Volume 6.

[134] Diaz-Perez, F.J.; Pino-Otin, M; et.al; Energy and Water Consumption and Carbon Footprint in Tourist Pools Supplied by Desalination Plants: Case Study, the Canary Islands. IEEE Access 2018, vol. 6.

[135] Wang, D.S. Raw water quality assessment for the treatment of drinking water. Environmental Earth Sciences 2016, 75/19.

[136] Chowdhury, S. Water quality degradation in the sources of drinking water: an assessment based on 18 years of data from 441 water supply systems. Environmental Monitoring and Assessment 2018, 190/7.

[137] Delpla, I.; Florea, M.; Rodriguez, M.J. Drinking Water Source Monitoring Using Early Warning Systems Based on Data Mining Techniques. Water Resources Management 2019, 33/1, pp. 129-140.

[138] Huang, P.; Zhu, N.;et.al. Real-Time Burst Detection in District Metering Areas in Water Distribution System Based on Patterns of Water Demand with Supervised Learning. Water Journal 2018, vol. 10.

[139] Cheng, W.; Fang, H.; Xu, G.; Chen, M. Using SCADA to Detect and Locate Bursts in a Long-Distance Water Pipeline. Water Journal 2018, vol. 10.

[140] Osman, M.; Ab-Mahfouz, A.; Page P. A Survey on Data Imputation Techniques: Water Distribution System as a Use Case. IEEE Access 2018, Volume 6.

[141] Kuriqi, A., Assessment and quantification of meteorological data for implementation of weather radar in mountainous regions. MAUSAM 2016, Volume 67, pp. 789-802.

[142] Sandu, M.; Bode, F.; Danca, P.; Voicu, I. Water flow structure optimization between the screenings and grit removals in a wastewater plant. In Proceedings of the Int. Conf. on Energy and Environment (CIEM), Bucharest, Romania, 19–20 October 2017.

[143] Black, K.; Mazier, S. Optimisation of stability and efficiency of wastewater treatment. In Proceedings of the IET Water Event 2013: Process Control and Automation, Nottingham, UK, 21–22 May 2013.

[144] J. Song, Y.C. Lee, J. Lee, Deep generative model with time series-image encoding for manufacturing fault detection in die casting process. J Intel. Manuf 34, 3001–3014 (2023).

[145] J. Villena Toro, A. Wiberg and M. Tarkian, Application of optimized convolutional neural network to fixture layout in automotive parts. Int J Adv Manuf Tech. 126, 339–353 (2023).

[146] S. Latif, M. Driss, et.al., Deep Learning for the Industrial Internet of Things (IIoT): A Comprehensive Survey of Techniques, Implementation Frameworks, Potential Applications, and Future Directions. Sensors 2021, 21, 7518. https://doi.org/10.3390/s21227518.

[147] N.H. Yu, S. Baek, Fault Detection in Automatic Manufacturing Processes via 2D Image Analysis Using a Combined CNN–LSTM Model. IFIP Adv. in Inf. and Comm. Tech., vol 663. Springer.

[148] W. Qian, Y. Guo, et.al, Digital twin driven production progress prediction for discrete manufacturing workshop, Robotics and Comp.-Integrated Manuf., 80, 2023, 102456.

[149] I. Sideris, F. Crivelli, et.al., GPyro: uncertainty-aware temperature predictions for additive manufacturing. J Inte Manuf 34, 243–259 (2023)

[150] Y. Han, N. Ding, et.al., An optimized long short-term memory network based fault diagnosis model for chemical processes, J. of Proc Control, vol. 92, 2020, pp. 161-168.

[151] M. Zhang, J. Li, et. al., Deep Learning for Short-Term Voltage Stability Assessment of Power Systems, IEEE Access, vol. 9, 2021, pp. 29711-29718.

[152] Y.L.Tsai, H.C. Chang, et.al, Using Convolutional Neural Networks in the Development of a Water Pipe Leakage and Location Identification System. Appl. Sci. 2022, 12, 8034.

[153] R. Palmitessa, P.S. Mikkelsen, et.al., Soft sensing of water depth in combined sewers using LSTM neural networks with missing observations. J. Hydro-Environ. Res. 2021, 38, 106–116.

[154] M. Carratù, S. D. Iacono, et.al., "Smart Water Meter Based on Deep Neural Network and Undersampling for PWNC Detection," in IEEE Trans. on Instr. and Meas., vol. 72, pp. 1-11, 2023, Art no. 1002211.

[155] K.T.N. Nguyen, François, B., et al. Prediction of water quality extremes with composite quantile regression neural network. Environ Monit Assess 195, 284 (2023).

[156] Robles-Velasco, A.; Ramos-Salgado, et.al., Artificial Neural Networks to Forecast Failures in Water Supply Pipes. Sustain. 2021, 13, 8226.

[157] https://towardsdatascience.com/lstm-for-predictive-maintenance-on-pump-sensor-data-b43486eb3210, accessed on the 4th of June 2024.

[158] S. Matzka, "Explainable Artificial Intelligence for Predictive Maintenance Applications", 2020 Third International Conference on Artificial Intelligence for Industries (AI4I), 2020, pp. 69-74. https://doi.org/10.1109/AI4I49448.2020.00023.

[159] C. Wolf, A. Kirmse, M. Burkhalter, M. Hoffmann and T. Meisen, "Model to assess the Economic Profitability of Predictive Maintenance Projects", 2019 International Conference on High Performance Computing & Simulation (HPCS), 2019, pp. 976-981. https://doi.org/10.1109/HPCS48598.2019.9188221.

[160] K. S. Kiangala and Z. Wang, "An Effective Predictive Maintenance Framework for Conveyor Motors Using Dual Time-Series Imaging and Convolutional Neural Network in an Industry 4.0 Environment", IEEE Access, 2020, vol. 8, pp. 121033-121049. https://doi.org/10.1109/ACCESS.2020.3006788.

[161] T. Gómez, G. Gémar, M. Molinos-Senante, R. Sala-Garrido, R. Caballero, "Assessing the efficiency of wastewater treatment plants: A double-bootstrap approach", Journal of Cleaner Production, vol. 164, pp. 315-324, 15 Oct., 2017

[162] H. Han, J. Qiao, "Nonlinear Model-Predictive Control for Industrial Processes: An Application to Wastewater Treatment Process", IEEE Trans. Ind. Electr., vol. 61, issue 4, pp. 1970-1982, April, 2014

[163] A. Hauser, F. Roedler, "Interoperability: the key for smart water management", Water Science and Technology-Water Supply, vol. 15, no. 1, pp. 207-214, 2015

[164] T. Robles, R. Alcarria, D. Martin, et.al., "An IoT based reference architecture for smart water management processes", Journal of Wireless Mobile Networks, Ubiquitous Computing and Dependable Applications, vol. 6/1, pp. 4-23, 2015

[165] P. Daal, G. Gruber, J. Langeveld, D. Muschalla, F. Clemens, "Performance evaluation of real time control in urban wastewater systems in practice: Review and perspective", Environmental Modelling & Software, vol. 95, pp. 90-101, Sept., 2017

[166] M. Zamora Iribarren, C.L. Garay-Rondero, et.al., "A Review of Industry 4.0 Assessment Instruments for Digital Transformation". Applied Sciences 2024, 14, 1693. https://doi.org/10.3390/app14051693

[167] S. Smuts, A. van der Merwe, "Key Industry 4.0 Organisational Capability Prioritisation towards Organisational Transformation", Informatics 2024, 11, 16. https://doi.org/10.3390/informatics11020016

[168] M. Gombár, A. Vagaská, A. Korauš, P. Račková, “Application of Structural Equation Modelling to Cybersecurity Risk Analysis in the Era of Industry 4.0” Mathematics 2024, 12, 343.

[169] F.J. Folgado, D. Calderón, I. González, A.J. Calderón, Review of Industry 4.0 from the Perspective of Automation and Supervision Systems: Definitions, Architectures and Recent Trends. Electronics 2024, 13, 782.

[170] O. Oguntola, et.al., “Towards Leveraging Artificial Intelligence for Sustainable Cement Manufacturing: A Systematic Review of AI Applications in Electrical Energy Consumption Optimization”. Sustainability 2024, 16, 4798.

[171] M. Shaloo, et.al., “Flexible automation of quality inspection in parts assembly using CNN-based machine learning”, Procedia Computer Science, vol. 232, 2024, pp. 2921-2932, ISSN 1877-0509

[172] European Commission, Industry 5.0 Towards a sustainable, human-centric and resilient European industry, Jan. 2021.

[173] P. Fraga-Lamas, T. M. Fernández-Caramés, A. M. Rosado da Cruz and S. I. Lopes, "An Overview of Blockchain for Industry 5.0: Towards Human-Centric, Sustainable and Resilient Applications," in IEEE Access, vol. 12, pp. 116162-116201, 2024, doi: 10.1109/ACCESS.2024.3435374

[174] X. Wang et al., "A Paradigm Shift for Modeling and Operation of Oil and Gas: From Industry 4.0 in CPS to Industry 5.0 in CPSS," in IEEE Trans. on Ind. Inf., 20/7, pp. 9186-9193, 2024, doi: 10.1109/TII.2024.3378848.

[175] F.-Y. Wang, J. Yang, X. Wang, J. Li, and Q.-L. Han, “Chat with ChatGPT on Industry 5.0: Learning and decision-making for intelligent industries,” IEEE/CAA J. Automatica Sinica, vol. 10, no. 4, pp. 831–834, Apr. 2023.

[176] F. Salcher, S. Finck and M. Hellwig, "A Smart Shop Floor Information System Architecture Based on the Unified Namespace," 2024 IEEE Int. C. on Eng., Tech., and Innovation (ICE/ITMC), Portugal, 2024, pp. 1-9.

[177] M. Nast, H. Raddatz, F. Golatowski and C. Haubelt, "A Novel OPC UA PubSub Protocol Binding Using MQTT for Sensor Networks (MQTT-SN)," 2024 IEEE 29th Int. C. on Emerg. Tech. and Fact. Autom. (ETFA), Padova, Italy, 2024, pp. 1-4, doi: 10.1109/ETFA61755.2024.10711055.

[178] G. Wang, "A Cost-Effective Long-Distance Industrial IoT System," 2024 33rd Int. Conf. on Comp. Com. and Networks (ICCCN), Kailua-Kona, HI, USA, 2024, pp. 1-5, doi: 10.1109/ICCCN61486.2024.10637593.

[179] P. Koprov, A. Ramachandran, et.al., “Streaming Machine Generated Data via the MQTT Sparkplug B Protocol for Smart Factory Operations”, Manufacturing Letters, vol. 33, Supplement, 2022, pp. 66-73, ISSN 2213-8463, https://doi.org/10.1016/j.mfglet.2022.07.016.

[180] Tan, C.L., et.al., "Nexus among blockchain visibility, supply chain integration and supply chain performance in the digital transformation era", Industrial Management & Data Systems, 123/1, pp. 229-252. 2023

[181] D. Ilieva, V. Gladchenko and D. Kolev, "Engineering Risk Analysis of Potential Outcomes and Threats of the Integration of ERP Systems," 2023 4th Int. Conf. on Com., Inf., Electr. and Energy Sys. (CIEES), Plovdiv, Bulgaria, 2023, pp. 1-6, doi: 10.1109/CIEES58940.2023.10378825.

[182] R. A. Williams, G. M. Duman, et.al., "Organizational Factors Enabling Augmented Analytic and BI 4.0 Implementation Success," in IEEE Engineering Management Review, doi: 10.1109/EMR.2024.3506865.

[183] J. Mäule, T. Kutzler, et.al, "Unified Namespace and Asset Administration Shell: A Winning Combination for Digital Production," 2024 IEEE 29th Int. Conf. on Emerg. Tech. and Fact. Autom. (ETFA), Padova, Italy, 2024, pp. 1-7, doi: 10.1109/ETFA61755.2024.10710821.

[184] R. N. G. Xavier, C. Cuarelli, et.al., "Architecture Proposal for SMT Production Line in the Context of Industry 4.0," 2023 15th IEEE Int. C. on Ind. App. (INDUSCON), Brazil, 2023, pp. 518-524.

[185] M. Wiboonrat, "Cybersecurity of Industrial Automation and Control System (IACS) Networks in Biomass Power Plants," 2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE), Helsinki, Finland, 2023, pp. 1-6, doi: 10.1109/ISIE51358.2023.10228108.

[186] López-Gómez, R.; Panizo, L.; Gallardo, M.-d.-M. Flextory: Flexible Software Factory of IoT Data Consumers. Sensors 2024, 24, 2550. https://doi.org/10.3390/s24082550

[187] Sufian, A.T.; Abdullah, B.M.; Miller, O.J. Smart Manufacturing Application in Precision Manufacturing. Appl. Sci. 2025, 15, 915. https://doi.org/10.3390/app15020915

[188] M. Sverko, T. G. Grbac and M. Mikuc, "SCADA Systems With Focus on Continuous Manufacturing and Steel Industry: A Survey on Architectures, Standards, Challenges and Industry 5.0," in IEEE Access, vol. 10, pp. 109395-109430, 2022, doi: 10.1109/ACCESS.2022.3211288.

[189] Aameri B, Poveda-Villalón M, Sanfilippo EM, et al. Deriving semantic validation rules from industrial standards: An OPC UA study. Semantic Web. 2024;15(2):517-554. doi:10.3233/SW-233342

[190] S. R, G. V S, I. R, D. K, B. N and V. A, "Seamless Integration of Legacy Industrial Systems with OPC UA for Enhanced Digital Transformation," 2024 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Chennai, India, 2024, pp. 1-6, doi: 10.1109/ICPECTS62210.2024.10780333.

[191] T. P. Garcia et al., "Connecting 3D Printer to the Industrial Internet of Things (IIoT) through MQTT Sparkplug B Protocol," 2023 IEEE 15th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), Coron, Palawan, Philippines, 2023, pp. 1-6, doi: 10.1109/HNICEM60674.2023.10589143.

[192] Pu, C.; Ding, X.; Wang, P.; Xie, S.; Chen, J. Semantic Interconnection Scheme for Industrial Wireless Sensor Networks and Industrial Internet with OPC UA Pub/Sub. Sensors 2022, 22, 7762. https://doi.org/10.3390/s22207762

[193] T. Covrig, L. Miclea and A. Ciobotaru, "Development of a Simulated Industrial Process and its Control with a PLC," 2024 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, Romania, 2024, pp. 1-4, doi: 10.1109/AQTR61889.2024.10554171.

[194] J. S, T. M, A. K. A, B. K. D and D. P, "Enhancing Factory Automation Through Facial Recognition Using Phoenix PLC and Factory I/O," 2024 9th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2024, pp. 2013-2019, doi: 10.1109/ICCES63552.2024.10860204.

[195] He, W.; Baig, M.J.A.; Iqbal, M.T. An Open-Source Supervisory Control and Data Acquisition Architecture for Photovoltaic System Monitoring Using ESP32, Banana Pi M4, and Node-RED. Energies 2024, 17, 2295. https://doi.org/10.3390/en17102295

[196] R. Kaundal, S. K. Soni and S. Rajguru, "SCADA-Enhanced Real-Time OEE Visualization Driving Industry 4.0 Advancements," 2024 International Conference on Smart Systems for applications in Electrical Sciences (ICSSES), Tumakuru, India, 2024, pp. 1-6, doi: 10.1109/ICSSES62373.2024.10561431.

[197] J. Barata and I. Kayser, “Industry 5.0– Past, Present, and Near Future,” Procedia Computer Science, vol. 219, pp. 778–788, Jan. 2023, doi: 10.1016/j.procs.2023.01.351.

[198] Architecting an Industrial Unified Namespace (UNS) From 0 to 1, 1st ed., I-Flow company https://i-flow.co., 2025, pp. 1-32.

[199] S. Rosenkranz, D. Staegemann, M. Volk and K. Turowski, "Explain ing the Business-Technological Age of Legacy Information Systems," IEEE Access, vol. 12, pp. 84579-84611, 2024, doi: 10.1109/AC CESS.2024.3414377.

[200] Y. Cho, N. Sang Do, 2024. "Design and Implementation of Digital Twin Factory Synchronized in Real-Time Using MQTT" Machines 12, no. 11: 759. https://doi.org/10.3390/machines12110759

[201] M. A. Bär, A. W. Colombo, J. L. Torres, E. Fernandez, M. Rico and M. L. Caliusco, "An Industry 4.0-Compliant Digital Product Passport Approach for Realising Dairy Product Traceability," IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society, Singapore, Singapore, 2023, pp. 1-9, doi: 10.1109/IECON51785.2023.10312481