FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Remote Monitoring and Control System for Industrial Machines

Guilherme Castro Coelho

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Gil Gonçalves

July 16, 2025

© Guilherme Coelho, 2025

Resumo

Este documento explora o desenvolvimento e a implementação de um sistema de monitorização e controlo remoto para máquinas industriais, no âmbito mais amplo da Indústria 4.0. Nesta era, é dada ênfase ao papel fundamental dos protocolos de comunicação e da modelação de dados na viabilização de sistemas industriais interoperáveis, escaláveis e inteligentes.

Este trabalho baseia-se numa solução anterior de monitorização remota industrial, identificando e enfrentando limitações críticas em termos de escalabilidade, capacidade de resposta e usabilidade, no âmbito de um pacote de trabalho existente conhecido como Advanced4i, que faz parte de um projeto nacional mais amplo denominado PRODUTECH R3 – Recuperação, Resiliência e Reindustrialização, uma Agenda Mobilizadora de Tecnologias de Produção para a Reindustrialização. Com base nessa fundação, esta tese propõe uma arquitetura redesenhada que incorpora protocolos avançados, uma interface LabVIEW refatorada e uma nova camada de middleware desenvolvida para melhorar o fluxo de dados e a sincronização do sistema. Um modelo de dados estruturado e uma interface gráfica de utilizador (GUI) otimizada também são introduzidos para facilitar a monitorização em tempo real e a configuração remota.

A solução final foi implementada e validada num ambiente industrial real na IDEPA, envolvendo a integração de mais de 30 sensores. Através de testes e avaliações, o sistema demonstrou melhor capacidade de resposta, redução na perda de dados e escalabilidade robusta, com integração de até 85 sensores, mesmo sob cargas operacionais elevadas. Estes resultados confirmam a eficácia das melhorias propostas e oferecem um roteiro prático para a implementação de tecnologias da Indústria 4.0 em contextos industriais semelhantes. Este trabalho contribui de forma significativa para a transformação digital dos processos de manufatura, alinhando-se com princípios-chave como modularidade, interoperabilidade e design centrado no utilizador.

Abstract

This Document explores the development and implementation of a remote monitoring and control system for industrial machines, within the broader framework of Industry 4.0. In this era, emphasis is being placed on the pivotal role of communication protocols and data modeling in enabling interoperable, scalable, and intelligent industrial systems.

This work is built upon a prior industrial remote monitoring solution, identifying and tackling critical limitations in scalability, responsiveness, and usability, In the scope of an existing work package known as Advanced4i, which is a part of a larger National project called PRODUTECH R3 – Recovery-Resilience-Reindustrialization, a Mobilizing Agenda of Production Technologies for Reindustrialization. Building on this foundation, these thesis proposes a redesigned architecture that incorporates advanced protocols, a refactored LabVIEW interface, and a newly developed middleware layer to enhance data flow and system synchronization. A structured data model and optimized graphical user interface (GUI) are also introduced to facilitate real-time monitoring and remote configuration.

The final solution was deployed and validated in a real-world industrial setting at IDEPA, involving the integration of over 30 sensors. Through testing and evaluation, the system demonstrated improved responsiveness, reduced data loss, and robust scalability with up to 85 sensors integrated, even under increased operational loads. These results confirm the effectiveness of the proposed enhancements and provide a practical roadmap for implementing Industry 4.0 technologies in similar industrial contexts. This work contributes meaningfully to the digital transformation of manufacturing processes by aligning with key principles of modularity, interoperability, and user-centric design.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. This framework consists of 17 interconnected goals designed to address pressing global challenges such as poverty, inequality, climate change, environmental degradation, and sustainable urban and industrial development.

The specific Sustainable Development Goals referenced in this work are:

- **SDG 9: Industry, Innovation and Infrastructure**
  This project contributes to sustainable industrialization by promoting the digital transformation of manufacturing processes through the adoption of Industry 4.0 technologies. By integrating intelligent data communication protocols and improving system interoperability and scalability, the developed solution supports resilient infrastructure and fosters innovation in the industrial sector.

- **SDG 11: Sustainable Cities and Communities**
  The proposed system enhances urban and industrial sustainability by improving energy efficiency, enabling remote monitoring, and reducing resource waste in industrial environments. These improvements contribute to more sustainable industrial ecosystems, which are integral to the development of inclusive, safe, resilient, and sustainable communities.

SDG	Target	Contribution	Performance Indicators and Metrics
9	9.4	Promoting digital transformation of industrial systems through scalable and interoperable architectures, facilitating sustainable and efficient industrialization.	Number of systems upgraded with Industry 4.0 technologies; Reduction in downtime and manual intervention.
	9.5	Supporting innovation through the integration of smart data acquisition, real-time control, and modern communication protocols.	Number of smart components integrated; Increase in process automation capabilities.
11	11.6	Improving industrial energy efficiency and reducing waste by enabling remote monitoring, predictive maintenance, and data-driven management.	Reduction in equipment failures; Improved energy usage metrics across monitored systems.
	11.b	Encouraging sustainable industrial practices that support urban resilience and smart infrastructure in manufacturing environments.	Number of digitalized assets contributing to smart infrastructure; Qualitative feedback from maintainers.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Gil Gonçalves, for his guidance and support throughout the development of this thesis. His expertise and insight were instrumental in shaping the direction and quality of my work.

I am also thankful to my co-supervisors, Liliana Antão and João Pinheiro, for their collaboration, technical assistance, and valuable feedback during this research.

Special thanks to my friends for their support and thoughtful discussions throughout the research and writing process.

Finally, I wish to thank my family for their unwavering support, patience, and encouragement throughout this journey.

Guilherme Coelho

"Say not always what you know, but always know what you say."

Claudius

Contents

1 Introduction **1**
- 1.1 Context and Motivation . . . 1
- 1.2 Problem . . . 2
- 1.3 Hypothesis . . . 2
- 1.4 Research Questions . . . 3
- 1.5 Thesis Organization . . . 3

2 State of The Art **5**
- 2.1 Industry 4.0 . . . 5
  - 2.1.1 RAMI4.0 . . . 6
  - 2.1.2 IIoT . . . 7
- 2.2 Communication Protocols . . . 9
  - 2.2.1 OPC-UA . . . 9
  - 2.2.2 MQTT . . . 11
  - 2.2.3 OPC-UA and MQTT application in the industry . . . 13
  - 2.2.4 CAN Bus . . . 14
  - 2.2.5 Bluetooth CAN Bus . . . 16
- 2.3 Data Model . . . 17

3 Existing Solution **19**
- 3.1 Previous solution Architecture . . . 19
- 3.2 Communication Protocol . . . 20
  - 3.2.1 OPC-UA . . . 21
- 3.3 Data Model . . . 21
  - 3.3.1 OPC-UA Server . . . 22
  - 3.3.2 Labview Implementation . . . 23
  - 3.3.3 Graphical User Interface (GUI) . . . 24
  - 3.3.4 Preliminary Work . . . 25

4 Implementation **27**
- 4.1 Solution Startup Sequence . . . 27
- 4.2 OPC-UA Server modifications . . . 29
- 4.3 Data acquisition Module Restructure . . . 30
  - 4.3.1 XML Parsing Overhaul . . . 31
  - 4.3.2 Synchronization with the OPC-UA Server . . . 34
- 4.4 Development of a Middleware Program . . . 34
- 4.5 Backoffice Interface . . . 35
- 4.6 Implementations Results . . . 36

5 Testing and validation **39**
5.1 Test Scenarios . . . 39
5.2 Test Environment and Configuration . . . 40
5.3 Test Case Overview . . . 41
5.4 Evaluation Metrics . . . 43
5.5 Tests Results . . . 43
5.5.1 Baseline Evaluation: Tests 1–3 . . . 43
5.5.2 Evaluation of Low Complexity Scenarios: Tests 4–6 . . . 51
5.5.3 High Complexity Scenarios: Tests 7-9 . . . 54
5.5.4 Stress Testing: Test 10 . . . 56
5.5.5 Backoffice implementation . . . 58
5.5.6 Main test conclusions . . . 59

6 Results and discussion **61**
6.1 Performance Evaluation . . . 61
6.2 Scalability and Limitations . . . 62
6.3 System Evaluation: Strengths and Weaknesses . . . 62
6.4 Research Questions Revisited . . . 63
6.5 MQTT Integration . . . 63
6.6 Conclusions . . . 64

7 Conclusion and Future Work **65**

References **69**

List of Figures

2.1 Industry 4.0 Architecture[2] . . . 5
2.2 RAMI 4.0 Architecture . . . 6
2.3 IIoT Architecture example . . . 8
2.4 OPC-UA Architecture example [14] . . . 11
2.5 MQTT Architecture example [14] . . . 12
2.6 CAN-BUS Architecture[19] . . . 14
2.7 CAN-BUS Architecture Example[19] . . . 15
2.8 CAN Bluetooth Architecture[21] . . . 16

3.1 Previous system Architecture [1] . . . 20
3.2 SOSD Architecture[1] . . . 22
3.3 Labview Flow chart . . . 23
3.4 Backoffice Architecture [1] . . . 25

4.1 Startup Sequence Flowchart . . . 28
4.2 OPC-UA Server Execution Flow and System Integration . . . 30
4.3 LabVIEW Architecture and Component Interactions . . . 31
4.4 Comparison Between Sequential and Parallel Modbus Data Acquisition . . . 33
4.5 Backoffice Front Page . . . 35
4.6 Main System Architecture . . . 38

5.1 Test Setup . . . 40
5.2 Distributed and Centralized Architectures . . . 42
5.3 CPU and RAM Usage Over Time — Old Architecture (Test 1) . . . 45
5.4 CPU and RAM Usage Over Time — New Architecture (Test 1) . . . 46
5.5 CPU and RAM Usage Over Time — Old Architecture (Test 2) . . . 48
5.6 CPU and RAM Usage Over Time — New Architecture (Test 2) . . . 48
5.7 CPU and RAM Usage Over Time — Legacy Architecture (Test 3) . . . 49
5.8 CPU and RAM Usage Over Time — New Architecture (Test 3) . . . 50
5.9 CPU and RAM Usage Over Time — New Architecture (Test 4) . . . 52
5.10 CPU and RAM Usage Over Time — Legacy Architecture (Test 4) . . . 52
5.11 Sensor data acquired during the Test using the New architecture . . . 54
5.12 Sensor data acquired during the Test using previous architecture . . . 54
5.13 System setup for Test 10 using Raspberry Pi 4 and Raspberry Pi 5 . . . 57
5.14 System setup for Test 10 using Raspberry Pi 4 and Compact Rio. . . . 58

List of Tables

5.1 Summary of Test Cases and Configuration Parameters 42
5.2 Comparison of Test 1: 2s Acquisition Interval 44
5.3 Test 1.1: 2s Acquisition Interval 46
5.4 Performance Metrics for Test 2: 1s Acquisition Interval 47
5.5 Comparison of Test 3: 0.5s Acquisition Interval 49
5.6 Comparison of Test 4: 2 Equipment, 2 Sensors, 2s Acquisition Interval 51
5.7 Comparison of Tests 5 and 6 with Higher Acquisition Frequencies 53
5.8 Performance Metrics for Test 7 55
5.9 Comparison of Test 8 and Test 9 Performance 55

List of Acronyms

AAS	Administration Shell
CAN	Controller Area Network
CPS	Cyber-Physical Systems
GUI	Graphical User Interface
IIoT	Industrial Internet of Things
IoT	Internet of Things
LTE	Long Term Evolution
MQTT	Message Queuing Telemetry Transport
OPC-UA	OPC Unified Architecture
RAMI 4.0	The Reference Architecture Model for Industry 4.0
SCADA	Supervisory Control and Data Acquisition
SSH	Secure Shell
SOA	Service-Oriented Architecture
SOSD	Smart Object Self Descriptor
XML	Extensible Markup Language

Chapter 1

Introduction

Industry 4.0 has revolutionized manufacturing processes as we know them, emphasizing digitalization and real-time data exchange. Central to this transformation is remote monitoring and control of industrial machines, which is essential for optimizing production, reducing costs, and improving overall production efficiency.

1.1 Context and Motivation

This work will be carried out within the framework of an existing work package known as Advanced4i, which is a part of a larger project called PRODUTECH R3 – Recovery-Resilience-Reindustrialization, a Mobilizing Agenda of Production Technologies for Reindustrialization[1]. Advance4i aims to strengthen the national industry's ability to compete in an increasingly demanding market in terms of sustainability while reducing energy and operating costs. Given the European Union's sustainability targets, it is necessary to integrate solutions based on IoT (Internet of Things), predictive maintenance, and energy efficiency technologies. Energy efficiency, which is often neglected in poorly digitized industrial environments, suffers from a lack of visibility of the associated costs. Advance4i aims to reverse this reality by providing transparency on energy consumption at different levels, from production lines to specific equipment and processes. Specifically, on this dissertation we will focus on adapting the basis of the digitalization architecture defined.

This document delves into the critical role of remote monitoring and control systems for industrial machines, within the context of Industry 4.0. Using the capabilities of the Controller Area Network (CAN), OPC Unified Architecture (OPC-UA) and Labview Software, this research aims to develop a robust and scalable solution for managing industrial machines.

The primary motivation for this work lies in the growing need for efficient and reliable systems that can directly access and control equipment on the shop floor at scale. By integrating these protocols, this thesis seeks to establish an Internet of Things (IoT) framework that enables predictive maintenance and enhances system reliability.

1.2 Problem

One of the primary challenges facing the industry is the implementation of reliable systems that prioritize scalability and user-friendliness. These two critical factors are often overlooked, especially during the initial stages of development, when the system is designed to handle a limited number of sensors or a small-scale deployment. At this stage, the focus tends to be on functionality rather than long-term viability, leading to systems that may perform adequately under minimal loads but falter when faced with increased demands.

When such systems are later scaled up in larger industrial settings, the lack of scalability becomes a significant bottleneck. This issue is compounded by poor user-friendliness, which makes the system cumbersome to operate and maintain as its complexity grows, usually operated only by technology experts. The result is a host of unforeseen challenges that require extensive effort and resources to address, often involving substantial rework or even a complete overhaul of the original system.

The initial design of the implemented solution was based on a limited operational scope, with the primary focus on integrating a small number of sensors. Consequently, scalability considerations were not a core aspect of the original proposal. As the system expanded to support additional components and increased data flow, several issues emerged, including inefficiencies in data handling and challenges in integrating new devices.

The following section outlines the potential hypothesis to solve the scalability and User-Friendliness issues. These solutions not only aim to rectify the shortcomings of the existing system, but also establish a foundation for a more robust and adaptable architecture.

1.3 Hypothesis

This problem can be answered with the integration of 3 key components in the basis digitalization architecture. These components will be the primary work on which the Dissertation will be based:

- **Integrating an Message Queuing Telemetry Transport(MQTT) server with the architecture already available**: The incorporation of an MQTT server in the existing communication layer, already integrating OPC-UA, will facilitate seamless communication between devices and enable the development of a scalable IoT infrastructure.

- **Data Model Modification:** Both the previous solution and the one proposed in this work are built upon the Smart Object Self Descriptor (SOSD) data model. However, the proposed solution introduces modifications to the original model aimed at improving scalability and modularity, addressing limitations observed in the earlier implementation.

- **Developing a graphical user interface(GUI)**: A user-friendly GUI will provide operators with real-time visibility into the shop floor, enabling effective monitoring and control; This will be achieved by modifying the GUI already developed so it can accommodate the new MQTT interface and display essential control information.

1.4 Research Questions

To approach this problem, 3 questions were formulated to better help the research and development of this solution:

- How can wireless communication technologies, such as Bluetooth CAN bus and Long Term Evolution(LTE), be effectively integrated to enable reliable and efficient remote monitoring and control of machines in industrial environments?
- How can the proposed system be designed to be scalable to accommodate a growing number of machines and interoperable with existing industrial automation systems and IoT platforms?
- What are the key design principles for developing a user-friendly and intuitive interface for remote monitoring and control of industrial machines ?

These questions would be the main focus of the research that is present in the following chapter.

1.5 Thesis Organization

This thesis is structured into six chapters, each addressing a specific aspect of the research and development process:
Chapter 2 presents a comprehensive review of the current state of the art in remote monitoring and control systems, with a particular focus on technologies and standards aligned with the Industry 4.0 paradigm. Key developments in data acquisition, communication protocols especially OPC UA and system scalability are discussed to establish the technological context and justify the need for improvement.

Chapter 3 outlines the preliminary work conducted prior to the development of the proposed solution. This includes an analysis of the existing architecture, identification of its limitations, and the motivation for designing a new system. The findings in this chapter form the foundation upon which the proposed solution is built.

Chapter 4 details the methodology adopted in this research, along with the implementation of the proposed architecture. It covers the design principles, system modeling strategies, software tools, and integration steps that were followed to realize the new solution. This chapter also describes the main hypothesis and objectives that guided the implementation process.

Chapter 5 focuses on the testing and validation of the system. A series of controlled tests, varying in complexity and data acquisition frequency, were executed to evaluate performance. This chapter also describes the experimental setup and introduces key performance indicators used in the assessment.

Chapter 6 presents a detailed analysis of the results obtained from the tests. Comparative metrics such as setup time, resource usage, stability, and scalability are examined to assess the

effectiveness of the proposed architecture against the legacy system. This chapter highlights both the advantages and trade-offs of the new solution.

Finally, Chapter 7 concludes the dissertation by summarizing the main contributions and outcomes of the work. It reflects on the implications of the results within the context of modern industrial systems and proposes directions for future research, especially regarding system generalization, optimization, and integration with other Industry 4.0 technologies.

Chapter 2

State of The Art

2.1 Industry 4.0

Industry 4.0 is a sort of manufacturing and production systems, integrating advanced technologies to achieve greater efficiency, flexibility, and control on the shop floor. The foundation of Industry 4.0 lies in the inclusion of the Internet of Things (IoT) and Cyber-Physical Systems (CPS), which seamlessly blend physical and digital layers to create intelligent, interconnected systems. All this can be seen in the following figure 2.1 [2][3].

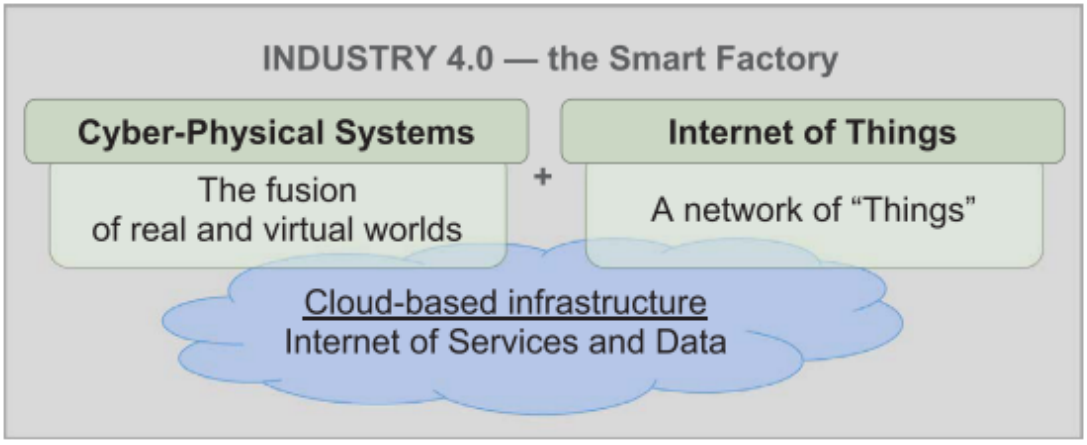

Figure 2.1: Industry 4.0 Architecture[2]

These advancements enable a shift from traditional production methods to data-driven, smart manufacturing processes. Key components of Industry 4.0 include communication protocols that manage data flow and processing, and the Asset Administration Shell (AAS), a crucial concept that organizes and manages digital representations of physical assets [4].

The AAS ensures standardized data handling, enabling interoperability and supporting the division of tasks between physical and digital domains. This framework facilitates enhanced control, real-time monitoring, and predictive decision-making, essential for modern industrial applications.

Industry 4.0 introduces the concept of smart components, which are components of CPS and possess the ability to communicate and interact within a connected environment [5][3].

2.1.1 RAMI4.0

The Reference Architecture Model for Industry 4.0 (RAMI 4.0) serves as a comprehensive framework for understanding and implementing Industry 4.0 concepts. This model adopts a Service-Oriented Architecture (SOA) to represent a multidimensional connectivity landscape. It connects factory hierarchy levels, machine lifecycle management, data types, and technology layers into a unified structure, as can be seen in the Figure 2.2.[2].

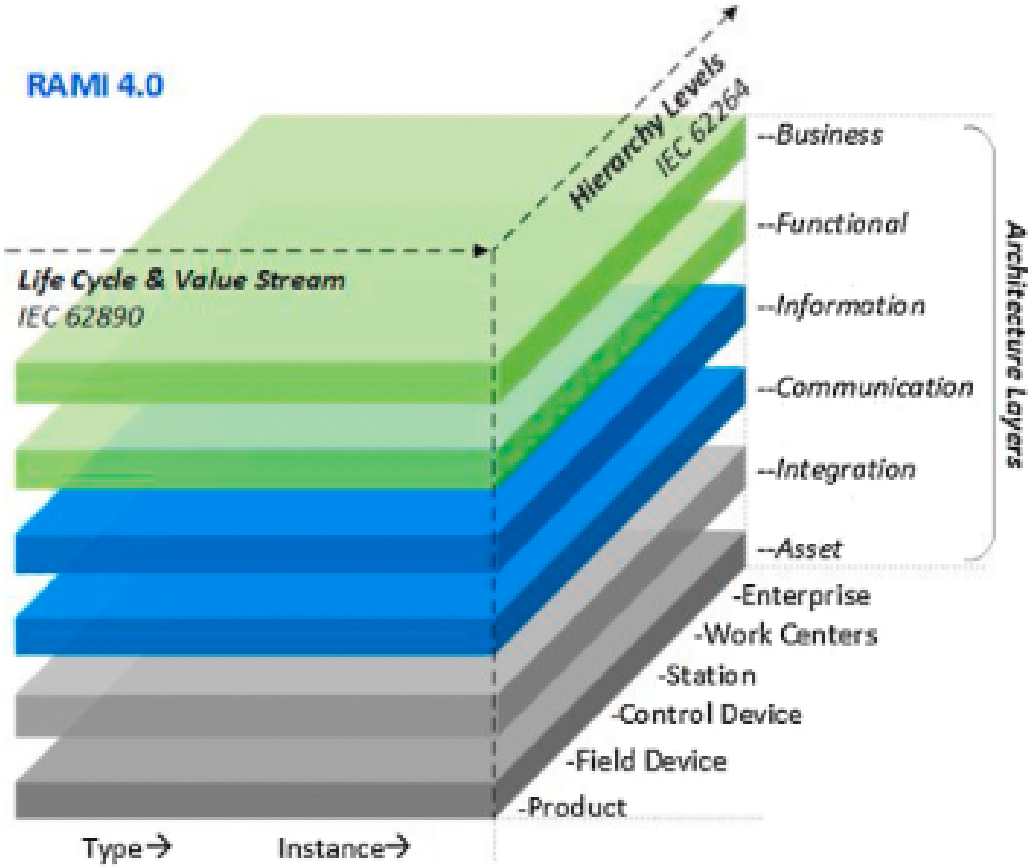

Figure 2.2: RAMI 4.0 Architecture[6]

One of the unique characteristics of RAMI 4.0 is its combination of the product lifecycle and value stream with a hierarchical approach to defining Industry 4.0 components. This framework highlights the organization, functionality, and interactions of the various elements involved in Industry 4.0 architectures [6].

The integration of the Asset Administration Shell (AAS) within RAMI 4.0 further enhances its applicability. The AAS acts as the digital twin of physical assets, providing a structured and standardized approach to managing data and interactions within the system. [2]

2.1.2 IIoT

The Industrial Internet of Things (IIoT) represents a crucial Part of Industry 4.0. It represents the application of IoT technologies within industrial settings to achieve goals specific to manufacturing and production. The IIoT connects smart objects, CPS, and advanced analytics, creating an interconnected environment that improves operational efficiency and decision-making [7][8].

However, its implementation comes with its set of challenges:

1. **Latency**: Ensuring real-time data transmission and processing.
2. **Scalability**: Managing the integration of numerous devices and systems.
3. **Interoperability**: Facilitating seamless communication between heterogeneous components.
4. **Reliability and availability**: Maintaining consistent system performance.
5. **Fault tolerance**: Ensuring resilience against failures and disruptions.

Despite these challenges, the IIoT's enables several critical components for industrial automation as: predictive maintenance, real-time monitoring, and adaptive production systems. Therefore, aligning with the objectives of Industry 4.0 to create smart and efficient manufacturing system [6][9].

A practical demonstration of these principles is provided in [10], where researchers implemented Smart Box technologies using common industrial automation protocols such as MQTT and OPC-UA. These protocols facilitate efficient communication and interoperability among devices, making them widely adopted in industrial settings. The system was built around a Raspberry Pi microcontroller, which served as the core component for demonstrating the integration of IIoT capabilities in industrial applications. This work highlights the potential of Smart Box technologies to simplify and enhance industrial processes, offering a scalable and cost-effective solution for modern factories.

A related study in [11] expands on these ideas by focusing on the abstraction and digitalization of the factory floor. This project integrates IIoT architectures to design a system that aligns with Industry 4.0 standards, emphasizing the importance of interoperability and real-time data processing. By employing widely used protocols such as OPC-UA and MQTT, the study illustrates their practical application in creating a connected and intelligent industrial environment. The project demonstrates how these protocols enable robust communication between devices, contributing to the realization of a fully digital and automated factory floor, as shown in the Figure2.3.

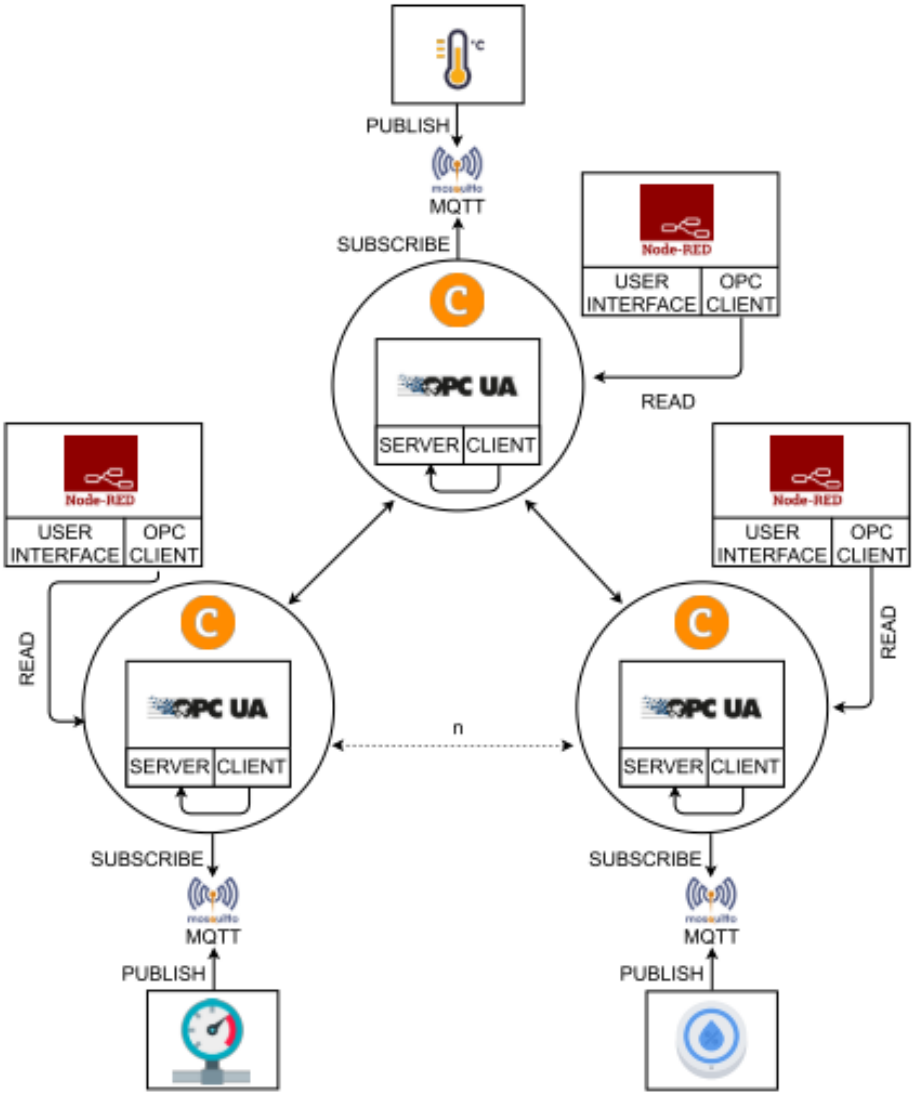

Figure 2.3: IIoT Architecture example[11]

Both studies underscore the growing importance of IIoT in industrial automation and its role in driving the transition towards smarter, more efficient, and interconnected industrial ecosystems. These implementations exemplify how leveraging IIoT technologies can enhance productivity and reduce costs.

2.2 Communication Protocols

2.2.1 OPC-UA

The OPC Unified Architecture (OPC-UA) is one of the most widely adopted and standardized protocols in industrial communication. Its significance spans various industries due to its exceptional features of data security, platform independence, and adaptability to diverse software environments. OPC-UA has gained prominence as the standard protocol for industrial communication due to its ability to provide a seamless and secure exchange of data across different systems, devices, and platforms.

This Protocol offers some qualities, such as:

- **Platform Independence and Versatility**
- **Compatibility with Embedded Systems**
- **Data Modeling Capabilities**
- **Low Latency and High Accuracy**

2.2.1.1 Datagram Modeling

One of the key features of OPC UA is its support for information modeling. The OPC UA information model is structured as a graph of interconnected nodes. Object nodes represent physical or logical components, variable nodes reflect their real-time state, and method nodes define services or actions they can perform. These nodes are linked through references, forming a hierarchical relationship graph. This model is published in the server's address space, making it accessible to OPC UA clients. While OPC UA enables seamless communication between compliant devices, integrating devices using other protocols often requires mapping their domain-specific data models to the OPC UA structure. However, most existing approaches focus on data mapping and overlook the direct construction of the OPC UA information model itself[12].

2.2.1.2 Platform Independence and Versatility

One of the core strengths of OPC-UA is its platform independence. Unlike many industrial communication protocols that are tightly coupled with specific operating systems (such as Windows), OPC-UA can operate across various software environments without being bound to a particular platform. This flexibility ensures that OPC-UA can be seamlessly integrated into a wide array of industrial applications, whether it is running on a Windows-based Supervisory Control and Data Acquisition (SCADA) system or a lightweight embedded device with limited computational resources. This ability to function on diverse platforms significantly simplifies system integration and reduces deployment complexity in industrial settings [1].

2.2.1.3 Compatibility with Embedded Systems

One advantage of OPC-UA is its suitability for embedded systems. With the growing trend of Industry 4.0 and IIoT as mentioned in the sections before. OPC-UA is designed with this in mind, making it possible to implement an OPC-UA interface even on devices with limited computational power. Embedded systems, which typically have resource constraints, can leverage OPC-UA's lightweight implementation to participate in industrial communication networks. This opens up opportunities for integrating a wide range of devices, from simple sensors to more sophisticated control units, into the larger system architecture without compromising the system's overall efficiency and scalability [1].

2.2.1.4 Data Modeling Capabilities

Another advantage of OPC-UA is its ability to provide advanced data modeling. OPC-UA supports a wide array of data types, allowing it to model not only raw data but also the relationships between different components in a system. This capability makes OPC-UA highly effective in industrial scenarios where complex systems require detailed and hierarchical representations of data. The protocol allows the creation of objects, variables, methods, and even more complex structures, facilitating a deep level of abstraction for industrial data. This level of modeling ensures that the data can be used effectively for various purposes, such as process monitoring and predictive maintenance [1].

2.2.1.5 Low Latency and High Accuracy

OPC-UA offers low latency in data communication, a crucial feature in industrial automation where real-time processing is essential. This low-latency characteristic ensures that the control systems, which are often responsible for managing operations on the factory floor, can respond to changes in real-time. By ensuring minimal delays in the transfer of data between devices and control systems, OPC-UA contributes to the high accuracy of factory floor control, leading to better operational efficiency and improved overall system performance [1].

2.2.1.6 OPC-UA in Industry 4.0

As mentioned before OPC-UA is one of the most used protocols in the Industry, it is also fundamental for the RAMI4.0 Framework. RAMI 4.0 outlines the architecture for the IIoT and smart factories, and OPC-UA is fully aligned with the needs of this model. The protocol is inherently compatible with the Administrative Shell concept in RAMI 4.0, which enables the digital representation of physical assets. The Administrative Shell is essential for connecting physical assets like machines, sensors, and actuators, to higher-level systems, enabling better management, configuration, and monitoring. The following Figure 2.4 shows an example of an OPC-UA architecture that employs critical RAMI 4.0, as digitalization of sensors and actuators via the factory floors [13].

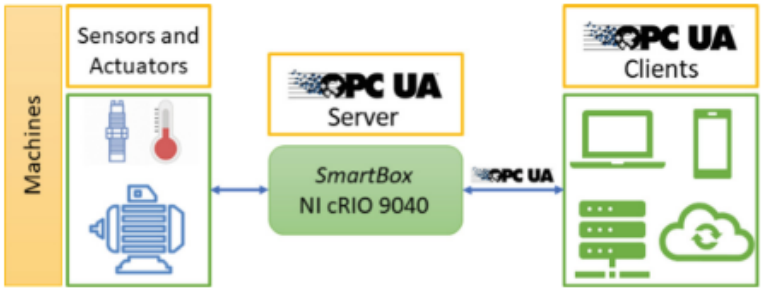

Figure 2.4: OPC-UA Architecture example [14]

2.2.1.7 Data formats supported

OPC-UA supports two primary data formats for communication, binary and XML, each with its own advantages depending on the use case.

- **Binary Format**: The binary format is used for high-performance scenarios where the devices involved have limited resources and need to prioritize speed over data interpretability. In this format, data is encoded as a bit vector, making it lightweight and efficient for devices with low computational power. The use of TCP (Transmission Control Protocol) ensures that messages are transferred with minimal overhead due to its low size, leading to higher performance and faster communication. The simplicity of the binary format results in lower computational costs during encoding and decoding, which is ideal for environments requiring rapid data exchanges, such as in real-time control systems or sensor networks [1].
- **XML Format**: The XML format is used for applications that operate at a higher layer in the communication stack. While it is computationally more demanding due to the overhead of encoding and decoding the data as structured text, XML provides a structure that is good for human interpretation. XML data is organized through tags, making it highly readable and suitable for systems that need to interact with complex data structures. For this type of communication, HTTP/SOAP is commonly employed to transmit XML messages, as the protocol is well-suited for web services and enables easy integration with higher-level business applications. Though XML format introduces slightly more latency due to the additional processing, it is essential in cases where data transparency and ease of interpretation are the priority [1].

2.2.2 MQTT

In the Industry 4.0, the integration of various communication protocols plays a crucial role in facilitating efficient data transfer across numerous devices and systems. One such widely used protocol is the Message Queuing Telemetry Transport (MQTT). This open message protocol is designed for environments where a small code footprint and low network bandwidth usage are essential, making it ideal for Internet of Things (IoT) applications. Additionally, MQTT is robust

in handling high latency or unreliable network connections, ensuring reliable data transfer even in challenging network conditions.

2.2.2.1 MQTT Architecture

MQTT operates on a publish-subscribe model, which relies on a central component called the Broker. The Broker is responsible for distributing information from the Publishers, which are IoT devices that send data, to the Subscribers, the devices or systems that receive this information. This model decouples the sending and receiving of data, making it a flexible and scalable solution for IoT ecosystems [15][14].

Unlike some other protocols, MQTT lacks a predefined data format and a standard information model. This contrasts with the previously mentioned protocol OPC UA, which offer a more structured approach to data representation. However, this simplicity in MQTT is often considered an advantage, as it allows easy configuration and deployment across a wide variety of IoT devices without the need for complex setup processes [14][15].

- **Publisher**: The IoT device responsible for sending data to the Broker. Publishers generate information and transmit it to be distributed to Subscribers.
- **Subscriber**: The device or system that receives information from the Broker. Subscribers register their interest in certain topics and receive updates whenever the Publisher sends new data.
- **Broker**: The central server that disseminates information from Publishers to the appropriate Subscribers. The Broker ensures that data is efficiently routed to all Subscribers who have expressed interest in the specific topics being published.

The following Figure 2.5 demonstrates a simple integration of an MQTT in an Industrial environment.

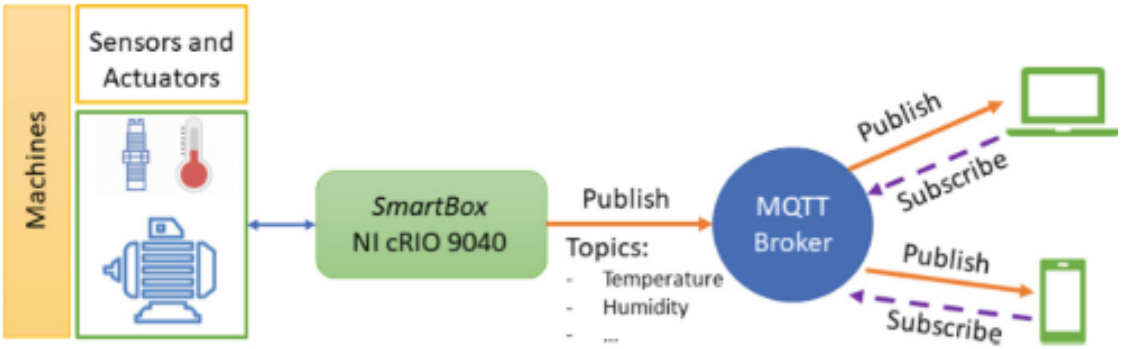

Figure 2.5: MQTT Architecture example [14]

MQTT's simple yet efficient design makes it a vital protocol in the Industry 4.0 environment, particularly for data transfer in IoT systems. Its low overhead and ability to function well in

unreliable networks enable seamless communication between devices, contributing to the real-time data exchange that is critical in modern industrial operations [2].

2.2.2.2 MQTT and OPC-UA

Despite the differences in their structure and approach, MQTT and OPC UA can be integrated to leverage the strengths of both protocols. In such integrations, MQTT is used as the transport method due to its ease of configuration and widespread support across IoT devices. By bridging these architectures, it is possible to create a system that benefits from MQTT's lightweight communication and OPC UA's rich data model and security features [16].

2.2.3 OPC-UA and MQTT application in the industry

The evolution of Industry 4.0 has introduced advanced communication protocols that ensure efficient, reliable, and secure data exchange in smart industrial systems. Among the prominent protocols explored in this context are OPC-UA and MQTT. Their applications, strengths, and challenges have been widely discussed in the literature, highlighting their significance in building smart grids, IoT models, and integrated industrial ecosystems.

The OPC-UA protocol has gained attention for its robust application in the smart grid environment within the broader framework of Industry 4.0. According to [17], this protocol has been tested under various scenarios characterized by high-frequency operations, many concurrent services, time-critical processes, and the handling of extensive datasets. These evaluations demonstrate the versatility of OPC-UA in addressing the demands of modern industrial communication systems. Furthermore, the study explores the protocol's performance under varying operational conditions, including different time measurements, providing valuable insights into its efficiency and reliability.

On the other hand, [15] provides an in-depth exploration of MQTT's applications, advantages, and implementation strategies within Industry 4.0 projects. This study emphasizes the protocol's ability to uphold critical components such as data security and privacy, making it a viable solution for use cases like smart industries and smart homes. By integrating MQTT into these environments, the paper showcases how it enables seamless communication while preserving the integrity and confidentiality of sensitive information.

A comprehensive understanding of how OPC-UA and MQTT can be applied across diverse Industry 4.0 scenarios is provided in several key references. For instance, [18] outlines the range of existing communication technologies available for Industry 4.0, emphasizing their roles and potential applications. In [16], a comparative analysis highlights the strengths and weaknesses of both OPC-UA and MQTT across various industrial parameters. Extensive testing was conducted to evaluate the efficiency and performance of these protocols, with a detailed assessment of factors such as speed, scalability, and operational efficiency.

Finally, [2] discusses the integration of these protocols within the Industry 4.0 framework and the RAMI 4.0 architecture. The study identifies challenges and benefits associated with implementing OPC-UA and MQTT, demonstrating how these protocols contribute to overcoming operational hurdles while enabling robust communication in smart industrial systems. The findings underscore the potential of combining these technologies to create a synergistic communication ecosystem tailored for Industry 4.0 applications.

2.2.4 CAN Bus

The Controller Area Network (CAN) bus is an extremely robust serial communication protocol designed to support high reliability and efficiency in data transmission. Unlike traditional communication protocols that rely on addresses, CAN bus is based on message identifiers. This means that messages are broadcast on the network, and each receiving node determines whether to process the message based on its identifier. This unique characteristic allows for flexible and efficient communication, making it highly suitable for IIoT applications [19][20].

CAN bus employs a multi-master structure, enabling any node on the network to send and receive messages. This structure supports both peer-to-peer and master-slave communication styles, making it adaptable to various system requirements. Initially developed for the automotive industry, CAN bus has since expanded its applications across multiple sectors due to its robustness and reliability [19][20].

In the context of IIoT, CAN bus addresses the demand for standards that enable remote access, scalability, and flexibility in the deployment of new sensors and devices. The protocol's ability to handle high bitrates and its inherent fault tolerance make it an ideal backbone for modern industrial networks. Its scalability ensures that systems can grow without significant overhauls. The following Figure shows the classical implementation of a CAN Bus network [19][20].

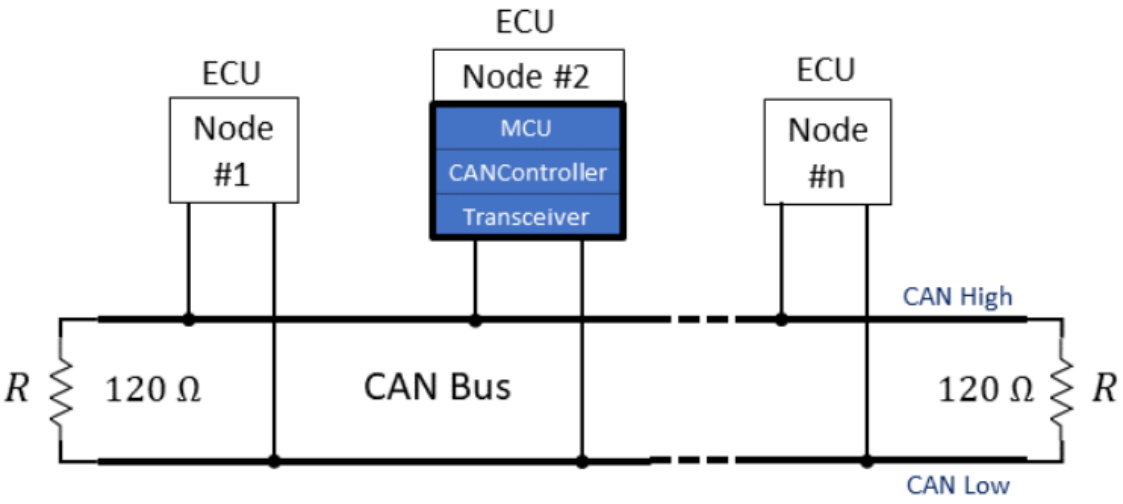

Figure 2.6: CAN-BUS Architecture[19]

All the nodes are connected to the same CAN line, which is then linked to a CAN transceiver. The transceiver is responsible for decoding the information exchanged within the CAN network. This shows the ease in integration of new sensors in the CAN Network [19][20].

In the work presented by [19], the implementation of the CAN bus communication protocol is depicted through its application in processing data from various sensors integrated into forestry equipment. This includes not only the collection and processing of sensor information but also the precise control of the harvester machinery itself. By leveraging this technology, the authors effectively demonstrate the efficiency and practicality of the protocol previously described. Additionally, this study provides valuable insights through extensive testing conducted across different types of forestry machines, highlighting the adaptability and robustness of the system, the research architecture can be seen in the following Figure 2.7.

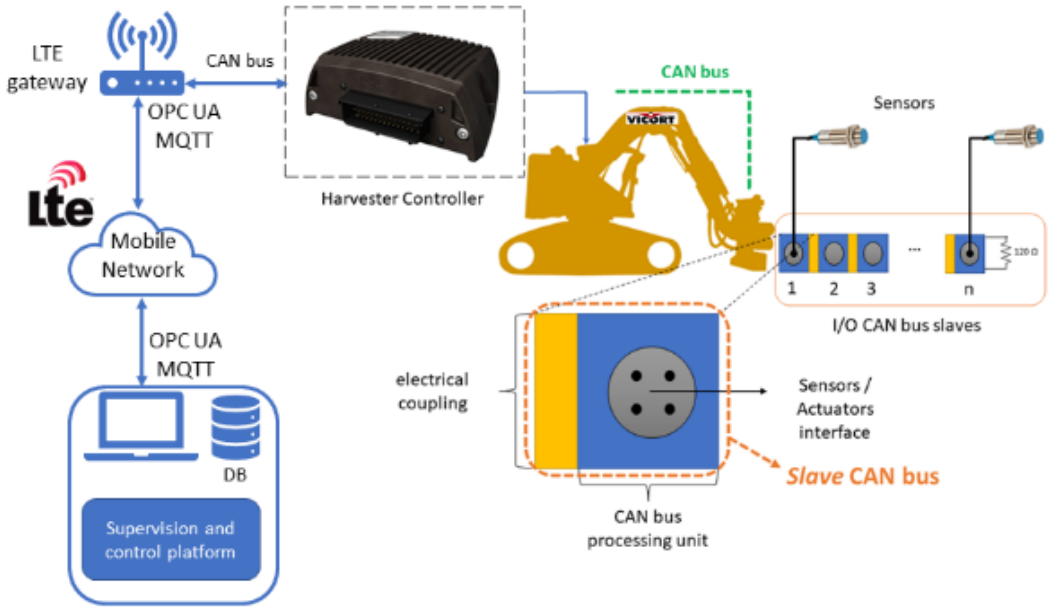

Figure 2.7: CAN-BUS Architecture Example[19]

Moreover, this implementation is integrated with the OPC-UA and MQTT protocols to construct a comprehensive control system tailored for harvesters. The work underscores the versatility of the technology and its potential to meet the diverse demands of industrial automation, particularly in the forestry sector. Importantly, it serves as a compelling example of how this technology can be successfully applied to various industrial domains.

Building on this, the research by [20] further substantiates these findings by showcasing another implementation of the CAN bus communication protocol in controlling forestry industrial machines, particularly in the operation of harvesters. This additional evidence reinforces the conclusion that the CAN bus protocol is not only highly effective but also broadly applicable across a wide range of sectors within the industrial automation industry.

2.2.5 Bluetooth CAN Bus

The integration of CAN bus with Bluetooth technology creates a hybrid network architecture known as the Bluetooth CAN Bus. This fusion leverages the strengths of both technologies, combining the robustness of CAN bus with the wireless capabilities of Bluetooth.

Bluetooth technology supports point-to-point and point-to-multipoint connections. In a typical Bluetooth network, a master node can manage up to seven active slave nodes, forming a star-type topology. By integrating a CAN bus node with a Bluetooth master device, we can create a CAN-Bluetooth node. This node serves as a bridge between the wired CAN bus network and the wireless Bluetooth network, allowing for seamless data exchange [21].

In this architecture, each CAN-Bluetooth node operates as a master device in the Bluetooth network. It forms a one-to-many master-slave network architecture with up to seven Bluetooth-based sensors or devices. This setup enables the extension of CAN bus communication to wireless devices, enhancing the flexibility and scalability of industrial networks [21].

The figure below illustrates the Bluetooth CAN Bus architecture, showcasing the integration of wired and wireless communication[21].

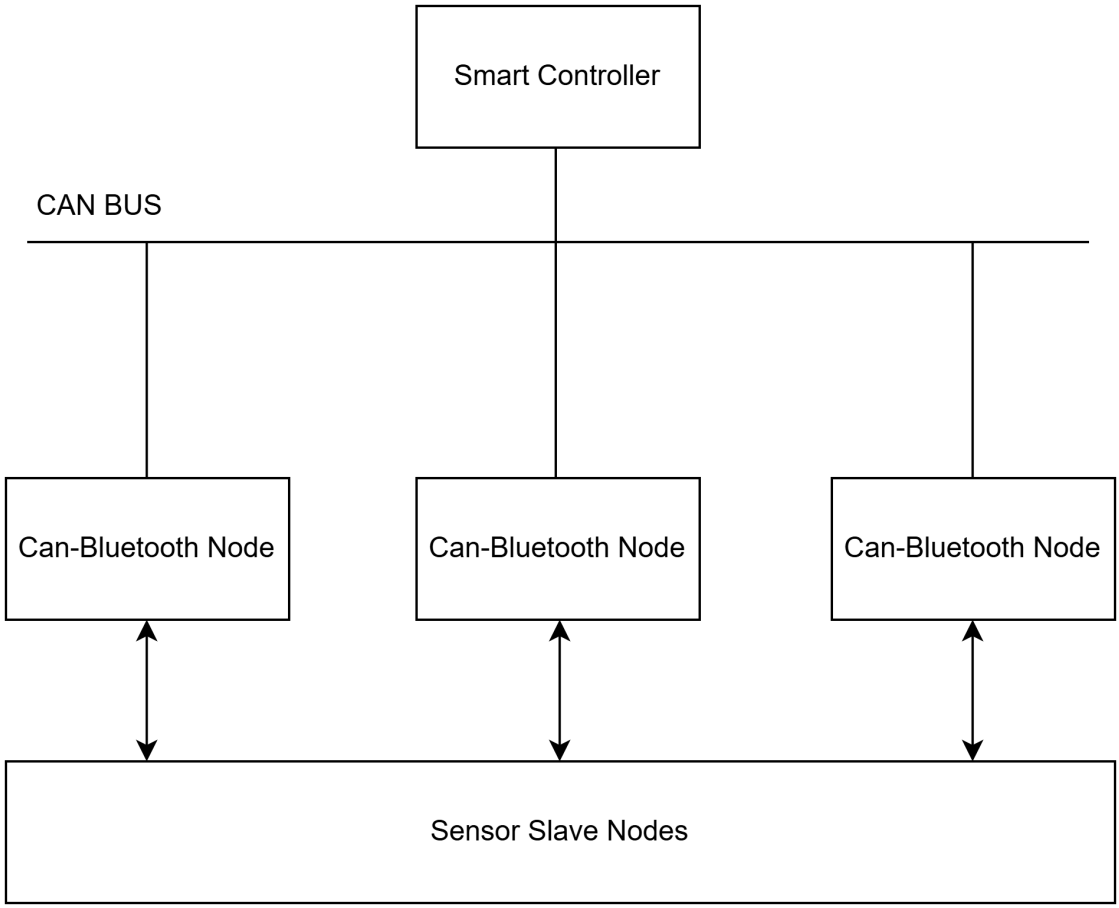

Figure 2.8: CAN Bluetooth Architecture[21]

The communication within a Bluetooth CAN Bus network follows Core steps principles:

- **Message Broadcasting**: CAN messages are broadcast to all nodes on the network, and each node decides whether to process the message based on its identifier.
- **Master-Slave Connections**: The Bluetooth segment of the network employs a master-slave structure, where the master node communicates with multiple slave nodes.
- **Data Conversion**: The CAN-Bluetooth intelligent node acts as a translator, converting CAN message to Bluetooth format and vice versa, ensuring seamless communication between wired and wireless segments.

Although this type of communication is more uncommon than the CAN Bus technology, it has its utilities and advantages. In the work presented by [21], the advantages discussed earlier are leveraged to develop a robust and secure protocol tailored for transmitting information via CAN-Bluetooth. This protocol is designed with a specific data format and structured protocol design that strictly adheres to the guiding principles outlined in the earlier sections of the study. Furthermore, the research goes beyond theoretical exploration by demonstrating the practical application of the protocol within a contained Industrial environment. The implementation not only validates the feasibility of the design but also showcases its effectiveness in meeting the demands of industrial communication, where reliability, security, and efficiency are of paramount importance. This comprehensive approach underlines the potential of the protocol to address critical challenges in wireless communication, particularly in contexts requiring stringent security and data integrity measures.

2.3 Data Model

In Industry 4.0, the concept of the Smart Component (also known as Component 4.0) extends the functionality of the ASS by enabling more advanced interaction capabilities. A Smart Component refers to an intelligent industrial device or unit equipped with sensors, actuators, and communication capabilities that allow it to process data and interact autonomously with other components within a network[22][23].

The Asset Administration Shell (ASS) is one of the main factors in the digital transformation of industrial systems, particularly within the framework of RAMI 4.0. The ASS acts as a digital representation of an asset, be it a machine, sensor, actuator, or any other industrial component by abstracting its properties, capabilities, and behavior [22][23].

The integration of Asset Administration Shells (ASS) and Smart Components within the RAMI 4.0 framework is central to the evolution of Industry 4.0. By abstracting the functionality of physical assets and mapping them into standardized digital models, ASS enables seamless communication and interaction across diverse industrial systems. The adoption of standardized interfaces, such as those based on OPC UA, ensures that Smart Components can interoperate both locally and remotely, facilitating real-time data exchange, operational flexibility, and intelligent decision-making [22][23].

Chapter 3

Existing Solution

This dissertation builds upon a foundational project aligned with Industry 4.0 principles, focusing on the integration and digitalization of factory operations to enhance control, monitoring, and overall efficiency. The solution is being implemented in a real manufacturing company IDEPA, a recognized leader in the labels, tapes, and textile accessories market. Approximately 30 sensors will be digitalized as part of this effort, laying the groundwork for a more connected and intelligent factory environment. Despite the success of the initial implementation, several challenges remain, particularly in terms of scalability and user accessibility. These challenges stem from limitations inherent to the Labview Implementation, which result in data loss and system crashes under heavy load. Furthermore, the existing solution lacks a straightforward and efficient approach to scaling up to accommodate additional devices or data streams. Addressing these limitations is a primary focus of this dissertation, with the goal of creating a system that is both more scalable and more user-friendly. In this chapter will focus on the description of the previously mentioned already implemented solution, ending with its problems and possible solutions.

One of the main challenges in industrial systems is ensuring scalability and user-friendliness factors often overlooked during initial development, where the focus is typically on basic functionality for small-scale sensor setups. As these systems grow, lack of scalability and poor usability become major obstacles, requiring significant rework or complete redesigns. The current solution was originally designed with a limited scope, leading to inefficiencies and integration difficulties as it expanded. To address these issues, the following chapter proposes changes aimed at creating a more scalable and user-friendly architecture. IDEPA plans to implement this improved solution on their factory floor to monitor and measure the facility's electric consumption.

3.1 Previous solution Architecture

The previous solution architecture, as illustrated in the Figure 3.1, integrates several key components to establish a robust and efficient data exchange system within the factory environment. These components include an OPC-UA server, a LabVIEW program, and a Backoffice, all working

in conjunction to ensure uninterrupted communication and data flow across the industrial infrastructure.

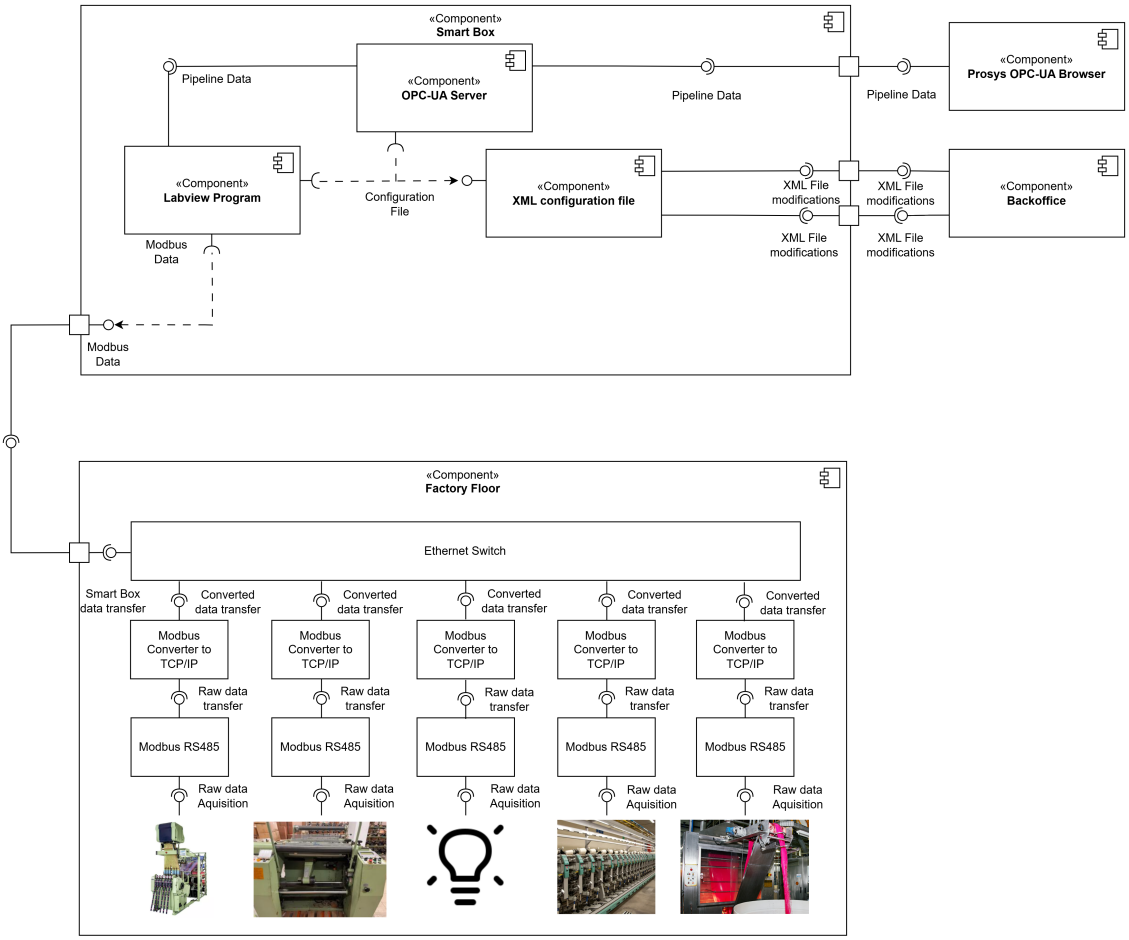

Figure 3.1: Previous system Architecture [1]

3.2 Communication Protocol

Industrial automation relies heavily on establishing a robust communication protocol to ensure seamless interaction between diverse industrial equipment. In the previous solution, OPC-UA was selected as the primary communication protocol due to its widespread adoption and alignment with Industry 4.0 standards. The current network topology is structured as follows: an OPC-UA server receives data from a data acquisition system implemented in LabVIEW. The configuration of this server is defined via an XML file, which enables flexible and scalable system setup. This XML configuration file is accessible through a locally hosted web server, which provides real-time monitoring of the OPC-UA server status, as well as tools for direct customization, management, and configuration. The web server can be deployed on any compatible machine, provided that the appropriate SSH key is available for secure access.

3.2.1 OPC-UA

OPC-UA is widely used in communication frameworks for Industry 4.0 projects, primarily due to its flexible architecture, which enables the definition of robust data models and standardized data exchange rules for a wide range of information types. This knowledge was the foundation that provided the information for the development of the Smart Object Data Model, which extends the OPC-UA Data Access Information Model to enable various data interactions [24][1].

OPC-UA also supports both binary and XML message encodings, allowing the integration of an XML configuration file within the system architecture. As previously described, this XML configuration file can be accessed through the local hosted web server, facilitating remote monitoring and configuration of the OPC-UA server without requiring direct access to the server and production disruption.

Despite its advantages, OPC-UA presents certain limitations—particularly in terms of the maximum data load per node and issues related to synchronization.

In the previous implementations, a significant drawback was the lack of responsiveness and limited control and visibility available to users interacting with the server. These issues posed challenges for scalability and hindered the broader deployment of the solution [24][1].

3.3 Data Model

Both solutions are built upon a shared data model known as the Smart Object Self Descriptor (SOSD). This model was specifically designed to be compatible with multiple communication protocols, including MQTT and OPC-UA, thereby enabling seamless data exchange across diverse devices and systems. The SOSD model embodies the key characteristics and benefits of a Component 4.0, a well-established construct within the RAMI 4.0 (Reference Architectural Model for Industry 4.0) framework. This alignment ensures compliance with Industry 4.0 principles, particularly those emphasizing interoperability, modularity, and decentralized control [24][1].

The SOSD model facilitates the transformation of physical assets such as machines, sensors, and actuators into digital entities that are accessible via the OPC-UA network. As a digital representation of Component 4.0, SOSD enables the modeling of complex industrial environments by providing a flexible and standardized framework to define attributes, behaviors, and interrelationships among various manufacturing components [24][1][14].

Through its integration with OPC-UA, SOSD empowers users to remotely monitor and control manufacturing processes, offering critical insights into system performance, fault diagnosis, and decision-making. A significant advantage of this model is its ability to incorporate off-the-shelf software components, reducing reliance on low-level programming and facilitating more agile and scalable system integration [24][1][14].

The SOSD Architecture can be seen in the following Figure 3.2, where we can see all the components and relationships that compose the model.

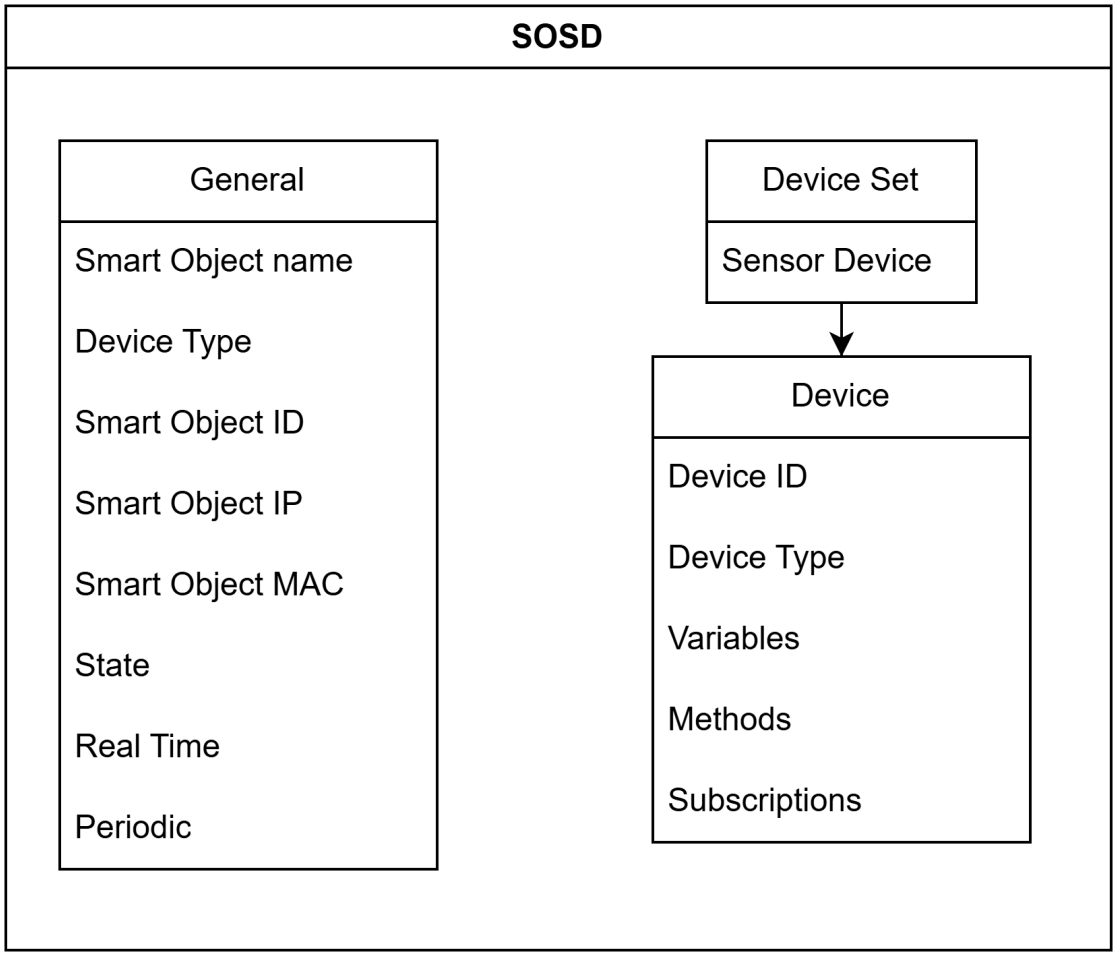

Figure 3.2: SOSD Architecture[1]

3.3.1 OPC-UA Server

One of the main components of the system is the OPC-UA server. This program is responsible for exposing real-time factory data to the user and relies heavily on the SOSD XML file previously discussed. This XML file serves as a foundational element of the server, enabling the structured creation of data pipelines necessary for efficient and reliable data transfer throughout the system.

Despite its central role, the OPC-UA server exhibits certain limitations that this dissertation aims to address, including:

- **Lack of responsiveness:** When the user attempts to terminate the server via programmatic interruption, the shutdown process fails, requiring the process to be manually killed from the console.

- **Inefficiency in pipeline creation:** The current implementation involves unnecessary steps during pipeline setup, which increases the time required to properly initialize the server.

The server follows a defined sequence of operations before it becomes available to receive data. First, it reads the XML file and constructs the corresponding data pipelines. Once all pipelines are generated, they are assigned to a designated namespace. Only after these steps are completed does the server initialize and begin waiting for incoming data.

3.3.2 Labview Implementation

The architecture of the LabVIEW program is depicted in the following Figure 3.3, illustrating both the internal data flow and the core design issues in the following section.

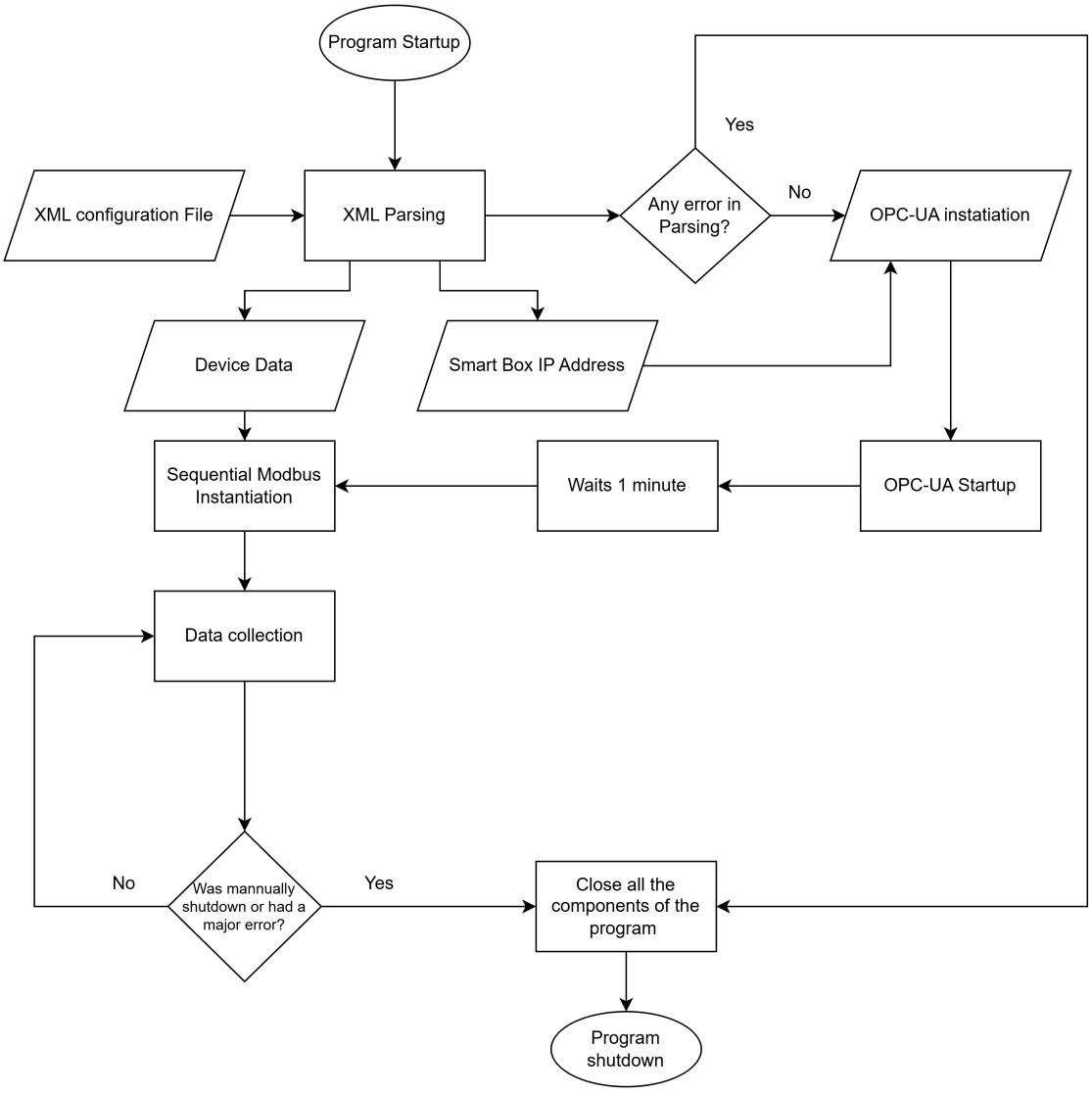

Figure 3.3: Labview Flow chart

The LabVIEW program functions as the data acquisition and processing service. It reads data from the Modbus bus, processes it, and writes it to the pipelines established by the OPC-UA server, as described in Section 3.3.1. To determine the appropriate Modbus registers to access,

the program also relies on the SOSD XML file during its initialization. This file is parsed into a structured array of devices, after which the program performs sequential reads of all associated Modbus registers and IP addresses.

While the program performs adequately at a small scale, it presents significant challenges when scaled, due to core implementation issues that this dissertation aims to address:

- **Lack of synchronization with the OPC-UA server:** The original design did not account for scalability and omitted any synchronization mechanism between the LabVIEW interface and the OPC-UA server. Consequently, the LabVIEW program may begin transmitting data before the server is ready to receive it.
- **Sequential Modbus data acquisition:** The program performs Modbus reads in a strictly sequential manner. This approach, while initially sufficient, becomes inefficient at scale and introduces fragility; any communication error on the Modbus bus can lead to significant disruptions in data collection.

Although it has these designs flaws, the system data collection capabilities can be enhanced with additional data sources. Ethernet switches can be connected to the Compact Rio industrial Computer, providing supplementary factory information. This expanded data collection enables a more comprehensive understanding of factory operations, leading to improved decision-making and optimization strategies[24][1].

3.3.3 Graphical User Interface (GUI)

Another key component of the system is the Backoffice, a locally hosted web server that provides a reliable Graphical User Interface (GUI) for system interaction.

The Backoffice runs on the same device as the OPC-UA server and the LabVIEW interface, specifically the CompactRIO 9040. These two components depend on a shared XML configuration file, as discussed earlier. This XML file encapsulates critical operational data, including sensor configuration parameters such as names, unique identifiers, and data acquisition settings.

The Backoffice enables external access to this XML file through its web interface. It hosts a GUI application that allows authorized maintainers to remotely view and modify system configuration in real time. Importantly, these modifications can be performed without interrupting data collection, thereby offering a powerful tool for continuous monitoring and remote system management. This remote access capability substantially enhances the system's flexibility and accessibility [1].

Despite its utility, the Backoffice presents several limitations, all adressed in this dissertation:

- **Insufficient system visibility:** Although the GUI provides basic information about the OPC-UA server, it lacks essential functionalities that could enhance user experience and improve control over factory operations, such as mobus information and sensors per equipments information.

- **Unreliable features:** Some implemented features do not function as intended or fail entirely. This lack of reliability has proven to be one of the most significant drawbacks of the current system architecture.

The complete system consists of multiple interconnected components, as illustrated in the Figure 3.4.

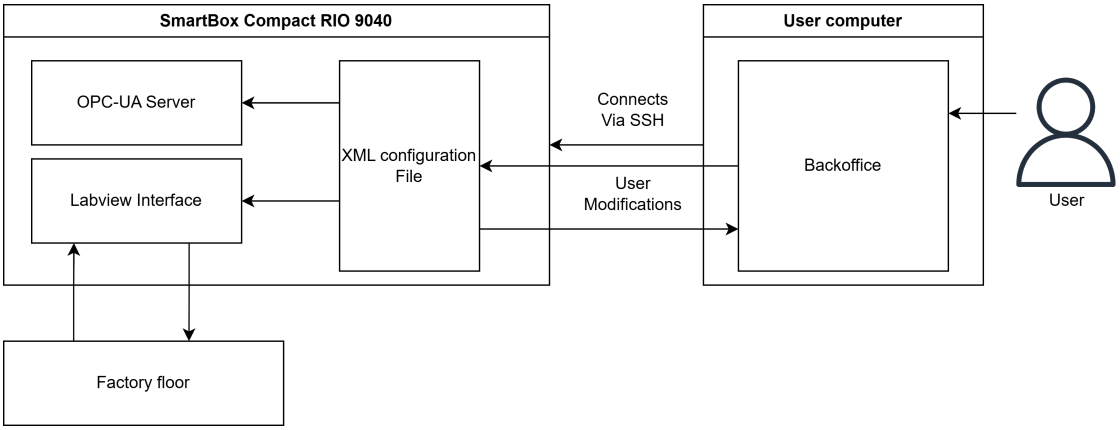

Figure 3.4: Backoffice Architecture [1]

3.3.4 Preliminary Work

Using this solution, efforts were made to integrate multiple sensors into the system. This integration process exposed many of the issues discussed in the previous sections of this chapter, including limited scalability and poor user-friendliness. Despite these challenges, 10 sensors were successfully implemented and operated correctly, making a total of 20 sensors implemented. Attempts to add more sensors led to system crashes and failures in data transmission, clearly demonstrating the architecture's lack of scalability.

Another major obstacle encountered during this dissertation work was the absence of comprehensive documentation. This significantly hindered efforts to modify or extend the system, adding to the complexity of maintaining and upgrading the solution.

Nevertheless, despite the architectural limitations and implementation challenges, this system contributes meaningfully to factory optimization. It supports improved decision-making and enhanced operational efficiency, aligning with the objectives of an Industry 4.0 system.

Chapter 4

Implementation

As discussed in previous chapters, the system is deployed on the CompactRIO 9040, which functions as the local data acquisition and control unit, this control unit is in the IDEPA factory. This embedded industrial controller interfaces directly with various sensors and machines on the factory floor, playing a pivotal role in enabling real-time monitoring and control. Additionally, it hosts the LabView interface and the OPC-UA server that exposes sensor data in accordance with the Industry 4.0 (I4.0) component model, facilitating interoperability and standardized data communication.

One of the initial directions explored during development was the integration of an MQTT broker to act as a messaging intermediary between the LabVIEW interface and the OPC-UA server. While MQTT is widely used in IoT architectures for lightweight and scalable data exchange, as discussed in chapter 2, its implementation in this specific context was found to introduce unnecessary complexity. The LabVIEW program already writes sensor data directly into multiple OPC-UA pipelines. Introducing an MQTT broker would have required LabVIEW to publish data to MQTT topics, which would then need to be relayed to the OPC-UA server, creating an additional communication layer without providing meaningful advantages. This added indirection would complicate synchronization and introduce potential points of failure.

As a result of this analysis, the MQTT-based approach was abandoned in favor of a more efficient and tightly integrated solution. Instead, a custom middleware program was developed to manage synchronization and enhance the flow of information within the system. This middleware ensures that all components are aware of the system state and can coordinate operations effectively. The design, implementation, and functional benefits of this middleware layer will be discussed in detail in the following sections.

4.1 Solution Startup Sequence

The solution follows a well-defined startup sequence that ensures proper initialization of all system components before commencing data acquisition and communication tasks. The sequence proceeds as follows:

1. The CompactRIO 9040 system powers on and boots into the National Instruments Linux Real-Time environment.

2. Upon startup, the LabVIEW application begins execution. It first parses the configuration details provided in an XML file, which includes definitions of the sensors and equipment structure.

3. After parsing the XML file, the LabVIEW application launches the custom middleware program. This middleware is responsible for handling communication with external devices and preparing system status information.

4. The middleware program scans the system to detect available and unavailable Modbus Ids. It logs this information and generates a status file containing details about the system configuration and connectivity. This file is later transmitted to the backoffice system for monitoring and validation purposes.

5. Once the port checks are completed, the middleware proceeds to initialize the OPC-UA server. During this time, the LabVIEW interface remains in a waiting state, monitoring for a synchronization signal.

6. Upon successful initialization, the OPC-UA server creates a signal file, which serves as a trigger for the LabVIEW interface to resume execution.

7. With the confirmation in place, the LabVIEW application continues by instantiating the Modbus communication objects for each sensor registered in the initial XML configuration. These instances enable real-time data acquisition from all connected devices.

This startup sequence describes the synchronized and robust initialization of the system components. The following flowchart (Figure 4.1) describes the complete flow of information:

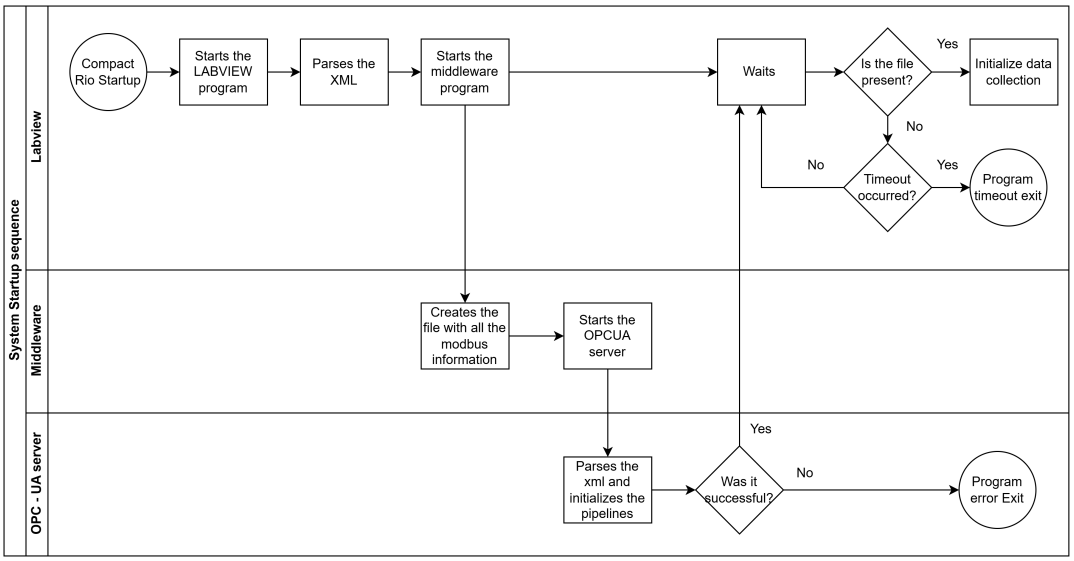

Figure 4.1: Startup Sequence Flowchart

4.2 OPC-UA Server modifications

The first focus of this dissertation was the correction and optimization of the OPC-UA server, which is a foundational component of the system architecture. As the main interface between the factory data and the end user, the OPC-UA server plays a critical role in data availability, system responsiveness, and overall user experience. Its performance directly impacts the effectiveness of downstream components, including the LabVIEW interface and the Backoffice application. Consequently, addressing its limitations was prioritized to enable a scalable and user-friendly solution.

The initial improvement targeted the pipeline creation process. In the previous solution implementation, the creation of data pipelines was hindered by redundant operations and legacy code, resulting in unnecessary delays during server initialization. A complete refactoring of the relevant codebase was undertaken to streamline this process. Obsolete sections that were no longer used in the final implementation were removed, reducing the system's complexity and improving maintainability. These changes decreased the time required to configure and launch the server.

Following the optimization of pipeline creation, the server's lack of responsiveness was addressed. In the previous version, the server failed to shut down gracefully when a termination command was issued, often requiring manual intervention to forcibly terminate the process via the console. To resolve this, a shutdown hook mechanism was implemented. This enhancement allows the server to cleanly exit and release system resources upon receiving a shutdown signal, thereby eliminating the need for manual termination and improving system reliability.

While these two improvements (pipeline optimization and controlled shutdown) solved the core technical issues of the OPC-UA server, additional features were also introduced to enhance system scalability and user-friendliness. A key enhancement was the introduction of a handshake mechanism between the OPC-UA server and the LabVIEW interface. Upon successful completion of its setup routine, the server generates a file that signals its readiness. The LabVIEW program monitors the presence of this file and begins data transmission only once it is detected, thereby preventing race conditions that had previously led to data loss or instability. This file is automatically deleted when the server shuts down, ensuring that each execution cycle begins with proper synchronization.

To further aid system transparency and user interaction, detailed logging was added throughout the server initialization process. These logs provide real-time feedback regarding the parsing of the XML configuration file, the creation of data pipelines, and the overall server setup. This added visibility not only facilitates debugging and maintenance but also enhances user confidence and operational awareness.

The complete execution flow of the OPC-UA server and its interactions with other components of the system are depicted in the following flowchart4.2. This diagram illustrates the sequence of operations performed by the server, the dependencies it creates, and its impact on the overall system behavior.

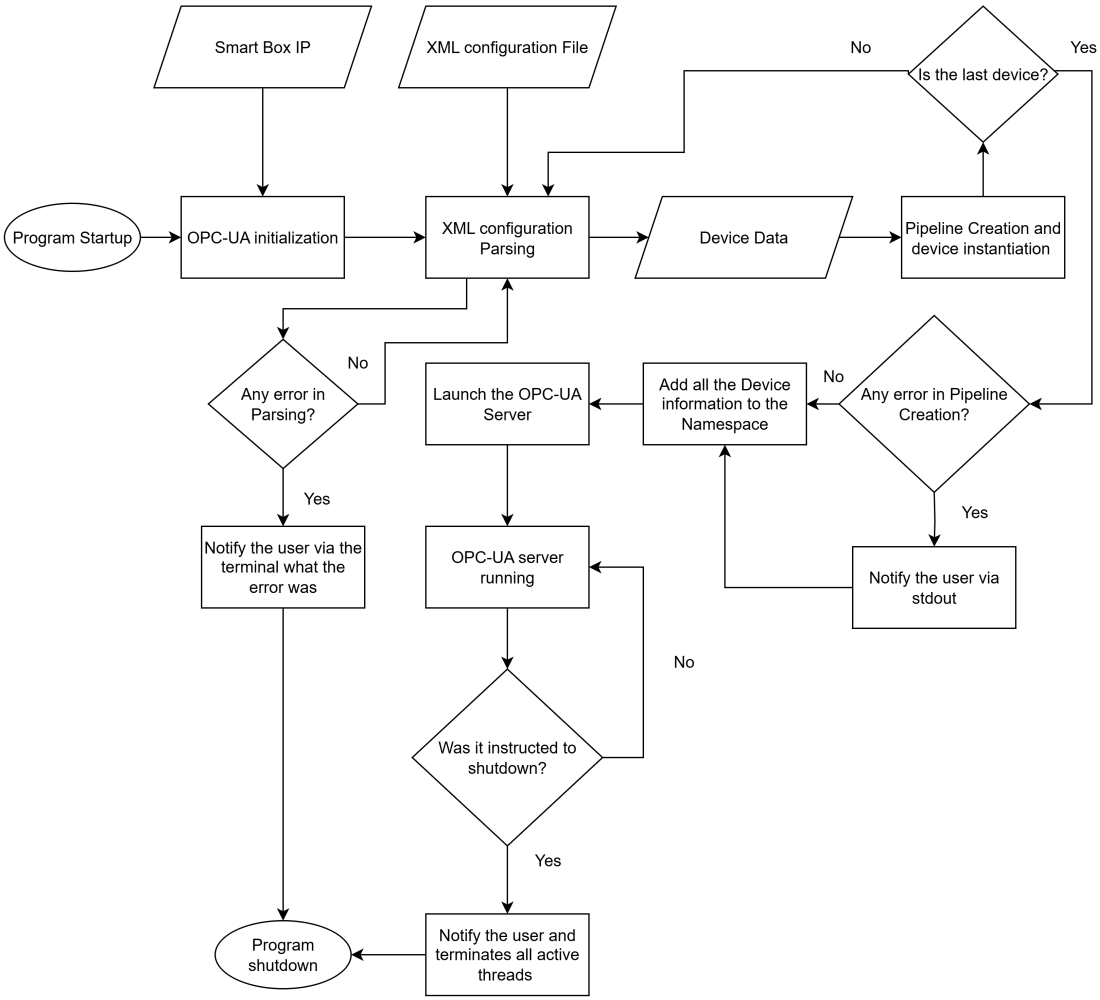

Figure 4.2: OPC-UA Server Execution Flow and System Integration

4.3 Data acquisition Module Restructure

Following the modifications made to the OPC-UA server, a significant overhaul of the data acquisition module was undertaken. This refactor aimed to address fundamental limitations present in the prior implementation, particularly those related to synchronization, scalability, and data processing efficiency. The restructured of the data acquisition module program can be categorized into three primary areas:

- **XML Parsing Refactor:** Introduction of two distinct data structures, *Equipments* and *Sensors*, replacing the previously monolithic *Devices* array.

- **OPC-UA Synchronization Mechanism:** Implementation of a file-based handshake protocol that ensures proper startup sequencing between the LabVIEW interface and the OPC-UA server.

- **Parallelized Data Acquisition:** Transition from a sequential to a multithreaded data collection architecture to enhance scalability and robustness.

The following Diagram 4.3 outlines the internal structure of the restructured LabVIEW interface and illustrates the flow of operations, highlighting the interdependence between components and their respective responsibilities.

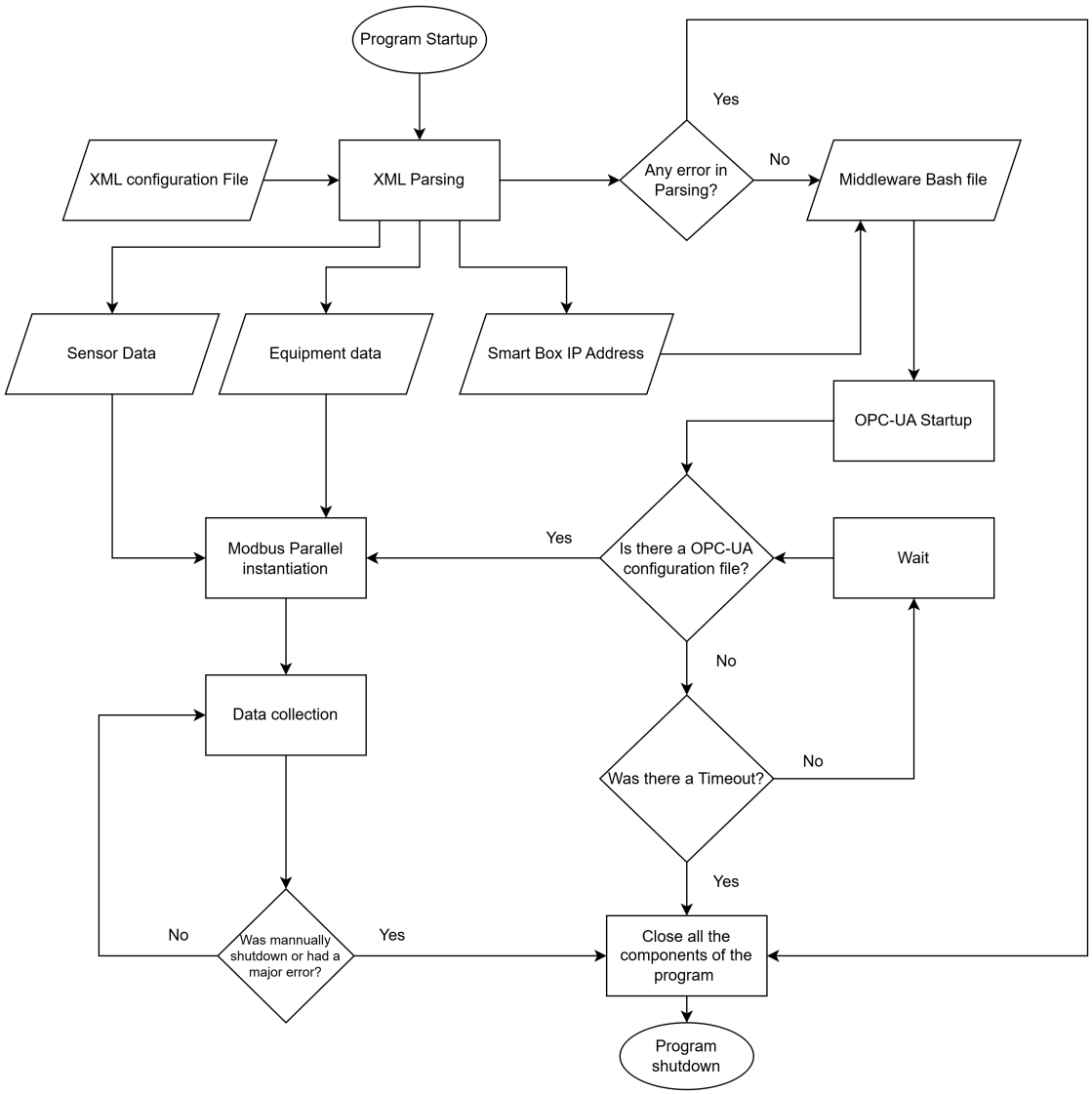

Figure 4.3: LabVIEW Architecture and Component Interactions

4.3.1 XML Parsing Overhaul

One of the most foundational changes involved refactoring the XML parsing mechanism. In the previous system, all data retrieved from the XML configuration file was consolidated into a single

array of Devices, which lacked clarity and imposed constraints on data manipulation and extension. In the new and extended version, this process was restructured to produce two independent yet related data structures: an Equipment array and a Sensor array.

The Equipment data structure encapsulates attributes such as equipment identifiers, names, and a list of associated sensor subscriptions. Meanwhile, the Sensor structure retains core characteristics from the original Devices array but includes an additional reference to the subscribed equipment. This modular separation enhances both readability and maintainability, and more importantly, lays the groundwork for the implementation of a parallel data acquisition model. Only sensors linked to existing equipment are considered valid for data collection, which enforces a cleaner and more logically consistent architecture.

4.3.1.1 Parallelized Modbus Data Acquisition

The most technically demanding enhancement involved converting the data acquisition logic from a sequential to a parallel execution model. The prior system utilized a single loop to sequentially initialize and poll all Modbus devices, a structure that imposed severe limitations on scalability and fault tolerance. For example, if a single sensor failed or was unresponsive, it could block or delay data collection for all other devices.

In the new architecture, a hierarchical multithreaded model was implemented. First, a separate thread is created for each equipment. Within each equipment thread, additional threads are created to manage communication with individual sensors. Each of these sensor specific threads independently initializes the corresponding Modbus connection and handles data polling. A comparison between these two implementations can be seen in the Figure 4.4 below.

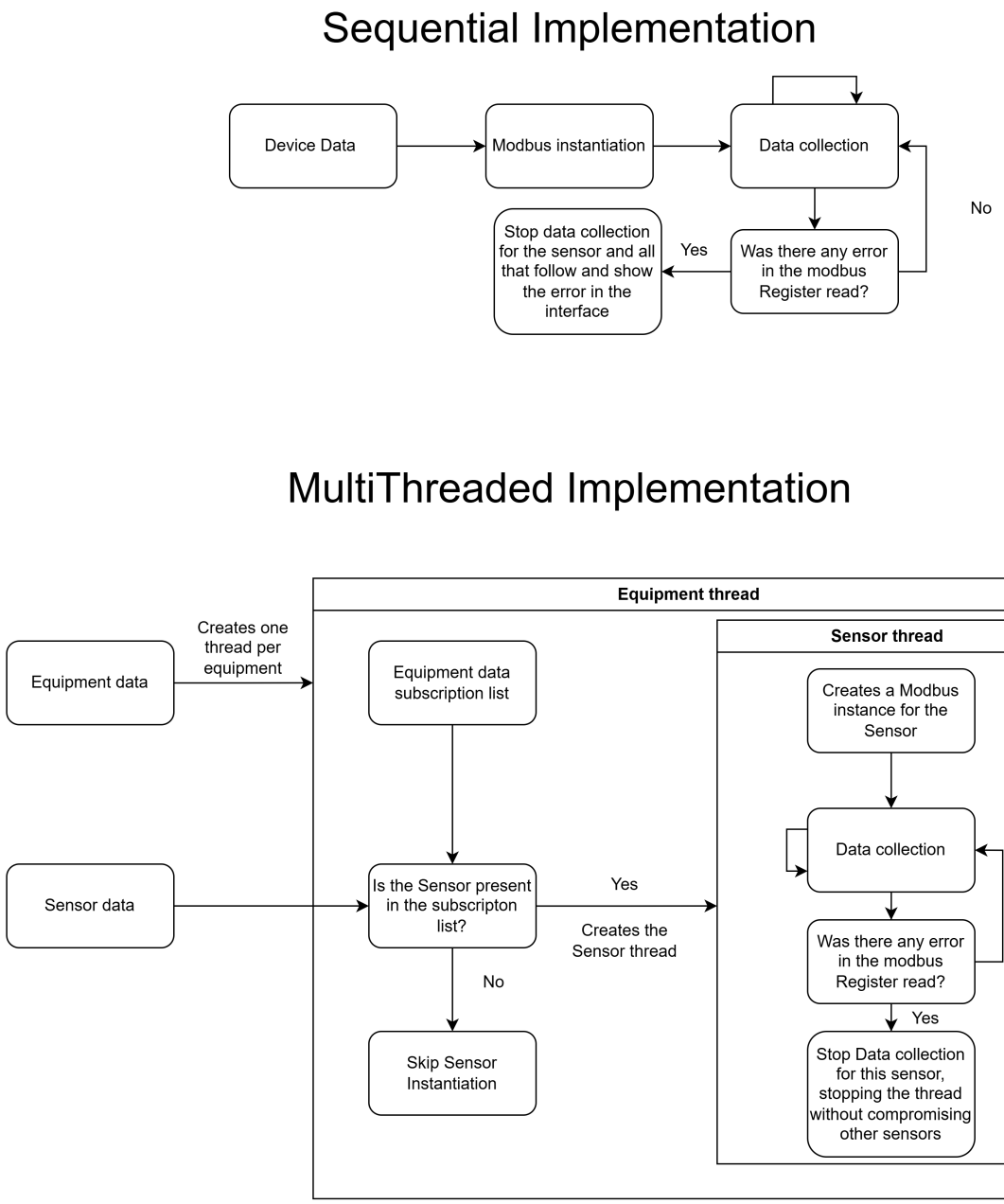

Figure 4.4: Comparison Between Sequential and Parallel Modbus Data Acquisition

This Figure 4.4 shows the complexity added with the multithreaded version. This transformation introduces substantial advantages. Most notably, system responsiveness is greatly improved. In the event that a single sensor encounters a fault, only the data associated with that sensor is impacted, while all other threads continue to operate normally. This modular failure isolation significantly enhances system robustness.

However, the adoption of parallelism also introduces new challenges. Most important among them is the increased computational load placed on the CPU due to the high number of threads, especially in deployments with a large number of sensors and equipments. This necessitates careful

resource management and may require tuning based on the hardware capabilities of the deployment environment.

4.3.2 Synchronization with the OPC-UA Server

After refactoring the data model, a critical issue observed in the original implementation, race conditions between the LabVIEW interface and the OPC-UA server was addressed. The lack of a proper synchronization mechanism frequently resulted in premature attempts by LabVIEW to send data to a server that had not yet finished initialization, causing data loss or crashes.

To mitigate this, a file-based synchronization method was introduced. Upon successful setup, the OPC-UA server generates a status file indicating readiness. The LabVIEW interface, prior to commencing any data acquisition, searches for the existence of this file. Once detected, the system proceeds with its regular data collection process. This method ensures orderly execution and prevents asynchronous startup failures.

In summary, the restructuring of the LabVIEW interface through the integration of improved XML data models, server synchronization mechanisms, and parallel execution—has laid a solid foundation for a scalable, reliable, and Industry 4.0 compliant data acquisition solution, ready for scalability and to be deployed in industrial factory floors, like the one present at IDEPA.

4.4 Development of a Middleware Program

As discussed in previous sections, a dedicated middleware application was developed to reinforce the overall system structure and improve communication between components. This middleware acts as an intermediary primarily bridging LabVIEW and the OPC-UA server but also serving as a centralized controller that abstracts and manages their interactions.

The middleware's first responsibility is to parse the system's XML configuration file and initiate a validation process for all defined Modbus registers. It sequentially reads each register, logging the status of every communication attempt. The results are then compiled into a diagnostic file that is displayed through the Backoffice interface, giving users a real-time overview of which Modbus channels are operating correctly and which are experiencing faults. This diagnostic report allows for quick identification and isolation of problematic sensors or equipment, thereby improving the maintainability and transparency of the system.

Once the Modbus channels have been validated, the middleware proceeds to launch the OPC-UA server. Importantly, it also assumes full control over the server's lifecycle. For example, if the user initiates a shutdown request , the middleware interprets the command and handles the server termination gracefully. This centralized control ensures that both LabVIEW and the OPC-UA server can function independently.

The main goal of implementing this middleware was to decouple the LabVIEW and OPC-UA components turning them into modular, separately manageable entities. In doing so, the system gains flexibility, improved fault tolerance, and simplified integration with additional services.

4.5 Backoffice Interface

As the previous solution, the new solution built upon the Backoffice existing application, designed as a centralized management and monitoring tool for configuration and runtime supervision. Implemented using a web technology stack, the Backoffice functions as an auxiliary middleware layer that interacts directly with the SOSD XML configuration. Its primary purpose is to provide a user-friendly interface that enhances control, flexibility, and system transparency for the main user.

One of the key functionalities of the Backoffice is to enable modifications to the SOSD XML configuration file, which serves as the core of the system's structural definition. Through a graphical user interface (GUI), authorized users can dynamically add or remove equipment, assign sensors, and adjust operational parameters without direct interaction with the underlying LabVIEW implementation. This decouples configuration management from application logic and aligns with Industry 4.0 (I4.0) principles, particularly the concept of the Administration Shell, where system components maintain a digital representation that can be updated independently of their runtime execution.

The Backoffice provides several core features that support system transparency and operational control:

- **System Configuration Overview:** A visual representation of the equipment-sensor hierarchy derived from the XML configuration file.
- **SOSD XML Management:** In-place editing of the configuration file via a GUI, allowing real-time updates and configuration validation.
- **Sensor Status Monitoring:** Display of sensor communication health and diagnostics based on middleware-generated reports.

The following Figure 4.5 shows the main page of the enhanced system:

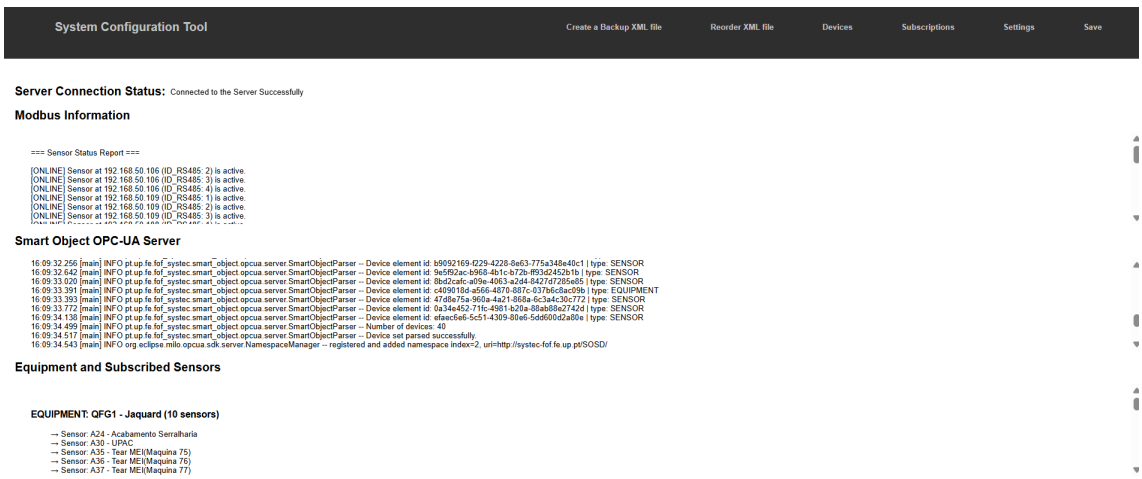

Figure 4.5: Backoffice Front Page

The interface presents a clear and hierarchical overview of all equipment and their associated sensors. This visualization allows operators to quickly understand the system's topology and verify whether individual devices are actively transmitting data. Furthermore, runtime issues such as OPC-UA node creation failures or server startup errors are automatically detected and displayed as alerts on the dashboard, streamlining troubleshooting efforts. In effect, the interface provides a live, readable representation of the SOSD model, functioning as a digital twin of the system's current state.

Users can modify the system configuration using form based inputs or built-in text editors. These modifications are saved to the SmartBox XML configuration file and become active after a system-wide restart, during which the OPC-UA server re-registers the updated components. This approach enables real-time adaptability and significantly reduces system downtime during integration of new equipment or sensors.

Additional user-oriented features were implemented to enhance usability and maintainability. These include options to:

- Create backups of the current configuration file.
- Reorder XML entries to improve display and readability within the GUI.
- Add, duplicate, or remove sensors and equipment via interactive tools with full operational stability.

This enhanced monitoring and configuration interface equips system operators with detailed visibility into both sensor-level data values and the structural health of connected components. As a result, the system achieves a higher degree of maintainability and responsiveness, essential for industrial applications.

Overall, the improvements introduced by the Backoffice interface support a modular, synchronized, and user-centric methodology. Modular data representations (e.g., Equipment and Sensor clusters), along with the decoupling of configuration and runtime logic, improve scalability and fault tolerance. Middleware-driven synchronization ensures robust coordination between LabVIEW and the OPC-UA server. The Backoffice complements these capabilities by offering comprehensive oversight and configurability, essential for seamless deployment and evolution in industrial environments. Furthermore, by strictly adhering to the SOSD information model and I4.0 component standards. It also maintains the core architecture represented with the figure 3.1 in the previous chapter 3 .

4.6 Implementations Results

The primary focus of this dissertation was to address and overcome the limitations identified in the previous implementation of the data model. In particular, the work aimed to enhance both the scalability and user-friendliness of the overall system architecture. To achieve these objectives, the following core improvements were made:

- **Implementation of mechanisms for OPC-UA server responsiveness and pipeline creation:** Modifications were made to hasten and stabilize the pipeline creation process and improve the server's ability to start and stop gracefully.

- **Restructuring of the LabVIEW interface:** The original implementation was restructured to improve performance, simplify code maintenance, and eliminate sequential bottlenecks in data acquisition, using parallelism.

- **Development of a dedicated middleware layer:** A new software component was introduced to serve as a bridge between the LabVIEW interface and the OPC-UA server, improving synchronization and modularity.

- **Optimization of the Backoffice interface:** The web-based GUI was enhanced to provide a more reliable and feature-complete user experience, offering better system visibility and interaction for maintainers.

Following the implementation of these improvements, the core architecture of the system underwent significant changes. While preserving the foundational principles and baseline functionality of the original solution, the updated architecture introduced new elements designed to improve system robustness and scalability. The revised architecture is illustrated in the following Figure 4.6 below.

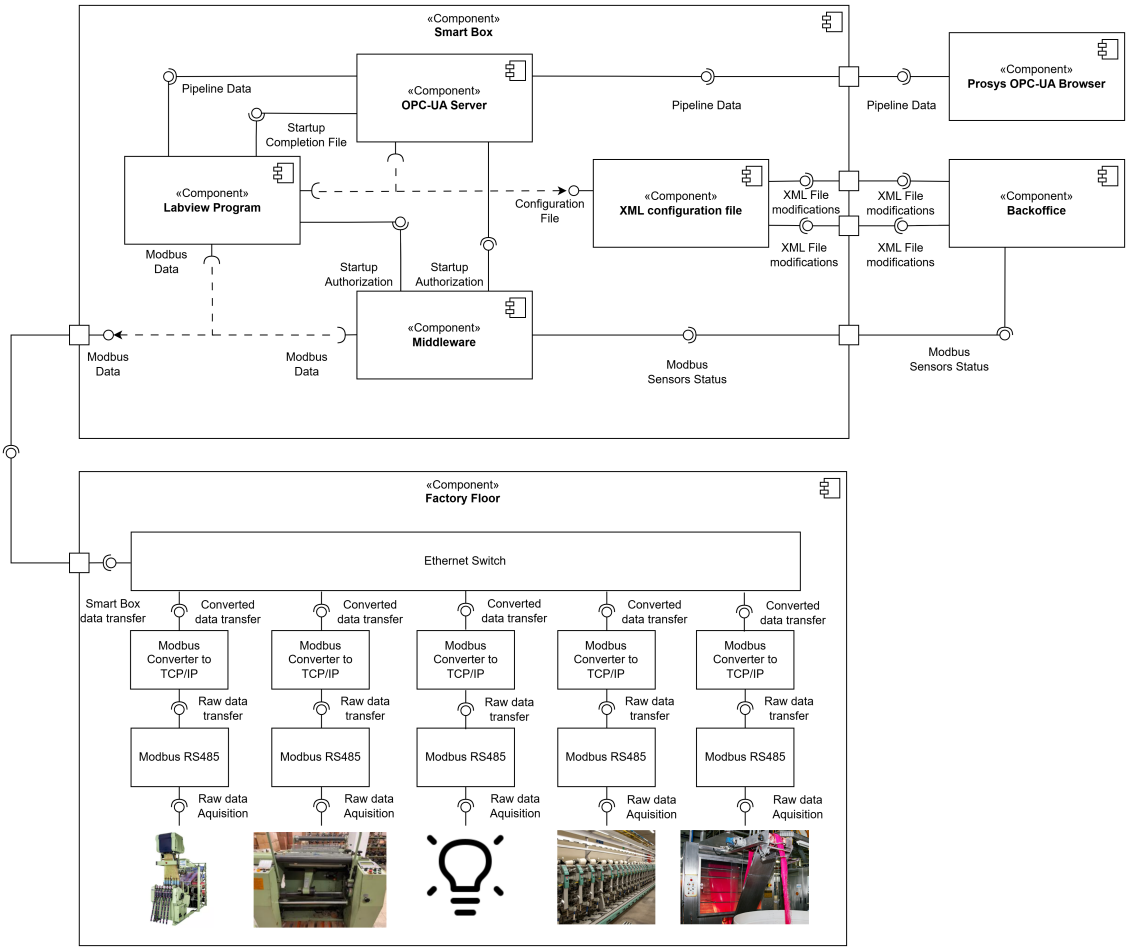

Figure 4.6: Main System Architecture

A comparison with the previous design reveals several key improvements. Most notably, the inclusion of the middleware program provides a well-defined communication channel between system components, effectively decoupling critical processes. This design removes several unwanted dependencies that had previously led to system instability and crashes. The overall communication flow between components was also enhanced, allowing for greater modularity, scalability, and ease of future upgrades.

Chapter 5

Testing and validation

To validate the effectiveness and scalability of the proposed solution, a comprehensive set of test cases was formulated. These tests were designed to simulate a wide range of operational scenarios with varying levels of system complexity. The primary objective of this testing is to validate the new architecture as a viable solution to implement on the IDEPA factory floor, and a comparison was made with the previous solution.

5.1 Test Scenarios

To thoroughly evaluate the system's behavior under different load conditions, four main categories of complexity were defined:

- **Low Complexity Scenario:** Minimal number of Equipments and sensors.
- **Medium Complexity Scenario (Baseline):** A moderate Equipments and sensors used as a reference point for comparison.
- **High Complexity Scenario:** Increased number of Equipments and sensors.
- **Stress Test Scenario:** Stress-testing conditions with maximum system load in terms of devices, variables, and acquisition rates.
- **Centralized Scenario:** This test serves as but rather to determine whether a centralized or distributed system configuration would produce different outcomes. This assessment aims to verify the adaptability of the solution for deployment in large-scale industrial environments such as IDEPA, where systems are typically distributed across the entire factory.

Each scenario was executed using both the original and the proposed architectures to provide a direct performance comparison. This methodology allowed for a consistent evaluation of improvements and potential trade-offs across various operating conditions.

5.2 Test Environment and Configuration

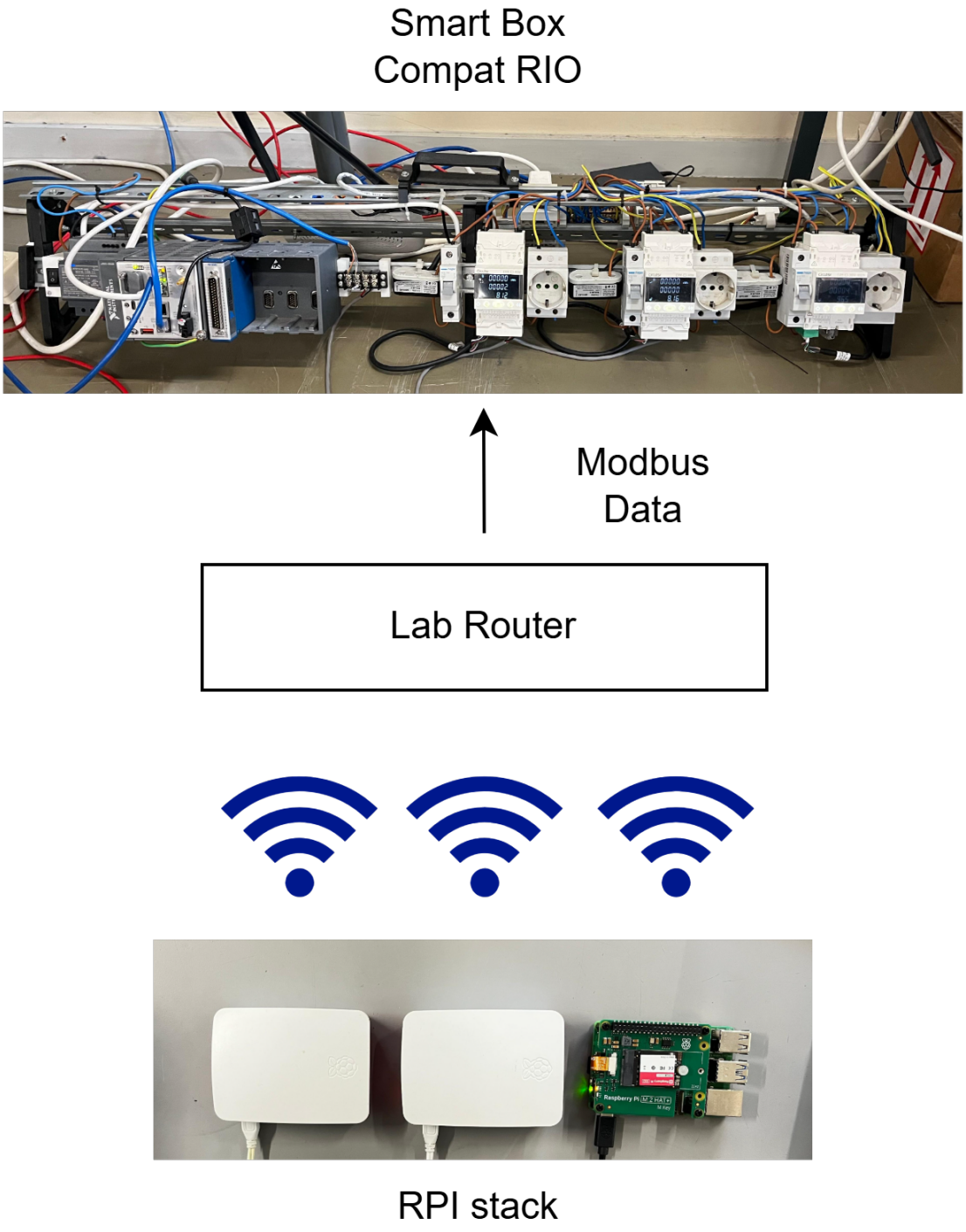

Figure 5.1: Test Setup

All tests were implemented on a CompactRIO 9040 system running the National Instruments Linux Real-Time operating environment. This platform was the platform that was needed and

implemented in the IDEPA factory. The factory is equipped with a range of industrial sensors used for monitoring various processes. These sensors are connected to a converter, which aggregates their outputs into logical units referred to as "equipment." Each piece of equipment represents a collection of sensors whose data, once processed by the converter, is transmitted through an Ethernet switch and subsequently routed to the Smart Box, as described in previous sections

The figure 5.1, represent the Test setup that was used, as the 3 raspberry Pis simulated the modbus Different equipments, each having 100 id's representing 100 sensors each, and then are read using the data collection tool operated by the compact rio

Critical system parameters were varied across test cases to assess the robustness and scalability of each architecture. These parameters included:

- **Number of Equipments:** Varying from 2 Equipments to 6, extending the network capabilities.

- **Number of Sensors per Equipment:** Ranging from 2 to 10 sensors.

- **Data Acquisition Frequency:** Ranging from low (2s) to high (0.5s) sampling rates.

These variations simulate realistic scenarios encountered in industrial applications, helping to identify bottlenecks and performance degradation trends.

5.3 Test Case Overview

The full list of the ten test cases is summarized in Table 5.1. The first three tests aim to establish a solid performance baseline under increasing acquisition frequencies. Test cases 4 through 6 maintain a low-complexity device configuration but vary the acquisition rate to evaluate system responsiveness. Test cases 7 through 9 increase both the number of equipments and sensors to examine scalability limits. Finally, Test 10 represents an open-ended stress test to probe the system's upper operational boundaries.

Table 5.1: Summary of Test Cases and Configuration Parameters

Test ID	Complexity Level	Equipments	Sensors/equipment	Acquisition Rate (s)
1	Medium - distributed	3	5	2
1.1	Medium - centralized	3	5	2
2	Medium - distributed	3	5	1
3	Medium - distributed	3	5	0.5
4	Low - distributed	2	2	2
4.1	Low - centralized	2	2	2
5	Low - distributed	2	2	1
6	Low - distributed	2	2	0.5
7	High - distributed	3	10	2
7.1	High - centralized	3	10	2
8	High - distributed	3	10	1
9	High - distributed	3	10	0.5
10	Extreme - distributed	6	10	2

Figure 5.2 illustrates the distinction between centralized and distributed sensor configurations. In the centralized architecture (shown on the right), all sensors are connected to single central Equipment. On the other hand, in the distributed configuration (shown on the left), sensors are deployed across multiple pieces of equipment throughout the factory floor. This different setups provide the same sensor load, one is 3 times 5 and the other is 15.

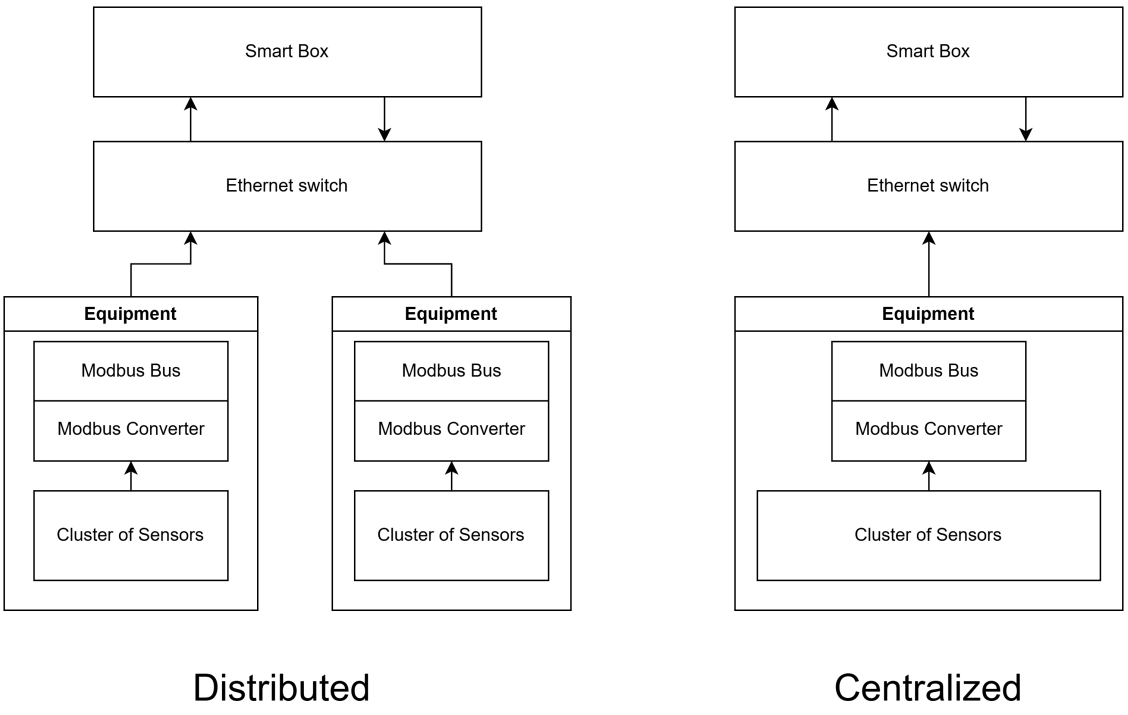

Figure 5.2: Distributed and Centralized Architectures

5.4 Evaluation Metrics

To objectively assess system performance and ensure comparability between the two architectures, a set of quantitative metrics was defined:

- **OPC UA Server Setup Time:** Time taken to initialize and expose the complete server namespace. This metric directly impacts system startup and reconfiguration times. This will be captured by analyzing the stdout produced by the OPC-UA server, which has timestamps that allow for efficient time measurement.
- **CPU Utilization:** Average and peak processor load during operation, indicating the computational efficiency of each architecture.
- **RAM Usage:** Memory footprint of the server during runtime; important for evaluating resource scaling with complexity. Both RAM and CPU are collected using a python script that captures the systems parameters and saves it to a file to be analysed later.
- **System Stability:** Number and frequency of runtime errors, data loss events, or system crashes. This metric provides insights into reliability under stress. This metric will be collect by analyzing the data that is collect along the testing.
- **Jitter:** Variability in the time difference between physical sensor data acquisition and the corresponding OPC UA node update. Lower jitter indicates better real-time performance and synchronization. A python script is responsible to collect the both timelines and calculate the jitter that then can be analyzed.

5.5 Tests Results

The goal of our test cases is:

1. To establish the limitations of the legacy system in terms of and scalability.
2. To quantify the benefits of the proposed architecture in mitigating these limitations under varying degrees of operational complexity.

Through detailed analysis of all the data collected for the required metrics, this chapter proves all the evidence that the solution is able to be implemented in high scale industrial systems, in this case it proved to be successful in the implementation in the industrial environment ate IDEPA.

5.5.1 Baseline Evaluation: Tests 1–3

To establish a reliable foundation for subsequent evaluations, a baseline test was executed. This comprises three distinct tests using a setup consisting of three equipment units, each with five sensors, for a total of fifteen sensor data streams. Each test scenario evaluated both the legacy (sequential) and the proposed (multithreaded) data acquisition architectures under increasing levels

of data acquisition frequency. These tests were designed to provide insight into system behavior in terms of CPU and RAM utilization, data acquisition latency (measured as jitter), and operational responsiveness (startup time).

All test scenarios were run for a duration of approximately 10 minutes, during which relevant performance data was captured for comparative analysis.

5.5.1.1 System Initialization and Responsiveness

Across all three test cases, system setup and initialization times were monitored. It was observed that in both architectures, the OPC-UA server initialization process took approximately the same duration, with a marginal improvement in the new architecture consistently faster by a few seconds. Though minor, this improvement was consistent throughout the testing sessions.

A more significant distinction was noted in the time taken for data acquisition to begin. In the legacy architecture, the system exhibited a consistent delay of one minute before data collection began. On the other hand, in the proposed architecture, data acquisition began almost instantaneously following initialization. This responsiveness is a result of the architecture's design, where thread instantiation occurs immediately after the middleware file handshake, bypassing any unnecessary delays.

This improvement in startup behavior is crucial for systems that demand rapid reconfiguration or frequent restart during industrial or laboratory operations, reducing downtime and increasing overall testing efficiency.

5.5.1.2 Test 1: Low-Frequency Acquisition (2 Seconds)

The first test case employed a data acquisition frequency of 2 seconds per sample, representing a low-stress scenario for both architectures. This test was designed to reveal baseline resource consumption and acquisition consistency.

Table 5.2: Comparison of Test 1: 2s Acquisition Interval

Metric	**Legacy Architecture**	**New Architecture**
CPU Usage Max (%)	99.00	100.00
CPU Usage Avg (%)	**8.90**	**8.16**
CPU Usage Min (%)	0.00	0.00
RAM Usage Max (%)	37.30	38.50
RAM Usage Avg (%)	**33.64**	**36.20**
RAM Usage Min (%)	20.80	18.40
Jitter Equipment 1 (s)	0.1813	0.0703
Jitter Equipment 2 (s)	0.1923	0.0699
Jitter Equipment 3 (s)	0.1798	0.0874

From Table 5.2, it can observed that CPU and RAM usage are comparable between the two architectures. Despite the proposed solution leveraging multiple threads, its average CPU and memory consumption remained within the same range as the legacy system. This result was somewhat unexpected, as multithreaded designs typically incur higher computational overhead. It is likely that process scheduling efficiency and background task balancing on the CompactRIO device contributed to this outcome.

However, a significant performance advantage of the proposed architecture is evident in the jitter values. The average jitter across all three equipment units is significantly lower in the new system, indicating more consistent timing between data samples. Reduced jitter is critical in applications where time-sensitive or deterministic behavior is required. The enhanced parallelism and reduced data serialization in the new system are likely contributors to this improvement.

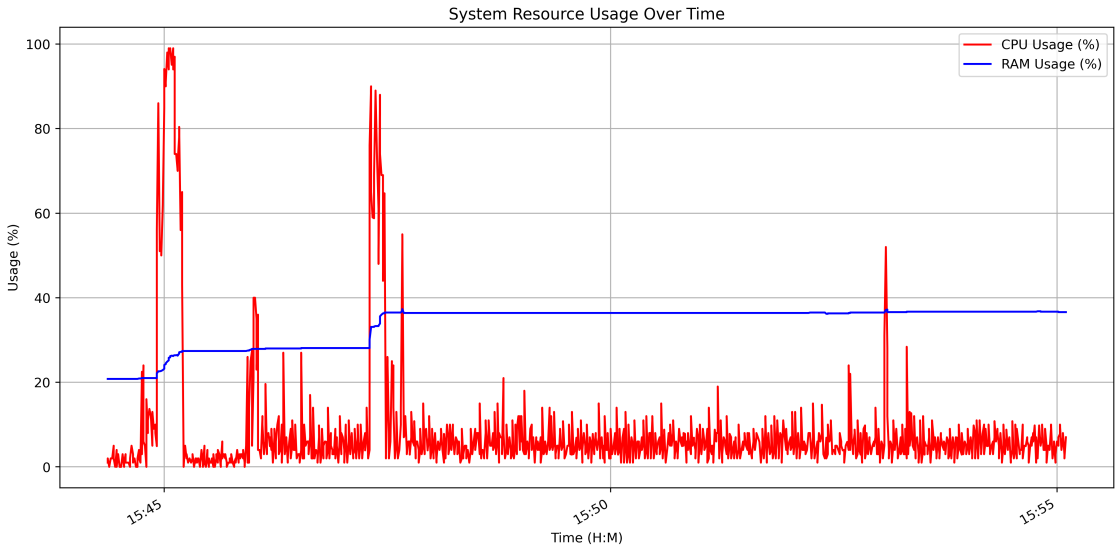

Figure 5.3: CPU and RAM Usage Over Time — Old Architecture (Test 1)

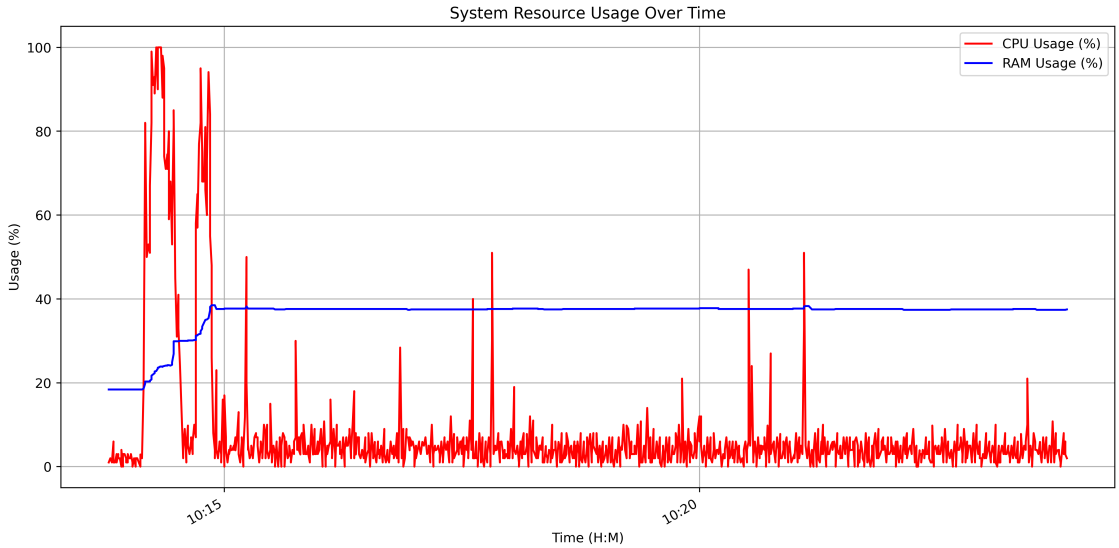

Figure 5.4: CPU and RAM Usage Over Time — New Architecture (Test 1)

The graphics (Figures 5.3 and 5.4) reveal clear differences in system behavior. The legacy system demonstrates three distinct resource usage spikes within the first three minutes: (1) OPC-UA server initialization, (2) Modbus data acquisition startup, and (3) jitter and performance logging initialization. In contrast, the new architecture shows only two spikes due to streamlined initialization and immediate execution of acquisition threads after the OPC-UA handshake, eliminating the need for serialized setup steps.

5.5.1.3 Centralized Test 1.1

This test serves as the baseline scenario for evaluating the subsequent experiments involving the Centralized System configuration. This baseline enables a more accurate assessment of how different sensor and equipment distribution strategies impact the overall system.

Table 5.3: Test 1.1: 2s Acquisition Interval

Metric	**Values**
CPU Usage Max (%)	100.00
CPU Usage Avg (%)	**7.98**
CPU Usage Min (%)	0.00
RAM Usage Max (%)	**35.60**
RAM Usage Avg (%)	33.90
RAM Usage Min (%)	18.40
Jitter Equipment 1 (s)	0.0563

As can be seen in the Table 5.3, the recorded values are nearly identical across the different configurations (Centralized and Distributed). The minor discrepancies observed can be attributed to background processes and internal system tasks running in an idle state. Consequently, it can be concluded that the distribution of sensors per equipment does not have a measurable effect on

the system's performance. This reinforces the robustness of the architecture, demonstrating its ability to maintain consistent behavior regardless of how sensors are grouped within individual equipment units. With this test (Test 1) we have the following main conclusions:

- Both architectures exhibit similar average CPU and RAM usage.
- The new architecture demonstrates significantly lower jitter.
- System responsiveness is enhanced in the new solution, reducing wait times.
- The solution that was developed is able to respond to centralized and Distributed forms of system configuration

5.5.1.4 Test 2: Medium-Frequency Acquisition (1 Second)

In the second test, the acquisition interval was halved to 1 second to assess performance under higher data throughput. Table 5.4 presents the results for computer resource usage by the solution.

Table 5.4: Performance Metrics for Test 2: 1s Acquisition Interval

Metric	**Legacy Architecture**	**New Architecture**
CPU Usage Max (%)	100.00	100.00
CPU Usage Avg (%)	**10.76**	**9.94**
CPU Usage Min (%)	0.00	0.00
RAM Usage Max (%)	39.90	40.20
RAM Usage Avg (%)	**37.48**	**37.80**
RAM Usage Min (%)	21.50	18.50
Jitter Equipment 1 (s)	0.1111	0.0440
Jitter Equipment 2 (s)	0.1814	0.0600
Jitter Equipment 3 (s)	0.2064	0.0475

As expected, the average CPU and RAM usage increased slightly due to the higher acquisition frequency. However, both systems remained well within operational limits. The new architecture continues to outperform the legacy system in terms of jitter, with more than 50% lower average variation across all equipment units. The following graphics prove the similarity in behavior of resource consumption of both solutions (Figures 5.6 and 5.5).

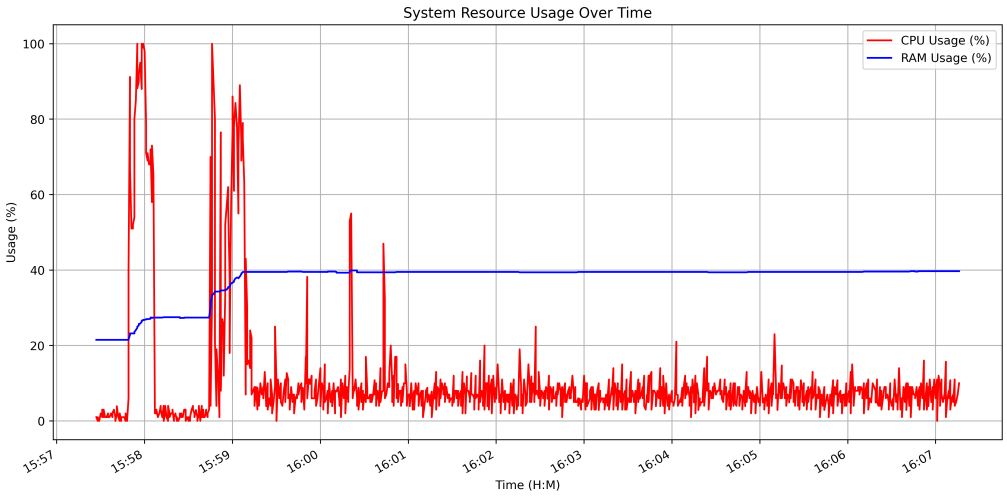

Figure 5.5: CPU and RAM Usage Over Time — Old Architecture (Test 2)

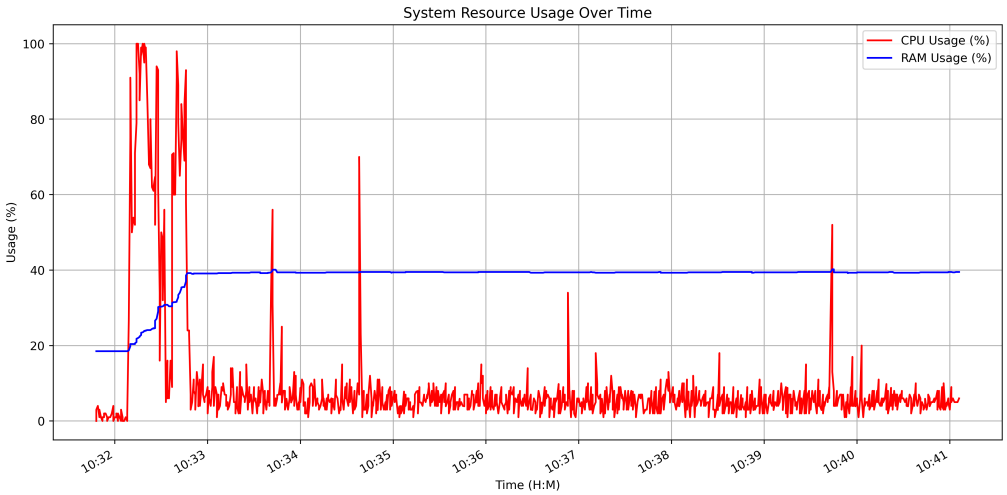

Figure 5.6: CPU and RAM Usage Over Time — New Architecture (Test 2)

With this test (Test 2) we can draw the following main conclusions:

- Both systems are capable of operating at 1-second intervals without data loss.
- Resource usage scales predictably with acquisition frequency.
- The new architecture maintains lower jitter and improved determinism.

5.5.1.5 Test 3: High-Frequency Acquisition (0.5 Seconds)

The final baseline test doubled the acquisition frequency once more, setting it to 0.5 seconds per sample. This test simulates high-throughput data environments where acquisition stress is maximized. Table 5.5 provides the results.

Table 5.5: Comparison of Test 3: 0.5s Acquisition Interval

Metric	**Legacy Architecture**	**New Architecture**
CPU Usage Max (%)	100.00	100.00
CPU Usage Avg (%)	**16.04**	**14.75**
CPU Usage Min (%)	0.00	0.00
RAM Usage Max (%)	37.60	36.20
RAM Usage Avg (%)	**35.96**	**34.68**
RAM Usage Min (%)	21.50	18.60
Jitter Equipment 1 (s)	0.0944	0.0453
Jitter Equipment 2 (s)	0.1529	0.0542
Jitter Equipment 3 (s)	0.0893	0.0619

Despite the increased load, both architectures maintained consistent data acquisition without observable data loss. CPU usage rose significantly compared to the lower-frequency tests, nearly doubling in both systems. Nevertheless, the new architecture continued to provide more consistent data delivery, with lower average jitter across all equipment. The impact this high scenario has in computer resources can be seen in the following Figures 5.8 and 5.7.

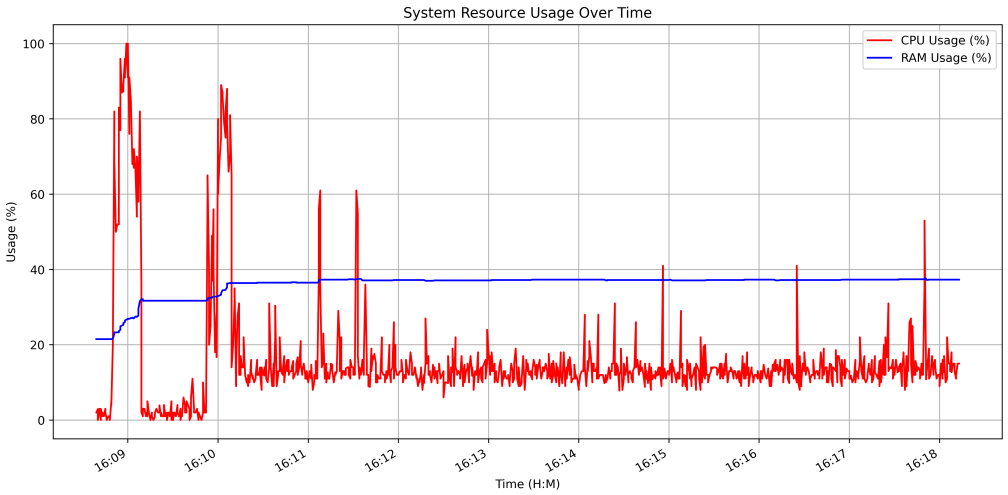

Figure 5.7: CPU and RAM Usage Over Time — Legacy Architecture (Test 3)

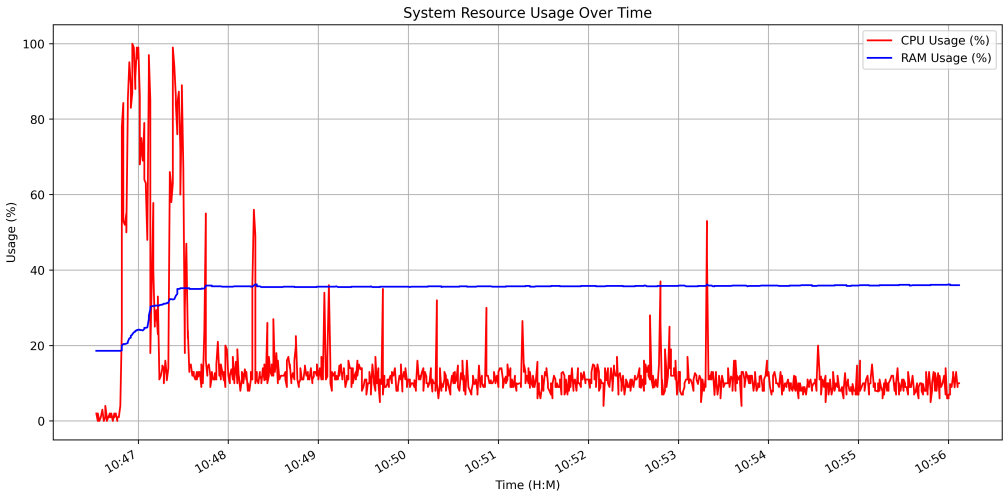

Figure 5.8: CPU and RAM Usage Over Time — New Architecture (Test 3)

We can conclude the following from Test 3:

- Both systems are capable of stable high-frequency data acquisition.
- CPU usage nearly doubles compared to the 2-second scenario, indicating higher processing demand.
- The proposed architecture maintains its jitter advantage even under stress.

5.5.1.6 Conclusions about Tests 1-3

These baseline tests confirm the capability of both the legacy and proposed architectures to manage medium-sized sensor arrays at varying acquisition rates. The following key takeaways were identified:

- The proposed architecture consistently outperformed the legacy system in jitter reduction, indicating superior timing accuracy and determinism.
- Contrary to initial expectations, the multithreaded design of the new architecture did not significantly increase CPU or RAM usage.
- Initialization delays in the legacy architecture present challenges in rapid deployment and iterative testing, whereas the proposed system offers immediate responsiveness.

The results from Tests 1 to 3 serve as a reference framework for subsequent evaluations under varying network conditions and complexity levels, demonstrating the robustness and scalability of the proposed solution.

5.5.2 Evaluation of Low Complexity Scenarios: Tests 4–6

This subsection presents the results and analysis of Tests 4, 5, and 6, which evaluate system behavior under low-complexity configurations involving two pieces of equipment, two sensors, and increasing data acquisition frequencies. The tests were conducted, as mentioned before, on a CompactRIO (cRIO) device. Both the legacy and the newly proposed architectures were assessed for comparison.

5.5.2.1 Test Setup

The previous solution required roughly 1 minute to begin data acquisition, primarily due to hard-coded delays in the LabVIEW interface. The initialization time of the OPC UA server remained consistent across all tests, at approximately 15–16 seconds for the legacy implementation and 12–13 seconds for the proposed architecture. This behavior is expected, as the variable parameter in these tests is the data acquisition frequency, not the number of sensors.

5.5.2.2 Performance Results for Test 4

Table 5.6: Comparison of Test 4: 2 Equipment, 2 Sensors, 2s Acquisition Interval

Metric	**Legacy Solution**	**Proposed Solution**
CPU Usage Max (%)	98.50	99.00
CPU Usage Avg (%)	**12.74**	**10.48**
CPU Usage Min (%)	1.50	2.00
RAM Usage Max (%)	32.30	32.90
RAM Usage Avg (%)	**30.59**	**29.87**
RAM Usage Min (%)	18.80	17.80
Jitter Equipment 1 (s)	0.2021	0.1892
Jitter Equipment 2 (s)	0.1530	0.1476

The results indicate that both architectures deliver similar jitter performance, with the proposed solution slightly outperforming the legacy system, as per the baseline scenarios. This improvement is likely due to the parallelization employed in the new architecture, contrasting with the serialized approach of the legacy implementation. RAM usage remains comparable, and small differences are attributed to background system processes and runtime variability.

5.5.2.3 Graphical Analysis

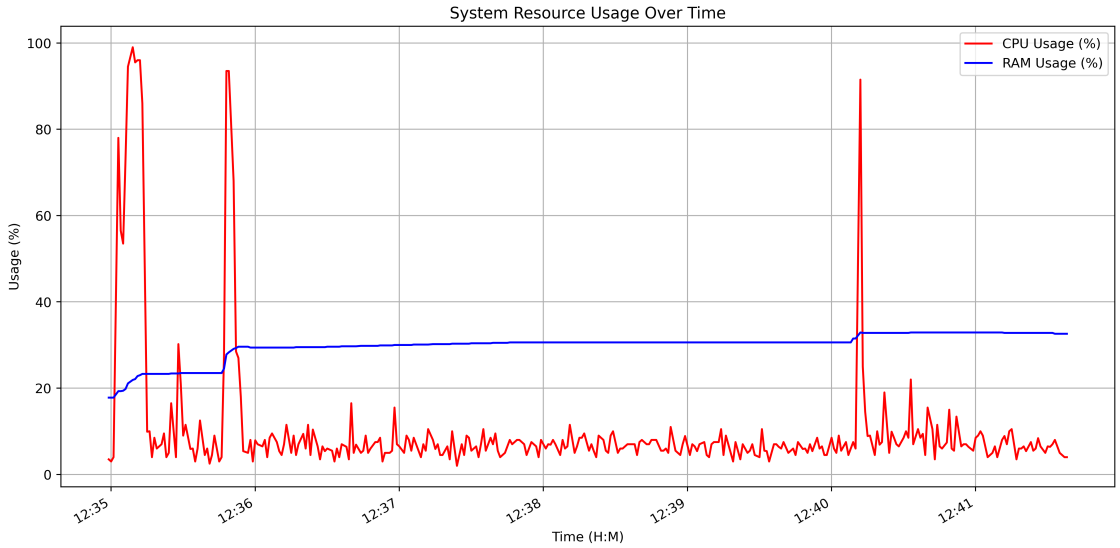

Figure 5.9: CPU and RAM Usage Over Time — New Architecture (Test 4)

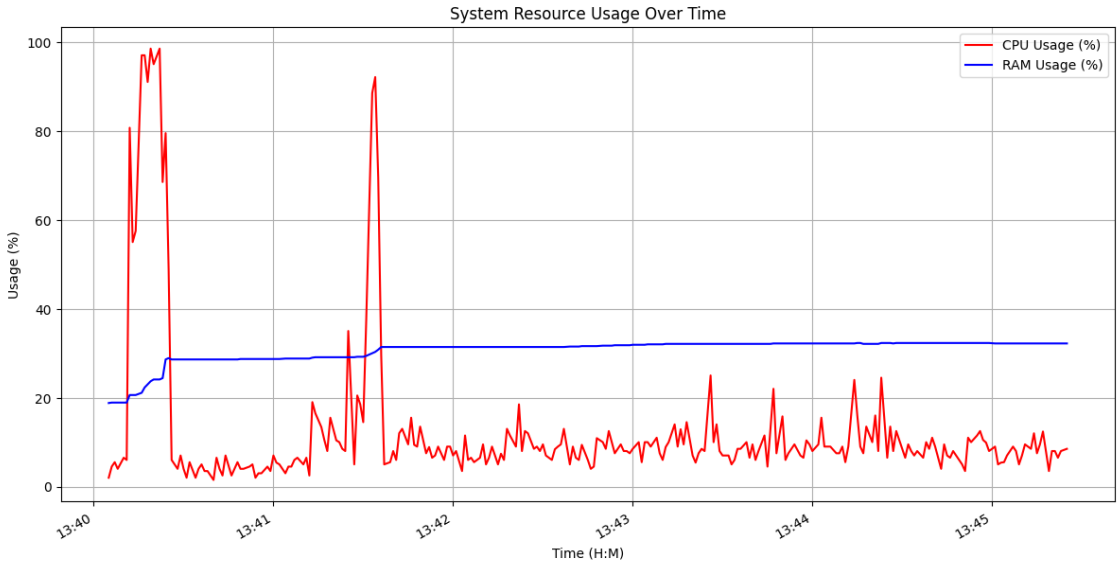

Figure 5.10: CPU and RAM Usage Over Time — Legacy Architecture (Test 4)

Figures 5.9 and 5.10 illustrate CPU and RAM utilization over time for both the proposed and legacy architectures. In both graphs, two distinct spikes in resource usage are observed.

The first spike is associated with the initialization of the OPC UA server, triggered by the LabVIEW interface. This spike occurs in both systems and reflects the multithreaded nature of the server, which increases computational load and memory consumption during startup.

The second spike corresponds to the initiation of the Modbus communication and data collection processes and testing data collection, also instantiated by LabVIEW. In the legacy architecture, this event occurs approximately one minute after execution begins, aligning with the previously mentioned hardcoded delay. In contrast, the proposed solution achieves the same initialization and data acquisition startup in significantly less time, demonstrating improved responsiveness.

Moreover, the RAM increase in the proposed architecture is more abrupt. This behavior is attributed to the parallelized implementation, which introduces higher instantaneous resource demands during concurrent task execution, particularly during the setup phase.

5.5.2.4 Performance Results for Tests 5 and 6

Table 5.7: Comparison of Tests 5 and 6 with Higher Acquisition Frequencies

Metrics	**Test 5**		**Test 6**	
	Legacy	New	Legacy	New
CPU Max (%)	98.50	100.00	100.00	99.00
CPU Avg (%)	**13.61**	**12.54**	**14.53**	**15.02**
CPU Min (%)	0.00	1.00	0.50	1.00
RAM Max (%)	35.20	32.30	33.90	31.40
RAM Avg (%)	**31.18**	**31.13**	**30.23**	**30.33**
RAM Min (%)	18.80	18.00	18.00	18.00
Jitter Equipment 1 (s)	0.1737	0.1304	0.1961	0.1523
Jitter Equipment 2 (s)	0.2006	0.1649	0.1288	0.1916

These tests confirm that both solutions scale effectively with increased data acquisition frequencies in low-complexity scenarios. CPU usage increases proportionally, as expected, while RAM usage remains stable across both implementations. The jitter remains within acceptable limits, with the proposed architecture again achieving slightly better performance due to its parallel execution model.

5.5.2.5 Conclusions about Tests 4-6

Two primary conclusions can be drawn from these tests:

- Both the legacy and proposed architectures demonstrate similar performance in terms of average CPU and RAM usage, with consistent peak activity.
- The proposed architecture offers a significant reduction in startup time for data acquisition, approximately 45 seconds faster, achieving improved responsiveness without compromising resource efficiency.

These results validate the capability of both systems to handle low-complexity data acquisition scenarios effectively. However, the proposed architecture presents clear advantages in startup

latency and jitter optimization, thereby laying the groundwork for superior scalability in more complex environments.

5.5.3 High Complexity Scenarios: Tests 7-9

Tests 7 to 9 were executed to validate the system's ability to manage increased complexity, specifically the monitoring and acquisition of data from 30 sensors distributed across three different equipment modules. These tests each lasted approximately 10 minutes.

5.5.3.1 Test 7: Baseline High Complexity Configuration

In this test, both the previous and proposed architectures were evaluated under the same configuration. The proposed architecture demonstrated reliable performance, successfully collecting and displaying data from all 30 sensors with minimal network latency. In contrast, the legacy solution encountered significant limitations. Due to the absence of a handshake mechanism between the OPC-UA server and the LabVIEW interface, the system lacked the ability to optimize server setup. This resulted in system failure during initialization, a previously mentioned limitation where the solution could not handle more than 20 sensors.

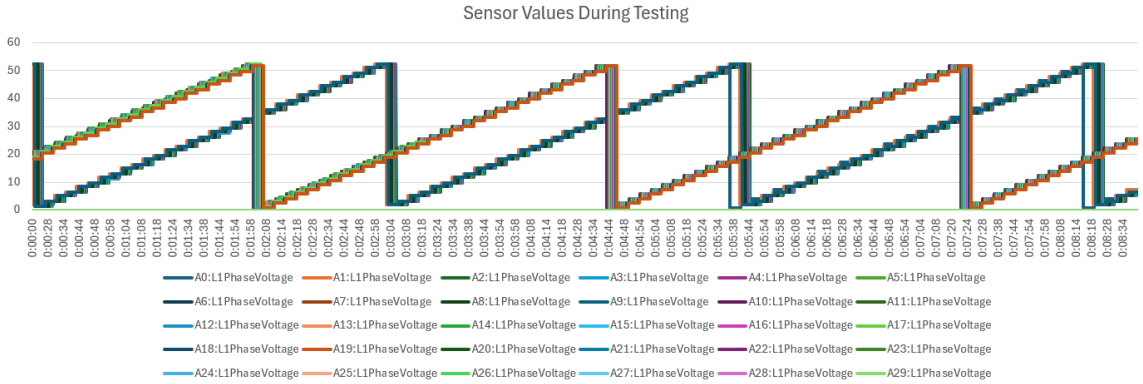

Figure 5.11: Sensor data acquired during the Test using the New architecture

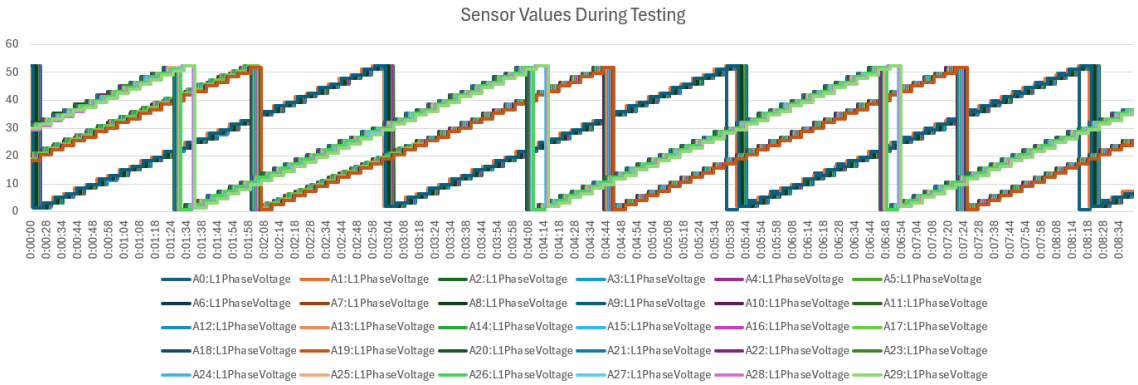

Figure 5.12: Sensor data acquired during the Test using previous architecture

As it can be observed in the Figure 5.11, the proposed solution maintained uninterrupted data flow of all the 30 sensors, as per the previous solution the Figure 5.12, shows that could only gather data from the first 20 sensors, this proves the scalability limitation of the previous architecture.

Resource usage data for the proposed solution during Test 7 is summarized below:

Table 5.8: Performance Metrics for Test 7

Metric	**Value**
CPU Usage Max (%)	100.00
CPU Usage Avg (%)	**12.79**
CPU Usage Min (%)	0.00
RAM Usage Max (%)	45.30
RAM Usage Avg (%)	**42.87**
RAM Usage Min (%)	20.70

Jitter analysis was also performed, with average jitter values recorded as follows:

- Equipment 1: 0.071164 seconds
- Equipment 2: 0.094439 seconds
- Equipment 3: 0.082088 seconds

These results show that, although the resource consumption was higher than in prior tests, the system operated within acceptable thresholds. Jitter remained stable, suggesting the network was not overloaded despite the increased sensor count.

5.5.3.2 Tests 8 and 9: Increased Acquisition Frequency

Tests 8 and 9 involved gradually increasing the data acquisition frequency. At a 1-second interval, the system remained stable and efficient, showing only marginal increases in CPU, RAM usage, and decrease in jitter.

Table 5.9: Comparison of Test 8 and Test 9 Performance

Metric	**Test 8**	**Test 9**
CPU Usage Max (%)	100.00	100.00
CPU Usage Avg (%)	**14.77**	**20.65**
CPU Usage Min (%)	0.00	0.00
RAM Usage Max (%)	42.50	42.50
RAM Usage Avg (%)	**40.36**	**40.26**
RAM Usage Min (%)	20.10	20.30
Jitter Equipment 1 (s)	0.0689	0.063089
Jitter Equipment 2 (s)	0.93057	0.0829
Jitter Equipment 3 (s)	0.0555	0.0863

However, when the acquisition frequency was increased to 0.5 seconds, the system encountered a critical issue after approximately 7 minutes. Data loss occurred due to Modbus server

disconnections, triggered by the excessive number of requests. The LabVIEW logs confirmed that the Modbus program could not handle the volume of concurrent threads, which points to a potential bottleneck in the LabVIEW Modbus layer rather than the overall system architecture. Instead of creating a thread for each equipment and sensor, creating a single thread per equipment will reduce overhead. However, this will compromise system safety, as a sensor failure will disrupt data acquisition for the entire equipment. Alternatively, improving the Modbus structure and optimizing the acquisition layer will reduce delays and prevent bus overload. Both approaches need proper testing, so its effectiveness is proven.

5.5.3.3 Conclusions about Test 7-9

After the conclusion of these tests the following insights were extracted:

- The previous solution is fundamentally limited to handling up to 20 sensors.
- The proposed solution scales effectively to 30 sensors with minor trade-offs in RAM and CPU usage.
- At acquisition frequencies below 1 second, the Modbus Labview implementation becomes a limiting factor, requiring optimization.

5.5.4 Stress Testing: Test 10

Test 10 was designed to push the architecture to its operational limits. The test utilized a configuration file specifying 60 sensors across 6 different equipment modules, with a 2-second acquisition rate, doubling the complexity of the previously mentioned tests.

5.5.4.1 Testbed Configuration

Since the IDEPA factory floor did not have 60 sensors sending data to the Modbus Bus, the system was emulated using three Raspberry Pi 4 units and one Raspberry Pi 5. These devices hosted the Modbus servers and clients, and a custom XML configuration was deployed.

5.5.4.2 Test Results on Raspberry Pi 5

The first deployment was carried out on a Raspberry Pi 5 as per the Figure 5.13, a more modern computing platform than the CompactRIO, albeit not optimized for industrial control. Despite this, the solution worked flawlessly, with no interruption in data collection. CPU and RAM usage were comparable to the CompactRIO in absolute terms, but slightly lower in percentage due to the more capable hardware. The following Figure 5.13 shows all the system components and how they are arranged during the testing.

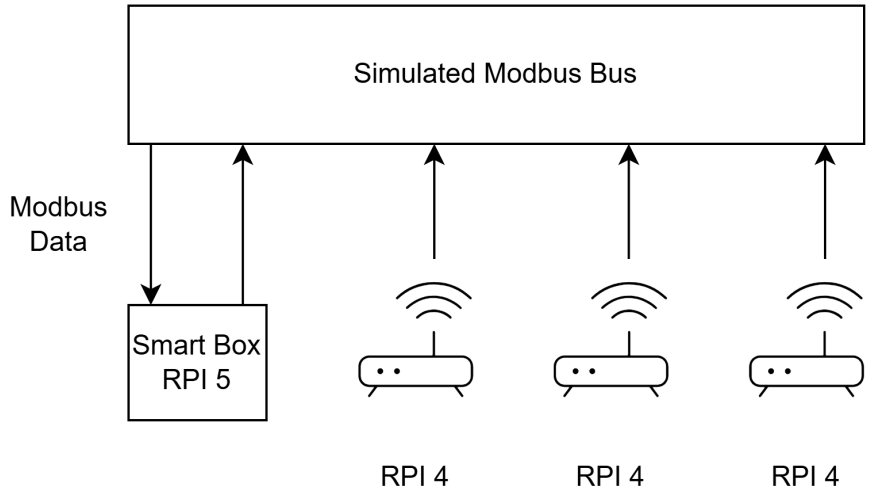

Figure 5.13: System setup for Test 10 using Raspberry Pi 4 and Raspberry Pi 5

Attempts to scale beyond 60 sensors failed due to thread allocation limitations, which was expected based on earlier architectural analysis.

5.5.4.3 Test Results on CompactRIO

The second phase of Test 10 involved re-deploying the architecture to the CompactRIO testbed present in FEUP as shows the Figure 5.14. This environment closely simulates the actual IDEPA factory floor. The results on the compact Rio were more successful, the system was capable of capturing 85 sensors without any issues, in contrast with the previously mentioned 60 on the Raspberry Pi 5. This is the case because the CompactRio is built for industrial setups and data processing, which the Raspberry is not. Attempts to move to the 86th sensor proved unsuccessful, as the system wasn't able to acquire data for that sensor, proving a bottleneck, but data from the previous 85 sensors would still flow as normal.

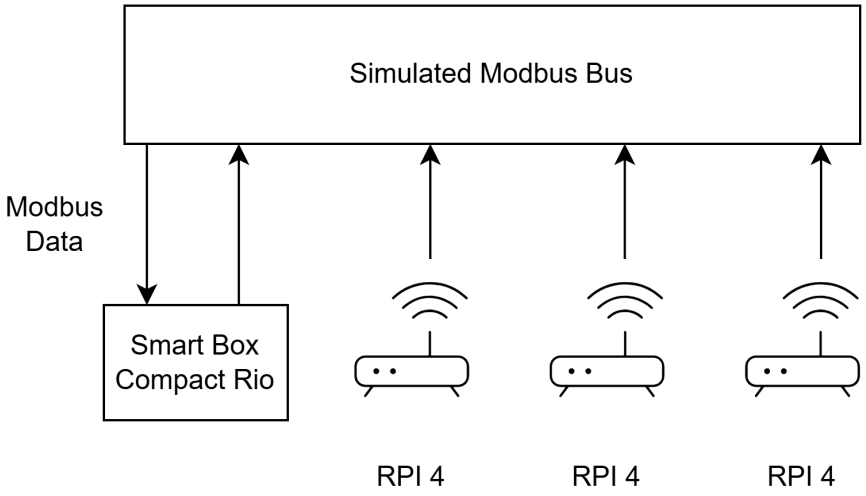

Figure 5.14: System setup for Test 10 using Raspberry Pi 4 and Compact Rio.

5.5.4.4 Conclusion about Test 10

The stress test confirmed the resilience and adaptability of the proposed architecture. The main conclusions are the following:

- The new system can manage 60 sensors in a device that is not ready to handle factory floor operations.
- The new system is able to acquire data from 85 sensors in the Compact RIO proving its scalability.
- It can operate reliably on both industrial-grade and consumer-grade hardware.
- Thread allocation and Modbus handling are the primary bottlenecks for further scalability, especially under high-frequency conditions.

5.5.5 Backoffice implementation

The backoffice tool developed as part of this project has proven to be a highly useful component during the testing phase. While it has not yet been validated in an industrial environment an important direction for future work it has demonstrated significant improvements in laboratory settings.

All intended functionalities were successfully tested in a controlled lab environment. For example, a process that previously took approximately 10 minutes can now be completed in just 1 minute. Furthermore, operations that once required manual deletion taking around 2 minutes can now be performed with a single click, reducing the time to approximately 1 second.

These results indicate the effectiveness and efficiency of the tool in non-industrial scenarios, reinforcing its value for future deployment and broader validation in real-world conditions.

5.5.6 Main test conclusions

The results obtained from Test 4.1 confirmed the expected behavior: the system, configured with 4 sensors and a single equipment unit, performed equivalently to its distributed architecture counterpart. A similar outcome was observed in Test 7.1, further validating the solution's consistency across different configurations.

These results reinforce the conclusion that the proposed architecture is both scalable and robust. It demonstrates that the system can reliably perform its intended functions without requiring changes to the existing factory layout, thus supporting seamless deployment in varied industrial setups.

Chapter 6

Results and discussion

This chapter presents and interprets the key findings obtained through comprehensive testing and evaluation of the proposed monitoring system. The results are analyzed from an industrial deployment perspective, with a particular focus on scalability, operational stability, system efficiency, and user experience. Emphasis is placed on the system's practical readiness for deployment in real-world manufacturing environments, such as the IDEPA factory floor. Additionally, the research questions posed in Chapter 1 are revisited and addressed based on these findings.

6.1 Performance Evaluation

Testing confirmed that the proposed system offers significant enhancements over the baseline implementation, particularly in terms of scalability, fault isolation, and user operability, key requirements for modern industrial control systems.

A core architectural change was the transition from a sequential to a multithreaded design. Contrary to initial expectations, the multithreaded implementation did not incur excessive computational overhead. The legacy system, operating sequentially, polled sensors linearly, resulting in a single point of failure, where one unresponsive sensor could stall the entire acquisition process. The newly implemented parallel architecture isolates such failures, ensuring uninterrupted data flow from remaining sensors, thus improving system reliability and robustness.

Tests, as detailed in Chapter 5, were conducted under varying loads, acquisition rates, and sensor configurations using the CompactRIO 9040 hardware—the same controller deployed on the IDEPA factory floor. These scenarios simulated conditions commonly found in medium-sized industrial environments. Performance metrics revealed that both the legacy and new architectures exhibited comparable CPU and memory footprints under most operating conditions. However, the new architecture significantly reduced acquisition jitter, thereby enhancing temporal precision and consistency of data streams.

These results indicate that the proposed solution is well-suited for deployment across industrial sites similar in scope to IDEPA, especially where distributed equipment layouts and real-time reliability are critical.

6.2 Scalability and Limitations

The system demonstrated reliable scalability in controlled test environments, consistently supporting up to 85 sensors during stress tests. This capability exceeds the sensor handling capacity of the previous solution and positions the architecture for broader industrial applicability.

However, limitations emerged when scaling beyond this threshold. During system initialization with over 85 sensors configured via the XML interface, startup failures were observed. These issues were traced to the excessive number of threads created, which exceeded system resource limits. This behavior highlights a practical ceiling on vertical scalability when using the current threading model.

Additionally, during high-frequency acquisition scenarios (0.5-second intervals) with 30 sensors, data loss occurred after sustained operation (approximately seven minutes). This suggests a throughput ceiling beyond which the system cannot guarantee consistent data integrity, signaling the need for further optimization in high-load environments.

6.3 System Evaluation: Strengths and Weaknesses

Advantages

- **Fault Isolation and Operational Continuity:** The multithreaded approach ensures that sensor failures do not propagate system-wide, which is essential for real-time industrial monitoring.

- **Robust Initialization Mechanism:** A handshake protocol between LabVIEW and the OPC-UA server ensures orderly and reliable system startup, reducing risk of boot-time errors.

- **Streamlined Configuration Management:** The Backoffice module offers centralized configuration through XML files, facilitating deployment and maintenance across multiple factory installations.

- **Real-Time Monitoring Capability:** Integration with the OPC-UA Prosys browser provides immediate visualization and status tracking, crucial for plant-floor diagnostics and control.

Disadvantages

- **Legacy Technology Stack:** Dependencies on older hardware and software components may limit future maintainability and performance scalability.

- **Higher Computational Load:** Although manageable under current use cases, the multithreaded model introduces additional CPU overhead, which may constrain deployment on lower-specification systems.

6.4 Research Questions Revisited

Based on the results and practical insights gained, the research questions posed in Chapter 1 can now be conclusively addressed:

1. **How can wireless communication technologies such as Bluetooth CAN Bus, and LTE be effectively integrated for remote monitoring and control of machines in industrial environments?**

   As discussed in Chapter 2, these technologies are highly relevant for Industry 4.0 applications. However, this implementation opted for wired communication between factory-floor devices and the CompactRIO 9040 controller to maximize reliability and minimize latency. This decision aligns with the operational demands of industrial monitoring, where deterministic communication is prioritized over flexibility.

2. **How can the proposed system be designed to be scalable and interoperable with existing industrial automation systems and IoT platforms?**

   Scalability and interoperability were core objectives of the design. The architecture supports a sensor count triple that of the legacy system, with mechanisms in place to handle partial failures and simplify expansion. Compatibility with the OPC-UA standard ensures seamless integration with modern SCADA systems and IoT platforms.

3. **What are the key design principles for developing a user-friendly interface for remote monitoring and control of industrial machines?**

   The proposed system emphasizes usability through the introduction of the Backoffice module, real-time visualization tools, and well-documented configuration protocols. These elements simplify deployment, facilitate remote supervision, and reduce the technical barrier for operators and maintenance personnel.

6.5 MQTT Integration

While MQTT was initially considered as a candidate protocol, it was ultimately excluded due to architectural and performance-related constraints, as outlined in Chapter 4. The LabVIEW environment requires data handling via defined pipelines, and layering MQTT on top would have introduced complexity and latency without delivering proportional benefits. Instead, a direct synchronization mechanism between LabVIEW and the OPC-UA server was implemented. This approach proved both simpler and more reliable, effectively eliminating the race conditions observed in previous versions.

6.6 Conclusions

The evaluation demonstrates that the proposed system meets the stringent requirements for deployment in industrial environments. It delivers enhanced scalability, greater operational resilience, and improved user interaction extending the reach of the already developed solution and fixing some problems to. While certain limitations were identified such as threading constraints and reliance on legacy components—they do not impede the immediate viability of the solution for deployment on medium-scale factory floors such as IDEPA.

The architecture provides a solid and extensible foundation for industrial monitoring systems, with the potential for further enhancements through codebase optimization and support for additional communication protocols. In its current form, the system is ready for production use and represents a meaningful advancement in remote equipment monitoring within the industrial automation landscape.

Chapter 7

Conclusion and Future Work

The integration of remote monitoring and control systems into industrial environments is a cornerstone of realizing the full potential of Industry 4.0. By adopting scalable communication protocols and intelligent digital interfaces, manufacturers can significantly enhance operational efficiency, minimize system downtime, and support data-driven decision-making in real time. This work contributes directly to these objectives by addressing the limitations found in legacy industrial systems—specifically targeting improvements in scalability, usability, and synchronism.

Through an in-depth analysis and redesign of the existing system deployed at IDEPA, several critical limitations were identified and mitigated. A key enhancement proposed was the integration of a lightweight MQTT server to complement the existing OPC-UA communication framework. While OPC-UA offers strong data modeling capabilities, it introduces synchronization complexity and limited performance under high sensor loads. MQTT, with its publish-subscribe model, provides a scalable and efficient alternative that simplifies data flow and supports greater concurrency. Although full MQTT implementation was not completed within this work, its value was clearly established, and its future integration is well justified.

In parallel, significant effort was directed toward improving the GUI, making it more accessible, intuitive, and informative. The updated interface addresses previous shortcomings by incorporating clear system status indicators, improved error notifications, and real-time visibility into sensor operations. These changes empower operators to interact confidently with the system and perform critical maintenance and configuration tasks with minimal disruption.

Furthermore, the proposed middleware successfully established synchronization between system components, reducing race conditions and improving startup reliability. The redesigned architecture leverages parallelized Modbus data acquisition and modular pipeline creation, significantly enhancing system scalability and fault tolerance.

In summary, this work validated the initial hypothesis by:

- Evaluating the integration of MQTT and demonstrating that this is a viable future enhancement;
- Redesigning the data model for improved modularity and parallelism, demonstrating that this has an impact in the scalability of the system;

- Successfully developing and deploying a comprehensive, user-friendly GUI, providing real-time visibility and enabling effective monitoring and control of the system.

Future Work

While this dissertation lays a strong foundation for scalable and robust industrial monitoring systems, several improvements are planned to ensure long-term sustainability and adaptability:

- **Hardware Modernization:** The current solution is built on the CompactRIO 9040 platform. For better performance and supportability, future iterations should consider migrating to a more modern industrial controller, depending on processing and I/O requirements.

- **Full Migration to MQTT Architecture:** The existing hybrid communication model should be replaced with a fully MQTT-based architecture. This transition will streamline the messaging workflow, reduce overhead, and enhance scalability. An MQTT broker could be integrated with the middleware to manage data distribution.

- **Replacement of LabVIEW with a Custom Data Acquisition Layer:** Due to LabVIEW's steep learning curve, licensing limitations, and integration challenges, future work should focus on developing a dedicated data acquisition application using Python, java, or C++, which would offer greater flexibility, maintainability, and open-source compatibility.

- **Refactoring and Elimination of Legacy Code:** A complete rewrite of the software stack is necessary to eliminate obsolete code, improve maintainability, and modernize the XML parsing logic used in pipeline configuration. The OPC-UA server (if retained) should also undergo major refactoring or be replaced with a lightweight MQTT-compatible data publisher.

Final Remarks

In conclusion, this dissertation demonstrates that the modernization of remote monitoring and control systems in industrial environments is not only achievable but also essential for sustainable manufacturing growth. By enhancing key architectural components' middleware, communication protocols, and user interface the solution developed herein achieves improved scalability, robustness, and usability.

The outcomes of this work directly support the ongoing digital transformation efforts at IDEPA and can be generalized to similar factory environments. More importantly, the modular and extensible design ensures that the solution is future-proof and adaptable to ongoing innovations in the industrial IoT landscape.

By continuing this trajectory (embracing modern hardware, eliminating legacy dependencies, and integrating lightweight protocols and advanced software tooling) future iterations of the system can serve as a model for next-generation industrial digitalization initiatives, fully aligned with the principles of RAMI 4.0 and Industry 4.0 at large.

In addition to the work described in this dissertation, two papers were also developed for the SUSTECH and ICINCO conferences. The paper submitted to SUSTECH has been accepted and presented, while the ICINCO submission is still under review.

References

[1] PRODUTECH, *D2.2 - modelos de dados - v1.3*, 2023.

[2] M. A. Pisching, M. A. Pessoa, F. Junqueira, D. J. dos Santos Filho, and P. E. Miyagi, "An architecture based on rami 4.0 to discover equipment to process operations required by products," *Computers and Industrial Engineering*, vol. 125, pp. 574–591, Nov. 2018, ISSN: 03608352. DOI: 10.1016/j.cie.2017.12.029.

[3] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (iiot): An analysis framework," *Computers in Industry*, vol. 101, pp. 1–12, Oct. 2018, ISSN: 01663615. DOI: 10.1016/j.compind.2018.04.015.

[4] L. Danys, I. Zolotova, D. Romero, *et al.*, "Visible light communication and localization: A study on tracking solutions for industry 4.0 and the operator 4.0," *Journal of Manufacturing Systems*, vol. 64, pp. 535–545, Jul. 2022, ISSN: 02786125. DOI: 10.1016/j.jmsy.2022.07.011.

[5] H. Xu, W. Yu, D. Griffith, and N. Golmie, *A survey on industrial internet of things: A cyber-physical systems perspective*, 2018. DOI: 10.1109/ACCESS.2018.2884906.

[6] J. Morgan, M. Halton, Y. Qiao, and J. G. Breslin, *Industry 4.0 smart reconfigurable manufacturing machines*, Apr. 2021. DOI: 10.1016/j.jmsy.2021.03.001.

[7] D. Witczak and S. Szymoniak, *Review of monitoring and control systems based on internet of things*, Oct. 2024. DOI: 10.3390/app14198943.

[8] G. Fortino, C. Savaglio, G. Spezzano, and M. Zhou, "Internet of things as system of systems: A review of methodologies, frameworks, platforms, and tools," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, pp. 223–236, 1 Jan. 2021, ISSN: 21682232. DOI: 10.1109/TSMC.2020.3042898.

[9] K. S. Hongyu Pei Breivold, *Internet of Things for Industrial Automation – Challenges and Technical Solutions*. Conference Publishing Services, IEEE Computer Society, 2015, ISBN: 9781509002146.

[10] S. Malhão, R. Dionísio, and P. Torres, *Industrial IoT Smartbox for the Shop Floor*. IEEE, 2019, ISBN: 9781728136370.

[11] R. A. Luchian, G. Stamatescu, I. Stamatescu, I. Fagarasan, and D. Popescu, "Iiot decentralized system monitoring for smart industry applications," in *2021 29th Mediterranean Conference on Control and Automation, MED 2021*, Institute of Electrical and Electronics Engineers Inc., Jun. 2021, pp. 1161–1166, ISBN: 9781665422581. DOI: 10.1109/MED51440.2021.9480341.

[12] X. Li, S. Zhang, P. Jiang, M. Deng, X. V. Wang, and C. Yin, "Knowledge graph based opc ua information model automatic construction method for heterogeneous devices integration," *Robotics and Computer-Integrated Manufacturing*, vol. 88, Aug. 2024, ISSN: 07365845. DOI: 10.1016/j.rcim.2024.102736.

[13] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, pp. 17–27, 1 Mar. 2017, ISSN: 19324529. DOI: 10.1109/MIE.2017.2649104.

[14] P. Torres, R. Dionísio, S. Malhão, L. Neto, and G. Gonçalves, "Machinery retrofiting for industry 4.0," *Lecture Notes in Mechanical Engineering*, pp. 213–220, 2022, ISSN: 21954364. DOI: 10.1007/978-3-030-79168-1_20/FIGURES/5. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-79168-1_20.

[15] C. Patel and N. Doshi, ""a novel mqtt security framework in generic iot model"," *Procedia Computer Science*, vol. 171, pp. 1399–1408, Jan. 2020, ISSN: 1877-0509. DOI: 10.1016/J.PROCS.2020.04.150.

[16] S. Profanter, A. Tekat, K. Dorofeev, M. Rickert, and A. Knoll, *Opc ua versus ros, dds, and mqtt: Performance evaluation of industry 4.0 protocols*. [Online]. Available: http://download.prismtech.com/docs/Vortex/html/ospl/DDSTutorial/qos..

[17] A. Claassen, S. Rohjans, S. Comsoc, and P. Lehnhoff, *Application of the opc ua for the smart grid*.

[18] M. Noor-A-Rahim, J. John, F. Firyaguna, *et al.*, "Wireless communications for smart manufacturing and industrial iot: Existing technologies, 5g and beyond," *Sensors*, vol. 23, 1 Jan. 2023, ISSN: 14248220. DOI: 10.3390/s23010073.

[19] G. Spencer, F. Mateus, P. Torres, R. Dionísio, and R. Martins, "Design of can bus communication interfaces for forestry machines," *Computers*, vol. 10, 11 Nov. 2021, ISSN: 2073431X. DOI: 10.3390/computers10110144.

[20] G. Spencer and P. M. Torres, "New can bus communication modules for digitizing forest machines functionalities in the context of forestry 4.0," *IEEE Access*, vol. 11, pp. 9058–9066, 2023, ISSN: 21693536. DOI: 10.1109/ACCESS.2022.3232286.

[21] X. Ren, C. Fu, T. Wang, and S. Jia, "Can bus network design based on bluetooth technology," in *Proceedings - International Conference on Electrical and Control Engineering, ICECE 2010*, 2010, pp. 560–564, ISBN: 9780769540313. DOI: 10.1109/iCECE.2010.144.

[22] L. neto Gil gonçalves, P. Torres, R. Dionisio, and S. Malhão, *An industry 4.0 self description information model for software components contained in the administration shell*.

[23] L. Neto, J. Reis, R. Silva, and G. Goncalves, "Sensor selcomp, a smart component for the industrial sensor cloud of the future," in *2017 IEEE International Conference on Industrial Technology (ICIT)*, IEEE, Mar. 2017, pp. 1256–1261, ISBN: 978-1-5090-5320-9. DOI: 10.1109/ICIT.2017.7915543. [Online]. Available: http://ieeexplore.ieee.org/document/7915543/.

[24] L. Neto, G. Gonçalves, P. Torres, and R. Dionísio, "On the development of a component model for the realization of industry 4.0," in *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, vol. 1, 2020, pp. 481–486. DOI: 10.1109/ICPS48405.2020.9274789.