

الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria

Ministry of Higher Education
and Scientific Research

University of Akli Mohand Oulhadj - Bouira -

Tasdawit Akli Muhend Ulhağ - Tubirett -

Faculty of Applied Sciences



وزارة التعليم العالي والبحث العلمي

جامعة أكلي محمد أولحاج

- البويرة -

كلية العلوم التطبيقية

Master's Thesis

Submitted to:

Department of: Electrical Engineering

Field: Science and Technology

Specialization: Industrial Electrotechnics

Submitted by:

Feddi Zakaria

and

Bouzid Ayoub

Title

**Automation of an intelligent SCADA system for
production line monitoring and supervision**

Defended on: 24/06 /2026

Before the jury composed of:

Mr. Y. Bensafia

University of Bouira

Chairperson

Mr. S. Issouani

University of Bouira

Examiner

Mr. S. Ameziane

University of Bouira

Supervisor

Academic Year: 2025–2026



ACKNOWLEDGEMENTS

*First and foremost, we praise **ALLAH** for granting us the strength, patience, health, and willpower required to successfully complete this Master's thesis.*

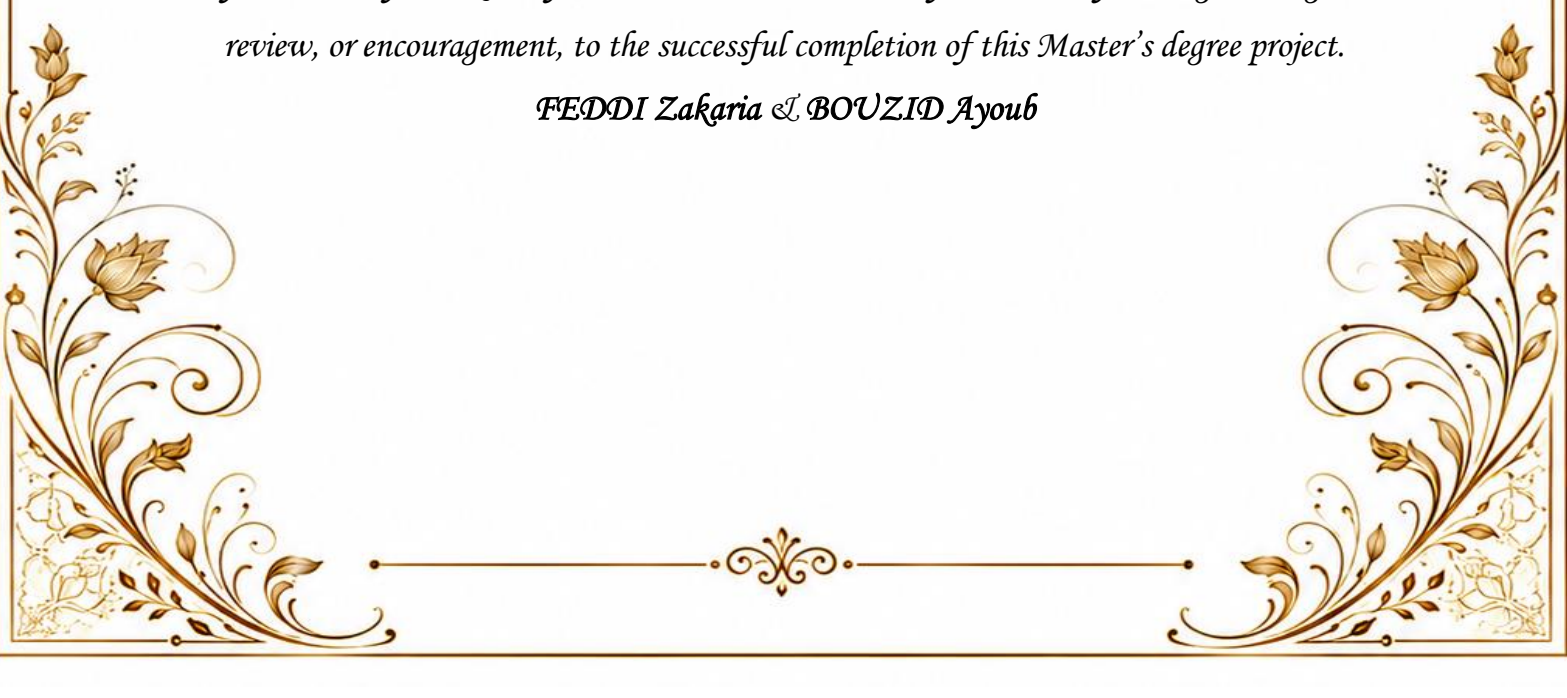
*We wish to express our deepest gratitude and sincere thanks to our academic supervisor **Mr. Sadek Ameziane**, for accepting to guide this thesis, and for his invaluable advice, constant availability, and trust throughout this project.*

Our earnest thanks also go to the distinguished members of the jury who graciously accepted to evaluate this manuscript and to enrich this work with their constructive feedback during our defense.

We would also like to extend our profound appreciation to the entire faculty and administrative staff of the Department of Electrical Engineering at the University of Bouira, for the high-quality education and technical support received during our studies in Industrial Electrical Engineering.

Finally, we warmly thank everyone who contributed, directly or indirectly, through their guidance, review, or encouragement, to the successful completion of this Master's degree project.

FEDDI Zakaria & BOUZID Ayoub



Abstract

Modern industrial environments face significant challenges due to unexpected downtime caused by mechanical and electrical faults. This Master's thesis presents an Intelligent SCADA System designed for real-time monitoring and automated fault diagnosis across multi-stage production lines. The architecture integrates a robust multi-protocol network linking TIA Portal, Factory I/O, Node-RED, and MATLAB, utilizing a Multi-Layer Perceptron to classify equipment states. To ensure operational continuity and high system availability, it deploys two completely independent diagnostic frameworks: an Online AI-Assisted layer powered by the Gemini LLM for complex web-based root-cause analysis, and a local Offline Weight-Based layer on the WinCC HMI.

Key Words: Intelligent SCADA, Tia Portal, Matlab, Multi-Layer Perceptron, Gemini LLM, HMI.

Resumé:

Les environnements industriels modernes sont confrontés à des défis majeurs liés aux arrêts de production imprévus causés par des défaillances mécaniques et électriques. Ce mémoire de Master présente un système SCADA intelligent dédié à la surveillance en temps réel et au diagnostic automatisé des pannes sur des lignes de production. L'architecture intègre un réseau multi-protocole robuste reliant TIA Portal, Factory I/O, Node-RED et MATLAB, utilisant un perceptron multicouche pour classifier l'état des équipements. Pour garantir la continuité de service et la haute disponibilité, le système déploie deux cadres de diagnostic totalement indépendants: une couche en ligne assistée par l'IA Gemini LLM sur le Web pour des analyses détaillées, et une couche locale hors ligne fondée sur les poids sur l'IHM WinCC.

Mot clés: SCADA intelligent, Tia Portal, Matlab, perceptron multicouche, Gemini LLM, IHM.

Table of Contents

Acknowledgements	i
Abstract	ii
Table of Contents	iii
List of Figure:	viii
List of Tables.....	ix
List of acronyms and symbols:	x
Acronyms	x
Symbols.....	xi
General Introduction:	1

Chapter I: Literature Review and Theoretical Background

I.1. Introduction:	3
I.2. SCADA System:	3
I.3. Brief history of SCADA:	4
I.4. SCADA Application:	4
I.5. SCADA System Components:	4
I.5.1. Master Terminal Unit (MTU):	5
I.5.2. Remote Terminal Unit (RTU):	5
I.5.3. programmable logic controller (PLC):	5
I.5.4. Communication Infrastructure:	5
I.5.5. Human Machine Interface (HMI):	5
I.5.6. Sensors:	6
I.6. The Architecture of a SCADA System:	6
I.7. SCADA communication protocols:	7
I.8. Evolution of SCADA Systems:	7
I.8.1. Monolithic (First Generation):	8

I.8.2. Distributed (Second Generation)	8
I.8.3. Networked (Third Generation)	9
I.8.4. IIoT/Cloud-based (Fourth Generation).....	10
I.9. Emergence of IoT, Cloud, and ML in SCADA	10
I.9.1. Internet of Things (IoT)	10
I.9.2. Cloud Service	11
I.9.3. Machine Learning (ML)	11
I.10. Technical and implementation challenges in IoT-Integrated SCADA	11
I.10.1. Integration Complexity	11
I.10.2. Data Management & Interoperability	11
I.10.3. High Implementation Costs	12
I.10.4. Cybersecurity vulnerability	12
I.11. Overview of maintenance Methods	12
I.11.1. Corrective (Reactive) Maintenance	12
I.11.2. Preventive Maintenance.....	12
I.11.3. Condition-Based Maintenance.....	13
I.11.4. Predictive Maintenance.....	13
I.12. Overview of Industrial Diagnosis Methods.....	13
I.12.1. Traditional Signal Processing Based Methods	13
I.12.1.1. Motor Current Signature Analysis (MCSA).....	13
I.12.1.2. Vibration and Noise Monitoring.....	13
I.12.1.3. Park Vector Conversion	13
I.12.2. Data-Driven Machine Learning Methods.....	14
I.12.2.1. Supervised Learning	14
I.12.2.2. Unsupervised Learning.....	14
I.12.3. Digital Twins (DT) & Knowledge Graphs (KG)	14

I.12.3.1. Digital Twins	14
I.12.3.2. Knowledge Graphs	14
I.12.4. Large Language Models (LLMs) & Multimodal Fusion.....	14
I.12.4.1. Multimodal Fusion	14
I.12.4.2. Few-Shot and Zero-Shot Diagnosis.....	15
I.13. Conclusion	15

Chapter II: System Design and Implementation

II.1. Introduction	16
II.2. Process Description and Specifications:	16
II.2.1 The Virtual Production Line:.....	16
II.2.1.1. Overview:	16
II.2.1.2. Components:.....	17
II.2.1.3. I/O Variables Table:.....	18
II.2.1.4. System modes and operation Sequence	20
II.2.2. The Physical Production Line	23
II.2.2.1. OULED MIMOUN	23
II.2.2.2. AGRODIV Group.....	23
II.2.2.3. MOLINO Company	24
II.2.2.4. Transformation line of wheat	24
II.2.2.5. Weighing and packaging section	24
II.3. Development Tools and Software Environment	29
II.3.1 TIA Portal V16	29
II.3.2. WinCC (Integrated / Runtime)	29
II.3.3. PLCSIM Advanced	29
II.3.4. Factory I/O	29
II.3.5. MATLAB 2023a.....	30

II.3.6. Node-RED	30
II.4. Communication Architecture Between Tools.....	30
II.4.1. OPC UA	30
II.4.2. S7 Protocol:	30
II.4.3. HTTPS.....	31
II.4.4. Web Socket Protocol	31
II.5. Application to Reel and Virtual Lines	31
II.5.1. Real-Time Integration via OPC UA.....	31
II.5.2. Data Preprocessing: Z-Score Normalization.....	32
II.5.3. Neural Network: Architecture and Training.....	32
II.5.4. Network Structure	32
II.5.5. Network shape.....	33
II.5.6. Dataset Generation and Training.....	33
II.6. Fault Diagnosis: Finding the Root Cause.....	34
II.6.1. Weight-Based Fault diagnosis Scada.....	34
II.6.1.1. Fault Weight Table.....	34
II.6.1.2. How the Priority Score is Calculated	35
II.6.2. AI-Assisted Fault Diagnosis SCADA	37
II.7. Conclusion.....	39

Chapter III: Results and Discussion

III.1. Introduction.....	40
III.2. Dataset visualization	40
III.3. Neural network results	41
III.3.1. Architecture & parameters	41
III.3.2. Training progress.....	42
III.3.2.1. Accuracy curves	42

III.3.2.2. Loss curve	43
III.3.3. Confusion matrix	44
III.3.4. Precision / Recall / F1	45
III.4. SCADA monitoring and fault diagnosis	46
III.4.1. Weight-Based fault diagnosis.....	46
III.4.1.1. Diagnosis example	46
III.4.1.2. HMI Interface.....	48
III.4.1.3. Diagnosis Results:.....	50
III.4.2. AI-Based Fault Diagnosis:	51
III.5. Discussion	52
III.5.1 Overall System Performance	52
III.5.2 Synthetic Data Limitations	52
III.6. Conclusion	53
General Conclusion:.....	55
References:.....	57
Appendix:.....	59

List of Figure:

Figure I.1: SCADA System General Layout	3
Figure I.2: Sensors to panel using 4–20 mA or voltage	4
Figure I.3: CIM Pyramid	7
Figure I.4: Monolithic architecture	8
Figure I.5: Distributed architecture	9
Figure I.6: Networked architecture	9
Figure I.7: IIoT/Cloud-based architecture	10
Figure II.1: Factory IO virtual production line simulation environment	17
Figure II.2: Virtual production line flowchart	22
Figure II.3: Sales Department of OULED MIMOUN	23
Figure II.4: AGRODIV logo	23
Figure II.5: Real production line flowchart	26
Figure II.6: Weight-based fault diagnosis communication architecture	36
Figure II.7: Weight-Based fault diagnosis architecture flow chart	36
Figure II.8: AI-Assisted Fault Diagnosis flow chart	38
Figure II.9: AI-Assisted fault diagnosis communication architecture	38
Figure III.1: Cylinder Dataset Curve	41
Figure III.2: Motor Dataset Curve	41
Figure III.3: Neural network architecture	42
Figure III.4: Accuracy curve of the pneumatic cylinder	43
Figure III.5: Accuracy curve of the motors	43
Figure III.6: Loss curve of the pneumatic cylinder	43
Figure III.7: Loss curve of the induction motor	44
Figure III.8: Pneumatic cylinder confusion matrix	44
Figure III.9: Motor confusion matrix	45
Figure III.10: Virtual production line critical condition alert popup displayed on the HMI	48
Figure III.11: Real production line critical condition alert popup displayed on the HMI	49
Figure III.12: Virtual production line diagnosis Suggest popup displayed on the HMI	49
Figure III.13: Real production line diagnosis Suggest popup displayed on the HMI	50
Figure III.14: Ai-Assisted Virtual Production line Web dashboard Scada	50
Figure III.15: Ai-Assisted Real Production line Web dashboard Scada	51

List of Tables

Table II.1: Virtual Production Line PLC Inputs	19
Table II.2: Virtual production line PLC Outputs	20
Table II.3: Real production line PLC Inputs	28
Table II.4: Real production line PLC Outputs	28
Table II.5: Network Structure	33
Table II.6: Fault sensor weight table. Each row sums to 1.0	34
Table III.1: Classification Results for the Pneumatic Cylinder NN	45
Table III.2: Classification Results for the Induction Motor NN	46
Table III.3: Normal Operating Ranges and Alert Thresholds for the Induction Motor	46
Table III.4: Real-Time Sensor Readings at Alarm Event	46
Table III.5: Confidence Score Calculation for Fault Diagnosis	47
Table III.6: Final Fault Ranking Based on Combined Confidence Score	47

List of acronyms and symbols:

Acronyms:

AI: Artificial Intelligence.

API: Application Programming Interface.

ANN: Artificial Neural Network.

ANSI: American National Standards Institute.

BN: Batch Normalization.

CAN: Controller Area Network.

CIM: Computer Integrated Manufacturing.

COTS: Commercial Off-The-Shelf.

DCS: Distributed Control System.

DDoS: Distributed Denial of Service.

DNP: Distributed Network Protocol.

DT: Digital Twins.

ERP: Enterprise Resource Planning.

FC: Fully Connected.

FFNN: Feed-Forward Neural Networks.

FFT: Fast Fourier Transform.

FN: False Negative.

FP: False Positive.

GAN: Generative Adversarial Network.

HMI: Human Machine Interface.

HTTP: Hypertext Transfer Protocol.

IBM: International Business Machines.

ICS: Industrial Control System.

IEC: International Electrotechnical Commission.

IIoT: Industrial Internet of Things.

IoT: Internet of Things.

ISA: International Society of Automation.

ISO: International Organization for Standardization.

KG: Knowledge Graphs.

LAN: Local Area Network.

LLM: Large Language Model.
MATLAB: Matrix Laboratory.
MES: Manufacturing Execution System.
ML: Machine Learning.
MLP: Multilayer Perceptron.
MTBF: Mean Time Between Failures.
MTU: Master Terminal Unit.
NC/NO: Normally Open/Normally Closed.
OPC UA: Open Platform Communications-Unified Architecture.
PdM: Predictive Maintenance.
PLC: Programmable Logic Controller.
Profibus: Process Field Bus.
ReLU: Rectified Linear Unit.
RF: Random Forest.
RMS: Root Mean Square.
RPM: Revolutions Per Minute.
RTU: Remote Terminal Unit.
SCADA: Supervisory Control and Data Acquisition.
SVM: Support Vector Machines.
TCP/IP: Transmission Control Protocol / Internet Protocol.
TIA Portal: Totally Integrated Automation Portal.
TP: True Positive.
WAN: Wide Area Networks.
WinCC: Windows Control Center.

Symbols:

H_f : History score.
 C_f : Confidence score.
 d_s : Deviation score.
 W_f : Weight assigned to a sensor.
 X_{norm} : Normalized value.
 X_{input} : Sensor reading.
 μ : Average sensor readings.

σ : Standard deviation.

General Introduction:

In the modern era of Industry 4.0, the control, monitoring and supervision of complex industrial manufacturing processes are largely dependent on the availability of accurate and real-time data acquisition and decentralized command systems. The increasing scale and the technical complexity of production lines imply that the traditional automation mechanisms are not sufficient to ensure the efficiency and the safety of the operations. In this context, Supervisory Control and Data Acquisition (SCADA) systems have become the backbone of industrial infrastructure management, acting as the bridge between the raw field instrumentation and the higher corporate management decisions.

SCADA systems were traditionally operated on closed, monolithic and isolated networks where plant alarms were managed mainly using rigid static logic. However, the rapid proliferation of industrial automation along with the technological advancements in Industrial Internet of Things (IIoT), Cloud Computing and Artificial Intelligence (AI), has fuelled a major paradigm shift. Today's SCADA architectures are evolving from passive visualization dashboards to intelligent, data-driven supervisory environments that can self-diagnose, provide advanced telemetry and proactively localize faults.

The main operational bottleneck for the manufacturing sector today is the unanticipated downtime due to mechanical and electrical faults. Traditional reactive maintenance methods often result in substantial collateral component damage and large financial losses, while standard preventative or condition-based strategies rely heavily on human intervention for the manual analysis of raw and complex sensor signals. So, it is vital to develop automated SCADA infrastructures that can autonomously interpret trends from sensors, isolate faults before they lead to catastrophic equipment failures, and provide actionable guidance to maintenance personnel.

The main goal of this Master's thesis is to design, develop and evaluate an Intelligent SCADA System for real-time monitoring and advanced automated fault diagnosis of a multi-stage manufacturing process. This work closes the gap between simulated validation and empirical application by building two different experimental platforms:

A Virtual Production Line: A complete assembly, packaging and high density sequential 54-cell warehouse storage system simulated with Factory I/O running on a simulated Programmable Logic Controller (PLC).

Physical Production Line: Based on the industrial grain processing specifications of the Ouled Mimoun mill (subsidiary of Agrodiv Group), it is dedicated to the automated weighing and packaging line of wheat and semolina products.

The goal of this thesis is to provide intelligent fault diagnosis by implementing two different analytical frameworks built on top of a core multi-protocol industrial network architecture (using OPC UA, the Siemens S7 Protocol, HTTPS APIs, and WebSockets):

Multilayer perceptron (MLP): Multilayer perceptron architectures trained on synthetic data to continuously categorize machine states into Normal, Warning, or Critical operational bands according to physical characteristics like vibration, current, temperature, pressure, and flow rates.

Diagnostic Layer: A Weight-Based Priority Score formulation (balancing statistical sensor deviations with structural machine histories) and a state-of-the-art AI-Assisted Diagnostic System powered by the Gemini Large Language Model (LLM). The latter mechanism maps raw anomalies into natural language root-cause explanations and prioritized maintenance step actions directly displayed onto an interactive web-based SCADA dashboard.

To give an orderly account of this development the remainder of this manuscript is divided into three major chapters:

Chapter I (Literature review and theoretical background): Sets the general theoretical framework of SCADA systems, illustrating their architectural levels concerning the ANSI/ISA-95 Computer Integrated Manufacturing (CIM) pyramid, communication structures, and the recent evolution towards Cloud/IoT models. It also discusses the current philosophies of industrial maintenance and reviews diagnostic approaches from traditional signal processing to modern Digital Twins and Multimodal LLMs.

Chapter II (System Design and Implementation): Describes the complete engineering specifications, the sequential operation flowcharts, and I/O programming of the physical wheat mill weighing system and the 3D virtual Factory I/O production line. The technical layout of the software environment (TIA Portal V16, PLCSIM Advanced, Node-RED, MATLAB) and the step-by-step mathematical formulation of the Z-score data preprocessing, neural network topologies and rule-based vs AI-driven diagnostic execution layers are described.

Chapter III (Results and Discussion): Presents the performance profiles of the trained neural network models in terms of accuracy curves, loss regressions and detailed confusion matrices. The visual layout of the built WinCC Human-Machine Interfaces (HMIs) and web dashboards is shown, live diagnostic fault-ranking outputs during critical events are demonstrated, system limitations with respect to synthetic data training are discussed, and the ultimate utility of the developed frameworks is confirmed.

Chapter I:

Literature review and theoretical background

I.1. Introduction:

Real time data acquisition and centralized command are important issues for the control and supervision of modern industrial processes. In this context, SCADA systems have become the backbone of management of industrial infrastructures. The objective of this chapter is to develop the theoretical bases of this study, by exploring the SCADA ecosystem, from its traditional components to its newest technological mutations and providing a complete review of the maintenance and industrial diagnosis methods that derive from it.

First, SCADA systems are defined, their history, applications and the essential components that form their architecture and communication protocols. Then, the evolution of the generation of these systems is analyzed with the emergence of the IoT, Cloud Computing and Machine Learning, and the technical and cybersecurity challenges of this integration are faced. Finally, an overview of the different industrial maintenance strategies is given, leading to an extensive study of diagnosis methods ranging from traditional signal processing techniques to state-of-the-art approaches like Digital Twins, Knowledge Graphs and Large Language Models (LLMs).

I.2. SCADA System:

SCADA is described as Supervisory Control and Data Acquisition. The SCADA system gathers data, forwards it to the central site, performs the necessary analysis and control, and presents the data on numerous operator screens or displays. Then, the resulting control actions are sent to the process [1].

SCADA is a control system that obtains data in real time from a control center over long distances through fiber optic cables or radio. This is important for industries that need to perform defect detection, valve management and remote set-point control without site visits [2].

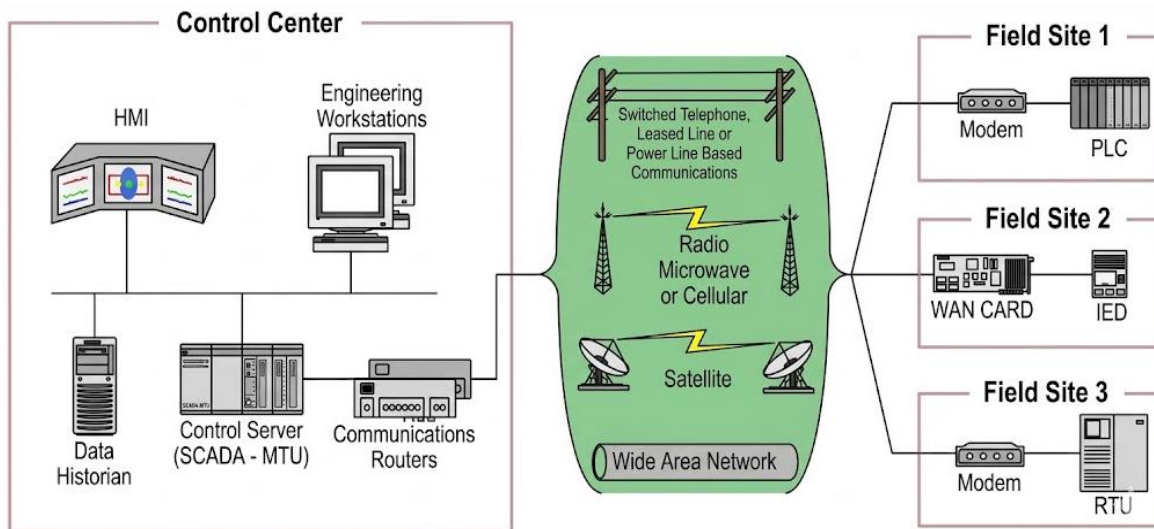


Figure I.1: SCADA System General Layout [3]

I.3. Brief history of SCADA:

SCADA has been around for as long as control systems have been around. Early SCADA systems collected and displayed process data by using strip chart recorders, lights, and meter panels. The operator exercised supervisory control over the control knobs manually. The devices have been and are used for data collection and supervisory control in factories, plants and power generating facilities [4].

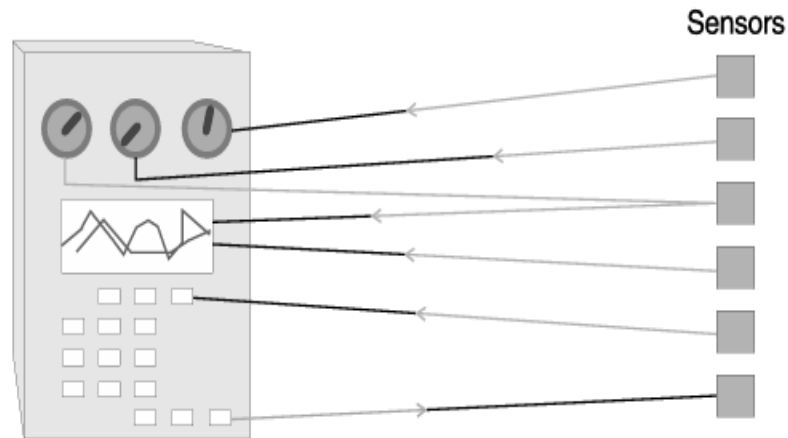


Figure I.2: Sensors to panel using 4–20 mA or voltage [4]

I.4. SCADA Application:

The most common areas of applicability for SCADA systems are as follows [5]:

- Monitoring systems for a production line or an entire company
- Telecommunication systems
- Power distribution systems
- Energy transmission systems
- Oil industry
- Gas and natural gas extraction
- Storage systems
- Irrigation systems
- Water distribution systems
- Sewerage systems parameters measuring equipment for fluids hydroelectric systems.

I.5. SCADA System Components:

SCADA system is typically divided into many main functional units. These units enable to operate the system as unified units, offering reliable control, monitoring and optimization for many industrial applications.

I.5.1. Master Terminal Unit (MTU):

Some industries refer to MTU as host computer some as server [2]. The MTU is the implementation of the supervisory control layer, normally a computer (e.g., personal computer). It periodically samples data from the process control layer. The MTU provides an important tool for the company management like alarm management, data logging, process trending and report production [6].

I.5.2. Remote Terminal Unit (RTU):

An RTU is a self-contained microprocessor-based data acquisition and control unit. RTU's main function is to control and monitor the equipment remotely from the central station [1]. The RTU is responsible for collecting analog values, alarm and status points, and metered amounts information from the field. This information is stored in the memory of the RTU until it is requested by the MTU. The RTU processes the information and transmits it to the MTU. In addition, the RTU opens and closes valves, switches on and off switches, provides the analog signals which are used as set points and outputs pulse trains for driving stepping motors according to the directions of the MTU [2].

I.5.3. programmable logic controller (PLC):

A PLC (Programmable Logic Controller) is a solid-state computer-based device for controlling industrial processes and machines. Originally intended to do the same logic operations as relays, drum switches, or mechanical timers or counters. Timed Signals Analog control is also a typical feature of PLC operation today. The advantage of a PLC over RTU options from various manufacturers is that it can perform a large variety of tasks and is easy to configure for different tasks [1]. PLCs are also much less space-efficient and physically compact than other alternative solutions. However, PLCs may not be suitable for some applications such as radio telemetry applications [4].

The PLCs consists of Microprocessor, Power supply, Input/Output module (analog and/or digital) and Communication module. They are programmed for a set of functions (arithmetic, logic, control, monitoring, etc...) and are able to take total control of industrial machinery and processes [6].

I.5.4. Communication Infrastructure:

The communication infrastructure is how a SCADA system communicates with field controllers and devices. This allows the system to collect data from those devices and send control commands back, providing monitoring and control.

I.5.5. Human Machine Interface (HMI):

Human-Machine Interfaces (HMIs) are the interaction and visualization layer of SCADA systems, allowing operators to monitor processes, analyze trends and respond to alerts. HMIs display data in a simple-to-understand way such as dashboards or graphical displays to aid in real-time decision

making. Operators can use HMIs to send commands, override automatic controls or change system parameters as needed [7]. It is a bridge between the hardware and the software [8].

One of the most important parts of a SCADA system is the HMI software. How reliable and compatible it is especially with all the devices that are plugged in. HMIs are usually built to fit the needs of the system. There are two kinds: Proprietary software is designed for a particular hardware type; it is less flexible COTS software is more flexible and can be used on multiple devices. In general, the HMI links the hardware and software, transparently presenting system data and making the data more comprehensible for the users [8].

I.5.6. Sensors:

SCADA systems depend on sensors, the heart of the system, to collect real-time data about machinery, equipment or environmental conditions. These devices measure parameters such as temperature, pressure, flow rate and voltage providing vital inputs for process control. Sensors are generally deployed strategically to cover all operational parameters and get accurate and timely data [7].

I.6. The Architecture of a SCADA System:

Computer-integrated manufacturing (CIM), also known as ISO/IEC 62264 or ANSI/ISA-95 standard, is described as a pyramid with functional tiers. This pyramid consists of two parts. The first part is the Supervision & Control part and it is responsible for collecting industrial data and execute operating instructions. The second part is called the Decision-Making section and it contains the tools to analyze operational data and to create management directives.

The Supervision & Control part has three functional levels: The first level is called Operational level or Level 0 and it includes the basic equipment like sensors and actuators. Level 1 layer is the Automation layer. It includes equipment with a certain level of intelligence that allows the automation of processes such as PLCs and DCS. The last layer of this section is the Level 2 layer, the Supervision layer. This layer covers all the Industrial Control Systems (ICS) like SCADA, DCS and Smart Grids.

The Decision-Making component is structured in two layers: The first layer is the Level 3 layer or the Industrial Process Management layer (MES – Manufacturing Execution System) based on an IT system that optimizes the production processes by analyzing real-time data from the production system. The second layer is the Level 4 layer, called Enterprise Resource Planning (ERP) layer, which offers a

comprehensive solution to define, monitor and manage business plans of industrial companies [9].

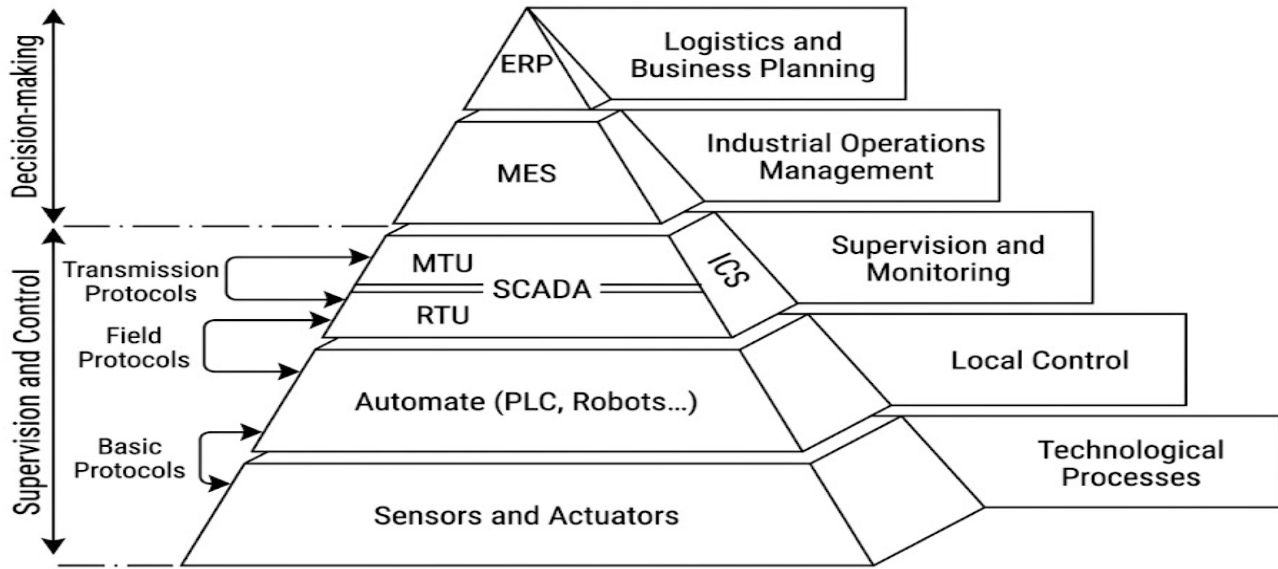


Figure I.3: CIM Pyramid [9]

I.7. SCADA communication protocols:

In the CIM pyramid, SCADA communication protocols are classified into basic protocols, field protocols and transport protocols (also called transmission protocols).

The most basic Protocols of Communication are Basic protocols. They are used for communication between operational equipment and PLCs and use frames with a very simple format and very small data sizes. Field protocols are proximity protocols that allow local digital communication between PLCs and RTUs with frames with larger data sizes like IEC-60870-5-103, IEC-61850, Profibus, CAN, Modbus, 1-ware... Lastly, transport protocols are good protocols that deal with the transmission of large data flows between MTUs and RTUs over wide area networks (WAN). These include IEC 60870-5-101/104, DNP3 and some flavors of Modbus. In the following we will focus on presentation of main transport protocols used in network remote control [9].

I.8. Evolution of SCADA Systems:

SCADA systems were historically deployed in small, isolated networks. However, the global introduction of industrial automation and communication systems led to a paradigm shift in the SCADA architecture. It migrated from its traditional monolithic architecture to a much more sophisticated networked and cloud-based architecture.

I.8.1. Monolithic (First Generation):

The first generation of SCADA systems were like mainframe computers, everything was centralized and there was not a lot of communication between stations. Each system had only basic links to the field devices, so they operated in isolation. There were early ideas of wide area communication but the modern networking protocols had not yet been invented. Instead, communication was done with simple proprietary protocols, and not very well. These systems had problems with a lot of data or lots of remote connections. There were also very few ways to connect to field devices and basic backup systems were mostly used to find failures and ensure there was always a backup [8].

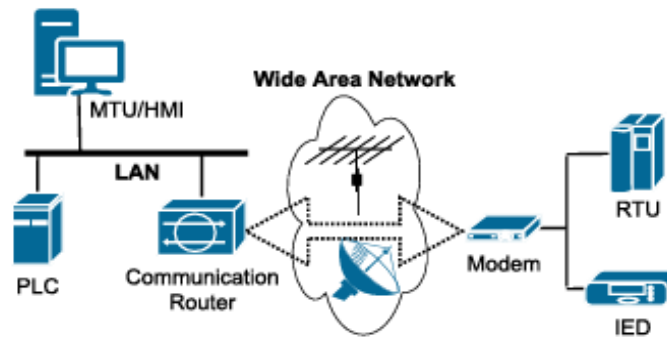


Figure I.4: Monolithic SCADA architecture [10]

I.8.2. Distributed (Second Generation):

The second generation of SCADA systems had a big impact on the architecture of the system from a centralized system to a distributed system that was connected by a Local Area Network (LAN). Each of the stations had a function to perform, such as communication, monitoring (HMI) or data storage. They were also able to communicate information in real time. This distribution has improved the speed, reliability and processing of the system. In the first generation one main station did it all. Now a few stations are working together. So, if one goes down, the system still works. Also, WAN technology was added so people could talk to remote stations but it was not as good as modern networks [8].

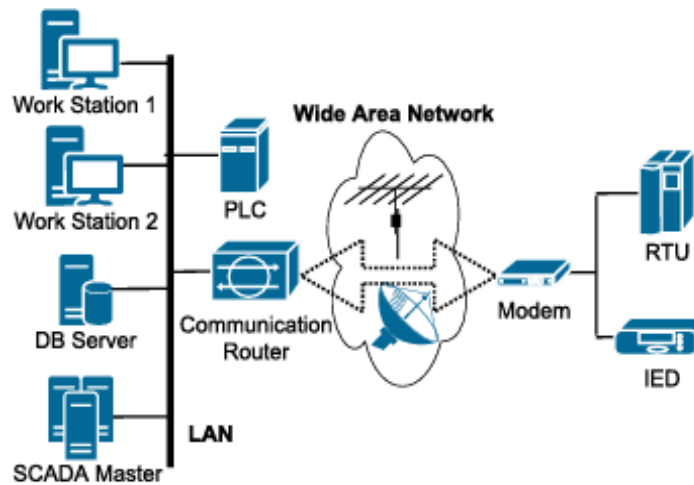


Figure I.5: Distributed SCADA architecture [10]

I.8.3. Networked (Third Generation):

The third generation of SCADA systems is like the second, but they make a major change, using open architecture instead of proprietary systems. This means that it supports common communication protocols such as TCP/IP over LAN and WAN so it can connect different systems and devices easily. Functions are still distributed across multiple stations, but now on larger networks, making them more scalable, more reliable and disaster recoverable. Data replication means that if one station fails, the other stations can continue to run. However, they are more susceptible to cyber threats due to their connectivity to larger networks and the Internet [8].

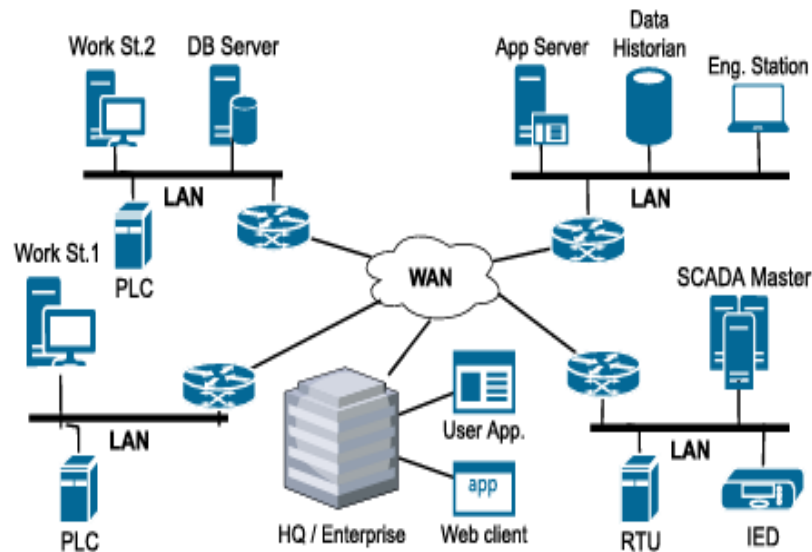


Figure I.6: Networked SCADA architecture [10]

I.8.4. IIoT/Cloud-based (Fourth Generation):

The fourth generation of SCADA systems is driven by the integration of Internet of Things (IoT) technologies. Such SCADA systems are called Internet of Things (IoT) based SCADA systems. This includes the integration of IoT and cloud computing to develop more advanced, flexible and efficient solutions. These systems enhance functionalities, reduce costs and ease maintenance with cloud-based infrastructures [11]. One of the best features of this generation is the ability to access real-time process data from anywhere, on different devices and operating systems." Cloud computing also provides the ability to deploy sophisticated monitoring algorithms, models, and statistical analysis tools [11].

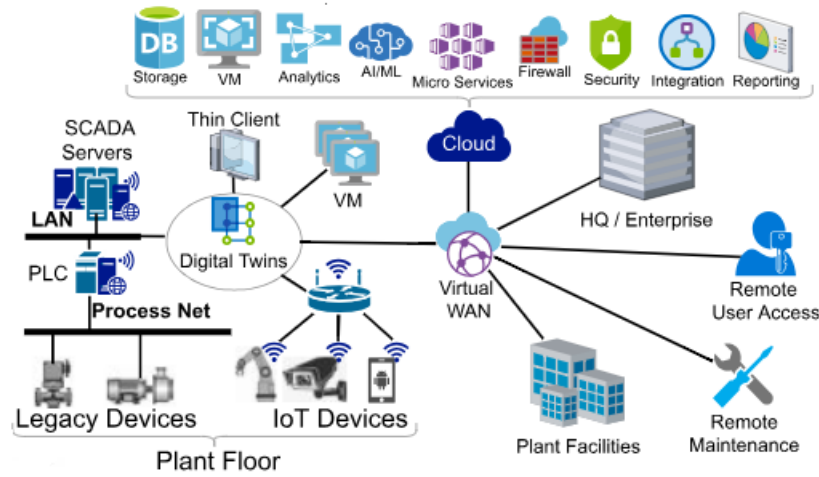


Figure I.7: IIoT/Cloud-based architecture [10]

I.9. Emergence of IoT, Cloud, and ML in SCADA:

Traditional SCADA systems are not able to support predictive maintenance and real-time decision making in complex environments due to lack of advanced analytics capabilities. These limitations have heightened the need for modernization to enable SCADA systems to meet the demands of today's industrial automation. These limitations are being overcome by modern solutions such as IoT, Machine Learning and Cloud computing integration, and improving connectivity and data processing capabilities.

I.9.1. Internet of Things (IoT):

The Internet of Things (IoT) is the massive network of physical objects around the world that are connected to the Internet, collecting and sharing data. Embedding sensing computing and communication capabilities [12] these devices can communicate with each other and with cloud and edge systems. The integration of IoT (Internet of Things) technology into SCADA systems has greatly increased their ability to collect and analyse data, and perform remote operations in real time. IoT enabled SCADA systems are

connected with the smart devices and sensors which communicate with each other over wireless networks to improve the ability to see and manage industrial operations [7].

I.9.2. Cloud Service:

Cloud computing significantly improves SCADA systems by providing flexible storage and stronger computing abilities. By using cloud integration, SCADA systems can handle a lot of data quickly as it comes in, allowing devices and operators to talk to each other smoothly. This feature helps cut down delays and makes the system react faster, which solves a big problem with old SCADA systems. Cloud platforms allow people to access systems from anywhere and manage everything from one place, which helps keep performance the same at different industrial locations [7].

The development of the Internet of Things (IoT), cloud computing and machine learning (ML) has generated significant innovations in SCADA systems, transforming them into robust, flexible and intelligent architectures able to face the challenges of modern industry [7].

I.9.3. Machine Learning (ML):

In addition to these developments, machine learning supports predictive analytics and anomaly detection. ML algorithms can analyze historical and real-time data to identify patterns and predict potential system failures. This allows for predictive maintenance that can help reduce downtime and operating costs. Moreover, machine learning improves decision-making in SCADA systems for automated responses to complex situations, enabling more efficient and flexible industrial operations in dynamic environments [7].

I.10. Technical and implementation challenges in IoT-Integrated SCADA:

The integration of IoT in the SCADA systems has several technical and implementation challenges [7]:

I.10.1. Integration Complexity:

However, integration of IoT technologies into the existing SCADA systems can be challenging as traditional SCADA systems were not originally designed to allow modern connectivity. The integration of new IoT devices into legacy infrastructure usually requires a lot of customization and specialized knowledge. Ensuring compatibility between PLCs and IoT devices is tricky and can even cause issues with operations if not done properly.

I.10.2. Data Management & Interoperability:

The proliferation of IoT devices generates enormous volumes of real-time data to be processed and stored efficiently. But lack of common protocols and formats can lead to fragmented systems and affect reliability.

I.10.3. High Implementation Costs:

The cost of implementation is another great challenge. SCADA Systems based on IoT require a huge investment in Hardware, Software and Cloud infrastructure. Plus, the ongoing costs of maintenance, security, and staff training.

I.10.4. Cybersecurity vulnerability:

The integration of IoT devices in SCADA systems has posed serious challenges to cybersecurity and has increased the vulnerability of these critical infrastructures to different kinds of threats. Below are some of the more common threats:

- Distributed Denial of Service (DDoS)
- Data breaches
- Ransomware poses

I.11. Overview of maintenance Methods:

Maintenance is the description of the procedures performed to preserve and restore the functionality of a machine aiming to reduce the malfunctioning and failures. The maintenance procedure includes tasks such as inspection, cleaning, lubrication, replacements and repairs [13]. Industrial maintenance has certainly come a long way from reactive fixes to highly sophisticated AI driven prognostics.

I.11.1. Corrective (Reactive) Maintenance:

This is the most basic level of maintenance which is commonly known as run-to-failure or breakdown maintenance. Here, no pre-emptive measure is taken up until the complete failure of a component. The main role is to bring a defective unit back to the desired operating condition. This approach does not require preplanning but it is believed to be unsustainable in modern industrial environments due to the risk of unpredictable downtime, safety issues and significant financial losses [13].

I.11.2. Preventive Maintenance:

This strategy entails the execution of planned tasks at specified time intervals or as needed to avoid deterioration and failure of the equipment. This is called as “time-driven” maintenance and involves regular maintenance tasks that are performed regularly such as inspection, cleaning, lubrication and replacement of parts irrespective of the condition of the machine at that time. The goal is to increase the reliability of the equipment, to extend the life span and to decrease the chances of a functional failure in general [14].

I.11.3. Condition-Based Maintenance:

Condition-based maintenance is defined in the standard EN 13306 as preventative maintenance, consisting of condition monitoring, inspection, testing, analysis and subsequent maintenance actions. Condition based maintenance is concerned with the prediction of a maintenance action based on the symptoms of degradation and deviation from the expected normal behaviour of an asset. The equipment has a number of sensors that track the equipment and collect relevant data throughout its operational life. Moreover, the context information such as humidity, temperature etc. can also add important information [15].

I.11.4. Predictive Maintenance:

Predictive maintenance is a predictive maintenance methodology that utilizes sensor data, real-time monitoring, and advanced analytics to anticipate when equipment failure is likely to happen. PdM is focused on optimizing maintenance schedules and reducing unplanned downtime, while increasing the life of the machines. With PdM, companies can perform maintenance of the equipment only when needed, thus reducing downtime and ensuring that the equipment is always working at an optimal level [16].

I.12. Overview of Industrial Diagnosis Methods:

The field of industrial fault diagnosis has evolved from manual inspection and physical models to data-driven artificial intelligence with a high complexity. The technological development of these methods can be divided into four main stages [17].

I.12.1. Traditional Signal Processing Based Methods:

Before the arrival of AI, diagnosis was carried out by experts interpreting physical signals:

I.12.1.1. Motor Current Signature Analysis (MCSA):

A well proven and widely used technology that measures electrical current to detect an extremely large number of mechanical and electrical faults.

I.12.1.2. Vibration and Noise Monitoring:

Studies the physical vibration of machinery in the time domain, a simple way to detect faults is to calculate the Root Mean Square (RMS) and check if the vibrations are above the allowed global standards. The Fast Fourier Transform (FFT) which is widely used with the reduction of calculation number is used to extract the fault feature in frequency domain [18].

I.12.1.3. Park Vector Conversion:

Converts three phases to two phase systems and enables visual detection of unbalanced voltage

supplies or short circuits in the stator, where a healthy motor draws a circular pattern and a faulty one draws an oval [18].

I.12.2. Data-Driven Machine Learning Methods:

As industrial systems became more complex, machine learning stepped in to map the relationships between extracted features like FFT or RMS data and specific faults.

I.12.2.1. Supervised Learning:

Algorithms like Artificial Neural Networks (ANN), Support Vector Machines (SVM), Random Forests (RF) and Decision Trees learn to classify faults by training on historical data sets with explicit “healthy” or “faulty” labels. These are very accurate approaches, but heavily rely on the availability of large amounts of good quality labeled data [17].

I.12.2.2. Unsupervised Learning:

Anomaly detection methods like K-means Clustering, Autoencoders and Generative Adversarial Networks (GANs) can identify anomalies without the need for explicit fault labels. They operate on the principle of extracting the intrinsic structure of normal operating data and recognizing deviations, which is critical when handling unknown fault modes or data scarcity [17].

I.12.3. Digital Twins (DT) & Knowledge Graphs (KG):

These technologies help to bridge the gap where it is too rare, dangerous or expensive to collect physical fault data:

I.12.3.1. Digital Twins:

This method creates a high-fidelity digital twin of the physical asset that is synchronized with live operating data. This allows engineers to virtually simulate various extreme operational conditions, to inject faults and to generate synthetic fault data for training diagnostic models without any risk to physical equipment [17].

I.12.3.2. Knowledge Graphs:

They organize heterogeneous data (e.g. equipment topology, operation logs and expert rules) into structured computable semantic networks. They explicitly model the relationships between equipment components and fault symptoms, which has strong interpretability, can trace the propagation path of faults and reduce the dependence of individual expert experience [17].

I.12.4. Large Language Models (LLMs) & Multimodal Fusion:

The latest stage of intelligent fusion utilizes the reasoning and representation capabilities of large generative models.

I.12.4.1. Multimodal Fusion:

LLMs can uniformly consume and align heterogeneous data from multiple sources, including

numerical sensor measurements (e.g., vibration and temperature) and textual records [17].

I.12.4.2. Few-Shot and Zero-Shot Diagnosis:

LLMs, with their rich pre-trained semantic knowledge, can be fine-tuned to specific fault scenarios with limited labeled data, or even identify unseen faults via reasoning and retrieval augmented generation [17].

I.13. Conclusion:

Finally, the deployment of SCADA systems has resulted in great improvements of industrial monitoring and control, with architectures able to collect, process and present data in real time. However, in the current fast development of the Internet of Things (IoT), cloud services and machine learning, the interaction of these modern technologies with SCADA environments has become a necessity, due to the great advantages they provide, especially in terms of advanced fault detection and predictive maintenance. Moreover, this technological evolution has revolutionized the industrial diagnosis with a significant shift from classical signal processing methods to advanced data-driven models, Digital Twins and Large Language Models (LLMs) for high-accuracy failure prediction. While the convergence brings huge benefits, it also brings some critical challenges to be addressed: complexity of integration, high cost of implementation.

Having established the foundations for this study, the theoretical concepts, evolution pathways and challenges are now presented. The next chapter will therefore move from theory to practice and will describe the concrete process specifications, development tools and communication architectures needed to implement these advanced monitoring and diagnostic solutions.

Chapter II:

System Design and Implementation

II.1. Introduction

This chapter builds upon the theoretical foundations presented in the previous chapter by describing the systems built in this project and the technologies used to build them. The first one is the description of two production lines, a simulated environment created via Factory I/O and a real industrial line, both programmed by Siemens TIA Portal together with the S7-1500 PLC.

Based on this process description, the engineering details of the intelligent supervision system designed on the SCADA infrastructure are explained. This covers the communication protocols used, the structure of the neural network, the data preprocessing needs and the main fault diagnosis technique. In the end, the aim of this chapter is to give a complete and understandable description of the system design, as well as the working principle of each single component, in order to establish the necessary background for the presentation of the final results in the next chapter.

II.2. Process Description and Specifications:

The section gives a comprehensive description of the operational characteristics of the two different production lines that have been used for the purpose of this project. In order to prove the effectiveness of the designed SCADA System model, it has been implemented into both the virtual production line, which has been created in Factory I/O, and the physical production line based on the grain milling factory of Ouled Mimoun.

II.2.1 The Virtual Production Line:

II.2.1.1. Overview:

At the beginning of the production line two different raw parts enter the system, each one representing one part of the final product. The first section is targeted to Machine 1 and designed to meet the required specifications. At the same time, the second part is sent to Machine 2 where it is molded into a complementary shape that fits the first. When both parts are properly made they are fitted together at an assembly station. Here they are carefully aligned and stacked into a single working unit. The unit is assembled but not quite finished. The finished products are then transported by a conveyor system to a buffer storage, where they are kept temporarily.



Figure II.1: Factory IO virtual production line simulation environment

II.2.1.2. Components:

The main components of the virtual production line are as follows, which contribute to the execution of different manufacturing and handling operations:

- **Conveyors:** The Conveyor is used to transport parts (Lid and base) between different stages of the production process.
- **Roller Conveyor:** The Roller conveyor is another type of conveyors used to transport boxes.
- **Loading conveyor:** The loading conveyor enables proper box alignment, allowing the stacker crane to grip and handle it effectively.
- **Stacker Crane:** The stacker crane is used to move, lift, and store loads to the storage system. It operates along defined axes to place and retrieve items.
- **Machining Center:** The machine is used for the manufacturing of the lid and the bases from the raw material.
- **Two Axes Pick and Place:** this machine comports vertical and horizontal pneumatic cylinders to pick and place the lid on the base.
- **Positioner:** The positioner uses a pneumatic cylinder to clamp the parts
- **Diffuse Sensors:** Detect parts.

- Push buttons NC/NO: Push buttons are manual control devices, they allow operators to start, stop, or reset the system.

II.2.1.3. I/O Variables Table:

The following tables presents the input and output variables used for monitoring and controlling the production process:

II.2.1.3.1. Inputs:**Table II.1:** Virtual Production Line PLC Inputs

Name	Data Type	Address	Comment
Start_Storage	Bool	%I0.0	Start the production line mode storage
Start_Transit	Bool	%I0.1	Start the production line mode transit
Start_Discharge	Bool	%I0.2	Start the production line mode discharge
Stop	Bool	%I0.3	Stop the production line
Base_at_place	Bool	%I0.4	Detect the base before clamp it
Base_clamped	Bool	%I0.5	Detect the base if clamped
Item_Detected	Bool	%I0.6	Detect the presence of the lid
Pos_at_limit	Bool	%I0.7	Detect the position of the base positioner
Part_leaving	Bool	%I1.0	Sensor Detect the leaving parts
FACTORY I/O (Reset)	Bool	%I1.1	Reset Factory io simulation
Reset_Faults	Bool	%I1.2	Reset the injected faults from the panel
Robot_busy_1	Bool	%I1.6	Detect whether the Robot is running or not
Robot_busy_2	Bool	%I1.7	Detect whether the Robot is running or not
Count_sensor	Bool	%I2.1	Counting passing parts
Empty_Box_Ready	Bool	%I2.0	Detects the presence of the empty box
At_Load	Bool	%I2.2	Detect the box if at the end of the loading conveyor
Stacker_crane_1_left_Limit	Bool	%I2.4	Detect when the stacker 1 crane at Left
Stacker_crane_1_right_Limit	Bool	%I2.5	Detect when the stacker 1 crane at Right
Stacker_crane_1_Mid_Limit	Bool	%I2.6	Detect when the stacker 1 crane at Mid
Stacker_1_moving_X	Bool	%I2.7	Detect the horizontal movement of stacker crane 1
Stacker_1_moving_Z	Bool	%I3.0	Detect the vertical movement of stacker crane 1

Loaded_Box_End	Bool	%I3.1	Detect the box at the end of the loaded box conveyor
At_55	Bool	%I3.2	Detect if if the stacker crane in home position
Lid_at_place	Bool	%I3.3	Detect the lid before clamp it
Lid_clamped	Bool	%I3.4	Detect the lid if clamped
X Position	Real	%ID30	Feedback the horizontal cylinder location
Z Position	Real	%ID34	Feedback the vertical cylinder location

II.2.1.3.2 Output:**Table II.2:** Virtual production line PLC Outputs

Name	Data Type	Address	Comment
Lids conveyor	Bool	%Q0.0	Command the lid conveyor
Clamp lid	Bool	%Q0.1	Command the lid clasper
Bases conveyor	Bool	%Q0.2	Command the base conveyor
Clamp base	Bool	%Q0.3	Command the base clasper
Grab	Bool	%Q0.4	Command the grasper
Pos. raise	Bool	%Q0.5	Command the pos raise
Start_Storage_light	Bool	%Q0.6	Control panel start light
Stop light	Bool	%Q0.7	Control light stop light
Start_Robot_1	Bool	%Q1.0	Start Robot 1
Start_Robot_2	Bool	%Q1.1	Start Robot 2
Stop_Robot_1	Bool	%Q1.2	Stop Robot 1
Stop_Robot_2	Bool	%Q1.3	Stop Robot 2
Robot_1_conv	Bool	%Q1.4	Command the Robot 1 conveyor
Robot_2_conv	Bool	%Q1.5	Command the Robot 2 conveyor
Empty_Box_Conveyor	Bool	%Q1.6	Command the empty boxes conveyor
Loaded_Box_Conveyor	Bool	%Q1.7	Command the loaded boxes conveyor
Load_Conveyor	Bool	%Q2.0	Command the load conveyor
Stacker_crane_1_left	Bool	%Q2.1	Command the stacker crane 1 to Left
Stacker_crane_1_right	Bool	%Q2.2	Command the stacker crane 1 to Right
Stacker_crane_1_lift	Bool	%Q2.3	Command the stacker crane 1 to Lift

Unload_Conveyor	Bool	%Q2.4	Command the unload conveyor
Shipping_Conveyor	Bool	%Q2.5	Command the shipping conveyor
Start_Transit_Light	Bool	%Q2.6	Control panel start transit light
Start_Discharge_Light	Bool	%Q2.7	Control panel start discharge light
Counter	DInt	%QD30	Display the number of the parts
X Set Point	Real	%QD34	Set the position of the horizontal cylinder
Z Set Point	Real	%QD38	Set the position of the vertical cylinder
Target_position_1	DInt	%QD42	Set the stacker crane 1 position

II.2.1.4. System modes and operation Sequence:

The system can operate in three different modes:

- **Storage mode:** Activated by pressing the Start Storage button. Parts are produced, assembled, packed into boxes, and stored sequentially in the warehouse cells.
- **Transit mode:** Activated by pressing the Start Transit button. Parts are produced, assembled, and packed into boxes, but the boxes are sent directly to the shipping conveyor without being stored.
- **Discharge mode:** Activated by pressing the Start Discharge button. Production remains stopped while the stacker crane unloads stored boxes from the warehouse and delivers them to the shipping conveyor. This mode can only be activated when the warehouse is completely full.

II.2.1.4.1. Storage Mode Sequence:

1. When the Start button is pressed, machining centers and conveyors (except the loaded box conveyor and the shipping conveyor) start operating.
2. When the empty box reaches the empty Box Ready sensor position, the empty boxes conveyor stops.
3. When a raw part arrives at the input of the machining center, the robot automatically picks it up. One machining station creates a lid from the part, the other a base.
4. When the machining center is busy, their conveyors stop.
5. After processing, the lid is transferred to the lids conveyor, and the base is transferred to the bases conveyor.
6. The lid moves to its end position, where a sensor detects it.
7. Similarly, the base moves until it reaches a stopper, where another sensor detects it.
8. When the lid and base are in position the positioner clamp them.

9. Once both Base and Lid are clamped sensors are activated, the pick-and-place sequence begins.
10. The vertical cylinder (Z-axis) moves downward toward the lid.
11. When the Item Detected sensor is triggered, the grabber picks the lid.
12. The Z-axis returns upward, then the horizontal axis (X-axis) moves toward the base position.
13. At the target position, the Z-axis moves downward again, and the gripper releases the lid onto the base.
14. The Z-axis returns to its home position, followed by the X-axis returning to its initial position, ready for the next cycle.
15. During the return of the cylinders, the stopper lifts, and both conveyors restart.
16. The assembled part exits the system, and the Part Leaving sensor detects it, causing the stopper to return to its blocking position to hold the next incoming part.
17. When the assembled part exits the conveyor, the Pieces Counter sensor is activated to count it.
18. After every two completed parts, a box is filled. the empty boxes conveyor moves the box to the loaded boxes conveyor and continues operating until another empty box arrives.
19. The loaded boxes conveyor delivers the filled box to the loading conveyor and stops when the box comes out of it.
20. The loading conveyor moves the box until it reaches its end position, where the At Load sensor is triggered, and the conveyor stops immediately.
21. After stopping, the stacker crane begins its cycle the crane moves horizontally to the left. Then moves upward to lift the box. After that it returns to the central position.
22. The crane then moves to Storage Cell 1.
23. Upon reaching the target cell, it moves to the right, lowers the box into place, and returns to the center position.
24. The crane returns to its initial position, ready to lift the next box.
25. Storage is performed sequentially from cell 1 to cell 54.
26. When all storage cells are full, the entire production line stops automatically.

II.2.1.4.2. Transit Mode Sequence:

1. When the Start Transit button is pressed, the production line operates normally and produces assembled parts.
2. Parts are counted and packed into boxes as in Storage Mode.

3. After a box is filled, it is transferred to the load conveyor by the loaded boxes conveyor.
4. The stacker crane moves to the left and lifts the box from the Load Conveyor.
5. The crane returns to the center position and then moves to the right toward the Shipping Conveyor.
6. The crane lowers the box onto the unload conveyor.
7. The unload conveyor starts moving the box into the shipping conveyor and stops when the box leaves it.
8. The crane returns to its home position, ready for the next transfer cycle.
9. The shipping conveyor starts moving the box toward leaving area.
10. Production continues as long as the Transit Mode remains active.

II.2.1.4.3. Discharge Mode Sequence:

1. The Start Discharge button becomes available only when all storage cells are full.
2. When the button is pressed, the production line remains stopped.
3. The stacker crane starts unloading stored boxes sequentially from the warehouse and places them at the unload conveyor.

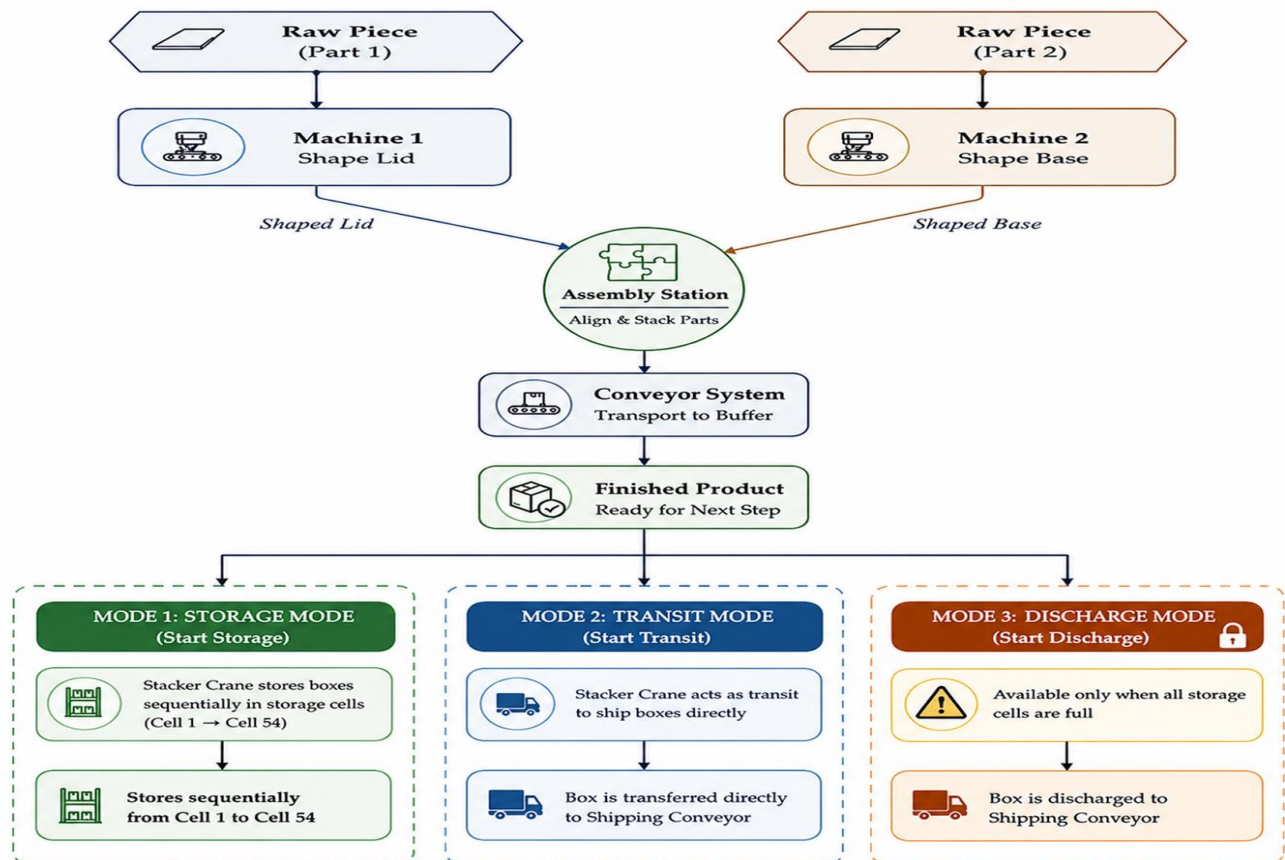


Figure II.1: Virtual production line flowchart

II.2.2. The Physical Production Line:

II.2.2.1. OULED MIMOUN:

OULED MIMOUN mills are a subsidiary of the ERIAD Group, a Public Economic Enterprise based in Sidi Bel Abbas [19].



Figure II.2: Sales Department of OULED MIMOUN [20]

II.2.2.2. AGRODIV Group:

The AGRO-INDUSTRIE Group, also known as AGRODIV, was established in February 2015 following the restructuring of the public commercial sector, as a result of the merger and absorption of three state owned enterprises: CEGRO, COJUB and TRAGRAL [21].



Figure II.3: AGRODIV logo [20]

II.2.2.3. MOLINO Company:

Molino is a manufacturer of equipment for the food industry founded in 1965 by Dr. Ömer Lütfi Alaybeyi with over 500 employees worldwide [22].

Following activities are in the scope of works of Molino [22]:

- Food City Complex
- Roots Pumps (Blowers)
- Civil Works and Steel Construction Buildings of Food Processing Plants
- Grain Storage Systems (Steel & Concrete)
- Seed Cleaning and Pulses Treatment
- Ship Loader
- Ship Loader (Mechanical)
- Combined Ship Loader & Unloader
- Ship Unloader (Pneumatic)
- Ship Unloader (Mechanical)

II.2.2.4. Transformation line of wheat:

Wheat processing lines are production lines, divided into sections. Each section is intended for a specific stage of production of flour and semolina from soft wheat. The sections are listed as follows:

- Grinding section.
- Cleaning section.
- Conveying section.
- Mixing and dosing section.
- Weighing and packaging section.

II.2.2.5. Weighing and packaging section:

II.2.2.5.1. Overview:

This section is the last stage of the process before storage. It is the primary focus of our study, where we aim to integrate an intelligent SCADA system to enhance monitoring and operational efficiency.

II.2.2.5.2. Components:

The following components constitute the main elements of the real production line and contribute to the execution of the different manufacturing and handling operations [20]:

- Gearmotors
- Dual speed motor

- Pneumatic cylinders
- Ultrasonic sensor
- Load cell (S type)
- Electromagnetic sensor
- Optical sensor
- Automatic blade
- Sewing machine

II.2.2.5.3. System operation sequence:

The following section describes the overall operation sequence of the system [20]:

1. When we press the SCyc5M button to select the 5 kg weight, or SCyc10M for 10 kg.
2. The first step is switching on the M0 gear motor that drives the screw conveyor, which carries the end product from the grinding line to the weighing and packaging station.
3. The top hopper is fitted with a level sensor C3 to detect the product level. The motor stops when the hopper is full.
4. Each exit is provided with a hatch controlled by a pneumatic cylinder VMA/VMB connected to electromagnetic sensors CTA/CTB respectively.
5. First, the discharge hatches are opened halfway for 5 seconds.
6. If the VTA/VTB hatch is open, the product will be directed to the weighing station (either A or B) to be weighed by load cells (C1/C2) at each station.
7. The first step is to run the motor at high speed (MA2/MB2) so that the hopper fills up quickly. The speeds of motors MA and MB are controlled by electromagnetic sensors CMA and CMB, respectively.
8. When weighing, if the C1/C2 load cell detects the weight being weighed reaching 80% of the target weight, the motors MA/MB will switch to low speed (MA1/MB1). When the weight is reached, the motors stop and the top hatches close.
9. If both load cell (CDFA/CDFB) stop both motors (MA/MB) at the same time, the gear motor (M0) will be stopped automatically.
10. A VA/VB actuator opens a hatch above the hopper at the end of weighing, it is connected to electromagnetic sensors (CVA/CVB).
11. Each hopper has a lower hopper below it, which ends in an opening through which the product falls into the bag.

12. The operator clamps manually. After two track-filling clamps are closed and no other track-filling clamps are closed, the filling is started. The dual-speed motor should also be stopped after the upper hatch and the discharge hatch are closed.
13. The system detects the end of the filling process when the C1/C2 load cell reads "0" weight and the clamps automatically open to release the bag. Next the lower hatch VA/VB will be closed.
14. We repeat the process, starting from the opening of the discharge hatch in the middle, if the level sensor detects that the upper hopper is not full, motor M0 restarts to refill it.
15. The sewing machine will be installed on the conveyor belt a proximity optical sensor CPH1 detects the presence of the bag at the entrance of the machine to start the MMC motor responsible for sewing. An automatic barrier BR prevents the entry of other bags. The bag leaving the machine is detected by a second proximity sensor CPH2 that stops the MMC and activates an automatic blade LA to cut the sewing thread. Next BR (BO: Open / BF: Closed) opens to let the next bag enter the sewing machine.
16. Finally, the bag is filled and sewn and ready for transportation to the warehouse.

The flowchart below illustrates how the production line works.

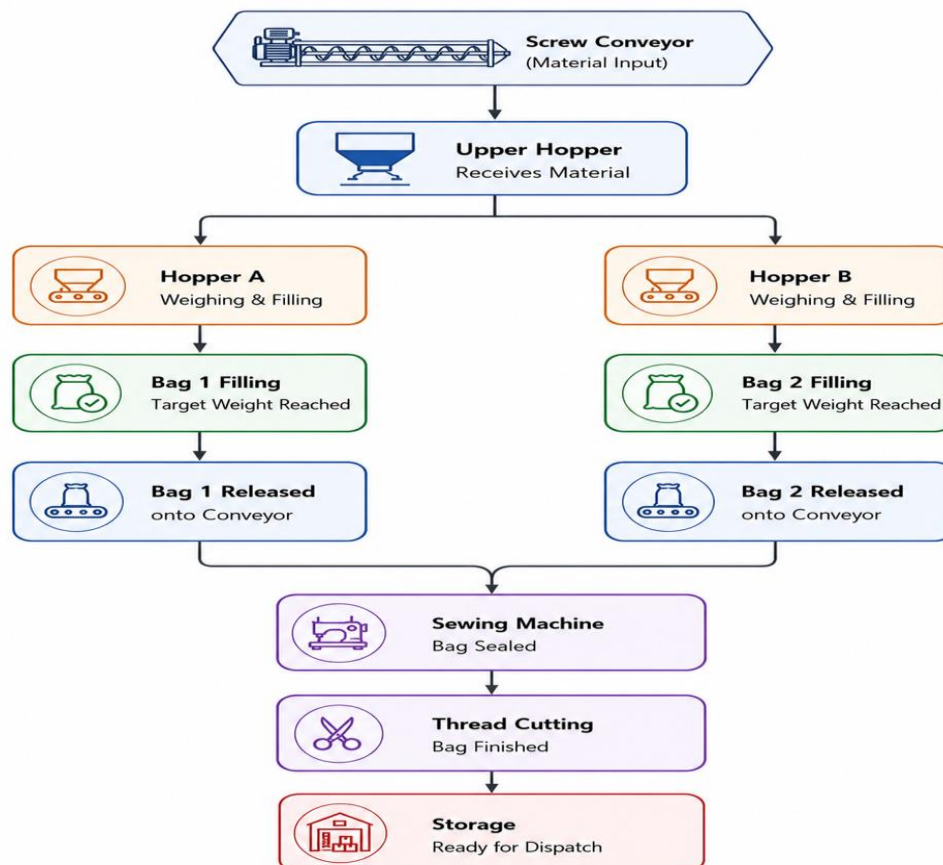


Figure II.4: Real production line flowchart

II.2.2.5.4. I/O Variables Table:

The following tables show the input and output variables used for monitoring and controlling the production process [20]:

II.2.2.5.4.1. Inputs:**Table II.3:** Real production line PLC Inputs [20].

Name	Data Type	Address	Comment
SCyc5Kg	Bool	%I0.0	Start Cycle 5kg
SCyc10Kg	Bool	%I0.1	Start Cycle 10kg
Stop	Bool	%I0.2	Stop
Emergency	Bool	%I0.3	Emergency Stop
Lvl_Sensor_1	Bool	%I0.4	Level Sensor
Cyl_A_Open	Bool	%I0.5	Detect The Cylinder A Open
Cyl_A_Mid	Bool	%I0.6	Detect The Cylinder A Mid
Cyl_A_at_End	Bool	%I0.7	Detect The Cylinder A full open
Cyl_B_Open	Bool	%I1.0	Detect The Cylinder B Open
Cyl_B_Mid	Bool	%I1.1	Detect The Cylinder B Mid
Cyl_B_at_End	Bool	%I1.2	Detect The Cylinder B full open
Ai_Scale_A	Int	%IW0	Weigh Scale analog input A
Ai_Scale_B	Int	%IW1	Weigh Scale analog input B
M1A_Fast	Bool	%I1.3	Detect Motor 1 Fast
M1A_Slow	Bool	%I1.4	Detect Motor 1 Slow
M1A_Off	Bool	%I1.5	Detect Motor 1 off
M2B_Fast	Bool	%I1.6	Detect Motor 2 Fast
M2B_Slow	Bool	%I1.7	Detect Motor 2 Slow
M2B_Off	Bool	%I2.0	Detect Motor 2 off
Cyl_A_Low_Open	Bool	%I3.0	Detect lower depot A cylinder
Cyl_B_Low_Open	Bool	%I3.1	Detect Lower depot B cylinder
Tong_A1_Closed	Bool	%I3.4	Detect depot A first tong
Tong_A2_Closed	Bool	%I3.5	Detect depot A sec tong
Tong_B1_Closed	Bool	%I3.6	Detect depot B first tong
Tong_B2_Closed	Bool	%I3.7	Detect depot B sec tong

CPH1	Bool	%I4.4	Optical proximity sensors of the sewing machine to detect the coming bag
CPH2	Bool	%I4.5	Optical proximity sensors of the sewing machine to detect the leaving bag
Cyl_A_up_Open	Bool	%I4.6	Detect Depot A upper cylinder
Cyl_B_up_Open	Bool	%I4.7	Detect Depot B upper cylinder

II.2.2.3.4.2. Output:**Table II.4:** Real production line PLC Outputs [20].

Name	Data Type	Address	Comment
Open_Cyl_B_Mid	Bool	%Q0.2	Set Cylinder B to the Mid
Open_Cyl_A_Mid	Bool	%Q0.1	Set Cylinder A to the Mid
M0	Bool	%Q0.0	Transport the semolina to the weighing and packaging station
Open_Cyl_B_to_end	Bool	%Q0.4	Set Cylinder B to the End
Open_Cyl_A_to_end	Bool	%Q0.3	Set Cylinder A to the End
Run_M1A_Slow	Bool	%Q0.5	Run motor A Slow
Run_M1A_Fast	Bool	%Q0.6	Run Motor A Fast
Run_M1B_Slow	Bool	%Q0.7	Run motor B Slow
Run_M1B_Fast	Bool	%Q1.0	Run Motor B Fast
Open_DepotA_Up_Cyl	Bool	%Q1.1	Open Depot A upper cyl
Open_DepotB_Up_Cyl	Bool	%Q1.2	Open Depot B upper cyl
Open_DepotA_Lw_Cyl	Bool	%Q1.3	Open Depot A Lower cyl
Open_DepotB_Lw_Cyl	Bool	%Q1.4	Open Depot B Lower cyl
Tong_A1_Cyl	Bool	%Q1.5	Detect the Tong A1 of Depot A cylinder
Tong_B1_Cyl	Bool	%Q1.6	Detect the Tong A1 of Depot A cylinder
MS	Bool	%Q1.7	Sewing Motor
Barrier	Bool	%Q2.0	Command the barrier
Blade	Bool	%Q2.1	Command the blade
MPT	Bool	%Q2.2	Command the Sewing motor
Tong_A2_Cyl	Bool	%Q2.3	Detect the Tong A2 of Depot A cylinder
Tong_B2_Cyl	Bool	%Q2.4	Detect the Tong B2 of Depot B cylinder

II.3. Development Tools and Software Environment:

II.3.1 TIA Portal V16:

The Siemens Totally Integrated Automation Portal (TIA Portal V16) is an integrated engineering framework for configuration, programming and commissioning industrial automation systems[23]. It provides a unified development environment for engineers to design and manage PLC, HMI, industrial networks and drive systems in one software platform. TIA Portal version 16 offers higher performance, better usability, and improved support for advanced automation functions, enabling efficient system integration and reducing engineering time. With TIA Portal, all automation tasks are gathered in one environment, providing consistency in project development, simplifying diagnostics and improving maintainability throughout the entire automation lifecycle [24].

II.3.2. WinCC (Integrated / Runtime):

SIMATIC WinCC (Windows Control Center) is a SCADA and HMI software from Siemens. It is used for real-time control, monitoring and visualization of industrial processes [25]. WinCC is a widely used solution in industrial automation, offering sophisticated capabilities for process visualization and real-time data monitoring. It has a high compatibility with a variety of operating systems and databases, and easy to connect to PLCs and other industrial devices. This integration allows operators to efficiently monitor and control industrial processes from one single, central interface. WinCC also provides flexible configuration and customization options, allowing for the development of monitoring systems customized to specific industrial needs. WinCC is one of the most popular SCADA and HMI solutions for modern automation systems thanks to a wide range of features and reliable performance [26].

II.3.3. PLCSIM Advanced:

SIMATIC S7-PLCSIM Advanced is advanced simulation software from Siemens to test and validate PLC programs in a virtual environment. It allows for the realistic simulation of SIMATIC S7 controllers enabling engineers to execute, monitor and debug automation code without the need for physical hardware [27]. The software has 2 mains communication interfaces, i.e. PLCSIM (Local) and PLCSIM Virtual Ethernet Adapter. The Local communication is further divided into: Local communication via Softbus and Local communication via TCP/IP [28].

II.3.4. Factory I/O:

Factory I/O is a 3D factory simulation for learning automation technologies. Designed to be easy to use, it allows to quickly build a virtual factory using a selection of common industrial parts [29]. This software allows students and engineers to quickly prototype and test different factory layouts in a practical way. Factory io can simulate an industrial process, showing how a system behaves in real life

[30]. Very flexible for learning and testing as it can connect to real PLC controllers as well as virtual ones such as PLCSIM.

II.3.5. MATLAB 2023a:

MATLAB (Matrix Laboratory) is a high-level language and environment for numerical computation, data analysis, algorithm development and system modeling, developed by MathWorks. It offers an interactive platform that combines computation, visualization and programming in one environment, and is widely used in academic research [31].

MATLAB is a software platform extensively utilized in industrial automation for data analysis, algorithm development, and system modeling. It offers a complete suite of functions and specialized toolboxes that simplify the implementation of advanced analytical techniques such as predictive maintenance and root cause analysis [32].

II.3.6. Node-RED:

Node-RED is a visual programming platform originally developed by IBM that enables users to create applications by wiring together predefined nodes using a drag-and-drop interface. These nodes can be hardware devices, software functions, web services or APIs. Applications are deployed with one click and data flows are created by linking nodes together. Built on Node.js, Node-RED inherits its event-driven and non-blocking model, making it suitable for real-time industrial and IoT applications. Its intuitive graphical environment makes development easier and less complex than traditional programming [33].

II.4. Communication Architecture Between Tools:

In this project, a multi-protocol communication strategy was chosen to enable seamless data exchange and a high interoperability among heterogeneous software platforms:

II.4.1. OPC UA:

Open Platform Communications-Unified Architecture is an open, cross-platform industry standard for secure, reliable and scalable communication between devices, systems and applications in automation and IoT environments. It is often referred to as a communication protocol but it is in fact a comprehensive industrial standard which defines a complete communication framework and a semantic information model [34].

II.4.2. S7 Protocol:

The s7 protocol is a proprietary industrial communication protocol by Siemens used for data exchange and control communication between SIMATIC S7 PLCs and engineering or supervisory systems like HMIs and SCADA platforms [35].

II.4.3. HTTPS:

HTTP (HyperText Transfer Protocol) is a stateless application-level protocol to communicate between client and server over the web. It uses a request–response model where clients can send requests to identified resources using URIs and servers provide responses containing the requested data. The secure version of HTTP is HTTPS. HTTPS uses TLS to provide confidentiality and integrity of transmitted data by encrypting the communication [36].

II.4.4. Web Socket Protocol:

The WebSocket protocol enables full-duplex communication over a single long-lived TCP connection. It starts with an HTTP handshake and then upgrades the connection to allow for ongoing two-way data exchange between client and server. This kind of design makes WebSocket particularly suitable for real-time applications such as monitoring systems, IoT platforms and industrial dashboards, where low latency and continuous data flow are necessary [37].

II.5. Application to Reel and Virtual Lines:

To achieve a reliable and high-performance industrial monitoring system, we developed two Intelligent SCADA Architectures that integrates computational analysis with industrial automation. The systems are designed to perform three core functions:

- Real-time data synchronization between the physical plant and the computational intelligence layer.
- Predictive condition classification using a trained neural network.
- Fault diagnosis to rank the most probable failure root causes.

II.5.1. Real-Time Integration via OPC UA:

The OPC UA protocol is the backbone of our system, offering a secure, bi-directional, real-time connection between the TIA Portal (S7-1500 PLC) and the MATLAB environment. This allows SCADA system to read physical parameters from the field instrumentation like Temperature (T), Current (I), Vibration (V) and Speed (N), and write back control commands and diagnostic results with low latency ensuring immediate response to critical events.

If the predictive engine predicts a failure, the SCADA system will automatically activate an emergency interlock to stop the motor drive through the bridge to OPC-UA. The hardware level closed loop response inhibits propagation of mechanical damage that would result from continued operation in the presence of a critical fault condition. A scenario whose financial and safety consequences, including considerable maintenance costs, production losses and possible risk for human safety, are widely reported in the literature [13].

II.5.2. Data Preprocessing: Z-Score Normalization:

The four sensor signals are on very different numerical scales. For example motor speed is around 1400 RPM, vibration is around 1-5 mm/s and current is around 9-30 A. If these values are used directly as inputs to the neural network, the larger numbers (e.g., speed) will be given more weight and the smaller ones (e.g., vibration) will be ignored, although vibration may contain important fault information. To fix this problem, we apply Z-Score Normalization to every input before it enters the network:

$$X_{norm} = \frac{X_{input} - \mu}{\sigma} \dots\dots\dots (II.1)$$

where μ is the mean (average) and σ is the standard deviation of each sensor, both calculated from the training dataset. At this point all four signals are on the same scale and centered around zero so the network can treat them equally. This is a standard and well-recommended step prior to training any neural network on sensor data [38].

II.5.3. Neural Network: Architecture and Training:

The core of our system is a Feed-Forward Neural Network (FFNN), also called a Multilayer Perceptron (MLP). This type of network is one of the most common and well-tested tools in machine learning. It has been successfully used in many industrial applications for classifying the condition of rotating machines [39].

The network is made up of layers of simple units called neurons. Each neuron takes a set of inputs, multiplies them by learned values called weights, adds a bias, and then passes the result through an activation function [39].

While training, the network learns the correct weights by seeing many examples, comparing its output to the correct answer, and adjusting its weights to reduce the error [39]. This process of adjustment is known as backpropagation and uses an optimization method called Adam in our case.

II.5.4. Network Structure:

The network we built has the following structure:

Table II.5: Network Structure

Parameter	Details
Input	4 values: Temperature, Current, Vibration, Speed
Hidden layers	4 layers: 128 → 64 → 32 → 16 neurons
Activation function	ReLU — used after every hidden layer
Extra techniques	Batch Normalization (×3) + Dropout (30%)

Training method	Supervised learning on generated data
Optimizer	Adam (learning rate = 0.001)
Early stopping	Stops training when accuracy stops improving
Output	3 values with Softmax → Normal / Warning / Critical

II.5.5. Network shape:

The network starts wide (128 neurons) and narrows ($64 \rightarrow 32 \rightarrow 16$) before the output. This structure forces the network to learn the most important patterns from the data gradually.

The full structure is:

$$\text{Input}(4) \xrightarrow{BN} FC(128) \xrightarrow{ReLU, BN} FC(64) \xrightarrow{ReLU, Drop, BN} FC(32) \xrightarrow{ReLU} FC(16) \xrightarrow{ReLU} FC(3) \xrightarrow{Softmax}$$

The three output classes are:

- Normal (0): The motor is working correctly. No action needed.
- Warning (1): Something is starting to go wrong. The technician should check the motor soon.
- Critical (2): A serious fault is detected. The system immediately stops the motor to prevent damage.

II.5.6. Dataset Generation and Training:

One of the major challenges in industrial machine learning is that real fault data is very hard to obtain, faults are rare. There was no real operational data available so we wrote a data generation script. The script simulates the behavior of the four sensor signals under each condition and generates a realistic dataset that can be used to retrain and improve the model when real data becomes available in the future. The dataset has three types of records:

- Normal records: Sensor values were generated around the known healthy operating point of the motor, with small random noise added to simulate real measurement variation.
- Warning records: One or more sensors were gradually moved away from the normal range to simulate five early-stage fault types: Bearing wear, Cooling degradation, Overload, Phase loss, and Shaft misalignment.
- Critical records: Sensor values were pushed beyond the safe operating limits to simulate active, serious fault conditions for each of the five fault types.

The number of records for each class was kept equal to make sure the network learns all three classes well. And the training was done in MATLAB with the following settings:

- Data split: 70% for training, 15% for validation, 15% for testing.
- Optimizer: Adam with a starting learning rate of 0.001, reduced by half every 80 epochs.

- Batch size: 128 samples per update step.
- Early stopping: If the validation accuracy does not improve for 15 checks in a row (one check every 50 training steps), training stops automatically. This saves time and prevents the network from memorizing the training data.
- Performance check: After training, accuracy, precision, recall, and F1-score were measured on the test set. A confusion matrix was also plotted to see which classes were sometimes mixed up.

II.6. Fault Diagnosis: Finding the Root Cause:

II.6.1. Weight-Based Fault diagnosis Scada:

The neural network tells us how serious the problem is (Normal/Warning/Critical), but it does not tell us what caused it. For a maintenance technician, knowing the cause is just as important as knowing that there is a problem. Without this information the technician would have to check every part of the motor. This wastes time and delays the repair. To tackle this, we added a second step after the neural network: a Weighted Fault Diagnosis module that examines the sensor readings and outputs a ranked list of the most likely fault causes.

II.6.1.1. Fault Weight Table:

The four sensors are affected differently by each type of faults. For example a bearing fault mainly leads to an increase in vibration, whereas a cooling failure mainly leads to an increase in temperature [40]. Using these known relationships, we constructed a weight table in which each row corresponds to one fault type and each column indicates the frequency of the specific fault type impacting each sensor.

Table II.6: Fault sensor weight table. Each row sums to 1.0

Fault Type	Temperature	Vibration	Current	Speed
Bearing Fault	0.05	0.85	0.05	0.05
Motor Overload	0.15	0.05	0.75	0.05
Cooling Failure	0.85	0.05	0.05	0.05
Rotor Imbalance	0.05	0.75	0.05	0.15
Phase Loss	0.05	0.05	0.85	0.05
Shaft Misalignment	0.05	0.65	0.05	0.25

For example, Bearing Fault has a weight of 0.85 for vibration, because bearing problems are most clearly visible in vibration signals, as established in the motor fault diagnosis literature [40].

II.6.1.2. How the Priority Score is Calculated:

When an alarm is triggered, the system calculates a Priority Score for each of the 6 possible faults:

$$S_f = 0.8 \times C_f + 0.2 \times H_f \dots\dots\dots (II.2)$$

where:

- **Cf-Confidence Score (80%):** The confidence score Cf was formulated by the authors based on the weighted deviation principle, where the deviation of each sensor from its normal operating range is weighted based on the fault-sensor relationship matrix. This measures how much each sensor has moved from its normal range weighted by fault table. If a sensor is within its normal range, it scores zero. The higher the score, the farther it is from the normal limit. The confidence score for the fault f is:

$$C_f = \sum_{s \in \{T,V,I,N\}} W_{f,s} \cdot d_s \dots\dots\dots (II.3)$$

where ds is the deviation of sensor s from its normal range, scaled between 0 and 1.

- **Hf-History Score (20%):** This checks how many times fault f has occurred before on this motor, using the saved alarm log. The more often a certain fault has occurred in the past the more likely it is to occur again. This allows for a more appropriate diagnosis for the particular machine being watched.

The system sorts all 6 faults by their Priority Score from highest to lowest and sends this ranked list to the HMI. The technician can then start checking the most likely cause first, saving inspection time.

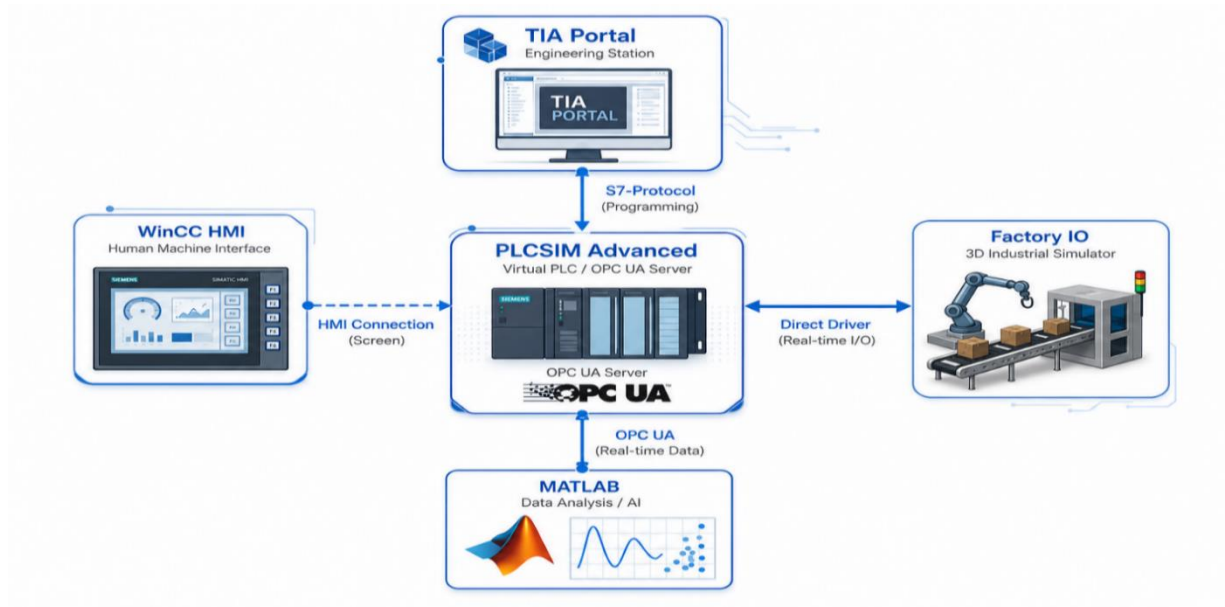


Figure II.5: weight-based fault diagnosis communication architecture

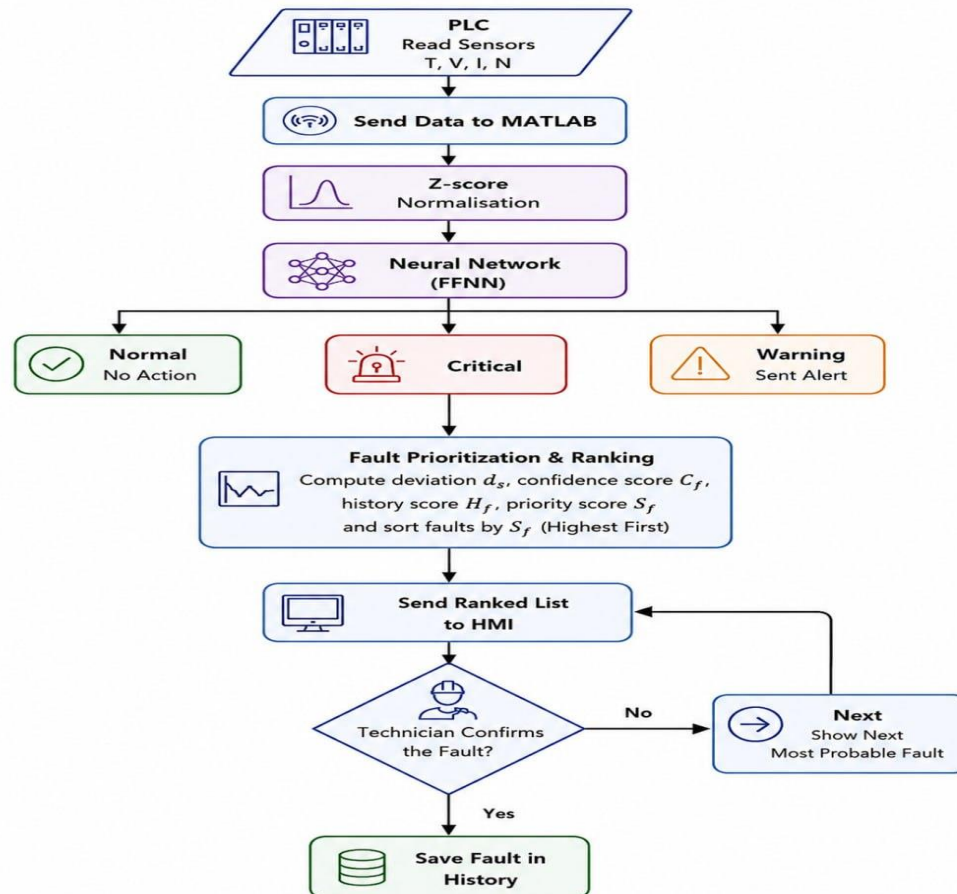


Figure II.6: Weight-Based fault diagnosis architecture flow chart

II.6.2. AI-Assisted Fault Diagnosis SCADA:

The second fault diagnosis architecture replaces the weighted fault diagnosis approach with an AI-driven diagnostic system based on the Gemini large language model. Instead of calculating fault priority scores from predefined sensor weights, the diagnosis process relies entirely on the AI assistant's interpretation of sensor measurements, operating conditions, and historical fault records.

Upon the neural network classifier's detection of a Critical condition, the system automatically queries the Gemini API with a structured prompt that contains the pre-fault sensor trends and the historical fault log.

The AI model provides a contextual diagnosis that explains the observed sensor behavior, identifies the most probable root cause, and proposes a ranked list of corrective actions. This output is then displayed directly on the SCADA dashboard, giving the technician actionable guidance without requiring them to interpret raw sensor data.

The language model reasons over particular trend patterns leading to the fault, instead of using a static rule base. For example, a simultaneous decrease of the supply pressure and increase of the flow rate are interpreted as an internal seal bypass or external leakage, which matches the air leak fault class detected by the classifier.

The diagnosis workflow is based on human-in-the-loop confirmation model. The AI output is reviewed and the equipment inspected. The technician then reports the confirmed fault in a natural language conversation with the AI assistant embedded in the dashboard. The assistant extracts the fault type from the technician's message, asks for an explicit confirmation and, if approved, forwards the confirmed fault label to the monitoring system for logging. This way the final record is the ground truth from the technician and not just an automated inference.

All fault events are logged in csv files, a history Log file for each system. Each Log includes the start and end times of the fault event for downstream analysis like mean time between failures and mean time to repair.

The AI-assisted fault diagnosis communication architecture is shown below:

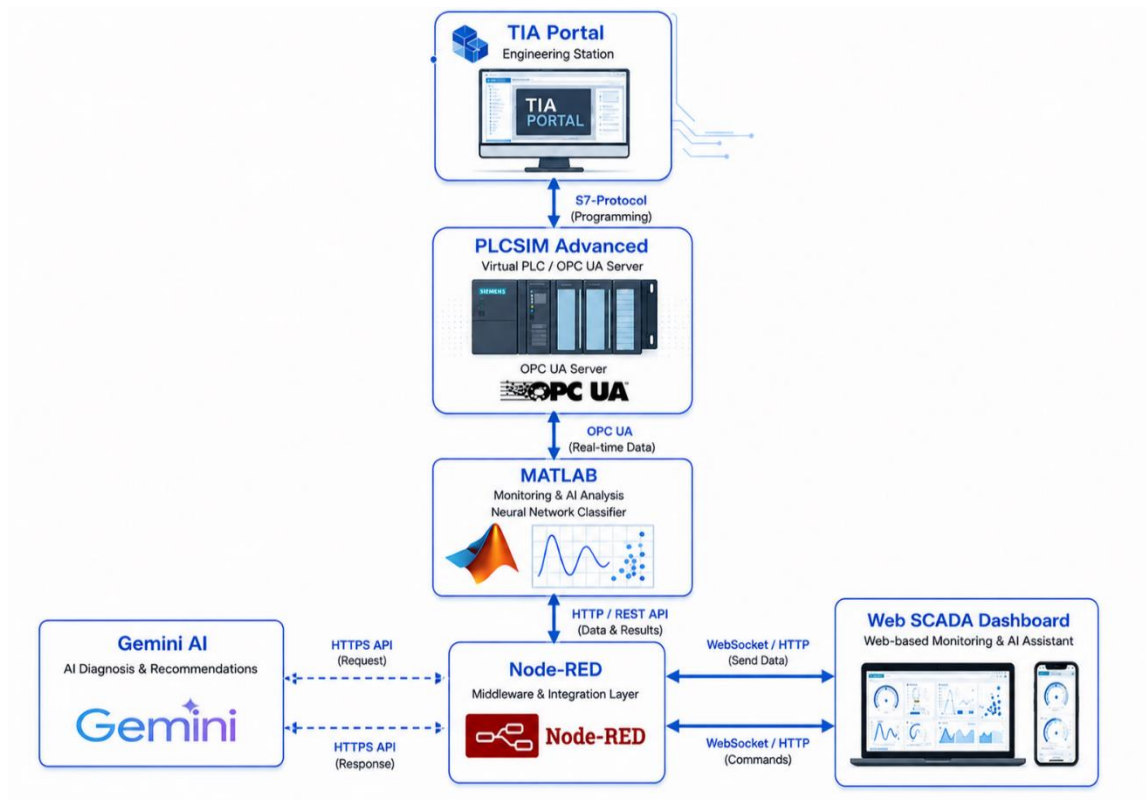


Figure II.: AI-Assisted fault diagnosis communication architecture

The AI-assisted fault diagnosis workflow is shown below:

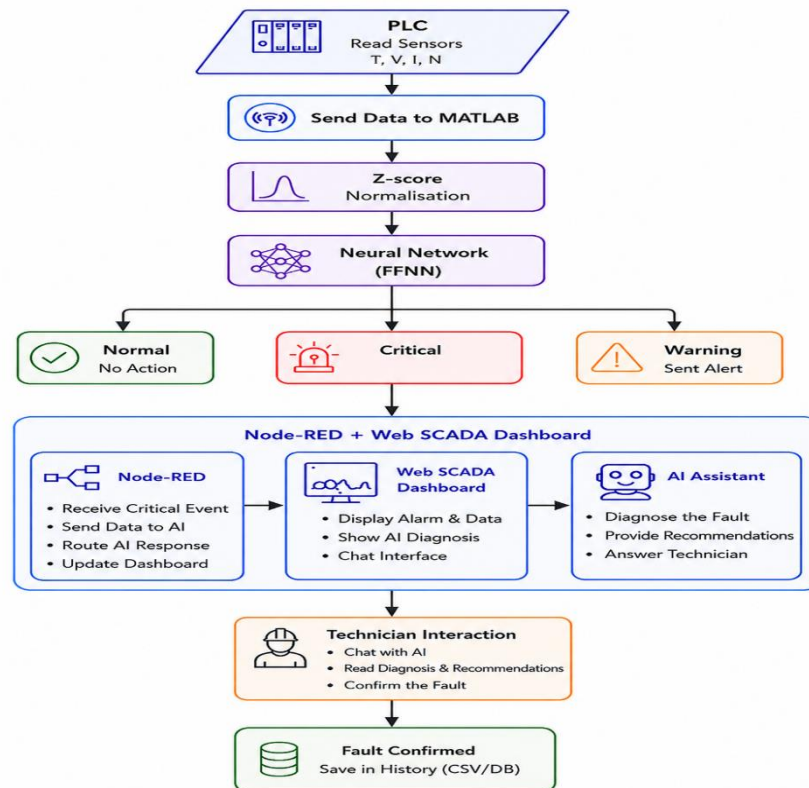


Figure II.7: AI-Assisted Fault Diagnosis flow chart

II.7. Conclusion:

In conclusion, this chapter has provided a comprehensive overview of the systems developed and the technological framework deployed throughout this project. By configuring both a simulated environment in Factory I/O and a real industrial production line, a robust platform for testing was successfully established. Furthermore, the implementation of the intelligent supervision system on top of the SCADA infrastructure has been detailed, demonstrating the seamless integration of communication protocols, data preprocessing, neural network training, and the final fault diagnosis methodology.

With the design of each component and the operational mechanics of the overall system now fully explained, the practical framework of this project is complete. Consequently, the ground is fully prepared to evaluate the efficiency of these implementations. The next chapter will therefore be dedicated to presenting the experimental results, performance analysis, and the final validation of the developed diagnostic system.

Chapter III:

Results and Discussion

III.1. Introduction:

After the design and implementation framework of the system presented in the previous chapter, in this third chapter, the results obtained from the developed predictive maintenance system are presented and discussed. First, we discuss the quality of the classification accuracy of the neural network model, the influence of using synthetic datasets, and the effectiveness of the weight-based diagnostic methodology.

In addition, the contribution of the AI assistant in enhancing the fault analysis in the SCADA environment is highlighted with respect to its ability to provide explanations, actionable maintenance recommendations, and key insights from the historical fault data. The aim of this chapter is to analyze the system in detail in terms of its operational efficiency.

III.2. Dataset visualization:

The two datasets follow the same structural design philosophy. Each of them captures four important process signals that were selected because of their physical relevance to the most common failure modes observed in their respective systems. The monitored signals for the induction motor are temperature ($^{\circ}\text{C}$), current (A), vibration (mm/s) and rotational speed (RPM). The signals for the pneumatic cylinder are pressure (bar), cycle time (ms), temperature ($^{\circ}\text{C}$), and air flow rate (L/min).

Both datasets are organized around three operational classes:

- Normal: representing stable operation under varying load conditions, where all signals remain within acceptable bounds.
- Warning: representing early-stage degradation corresponding to five distinct fault fingerprints per system. Each fault produces a characteristic signal pattern while remaining partially overlapping with the Normal class boundaries.
- Critical: representing severe fault conditions where one or more signals significantly exceed safe thresholds, requiring immediate intervention.

The figures below present the four monitored signals of each system across the full 90-day period, color-coded by operational class:

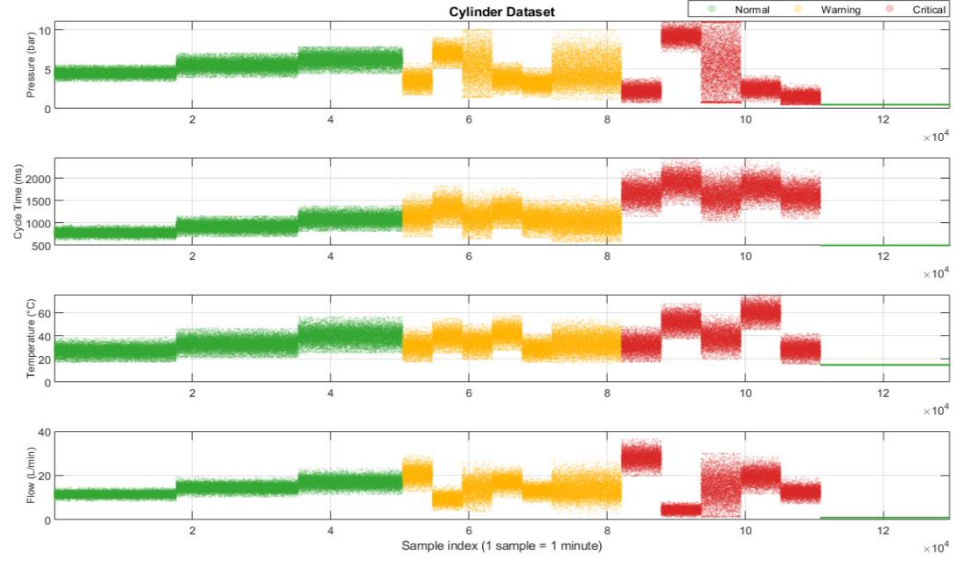


Figure III.1: Pneumatic Cylinder Dataset Curve

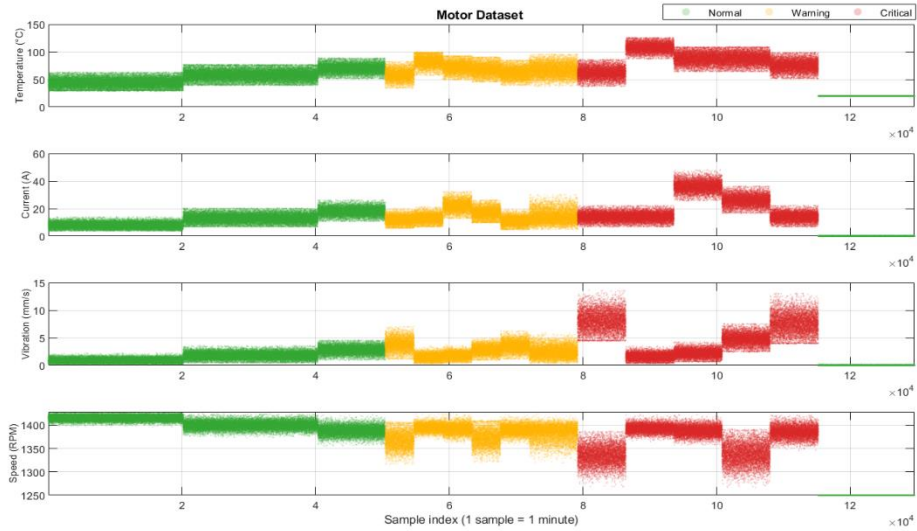


Figure III.2: Induction Motor Dataset Curve

III.3. Neural network results:

This section presents the results obtained from training and testing the neural network classifiers developed for the induction motor and the pneumatic cylinder. The two systems had the same network architecture, the only difference being the type of input signals. Performance evaluation is done using training curves, accuracy evolution and confusion matrix analysis.

III.3.1. Architecture & parameters:

Both classifiers have the same feedforward neural network architecture in order to achieve a balance between model capacity and generalization. The network has four normalized input signals and

maps them to one of three output classes: Normal, Warning or Critical. The architecture is composed of four fully connected layers with decreasing number of neurons, combined with batch normalization layers to stabilize the training, a dropout layer to reduce overfitting and ReLU activation functions to introduce non linearity.

The complete network structure and the number of neurons in each layer is shown in the figure below.

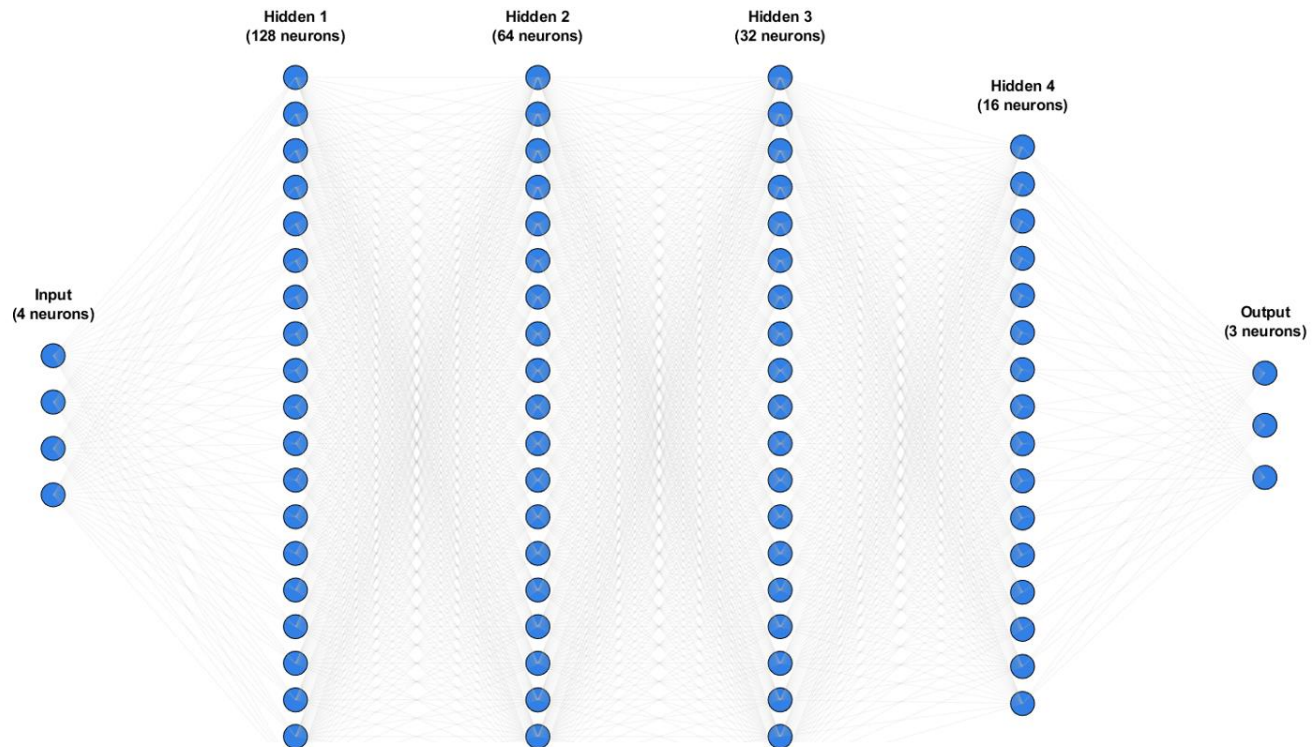


Figure III.3: Neural network architecture

III.3.2. Training progress:

The training was tracked by the accuracy and loss of training and validation sets simultaneously. We applied early stopping to prevent overfitting, which automatically halts training when the validation performance ceases to improve.

The results for both systems are shown in the figures below:

III.3.2.1. Accuracy curves:

Monitors percentage of correctly classified samples at each validation checkpoint The converging training and validation curves confirm good generalization for both systems.

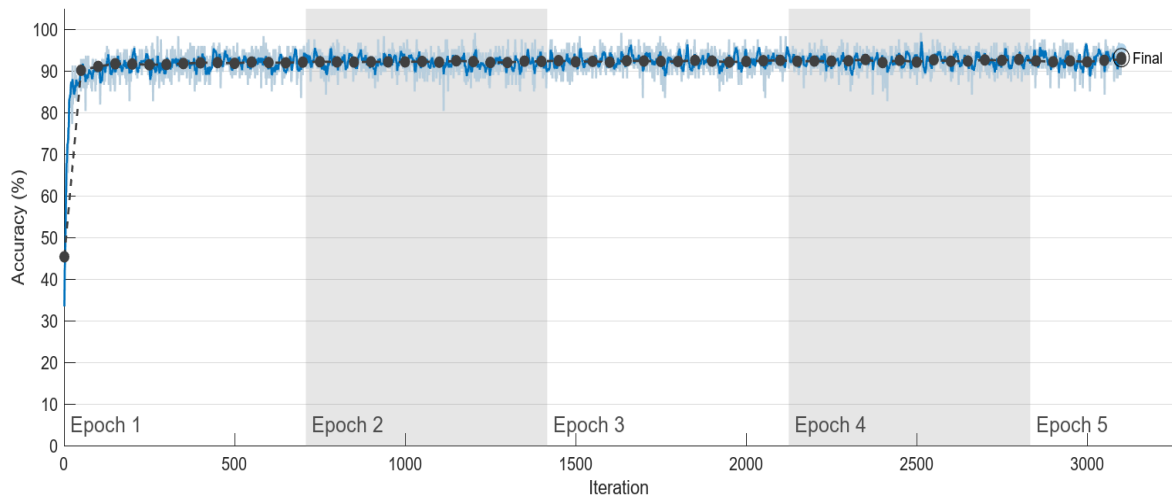


Figure III.4: Accuracy curve of the pneumatic cylinder

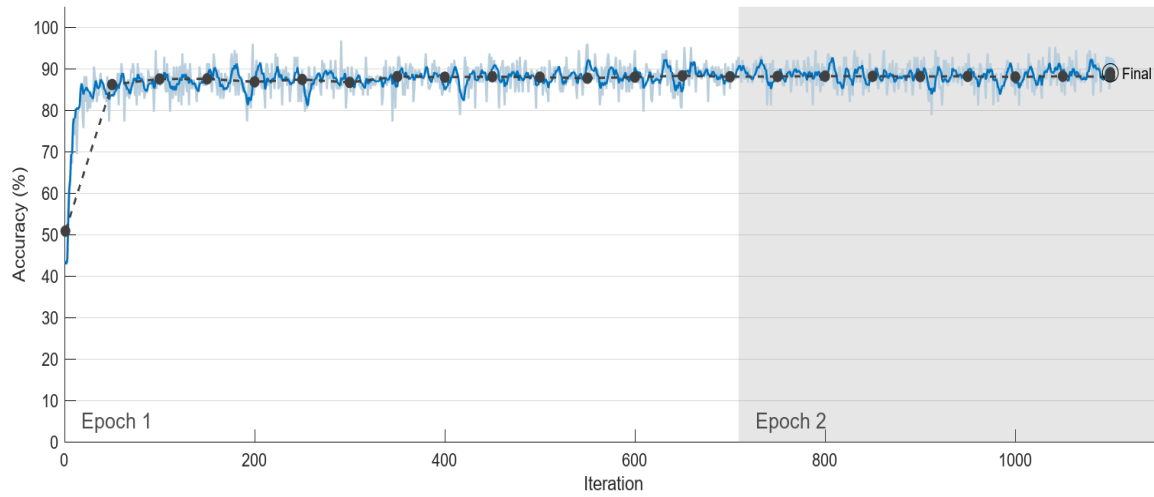


Figure III.5: Accuracy curve of the motor

III.3.2.2. Loss curve:

Calculate cross-entropy loss of predicted and true labels. The loss on both sets is steadily decreasing, which confirms that the network is learning meaningful patterns instead of memorizing the training data.

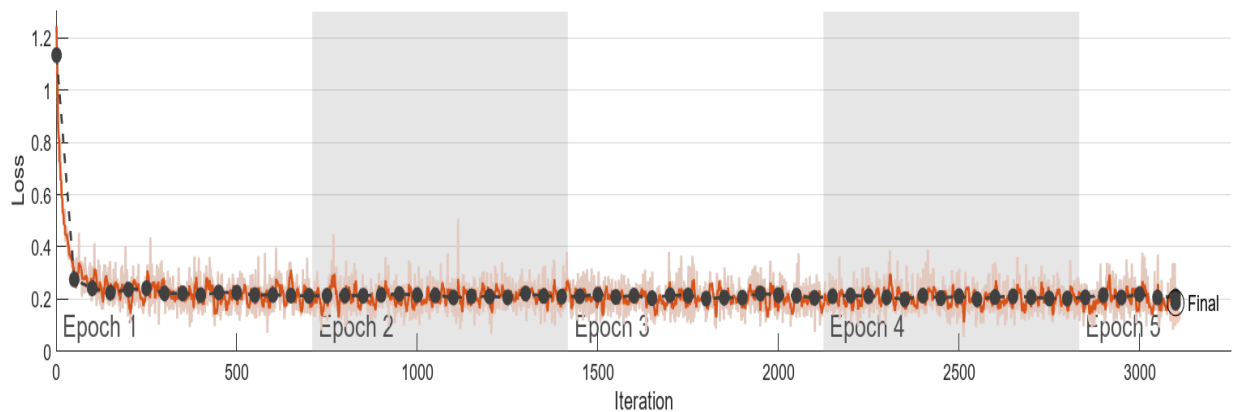


Figure III.6: Loss curve of the pneumatic cylinder

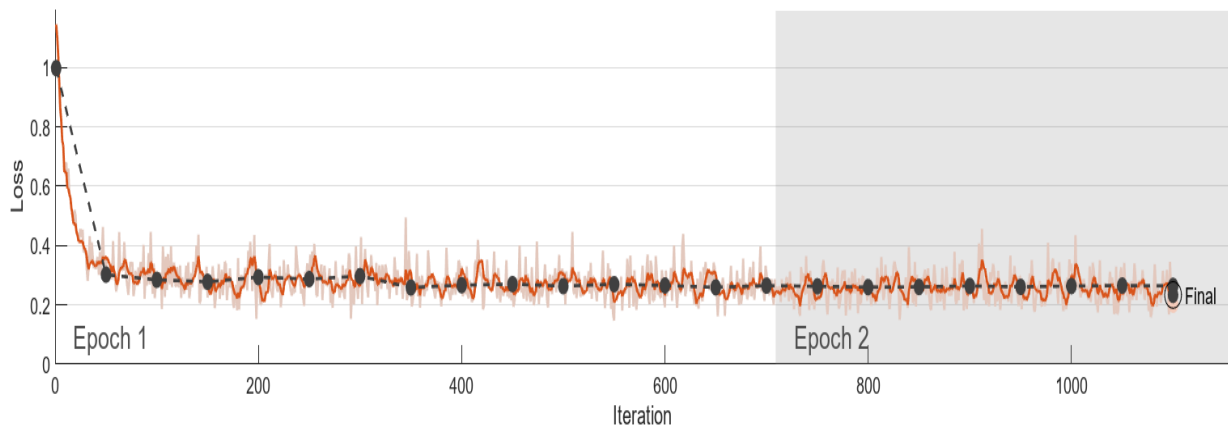


Figure III.7: Loss curve of the induction motor

III.3.3. Confusion matrix:

Compares predicted labels against true labels across all three classes. Rows represent true classes, columns represent predicted classes. Diagonal elements are correct predictions; off-diagonal elements are misclassifications [41].

Both systems show the same pattern, Normal and Critical are well classified, Warning has the highest confusion due to its transition state between the two states. The overall accuracy of induction motor is 90.05% and pneumatic cylinder is 93.61%.

Confusion Matrix

		Predicted Class		
		Normal	Warning	Critical
True Class	Normal	10258	134	0
	Warning	708	3909	115
	Critical	28	258	4030

Figure III.8: Pneumatic cylinder confusion matrix

		Predicted Class		
		Normal	Warning	Critical
True Class	Normal	9046	669	12
	Warning	844	3279	149
	Critical	11	249	5181

Figure III.9: Motor confusion matrix

III.3.4. Precision / Recall / F1:

The developed diagnostic system is tested using three standard evaluation metrics i.e. Precision, Recall and F1-score. These metrics measure the accuracy of the fault predictions, the ability to detect actual faults and the trade-off between both. These metrics give a comprehensive understanding of how effective the classifier is. The mathematical expressions used to calculate these metrics are given below [41].

$$\textbf{Precision} = \frac{TP}{TP+FP} \dots\dots\dots (\text{III.4})$$

$$\textbf{Recall} = \frac{TP}{TP+FN} \dots\dots\dots (\text{III.7})$$

$$\textbf{F1} = 2 \times \frac{\textbf{Precision} \times \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}} \dots\dots\dots (\text{III.6})$$

Pneumatic Cylinder:

Table III.1: Classification Results for the Pneumatic Cylinder NN

Class	Precision	Recall	F1
Normal	93.3%	98.7%	95.9%
Warning	90.9%	82.6%	86.6%
Critical	97.2%	93.4%	95.3%

Induction Motor:

Table III.2: Classification Results for the Induction Motor NN

Class	Precision	Recall	F1
Normal	91.4%	93%	92.2%
Warning	78.1%	76.8%	77.4%
Critical	97.0%	95.2%	96.1%

The warning is always the lowest F1 in both systems. Critical detection results in the highest precision which is the most safety-relevant outcome for a predictive maintenance application.

III.4. SCADA monitoring and fault diagnosis:

III.4.1. Weight-Based fault diagnosis:

III.4.1.1. Diagnosis example:

This example illustrates how the fault diagnosis algorithm works in a step-by-step manner based on the sensor readings captured in the Bearing Fault test scenario:

Normal operating limits:

Table III.3: Normal Operating Ranges and Alert Thresholds for the Induction Motor

Sensor	Normal Low	Normal high	Alert Threshold
Temperature	35	90	100
Vibration	0.5	5	6
Current	6	28	32
Speed	1350	1425	1320

Sensor readings at the moment of the alarm:

Table III.4: Real-Time Sensor Readings at Alarm Event

Sensor	Measured Value	Status
Temperature	62	Normal
Vibration	8	Above limit
Current	14	Normal
Speed	1335	Below limit

Step 01: Compute the deviation ds for each sensor:

The deviation measures how far each sensor has gone beyond its normal limit, on a scale from 0 (inside normal range) to 1 (reached the alert threshold):

$$d_s = \min \left(1.0, \frac{\text{over}}{\text{ref}} \right) \dots\dots\dots (\text{II.7})$$

Where:

- **Over:** how much the sensor went over the normal limit,
- **Ref:** the difference between the normal limit and the alert threshold.

Temperature: 62 °C is within the normal range (35 – 90 °C) → dT = 0

Vibration: 8.0 mm/s exceeds the upper limit of 5.0 mm/s

$$Over = 8.0 - 5.0 = 3.0$$

$$Ref = 6.0 - 5.0 = 1.0$$

$$dV = \min\left(1.0, \frac{3.0}{1.0}\right) = 1.0$$

Current: 14.0 A within normal range (6 – 28 A) → dI = 0

Speed: 1335 is less than the lower limit of 1350 RPM

$$Over = 1350 - 1335 = 15$$

$$Ref = 1350 - 1320 = 30$$

$$dN = \min\left(1.0, \frac{15}{30}\right) = 0.5$$

Deviation vector:

$$d = [dT, dV, dI, dN] = [0, 1, 0, 0.5]$$

Step 02: Compute the confidence score Cf for each fault

Each fault has a weight for each sensor from the fault weight table. The confidence score is:

$$C_f = \sum W_{f.s} \times d_s \dots\dots\dots (II.8)$$

Table III.5: Confidence Score Calculation for Fault Diagnosis

	Fault Type	Calculation	Cf
1	Bearing Fault	(0.05)(0) + (0.85)(1.0) + (0.05)(0) + (0.05)(0.5)	0.875
2	Motor Overload	(0.15)(0) + (0.05)(1.0) + (0.75)(0) + (0.05)(0.5)	0.075
3	Cooling Failure	(0.85)(0) + (0.05)(1.0) + (0.05)(0) + (0.05)(0.5)	0.075
4	Rotor Imbalance	(0.05)(0) + (0.75)(1.0) + (0.05)(0) + (0.15)(0.5)	0.825
5	Phase Loss	(0.05)(0) + (0.05)(1.0) + (0.85)(0) + (0.05)(0.5)	0.075
6	Shaft Misalignment	(0.05)(0) + (0.65)(1.0) + (0.05)(0) + (0.25)(0.5)	0.775

Step 3: Compute the final priority score Sf

The final score combines the confidence score (80%) with the history score (20%). For this example, the alarm log shows that Bearing Fault has occurred 5 times before on this motor while all other faults have no history. After normalizing H_f (highest count = 1.0, all others = 0):

$$S_f = 0.8 \times C_f + 0.2 \times H_f \dots\dots\dots (II.9)$$

Table III.6: Final Fault Ranking Based on Combined Confidence Score

Rank	Fault Type	C_f	H_f	$S_f = 0.8 \times C_f + 0.2 \times H_f$
1st	Bearing Fault	0.875	1.0	0.900
2nd	Rotor Imbalance	0.825	0.0	0.660
3rd	Shaft Misalignment	0.775	0.0	0.620
4–6th	Remaining faults	0.075	0.0	0.060

III.4.1.2. HMI Interface:

The diagnostic workflow proceeds as follows: when the classifier detects a sustained Critical condition, an alert screen is immediately presented to the operator. This screen tells the technician that a critical state has been identified and gives two options:

- **Ignore:** This will dismiss the alert and allows the system to continue operating.
- **Start Diagnosis:** which commands a controlled motor stop and transitions to the diagnostic suggestion screen.

This two-step ensures that the system is a decision-support tool, not an autonomous controller, and maintains production continuity when the operator considers the alert to be a false positive. The Figures below shows the warning message that will be displayed when a critical condition is detected. The technician is notified of the problem identified and can choose to start the diagnostic process or ignore it.

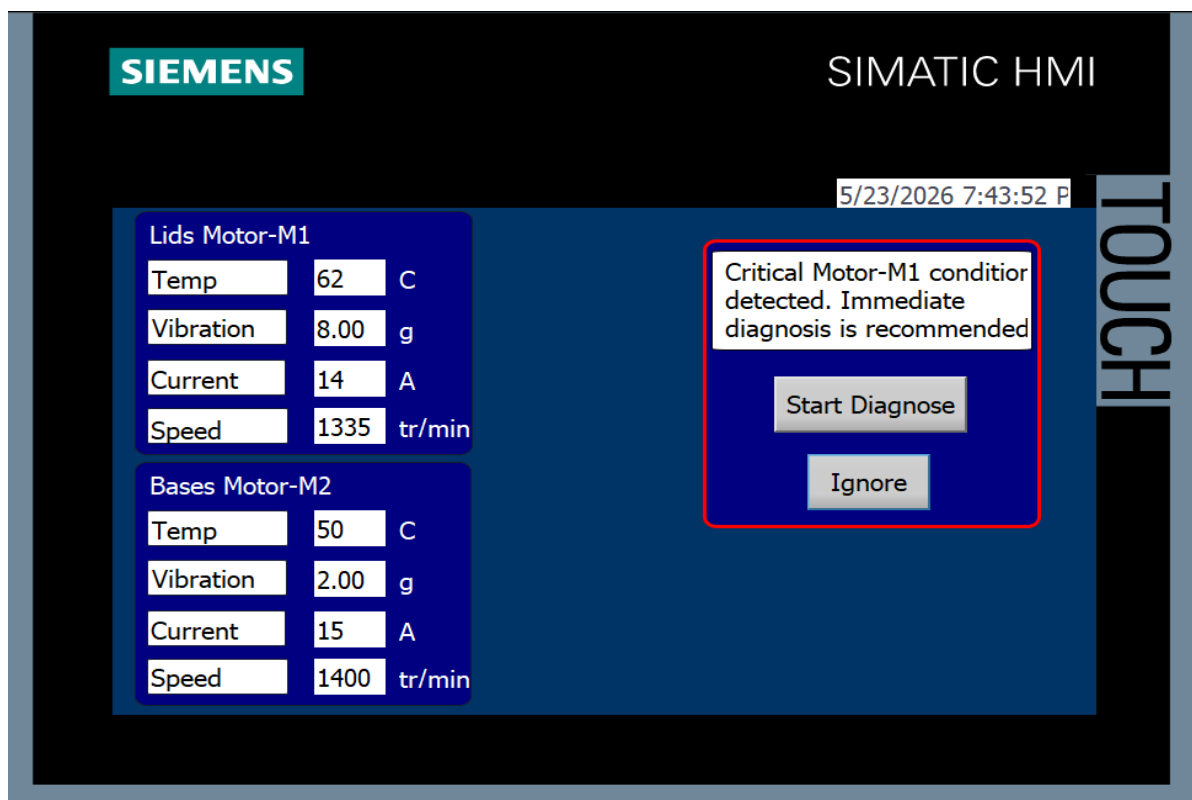


Figure III.10: Virtual production line critical condition alert popup displayed on the HMI

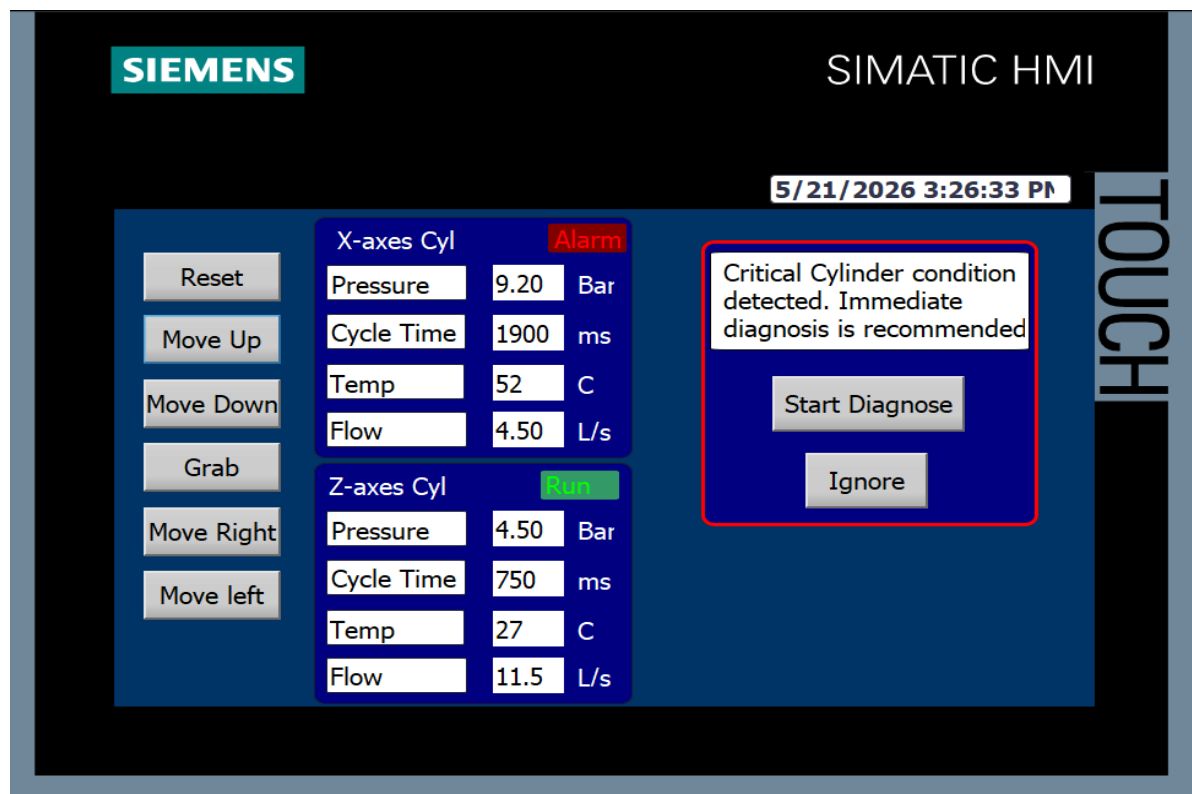


Figure III.11: Real production line critical condition alert popup displayed on the HMI

III.4.1.3. Diagnosis Results:

The results demonstrate that the system could identify the most probable cause of the fault and rank the other possibilities. In the Virtual production line, the air leak fault was identified as the most probable cause.

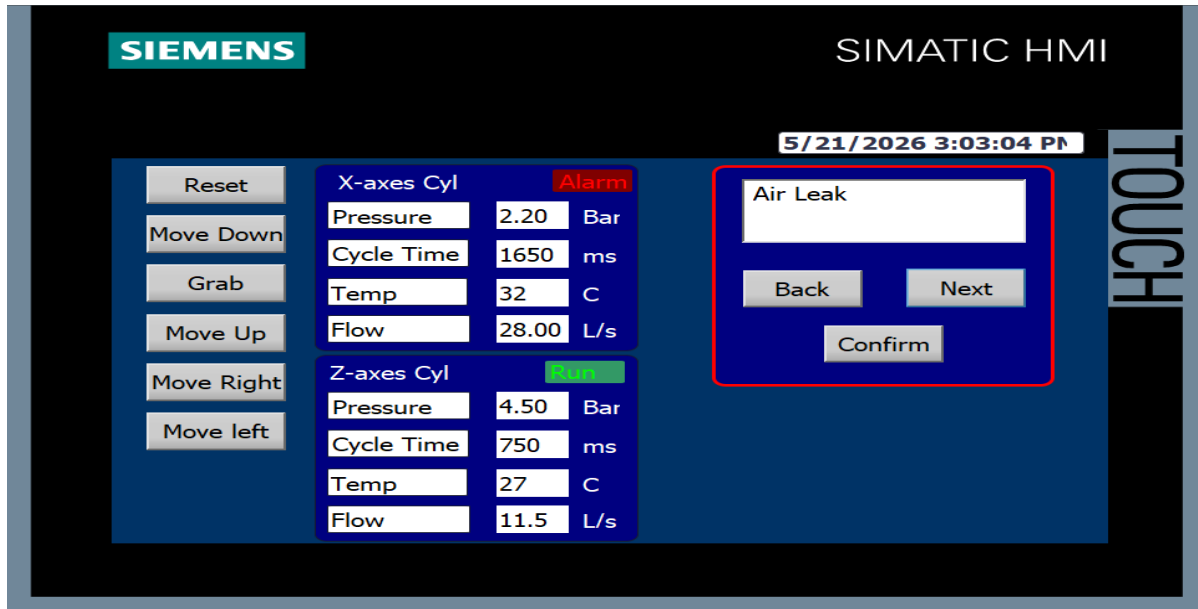


Figure III.12: Virtual production line diagnosis Suggest popup displayed on the HMI

The same behavior can be observed for the real production line. The system was able to parse the available sensor data and produce a ranked list of possible faults. The pop-up on the HMI screen shown in the figure below indicates a bearing fault as the most likely cause.

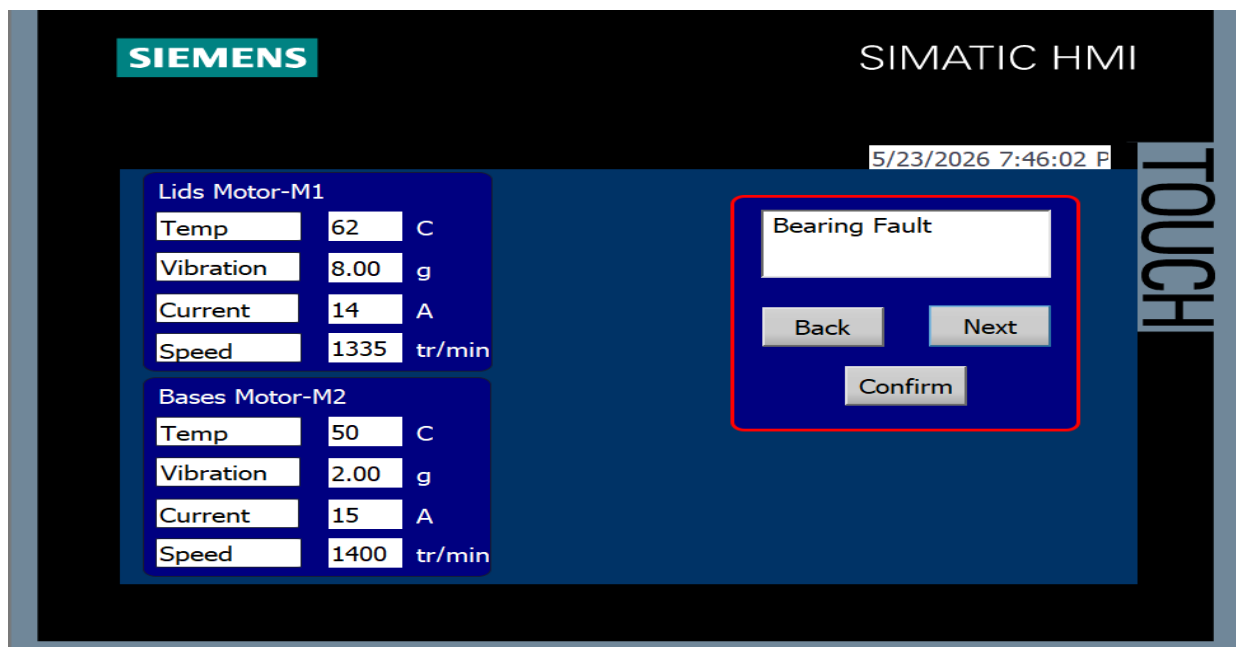


Figure III.13: Real production line diagnosis Suggest popup displayed on the HMI

III.4.2. AI-Based Fault Diagnosis:

Figures presents the diagnostic result generated by the AI assistant for the virtual production line. In addition to identifying the fault, the assistant provides a clear explanation and maintenance recommendations.

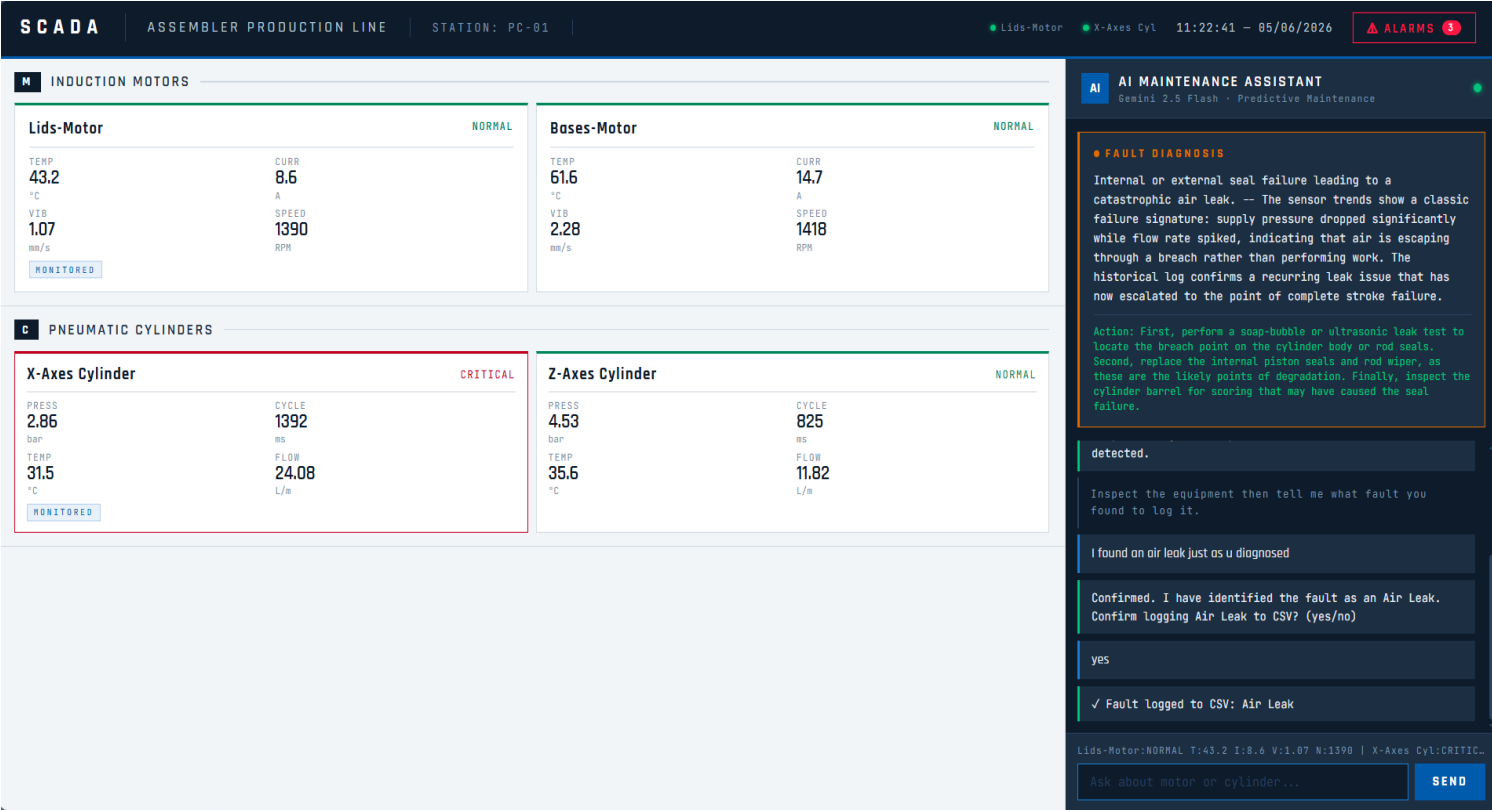


Figure III.14: Ai-Assisted Virtual Production line Web dashboard Scada

The same type of diagnostic output was obtained for the real production line. The AI assistant was able to analyze the fault information and provide useful guidance for maintenance personnel.

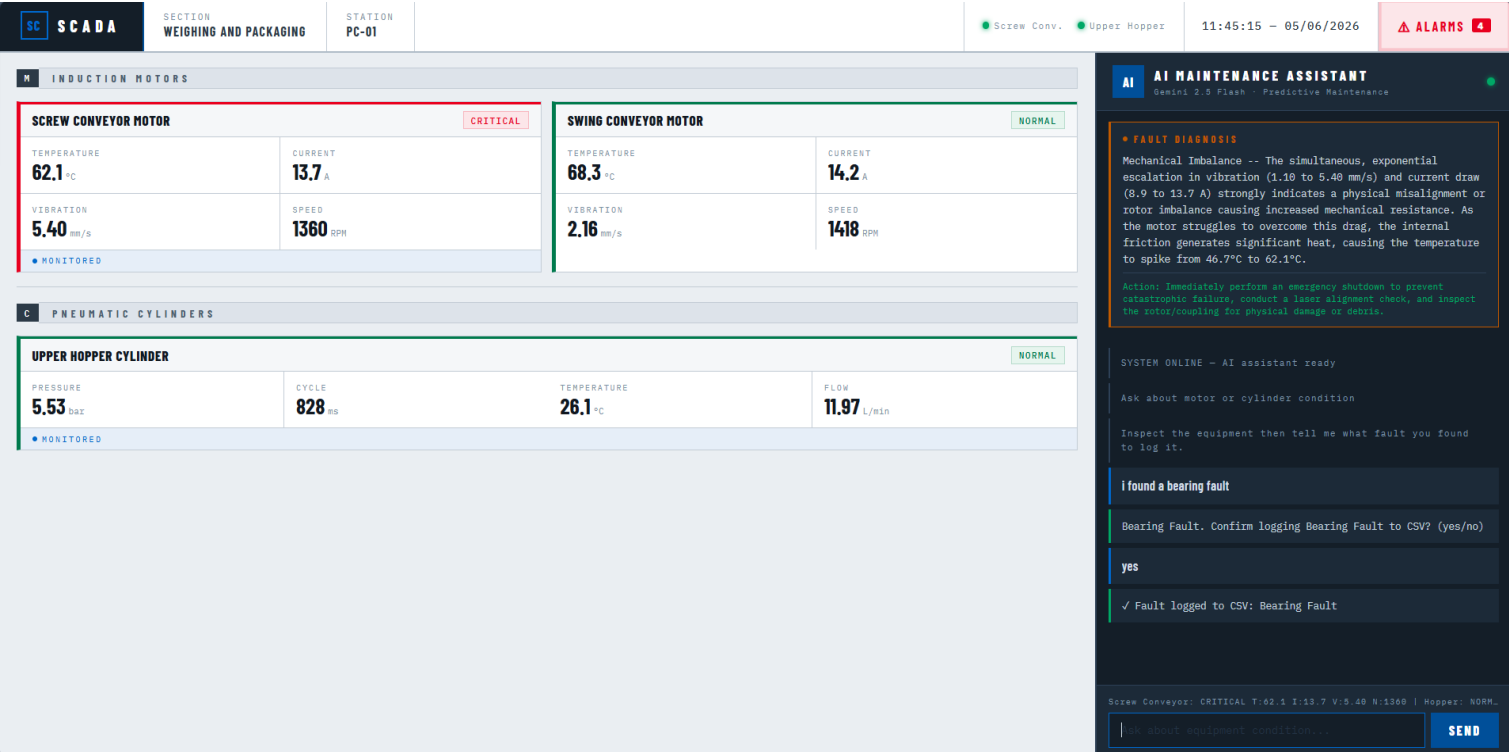


Figure III.15: Ai-Assisted real Production line Web dashboard Scada

III.5. Discussion:

III.5.1 Overall System Performance:

Overall, the results demonstrate that both diagnostic approaches can support fault identification and maintenance decision-making. The weight-based method provides quick fault suggestions, while the AI assistant provides additional explanations and recommendations that help technicians better understand the detected issue. In addition, the AI assistant is not limited to fault diagnosis only. By using the recorded fault history stored in CSV files, it can also provide useful maintenance indicators such as the mean time between failures (MTBF), fault occurrence statistics, downtime analysis, and other performance related information without requiring additional programming or complex calculations from the user.

III.5.2 Synthetic Data Limitations:

The main limitation of this work is that the models were trained using synthetically generated datasets. Although the generated data were designed to resemble realistic operating conditions by introducing overlaps between classes and applying Gaussian distributions, they still cannot fully capture the complexity of real industrial environments. In practice, machine signals are influenced by many factors such as sensor noise, component wear, changing loads, and environmental conditions, which are difficult to reproduce accurately in synthetic data.

This limitation can be observed in the performance of the Warning class, which achieved the lowest F1-score in both case studies (77.4% for the motor and 86.6% for the pneumatic cylinder). This result is expected because the Warning state represents the transition between normal operation and critical conditions, making it more difficult to distinguish from the neighboring classes.

Another limitation is that the fault patterns used to generate the datasets were based on expected theoretical behavior. In real industrial systems, the same fault may produce different signal patterns depending on factors such as machine type, operating conditions, and the surrounding environment. As a result, although the proposed models achieved promising results on the generated datasets, additional training or fine-tuning using real recorded industrial data would likely be required before deployment in a production environment

III.6. Conclusion:

This third and final chapter was dedicated to the presentation, analysis, and comprehensive discussion of the experimental results obtained from implementing our intelligent SCADA system. Through a rigorous evaluation process, we successfully validated the efficiency of the multi-protocol architecture and the developed diagnostic layers across both testbeds: the virtual 3D production line (Factory I/O) and the packaging area replication of the real milling factory (Ouled Mimoun).

The performance of the Feed-Forward Neural Networks (FFNN), trained to classify the operating states of induction motors and pneumatic cylinders, demonstrated an excellent ability to isolate nominal behaviors (Normal) from actual fault situations (Critical). A granular analysis of the confusion matrices and performance metrics highlighted the inherent complexity of the Warning class. The results revealed an F1-score of 77.4% for the motor compared to 86.6% for the cylinder, reflecting the highly dynamic and noisy nature of the electrical and vibrational signals from the motor in contrast to the more linear responses of pneumatic variables (pressure and flow rate).

Beyond mere detection, the two root-cause diagnostic methodologies proved their distinct operational advantages. The local Weight-Based method successfully demonstrated its capability as a robust "Offline" solution, providing instantaneous decision-support directly on the local WinCC HMI screen while fully eliminating internet-related cybersecurity vulnerabilities. Conversely, the AI-Assisted Gemini LLM framework excelled as an "Online" solution, translating complex trend anomalies into natural language diagnostic descriptions and actionable maintenance steps via an interactive web dashboard.

In conclusion, although the reliance on synthetic training data represents a technical limitation for direct industrial deployment, this chapter has demonstrated the viability and high potential of our

architecture. By successfully converging classical automation, data processing through neural networks, and independent online/offline diagnostic strategies, this work establishes solid foundations for the evolution of future supervisory systems toward smarter predictive and prescriptive maintenance

General Conclusion:

This Master's thesis successfully demonstrates the engineering, programming and live implementation of an Intelligent SCADA System for modern production line monitoring, supervision and automated fault diagnosis. This research project combines classic industrial automation practices with modern machine learning techniques and independent diagnostic methodologies to provide a flexible framework for the transformation of raw telemetry data into direct actionable maintenance support decisions.

The project showed that it was architecturally possible to combine incompatible hardware and software ecosystems. Data was routed securely and seamlessly through a well-designed multi-protocol communication network from edge industrial automation software (Siemens TIA Portal V16 and PLCSIM Advanced) to visual environments (Factory I/O), data processing nodes (Node-RED) and numerical algorithmic spaces (MATLAB) using OPC UA, Siemens S7 protocols, WebSockets and encrypted HTTPS APIs.

This architecture was tested in the experimental phase on two independent lines, a virtual 3D assembly and high-density sequential 54-cell storage environment in Factory I/O and a real industrial packaging replication referring to the grain-processing activities of the Ouled Mimoun wheat mill. Real-time sensor streams have been trained successfully for both the platforms using Feed-Forward Neural Networks (FFNNs) for classifying into Normal, Warning and Critical states. The trained neural network classifiers performed well mapping the overlapping variables of induction motors and pneumatic cylinders.

Once a critical operational abnormality was detected, the built supervision layer evaluated two independent, reliable fault localization methods designed to balance deep analytics with field industrial security:

Offline Weight-Based Deviation Method: The method was fully localized at the WinCC HMI edge without an active Internet connection, thus isolating the core industrial network from web-based cyber risks. It combined the statistical deviation of sensors (Z-score) and fault history matrices to produce instant prioritized failure causes for local operators.

The Online AI-Assisted Gemini LLM System: This was the primary connected analytics layer that transformed multi-sensor trend anomalies into rich natural language explanations, identified the precise root causes, and proposed sequential corrective measures via an interactive web dashboard, complemented by Human-in-the-loop maintenance logging.

However, some engineering limitations need to be solved before this system can be directly applied inside a live factory floor despite these promising outcomes. The main limitation of this study is the use of synthetically generated fault datasets. Synthetic structures cannot fully replicate the environmental complexities, load variations, or sensor-degradation realities of actual operating environments. This was reflected in the classification profile of the Warning state that had lower F1-scores across the two case studies (77.4% for the motor and 86.6% for the pneumatic cylinder) due to the very subtle, transitional nature of early-stage fault profiles.

References:

- [1] D. Bailey and E. Wright, *Practical SCADA for Industry*. Oxford, U.K.: Newnes, 2003.
- [2] S. A. Boyer, *SCADA: Supervisory control and data acquisition*, 3rd ed. Research Triangle Park, NC, USA: ISA, 2004.
- [3] K. Stouffer, J. Falco, and K. Kent, “*Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control Systems Security*,” National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST Spec. Publ. 800-82, 2006.
- [4] G. Clarke and D. Reynders, *Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems*. Oxford, U.K.: Newnes, 2004.
- [5] S.-A. Tcaciuc, “Performances Analysis of a SCADA Architecture for Industrial Processes,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 11, pp. 456–460, 2017, doi:10.14569/IJACSA.2017.081155.
- [6] J. Lima, J. F. P. Moreira, and R. M. Sousa, “Remote supervision of production processes in the food industry,” in *Proc. 2015 IEEE Int. Conf. Ind. Eng. Eng. Manag. (IEEM)*, Singapore, 2015, pp. 1123–1127, doi: 10.1109/IEEM.2015.7385823, ISBN 978-1-4673-8066-9.
- [7] A. Enemosah and O. G. Ifeanyi, “SCADA in the era of IoT: Automation, cloud-driven security, and machine learning applications,” *Int. J. Sci. Res. Arch.*, vol. 13, no. 1, pp. 3417–3435, 2024, doi: 10.30574/ijrsra.2024.13.1.1975.
- [8] M. S. S. Al-Dahmahi, N. M. Noor, and M. S. B. Ahmad, “Performance analysis of SCADA system architecture for power distribution automation,” *Am. J. Appl. Sci.*, vol. 11, no. 9, pp. 1418–1425, 2014, doi: 10.3844/ajassp.2014.1418.1425.
- [9] T. Cherifi, “*Développement d’un protocole SCADA temps réel sécurisé à base de l’IEC 60870-5-101*,” Ph.D. dissertation, Dept. Electronics, École Nationale Polytechnique (ENP), Algiers, Algeria, 2019.
- [10] M. Sverko, T. G. Grbac, and M. Mikuc, “SCADA systems with focus on continuous manufacturing and steel industry: A survey on architectures, standards, challenges and Industry 5.0,” *IEEE Access*, vol. 10, pp. 109395–109430, 2022, doi: 10.1109/ACCESS.2022.3211288.

- [11] P. Venkateswaran, “*Leveraging community structure and behavior for smart infrastructure*,” Ph.D. dissertation, University of California, Irvine, CA, USA, 2020.
- [12] F. J. Folgado, D. Calderón, I. González, and A. J. Calderón, “Review of Industry 4.0 from the perspective of automation and supervision systems: Definitions, architectures and recent trends,” *Electronics*, vol. 13, no. 4, Art. no. 782, 2024, doi: 10.3390/electronics13040782.
- [13] D. Neupane, M. R. Bouadjene, R. Dazeley, and S. Aryal, “Data-driven machinery fault detection: A comprehensive review,” *Expert Systems with Applications*, 2024.
- [14] M. Belim, T. Meireles, G. Gonçalves, and R. Pinto, “Forecasting models analysis for predictive maintenance,” *Frontiers in Manufacturing Technology*, 2024.
- [15] O. Merkt, “On the use of predictive models for improving the quality of industrial maintenance: An analytical literature review of maintenance strategies,” in *Proc. 2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Leipzig, Germany, 2019.
- [16] B. Alsulami, K. Kadsah, and E. Alhassan, “The role of predictive maintenance in achieving operational excellence,” *International Journal of Business and Management*, 2026.
- [17] Q. Nie, J. Geng, and C. Liu, “*A review of fault diagnosis methods: From traditional machine learning to large language model fusion paradigm*,” 2026.
- [18] R. N. Kiter, M. M. Ezzaldeen, Y. I. Almashhdany, and F. L. Salim, “Detection and Diagnosis of induction motor faults by intelligent techniques,” *Journal of Engineering*, 2017.
- [19] Eriadom, “*Les Moulins d’Ouled Mimoun: Présentation de la Filiale*,” Les Moulins d’Ouled Mimoun. Accessed: Jun. 02, 2026. Available: <https://eriadom.blogspot.com/2014/11/presentation-de-la-filiale.html>
- [20] I. Achar and M. Habchaoui, “*Modélisation, programmation et simulation d’une chaîne de production*,” Master’s thesis, Dept. Electr. Electron. Eng., University of Tlemcen, Tlemcen, Algeria, 2023.
- [21] [Online]: <https://www.industrie.gov.dz/fr/agrodiv/>
- [22] [Online]: <https://www.molino.com.tr/en-us/corporate/histoire/>

- [23] [Online]: <https://press.siemens.com/global/en/pressrelease/engineering-framework-ready-continuous-software-development>
- [24] [Online]: <https://support.industry.siemens.com/cs/ww/en/view/109771626>
- [25] [Online]: <https://support.industry.siemens.com>
- [26] S. Yu, “Design of a Water Heater Heating System Based on Industrial Control Configuration Software WinCC,” *International Journal of Mechanical and Electrical Engineering*, vol. 4, no. 2, pp. 45-60, Dec. 2024.
- [27] [Online]: https://cache.industry.siemens.com/dl/files/484/109773484/att_1004713/v1/s7-plcsim_advanced_function_manual_en-US_en-US.pdf
- [28] D. Delgado Sobrino, R. Roman, R. Holubek, and K. Velíšek, “Into the early steps of Virtual Commissioning in Tecnomatix Plant Simulation using S7-PLCSIM Advanced and STEP 7 TIA Portal,” *MATEC Web of Conferences*, vol. 299, p. 02005, Jan. 2019.
- [29] [Online]: <https://docs.factoryio.com/#about>
- [30] D. Dsouza, J. Olusola, J. Albrecht, E. Houck, P. Hazaveh, and N. Rawashdeh, “Automating an Industrial Dishwashing System Using Hardware-in-the-Loop PLC Simulation with Factory I/O,” Jul. 2024. doi: 10.18260/1-2--46632.
- [31] [Online]: <https://www.mathworks.com/help/matlab/index.html>
- [32] J. Chukwunweike, C. Eze, I. Abubakar, and A. Adeniran, "Integrating deep learning, MATLAB, and advanced CAD for predictive root cause analysis in PLC systems: A multi-tool approach to enhancing industrial automation and reliability," *World Journal of Advanced Research and Reviews*, vol. 23, no. 2, pp. 2538-2557, Sep. 2024.
- [33] H. S. Jaafar, A. A. Abed, and A.-N. Sharkawy, “Advanced IIoT Integration: Combining OPC UA With a PLC, Node-RED, and KEPServerEX for Secure and Remote Operations,” *SVU-International Journal of Engineering Sciences and Applications*, vol. 7, no. 1, pp. 40–58, Jun. 2026, doi: 10.21608/svusrc.2026.452719.1315.

- [34] H. O. Velesaca, J. A. Holgado-Terriza, and J. M. Gutierrez-Guerrero, "Industrial Process Automation Through Machine Learning and OPC-UA: A Systematic Literature Review," *Electronics*, Sep. 2025.
- [35] W. Alsabbagh and P. Langendoerfer, "*You Are What You Attack: Breaking the Cryptographically Protected S7 Protocol*," ResearchGate, May 2026. Available: <https://www.researchgate.net> (Accessed: May 11, 2026).
- [36] R. Fielding, M. Nottingham, and J. Reschke, "*HTTP Semantics*," IETF, RFC 9110, Jun. 2022. Available: <https://www.rfc-editor.org/rfc/rfc9110.html>
- [37] I. Fette and A. Melnikov, "*The WebSocket Protocol*," IETF, RFC 6455, Dec. 2011. Available: <https://www.rfc-editor.org/rfc/rfc6455.html>
- [38] A. Abdulkareem, T. Anyim, O. Popoola, J. Abubakar, and A. Ayoade, "Prediction of Induction Motor Faults Using Machine Learning," *Heliyon*, 2025.
- [39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. Available: <http://www.deeplearningbook.org>
- [40] S. Nandi, H. A. Toliyat, and X. Li, "Condition Monitoring and Fault Diagnosis of Electrical Motors—A Review," *IEEE Transactions on Energy Conversion*, Dec. 2005.
- [41] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, Jul. 2009.

Appendix:

This appendix presents a brief overview of the softwares used in this project. For each software, screenshots of the main interface with the names of the most important interface elements and a concise description of their functions. This appendix aims to help better understand the software environments used during the design, simulation, control, monitoring, and diagnostic phases of the project.

I. Tia Portal:

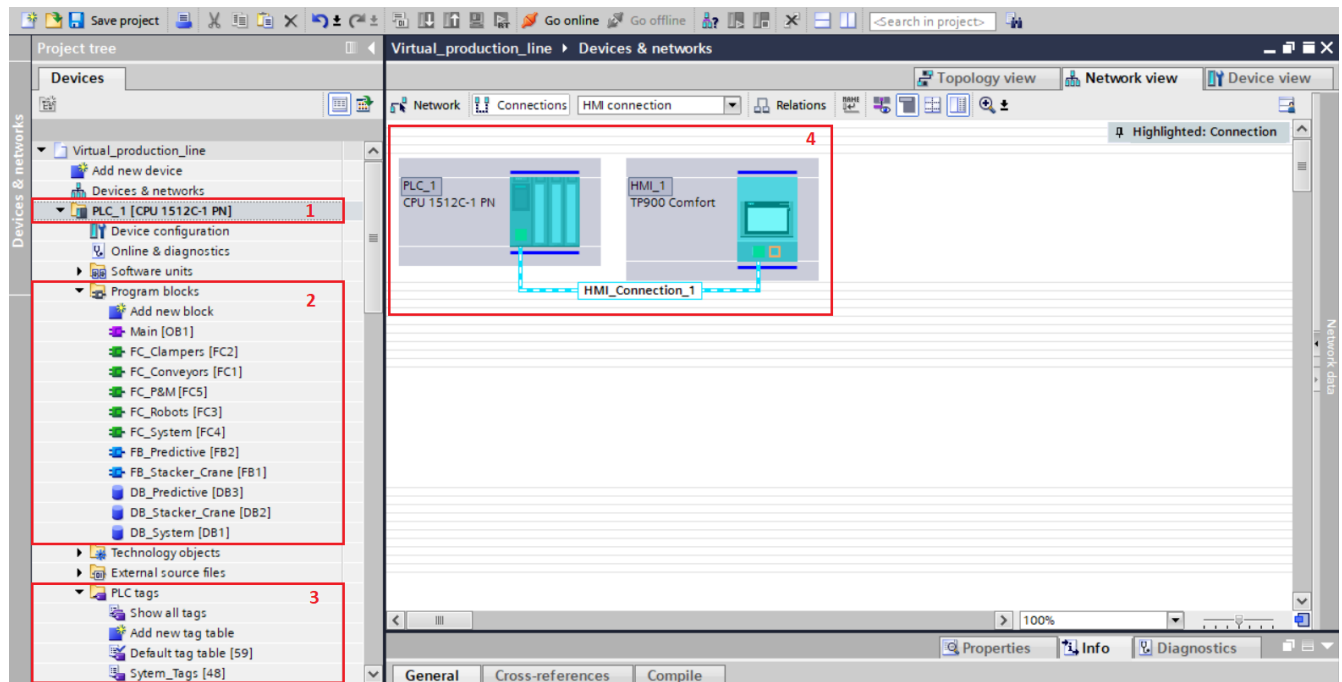


Figure 1: Tia Portal interface 1

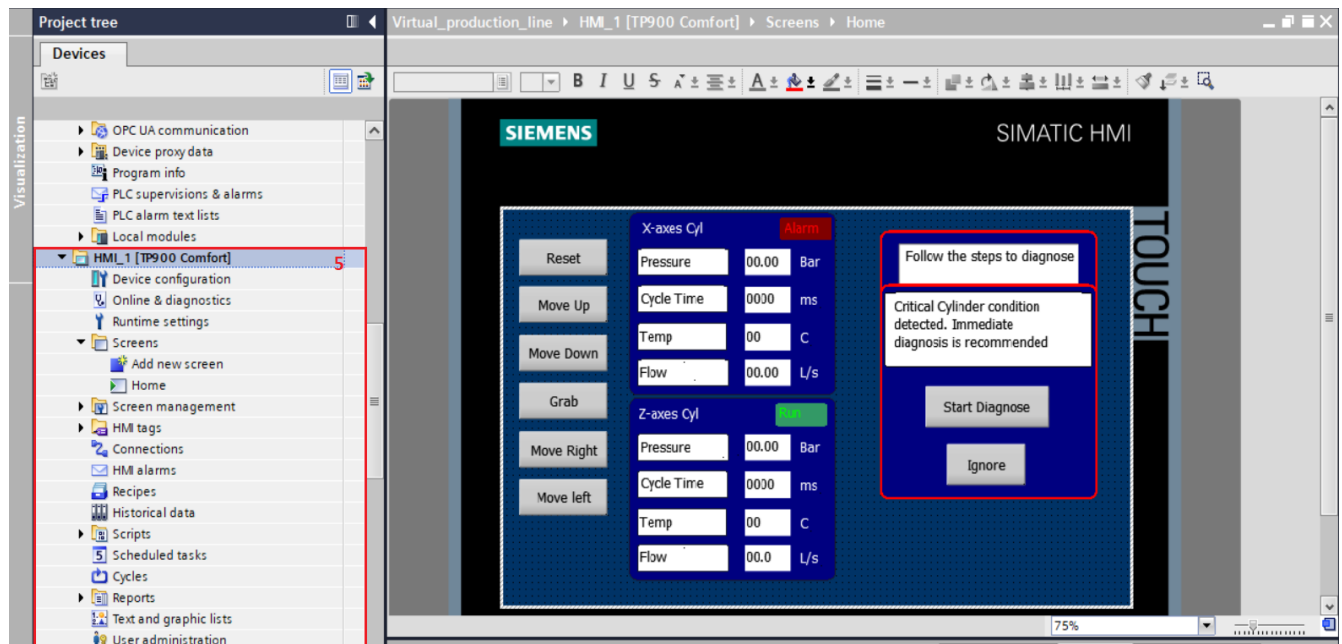


Figure 2: Tia Portal interface 2

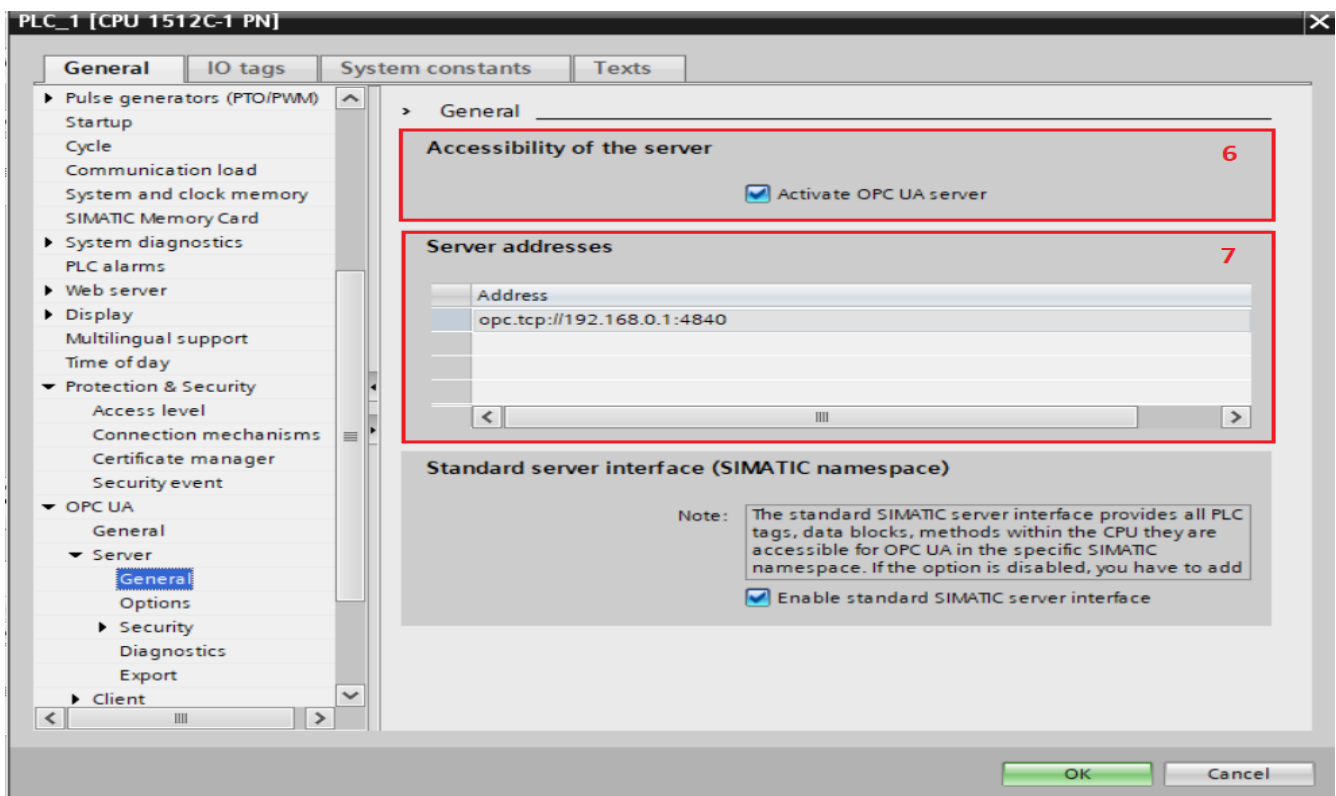


Figure 3: Tia Portal interface 3

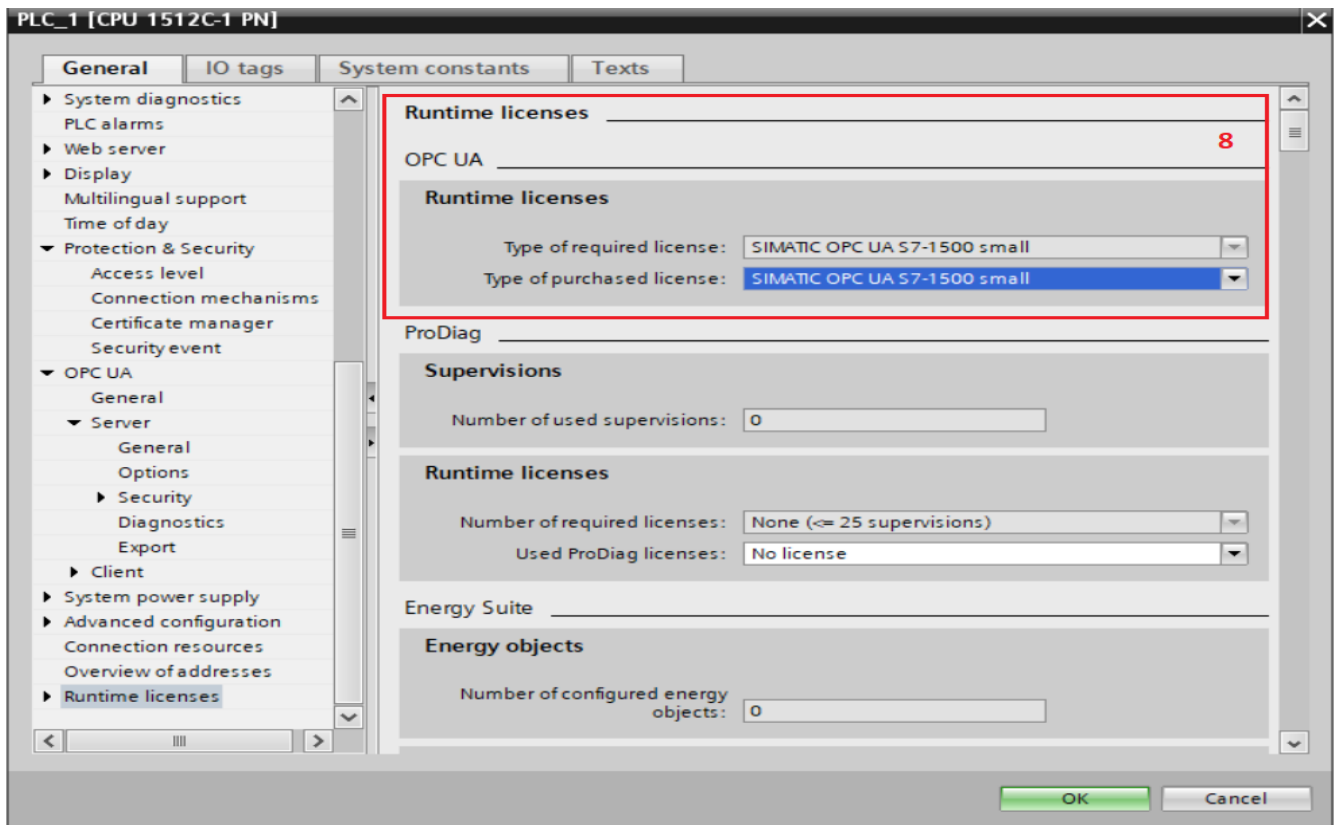


Figure 4: Tia Portal interface 4

Table 1: Description of the Tia portal interface

NO.	Interface element	Description
1	PLC CPU 1512C-PN	CPU 1512C-1 PN was selected instead of the S7-1200 because it supports PLCSIM Advanced and provides native OPC UA communication for integration with external applications.
2	Program Block	Main (OB01): Main program cycle where all functions and function blocks are called. Function Block (FB): Used for operations that require memory retention through an associated data block. Function (FC): Used for simple operations that do not require memory storage. Data Block (DB): Stores data and parameters used by the program and by function blocks.
3	PLC Tags	Defines the variables used to exchange and store data within the PLC program.
4	Network view	Displays the network configuration and communication links between devices, such as the PLC and HMI.
5	HMI TP900 Comfort	TP900 Comfort was selected because it features a touchscreen interface and provides efficient visualization, making it well suited for this project.
6	Accessibility of the server	Activates the OPC UA server for data exchange with external applications
7	Server adress	Displays the address used by clients to connect to the OPC UA server
8	Run time licenses	Specifies the OPC UA license required and used by the PLC (The Small license was selected as it provides sufficient OPC UA capacity for the project's communication requirements.

II. PLCSIM Advanced:

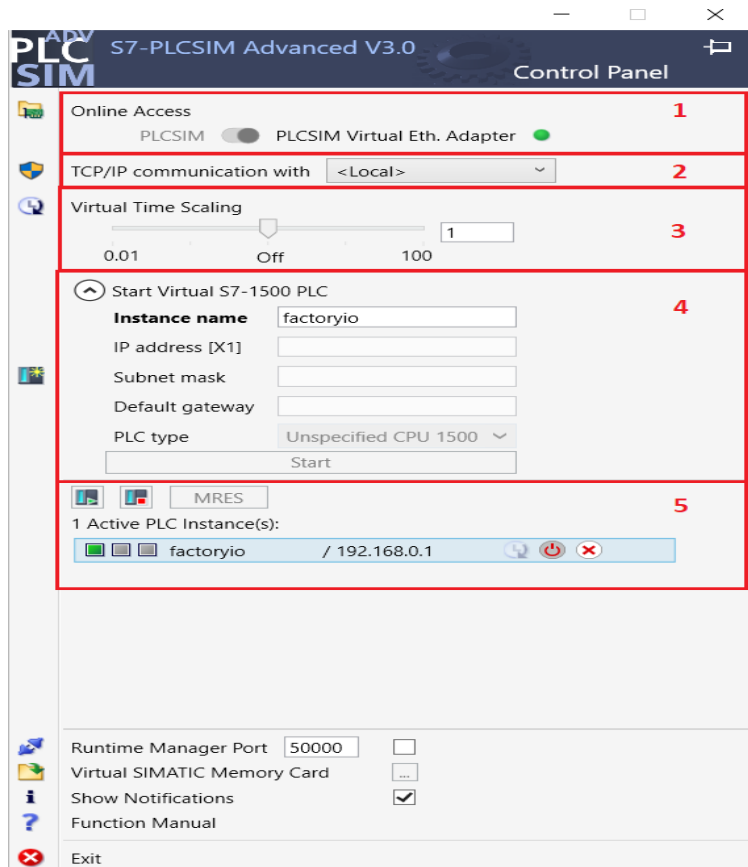


Figure 5: PLCSIM ADV interface

Table 2: Description of the PLCSIM Advanced Interface

NO.	Description
1	Configures online communication with the virtual PLC.
2	Selects the communication network interface. (The instance name must be set to "factoryio" to enable communication with Factory I/O)
3	Controls the simulation speed.
4	Creates and configures a virtual PLC.
5	Manages and monitors virtual PLC instances.

III. Node-RED:

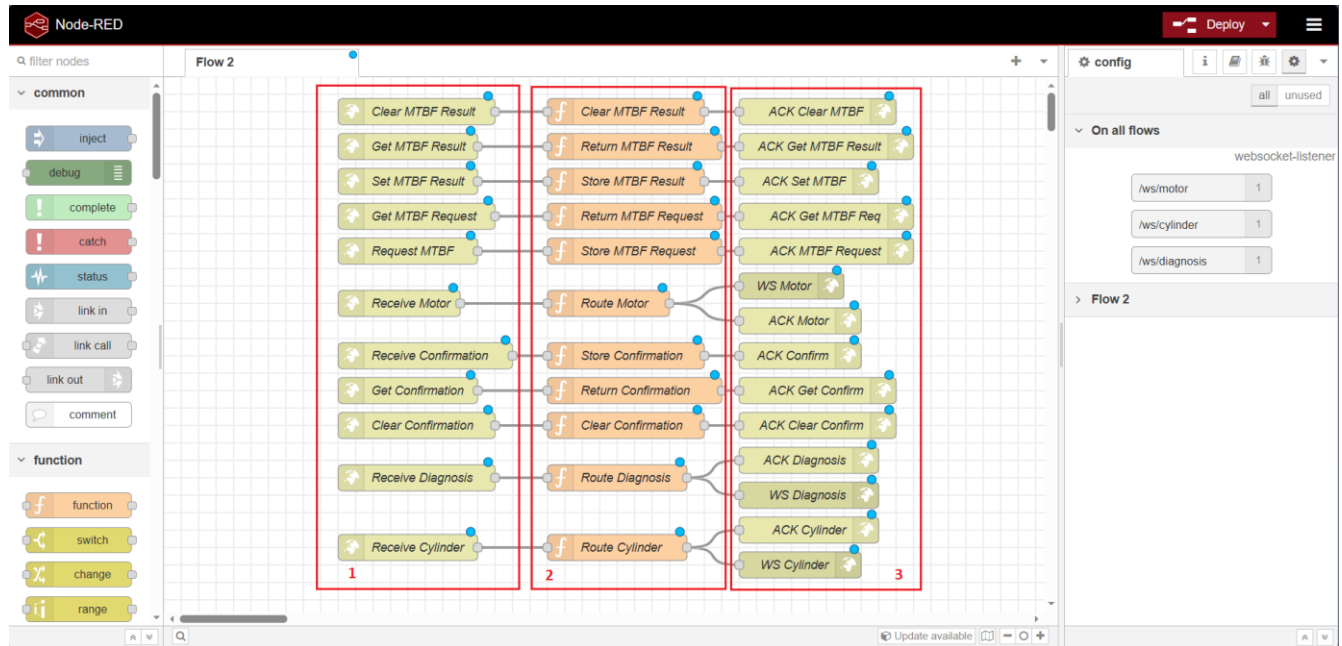


Figure 6: Node-RED interface

Table 3: Description of the Node-RED interface

NO.	Interface element	Description
1	HTTP In nodes	Opens the door and receives the request.
2	Function nodes	Processes the data, stores or reads it.
3	HTTP Response nodes	Replies with "received" and closes the connection.

IV. Factory IO:

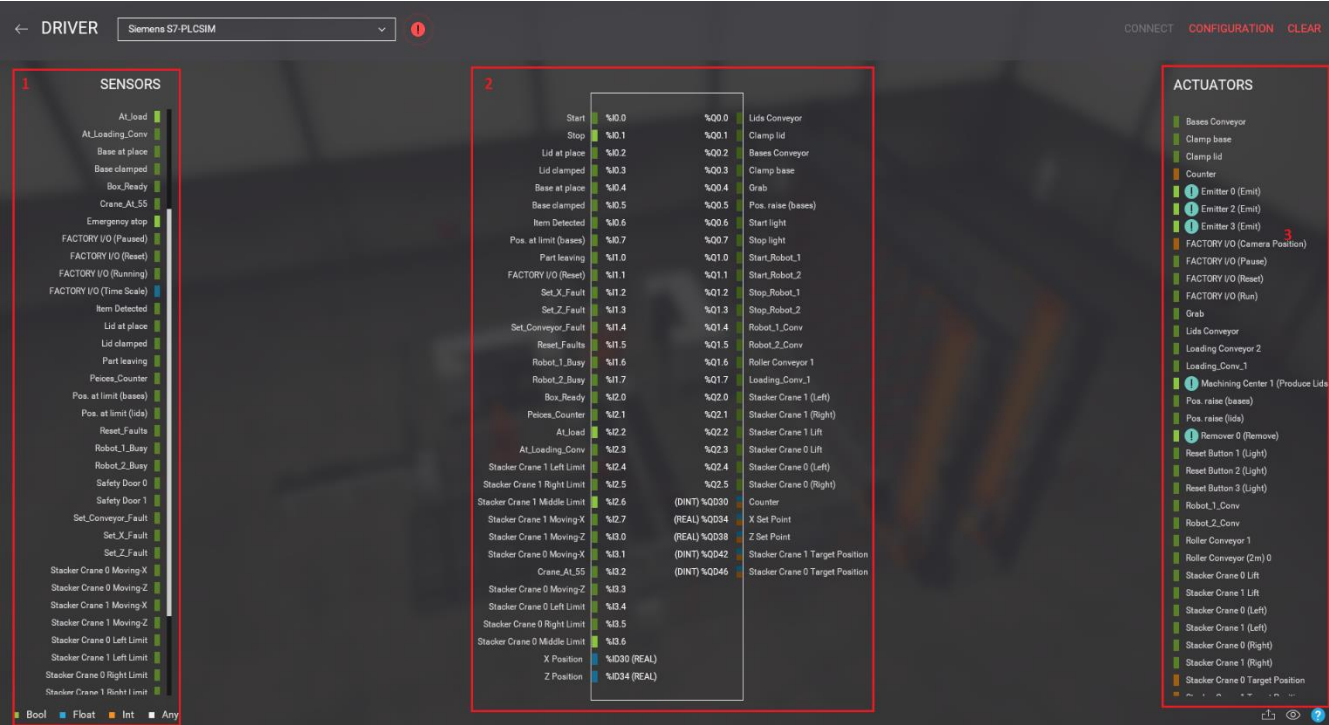


Figure 7: Factory IO driver interface

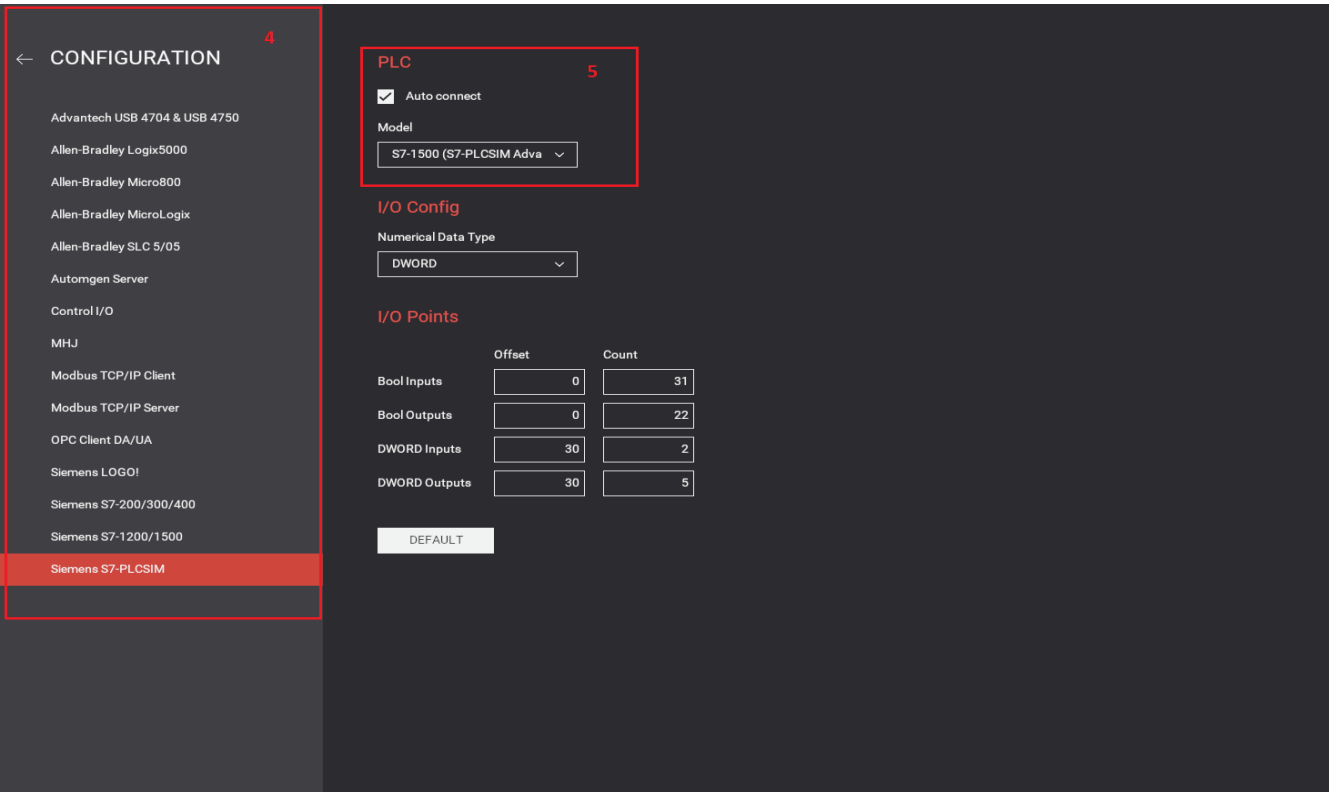


Figure 8: Factory IO Configuration interface

Table 4: Description of the Factory IO driver and the Configuration interface

NO.	Interface element	Description
1	Sensors	Contains the tags associated with the PLC inputs used to receive data from Factory I/O sensors.
2	I/O Mapping Area	Used to link Factory I/O tags with the corresponding PLC addresses.
3	Actuators	Contains the tags associated with the PLC outputs used to control Factory I/O actuators.
4	Driver Selection	Provides various PLC drivers and communication methods. In this project, the Siemens S7-PLCSIM driver was selected.
5	PLC Model	Defines the PLC type used for communication. The S7-1500 (S7-PLCSIM Advanced) model was selected to match the virtual PLC created in PLCSIM Advanced.

V. Matlab:

```
%% ===== OPC UA CONNECTION =====  
fprintf('Connecting to OPC UA...\n');  
uaClient = opcua('opc.tcp://192.168.0.1:4840');  
while ~strcmp(uaClient.Status, 'Connected')  
    try  
        connect(uaClient);  
        fprintf('Connected.\n');  
    catch ME  
        fprintf('Retrying... (%s)\n', ME.message);  
        pause(3);  
    end  
end
```

Figure 9: OPC UA Connection script

```

%% ===== NODE DEFINITIONS =====
nd.start_sys      = opcuanode(3, '"DB_System"."Start"',      uaClient);
nd.reset          = opcuanode(3, '"DB_System"."Reset"',      uaClient);
nd.pause          = opcuanode(3, '"DB_System"."Pause"',      uaClient);

nd.test_im1       = opcuanode(3, '"DB_Predictive"."IM_Test_1"', uaClient);
nd.test_im2       = opcuanode(3, '"DB_Predictive"."IM_Test_2"', uaClient);
nd.im_temp        = opcuanode(3, '"DB_Predictive"."IM_Temp"',  uaClient);
nd.im_curr        = opcuanode(3, '"DB_Predictive"."Current"',  uaClient);
nd.im_vib         = opcuanode(3, '"DB_Predictive"."Vibration"', uaClient);
nd.im_speed       = opcuanode(3, '"DB_Predictive"."Speed"',    uaClient);
nd.im_status      = opcuanode(3, '"DB_Predictive"."IM_Status"', uaClient);
nd.im_sev         = opcuanode(3, '"DB_Predictive"."IM_Severity"', uaClient);
nd.im_fault       = opcuanode(3, '"DB_Predictive"."IM_Motor_Fault"', uaClient);
nd.im_timer       = opcuanode(3, '"DB_Predictive"."IM_Phase_Timer"', uaClient);
nd.im_reset_done  = opcuanode(3, '"DB_Predictive"."IM_Reset_Done"', uaClient);
nd.im_end_scen    = opcuanode(3, '"DB_Predictive"."IM_End_scenario"', uaClient);
nd.im_diag        = opcuanode(3, '"DB_Predictive"."IM_Diag_Sorted_Steps"', uaClient);

nd.test_cyl1      = opcuanode(3, '"DB_Predictive"."Cyl_Test_1"', uaClient);
nd.test_cyl2      = opcuanode(3, '"DB_Predictive"."Cyl_Test_2"', uaClient);
nd.cy_press       = opcuanode(3, '"DB_Predictive"."Pressure"',  uaClient);
nd.cy_cycle       = opcuanode(3, '"DB_Predictive"."CycleTime"',  uaClient);
nd.cy_temp        = opcuanode(3, '"DB_Predictive"."Cyl_Temp"',    uaClient);
nd.cy_flow        = opcuanode(3, '"DB_Predictive"."Flow"',       uaClient);
nd.cy_status      = opcuanode(3, '"DB_Predictive"."Cyl_Status"', uaClient);
nd.cy_sev         = opcuanode(3, '"DB_Predictive"."Cyl_Severity"', uaClient);
nd.cy_fault       = opcuanode(3, '"DB_Predictive"."Cyl_Fault"',  uaClient);
nd.cy_timer       = opcuanode(3, '"DB_Predictive"."Cyl_Phase_Timer"', uaClient);
nd.x_cyl_fault    = opcuanode(3, '"DB_Predictive"."X_Cyl_Fault"',  uaClient);
nd.x_fault_on     = opcuanode(3, '"DB_Predictive"."X_Fault_On"',  uaClient);
nd.cy_end_scen    = opcuanode(3, '"DB_Predictive"."Cyl_End_scenario"', uaClient);
nd.cy_diag        = opcuanode(3, '"DB_Predictive"."Cyl_Diag_Sorted_Steps"', uaClient);

```

Figure 10: Node definition script

```

%% ===== GEMINI CONFIG =====
GEMINI_KEY = [REDACTED];
GEMINI_MODEL = 'gemini-3.1-flash-lite';
GEMINI_URL = sprintf(['https://generativelanguage.googleapis.com/' ...
    'v1beta/models/%s:generateContent?key=%s'], ...
    GEMINI_MODEL, GEMINI_KEY);

```

Figure 11: Gemini config script

Table 5: Description of OPC UA Connection, Node and Gemini Config

Element	Description
OPC UA Connection	Establishes communication with the PLCSIM Advanced through OPC UA.
Node Definition	Define the PLC variables that can be accessed by MATLAB through OPC UA communication.
Gemini Config	Configuration used to access the Gemini AI service.

