PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

MOHAMED BOUDIAF UNIVERSITY - M'SILA

FACULTY OF TECHNOLOGY | **FIELD: SCIENCE and TECHNOLOGY**

DEPARTMENT OF ELECTRONICS | **SECTOR: ELECTRONICS**

N° :…….. | **OPTION: INSTRUMENTATION**

Thesis submitted in partial fulfillment
of the requirements for the Academic Master's Degree
In accordance with Ministerial Decree No. 1275

By:

RAGGAI Abdeloihab

Titled

Integrated learning system for industrial automation and instrumentation Based on PID-Controller Using Tia Portal and Factory I/O

Defended before the examination board composed of:

Name and Surname	Rank/Position	Role	Affiliation
BOUGUERRA Abderrahmen	Professor	Jury President	University of M'sila
GHELLAB Mohamed Zinelaabidine	Senior Lecturer	Supervisor	University of M'sila
KHENNOUF Salah	Senior Lecturer	Examiner	University of M'sila
AIB Abdelghani	Senior Lecturer	Examiner	University of M'sila

ACADEMIC YEAR: 2024–2025

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Dr. Mohamed Zinelaabidine Ghellab, for his invaluable guidance, continuous support, and insightful feedback throughout this project. His expertise and patience were key in overcoming many of the challenges I faced during my research.

My sincere thanks also go to the staff and mentors from the Business Incubator, whose professional advice and encouragement helped enrich the econimical dimension of this work.

I am especially thankful to the university administration of M’sila University,for their availability, collaboration, and for providing an environment conducive to academic achievement.

I also wish to thank all faculty members and technical staff who contributed directly or indirectly to the success of this academic endeavor.

Lastly, I extend my appreciation to all those who shared knowledge, answered questions, or helped me along the way, your contributions are sincerely valued.

Dedication

I wholeheartedly dedicate this work to my family — the foundation of my strength — especially my dear father, my cherished grandmother, and my loving eldest sister.

To all my friends who stood by me, with special gratitude to my friend Muslim and his generous family, for their genuine support and kindness.

To my teachers, who guided me with patience and wisdom throughout my educational journey from childhood until today — your impact on my life is invaluable.

This dedication extends to every person who, whether near or far, knowingly or unknowingly, contributed to the person I am today.

To all who have wished me well, and to those who will in the future — thank you.

And finally, to all the kind-hearted souls we have yet to meet, and those with whom we will continue walking the path of life — this is also for you.

May every kind soul who supported me in any way be rewarded with peace, joy, and success.

الإهداء

أهدي هذا العمل من أعماق قلبي إلى عائلتي، أساس قوتي، وبشكل خاص لوالدي العزيز ، وجدتي الغالية، وأختي الحنونة الكبرى.

و الى جميع الاصدقاء الذين كانوا سندًا لي، مع خالص الامتنان لصديقي مسلم وعائلته، على دعمهم الصادق وطيبتهم.

إلى أساتذتي الذين رافقوني بصبر وحكمة طوال مسيرتي التعليمية منذ الطفولة وحتى اليوم، إن تأثيركم في حياتي لا يُقدر بثمن.

كما أهدي هذا العمل إلى كل من ساهم، من قريب أو بعيد، بوعي أو دون وعي، في تكويني و وصولي إلى ما أنا عليه اليوم

لكل من تمنى لي الخير، والدين سيتمنونه لي لاحقًا، شكرًا لكم.

وأخيرًا، إلى كل الأرواح الطيبة التي لم نلتقِ بها بعد، والى الأرواح التي سنواصل معها مشوار الحياة ،هذا الإهداء لكم أيضًا.

أسأل الله أن يجزي كل نفس طيبة ساندتني بأي شكل من الأشكال، خير الجزاء، سرور ونجاح.

Abstract

The thesis titled **"Integrated Learning System for Industrial Automation and Instrumentation: Based on PID-Controller Using TIA Portal and Factory I/O"** aims to bridge the gap between theoretical knowledge and practical application in the field of industrial automation by designing and implementing a closed-loop control system to regulate the water level in a tank using a PID controller. The TIA Portal platform was used for PLC programming and HMI design, and it was integrated with a 3D simulation using Factory I/O, providing a realistic virtual environment. After analyzing the system's open-loop response and applying the Ziegler–Nichols method to determine tuning parameters, the PID controller was activated and tested under various setpoint changes and disturbances. Results showed that the PI controller provided smoother and more stable responses, making it the most suitable for this application. The project demonstrates the importance of combining theoretical knowledge with hands-on skills using modern industrial tools.

Résumé

Le mémoire intitulé **« Système d'apprentissage intégré pour l'automatisation industrielle et l'instrumentation : Basé sur un régulateur PID utilisant TIA Portal et Factory I/O »** vise à combler le fossé entre la théorie et la pratique dans le domaine de l'automatisation industrielle, à travers la conception et la mise en œuvre d'un système de régulation en boucle fermée pour contrôler le niveau d'eau dans un réservoir à l'aide d'un régulateur PID. La plateforme TIA Portal a été utilisée pour la programmation du PLC et la conception de l'interface HMI, en la connectant à une simulation 3D via Factory I/O, créant ainsi un environnement virtuel réaliste. Après l'analyse de la réponse en boucle ouverte du système et l'application de la méthode de Ziegler–Nichols pour le réglage des paramètres, le régulateur PID a été activé et testé sous différents changements de consigne et perturbations. Les résultats ont montré que le régulateur PI offrait une réponse plus fluide et plus stable, ce qui le rend plus adapté à cette application. Le projet met en évidence l'importance d'intégrer les connaissances théoriques avec des compétences pratiques à l'aide d'outils industriels modernes.

الملخص

المذكرة بالعنوان

Integrated Learning System for Industrial Automation and Instrumentation : Based on PID-Controller Using TIA Portal and Factory I/O

تهدف إلى الربط بين الجانب النظري والتطبيقي في مجال الأتمتة الصناعية من خلال تصميم وتنفيذ نظام تحكم مغلق لتنظيم، مستوى الماء في خزان باستخدام متحكم، PID. تم استخدام منصة TIA Portal لبرمجة PLC وإنشاء واجهة HMI وربط مما أتاح تجربة واقعية افتراضية. بعد تحليل استجابة النظام في الحلقة المفتوحة Factory I/O ذلك بمحاكاة ثلاثية الأبعاد عبر واختبار أدائه تحت ظروف مختلفة من PID لاستخلاص قيم الضبط، تم تفعيل المتحكم Ziegler–Nichols وتطبيق طريقة وفّر استجابة أكثر سلاسة واستقرارًا، مما يجعله الأنسب لهذا التطبيق، PI الانتقال والاضطراب. أظهرت النتائج أن المتحكم .وتبرهن المذكرة على أهمية الدمج بين المعرفة النظرية والمهارات العملية باستخدام أدوات صناعية حديثة.

TABLE OF CONTENTS

List of Figure

Figure III.20: Initial Closed-Loop Response (Setpoint = 200 cm, Valve Opening = 50%) with Synchronized HMI and Factory I/O Views.

Figure III.21: Steady-State Achievement: Tank Level Regulated at 200 cm (Combined HMI and Factory I/O Views).

List of Tables

Introduction

In modern industrial automation, precision, reliability, and safety are crucial requirements in the design and operation of control systems. Whether the objective is to maintain a specific temperature in a furnace, ensure the consistent flow of fluid through a pipeline, or regulate the level of liquid in a storage tank, control systems play a central role in achieving stable and efficient process behavior [1][2].

Among the various strategies developed over the past century, Proportional-Integral-Derivative (PID) control remains the most widely used method for continuous process regulation. It strikes a balance between simplicity and performance, making it the default choice in nearly 95% of all industrial control loops. The PID controller's adaptability to a wide range of processes, combined with its ability to correct both steady-state and transient errors, makes it ideal for systems like the one studied in this project, the regulation of liquid level in a tank [3][4].

This project centers around the implementation of a closed-loop control system for managing the water level within a tank using a PID controller. However, the success of such a control application depends not only on understanding the theoretical aspects of the PID algorithm but also on the ability to deploy it effectively in a programmable logic controller (PLC) environment. For this reason, the Siemens TIA Portal software suite is chosen as the core development platform. It enables integrated PLC programming, HMI design, and PID parameterization within a unified engineering environment.

Furthermore, to test and visualize the performance of the control system without requiring physical equipment, we utilize Factory I/O, a 3D industrial simulation platform. This powerful software allows us to build a digital twin of the real system, including the tank, valves, sensors, and actuators, and connect it in real time to our virtual PLC logic. The result is a realistic, interactive simulation that not only reflects real-world conditions but also enables experimentation, fault injection, and iterative design without any physical risks.

By combining theoretical knowledge (PID control), industrial programming tools (TIA Portal), and advanced simulation environments (Factory I/O), this project provides a comprehensive platform for both learning and prototyping.

The rest of this chapter provides a detailed explanation of the three core components: PID controllers, TIA Portal, and Factory I/O. This foundation is necessary before moving on to the practical implementation steps in the last chapter.

CHAPTER ONE

Control Systems and PID-Controller

I.1 Control Systems

A system is defined as a set of interconnected components organized to achieve a specific function or objective. In engineering, systems may involve mechanical, electrical, thermal, chemical, or hydraulic processes, each characterized by variables that evolve over time in response to internal dynamics or external stimuli.

A control system is a system designed to regulate the behavior of another system (the process or plant) to ensure that specific performance criteria are met. It operates by manipulating input variables to drive the system output toward a desired reference value, or setpoint, despite the presence of disturbances and uncertainties [1][2].

Control systems are fundamental in industrial automation and are used to maintain desired operating conditions for variables such as temperature, pressure, level, flow rate, and motor speed. Depending on whether or not they incorporate feedback, control systems are classified into two main categories:

- **Open-Loop Control Systems:** These systems operate based on predefined inputs without using feedback. They cannot automatically correct deviations or adapt to changes in the environment [3].
- **Closed-Loop (Feedback) Control Systems:** These systems continuously monitor the process output and compare it to the desired setpoint. The resulting error signal is used to adjust the input and minimize the deviation [4].

Feedback control systems are preferred in most industrial applications due to their ability to improve accuracy, reject disturbances, and ensure stable operation over time [5].

Despite their advantages, control systems face several challenges, including:

- Process non-linearities.
- External disturbances.
- Time delays (dead time).
- Measurement noise and sensor inaccuracies.
- Unmodeled dynamics or parameter variations [6][7] .

To address these issues effectively, various control strategies have been developed. Among them, **Proportional–Integral–Derivative (PID)** control has emerged as the most widely used approach in industrial environments due to its conceptual simplicity, ease of implementation, and proven robustness in a wide range of applications [8][9].

In the following sections, we explore the theory, structure, tuning, and practical application of PID controllers in the context of level control in a simulated tank system.

I.2 PID Regulation

Proportional–Integral–Derivative (PID) control is one of the most widely adopted strategies in automatic control systems, particularly in industrial and process automation. It is designed to continuously regulate system outputs by minimizing the difference between a desired setpoint and the actual process value.

A PID controller operates based on three control actions: proportional, integral, and derivative. Each of these components plays a distinct role in correcting system deviations. The proportional part responds to the current error, the integral part considers the accumulation of past errors, and the derivative part anticipates future trends in error [11][12]. Together, they provide a flexible and powerful method of dynamic correction.

The main objective of a PID controller is to achieve fast, stable, and accurate system responses while minimizing overshoot and steady-state error. This makes it suitable for a wide variety of control problems, including those involving temperature, pressure, fluid level, speed, and position.

Due to its conceptual simplicity, ease of implementation, and effectiveness, PID control is considered a fundamental technique and is often used as the basis for more advanced control algorithms. Its application extends across various engineering fields, and it serves as a critical component in both theoretical studies and practical industrial systems [14][15].

In these sections, we explore the theoretical basis of PID control, examine its structure and tuning parameters, and discuss methods of implementation and optimization in practical control systems.

I.3. General Principle of Feedback Control

A feedback (closed-loop) control system continuously measures a process variable (PV) and compares it against a setpoint (SP) . The resulting error:

$$e(t) = SP - PV\ (t) \qquad (1.1)$$

drives the controller to generate an actuation signal **u(t),** which manipulates the plant (tank, pump, valve) to reduce error toward zero [1][2] .

- Basic Closed-Loop Block Diagram:

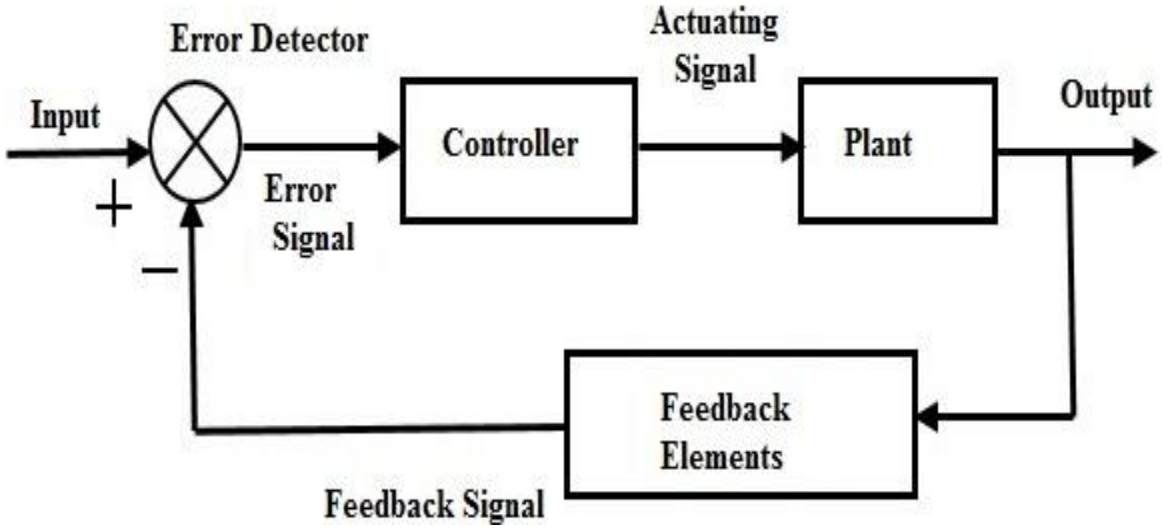

Figure I.1 : Closed-Loop Block Diagram

Input (Reference or Setpoint, SP)

The **input** represents the desired value or target condition that the system must achieve or maintain. It is typically denoted as r(t) or **Setpoint (SP)**. This value is defined externally by the user or an external command signal.

r(t)=desired system output

Error Detector:

The error detector performs a real-time subtraction between the setpoint and the actual output y(t) to generate the error signal e(t):

$$e(t) = y(t) - r(t) \qquad (1.2)$$

This signal quantifies the difference between the desired behavior and the current state of the system [3].

Error Signal:

The error signal is a dynamic function of time that serves as the primary input to the controller. It reflects how far the process variable deviates from the desired reference. The magnitude and sign of e(t) determine how the controller should react [4].

Controller:

The controller is the decision-making unit that processes the error signal and generates an actuating signal.

Actuating Signal:

The actuating signal u(t) is the output of the controller. It is an analog or digital signal that is applied to the plant to alter its behavior and reduce the system error. This signal represents the control effort applied to the system [5] .

Plant (Process)

The plant refers to the physical system being controlled, such as a mechanical, electrical, thermal, or fluid process. It is the object that receives the actuating signal and produces an output in response. The behavior of the plant is typically modeled by a transfer function (1.3):

$$G(s) = \frac{Y(s)}{U(s)} \qquad (1.3)$$

Where:

- **G(s):** System transfer function.
- **U(s):** Laplace transform of the input (control signal).
- **Y(s):** Laplace transform of the output [2][6].

Output (Process Variable, PV) :

The output y(t), also known as the process variable (PV), is the actual value of the variable being regulated by the control system. This value is continuously monitored and compared to the setpoint.

Feedback Elements:

Feedback elements include sensors, transducers, or any devices used to measure the output variable. These components convert physical quantities (e.g., temperature, pressure, position) into electrical signals suitable for processing by the controller. These elements may include:

- Amplification.
- Scaling.
- Filtering [7].

Feedback Signal:

The feedback signal is the measured and possibly conditioned version of the output, which is fed back into the system to be compared with the setpoint. This signal closes the loop and allows the system to adjust its behavior dynamically [8].

A closed-loop system enhances precision, compensates for disturbances, and ensures system stability through continuous self-correction. It is widely used in industrial control systems, robotics, process automation, and many other engineering applications [9][10].

I.4. Performance Metrics for Regulated Systems

Evaluating the performance of a closed-loop control system is essential for ensuring its effectiveness and reliability. The following key metrics are used to assess how well a system responds to changes in input and rejects disturbances :

- **Speed (Rapidity)**

Speed refers to how quickly the system responds to changes in the setpoint or external disturbances. It is primarily measured by:

- **Rise Time (Tr)**:
  The time required for the process variable (PV) to rise from **10% to 90%** of the final steady-state value (setpoint) after a step input. A shorter rise time indicates a faster initial response [3].

- **Settling Time (Ts)**:
  The time it takes for the system output to remain within a specified tolerance band (commonly **±2%**) of the final value without further oscillations. This parameter reflects how long the system takes to stabilize [4].

- **Accuracy (Precision)**

Accuracy reflects the system's ability to reach and maintain the desired output:

- **Steady-State Error (e_{ss})**:

The final error between the setpoint and the actual output after all transients have dissipated:

$$\text{ess} = \lim_{t \to \infty} (SP - PV) \qquad (1.4)$$

Ideally, a well-tuned system aims for $e_{ss} = 0$ [2].

- **Stability**

The maximum peak value the output reaches above the setpoint, expressed as a percentage:

$$\text{Mp} = \frac{max(PV) - SP \times 100\%.}{SP} \qquad (1.5)$$

Large overshoots may lead to instability or undesired system behavior [5].

- **Damping Ratio (ζ)**:

A dimensionless parameter that determines the nature of oscillations.

- $\zeta > 1$: Overdamped (slow, no oscillation).
- $\zeta = 1$: Critically damped (fastest stable response).
- $0 < \zeta < 1$: Underdamped (oscillatory).
- $\zeta = 0$: Undamped (sustained oscillations) [6].

- **Gain Margin and Phase Margin:**

These are frequency-domain criteria that assess robustness to gain or phase variations [7].

- **Gain Margin**: How much gain can increase before the system becomes unstable.
- **Phase Margin**: How much additional phase lag the system can tolerate before instability.

I.5. Objectives of Control

The primary goal of a control system, particularly one utilizing PID regulation is to ensure the desired behavior of the system under various operating conditions. A well-designed PID controller aims to fulfill the following objectives:

- **Setpoint Tracking**

The system must be able to follow changes in the reference input or setpoint accurately and swiftly.

- This involves minimizing:
    - **Rise Time (Tr)**: The time it takes for the output to approach the setpoint.
    - **Settling Time (Ts)**: The time required for the output to remain within a close range of the setpoint.

Efficient tracking is essential in systems where rapid response is critical (e.g., motion control, robotics) [9].

- **Elimination of Steady-State Error**

The controller should drive the system output to match the setpoint exactly after transients die out.

- The goal is to minimize or eliminate the Steady-State Error (e_{ss}) [1]:

$$0 = \lim_{t \to \infty} (SP - PV) \qquad (1.6)$$

- **System Stability**

The system must remain stable under all operating conditions.

- This includes:

    - Avoiding sustained oscillations.
    - Limiting overshoot (M_p) to acceptable levels.
    - Ensuring bounded response to inputs and disturbances [1].

Stability is a fundamental requirement; without it, the system may become unsafe or unusable.

- **Disturbance Rejection**

The system should be capable of resisting and compensating for external disturbances (e.g., noise, load changes).

- A robust controller minimizes the effect of:

    - Input disturbances
    - Sensor noise
    - Model uncertainties [10].

The effectiveness of disturbance rejection is a hallmark of closed-loop control systems and a major advantage of PID regulation.

So, a high-performance PID controller should:

- React quickly to desired changes,
- Settle accurately at the target value,
- Remain stable under all conditions,
- And handle unexpected changes in the environment gracefully.

I.6. Classical PID Algorithm

The PID control law is:

$$u(t) = K_p\, e(t) \;+\; K_i \int_0^t e(\tau)\, d\tau \;+\; K_d \frac{d\, e(t)}{dt} \qquad (1.7)$$

where:

$K_p\, e(t)$ Proportional (P) term, provides immediate correction.

$K_i \int_0^t e(\tau)\, d\tau$ Integral (I) term, removes offset by accumulating past errors.

$K_d \frac{d\, e(t)}{dt}$ Derivative (D) term, predicts error trends, enhancing damping [2][11].

I.7. PID Tuning Parameters

PID tuning involves adjusting three key parameters that directly influence the performance and stability of the control system.

- **Kp** (Proportional gain): increases loop responsiveness but can amplify overshoot.
- **Ki** (Integral gain): eliminates steady-state error but may induce oscillations if too large.
- **Kd** (Derivative gain): adds predictive damping, reducing overshoot but sensitive to measurement noise.

Balancing these three gains is central to controller design [12].

I.8. Control Modes

Control modes are fundamental components in the design of automatic control systems. Each mode offers specific characteristics that affect system stability, accuracy, and responsiveness.

- P-only : fast response; nonzero offset.
- PI : zero offset; moderate overshoot.
- PD : damping; cannot correct offset.
- PID : combines benefits; requires careful tuning .

Understanding the strengths and limitations of each control mode allows for appropriate selection based on system requirements. Effective implementation depends on proper tuning and application context.

I.9. PID Tuning Methods

In order to ensure optimal performance of a PID controller, its parameters proportional gain (Kp), integral time (Ti), and derivative time (Td), must be appropriately tuned. Several widely accepted tuning methods exist, ranging from empirical formulas to trial-and-error approaches. The most prominent techniques are described below.

I.9.1. Ziegler–Nichols Ultimate Gain Method

The Ziegler–Nichols tuning method, introduced in 1942, is one of the most popular empirical procedures for tuning PID controllers. The method is based on measuring the system's response to increasing gain until it reaches the point of marginal stability [3][4].

The steps are as follows:

- Disable the integral and derivative terms in the controller.
- Gradually increase the proportional gain (Kp) until the output begins to oscillate with constant amplitude—this value is called the Ultimate Gain (Ku).
- Record the period of oscillation (Pu), known as the Ultimate Period.
- Apply the following formulas based on the desired controller type:

Controller Type	Kp	Ti	Td
P	0.50 × Ku	–	–

PI	0.45 × Ku	Pu / 1.2	–
PID	0.60 × Ku	Pu / 2	Pu / 8

Table I.1: Ziegler–Nichols Ultimate Gain Tuning Rules [5] .

This method is straightforward but assumes that the system can safely oscillate, which may not always be feasible in sensitive or critical applications [6].

II.9.2. Reaction Curve (Step Response) Method

This approach, also known as the Process Reaction Curve (PRC) method, is based on analyzing the open-loop response of the system to a step input. The system's response is approximated by a first-order plus dead-time model, characterized by [7][8] :

- L: Dead time or delay
- T: Time constant (time to reach ~63.2% of the final value)

Using these parameters, the PID gains can be computed as:

Controller Type	Kp	Ti	Td
P	T / L	–	–
PI	0.9 × T / L	3.3 × L	–
PID	1.2 × T / L	2 × L	0.5 × L

Table I.2: PID Tuning Parameters Based on Open-Loop Step Response [22][9].

This method is particularly useful when online tuning is not possible, as it relies on data from open-loop system testing .

I.9.3 Manual Trial-and-Error Tuning

The manual tuning method involves iteratively adjusting the PID parameters while observing the system response. Though less systematic, it remains practical and effective when used by experienced operators [10]:

- Start with small values for Kp, Ti, and Td.
- Gradually increase Kp until a steady and quick response is achieved without overshooting excessively.
- Adjust Ti to eliminate steady-state error. Introduce Td to reduce overshoot and enhance stability.

This method relies on performance metrics such as:

- tr (rise time)
- Mp (maximum overshoot)
- ts (settling time)

While time-consuming, manual tuning offers flexibility in real-time systems and can fine-tune a controller that has been roughly initialized using another method [11].

Conclusion

In this chapter, we explored the foundational principles of control systems, focusing on their structure, classification, and performance criteria. Special emphasis was placed on PID regulation, the most widely adopted control strategy in industrial automation. We discussed the individual roles of the proportional, integral, and derivative components and their combined effect on system stability and accuracy.

Additionally, the chapter introduced various tuning techniques such as Ziegler–Nichols and the Reaction Curve method. that help optimize PID parameters for real-world applications. These theoretical concepts set the groundwork for the practical implementation and simulation of control systems, which will be addressed in the following chapters using tools like TIA Portal and Factory I/O.

CHAPTER TWO

The TIA Portal and Factory I/O Overview

II.1. Overview of TIA Portal

II.1.1. Introduction to TIA Portal

Totally Integrated Automation (TIA) Portal is Siemens' flagship engineering framework designed to integrate and unify automation engineering tasks. It provides a single interface for configuring, programming, testing, and diagnosing devices such as programmable logic controllers (PLCs), Human-Machine Interfaces (HMIs), distributed I/O systems, and drives. TIA Portal is widely used in industrial automation due to its flexibility, modularity, and advanced features that streamline the automation lifecycle.

Its primary goal is to improve the efficiency of engineering workflows through a consistent database, shared libraries, and a uniform graphical environment. It allows engineers to reduce engineering time, minimize configuration errors, and enhance productivity across various stages of project development.

II.1.2 Key Components of TIA Portal

TIA Portal is composed of several key modules and components that support different aspects of industrial automation [14][15]:

- **STEP 7 (Basic and Professional):** Used for PLC programming using languages such as LAD (Ladder), FBD (Function Block Diagram), STL (Statement List), and SCL (Structured Control Language).
- **WinCC (Basic, Comfort, Advanced, and Professional):** Provides tools for HMI design and visualization, including the creation of operator panels, alarms, trends, and input/output fields.
- **Startdrive:** Facilitates the configuration and commissioning of Siemens SINAMICS drives.
- **Safety Advanced:** Supports safety programming for systems that require compliance with safety standards (e.g., SIL, PL) [16].

II.1.3 Supported Devices

TIA Portal supports a wide range of Siemens automation hardware, including [14][17]:

- **S7-1200 and S7-1500 PLCs:** Designed for small to complex control tasks.
- **ET 200 Distributed I/O:** Modular I/O systems that can be configured within the portal.
- **TP and KP HMI Panels:** Touch and key-operated panels for real-time operator interface.
- **SINAMICS Drives:** Integrated via the Startdrive module.

II.1.4 Engineering Workflow in TIA Portal

TIA Portal facilitates a structured engineering process typically consisting of the following steps [14][15]:

1. **Project Creation:** Engineers start by creating a new project in which hardware components, network settings, and programs are defined.
2. **Hardware Configuration:** Devices such as PLCs, HMIs, and I/O modules are added from the hardware catalog and interconnected.
3. **Tag Table Definition:** Global and local variables (tags) are defined for signal acquisition, processing, and interfacing.
4. **Programming:** Logical operations are written using graphical or textual languages supported by IEC 61131-3.
5. **Simulation and Testing:** Built-in tools like PLCSIM and HMI simulators allow testing of logic and user interfaces before physical deployment.
6. **Commissioning:** The project is downloaded to physical devices via Ethernet or USB and monitored in real-time.

II.1.5 Benefits of TIA Portal

The platform provides several benefits for industrial engineers [14][17]:

- **Unified Platform:** All automation tasks are handled in one environment, reducing the need for multiple tools.
- **Data Consistency:** Shared tag databases ensure synchronized variables across PLC and HMI components.
- **Reusability:** Libraries and reusable blocks promote efficient and modular engineering.
- **Diagnostic Tools:** Built-in monitoring and error-handling features help troubleshoot and optimize the system.
- **Simulation Capabilities:** TIA Portal includes PLCSIM for virtual PLC testing, allowing engineers to test logic without connecting to real hardware.

II.1.6 TIA Portal Programming Languages

TIA Portal supports four standard IEC 61131-3 programming languages [15][16]:

- **Ladder Diagram (LAD):** Graphical and intuitive, ideal for simple logic and widely used by electricians.
- **Function Block Diagram (FBD):** Best for process control and analog signal processing.
- **Structured Control Language (SCL):** High-level language suitable for complex algorithms.
- **Statement List (STL):** Low-level, assembly-like language offering detailed control.
- The **Instruction List** is no longer supported in modern platforms. The **Sequential Function Chart** is implemented via the **S7 Graph** environment.

Each language can be selected based on the application requirement and engineer preference. Mixed-language programming is also supported, allowing different parts of the project to be written in the most suitable language.

II.1.7 Functional Programming Structures

Among the notable blocks used in automation projects, TIA Portal provides built-in technology objects and custom blocks that support modular, reusable, and specialized logic development. Commonly used blocks include [14][18]:

- **Organization Blocks (OB):** These are the entry points of the program. OB1 runs cyclically and acts as the main loop. OB30 (Cyclic Interrupt) is used for time-sensitive tasks.
- **Function Blocks (FB):** These hold logic that can be executed multiple times with different instance data. Ideal for encapsulating control algorithms.
- **Functions (FC):** Contain reusable code without memory; used when no internal state storage is required.
- **Data Blocks (DB):** Used to store and manage data related to FBs or global variables.

Of particular importance is the **PID_Compact** block:

- **PID_Compact Block:** This is a built-in function block provided by Siemens for implementing closed-loop control. It includes parameters for tuning (Kp, Ki, Kd), setpoint management, output limitation, and operating modes (manual, automatic, etc.). It can be added from the Technology Objects section and configured through an intuitive wizard, supporting process types such as temperature, pressure, level, and speed control.

The use of such structured blocks ensures clearer organization, scalability, and maintainability of complex control systems.

II.1.8 Integration with External Tools

TIA Portal also integrates with tools such as [14][17]:

- **Factory I/O:** A 3D process simulation software that can communicate with TIA Portal for real-time simulation.
- **SIMATIC Energy Suite:** Enables energy management within the automation project.
- **Version Control Systems (e.g., Git):** Through TIA Portal Openness API, engineers can automate tasks and integrate with external software.

II.2. Factory I/O Overview

Factory I/O (by Real Games) is a PC-based 3D simulation environment designed to emulate industrial automation processes with real-time I/O connectivity to physical or virtual controllers. It enables creation of rich, interactive "digital twin" scenes without the need for CAD expertise [20][21].

II.2.1. Core Features

- Drag-and-Drop Scene Building

A comprehensive library of preconfigured components—conveyors, tanks, sensors, valves, robots, and more—can be placed into the workspace and wired together visually.

Users adjust physical parameters (flow coefficients, tank dimensions, sensor calibration ranges) via property panels [20].

- Real-Time I/O Protocol Support

PROFINET, EtherNet/IP, Modbus TCP, OPC UA interfaces allow seamless connectivity with Siemens, Rockwell, and other PLC platforms.

Virtual Ethernet adapters emulate industrial network behavior, enabling students to practice IP addressing, subnetting, and network diagnostics [20][24].

- Interactive Control & Visualization

Live status indicators display sensor readings, actuator states, and internal variables.

Animated 3D models show fluid levels rising/falling, conveyor movement, and valve positions, reinforcing the link between code and process [21].

- Fault-Injection & Training Modes

Built-in "Fault" controls let instructors introduce simulated failures—sensor drift, blocked pipes, motor stalls—to test troubleshooting and recovery logic under safe conditions.

Step-By-Step mode pauses simulation after each code scan to examine I/O changes and program flow [20][25].

- Scenario & Recipe Management

Sequence Editor scripts step changes (e.g., setpoint jumps, inflow pulses) and timed events for automated testing of control strategies.

- Educational Licensing & Resources

Academic packages provide multi-seat licenses, access to full scene libraries, and tutorial walkthroughs covering beginners' exercises through advanced process-control scenarios [20][21][25].

II.2.2. User Interface

- Scene Workspace:

The central 3D viewport where components are assembled; right-click menus and drag-handles simplify alignment and wiring [20].

- I/O Mapping Panel:

A tabular interface listing each virtual I/O port alongside its mapped PLC tag or network address (e.g., AI0 → IW0, DO1 → Q0.1) [20][24].

- Toolbar & Controls:

Play/Pause/Stop buttons control simulation execution.

Fault Buttons emulate component failures.

Parameter Sliders adjust process variables in real time [21].

II.2.3. Integration with Control Logic

- Tag Synchronization:

Upon connection, Factory I/O automatically discovers controllers on the network and synchronizes I/O port names with the PLC's tag list, minimizing manual mapping errors [20][24].

- Bi-Directional Data Flow:

Sensor outputs in Factory I/O update PLC inputs each scan cycle; PLC output changes immediately affect 3D animations (e.g., pump start/stop, valve open/close) [20][24].

II.2.4. Educational Benefits

- Risk-Free Experimentation:

Students explore fault scenarios without damaging real hardware or interrupting live production.

- Rapid Iteration:

Code adjustments in the PLC can be tested instantaneously in the virtual plant, shortening the learning cycle.

- Visual Feedback:

Intuitive 3D animations reinforce abstract control concepts, aiding comprehension of PID tuning effects, system dynamics, and failure modes.

Conclusion

The TIA Portal and Factory I/O platforms constitute a powerful and complementary environment for automation engineering. TIA Portal offers a fully integrated suite for programming, simulation, and diagnostics of PLCs and HMI systems, while Factory I/O provides a dynamic and realistic 3D simulation space to visualize industrial processes in real time.

One of the most significant strengths of combining these two tools lies in their seamless communication capabilities via standard protocols such as PLCSIM and PROFINET. This integration allows for the development, testing, and fine-tuning of control strategies—including advanced PID regulation—within a safe, virtual environment before real-world implementation.

In the context of my upcoming project, this synergy is particularly valuable. It enables iterative design, step-by-step validation, and controlled experimentation of complex process control scenarios, all without the risks associated with physical hardware. Thus, the combined use of TIA Portal and Factory I/O will serve as a foundational stage in designing and validating the control system.

CHAPTER THREE

Implementation and Validation of the PID-Controller for Tank System

Introduction

This chapter presents the design, implementation, and validation of our experimental work. A PID-controlled tank process fully simulated in Factory I/O and driven by a Siemens PLC programmed in TIA Portal . Its primary purpose is to show how:

all converge to create a realistic, hands-on training module. Through detailed descriptions of system specifications, control-architecture diagrams, step-by-step implementation in both TIA Portal and Factory I/O, and comprehensive performance analysis.

Note: the entire process including level measurement, pump actuation, and valve operation is executed inside Factory I/O .

This final chapter marks the transition from theoretical understanding to practical application. In the previous chapter, we introduced and explored the essential technologies and control strategies involved in our project — namely PID control theory, Siemens TIA Portal, and the Factory I/O simulation platform.

In Chapter 1, we detailed the fundamentals of PID control, including its feedback structure, tuning methods such as Ziegler-Nichols, and the purpose of each parameter (Kp, Ki, Kd). These concepts are directly applied in our implementation of the PID_Compact block within the TIA Portal environment, highlighting the controller's role in regulating tank level in real-time.

Additionally, the introduction to TIA Portal in the second chapter equipped us with an understanding of how control logic is structured, programmed, and simulated using Siemens hardware and software. The modular programming techniques, use of OB1 and OB30, tag tables, and hardware configuration directly reflect what was covered in the theoretical overview.

Similarly, the section on Factory I/O helped establish its importance as a virtual tool. We discussed its built-in sensors, actuators, and compatibility with PLC simulation software. The knowledge of its structure, object libraries, and communication settings was necessary to correctly integrate it with PLCSIM via the S7-1500 driver.

This chapter now demonstrates how these technologies come together to form a complete control loop, from input via the HMI to processing inside the PLC, output to actuators, and feedback from sensors replicating an industrial tank-level control process

The following sections will therefore present, in a step-by-step manner, how this theoretical foundation has been applied in the configuration, programming, simulation, and validation of our closed-loop level control system.

III.1. System Architecture and Components

III.1.1. Components in TIA Portal:

- CPU: Siemens S7-1500 series, model CPU 1513-1 PN**,** used as the main programmable logic controller (PLC) [14].
- HMI Panel: TP900 Comfort Panel**,** used to monitor and interact with the process through graphical interface elements such as buttons, level indicators, and alarms [15].

III.1.2. Components in Factory I/O:

- Tank: Represents the main water storage unit.
- Filling Valve**:** Controls the inflow of water into the tank.
- Discharge Valve: Controls the outflow of water from the tank.
- Level Sensor (Level Meter): Measures the current water level inside the tank and provides continuous feedback to the PLC.
- Flow Meter**:** Monitors the flow rate of the liquid being discharged from the tank using the actuator [20].

III.2. Simulation Environment Setup

To begin the simulation process in Factory I/O, the following steps were performed:

First, upon launching the software, the user interface opens by default to the "Documentation and Tutorials" tab, as shown in Figure 2. From the same menu, we click on "New" to start a new project [20].

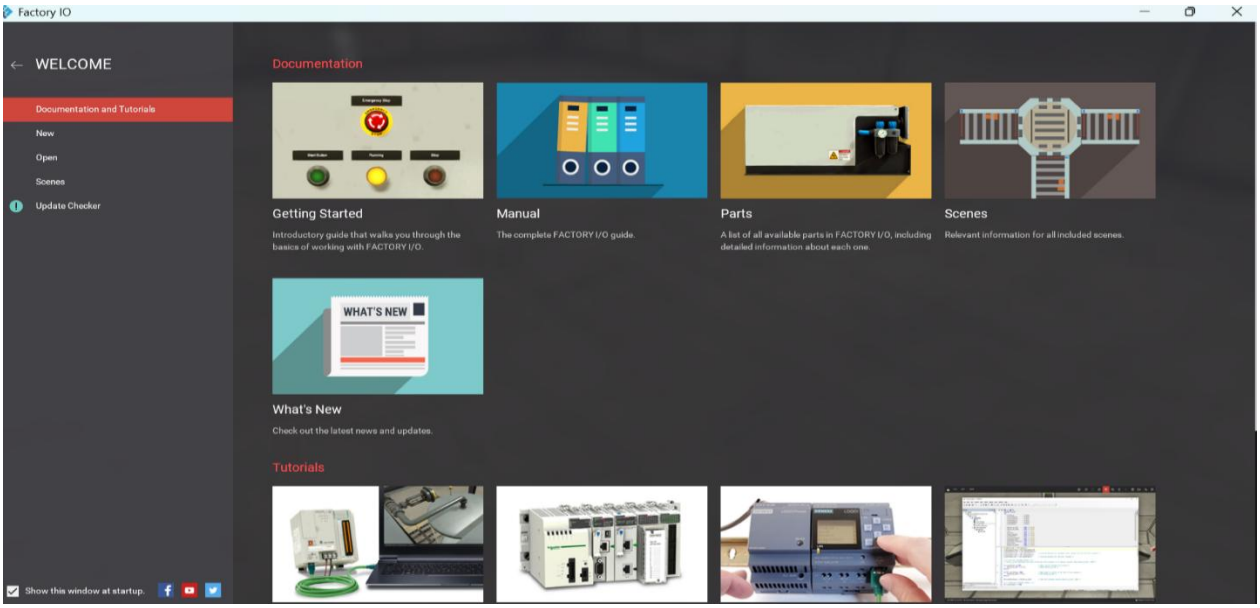

Figure III.1 : User interface of Factory io

III.2.1. Factory I/O Configuration

Once selected, the software transitions to the main workspace interface, where we can freely place and configure industrial components. This 3D workspace allows the insertion of devices such as tanks, valves, sensors, and actuators required for the simulation of the water tank filling process [20][21].

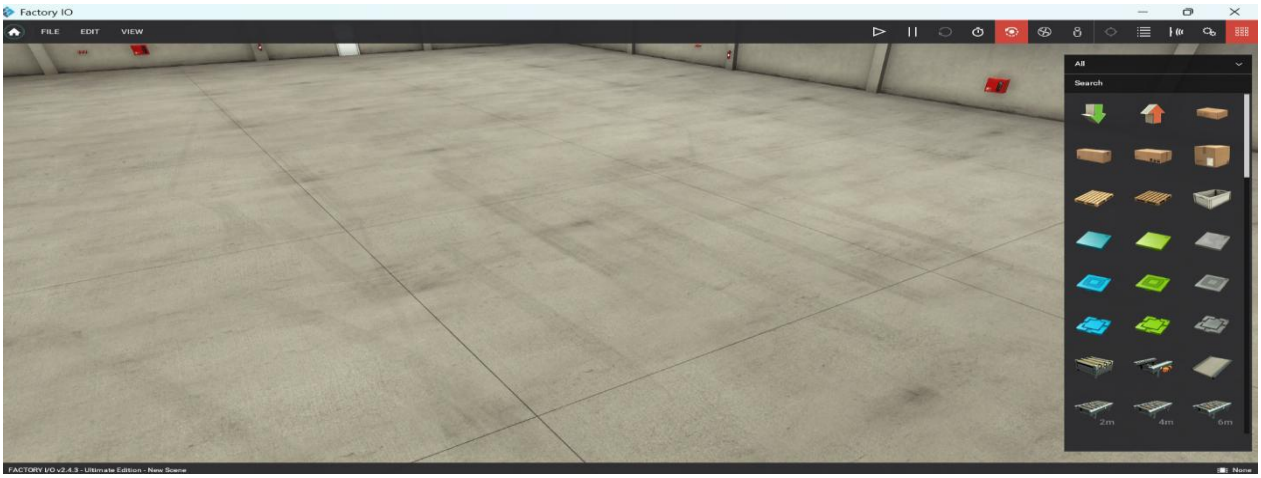

Figure III.2 : the workspace interface of Factory io

As shown in Figure III.3**,** we proceeded by selecting the tank system that we planned to work on. This predefined model includes the essential components required for our simulation, such as the tank itself, the filling and discharge valves, the level sensor, and the flow meter.

Choosing this model allowed us to save time and focus on programming and integration rather than assembling individual components manually.

Figure III.3 : the tank system

III.2.2. Driver selection

After selecting the appropriate tank model, we navigated to the top menu bar, clicked "File," and then selected "Drivers."

From the list of available drivers, we select the Siemens S7-1200/1500. This selection ensures compatibility with the PLC hardware settings in the TIA Portal and allows us to establish a convenient connection for real-time simulation between the Factory I/O module and the TIA Portal [20].

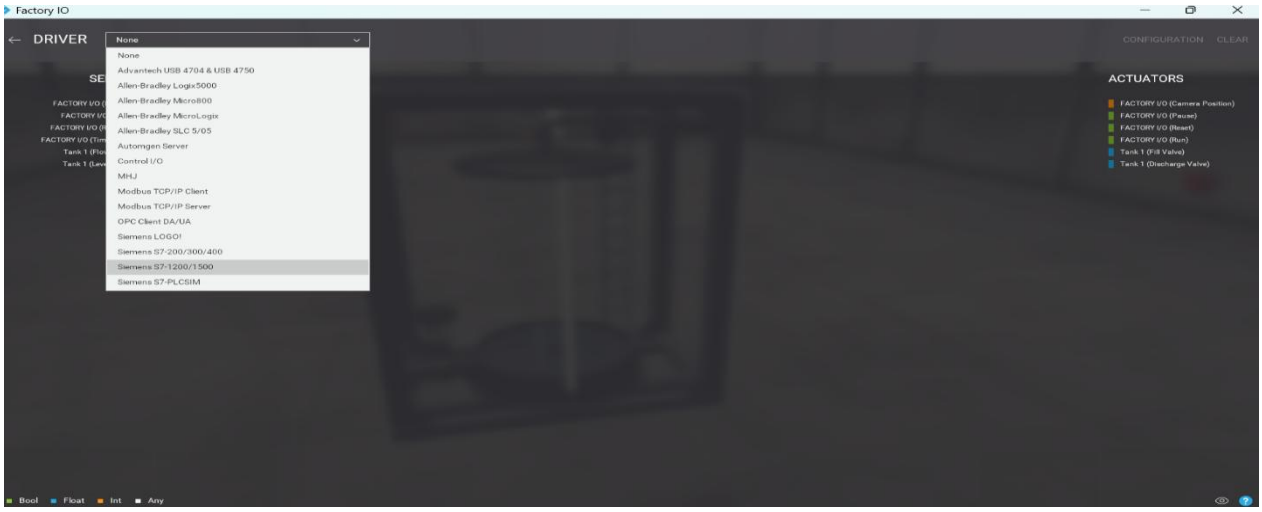

Figure III.4 : Hardware Configuration in Factory io

At this point, we could see in the figure the CPU appearing in the Drivers window within Factory I/O, indicating that the selected driver (Siemens S7-1200/1500) had been successfully configured inside the software.

It is important to note that this step does not represent an actual connection to TIA Portal yet.it simply shows that Factory I/O is now prepared to simulate communication with a compatible Siemens PLC.

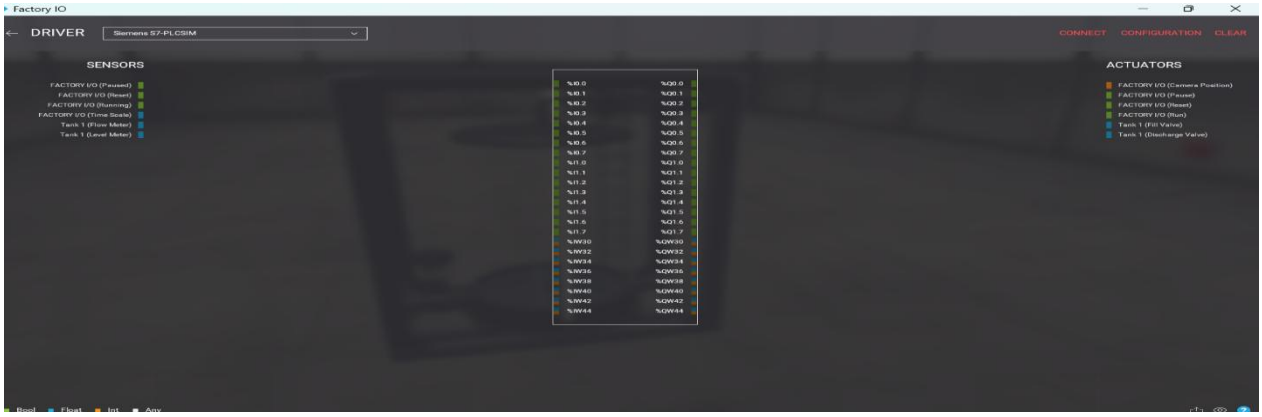

Figure III.5 : The PLC in Factory io.

III.2.3. I/O address assignment

After completing the driver configuration, we move on to defining the I/O addresses for the components used in our simulation.

In our setup, we assign the following addresses:

- **Inputs:**
  - `%IW0` (INT): assigned to the **Level Meter**, which measures the water level inside the tank.
  - `%IW2` (INT): assigned to the **Flow Meter**, which measures the flow rate during the discharge process.
- **Outputs:**
  - `%QW30` (INT): assigned to the **Filling Valve**, which controls the inflow of water into the tank.
  - `%QW32` (INT): assigned to the **Discharge Valve**, which controls the outflow of water from the tank.

These address mappings ensure proper data exchange later when we link the simulation to TIA Portal [15][20][21].

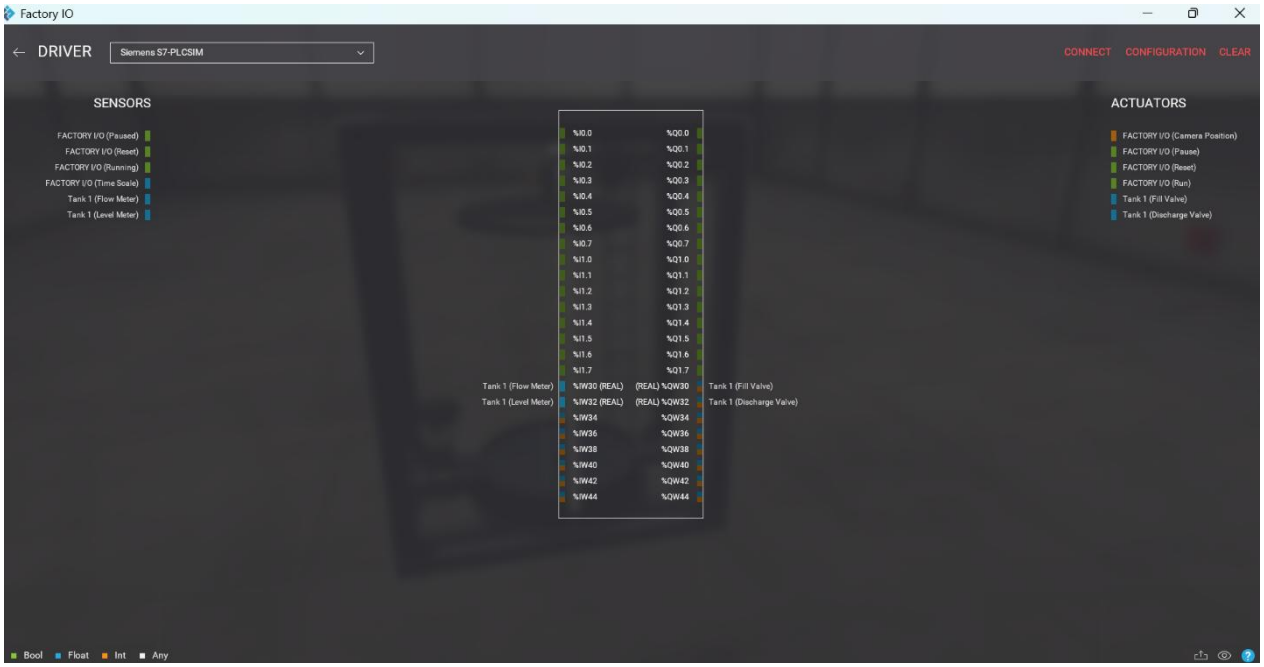

Figure III.6 : Defined I/O Addresses in Factory io.

III.2. Preparing the Factory I/O Template

Before starting the setup in TIA Portal, we ensure proper communication between Factory I/O and the PLC. Factory I/O requires a specific configuration template to establish this connection correctly [20].

III.2.1. Downloading and importing from the official website

We visit the official Factory I/O website, navigate to the Downloads section, and select the appropriate template. In our case, we choose the Factory I/O Template for S7-1500 (v15/v16/v17) because we use TIA Portal v16 and the S7-1500 series PLC [21].

This template provides a ready-to-use configuration with all necessary data blocks and settings to facilitate smooth integration with Factory I/O [21].

After downloading the template, we open TIA Portal. The main interface appears figure x, showing the startup screen and a list of recently opened or saved projects.

In our case, we click on Browse to locate and open the template project that we previously downloaded from the Factory I/O website, Figure III.7.

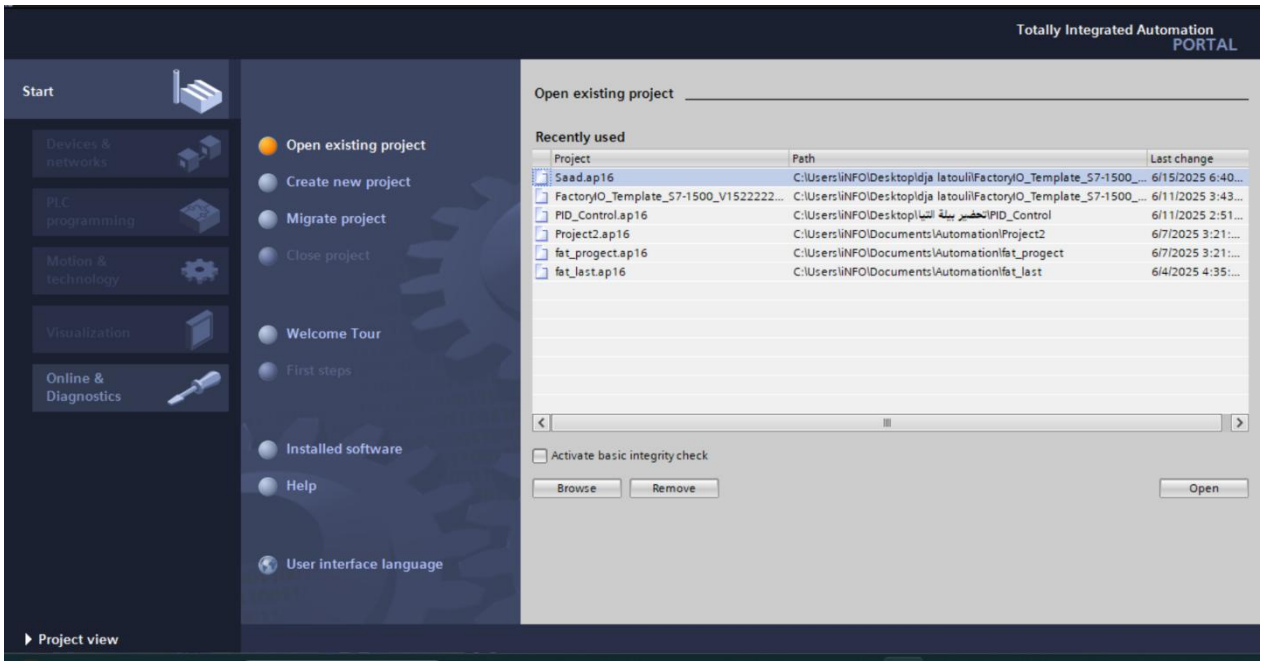

Figure III.7 : Tia Portal Interface : Project Selection Window

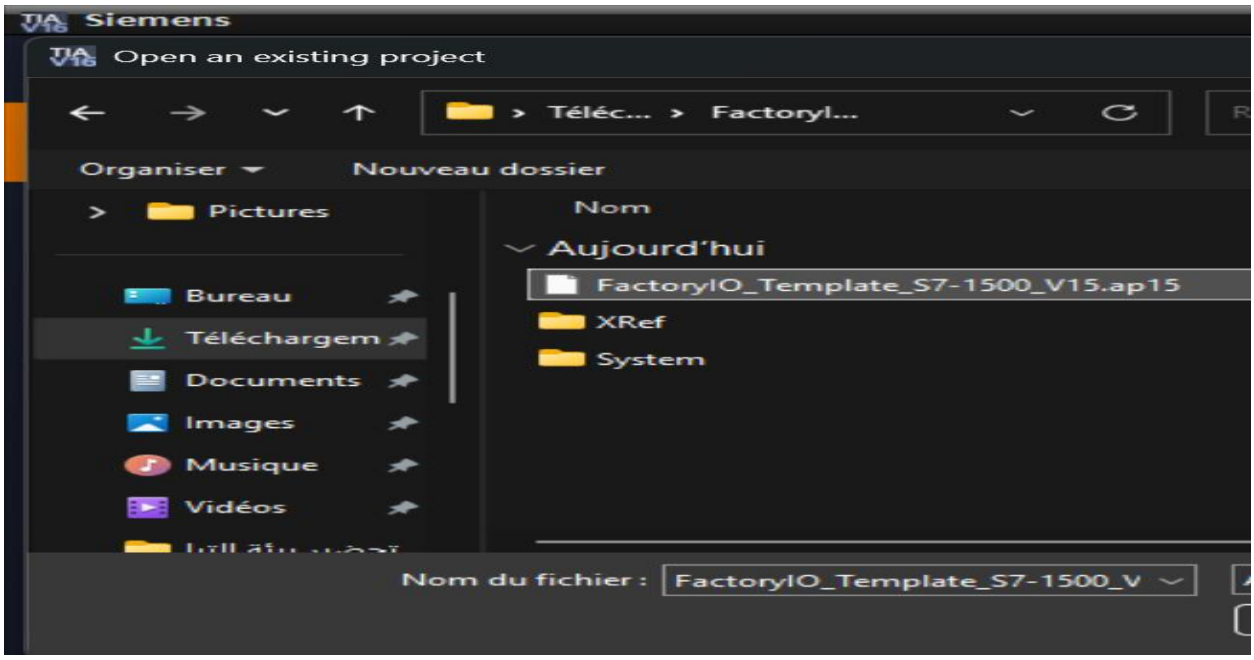

Figure III.8 : Browsing and Opening the Template

III.2.2. Adjusting for compatibility with TIA Portal v16

After selecting the template file, a window appears prompting us to upgrade the project because it was originally created in an earlier version of TIA Portal [15].

We click on Upgrade, and the system starts converting the project to match our current version (v16). This process may take a few moments depending on the system performance [14].

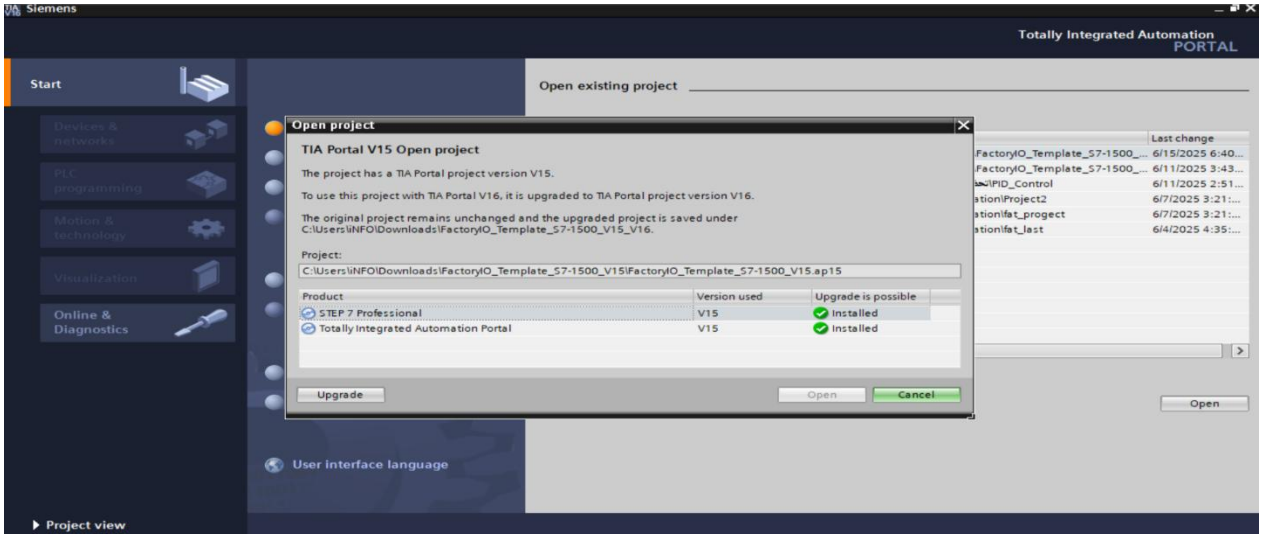

Figure III.9 : Project Upgrade Prompt in Tia Portal

III.3. TIA Portal Setup and Programming

Once the upgrade process is complete, the project opens and we can see its structure in the Project Tree.

As we explore the tree, we notice that the configured PLC is a CPU 1511-1 PN, which does not match the model we intend to use in our project. Our target device is CPU 1513-1 PN, so we need to replace the existing CPU to ensure compatibility with our simulation setup.

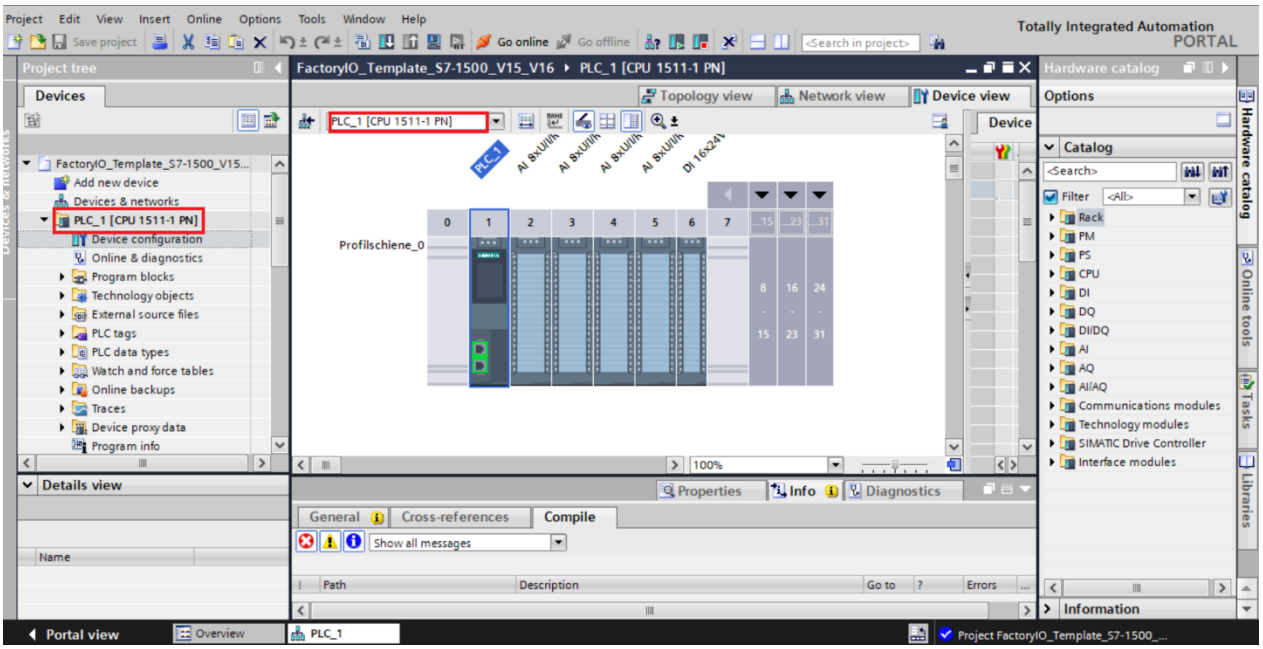

Figure III.10 : The Configured CPU 1511-1 PN that Comes with the Template

III.3.1. Project Initialization and Device Configuration

To proceed, we replace the existing CPU 1511-1 PN with the correct model, CPU 1513-1 PN, which matches the hardware configuration we define earlier.

III.3.1.1 Adding the HMI

Next, we add an HMI panel, specifically the TP900 Comfort, to our project. Once both devices are added as we can see in the figure x, we go to the Network View to ensure proper communication between the PLC and the HMI [14][15].

III.3.1.2 Connecting the HMI via PROFINET

In the Network View, we connect both devices through the PROFINET interface, assigning them to the same subnet to allow seamless data exchange during the simulation [14][15].

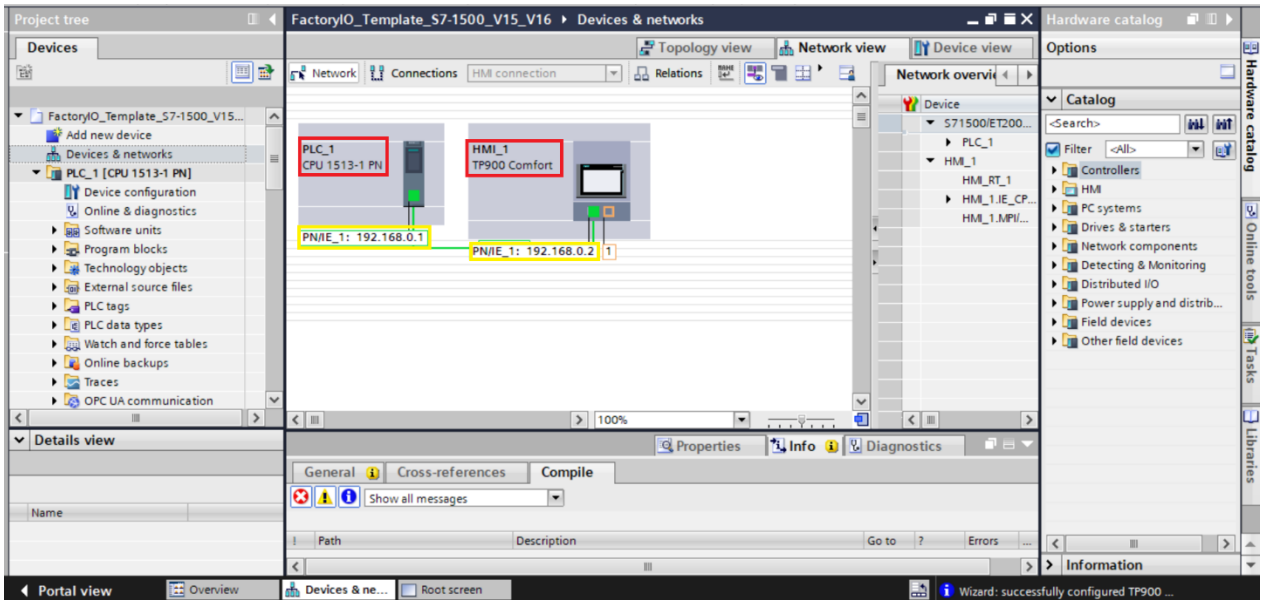

Figure III.11 : The connection between the HMI the PLC via PROFINET

III.3.2. Organization Blocks

In this step, we work with the program structure provided by the Factory I/O template and add additional blocks as needed to control the system As shown in the figure x we find.

- **Main [OB1]**

This is the default organization block in every TIA Portal project.

It executes cyclically and serves as the main entry point of the program.

We use OB1 to call our function blocks in a structured manner.

- **Factory IO Communication Block (FB)**

This Function Block (FB) is included in the downloaded template and is responsible for handling communication between the PLC and Factory I/O.

It maps the inputs and outputs of the simulation to the PLC tags, and must be called inside OB1 for proper data exchange.

- **Cyclic Interrupt [OB30]**

We create a new Cyclic Interrupt Organization Block (OB30) to handle time-sensitive logic. This block executes at regular time intervals, independently from OB1.

To add OB30, we right-click on Program Blocks, we choose Add new block, select Organization Block, and then choose Cyclic Interrupt as the type. We assign it the number OB30. Once created, we go to the PLC Properties → Cycle and Clock Memory and activate Interrupt OB30 by setting its cycle time. In our case, we use a cycle time of, for example, 100 ms, depending on the desired responsiveness of the control loop. Inside OB30, we place the logic that must run consistently and quickly, such as reading sensor values.

All these blocks can be easily located and accessed from the Project Tree, as illustrated in Figure III.12 [18] [15].

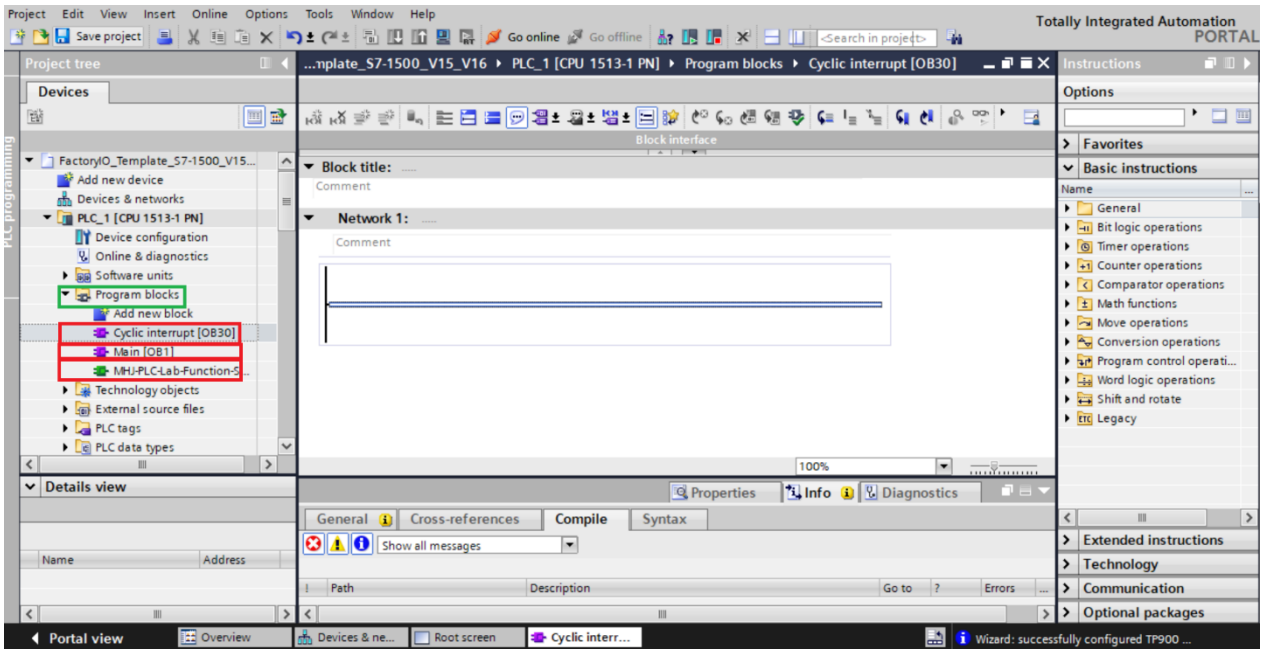

Figure III.12 : OB1, OB30 and Communication FB in Project Tree.

III.3.3. Signal Processing and Scaling

After setting up the hardware configuration, we focus on building the control logic inside the program blocks. Our main logic is organized within **OB1**, which we divide into structured **networks** to improve readability and function separation.

- Network 1 – Factory I/O Communication Block

In this network, we call the communication **Function Block (FC)** provided by the Factory I/O template.
This block manages the **data exchange between the PLC and the Factory I/O simulation**, ensuring that sensor values and actuator commands are correctly transferred in both directions. [3]

- Network 2 – Signal Conversion: INT to REAL

In this part of the program, we handle the **conversion of raw sensor data** into meaningful physical values.
We receive the input from the **Level Meter** in the tank as an **INT value** ranging from `0` to `27648`, which is the default scaling range for analog inputs on S7-1500 PLCs.

To process this signal:

1. We use the instruction **NORM_X** to normalize the input.
    - `NORM_X` converts the **raw INT input** into a **normalized REAL value** between `0.0` and `1.0`.
    - This instruction takes the current value, minimum, and maximum (e.g., 0 to 27648), and outputs a normalized ratio.

2. The output of `NORM_X` serves as the input to **SCALE_X**.
    - `SCALE_X` then converts the normalized value to the actual **physical range** of the tank level, which is from `0` to `300` centimeters.
    - This gives us a real-world representation of the water level in the tank.

This signal scaling process is essential for accurate monitoring and control of the tank's water level, allowing us to apply further logic based on meaningful values.

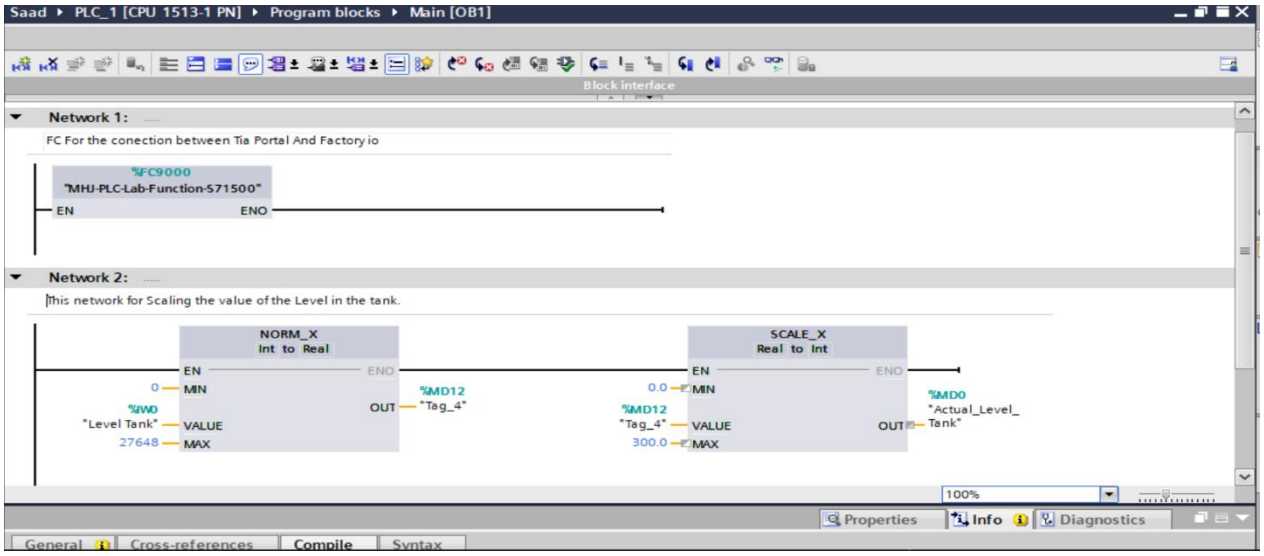

Figure III.13 : Analog Input Scaling for Level Meter.

III.3.4. PID Controller Implementation

After completing the signal processing inside OB1, we move to the Cyclic Interrupt Block OB30, where we implement the PID control loop**.**

III.3.4.1. Inserting the PID_Compact Block

From the Technology Objects section, we insert a PID_Compact block into Network 1 of OB30 [18][19].

- We connect the Input (PV_IN) to the actual tank level, which we obtain from the scaling process (using `NORM_X` and `SCALE_X`) inside OB1.
- We connect the Setpoint (SP_INT) to a tag that we later link to the HMI, allowing us to enter the desired level manually.
- We optionally connect PV_PER directly to the raw input from the Level Meter in case we want to bypass preprocessing and use the raw analog signal.
- The Output (LMN_PER) is connected to the actuator that controls the fill valve.

We use a MOVE instruction to transfer the output from the PID block to a specific Memory Word (MW), and then move it again to a Peripheral Output Word (QW) connected to the valve in Factory I/O.

III.3.4.2. PID Block Configuration

We open the PID block's configuration window and adjust the following settings:

- Under Control Type, we select Length**,** and set the unit to centimeters, matching the actual tank in Factory I/O.
- In the Set Mode, we choose Manual Mode, allowing us to manually control the output later from the HMI.

We define the process limits to reflect the physical characteristics of the tank:

- Minimum: **0 cm**
- Maximum: **300 cm**

Under PID Parameters, we enable Manual Entry to allow custom tuning.

- In the **Tuning Rule** section, we set the **Controller Structure** to **PID**, giving us proportional, integral, and derivative control.

With these settings, the PID_Compact block is fully configured to control the water level in the tank based on the desired setpoint and current level input [18][19].

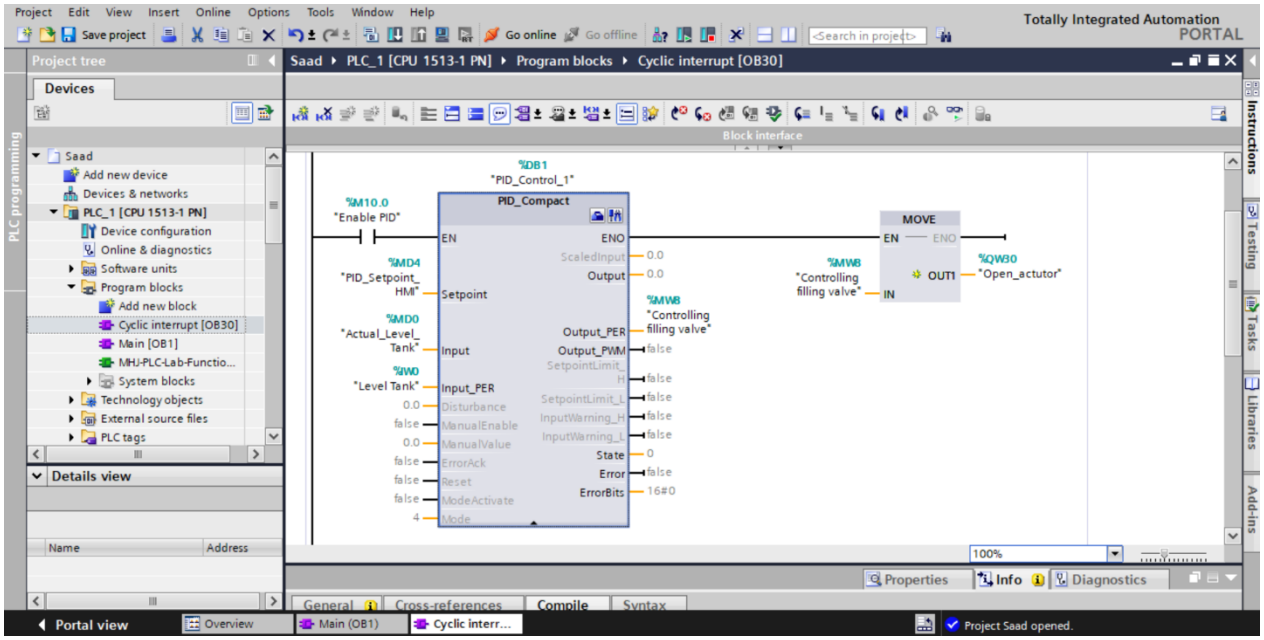

Figure III.14 : PID_Compact Block Implementation and Configuration in OB30

- **error computation**

In **Network 2** of the **Cyclic Interrupt OB30**, we calculate the **control error** between the desired water level and the actual level of the tank.

To do this, we use the **SUB (Subtract)** instruction to perform the following operation:

```
Error = Setpoint - Process Value
```

- The **Setpoint** is the value we define (or set via HMI) to represent the desired tank level.
- The **Process Value** is the real-time value obtained from the tank, after the scaling process.
- The result of this subtraction represents the **instantaneous error**, which we may use for monitoring, diagnostics, or further control calculations.

Placing this operation inside **OB30** ensures that the error is updated regularly and in sync with the **PID_Compact block**, which also runs in this interrupt block [15][21]..

This design maintains **data consistency and timing accuracy**, as both the PID control logic and the error calculation operate at the same time interval. This logic is illustrated in **Figure III.15**, which shows the subtraction operation implemented within Network 2 of OB30.

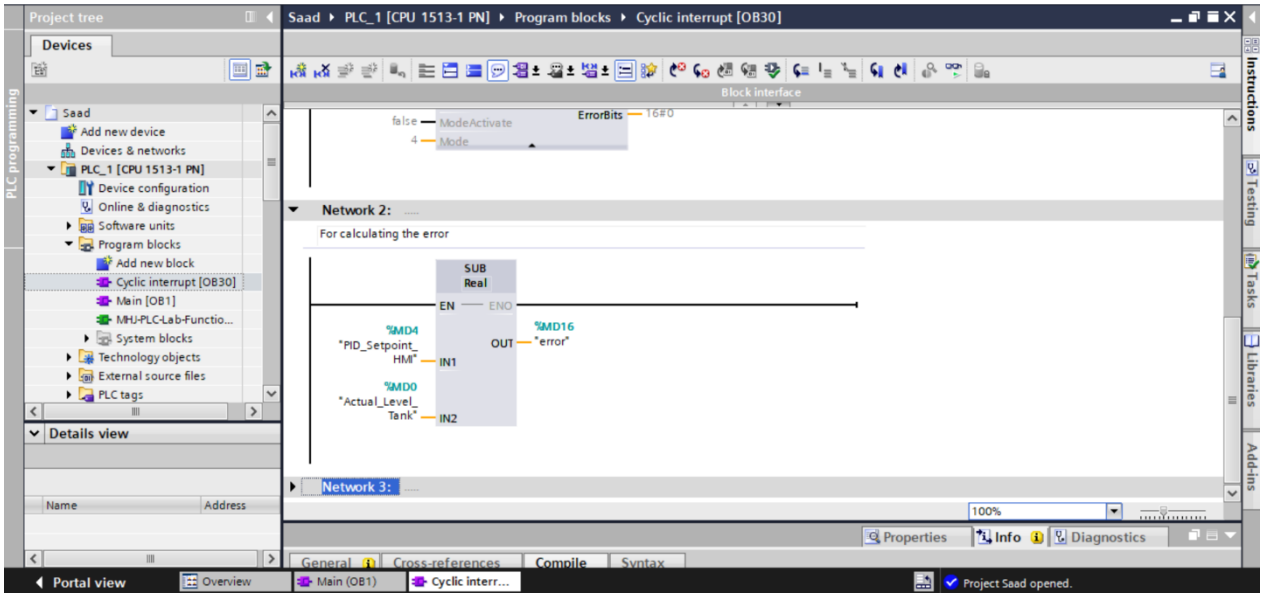

Figure III.15 : Error Computation Logic in OB30 Using SUB Instruction

III.5. Tag Tables and HMI Design

The PLC Tag Table is designed to define and manage all the variables used in the control logic. These tags serve as the interface between the PLC program, the HMI, and the external devices simulated in Factory I/O. Each tag plays a specific role in data acquisition, control signal generation, or human-machine interaction.

III.5.1. Summary of key memory tags and their functions

The following summarizes the key tags used in this project:

- **Enable_PID (%M10.0, Bool):**

This tag acts as a digital control signal used to enable or disable the PID block. It is configured as a Normally Open contact in the program and allows us to control whether the PID controller is active or not.

- **Level_Tank (%IW0, Int):**

This tag is linked to the level sensor in the tank within Factory I/O. The raw analog signal (0 to 27648) is used as input for the NORM_X instruction to normalize the value, and it is also connected to the PV_PER input of the PID block to optionally allow direct processing of the sensor signal.

- **Controlling_filling_valve (%MW8, Int):**

This tag receives the analog output from the PID controller (LMN_PER), representing the calculated opening percentage of the fill valve. This value is then transferred using a MOVE instruction to the actual output that drives the actuator.

- **Open_Actuator (%QW30, Int):**

This output tag is responsible for controlling the filling valve in Factory I/O. It receives its value from MW8 and sends the final control signal to the virtual actuator.

- **PID_Setpoint_HMI (%MD4, Real):**

This tag holds the setpoint value for the PID controller, which is configured and adjusted by the user via the HMI interface. It allows the operator to define the desired water level in centimeters.

- **Actual_Level_Tank (%MD0, Real):**

This tag contains the actual tank level in centimeters (0–300 cm), obtained after processing the raw input value through NORM_X and SCALE_X. It represents the real physical value used for feedback in the PID control loop.

- **Tag_4 (Intermediate Value) (%MD12, Real):**

This internal tag transfers the output from NORM_X to SCALE_X. It acts as a bridge between normalization and scaling steps.

- **Error (%MD16, Real):**

This tag stores the control error calculated by subtracting the actual tank level from the desired setpoint. The subtraction is performed in a SUB instruction located in the Cyclic Interrupt OB30, ensuring that error calculation is synchronized with the PID execution cycle.

This structured tagging approach ensures clarity in variable usage and traceability throughout the project, from simulation to control and visualization.

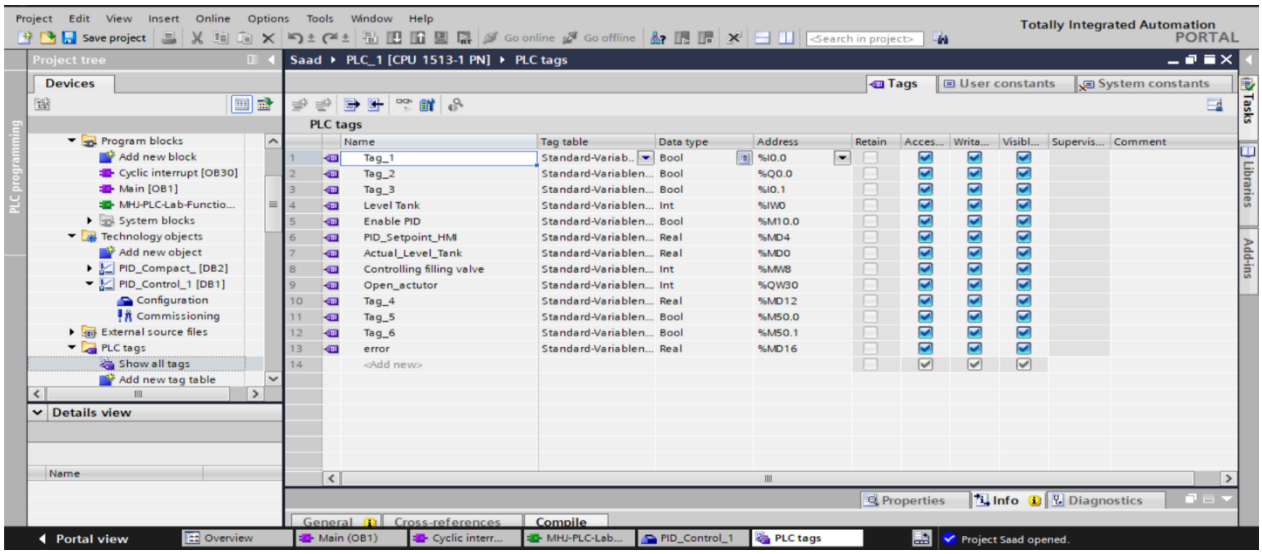

Figure III.16 : Snapshot of PLC Tag Table in TIA Portal

III.5.2. HMI Interface

The HMI screen shown in the image was designed to provide an interactive, real-time monitoring and control interface for a closed-loop PID system regulating tank level. This interface allows the operator to visualize process variables, enter control parameters, and manage the system's operation seamlessly.

Layout Overview :

The screen is composed of the following essential components, organized logically from left to right:

- **Setpoint Input (SP):**

This input field is bound to the tag %MD4 (PID_Setpoint_HMI) of type Real. The operator uses this field to enter the desired tank level in centimeters, ranging from 0 to 300 cm. This value is linked directly to the SP_INT input of the PID block.

- **Error Display:**

A real-time numeric display shows the control error value. It is computed in the OB30 block as the difference between the setpoint and actual level, using the instruction SUB.Connected to tag %MD16 (error).

- **Enable PID Button:**

A digital button configured to write to %M10.0 (Enable_PID). Acts as a manual switch to activate or deactivate the PID block, ensuring safe and controlled testing.

- **PID Parameters Display (Ki, Ti, Td):**

Real fields displaying the proportional (Kp), integral (Ti), and derivative (Td) gains of the PID. These fields are optional in runtime for tuning purposes, based on the values configured in the PID Compact block. Tuning is enabled via the “Manual Entry” setting in the PID block parameters.

- **Process Variable (PV) Display:**

Shows the actual tank level in centimeters in real-time. Bound to tag %MD0 (Actual_Level_Tank), which reflects the normalized and scaled signal coming from the level sensor.

- **PID Output (OUT):**

Displays the output of the PID controller. This value represents the opening percentage of the filling valve and is transferred to the actuator via %MW8 → %QW30.

- **Graphical Tank Representation:**

A symbolic graphic of the tank reflects its current fill level visually. The level is dynamically updated based on the process variable, offering an intuitive process overview. At the bottom of the screen, a chart to present :

used for trend monitoring of setpoint, process value, and output over time, though it requires configuration under "Traces" or "Trends" in the HMI settings.

This HMI interface provides all necessary control and feedback tools for the operator to:

- Adjust the system setpoint.
- Monitor the current tank level.
- Enable/disable PID control.

- Observe error and PID behavior in real time.

The screen is compact yet comprehensive, ensuring usability and clarity, and complements the control logic implemented in OB1 and OB30.

The interface shown in **Figure III.17** represents the HMI screen as designed , prior to runtime simulation. It highlights the layout and structure of interactive elements used for monitoring and controlling the tank level.

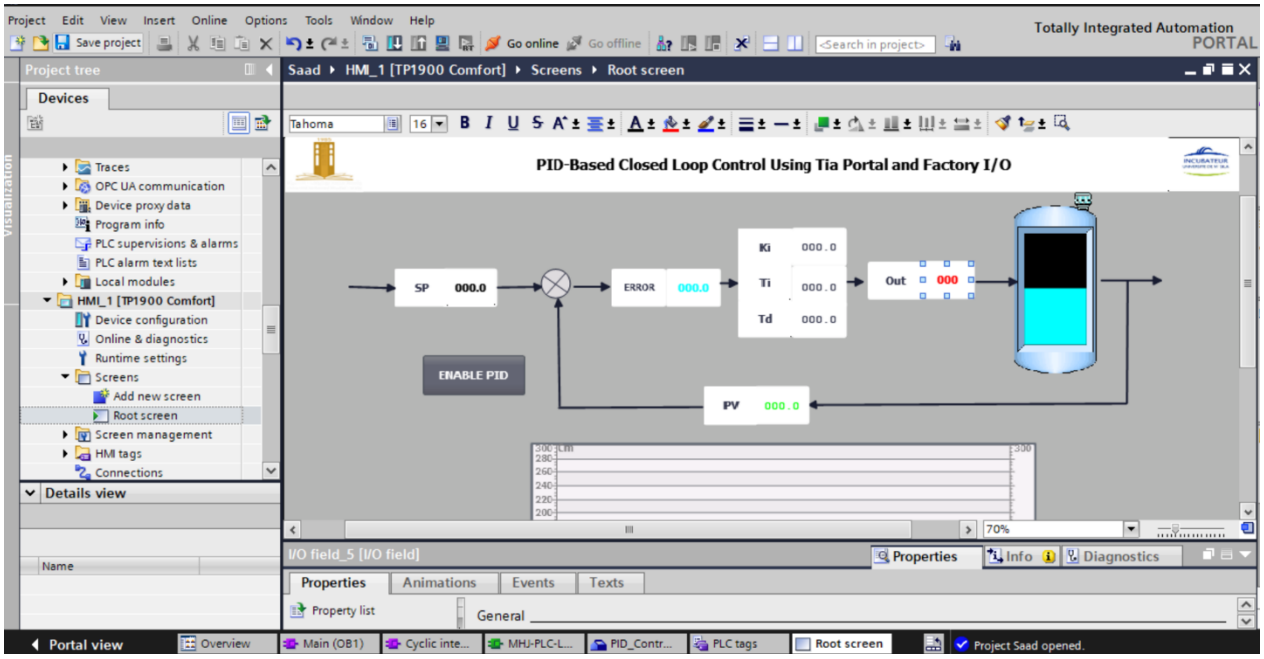

Figure III.17 : HMI Layout (Design View).

III.5.3. HMI tag table

The HMI tag table was configured in the TIA Portal to establish communication between the HMI screen and the PLC. These tags allow for both visualization and control of key process variables within the PID-based closed-loop control system.

The following tags were defined and used:

- **Actual_Level_Tank:**

A Real-type variable representing the actual measured value of the tank level (Process Variable - PV). It is connected to %MD0 in the PLC and is visualized numerically and graphically on the tank image in the HMI screen.

- **Enable PID:**

A Bool-type tag used to activate or deactivate the PID controller. It is linked to %M10.0 in the PLC and is triggered by a button on the HMI interface labeled "ENABLE PID".

- **error:**

A Real variable showing the real-time error value, which is the difference between the setpoint and the actual level. This value is computed cyclically in OB30 and updated continuously for monitoring purposes.

- **PID_Control_1_CtrlParamsBack.Ki, Ti, and Td:**

These are Real-type variables used to monitor (and optionally adjust) the PID tuning parameters. They represent the integral gain (Ki), integral time (Ti), and derivative time (Td) respectively, and are extracted from the PID Compact block instance DB.

- **PID_Control_1_Output_PER:**

An Int variable representing the PID output in percentage form. This value is ultimately used to control the opening of the filling valve in Factory I/O.

- **PID_Setpoint_HMI:**

A Real-type tag that serves as the user-defined setpoint input from the HMI. It is connected to %MD4 in the PLC and allows the operator to define the desired tank level directly from the interface.

The next figure, **Figure III.18**, shows the HMI tag table as configured in TIA Portal, illustrating the linkage between the interface elements and the corresponding PLC addresses.

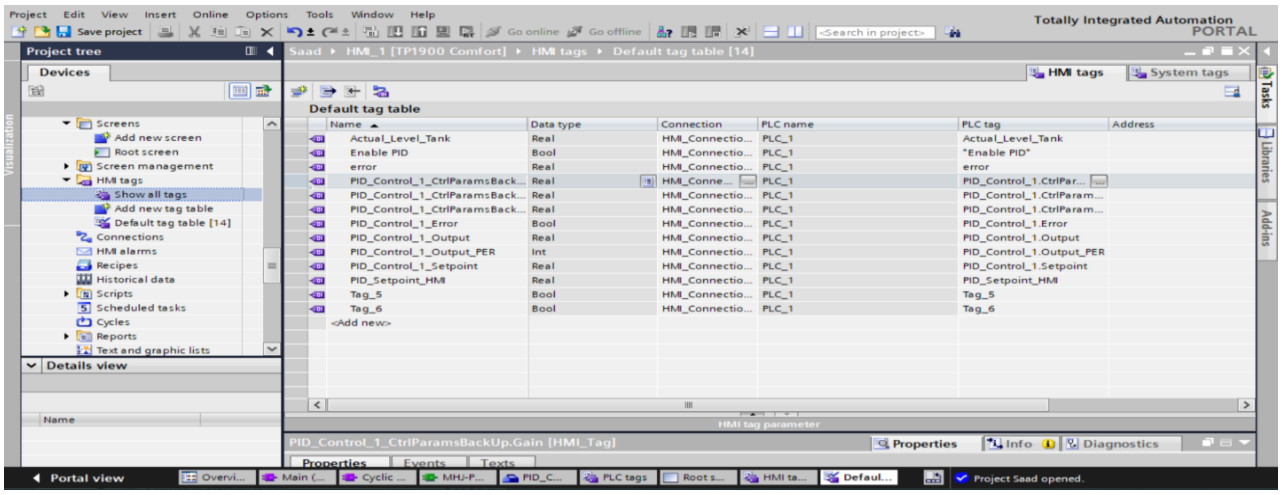

Figure III.18 : HMI Tag Table in TIA Portal

III.6. Simulation Execution

After completing the programming and configuration of the project, the following steps are performed to launch the simulation and record the system's open-loop response before enabling the PID controller [15][18][21]:

III.6.1. Compile and Save the Project

We begin by compiling the PLC project within TIA Portal to ensure that no errors are present. Once the compilation is complete and error-free, we save the project.

Simultaneously, the scene created in Factory I/O, including all virtual components (tank, valves, sensors), is saved to preserve the simulation environment.

III.6.2. Launching the PLC Simulation

We then move to the Project Tree in TIA Portal and click once on the configured PLC. From the top menu bar, we click on "Start Simulation", which launches the PLCSIM window**.**

Initially, the CPU appears in STOP mode. To activate it, we must load the project into PLCSIM. In the pop-up Load Preview window, we click:

- Load
- Then Start Module
- Finally, Finish

At this point, the simulated CPU switches to RUN mode, indicating it is ready to execute the logic.

III.6.3. Simulating the HMI Runtime

Next, we simulate the HMI interface. We select the configured HMI panel (e.g., TP900 Comfort) in the Project Tree, then click on “Start Simulation” from the top menu.

This launches the SIMATIC WinCC Runtime Advanced, where the HMI screen appears and becomes operational, ready for user interaction and monitoring.

III.6.4. Connecting Factory I/O to PLCSIM

With the PLC and HMI simulations running, we switch to Factory I/O:

- Open the saved project scene.
- Go to File > Drivers in the top menu bar.
- Select the “Siemens S7-1200/1500” driver.
- Click “Connect” to link Factory I/O with PLCSIM.

This establishes communication between the virtual environment and the PLC simulation.

II.6.5. Starting the Factory I/O Simulation

Once the connection is successful, we return to the main screen in Factory I/O and click on “Run”. This activates the simulation in real time, where sensor signals and actuator responses are exchanged with the PLC.

II.6.6 Activating Monitoring in TIA Portal

Finally, we return to TIA Portal and press:

- “Go Online” to connect with the PLCSIM instance.
- Then click “Monitoring On/Off” to activate online monitoring of logic execution, variable states, and process feedback.

At this stage, the full simulation cycle is live, enabling us to observe the open-loop behavior of the tank system.

III.7. Open-Loop Testing Procedure

To obtain the system's open-loop step response essential for later PID tuning, we perform the following steps while the full simulation environment (PLC, HMI, and Factory I/O) is running [18][19].

III.7.1. Open-Loop Step Response

Deactivating the PID Block Temporarily To ensure that the PID controller does not influence the system response, we must deactivate it. This allows the filling valve to be opened freely, and the tank level to rise without any automatic control intervention.

We open the Cyclic Interrupt OB where the PID_Compact block is placed. We click on "Commissioning" from within the block interface. Then we press the "Stop PID_Compact" button.

At this point, the PID controller enters a Disabled-Inactive state. Although structurally still in the program, it acts as a passive link similar to a simple wire without applying any control logic.

- **Initiating the Open-Loop Step Test**

We switch to Factory I/O and manually open the filling valve to 100% instantly (step input). This step change causes water to begin filling the tank, and the level increases over time.

- **Monitoring and Recording the Response**

We return to TIA Portal and monitor the real-time value of the tank level via: The HMI screen, where the process value is visualized. Or directly through the monitored PLC tags inside the program.

The tank level evolution is recorded over time to form the system's step response curve as shown in Figure III.19.

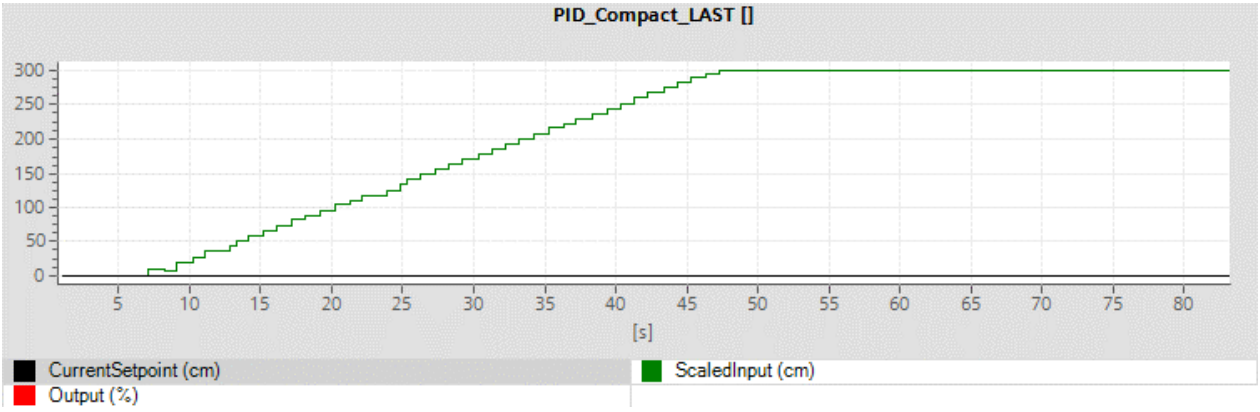

Figure III.19 : Step Response Curve Captured from TIA Portal Monitoring during Factory I/O Simulation

III.7.2 Step Response Analysis and Ziegler–Nichols Tuning Method

from The recorded step response curve of the tank filling process obtained while the PID controller was disabled we got our Observation therefore we can say that we have :

- Step begins (delay time L) at: 7.5 seconds
- Point of inflection (maximum slope): 24 seconds
- Final steady-state value reached at: 50 seconds

From this, we calculate:

- L (Delay Time) = 7.5 s
- T (Time Constant) = 24 − 7.5 = 16.5 s

Having determined L and T, we proceeded to calculate the controller parameters using the Ziegler–Nichols Open-Loop Tuning Method, which provides empirical formulas for different controller types as summarized in the following table:

Controller Type	Kp	Ti	Td
P	Kp=LT	—	—
PI	Kp=0.9·LT	Ti=3.33·L	—
PID	Kp=1.2·LT	Ti=2·L	Td=0.5·L

Table II.1: Empirical Ziegler–Nichols Open-Loop Tuning Formulas for P, PI, and PID Controllers [22][9].

Substituting the values **T = 16.5 s** and **L = 7.5 s** into the above expressions, we obtained the following numerical results:

Controller	Kp	Ti	Td
P	2.20	—	—
PI	1.98	24.98	—

PID	2.64	15.0	3.75

Table II.2: Computed P, PID and PI Tuning Parameters Based on L = 7.5 s and T = 16.5 s

These parameters served as the initial tuning values for our PID_Compact block in the TIA Portal environment. In the next phase, we integrated them into the controller and tested the closed-loop performance of the system under varying conditions to validate their effectiveness.

III.8. Closed-Loop Operation and PID Activation

After extracting the initial tuning parameters from the open-loop response, we activated the PID controller and tested the system in closed-loop mode to evaluate its behavior under varying setpoints and disturbance conditions.

III.8.1. Activating the PID Controller

We navigated to the Cyclic Interrupt OB, where the `PID_Compact` block is integrated.

- We opened the Commissioning window.
- Then, we clicked "Start PID_Compact" to activate the control block and enable automatic regulation of the filling valve based on the tank level and the target setpoint [18][23].

III.8.2. Controller Parameters Based on Ziegler–Nichols Tuning

Two controller configurations were tested, both derived using the Ziegler–Nichols open-loop method:

a) PID Controller:

- Kp = 2.64
- Ti = 15.0
- Td = 3.75

b) PI Controller:

- Kp = 1.98
- Ti = 24.98

These parameter sets were entered using the standard configuration interface in the PID_Compact block under the Tuning Rule section.

III.8.3. Testing Procedure: Setpoint Transitions with Disturbance Variation

To comprehensively evaluate system performance and controller robustness, we designed a series of tests combining both setpoint transitions and disturbance variations. The procedure involved:

- Starting from an initial tank level of **0 cm**.
- Applying a first setpoint of **100 cm**, and observing the system until it reached stability.
- Increasing the setpoint to **300 cm**, then later decreasing it to **200 cm** once stability was achieved.

In parallel with each setpoint change, we manually varied the discharge valve opening percentage (e.g., 30%, 50%, and 70%) to simulate unpredictable fluid losses and observe the controller's ability to maintain the desired level [18][23].

These combined conditions helped test both the tracking capability and the disturbance rejection performance of each control configuration. The system was monitored in real time using WinCC Runtime Advanced**,** focusing on:

- Rise time
- Overshoot
- Settling time
- Steady-state error

III.8.4. Final Evaluation and Controller Selection

After conducting all tests, the **Ziegler–Nichols PI controller** emerged as the most effective solution. It delivered:

- Low overshoot
- Smooth recovery from disturbances
- Stable performance across different setpoint transitions
- Easier configuration due to fewer tuning parameters

III.8.5. Final Demonstrated Response

The final system response we selected for documentation corresponds to the following test condition:

- The **discharge valve** was set to a fixed opening of **50%**.
- The **initial tank level** was at **00 cm**.
- A **setpoint of 200 cm** was applied and held constant.

We captured snapshots from the real-time simulation environment combining SIMATIC WinCC Runtime Advanced (HMI interface) and Factory I/O (physical process simulation). These visual results serve as concrete evidence of the system's performance after the entire development, tuning, and validation cycle.

In the first image, we observe the system's immediate response after applying a setpoint of 200 cm, with the discharge valve fixed at 50% opening. The key indicators on the HMI screen confirm the correct functioning of the controller:

- SP (Setpoint): 200.0
- PV (Process Value): 202.0
- Error: -2.0
- Controller Output (Out): 55
- Controller Parameters (PI): Kp = 2.0, Ti = 25.0, Td = 0.0

Meanwhile, the graph in the lower section of the screen shows the system’s trajectory toward the setpoint, where the green curve (PV) is approaching the black line (SP), while the cyan line (Error) is decreasing progressively.

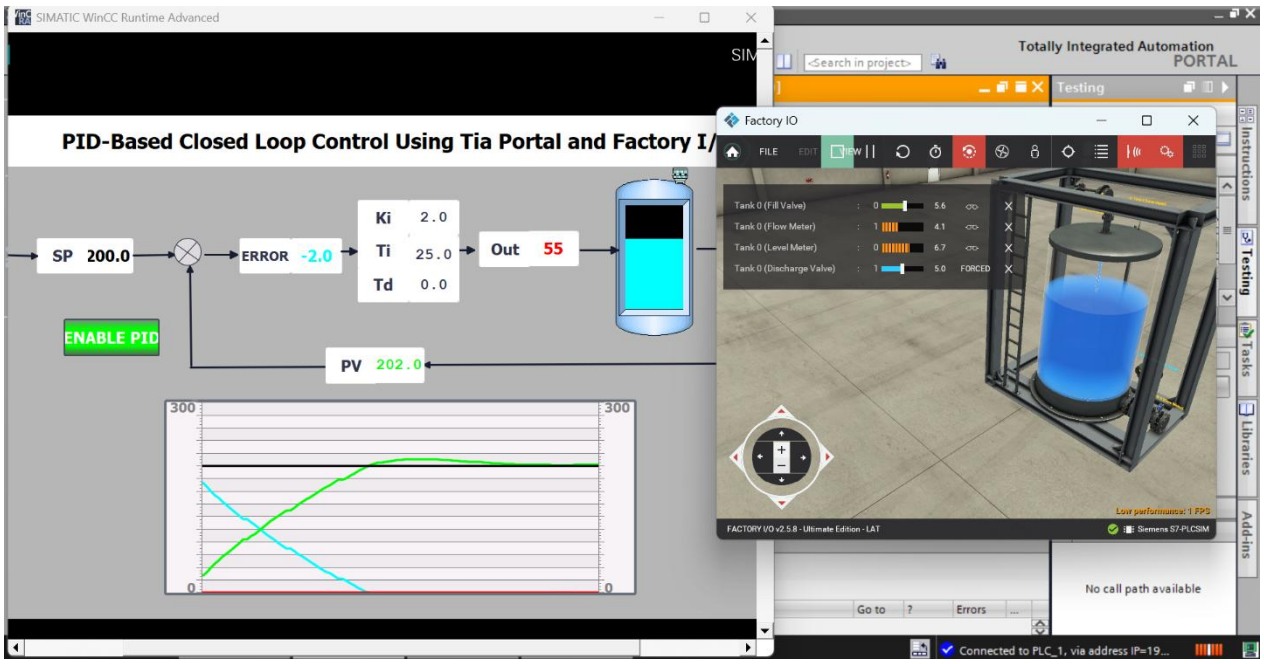

Figure III.20: Initial Closed-Loop Response (Setpoint = 200 cm, Valve Opening = 50%) with Synchronized HMI and Factory I/O Views

In the second image, captured shortly afterward, the system reaches a perfect steady-state:

- PV equals SP (200.0 cm) with Error = 0.0, indicating full convergence.
- The controller maintains stable output to counterbalance the continuous discharge effect.The plotted curves confirm minimal overshoot and excellent regulation.

This visual validation illustrates that the PI controller, tuned using the Ziegler–Nichols open-loop method, not only stabilized the tank level effectively but also responded well to both transient and steady-state conditions. Such behavior underlines its suitability for real-world applications involving liquid level control in process automation.

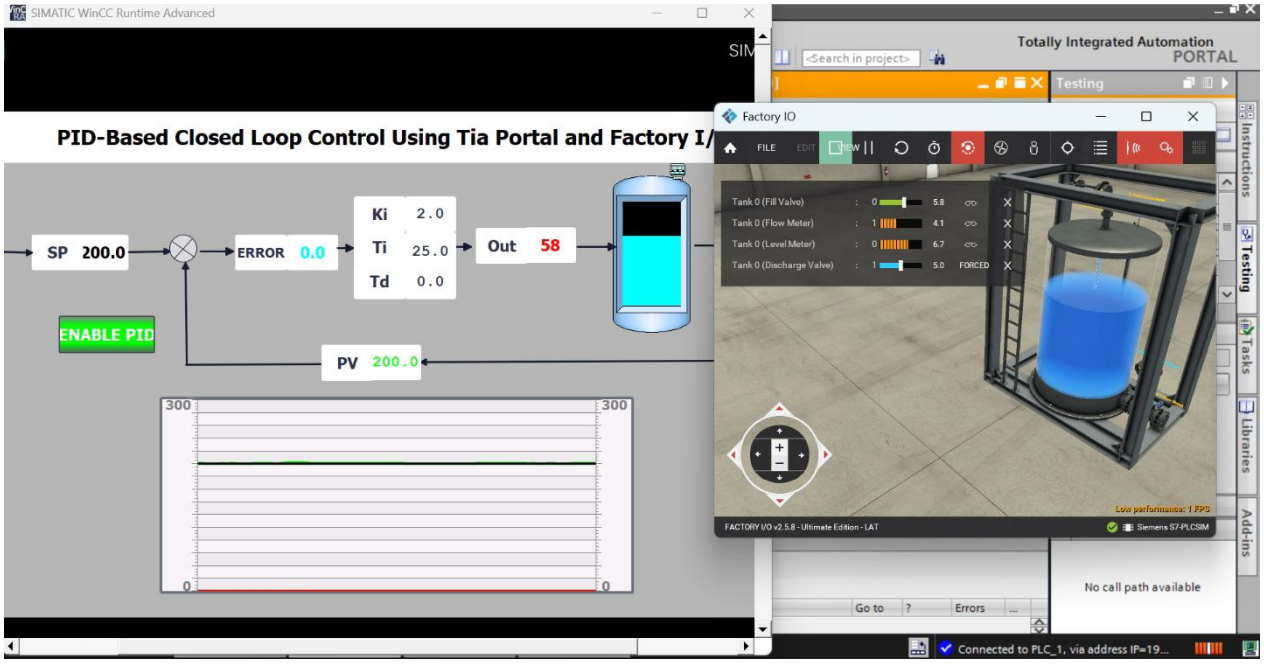

Figure III.21: Steady-State Achievement: Tank Level Regulated at 200 cm (Combined HMI and Factory I/O Views).

Conclusion

This chapter demonstrated the complete implementation and validation of a closed-loop PID control system for tank level regulation using TIA Portal and Factory I/O. We began by building the simulation environment, configuring hardware components, and assigning I/O addresses. Through careful integration between Factory I/O and TIA Portal, we established real-time communication and executed a realistic process simulation.

Open-loop testing allowed us to analyze the system's response and apply the Ziegler–Nichols method for PID tuning. Both PID and PI configurations were tested, with the PI controller offering superior performance in terms of stability, disturbance rejection, and ease of tuning.

The structured use of OB1 and OB30 ensured responsive control execution, while the HMI interface provided a user-friendly platform for monitoring, interaction, and real-time parameter adjustment.

In conclusion, this project successfully bridged theory and practice. It highlighted the value of simulation tools in control engineering and emphasized the importance of precise tuning and structured programming in developing efficient industrial automation solutions.

General Conclusion

In conclusion, this project has demonstrated the value of integrating theoretical knowledge with practical application in the field of industrial automation and control. By utilizing advanced tools such as TIA Portal and Factory I/O, we successfully designed and implemented a closed-loop control system based on a PID algorithm to regulate the water level in a tank. This allowed for realistic simulation of operational scenarios and thorough analysis of the system's dynamic behavior under varying conditions. The work highlighted the importance of precise tuning, structured programming, and effective communication between industrial components. The results confirmed the effectiveness of an integrated learning environment in building a solid foundation for students and engineers aiming to enhance their skills in automation and control. As such, this experience serves as a replicable and scalable model for future educational or research-oriented projects.

Références

[1] Åström, K. J., & Hägglund, T. (2006). *Advanced PID Control*. Instrumentation, Systems, and Automation Society (ISA).

[2] Seborg, D. E., Edgar, T. F., Mellichamp, D. A., & Doyle, F. J. (2010). *Process Dynamics and Control* (3rd ed.). John Wiley & Sons.

[3] Ziegler, J. G., & Nichols, N. B. (1942). "Optimum Settings for Automatic Controllers." *Transactions of the ASME*, 64(8), 759–768.

[4] Dorf, R. C., & Bishop, R. H. (2017). *Modern Control Systems* (13th ed.). Pearson.

[5] Nise, N. S. (2015). *Control Systems Engineering* (7th ed.). Wiley.

[6] Bequette, B. W. (2003). *Process Control: Modeling, Design, and Simulation*. Prentice Hall.

[7] Kuo, B. C., & Golnaraghi, F. (2002). *Automatic Control Systems* (8th ed.). John Wiley & Sons.

[8] Marlin, T. E. (2000). *Process Control: Designing Processes and Control Systems for Dynamic Performance*. McGraw-Hill.

[9] Ogata, K. (2010). *Modern Control Engineering* (5th ed.). Prentice Hall.

[10] Johnson, C. D. (2006). *Process Control Instrumentation Technology*. Pearson Education.

[11] Franklin, G. F., Powell, J. D., & Emami-Naeini, A. (2015). *Feedback Control of Dynamic Systems* (7th ed.). Pearson.

[12] Bennett, S. (1993). *A History of Control Engineering, 1930–1955*. Institution of Engineering and Technology.

[13] Coughanowr, D. R., & LeBlanc, S. E. (2008). *Process Systems Analysis and Control* (3rd ed.). McGraw-Hill.

[14] Siemens AG. (2020). *TIA Portal V17 – Getting Started Manual*.

[15] Siemens AG. (2021). *SIMATIC STEP 7 Basic/Professional V17 and WinCC V17*.

[16] Berger, H. (2012). *Automating with SIMATIC S7-1500*. Publicis.

[17] Katalinic, B. (2015). *DAAAM Scientific Book – TIA Portal in Education and Industry*.

[18] Siemens AG. (2021). *PID Control in TIA Portal – Application Notes*.

[19] Rentschler, M. & Schmieder, M. (2020). *Automating with TIA Portal*. Springer.

[20] Real Games. (2021). *Factory I/O – User Manual*. Real Games.

[21] Real Games. (2022). *Factory I/O Educational Resources Guide*.

[22] Frey, G., & Litz, L. (2019). *Modeling and Simulation in Automation Systems*. Springer.

[23] Lipták, B. G. (2018). *Instrument Engineers' Handbook: Process Control and Optimization*. CRC Press.

[24] Real Games. (2020). *Using Factory I/O with Siemens TIA Portal*. Real Games Tutorials.

[25] Vicente, P., & Sousa, J. (2020). "Simulation-Based Training in Industrial Automation Using Factory I/O," *Procedia Computer Science*, 176, 2922–2927.