PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

Ministry of Higher Education and Scientific Research

University 8 Mai 1945 – Guelma

Faculty of Science and Technology

Department of Electronics and Telecommunications

End of Study Master's Thesis

Presentation for the ACADEMIC MASTER'S Diploma

Domain: **Science and Technology**

Sector: **Electronics**

Specialty:**Instrumentation**

Supervision and Control of a DC motor using a SIEMENS Programmable Logic Controllers and a SCADA System

---

Presented by:

Himoud anis

Chouini mohammed lakhdar

Supervisor:

Dr. NEMISSI Mohamed

JUNE 2025

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Dédicace

In the name of Allah, the most merciful, and the most compassionate

I dedicate this work and give special thanks, to my beloved parents my father ‘**Salah**’ who taught me the value of hard work and my mother ‘**Fadila**’ for her love, tenderness, encouragement and care.

To my brothers ‘**Wassim**’ and ‘**Younes**’ for their support.

ANIS

In the name of Allah, the most merciful, and the most compassionate

I dedicate this work and give special thanks, to my beloved parents my father may Allah have mercy on him ‘**Ibraim**’ who taught me the value of hard work and my mother ‘**Samia**’ for her love, tenderness, encouragement and care.

To my sisters ‘**Nehed**’ and ‘**Ahlam**’ for their support.

Mohammed Lakhder

Acknowledgement

First and foremost, we want to thank Allah, the most powerful of all, for providing us with the fortitude and perseverance to complete this task.

Dear and beloved supervisor Dr. NEMISSI Mohamed worked closely with us from the start, commented on our work at every stage, and insisted on perfection until the completion of our dissertation. Without his guidance, assistance, and encouragement, we would never have been able to complete this piece of work.

Summary

This Master's thesis focuses on the implementation and supervision of a DC motor control system using a PLC (Programmable Logic Controller) and a SCADA (Supervisory Control and Data Acquisition) interface, specifically employing a PID (Proportional-Integral-Derivative) control algorithm. The work details the configuration and programming of the PLC for precise motor speed regulation, including the scaling of analog inputs and outputs, and the integration of the PID function block. Concurrently, it outlines the development of a SCADA interface that provides real-time monitoring of motor parameters, allows for the adjustment of PID tuning gains (Proportional, Integral, and Derivative), and facilitates operational control through dedicated commands for starting, stopping, and resetting the system. This research demonstrates a practical application of industrial automation principles, highlighting the synergistic relationship between PLCs for robust control execution and SCADA systems for comprehensive supervisory management inachieving efficient and accurate DC motor operation.

Résumé

Ce mémoire de Masterporte sur la mise enœuvre et la supervision d'un système de contrôle de moteur à courant continuutilisant un automate programmable (PLC) et une interface SCADA (Supervisory Control and Data Acquisition), en particulier enemployant un algorithme de contrôle PID (Proportional-Integral-Derivative). Ce travail détaille la configuration et la programmation de l'automate pour unerégulationprécise de la vitesse du moteur, y compris la mise à l'échelle des entrées et sorties analogiques, et l'intégration du bloc fonctionnel PID. Parallèlement, il décrit le développementd'une interface SCADA qui fournitune surveillance en temps réel des paramètres du moteur, permetl'ajustement des gains de réglage PID (proportionnel, intégral et dérivé) et facilite le contrôleopérationnel grâce à des commandesdédiées au démarrage, à l'arrêt et à la réinitialisation du système. Cette recherche démontreune application pratique des principes de l'automatisationindustrielle, ensoulignant la relation synergique entre les automates programmables pour uneexécutionrobuste des commandes et les systèmes SCADA pour une gestion complète de la supervision afind'obtenir un fonctionnementefficace et précis du moteur à courant continu.

ملخص

تركز رسالة الماستر هذه على تنفيذو مراقبة نظام التحكم في محرك التيار المستمر والإشراف عليه باستخدام وحدة تحكم منطقية قابلة للبرمجة (PLC) وواجهة SCADA (التحكم الإشرافي والحصول على البيانات)، وتحديداً باستخدام خوارزمية التحكم التناسبي-التكاملي-المشتق .(PID) يشرح العمل بالتفصيل تكوين وبرمجة PLC لتنظيم سرعة المحرك بدقة، بما في ذلك قياس المدخلات والمخرجات التناظرية، ودمج حزمة وظائف .PID وفي الوقت نفسه، يوضح العمل تطوير واجهة SCADA التي توفر مراقبة في الوقت الفعلي لمعلمات المحرك، وتسمح بتعديل مكاسب ضبط PID (النسبية والتكاملية والمشتقة)، وتسهل التحكم التشغيلي من خلال أوامر مخصصة لبدء تشغيل النظام وإيقافه وإعادة ضبطه. يوضح هذا البحث تطبيقًا عمليًا لمبادئ الأتمتة الصناعية، ويسلط الضوء على العلاقة التآزرية بين أجهزة التحكم المنطقي القابلة للبرمجة لتنفيذ التحكم القوي وأنظمة SCADA للإدارة الإشرافية الشاملة في تحقيق تشغيل محرك تيار مستمر فعال ودقيق .

Table of Contents

List of figures

List of Tables

List of Abbreviations

API: Automate programmable industrial

SCADA: supervisory control and data acquisition

OB: Organization block

FB: Function Block

DB: Data Block

FC: Function Call

HMI : Human Machine Interface

GRAFCET : GRApheFonctionnel de Commande Etape Transition

CPU: Central Processing Unit

PLC: programmable logic controller

PID: Proportional-Integral-Derivative

DC: direct current

DCS: distributed control systems

VDU:video display unit

MTU: master terminal unit

RTU: Remote terminal unit

LANs: Local Area Networks

I/O: input/output

A/D: analog to digital

MPI: Multi-Point Interface

IEC: The International Electrotechnical Commission

SISO: Single-Input Single-Output

FOPLD: First-Order Plus Dead Time

General introduction

In the vast landscape of modern industrial automation and robotics, the precise and efficient control of DC motors stand as a foundational requirement. These versatile electromechanical devices are ubiquitous, driving applications from intricate robotic systems to large-scale manufacturing processes, owing to their robust performance, ease of control, and high torque capabilities. However, achieving optimal dynamic response, maintaining desired operational parameters, and mitigating disturbances necessitate sophisticated control strategies. Traditional open-loop control often falls short in meeting the stringent demands for accuracy and stability in varying load conditions or environmental fluctuations. This inherent limitation underscores the critical importance of closed-loop feedback systems, among which the Proportional-Integral-Derivative (PID) controller has emerged as a cornerstone due to its simplicity, robustness, and widespread applicability across diverse control systems.

The integration of PID control with programmable logic controllers (PLCs) provides a powerful and reliable platform for industrial automation. PLCs offer the computational power and real-time processing capabilities required for effective PID implementation, ensuring precise regulation of motor variables such as speed or position. Furthermore, the advent of Supervisory Control and Data Acquisition (SCADA) systems has revolutionized industrial operations by providing intuitive human-machine interfaces (HMIs) for centralized monitoring, data logging, and remote control. A well-designed SCADA interface not only enhances operational efficiency but also provides critical insights into system performance, facilitates fault diagnosis, and enables on-the-fly parameter adjustments.

This Master's thesis delves into the design, implementation, and experimental validation of a comprehensive control system for a DC motor utilizing a PID algorithm managed by a Siemens PLC, supervised through a dedicated SCADA interface. The objective is to demonstrate the effectiveness of this integrated approach in achieving robust and accurate motor control. This work will systematically explore the theoretical underpinnings of DC motor dynamics and PID control, detail the PLC programming methodology, and illustrate the development of a user-friendly SCADA HMI. Through this investigation, we aim to contribute to the understanding and practical application of advanced control techniques in industrial settings, ultimately showcasing a practical solution for precision motor control.

This thesis contains three chapters namely “Introduction to automation and control systems”, “Basics of PLCs” and “Application”. Chapter I lays the foundational knowledge, delving into the core concepts of industrial automation, its advantages, and various types, while also introducing the essential roles of Programmable Logic Controllers (PLCs) and Supervisory Control and Data Acquisition (SCADA) systems within modern industrial frameworks. Building upon this theoretical background, Chapter II details the methodical design and practical implementation of the DC motor control system, outlining the specific PLC programming for PID control, including data scaling and the configuration of the PID function block. Finally, Chapter III focuses on the development and functionalities of the SCADA interface, demonstrating its utility for real-time process supervision and operator interaction, and critically analyzes the control results obtained through varying PID parameter tunings, showcasing the system's dynamic performance.

Chapter I: introduction to automation and control systems

I) Chapter I: introduction to automation and control systems

I.1 Introduction

The term 'automation' was coined in the mid-1940s by the US automobile industry to describe the automatic handling of parts between production machines and their continuous processing within them. Advances in computers and control systems have broadened the scope of automation. Automation had already existed for many years on a small scale by the middle of the 20th century, using mechanical devices to automate the production of simple items. However, it was only with the addition of the computer that the concept became truly practical, as its flexibility allowed it to perform almost any task.[1]

I.2 Definition of automation

Automation can be defined as the process of carrying out a series of operations using specialized equipment and devices that perform and control manufacturing processes, with little or no human labor. Full automation is achieved through the use of various devices, sensors, actuators and techniques that can observe the manufacturing process, make decisions about necessary changes to the operation and control all aspects of it.
In industry, automation is the process by which various production operations are converted from manual to automated or mechanized processes.
It is a step beyond mechanization, where human operators are provided with machinery to assist them in their work. Industrial robotics are said to be the most visible part of automation. Modern automated processes are mostly computer-controlled, with sensors and actuators monitoring progress and controlling the sequences of events until the process is complete. These decisions ensure that the process is completed accurately and quickly.[1]

I.3 Advantages of automation

Automation technologies are driving rapid productivity growth in nearly all sectors of manufacturing. When considering the implementation of automation in the manufacturing sector, the primary association is often with the utilization of robots. The automotive industry was the early adopter of robotics, using these automated machines for material handling, processing operations, and assembly and inspection. The application of automation is a viable proposition in the context of manufacturing across the entire spectrum of types. [1]

The advantages of automation are manifold:

- higher productivity levels.
- lower production costs.
- Minimization of human fatigue.
- Less floor area required.
- Reduced maintenance requirements.
- The enhancement of working conditions for workers.
- Effective control over production process.
- The enhancement of product quality.
- a reduction in accidents has been observed, which consequently results in enhanced safety measures for workers.
- The production of uniform components is a key element of the process.

I.4. Types of automation

a) Fixed Automation (Hard Automation)

The term 'fixed automation' is used to denote the utilization of specialized equipment for the automation of a predetermined sequence of processing or assembly operations. It is generally associated with high production rates and it is relatively difficult to accommodate changes in the product design. This phenomenon is referred to as "hard automation".

The implementation of fixed automation is only logical when product designs are stable and product life cycles are extensive.[2]

Advantages

- Maximum efficiency.
- The unit cost is minimal.
- Automated material handling—fast and efficient movement of parts.
- A negligible amount of waste is generated during the production process.

Disadvantages

- Large initial investment.

- Inflexible in accommodating product variety.

b) Programmable Automation

In the context of programmable automation, the equipment is designed to accommodate specific product changes, and the processing or assembly operations can be modified by altering the control program.

In programmable automation, the process of reconfiguring a system for a new product is often time-consuming. This is due to the fact that it necessitates the reprogramming of the machines, as well as the procurement of new fixtures and tools. Examples of such technologies include numerically controlled machines and industrial robots,etc...[2]

Advantages

- Flexibility to deal with variations and changes in product.
- The cost of the unit is minimal when large quantities are produced.

Disadvantages

- The introduction of a new product necessitates a protracted period of preparation.
- the unit cost is elevated in comparison to that of fixed automation.

c) Flexible Automation (Soft Automation)

In the domain of flexible automation, the manufacturing equipment is designed to produce a wide range of products or components, with minimal time required for transition between different items. Consequently, a flexible manufacturing system can be utilized to produce diverse combinations of products in accordance with any specified schedule. A flexible automation system facilitates the expeditious incorporation of alterations to the product, which may be redesigned in response to evolving market conditions and consumer feedback. This adaptability enables the swift introduction of new product lines.[2]

Advantages

- Flexibility to deal with product design variations.
- Customized products.

Disadvantages

- Large initial investment.
- the unit cost is elevated in comparison to that of fixed or programmable automation.

I.5 Industrial Control Networks

I.5.1. Definition

The integration of automated units within industrial facilities necessitated the implementation of an efficient method for interconnecting these units, facilitating communication between them, and enabling the transfer of supervisory data to monitoring devices. This development subsequently gave rise to the integration of communication networks within industrial sectors.

It is evident that industrial networks are composed of three major control components, which are based on specialized functions. These components include PLC, SCADA, and distributed control systems (DCS).[3]

All three elements under discussion pertain to field instruments (i.e. sensors and actuators), smart field devices, supervisory control PCs, distributed I/O controllers and HMI. These devices are interconnected and communicate via a robust and efficacious communication network known as industrial networks. In such networks, data or control signals are transmitted either by wired or wireless media. The transmission of data via wired connections can be facilitated by various cable types, including twisted pair, coaxial cable, and fiber optics. Concurrently, radio waves are utilized for the transmission of data within industrial wireless networks.[3]

I.5.2. Communication Technologies for Industrial Networks

It is an irrefutable fact that traditional wired communication technologies have played a crucial role in the realm of industrial monitoring and control networks. Consequently, the communication was facilitated through the utilization of point-to-point wired systems. However, such systems entail

A substantial quantity of wiring was observed, which consequently gave rise to a considerable number of physical points of failure, including connectors and wire harnesses. This phenomenon ultimately led to the system's marked unreliability. The aforementioned drawbacks led to the substitution of point-to-point systems with advanced industrial communication technologies.[3]

Fieldbuses

In factories control systems are utilized to facilitate seamless communication between the controller and the actuator or transducer. This communication is typically achieved through the implementation of direct connections, thereby ensuring the efficient and precise operation of the system as a whole. As the system became more complex and networking technologies evolved, a transition from the analog system to the digital system was eventually affected. The development of fieldbuses was initiated to facilitate the integration of these digital components. In the course of recent decades, the industry has developed a plethora of fieldbus protocols (e.g., Foundation Fieldbus H1, ControlNet, PROFIBUS, CAN, etc.). In comparison with conventional point-to-point systems, fieldbuses offer enhanced reliability and visibility. They also facilitate capabilities such as distributed control, diagnostics, safety, and device interoperability.[3]

Industrial Ethernet

For a considerable number of years, the Controller Area Network (CAN) and CANopen, a higher layer protocol based on CAN, have been demonstrated to be the optimal solution for low-cost industrial embedded networking. However, the most evident limitations of these technologies are the restricted baud rate and the limited network length. The most significant challenge currently being posed to CANopen is that of industrial Ethernet technologies, which have emerged as the preferred low-cost industrial networking solution for businesses and enterprises over the course of several decades.

In recent years, Ethernet has been incorporated into industrial settings for the purposes of I/O and networking. It is gradually replacing multiple fieldbuses and proprietary networks, or working with them.[3]

The following are some of the benefits associated with the transition to Ethernet:

- Fewer smaller networks
- The cost is reduced.
- an increase in velocity is associated with this phenomenon.
- The establishment of a connection to the factory IT network.
- Connection to the Internet

I.6. Supervision systems (SCADA)

SCADA is the technology that enables a user to collect data from one or more distant facilities and to send limited control instructions to those facilities. The utilization of SCADA systems obviates the necessity for an operator to be assigned to stay at or to visit remote locations with regularity when those facilities are operating in accordance with standard parameters. The SCADA system encompasses not only the operator interface and the manipulation of application-related data, but also extends beyond these domains.[4]

I.6.1. Definition of SCADA systems

SCADA is an acronym formed from the initial letters of the terms 'supervisory control and data acquisition'. A SCADA system enables an operator situated in a location central to a widely distributed process to make set point changes, open or close valves or switches, monitor alarms and gather measurement information in distant process controllers. In instances where the spatial extent of the process is substantial, extending over hundreds or even thousands of kilometers from one end to the other, the merits of SCADA become evident. This is particularly pronounced in scenarios where routine visits are required to monitor facility operations, and SCADA plays a pivotal role in reducing the associated costs. The value of these benefits will increase in proportion to the remoteness of the facilities and the extent of the effort required.[4]

I.6.2. Component of SCADA systems

As illustrated in Figure(I.1), the SCADA system is comprised of several key components. The operator is situated at the core of the system, with access to the system being facilitated by an operator interface device, which is occasionally referred to as an "operator console". The operator console functions as the operator's window into the process. The apparatus comprises a video display unit (VDU) that displays real-time data about the process, and a keyboard for inputting the operator's commands or messages back to the process. It is acknowledged that alternative cursor-positioning devices, including but not limited to a trackball, a mouse, or a touch screen, may be utilized.[4]

The operator's input device is typically a computer keyboard; however, pointing devices such as touchscreens and mice are gaining popularity.

The operator engages with the master terminal unit (MTU), which functions as the system controller. It has been observed that within certain industrial contexts, the term "host computer" is employed in lieu of "MTU". The term 'server' is also employed by others.

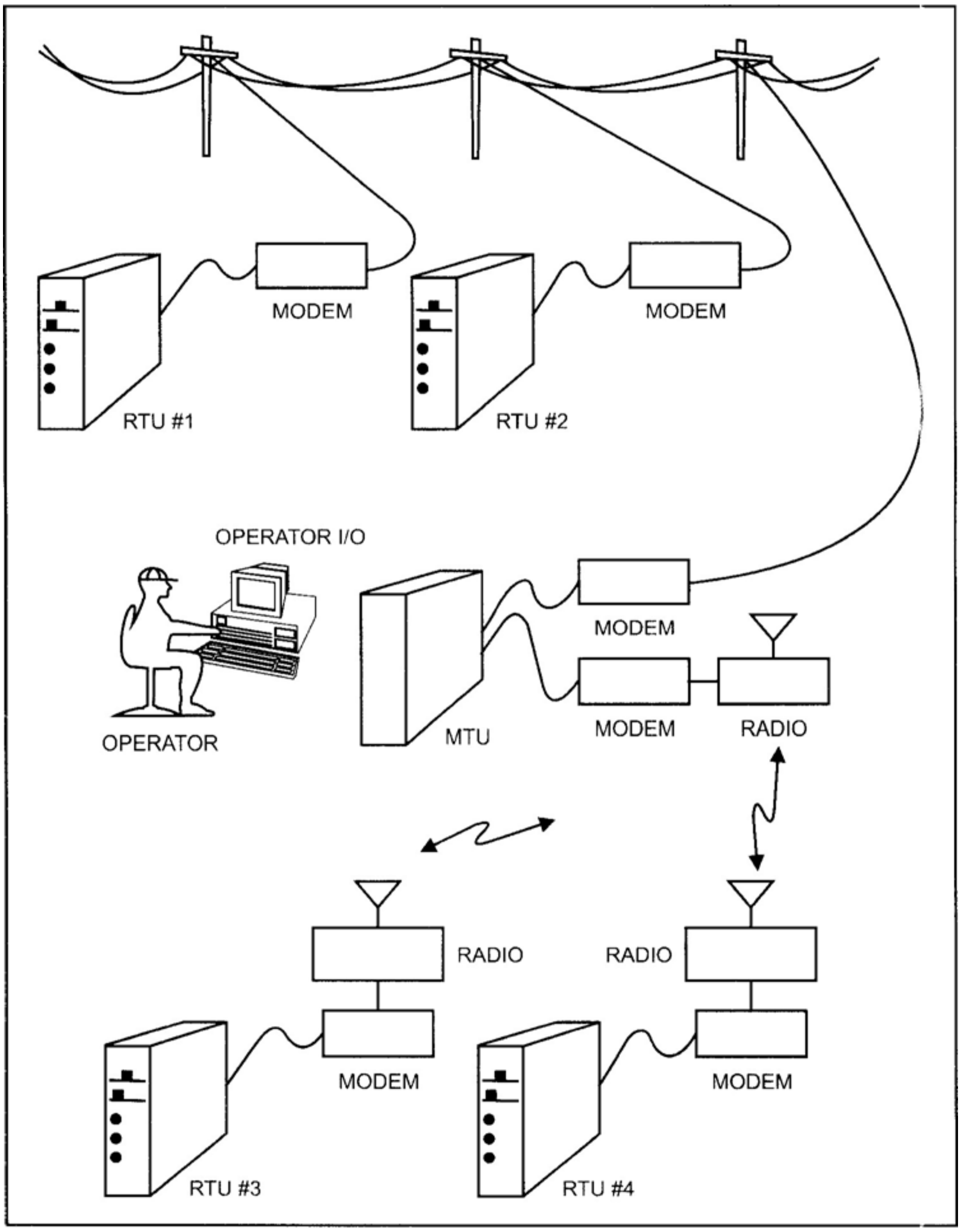


*Figure I.1**Major Components of a SCADA System***

In the context of contemporary SCADA systems, the MTU is invariably derived from a computer. The apparatus has the capacity to monitor and control the field, even in the absence of the operator.

It is imperative that communication is established between MTUs and RTUs that are not situated in the central location. The number of RTUs in a SCADA system can vary significantly, with the potential range extending from a minimal of one unit to a substantial number, potentially reaching several hundred.[4]

I.7 Architecture of Industrial Control Networks

Industrial control networks are typically configured in a hierarchical arrangement, as depicted in Figure(I.2). This configuration comprises three distinct levels: the informational level, the control level, and the device level. It is important to note that each level of the game has unique requirements, which in turn affect the network that is to be used for that particular level.

The device level comprises field devices, including sensors and actuators that are integral to processes and machinery. The objective of this level is to facilitate the transfer of information between the specified devices and the technical process elements, such as PLCs.[4]

In the contemporary era, fieldbus technology has emerged as the most sophisticated communication network utilized at the field level. It plays a pivotal role in enabling distributed control among a diverse range of smart field devices and controllers.[4]

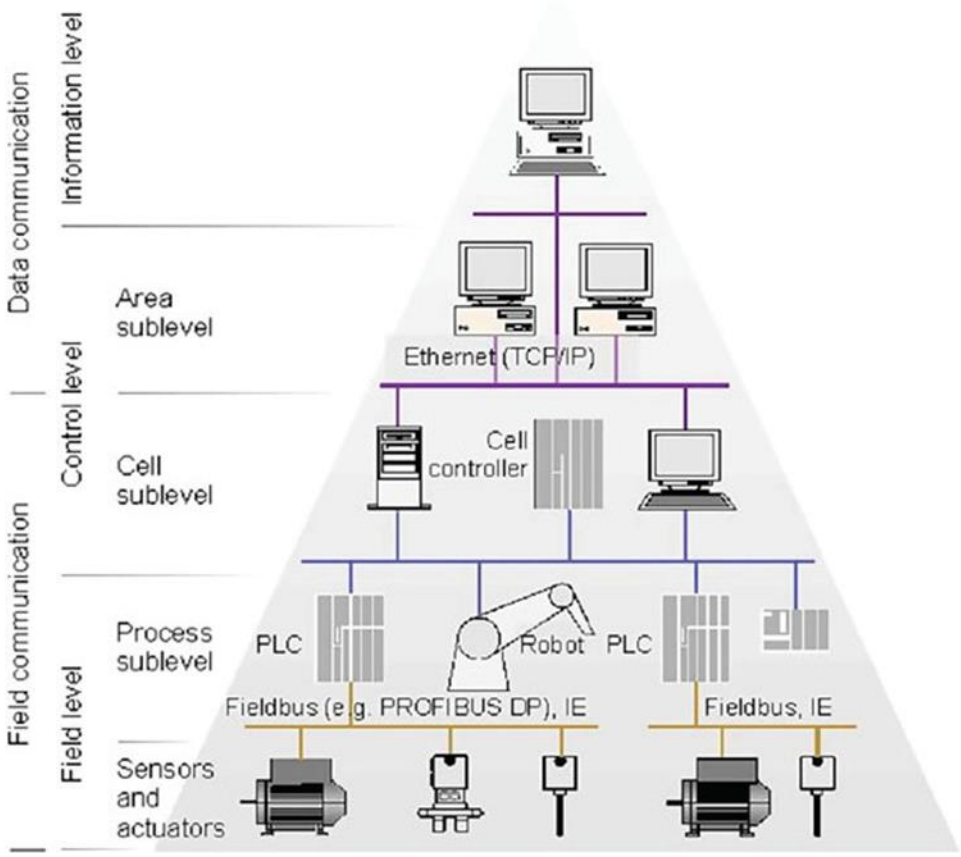


Figure I.2 ***Three-level architecture of industrial control networks***

The control level involves networking machines, work cells, and work areas. This is the level at which SCADA is implemented.

The tasks inherent to this level of the system include the configuration of automation devices, the loading of program data and process variables data, the adjustment of set variables, the supervision of control, the displaying of variables data on HMIs, historical archiving, and so forth. The control level of the network must achieve the predefined requirements, including the following: determinism, repeatability, a short response time, high-speed transmission, short data lengths, machine synchronization, and constant use of critical data.

Local Area Networks (LANs) are extensively utilized as communication networks at this level in order to attain the desired characteristics.

The informational level represents the pinnacle of the industrial system, serving as a nexus for information aggregation from the control level, which constitutes the lower echelon of the system. The focus of this study is on the management of large volumes of data that are not under constant use or deemed time critical. Large-scale networks are present at this level. It is evident that Ethernet WANs are frequently utilized as information-level networks for the purposes of factory planning and the facilitation of management information exchange.[4]

Chapter II:Basics of PLCs

II) Chapter II: basics of PLCs

II.1 Introduction to PLCs:

PLCs first emerged in the industry during the 1960s, exhibiting a markedly divergent form from contemporary iterations. These early controllers were constructed using logical components that merely substituted for the function of the auxiliary relays. It is evident that primitive PLCs exhibited a high degree of reliability when contrasted with classical electromechanical relays. A notable advantage of PLCs was their compact size, which made them a more efficient solution for automation applications. Subsequently, the evolution of these systems passed through multiple stages, with the most significant being the incorporation of digital components for timing, synchronization and counting, and the use of microprocessors. Contemporary PLCs can be categorized as either simple or complex, and are available in a variety of sizes. They are equipped with a range of extensions and interfaces that address the diverse requirements found at factory level, including the need to communicate with other devices and computers.PLCspossess the fundamental architecture of a personal computer, with two salient differences. The first issue pertains to the available hardware for the I/Os of the PLC, while the second concerns the microcontroller's operational mode and its interaction with the other electronic components of thePLC.[5]

In Figure (II.1) the basic parts of a PLC are presented: the CPU, the I/O modules, the RAM, and the power supply.

A programming device constitutes a peripheral device that is utilized exclusively during the programming stage of PLCs. Its removal is a subsequent process, as it is not a prerequisite for the operation of the system.

The role of a PLC in the modern context is to transform the hardwiring into flexible software, thereby serving as an expert tool for the industrial engineer to solve complex problems.[5]

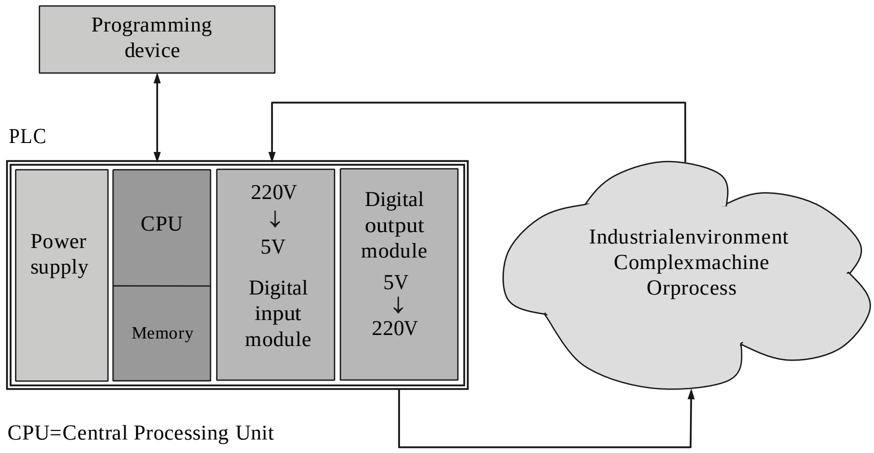


*Figure II.1****Internal structure of a programmable logic controller.***

II.2 Advantages of PLCs:

The widespread adoption of PLCs in industrial automation is attributable to their numerous significant advantages, which include:

Adequacy of the contacts. In the development of a conventional automation system, the evaluation of the efficiency of the auxiliary switching contacts of the power relays is a fundamental aspect of the design of the corresponding automation circuit. In circumstances where the number of auxiliary contacts required is substantial, the engineer may opt to incorporate these additional auxiliary contact blocks or establish a parallel connection with a secondary relay, thereby utilizing these contacts as auxiliary ones. However, in the case of a PLC, this is not an issue, since the adequacy of the contacts is unlimited. This is due to the fact that each internal memory bit location of a PLC can take the role of an auxiliary relay. Consequently, this could be utilized as many times as desired in a corresponding automation program.

Time saving. In the context of developing a programmable automation system equipped with a PLC, it is possible to concurrently undertake the programming of the system (that is to say, the design of the automation circuit) alongside the installation of the PLC and its connections to the input/output (I/O) devices. This is due to the fact that the program is written in the programming device. In the context of conventional automation, this approach is not feasible. The sequence of events necessitates the initial design of the automation circuit, followed by the construction of the industrial electrical enclosure in accordance with the designed automation circuit. This process is then followed by the installation and connections to the input and output devices.

Reduced need for space. It is evident that, due to their digital nature, PLCs exhibit a comparatively diminutive volume. Moreover, these devices are equipped with numerous timers, counters, and a substantial number of auxiliary relays, thereby resulting in a significantly reduced volume when compared to that of a conventional industrial automation enclosure that contains an equivalent quantity of auxiliary equipment.

Easy automation modification. The modification of a conventional automation circuit is a process that can only be achieved through the removal of cables, the addition of new equipment, and, in the most unfavorable circumstances, the suspension of the operation of the control system for some time. However, in the case of PLCs, all of the aforementioned modifications are effectively equivalent to the direct alteration of the corresponding program. Following the requisite amendments, the program can be downloaded onto the PLC online or with a pause in the overall operation that lasts for a few seconds.

Easy fault detection. The utilization of a PLC programming device facilitates direct observation of the status of the PLC's internal components and the execution of the loaded program. Furthermore, the ON or OFF state of all input and output devices can be subject to additional monitoring through the utilization of indicator lights. Furthermore, the potential for conducting "forced" (virtual) or simulated notional state transitions of an input device, for the purpose of observing the reaction of the automation system and the overall control logic, can be directly executed within the PLC. This capability is primarily attributable to the digital architecture of the PLC, which facilitates the execution of diverse testing procedures that are instrumental in troubleshooting.

Modern and working tools.The adoption of PLCs has had a profound impact on the working environment of engineers, leading to a transition from the use of cables, auxiliary relays, screwdrivers and other traditional tools to a more PC-like working environment. Engineers are required to utilize a keyboard or mouse in conjunction with a program, whether in Windows or an alternative environment. Their task is to simply print the automation program, rather than designing the automation circuit and applying their expertise in digital systems. The advent of these concepts has engendered a novel and contemporary operating environment, particularly when juxtaposed with its precursor in preceding decades.[5]

II.3 Modular Construction of a PLC:

It is evident that PLC manufacturing companies have adopted two basic types of PLC construction: the compact form and the modular shape. As demonstrated in Figure (II.2 a), PLCs of the first category are characterized by their solid structures, which are typically non-expandable, and possess a designated number of inputs and outputs. The integrated structure incorporates not only the I/Os, but also the power supply, microprocessor, interface with a programming device, STOP-RUN mode switch, LED indicators, and other components. PLCs of this category are primarily suited to small-scale automation applications, whereas medium and large PLCs adhere to the principles of modular construction. This configuration signifies that, in this particular instance, PLCs are comprised of autonomous, modular subunits that are mounted on a shared base. As demonstrated in Figure (II.2b), PLCs of this category are constituted by a base with particular dimensions, thus resulting in a specific spatial capacity for equipment such as a power supply module, microprocessor unit, and various I/O modules.[5]

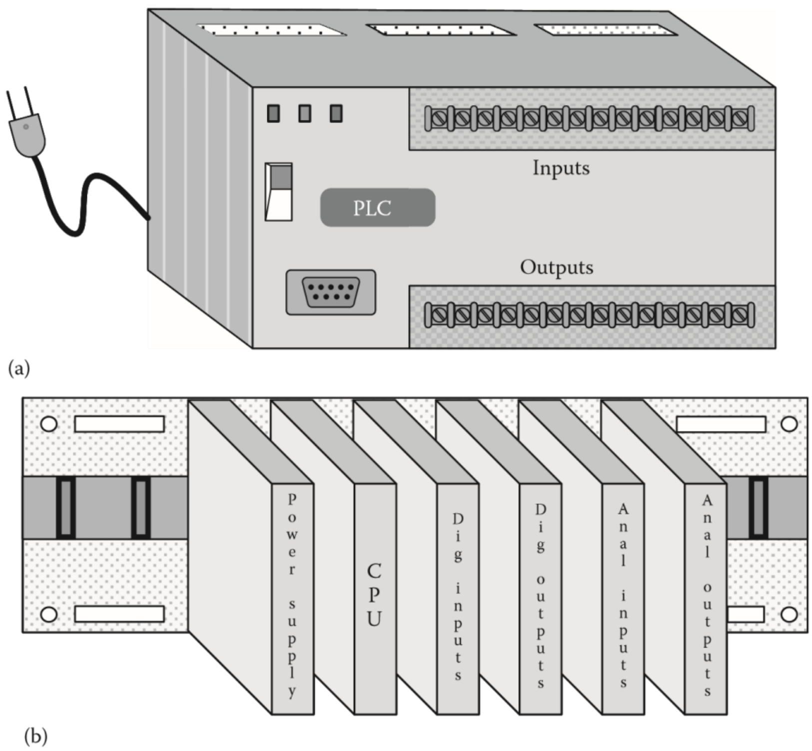


Figure II.2 ***Compact (a) and modular (b) structural form of a PLC.***

For the majority of PLCs, input/output (I/O) modules represent a common and fundamental component of equipment, often ranking second in importance only to the microprocessor unit. These I/O modules constitute the interface between the various I/O devices of the industrial automation system and the microprocessor that executes the desired program, thereby performing the control logic.[5]

II.3.1Digital Modules

II.3.1.1 Digital Input Modules

Digital inputs modules are engineered to receive discrete signals from the input devices and convert them appropriately for further processing, while ensuring electrical isolation before transferring them to the memory of the PLC. These signals are electrical voltages that can have different values.Consequently, the input module performs a conversion of the voltage level and a modulation of the input signals, so that it becomes compatible with the operating voltage (5 V DC) of the microprocessor and the other electronic components.[5]

The following devices are typically utilized as input devices:

- Limit switches
- Selector switches
- Buttons of all kinds
- Rotary switches
- Switching relay contacts
- Photoelectric switches
- Proximity switches

II.3.1.2 Digital Output Modules

The results of the logical processing of an automation program are stored in the output image table and subsequently transferred to the output devices via the digital output module. In general, it can be stated that the digital output module processes low-voltage signals from the microprocessor in a way that is the opposite of that of the digital input module.

Upon transfer of the logic signals from the output image table to the output module in a serial manner, the demultiplexing unit performs the task of decoding these signals and transmitting them to the individual outputs of the output module. It is imperative that the output module possesses the capacity for memory, in order to facilitate the retention and storage of the signals transmitted by the microprocessor during the scanning process. These signals are transmitted periodically, and the output module must be able to store them until the next transmission from the microprocessor. The maintenance of these signals is performed by an integrated circuit that is known as a "latch" or "hold".[5]

The following output devices are typically utilized:

- Power or auxiliary relays
- Solenoids of electro valves
- Indicating lights
- Devices for alarm indication
- Coils of electro-pneumatic valves
- Electronic devices that require an activation input or a change in their ON-OFF state

II.3.2 Analog Modules

II.3.2.1 Analog Input/Output Modules

Since the inception of PLCs, these devices have exclusively comprised units of discrete inputs/outputs, thereby incorporating solely devices of an ON-OFF nature. Consequently, these devices have been able to be connected to PLCs and, correspondingly, to be controlled. It is evident that, in consideration of the aforementioned restriction, that period was characterized by a limitation of the capacity of Programmable Logic Controllers (PLCs) to regulate discrete components of industrial processes, specifically those of a digital nature. It is an irrefutable fact that every analog control scheme necessitates the utilization of analog inputs and outputs. Consequently, and given the considerable advancements that have been made in the field of electronics, specific analog modules have been incorporated into PLCs to provide the capacity to interact with analog signals. This development has had a substantial impact on the domain of industrial control.[5]

II.3.2.2Analog Input Modules

An analog input module comprises all the necessary circuits for the connection of analog signals and their corresponding alteration into digital ones. The fundamental component of an analog input module is the analog to digital (A/D) converter, which transforms the analog input signals into a digital value, as illustrated in the simplified diagram of Figure (II.3 a). Subsequently, the digital value is transferred to the analog input's storage memory, where the central processing unit can access it based on the requirements stated in the execution program in the PLC. In instances where the analog input module comprises a substantial number of inputs, resulting in a consequent high cost, it becomes uneconomical for manufacturers to provide discrete modules for each analog signal present within an industrial application. It is for this reason that the analog input module is equipped with a signal-adapting block. This is a simple electrical circuit that is able to transform the input analog signal into a signal that is compatible with the A/D converter that is utilized. Therefore, in the event of an alteration to the type of analog input within such a module (for example, a change to the input sensor), it is sufficient to modify solely the signal adapting block.[5]

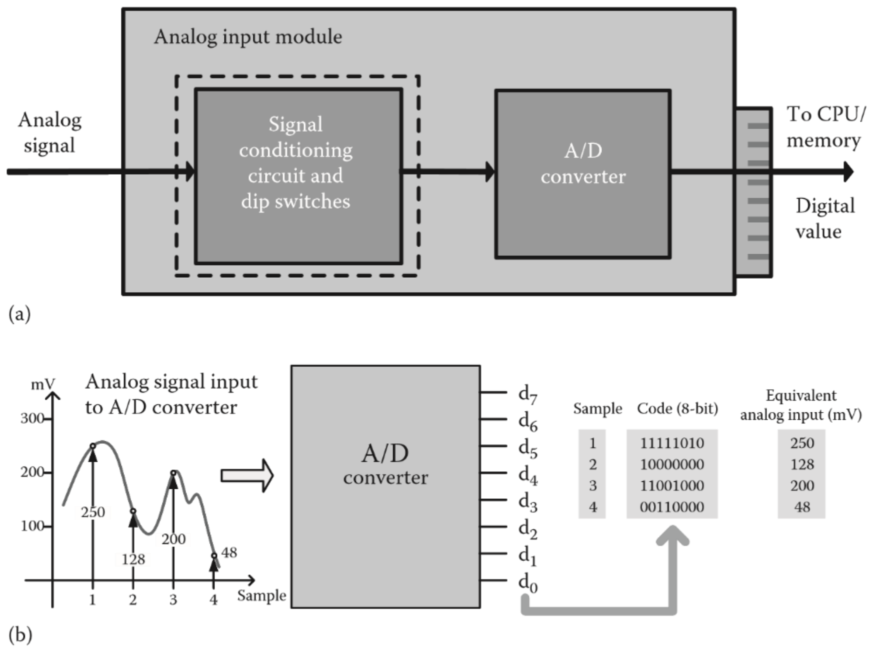


*Figure II.3**Simplified block diagram of an analog input module (a) and the basic principle of A/D conversion (b).***

It is evident that the analog signals supplied via the analog input module adhere to standard protocols, thus ensuring seamless interoperability with the output signals of the analog input devices (i.e., sensors or transducers). As illustrated in Table (II.1), the most prevalent analog input devices and the standard analog signals present in the analog input modules are delineated.[5]

*Table II.1****Usual Analog Devices and Standard Ranges of Analog Signals***

Analog Input Devices	***AnalogSignals***
Temperaturesensors	**4–20 mA**
Pressure sensors	**0–5 V DC**
Flow meters	**0–10 V DC**
Transducers of anykind	**± 10 V DC**
Electronic devices with analog output	**± 50 mV DC** **± 500 mV DC**

II.3.2.3 Accuracy in the Conversion of Analog Signals

Depending on the manufacturer and the model, the modules of analog inputs can exhibit varying levels of resolution. These levels are indicative of the accuracy of the conversion of analog signals. The resolution of an A/D converter is defined as the smallest discernible difference that the converter is capable of detecting in relation to the varying input signal. This concept is directly related to the number of bits comprising the binary word that represents the digital value of the analog signal. The number of bits is directly connected to the maximum number and the size of steps into the range of variation of the input signal. Consequently, the determination of the desired resolution for an A/D converter is a very important factor for the selection of the corresponding hardware. It is evident that an A/D converter that utilizes data words of 8 bits will have a resolution of 1 step in 256 steps of the full-scale variation of the input signal.[5]

II.3.2.4 Analog output Modules

The function of the analog output modules is to receive numerical data in digital form from the central processing unit of the PLC, transforming them into an analog voltage or current, in order to control the operation of an analog device. The digital values of the analog signals are typically words of 16 bits, created from the processing of program instructions. These are temporarily stored in the input/output memory of the PLC before being transmitted to the analog output modules for their conversion into analog signals, which are usually of 4–20 mA current or −10 V DC to +10 V DC. The process is illustrated in Figure (II.4). Notwithstanding the capacity of the digital domain to extend up to 16 bits, it is noteworthy that merely 11 of these bits are allocated for the digital representation of the analog signal, encompassing the useful converted data. The residual bits are utilized for the conveyance of auxiliary general information or timing signals.[5]

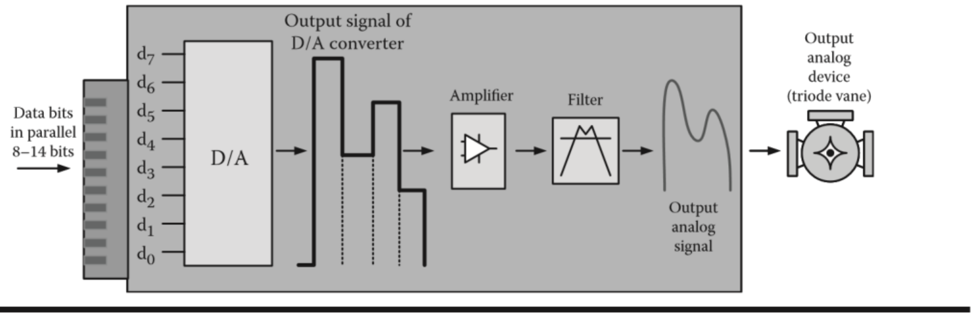


Figure II.4 ***Block diagram of operations inside an analog output module for converting a digital value signal to analog***

II.3.3 Special Purpose Input/Output Modules

II.3.3.1 Three Terms (PID) Control Modules

In industrial production process, continuous control of physical variables—such as hydraulic pressure, temperature, or motor speed—is essential. To support this, PLC units must be equipped with specific hardware. Most modern PLCs on the market include built-in software for analog control using a three-term PID (Proportional-Integral-Derivative) algorithm. However, because the PID algorithm involves complex mathematical computations, it can increase the PLC's scanning time significantly, potentially leading to inadequate control performance. To address this, manufacturers offer specialized PID control modules. These independent units come with their own microprocessors. They also feature dedicated analog input and output channels, which are connected directly to the controlled process. [5]

These modules are designed to perform real-time closed-loop control using PID algorithms. Instead of executing PID logic in the main CPU of the PLC, these modules handle it independently.[5]

Key Features:

- Built-in PID control logic
- Independent control loops (often multiple per module)
- Inputs: analog signals (e.g. from temperature sensors, flow transmitters)
- Outputs: analog (e.g. to actuators, valves)
- Real-time, deterministic response

Common Use Cases:

- Temperature control
- Pressure regulation
- Flow control in process industries
- Motor speed control

Manufacturers & Examples:

- Siemens: PID Compact or PID Control instructions within function modules
- Allen-Bradley: Enhanced PID modules (e.g. 1756-PID in ControlLogix)
- Honeywell: HC900 loop control modules

II.3.3.2Communication Modules

In circumstances where a PLC is required to communicate with another PLC, or with devices such as monitors, printers, or remote I/O modules, or indeed with another network of PLCs for the exchange of specific information, a communication module becomes indispensable. The simple, compact-type PLCs are generally equipped with a standardized communication port for establishing a connection to a personal computer (PC). A proportion of these PLCs are also equipped with an additional communication port, which facilitates connection to a specific industrial network. It is a general rule that communication modules are equipped with their dedicated microprocessor, which is referred to as a communication processor (CP). The communication processor is responsible for implementing the communication protocol for communicating with the external devices, either independently or networked, and with the central processing unit of the PLC. The final function, as outlined in the executed program, involves the transmission and reception of alphanumeric data to and from external, independent devices via the communication processor. [5]

II.3.4 Central Processing Unit

Figure (II.5)indicates the overall architecture of the main units of the PLC and their interconnections within the CPU in a rather simplified way.

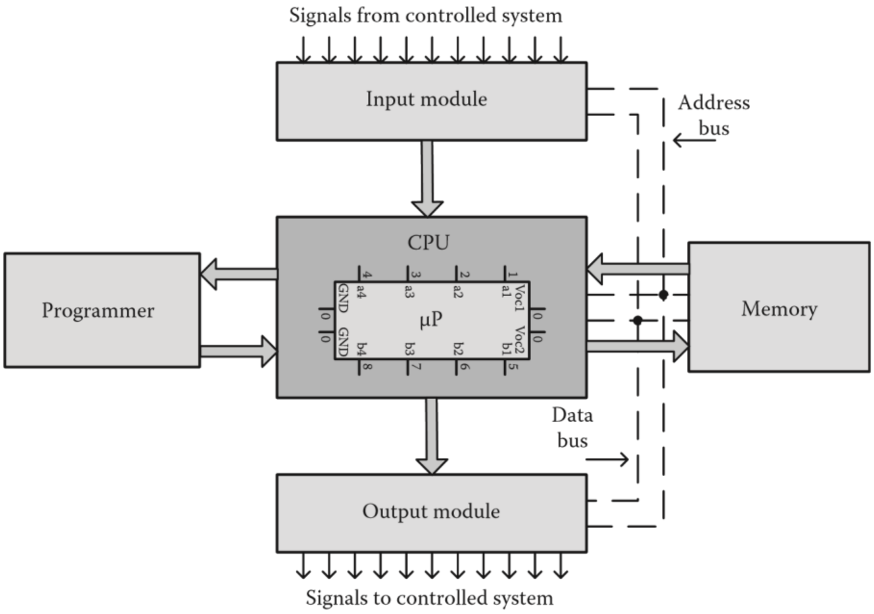


*Figure II.5**Operational interconnection between a CPU and the other units of a PLC.***

In more detail, the CPU of a standard PLC is capable of performing the following fundamental operations:

1. The operations which are concerned with the input and output of data. These operations facilitate communication between the PLC and the external environment, encompassing the scanning of inputs and the updating of outputs.

2. The following operations are logical and arithmetic in nature. It is evident that the operations under discussion contain all the logical functions of Boolean algebra (AND, OR, NAND, etc.) and basic arithmetic operations (such as addition, subtraction, multiplication, etc.). Within the same category, all specific operations such as timing, counting, and comparison can also be included.

3. The following operations are to be performed: reading, writing, and special handling. These operations encompass actions that are applied to the content of memory locations, potentially involving data or instructions.

4. The following section will address the topic of communication operations. These operations encompass particular functions that the PLC is designed to execute when communicating with external peripheral I/O modules, specific modules, and another PLC, amongst other scenarios. [5]

II.3.4.1 Memory Organization

The executable program, the user's program (the automation program), the states of the I/Os, and various data of non-permanent nature are stored in the memory of a PLC. The term "memory organization" is defined as the fragmentation of memory into various sectors, with each sector having a specific function and a corresponding size.

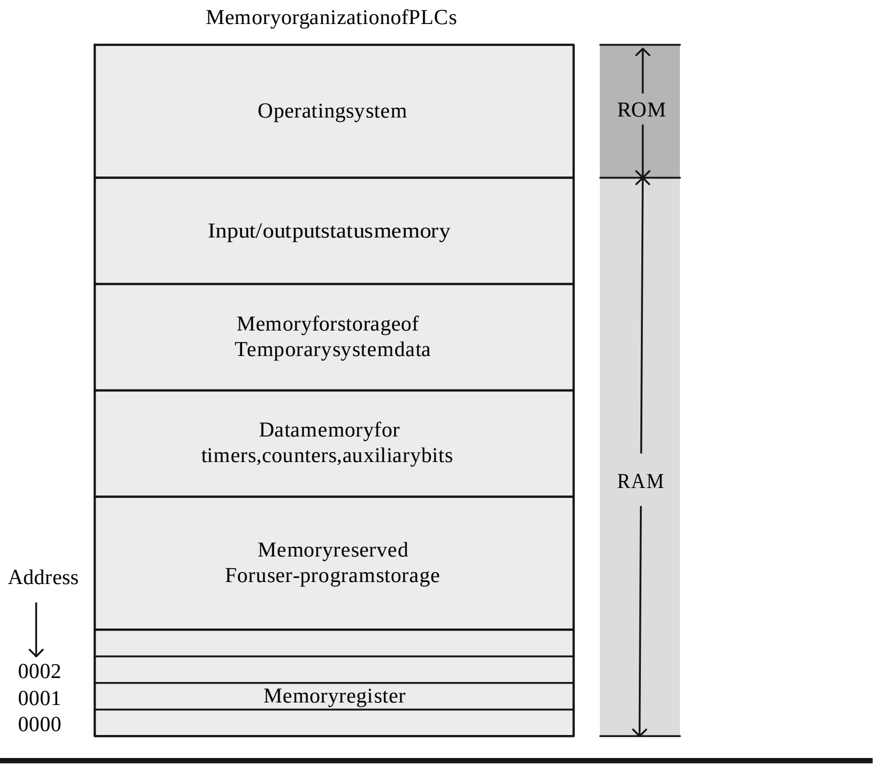


Figure II.6 ***Usual organization of memory in PLCs.***

Despite the heterogeneity in the organizational structure of memory across PLC manufacturers, a high degree of similarity in the general memory architecture of PLCs is evident. As illustrated in Figure (II.6), a standard fragmentation of a PLC memory is depicted across five sectors. The dimensions of each memory sector, and consequently the aggregate size of the memory, are contingent on the dimensions and characteristics of the PLC. To illustrate this point, consider the discrepancy in memory size between a small PLC controlling 128 input/outputs (I/Os) and a larger PLC controlling 1,024 I/Os. [5]

II.4 Local and Peripheral I/O System

In scenarios where a substantial number of I/Os are governed by a centralized PLC, it is anticipated that a proportion of the I/O devices will be situated at a considerable distance from the PLC. In such cases, a cost-effective and technically convenient solution for connecting the devices is to remove the I/O modules from the PLC, place them in close proximity to the I/O devices, and to remotely connect them with the PLC through a "communicative type" link, as illustrated in Figure (II.7 b). Alternatively, the remote I/O device should be connected directly to the respective I/O modules via multiple power lines, as illustrated in Figure (II.7a).

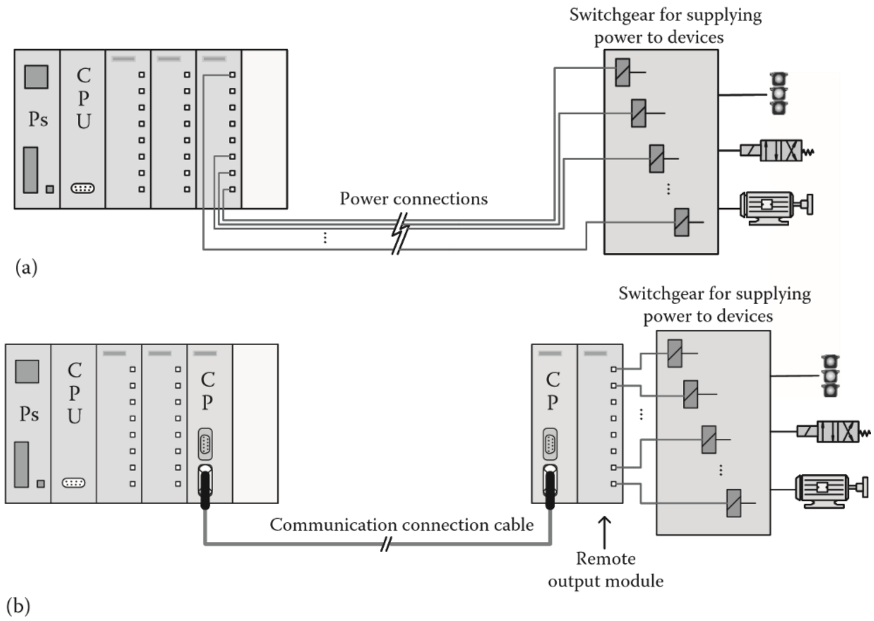


Figure II.7 ***Two possible ways for connecting remote I/O devices in a PLC: through power lines (a) and through communication processors (b).***

This approach, however, not only results in a substantial increase in installation costs, but also renders the connection more vulnerable to electromagnetic noise interference and functional failure. In instances where the PLC necessitates extension beyond a single rack to accommodate additional Input/Output (I/O) hardware, the implementation of a communicative type link between the Central Processing Unit (CPU) and the supplementary racks becomes a viable option. [5]

II.5 Programming of PLCs

A PLC is a digital device that is employed to monitor, process and generate data from industrial processes. In order to execute the desired functions, the PLC requires a software program that contains the necessary instructions. The program is comprised of two distinct components, each with its own specific objectives. The initial segment pertains to the operating system, which is responsible for all internal functions of the PLC. This system is developed by the manufacturer of the PLC and is not subject to modification by the user. The second component of the PLC's program pertains to the execution of logical and other functions as desired by the user. These functions must be expressed in a language comprehensible to the PLC. [5]

The programming of PLCs has evolved through various stages in parallel with the technological evolution of these devices.

PLC manufacturers have historically incorporated a variety of programming languages, encompassing both graphical and text-based forms, while concurrently maintaining the utilization of the ladder language. This approach is primarily driven by the recognition of several pivotal benefits [5], which include:

- It is a symbolic or graphical language with a representation that is very simple and closely resembles the structure of classic automation circuits. This is why it is not only familiar to all engineers dealing with industrial automation, but to technical electricians as well.
- The apparatus under consideration has been demonstrated to exhibit both short execution times for instructions and for program branches. This property is such that all branches are considered to be running in parallel mode, in a manner reminiscent of conventional automation circuits.
- The software in question has been developed to facilitate online programming, thereby enabling the user to make alterations to the program whilst the system is operational. Furthermore, the program can becompiled in real time.

II.5.1 The IEC 61131 Standard

The International Electrotechnical Commission (IEC) was established to address the chaotic situation regarding the incompatibility of programming languages forPLCs. The establishment of the open standard IEC 61131 was the initial response to this issue. The third part of this standard, the so-called IEC 61131-3, was issued in 2003 (second edition) and referred to the programming languages of PLCs. It is evident that all the major PLC manufacturers have now adopted this standard, with their products being developed in accordance with it. [5]

The IEC 61131-3 standard encompasses five distinct approaches to programming control systems, particularly in the context of PLCs.These languages can be seamlessly integrated to generate a composite code, or utilized in other types of control systems beyond PLCs, such as smart sensors, programmable drivers, process controllers, SCADA software, and related domains. The primary objective of the standard is to standardize existing programming languages, rather than to prevent the development of new languages. [5]

The five embedded languages for the PLCs are as follows:

a) Ladder Diagram, **LD, or LAD**—Graphic Language.

A program written in a Ladder Diagram language is similar to the classic wired automation circuits.

b) Function Block Diagram(FBD)—Graphic Language.

The function block diagram language comprises symbol blocks of functional operations, which implement a variety of functions ranging from the simple AND function from Boolean algebra to PID control.

The input and output variables of the PLC are connected to the inputs and outputs of the functional blocks, respectively, by interconnection lines or are simply declared before them. The output of one block may be connected to the input of another functional block.

c) Instruction List (IL)—Text-Based Language.

The instruction list is a low-level language that bears a resemblance to assembly language, yet it is situated at a higher level than LAD and FBD. IL is also referred to as the statement list (STL) or Boolean language. It is evident that the fundamental IL commands are merely the rudimentary operations of Boolean algebra (e.g. AND, OR, AND NOT, OR NOT, etc.) condensed.

d) Structured Text (ST)—Language Based on Text.

ST is a high-level language whose syntax bears a resemblance to that of Pascal. Individuals with prior experience in programming languages such as Basic or C will find the transition to ST programming relatively seamless. It is evident that, in general, ST is capable of supporting complex and embedded instructions.

e) Sequential Function Chart (SFC)

The SFC language is based on the GRAFCET (graphical function chart), a software tool of French origin which bears a strong resemblance to Petri networks. The GRAFCET language was established in 1977 by a French working group in order to provide a graphical tool for describing and defining sequential industrial processes that are controlled by logical controllers. This language was standardized according to the IEC 848 standard.

The IEC 61131-3 standard, incorporating the five programming languages previously referenced, has facilitated the harmonization of software between different PLC manufacturers, thereby contributing to the establishment of a uniform standard for programming. However, there are still necessary steps to be realized for achieving full compatibility, especially in the context of transferring programs from one PLC to another from a different manufacturer. [5]

II.6 On the Installation of PLCs

The installation and operation of PLCs must follow to specific rules and standards to minimize the risk of operational issues. This is especially important in severe industrial environments. Special care must be taken when installing PLCs in such conditions[5]. In addition to themanufacturer's guidelines, there are also other practices that contribute to the reliable performance of PLCs, which will can summarized as follow:

- Electrical Enclosure for the PLC Installation
- Electromagnetic Interference
- Grounding
- Electromagnetic Shielding of Cables
- Lightning Protection

II.7PID Control in PLCs

In the context ofPLCs, the PID controller is implemented in two distinct ways: In the first case, the PID algorithm is integrated in the programming software of the PLC (software controller) and is called as a subroutine, which means as an FB that has been developed by the manufacturer of the PLC. The PID algorithm is available in the relevant library, thus requiring only the declaration of variables and the parameters of the controlled process. The values of these parameters, which are necessary for the operation of the PID FB, are stored in a corresponding DB.

The physical magnitude that governs a PID FB is thus considered a variable derived from one of the analog inputs of the PLC, and consequently, the PID FB regulates the value of an analog output of the PLC. [5]

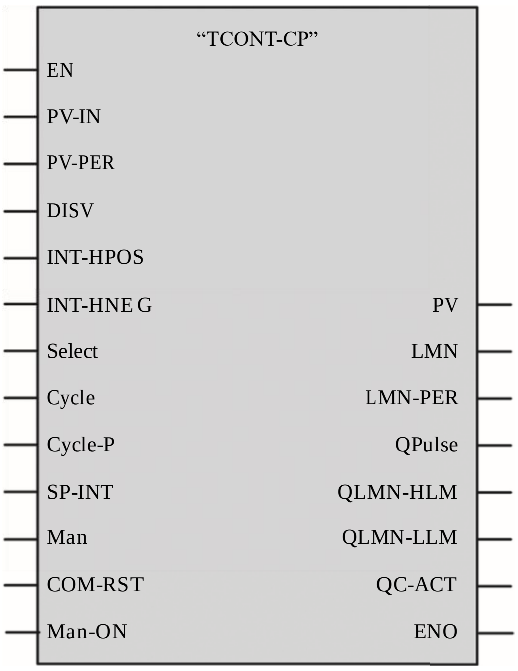


Figure II.8 ***Example of parameters to be assigned in a PID FB program (Siemens).***

For instance, Figure (II.8)presents the graphical representation of the PID FB's call in the FBD language of the Siemens Step7 software, encompassing over 20 parameters that can be designated in accordance with the intended function of the PID controller.

A secondary approach involves the provision of a discrete PID control module. This module is equipped with an autonomous microprocessor, as well as independent analogue inputs and outputs. It is evident that the PID control module fulfils the function of control, irrespective of the scan cycle of the CPU of the PLC. However, it is important to note that communication between the PID control module and the PLC occurs in order to exchange the values of the parameters and variables. Furthermore, it is evident that the independent module of the PID control system possesses a significantly more extensive array of features when compared with the built-in FB to the software algorithm. Moreover, the functionality of this module extends to the implementation of multiple PID control loops, contingent on the quantity of pairs of analog inputs and outputs. As illustrated in Figure (II.9), the block diagram depicts four PID control loops, which can be configured to implement a PID control module with four analog inputs and outputs. The bars between the controllers and the analog inputs and outputs represent the capacity of each PID controller to be integrated with any (one or more) from the analog inputs and outputs. It is evident that the four PID controllers have the capacity to function in an autonomous manner, or in a combined fashion, in a variety of different ways. [5]

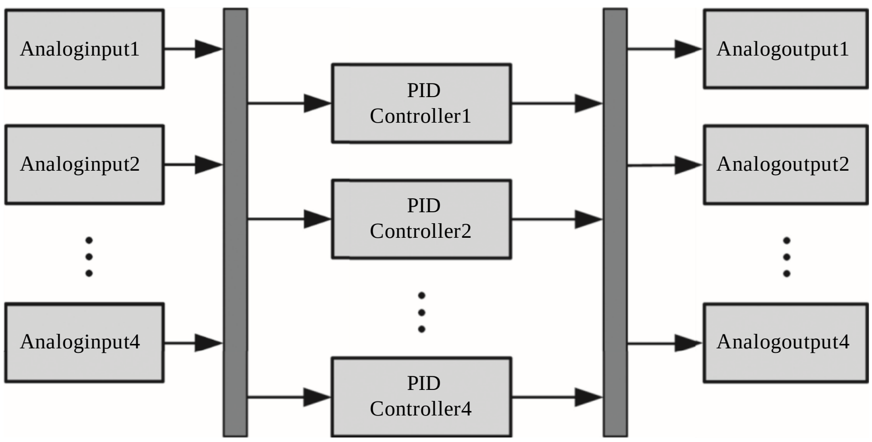


Figure II.9 ***An independent and separate module of PID control offers multiple control loops.***

ChapterIII: Application

III) ChapterIII: Application

III.1 Introduction

Automatic control systems and PLC are essential in modern industrial operations, offering significant benefits such as increased efficiency, precision, cost reduction, and safety. PLCs automate processes, ensuring consistent quality and eliminating human error, while also providing flexibility for reprogramming and scalability to adapt to changing needs. These systems enable real-time monitoring, diagnostics, and data logging, which help detect issues early, minimize downtime, and optimize performance. Additionally, PLCs improve safety by responding to hazardous conditions and enhance environmental control in systems like HVAC and water treatment. Overall, PLCs drive productivity, reduce operational costs, and contribute to sustainable and reliable industrial processes. The aim of this work is to achieve accurate, efficient, and automated motor operation with real-time monitoring and control. The PID controller ensures precise speed regulation by continuously correcting deviations from the desired setpoint, while the PLC executes control logic for consistent and reliable performance. SCADA provides a user-friendly interface for real-time supervision, data logging, fault detection, and remote control, enhancing operational safety and enabling predictive maintenance. Together, these technologies improve system efficiency, reduce human intervention, and extend the motor's lifespan in industrial applications.

This works aims at implementing and supervising a DC motor control system using a Siemens PLC (S7-300) and a SCADA interface. Specifically, it employs a PID (Proportional-Integral-Derivative) control algorithm. This chapter details the configuration and programming of the PLC for precise motor speed regulation, including the scaling of analog inputs and outputs, and the integration of the PID function block.

III.2 PID Control

III.2.1. Principals of PID controller

The ideal continuous time domain PID controller for a SISO process is expressed in the Laplace domain as follows:

$$U(s) = G_c(s)E(s) \quad \text{(III.1)}$$

With:

$$G_c(s) = k_c(1 + \frac{1}{T_i S} + T_d S) \qquad \text{(III.2)}$$

Andwith $K_c = proportional\ gain$, $T_i =$ integral time constant and $T_d =$ derivative time constant. if $T_i = \infty\ and\ T_d = 0$ (i.e. P control), then it is clear that the closed loop measured value y will always be less than the desired value r (forprocesses without an integrator term, as a positive error is necessary to keep the measured value constant, and less than the desired value). The introduction of integral action facilitates the achievement of equality between the measured value and the desired value, as a constant error produces an increasing controller output. The introduction of derivative action means that changes in the desired value may be anticipated, and thus an appropriate correction may be added priorto the actual change. Thus, in simplified terms, the PID controller allows contributions from present controller inputs, past controller inputs and future controller inputs.Manyvariations of the PID controller structure have been proposed (indeed, the PI controller structure is itself a subset of the PID controller structure).

1. The ideal PI controller structure:

$$G_c(s) = k_c\left(1 + \frac{1}{T_i S}\right) \qquad \text{(III.3)}$$

2. The ideal PID controller structure:

$$G_c(s) = k_c(1 + \frac{1}{T_i S} + T_d S) \qquad \text{(III.4)}$$

The ideal PID controller structure is used in the following products:

a. Allen Bradley PLC5 Product
b. Bailey FC19 PID algorithm
c. Honeywell TDC3000 Process Manager Type A

III.2.2. Process modeling

In control systems, various process models are used to approximate dynamic behavior. The Delay Model captures pure time delay without any dynamic response, representing processes that respond after a fixed time. The Delay Model is expressed as:

$$G_m(s) = k_m e^{-s\tau_m} \quad \text{(III.5)}$$

Where$k_m$is the static gain, andτ_m is the time delay, representing a system that reacts only after a fixed delay without any dynamics. The Delay Model with a Zero includes a zero in the transfer function and is given by:

$$G_m(s) = k_m(1 + Z_m s)e^{-s\tau_m} \quad \text{(III.6)}$$

WhereZ_m is the zero's time constant.

First order lag plus time delay (FOLPD) or FOPLD (First-Order Plus Dead Time) model is a common model used to approximate the behavior of many industrial processes in control systems. It is given by:

$$G_m(s) = \frac{k_m e^{-s\tau_m}}{1+sT_m} \quad \text{(III.7)}$$

Finally, the FOLPD Model with a Zero combines all elements capturing both the lag, time delay, and an initial zero dynamic, allowing for a more accurate approximation of complex process behaviors.It is given by:

$$G_m(s) = \frac{k_m(1+Z_m s)e^{-s\tau_m}}{1+sT_m} \quad \text{(III.8)}$$

III.2.3. Tuning rules

This section presents two well-known tuning methods, namely Ziegler–Nichols and Cohen–Coon, which are commonly used for determining suitable parameters for both PI and PID controllers. These methods provide systematic procedures to enhance control performance by adjusting the proportional, integral, and derivative gains based on the characteristics of the process. [6]

Folpd model

$$G_m(s) = \frac{\mathrm{K_m e^{-s\tau_m}}}{\mathrm{1+sT_m}} \quad \text{(III.9)}$$

Ideal PI controller

$$G_c(s) = K_c\left(1 + \frac{1}{T_i s}\right) \qquad \text{(III.10)}$$

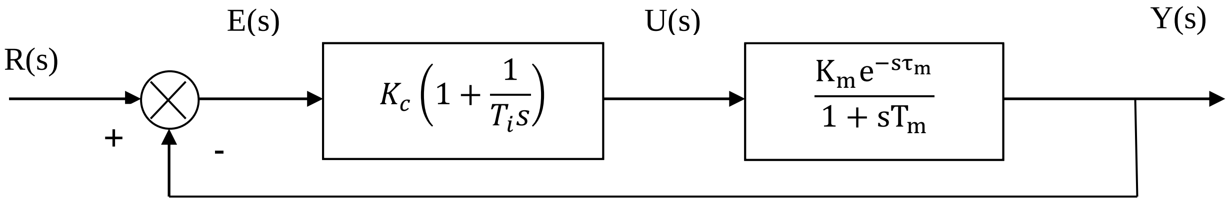


Table III.1PI controller tuning rules – FOLPD model

Rule	K_c	T_i	Comment
	Process reaction		
Ziegler and Nichols (1942). Model:Methode 2	$\frac{0.9T}{K_m \tau_m}$	3.33τ	Quarter decay ratio; $\tau_m/T_m \leq 1.$
Cohen and coon (1953). Model:Methode 2	$\frac{1}{k_m}\left(0.9\frac{T_m}{\tau_m} + 0.083\right)$	^1T_i	Quarter decay ratio $0 < \tau_m/T_m \leq 1.0$

$$^1T_i = T_m\left(\frac{3.33\left(\frac{\tau_m}{T_m}\right) + 0.31\left(\frac{\tau_m}{T_m}\right)^2}{1 + 2.22\left(\frac{\tau_m}{T_m}\right)}\right) \qquad \text{(III.11)}$$

Ideal PID controller

$$G_c(s) = K_c\left(1 + \frac{1}{T_i s} + T_d s\right) \qquad \text{(III.12)}$$

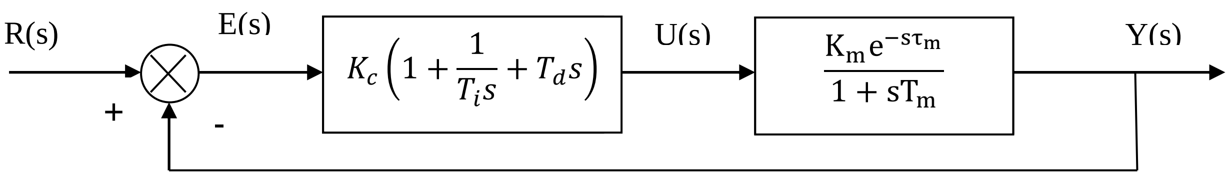


Table III.2PID controller tuning rules – FOLPD model

Rule	K_c	T_i	T_d	Comment
Zieglerand Nichols(1942). Model:Methode 2	$\frac{X_1 T_m}{K_m \tau_m}$	$2\tau_m$	$0.5\tau_m$	1.2≤ $X_1 \leq 2$; Quarter decay ratio.
Cohen and coon(1953). Model:Methode 2	^2K_c	T_i	$\frac{0.37\tau_m}{1+0.19(\frac{\tau_m}{T_m})}$	Quarter decay ratio $0.1 < \tau_m/T_m \leq 1$

$$^2K_c = \frac{1}{K_m}\left(1.35\ \frac{T_m}{\tau_m} + \ 0.25\right) \quad \text{(III.16)}$$

$$Ti \ = Tm \left[\frac{\left(2.5\left(\frac{\tau m}{Tm}\right) + 0.46\left(\frac{\tau m}{Tm}\right)^2\right)}{\left(1 + 0.61\ \left(\frac{\tau m}{Tm}\right)\right)}\right] \quad \text{(III.17)}$$

III.2.4.Examples

a) PI vs. PID

To compare the performance of PI and PID controllers, a first-order process with time delay was considered to model a simple DC motor system. To regulate the system's behavior, two classical control strategies were evaluated: PI and PID controllers. Both controllers were tuned with the same parameters (i.e. Kp and Ki). The PID controller incorporated a strong derivative term to enhance response speed and damping. Both controllers were implemented in unity feedback configurations, and their closed-loop performances were analyzed through step response simulations.

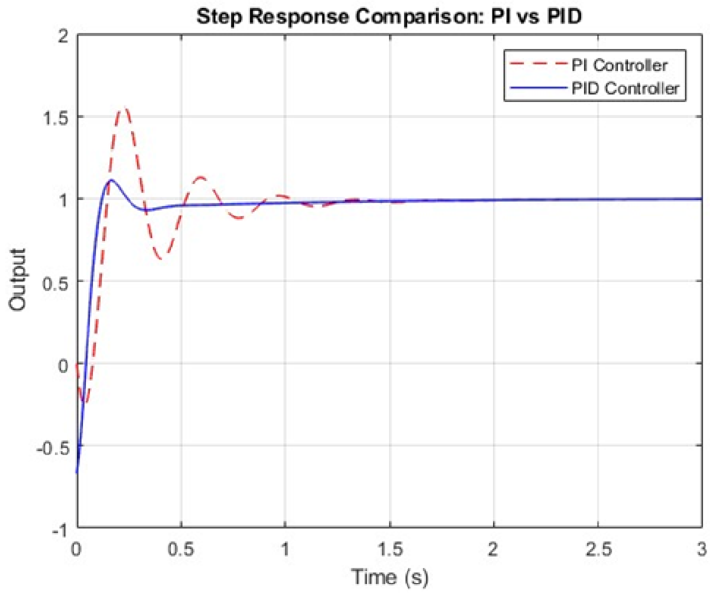


*Figure III.1****Comparison of the performance of PI and PID controllers on a first-order process with time delay***

Figure (III.1) illustrates the response of the process with both controllers. It showed that the PID controller offered a more responsive and stable performance compared to the PI controller. Specifically, the PID controller reduced the rise time and settling time, and significantly dampened the overshoot due to the added derivative action. In contrast, the PI controller, while capable of eliminating steady-state error, exhibited a slower response and higher overshoot. The step response plots confirmed that the PID controller was more effective in handling the time delay and improving transient performance. This comparison highlights the importance of derivative action in enhancing system stability and speed, especially in processes with inherent delays.

b) Effect of PID parameters

To analyze the effect PID parameters, a first-order process with time delay was also considered. To regulate the system's behavior, three PID controllers with different parameters were tested. Figure (III.2) compares the step response under three PID tuning strategies: aggressive, moderate, and conservative. The aggressive PID shows a fast rise time but significant overshoot, trading stability for speed, making it suitable for systems where rapid response is critical. The moderate PID strikes a balance, offering a reasonably quick rise with minimal overshoot and smoother settling, ideal for general-purpose control. In contrast, the conservative PID has the slowest response but no overshoot, prioritizing stability over speed, which is useful in precision applications where overshoot is unacceptable. The graph visually demonstrates the classic control trade-off: faster response leads to higher overshoot, while slower tuning ensures stability.

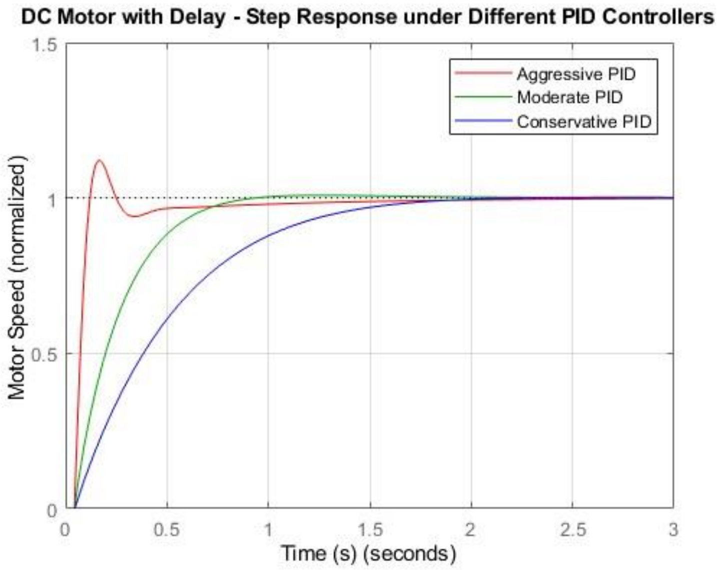


Figure III.2 ***Comparison of the performance PID controller, with different parameters, on a first-order process with time delay***

Figure (III.2) compares the step response of a DC motor under three PID tuning strategies: aggressive, moderate, and conservative. The aggressive PID shows a fast rise time but significant overshoot, trading stability for speed, making it suitable for systems where rapid response is critical. The moderate PID strikes a balance, offering a reasonably quick rise with minimal overshoot and smoother settling, ideal for general-purpose control. In contrast, the conservative PID has the slowest response but no overshoot, prioritizing stability over speed, which is useful in precision applications where overshoot is unacceptable. The graph visually demonstrates the classic control trade-off: faster response leads to higher overshoot, while slower tuning ensures stability.

III.3 Description of the DC Motor Control System:

The experimental setup is built around a DC motor (see Figure III.3), which serves as the primary actuator in the control system. The motor is powered by a regulated DC power supply that provides the necessary voltage and current to drive the system. A power amplifier is placed between the power supply and the motor to modulate the input voltage according to the control signal generated by the PID controller. The motor's speed is continuously monitored using a speed sensor, which converts the mechanical rotation into an electrical signal representing the motor's actual speed. This feedback signal is routed back to the PID controller (or the PLC) for closed-loop regulation.

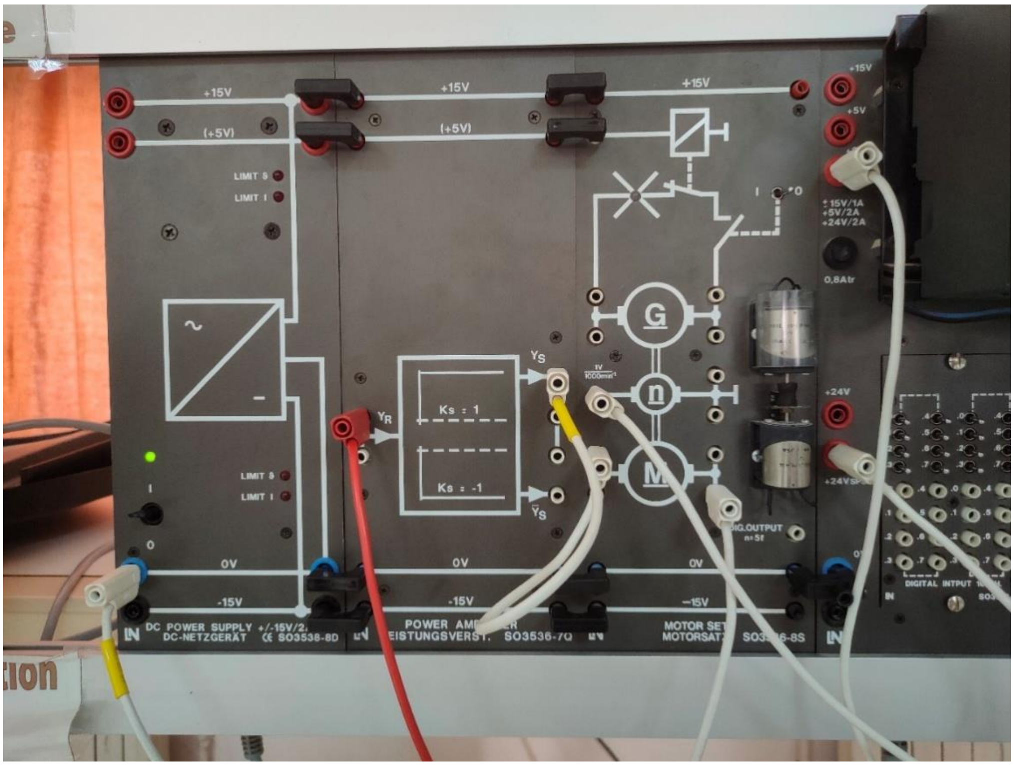


*Figure III.3**Architecture of the DC Motor Control Process***

Additionally, the setup includes a DC generator mechanically coupled to the motor shaft, simulating a load and allowing energy conversion measurements. The generator's output can also be monitored as part of the system's response evaluation. Signal acquisition and exchange between these components are managed through analog connected to the Siemens S7-300 PLC, which handles real-time data processing and execution of the PID control logic.

III.4. Speed control and Supervision of the Dc motor

III.4.1 Configuration of the PLC:

III.4.1.1 Presentation of step7:

The fundamental programming and configuration tool for SIEMENS automation is called STEP 7. It is a component of the SIMATIC software market[7]. The following features are available in STEP 7 for automating an industrial process:

- Hardware configuration and parameterization
- Communication parameterization
- Programming
- Testing,commissioning and maintenance
- Documentation, archiving
- Diagnostic and operationalfunctions
- A user interface diagnosis that meets modern ergonomic knowledge with very easy learning

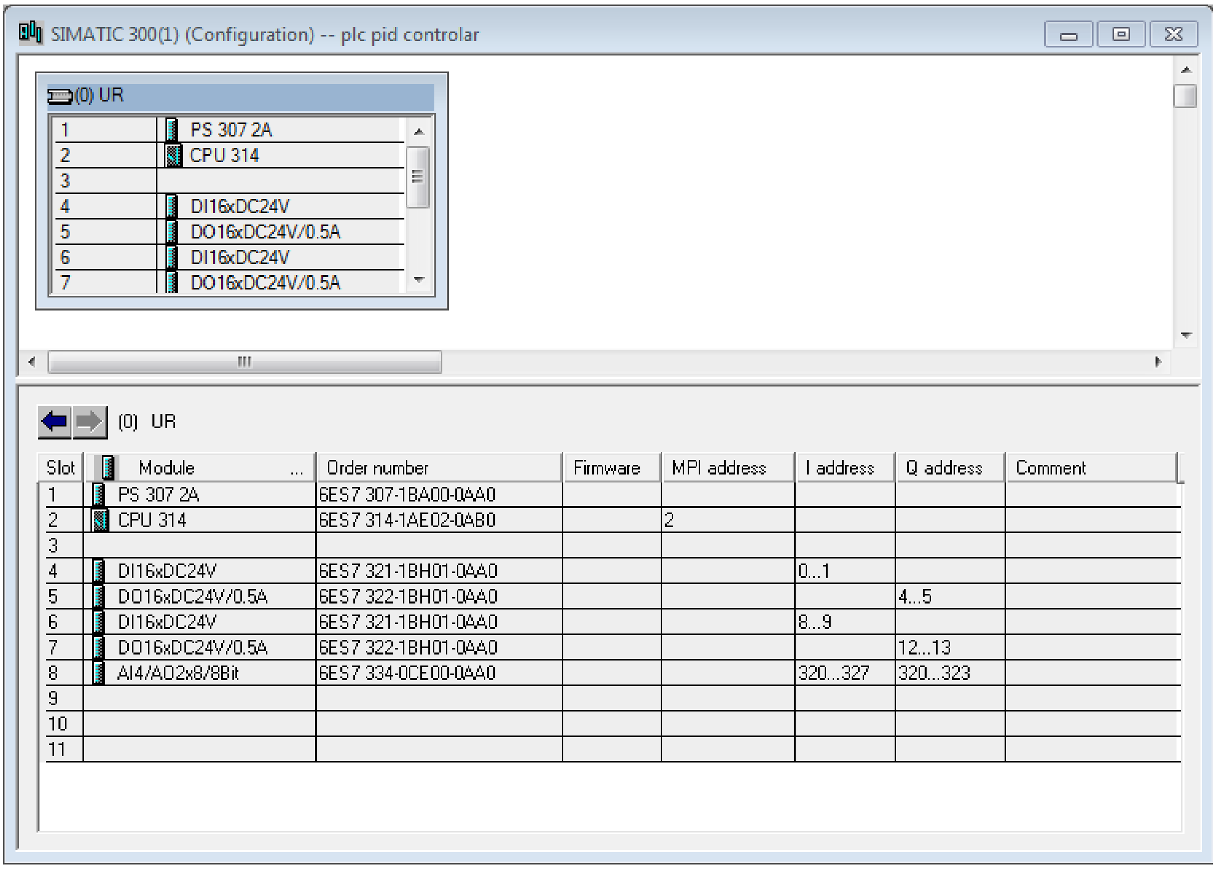


*Figure III.4**Hardware configuration of the CPU used***

III.4.1.2 Hardware:

This stage involves configuring the hardware: the station in use contains an SM 334 AI4/A02x8/8bit analog input/output module and a CPU 314. The setup of the PLC in use issummarized in figure(III.4). [7]

III.4.2 Programming of the PLC:

First,the PC is used to create the necessary program with the STEP7 software. Specifically, the S7 program must be developed within a project that includes the relevant station. Once the PC is connected to the PLC's MPI interface, the program can be loaded into the programmable logic controller's memory using the loading function.

This Siemens PLC program employs a classic feedback control loop, specifically a PID controller (FB41), to regulate a DC motor's state, its angular velocity. The process begins with an analog input (PIW320) acquiring the motor's actual state, which is then linearly scaled (FC105) into a "Process Variable" (PV). This PV is continuously compared against a desired "Setpoint" (SP_INT) to generate an error signal. The PID algorithm then calculates a corrective output based on three terms: the proportional term, which is directly proportional to the current error; the integral term, which accumulates past errors to eliminate steady-state offset; and the derivative term, which anticipates future errors by responding to the rate of change of the error. The resulting control effort (LMN_PER) is subsequently unscaled (FC106) and converted into an analog output signal (PQW320) to drive the DC motor's power electronics, thereby adjusting its operation to minimize the error and maintain the desired setpoint. This entire control cycle is executed periodically, typically within a cyclic interrupt block (OB35), ensuring real-time regulation. [7]

III.4.3 Supervision using SCADA

In this work, we developed and implemented a SCADA interface, as visually presented in Figures (III.5 a, b, c), which serves as the Human-Machine Interface (HMI) for real-time supervision and control of a DC motor regulated by a PID algorithm. This graphical environment empowers operators with direct control actions via dedicated buttons: a green button to initiate motor operation, a red button to affect an immediate motor stop, and a yellow button specifically for a complete reset of the PID controller, which typically clears accumulated integral error and resets internal states for re-initialization.

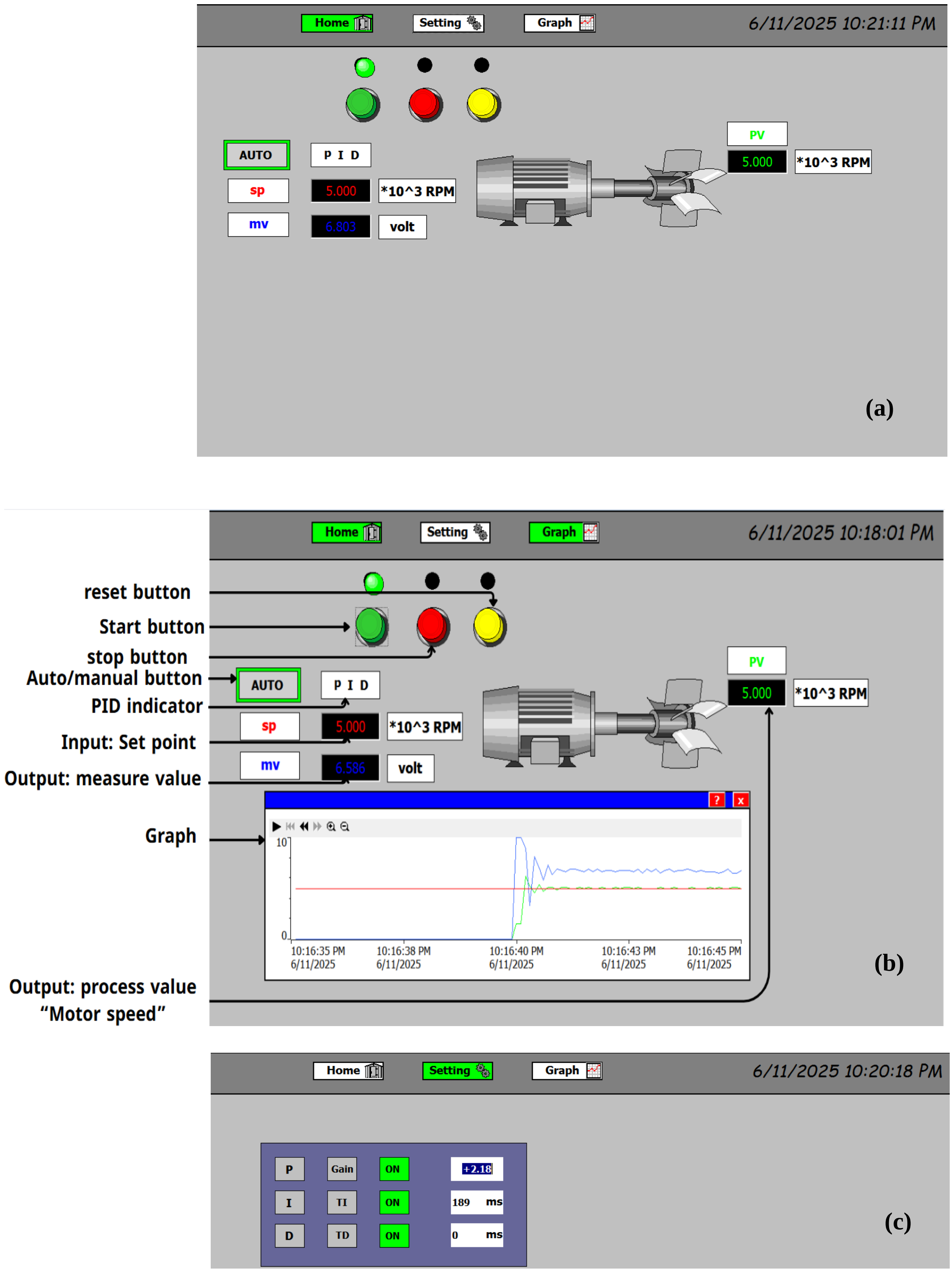


*Figure III.5**SCADA interface***

(a) Home page without graphs (b) Home page with graphs (c) Setting page

Crucially for process optimization, the interface also provides a dedicated "Setting" screen where the PID controller's tuning parameters—Proportional Gain (Gain), Integral Time (TI), and Derivative Time (TD) are clearly displayed and can be adjusted. This combination of discrete control commands and real-time parameter tuning, alongside typical displays for the motor's actual process variable (e.g., speed), enables operators to precisely manage the motor's behavior, ensuring stable, accurate, and responsive performance in accordance with dynamic operational demands.

III.4.4 Control results

In this work, we applied three methods for tuning the PID parameters: the Ziegler–Nichols method, the Cohen–Coon method, and the MATLAB pidtune function. Before applying these methods to the actual motor, we first tested them in MATLAB on a simulated motor model with the same parameters as the real one, for both PI and PID control.

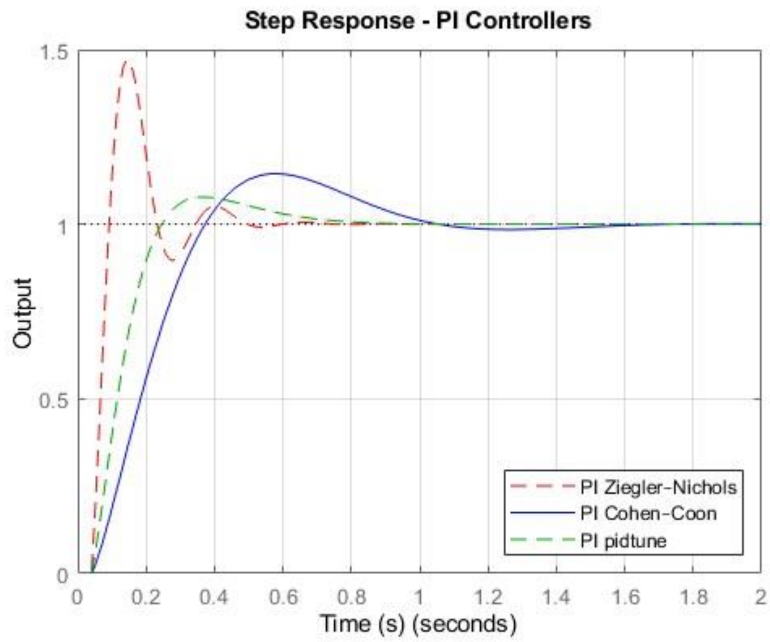


Figure III.6 ***Temporal response of the motor model, configured with the same parameters as the real motor, using PI controllers tuned by the three methods***

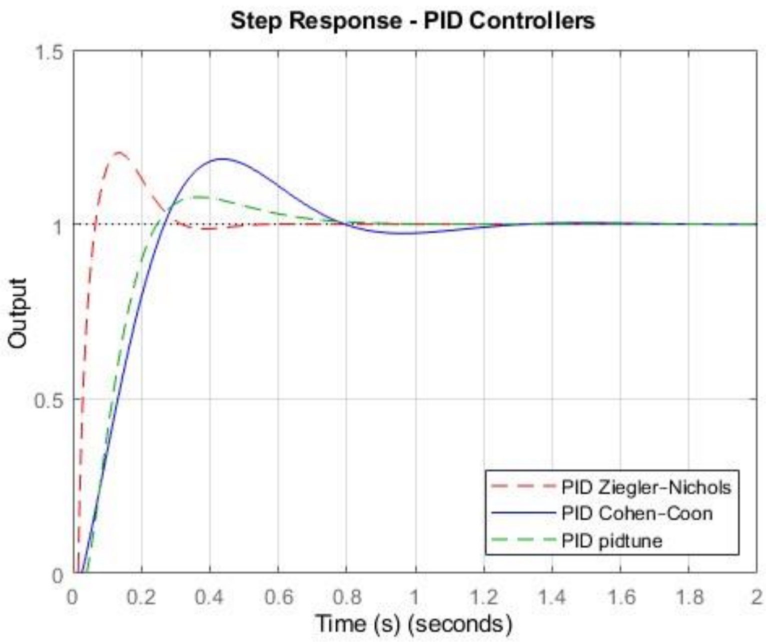


Figure III.7 ***Temporal response of the motor model, configured with the same parameters as the real motor, using PID controllers tuned by the three methods***

Figures (III.6) and (III.7) compare the step responses of PI and PID controllers tuned using the three selected methods. In both cases, the Ziegler–Nichols method yields a faster response but with more visible oscillations. The Cohen–Coon method provides a smoother response, offering a better trade-off between speed and stability. The third method, MATLAB's pidtune function, represents an optimized tuning strategy that potentially achieves a balanced compromise between responsiveness, overshoot, and settling time.

Afterward, we conducted the experiment on the real motor using the PID block integrated in the Siemens PLC S7-300. A SCADA interface was employed to enter the setpoint and controller parameters, and to monitor the setpoint, control signal, and process variable (i.e., motor speed). Figures (III.8), (III.9), and (III.10) illustrate the step responses obtained with the three tuning methods. The PID response tuned with the Cohen–Coon method shows a transient behavior where the system quickly stabilizes after an initial disturbance, reaching steady state with minimal oscillation—indicating a well-balanced trade-off between speed and stability. In contrast, the Ziegler–Nichols-tuned PID response is highly oscillatory, with significant overshoot and undershoot around the setpoint and poor settling. This behavior is typical of an aggressively tuned controller with insufficient damping. Among the three methods, the best results were obtained using MATLAB's pidtune function. The corresponding graph shows a rapid and stable response, with both low overshoot and quick settling. Compared to the Cohen–Coon method, pidtune provides slightly better overall performance, achieving a more desirable balance between responsiveness and stability. These results highlight the advantage of using optimization-based tuning tools like pidtune for real-world control applications.

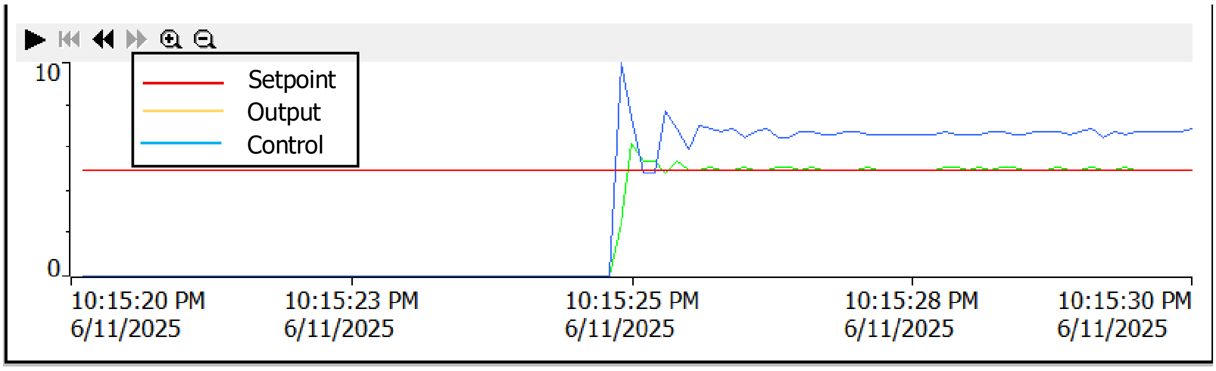


Figure III.8 ***Step response of the real motor with PID parameters tuned using the pidtune function***

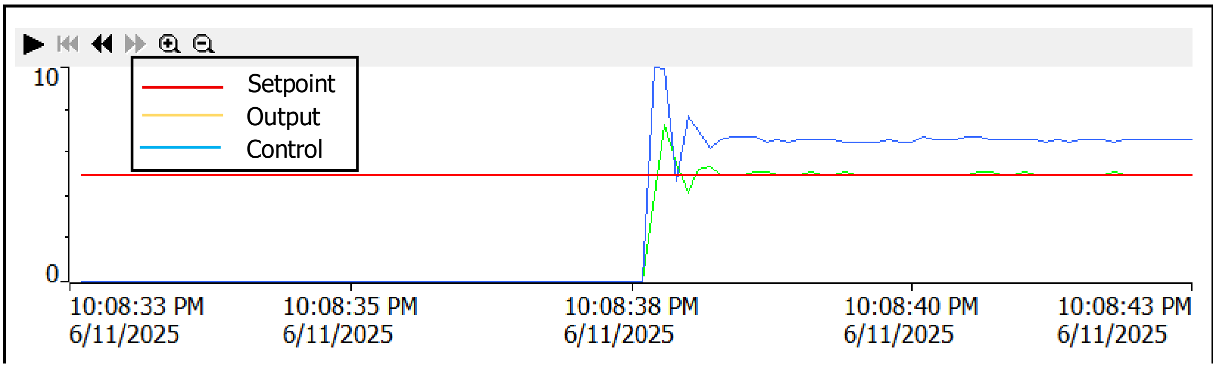


Figure III.9 ***Step response of the real motor with PID parameters tuned using the Cohen coon method***

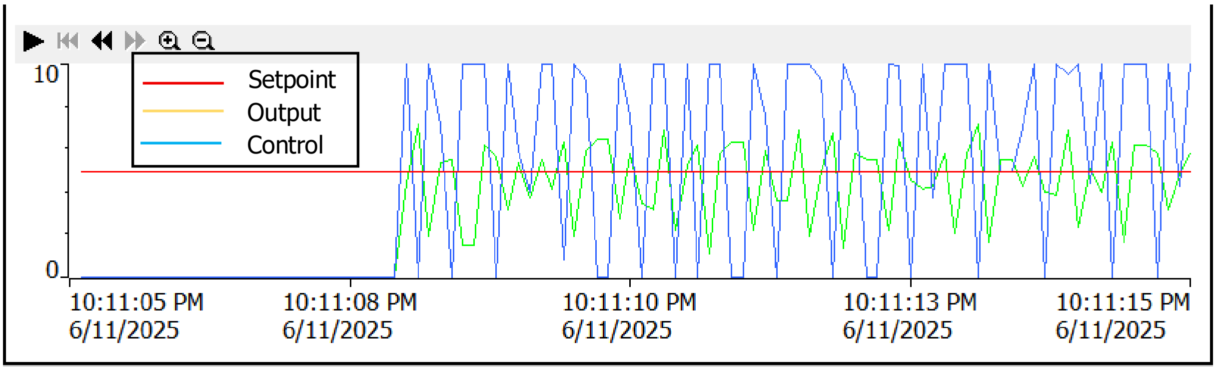


Figure III.10 ***Step response of the real motor with PID parameters tuned using the Ziegler–Nichols method***

In addition, we observed that the motor's transient response varied with each run. For example, Figure (III.11) illustrates three different responses obtained using the same parameters tuned with the Cohen–Coon method. This highlights the challenges of achieving consistent performance under real-world conditions. The observed variations underscore the influence of unmodeled dynamics, external disturbances, and the limitations of the implemented tools. These factors often cause discrepancies between theoretical predictions and the actual behavior of physical systems.

Figure III.11 ***Examples of step responses from three runs of the real motor using the same PID parameters tuned with the Cohen–Coon method***

III.5 Conclusion

We observe that the experimental results obtained from the real motor closely align with the MATLAB simulation outcomes, providing valuable comparative insights into control system performance. For example, the simulated response of the PID controller tuned using the Ziegler–Nichols method—marked by significant overshoot and sustained oscillations—closely mirrors the erratic and oscillatory behavior recorded in the real-world Ziegler–Nichols implementation. This consistency suggests that the Ziegler–Nichols method tends to produce underdamped responses, regardless of whether the system is simulated or physical.

In contrast, the simulated responses obtained using the Cohen–Coon and pidtune methods exhibit rapid settling and minimal oscillations following an initial disturbance. These characteristics strongly correlate with the stable and well-damped behaviors observed in their respective real-world counterparts, indicating more robust and reliable tuning. While slight differences exist between the simulated and actual responses—likely due to real-world complexities such as unmodeled dynamics, system nonlinearities, or external disturbances—the overall trends remain consistent, reinforcing the predictive value of simulation in controller design.

General conclusion

This Master's thesis successfully presented the design, implementation, and experimental validation of a robust and efficient control system for a DC motor, leveraging the synergistic capabilities of a PID controller, a SiemensPLC, and a SCADA interface. The theoretical foundation laid out the essential principles of DC motor operation and the mathematical framework of PID control, providing the necessary context for the system's development.

The practical realization involved meticulous PLC programming, where the PID algorithm was precisely configured to regulate the DC motor's speed (or position) by continuously comparing the process variable with a defined setpoint. The integration of the PID functionality within the PLC demonstrated its capacity to deliver real-time, accurate control, effectively minimizing errors and ensuring stable system response. Furthermore, the development of a user-friendly SCADA interface proved instrumental in providing comprehensive real-time monitoring, intuitive control, and remote parameter adjustment capabilities. The ability to initiate, stop, and reset the motor, along with the flexibility to fine-tune the PID gains (Kp, Ki, Kd) directly from the HMI, significantly enhanced operational efficiency and facilitated effective system optimization.

In a real-world environment, achieving identical performance and results can be sometimes challenging, forexample, the presence of a defect in the tools used like the DC motor or the work station.

In conclusion, this work has successfully demonstrated a practical and effective solution for precision DC motor control in an industrial context. The combined strength of PID control for its proven robustness, PLC for its reliability and real-time execution, and SCADA for its supervisory and data visualization features, culminates in a powerful automation architecture. This project not only reinforces fundamental concepts in control engineering and industrial automation but also offers a tangible framework that can be adapted and expanded for more complex control applications, contributing to the advancement of automated systems in various engineering domains.

References:

[1] Boyer, S. A. (2009). SCADA: supervisory control and data acquisition. International Society of Automation.

[2] Kim, D. S., &Tran-Dang, H. (2019). Industrial sensors and controls in communication networks. In Computer Communications and Networks. Cham, Switzerland: Springer.

[3] Westcott, J. R. (2023). Industrial automation and robotics.

[4] McCrady, S. G. (2013). Designing SCADA application software: a practical approach. Elsevier.

[5] Manesis, S., & Nikolakopoulos, G. (2018). Introduction to industrial automation. CRC Press.

[6] O'dwyer, A. (2009). Handbook of PI and PID controller tuning rules. World Scientific.

[7] STEP 7V5.6 SP1 Programming Software for SIMATIC S7/C7–Siemens

Annex

OB35 - <offline>

"CYC_INT5" Cyclic Interrupt 5

Name: **Family:**

Author: **Version:** 0.1

Block version: 2

Time stamp Code: 01/01/2009 09:17:47 PM

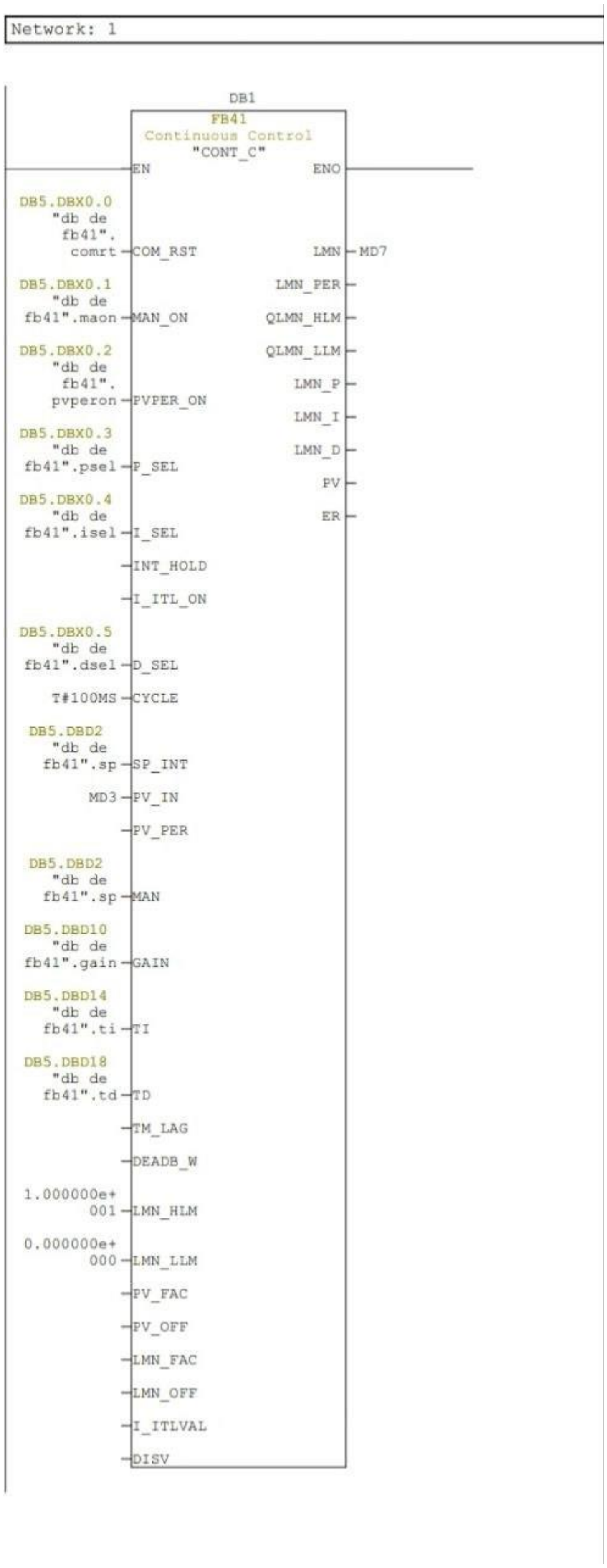
Interface: 02/15/1996 04:51:11 PM

Lengths (block/logic/data): 00392 00278 00026

Name	Data Type	Address	Comment
TEMP		0.0	
OB35_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB35_STRT_INF	Byte	1.0	16#36 (OB 35 has started)
OB35_PRIORITY	Byte	2.0	Priority of OB Execution
OB35_OB_NUMBR	Byte	3.0	35 (Organization block 35, OB35)
OB35_RESERVED_1	Byte	4.0	Reserved for system
OB35_RESERVED_2	Byte	5.0	Reserved for system
OB35_PHASE_OFFSET	Word	6.0	Phase offset (msec)
OB35_RESERVED_3	Int	8.0	Reserved for system
OB35_EXC_FREQ	Int	10.0	Frequency of execution (msec)
OB35_DATE_TIME	Date_And_Time	12.0	Date and time OB35 started

Block: OB35 "Cyclic Interrupt"

Annex



Network: 1
DB1
FB41
Continuous Control
"CONT_C"
EN
ENO
DB5.DBX0.0
"db de fb41". comrt
COM_RST
LMN
MD7
LMN_PER
DB5.DBX0.1
"db de fb41".maon
MAN_ON
QLMN_HLM
DB5.DBX0.2
"db de fb41". pvperon
PVPER_ON
QLMN_LLM
LMN_P
LMN_I
DB5.DBX0.3
"db de fb41".psel
P_SEL
LMN_D
PV
DB5.DBX0.4
"db de fb41".isel
I_SEL
ER
INT_HOLD
I_ITL_ON
DB5.DBX0.5
"db de fb41".dsel
D_SEL
T#100MS
CYCLE
DB5.DBD2
"db de fb41".sp
SP_INT
MD3
PV_IN
PV_PER
DB5.DBD2
"db de fb41".sp
MAN
DB5.DBD10
"db de fb41".gain
GAIN
DB5.DBD14
"db de fb41".ti
TI
DB5.DBD18
"db de fb41".td
TD
TM_LAG
DEADB_W
1.000000e+001
LMN_HLM
0.000000e+000
LMN_LLM
PV_FAC
PV_OFF
LMN_FAC
LMN_OFF
I_ITLVAL
DISV

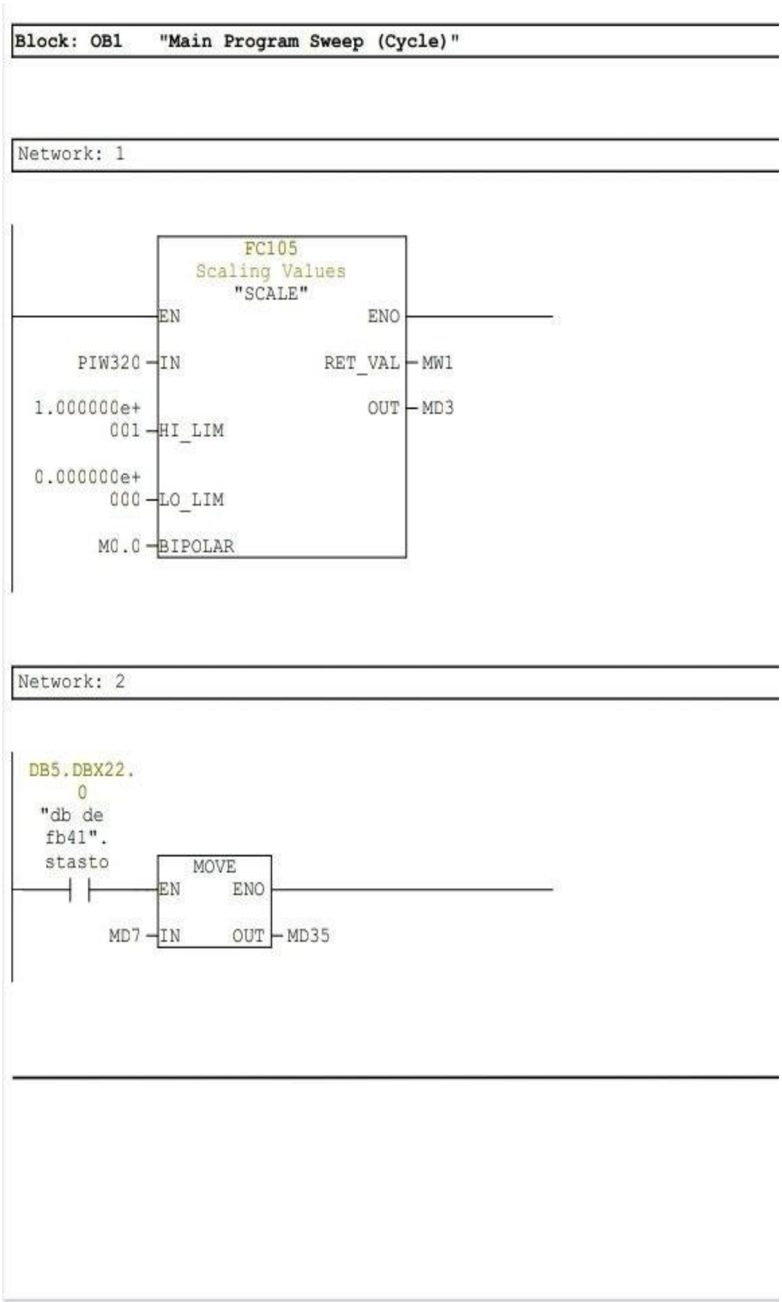
OB1 - <offline>

""

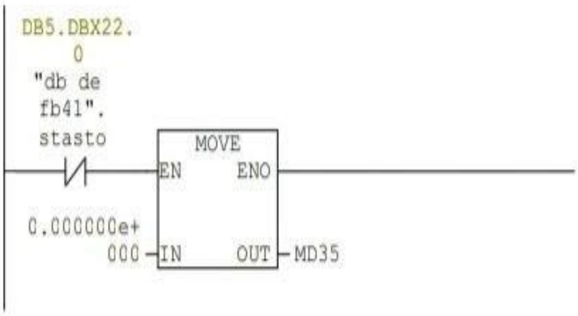
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 01/01/2009 09:14:23 PM
Interface: 02/15/1996 04:51:12 PM
Lengths (block/logic/data): 00298 00174 00030

Name	Data Type	Address	Comment
TEMP		0.0	
OB1_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1	Byte	1.0	1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY	Byte	2.0	Priority of OB Execution
OB1_OB_NUMBR	Byte	3.0	1 (Organization block 1, OB1)
OB1_RESERVED_1	Byte	4.0	Reserved for system
OB1_RESERVED_2	Byte	5.0	Reserved for system
OB1_PREV_CYCLE	Int	6.0	Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE	Int	8.0	Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE	Int	10.0	Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME	Date_And_Time	12.0	Date and time OB1 started

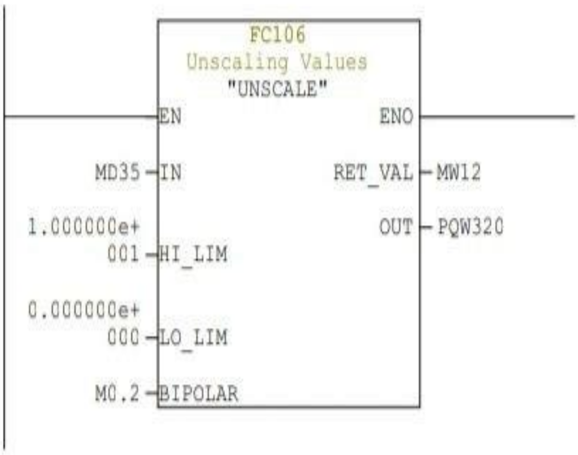
Annex


Block: OB1 "Main Program Sweep (Cycle)"
Network: 1
FC105
Scaling Values
"SCALE"
EN
ENO
PIW320
IN
RET_VAL
MW1
1.000000e+
001
HI_LIM
OUT
MD3
0.000000e+
000
LO_LIM
M0.0
BIPOLAR
Network: 2
DB5.DBX22.
0
"db de
fb41".
stasto
MOVE
EN
ENO
MD7
IN
OUT
MD35

Network: 3



Network: 4



Block: DB5

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	comrt	BOOL	FALSE	
+0.1	maon	BOOL	FALSE	
+0.2	pvperon	BOOL	FALSE	
+0.3	psel	BOOL	TRUE	
+0.4	isel	BOOL	TRUE	
+0.5	dsel	BOOL	TRUE	
+2.0	sp	REAL	0.000000e+000	
+6.0	man	REAL	0.000000e+000	
+10.0	gain	REAL	7.200000e+000	
+14.0	ti	TIME	T#13MS	
+18.0	td	TIME	T#180MS	
+22.0	stasto	BOOL	FALSE	
=24.0		END_STRUCT		