Vanessa Mariro

Integration of Embedded System and PLC for Industrial Automation

School of Technology and Innovations
Master's in Smart Energy - Robotics

Vaasa 2026

UNIVERSITY OF VAASA
School of Technology and Innovations
Author: Vanessa Mariro
Title of the Thesis: Integration of Embedded System and PLC for Industrial Automation.
Degree: Master of Science in Technology
Programme: Smart Energy - Robotics
Supervisor: Janne Koljonen and Jani Boutellier
Year: 2026 **Pages:** 54

ABSTRACT:

Industrial automation systems increasingly require the integration of different control technologies to support intelligent and efficient manufacturing processes. This thesis investigates the integration of a Siemens Programmable Logic Controller (PLC), an Arduino-based embedded system, and a robotic arm within a unified automation architecture. The objective was to establish reliable communication between the PLC and the embedded system, which served as a communication gateway for processing and transmitting robot control commands to the robotic arm.

The developed system consisted of a Siemens ET 200SP PLC, an Arduino Uno with an Ethernet Shield, and a Waveshare RoArm-M2-S robotic arm. The PLC generated robot joint angle values and transmitted them to the Arduino using the Modbus TCP protocol. The Arduino processed the data and forwarded movement commands to the robotic arm through UART serial communication.

System performance was evaluated using TIA Portal online monitoring tools, Modbus diagnostics, outputs from the Arduino Serial Monitor, and observation of robotic arm movements. The results confirmed successful communication between all devices and reliable execution of robot movement commands.

The thesis demonstrates that embedded systems can effectively serve as communication gateways between industrial PLCs and robotic devices. Furthermore, the integration of Modbus TCP and UART protocols provides a practical solution for industrial automation applications requiring data exchange between industrial controllers and external devices.

KEYWORDS: Industrial automation, Programmable Logic Controller, Embedded Systems, Arduino, UART communication, Robotic arm, Modbus TCP.

Contents

Figures

Abbreviations

AI	Artificial Intelligence
CPU	Central Processing Unit
CPS	Cyber-Physical System
DB	Data Block
GND	Ground
HMI	Human Machine Interface
I/O	Input/ Output
IDE	Integrated Development
IEC	International Electrotechnical Commission
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
LAD	Ladder Logic Diagram
MB_CLIENT	Modbus Client
OB	Organization Block
PLC	Programmable Logic Controller
PN/IE	PROFINET/ Industrial Ethernet
PWM	Pulse Width Modulation
RX	Receive
SCADA	Supervisory Control and Data Acquisition
SPI	Serial Peripheral Interface
TCP/IP	Transmission Control Protocol / Internet Protocol

TIA Portal	Totally Integrated Automation Portal
TX	Transmit
UART	Universal Asynchronous Receiver/Transmitter

1 Introduction

Industrial automation has revolutionized the manufacturing sector and has become essential for modern manufacturing processes, transforming manufacturing from human manual labor into machine-driven systems powered by Artificial Intelligence (AI), enabling machines to operate with minimal or no human supervision.

Industry 4.0 relies heavily on embedded and intelligent control systems to improve efficiency and enable smart production processes (Shylla et al., 2023).
Industry 4.0 aims to optimize industrial production processes and has led to the development of key advanced technologies such as programmable logic controllers (PLCs) and embedded systems. PLCs are industrial digital computers with automation systems that control industrial machinery, while embedded systems are specialized computing systems designed to perform specific tasks.

PLCs are widely used in industrial environments due to their robustness, real-time control capabilities, and reliability (Bolton, 2015). The integration of embedded systems and PLCs creates a robust and efficient automation system that allows factories to achieve smart process control and real-time decision-making. This integration helps industries improve product quality, reduce manufacturing errors, and increase productivity, making manufacturing processes more adaptable, flexible, and efficient.

Modern industrial systems require interconnection and intelligent control architectures rather than isolated systems (Ferrarini et al., 2008). In today's competitive manufacturing environment, the integration of advanced technologies is crucial for maintaining the efficiency of machinery. Relying only on traditional PLCs and embedded systems is not sufficient, as customer demands for high-quality products increase and production processes become more complex.
Industrial automation systems are complex distributed systems that require high levels of reliability, safety, and efficiency (Sinha et al., 2019). Integrating both PLCs and

embedded systems helps industries develop intelligent, responsive automation systems that are capable of managing equipment in real-time and performing dynamic tasks effectively and efficiently.

1.1 Scope of the project

This thesis explores the integration and implementation of Programmable Logic Controllers (PLCs) and embedded systems in industrial manufacturing processes. The thesis aims to demonstrate the benefits of integrating these technologies in modern production operations. The scope includes analyzing how this integration improves communication, automation efficiency, and coordination between industrial controllers and robotic systems.

The thesis focuses on the practical integration of an Arduino Uno embedded system and a Siemens ET 200SP PLC to control a Waveshare RoArm-M2-S robotic arm. Communication between the PLC and the embedded system is implemented using the Modbus TCP protocol over an Ethernet network, while UART serial communication is used to transmit robot movement commands from the Arduino to the robotic arm controller.

The project covers the configuration of the PLC and Arduino, the implementation of the communication architecture, and the development of the control programs required for data exchange and robot control. Embedded systems support flexible communication, device-level control, and real-time interaction in industrial automation applications (Shylla et al., 2023). The implemented system is evaluated based on the successful transmission of robot joint angle values, the execution of robot movement commands, and the performance and reliability of the integrated communication system. The thesis also discusses the limitations of the implemented system, including the absence of feedback control, and proposes future improvements through the integration of sensors to support closed-loop robotic arm control.

1.2 Project objectives

The main objective of this thesis is to integrate an Arduino embedded system with a Siemens PLC for industrial automation applications. The system aims to establish reliable communication between the PLC and the embedded system using Modbus TCP and to transmit robot control commands from the embedded system to the robotic arm through UART serial communication.

To achieve this objective, the project establishes reliable Modbus TCP communication between the Siemens PLC and the Arduino embedded system and implements the Arduino as a communication gateway between the PLC and the robotic arm controller. The system is designed to transmit robot joint angle values and movement commands from the PLC to the robotic arm through the Arduino. In addition, the project verifies the correct operation of the developed system using TIA Portal online monitoring, the Test_Modbus watch table, the Arduino Serial Monitor, and observation of the robotic arm movement. Finally, the performance and reliability of the integrated PLC-embedded system communication architecture are evaluated.

1.3 Research questions

1. How can reliable communication be established between a Siemens ET 200SP and an embedded system (Using Modbus TCP)?
2. How can an embedded system act as a communication gateway between industrial PLC and the robotic controller hardware?
3. How can UART serial communication be used to transmit robot control commands from embedded systems to the robot controller board?
4. How effective is the integration of Modbus TCP and UART communication in industrial automation applications?
5. What are the performance and reliability characteristics of the proposed PLC-embedded-robot communication?

2 Literature review

This chapter provides a literature review of industrial automation systems, embedded systems, Programmable Logic Controllers (PLCs), industrial communication protocols, and robotic control systems. It also examines industrial automation within Industry 4.0 environments and the technologies used in the system.

2.1 Industrial Automation Systems

Industrial automation is the use of control systems, computational technologies, and intelligent devices to operate industrial processes with minimal human intervention. Automation systems combine technologies such as sensors, Programmable Logic Controllers (PLCs), embedded systems, robotics, communication networks, and industrial software to monitor and control industrial operations efficiently. These systems are designed to improve productivity, operational safety, accuracy, and system reliability within manufacturing environments.

Automation technologies such as embedded systems and PLCs play a major role in industrial processes because they provide automatic control capabilities for machinery and manufacturing operations. PLCs are widely used in industrial automation due to their robustness, reliability, ease of programming, and ability to operate under harsh conditions (Bolton, 2015). Embedded systems, on the other hand, provide flexible and low-cost solutions for communication, signal processing, and device-level control operations. The integration of PLCs and embedded systems allows industrial automation systems to achieve both reliable, high-level control and flexible low-level device interaction.

The development of Industry 4.0 has significantly transformed industrial automation systems by introducing intelligent and interconnected technologies into manufacturing environments. Industry 4.0, also referred to as Fourth Industrial Revolution, integrates technologies such as Internet of Things (IoT), Artificial Intelligence (AI), cloud computing, cyber-physical systems (CPS), and industrial communication networks into industrial

processes (Lasi et al., 2014). These technologies enable machines, controllers, sensors, and embedded devices to communicate, exchange data, and make decentralized decisions in real time, creating smart and adaptive industrial environments (Hermann et al., 2016).

2.2 Comparison Between Traditional Industrial Automation and Industry 4.0

Traditional industrial automation systems mainly operate using fixed and hierarchical control structures based on technologies such as PLCs, SCADA systems, and relay-based control systems. These systems were primarily designed for repetitive mass production operations and offered limited flexibility for system expansion and machine–to-machine communication (Folgado et al., 2024).

In contrast, Industry 4.0 automation systems integrate IoT, CPS, AI, and cloud technology to establish intelligent and interconnected manufacturing environments. Industry 4.0 allows industrial devices such as PLCs, embedded systems, robots, and sensors to exchange information continuously through communication networks, enabling real-time monitoring and autonomous operational decision-making (Lasi et al., 2014).

Furthermore, Industry 4.0 technologies improve operational flexibility, responsiveness, predictive maintenance, and smart production management compared to traditional automation systems capable of responding efficiently to changing industrial requirements and production conditions (Hermann et al., 2016). A comparison between traditional automation and Industry 4.0 reveals significant differences in communication capabilities, monitoring, decision-making, and system flexibility. While traditional automation systems rely on fixed control structures and limited machine-to-machine communication, Industry 4.0 supports interconnected devices, real-time monitoring, autonomous decision-making, and intelligent production processes, resulting in more adaptive and efficient manufacturing environments.

2.3 Programmable Logic Controllers (PLCs)

Programmable Logic Controllers (PLCs) are industrial digital computers designed specifically for automation and machine control applications within industrial environments. PLC systems are widely used in manufacturing industries because they provide reliable real-time operation, high stability, and continuous control capability under harsh industrial conditions (Bolton, 2015). PLCs are capable of performing industrial tasks such as sequencing, timing, logic operations, process monitoring, and machine control.

PLCs use programming memory to store instructions for logic operations, sequencing, timing, counting, and process control functions, enabling industrial processes to be automated and controlled efficiently (Bolton, 2015). Modern PLC systems such as Siemens SIMATIC controllers support modular hardware architecture, industrial communication protocols, and high-speed processing capabilities. These features enable PLCs to communicate with industrial devices such as sensors, embedded systems, robotic systems, and Human-Machine Interface (HMIs) through industrial communication networks.

PLCs are commonly programmed using Ladder Logic Diagram (LAD), Structured Text (ST), and other standardized programming languages. Among these methods, Ladder Logic is one of the most widely implemented programming approaches because it is based on traditional electrical relay logic and provides a graphical representation of industrial automation processes. Ladder Logic simplifies the design, monitoring, troubleshooting, and maintenance of industrial control systems (Thramboulidis, 2014). In addition, engineering software such as Siemens TIA Portal allows PLC hardware configuration, communication setup, program development, monitoring, and diagnostics to be implemented within a single software environment.

PLC systems are widely implemented in industrial applications such as robots, especially in robotic manufacturing lines, conveyor systems, packaging systems, process industries, and automated production systems due to their reliability and predictable operation

(Koondhar et al., 2023). However, PLCs have limitations when performing advanced robotic motion control operations such as high-resolution Pulse Width Modulation (PWM), complex servo motor control, and low-level actuator signal processing. These operations often require embedded systems and microcontrollers that provide faster low-level signal processing and flexible hardware interfacing.

2.4 Embedded systems

Embedded systems are integrated hardware and software platforms developed to perform dedicated functions within larger electrical or mechanical systems. Unlike general-purpose computers, embedded systems are developed to execute specific tasks with high efficiency, reliability, and real-time responsiveness (Wolf, 2022). In industrial automation, embedded systems are commonly used for device control, communication processing, robotic operations, sensor interfacing, and real-time monitoring tasks.

Embedded systems provide flexible hardware interfacing and the low-level control capability required in industrial automation systems (Zurawski, 2018). Their ability to process data quickly and interact directly with hardware devices makes them suitable for modern automation and robotic applications. With the development of Industry 4.0, embedded systems have become increasingly essential in smart manufacturing systems. These systems enable communication between machines, controllers, sensors, and robots through industrial communication networks such as Modbus TCP/IP, PROFINET, Ethernet/IP, and UART serial communication. They support real-time data exchange and intelligent automation operations within interconnected industrial systems (Hermann et al., 2016).

Embedded systems are often integrated with Programmable Logic Controllers (PLCs) to improve automation flexibility and communication capabilities. In these systems, PLCs usually handle high-level industrial control operations, while embedded systems perform low-level hardware communication and actuator control tasks. Microcontroller

devices such as Arduino are widely adopted in automation applications because they are cost-effective, flexible, and support communication protocols such as UART and Ethernet-based communication. In this project, an Arduino Uno with an Ethernet Shield functions as an intermediate embedded controller between the Siemens PLC and the robotic arm. The Arduino receives commands from the PLC through Modbus TCP/IP communication and transfers motion commands to the robot arm using UART serial communication.

Embedded systems are widely used in industrial automation due to their low cost, flexibility, and ability to interact with hardware devices. Their processing capabilities and communication capabilities make them suitable for applications involving actuators, sensors, and robotic systems.

2.5 Integration of Embedded Systems and PLCs

Modern industrial automation systems increasingly rely on the integration of different control technologies to improve device and machine communication, flexibility, and overall system performance. The development of Industry 4.0 has accelerated the need for interconnected automation systems in which sensors, embedded devices, and robotic systems exchange data through communication networks to achieve intelligent and efficient operation (Lasi et al., 2014; Hermann et al., 2016).

The integration of embedded systems and PLCs combines the reliability and industrial robustness of PLC-based control with the flexibility and communication capabilities of embedded devices. This approach allows automation systems to perform complex operations that may be difficult to implement using a single controller alone. According to Minchala et al. (2020), embedded systems play an important role in industrial automation by providing communication management, data processing, and device-level control functions that complement industrial control systems. Their integration with PLC-based architecture provides reliable communication and flexible control within industrial automation.

Furthermore, integrating embedded systems and PLCs supports the development of distributed control architectures. In a distributed automation system, different devices are assigned specific responsibilities while operating together as a coordinated system. This architecture improves system modularity, simplifies maintenance, and enhances flexibility for future upgrades and expansion (Hermann et al., 2016). Minchala et al. (2020) further highlighted that embedded systems platforms can effectively serve as communication interfaces between industrial controllers and external devices by processing communication data and supporting hardware-level interactions.

In the context of this project, integration is achieved through communication between a Siemens SIMATIC ET 200SP PLC and an Arduino Uno equipped with an Ethernet Shield. The PLC generates automation commands and transfers data to the Arduino through Modbus TCP/IP communication over Ethernet. The Arduino microcontroller then processes the received data and forwards control commands to the robotic arm controller through UART serial communication. This structure establishes a communication pathway between the industrial control layer and the robotic execution layer, enabling coordinated operation of the overall system.

The integration of embedded systems and PLCs provides a practical solution for connecting industrial control systems with robotic devices while maintaining reliable communication, modular system design, and future expandability. Such architectures are increasingly implemented in modern industrial automation systems where multiple devices must communicate and operate together efficiently.

2.6 Industrial communication protocols

Industrial communication protocols are important technologies for enabling data exchange between embedded systems, robotic devices, and actuators within automation systems. These protocols establish standardized communication methods that allow different devices to coordinate their operations and share information efficiently. As industrial automation systems become increasingly interconnected, communication

technologies have become a fundamental component of modern control architecture (Mahalik, 2007).

The development of Industry 4.0 has increased the demand for seamless communication between automation devices. Modern industrial systems rely on continuous information exchange to support monitoring, control, diagnostics, and decision-making processes. According to Xu et al. (2018), communication networks provide the foundation for intelligent and interconnected industrial systems by enabling real-time data sharing between devices. Similarly, Shylla et al. (2023) highlighted that communication capabilities are essential for integrating embedded systems, controllers, and other automation devices within Industry 4.0 systems.

2.6.1 Modbus TCP/IP communication

Modbus TCP/IP is an Ethernet-based industrial communication protocol that combines the Modbus communication standard with TCP/IP networking technology. The protocol allows devices to exchange data over Ethernet networks while maintaining a simple and efficient communication structure. Due to its interoperability and compatibility with equipment from different manufacturers, Modbus TCP/IP remains one of the most widely used communication protocols in industrial automation systems (Modbus Organization, 2024).

Modbus TCP/IP communication follows a client-server architecture. In this communication model, the client initiates communication requests, while the server waits for the requests and responds by providing or updating data stored in Modbus registers. This architecture supports reliable and organized data exchange between industrial devices connected through an Ethernet network.
Xu et al. (2018) explained that industrial communication networks provide the foundation for interconnected automation systems by enabling reliable data exchange, system integration, and scalable communication architectures required in Industry 4.0 environments. These capabilities make Modbus TCP/IP suitable for applications that require

communication between PLCs, embedded systems, Human-Machine Interfaces (HMIs), and other industrial devices.

In this project, Modbus TCP/IP is implemented to establish communication between the Siemens SIMATIC ET 200SP PLC and the Arduino Uno equipped with an Ethernet Shield. Through this communication network, control parameters generated by the PLC are transmitted to the Arduino for processing and subsequent execution by the robotic system.

2.6.2 UART serial communication

Universal Asynchronous Receiver-Transmitter (UART) is a serial communication protocol commonly used for direct communication between electronic devices. UART communication transfers data through dedicated Transmit (TX) and Receive (RX) lines, allowing devices to exchange information without requiring a network infrastructure. The protocol is widely implemented in embedded systems due to its simple implementation, low hardware requirements, and reliable short-distance communication performance (Wolf, 2022).

UART communication is frequently implemented in microcontroller-based systems, robotic applications, and motor controllers where direct device-to-device communication is required. Its straightforward hardware configuration and low communication overhead make it particularly suitable for embedded control applications.

In the developed system, UART communication is used between the Arduino Uno and the RoArm-M2-S robotic arm controller. After receiving control data from the PLC through Modbus TCP/IP communication, the Arduino transmits robot arm movement commands to the robotic arm via UART communication. This configuration provides a direct communication interface between the embedded controller and the robot control board.

The implementation of both Modbus TCP/IP and UART communication creates a layered communication architecture that combines industrial Ethernet communication with direct hardware-level device control. This approach facilitates reliable communication between the PLC, embedded system, and robotic arm to support efficient operation of the overall automation system.

2.7 Robotic arm control systems

Robotic arms are programmable electromechanical systems designed to perform tasks such as object handling, assembly, inspection, and material transportation. They are widely used in industrial automation because they provide high precision, repeatability, and operational efficiency (Craig, 2018). A robotic arm consists of multiple joints and actuators that work together to produce controlled movement. These movements are coordinated by a control system that processes commands and positions the robot according to the required task (Spong et al., 2020). As a result, robotic arms can perform automated tasks accurately and efficiently in industrial environments.

With the advancement of modern automation systems, robotic arms are commonly integrated with controllers, sensors, and communication networks to perform automated operations efficiently (Hermann et al., 2016). In many applications, industrial controllers generate high-level commands while embedded systems handle communication and motion execution (Siciliano & Khatib, 2016).

In this project, the RoArm-M2-S robotic arm functions as the final execution device. Motion commands are generated by the Siemens PLC, processed by the Arduino embedded system, and then transmitted to the robotic arm through UART serial communication. This integration supports coordinated robot movement within the proposed industrial automation system.

2.8 TIA Portal and Ladder Programming

TIA Portal (Totally Integrated Automation Portal) is Siemens' engineering software platform used for programming, configuring, monitoring, and diagnosing industrial automation systems. The software provides a single environment for managing PLC programs, communication settings, hardware configuration, and system diagnostics, simplifying the development and maintenance of automation projects (Siemens AG, 2024).

Ladder Logic Diagram (LAD) is one of the most widely used PLC programming languages supported by TIA Portal and defined in the IEC 61131-3 standard. Ladder Logic uses graphical symbols that resemble electrical relay circuits, making it easier for users, engineers, and technicians to design and troubleshoot automation operations (Bolton, 2015).

The project implements TIA Portal to configure the Siemens PLC, establish communication settings, and develop the Ladder Logic program responsible for generating and managing robot control commands within the automation system.
Ladder Logic is widely implemented in industrial automation because it provides a clear representation of control operations such as sequencing, timing, interlocking, and machine control. Its visual structure simplifies program development and maintenance while ensuring reliable and predictable system operation (Christoulakis et al., 2016). The integrated tools within TIA Portal enable efficient hardware configuration, communication management, program monitoring, and system diagnostics within a single engineering platform.

3 System Architecture

This chapter presents the system architecture of the developed automation system and describes the hardware and software components used in the project. The chapter also discusses the communication framework, network configuration, and control structure implemented to enable communication between the Siemens PLC, the Arduino embedded system, and robotic arm. Furthermore, the chapter explains how these components were integrated to achieve coordinated system operation and robotic arm control.

3.1 Project System Overview

This project presents the design and implementation of an industrial robotic arm control system integrating an embedded system (Arduino Uno with Ethernet Shield), a Siemens Programmable Logic Controller (PLC), and a Waveshare (RoArm-M2-S) robot arm. The objective of the project is to enable the Siemens PLC to indirectly control the robotic arm through the embedded system using Ethernet and serial communication technologies.

The proposed system implements a distributed control structure in which the PLC is responsible for the overall high-level decision-making and control functions, while the embedded system handles low-level signal processing and actuator control. This architectural setup is widely implemented in modern industrial automation to enhance scalability, flexibility, and overall performance within a modular design (Marcon, 2023). The system architecture reflects Industry 4.0 concepts where interconnected intelligent devices communicate and cooperate through industrial networks to achieve automation functionality (Lasi et al., 2014).

3.2 Hardware Components Implemented

The system consists of five main hardware components such as a Siemens PLC, an Arduino Uno with an Ethernet Shield, a network switch, a RoArm-M2-S robotic arm, and a

laptop, as shown in Figure 1. These components were integrated to establish communication between the industrial control system and the robotic platform.

The Siemens PLC functions as the primary controller of the system and is responsible for generating robot control data, initiating communication with the Arduino, and managing the overall operation of the automation system. The Arduino Uno equipped with an Ethernet Shield is used as an embedded communication interface between the PLC and the robotic arm. The Arduino's role is to receive control data from the PLC, process the received data into robot movement commands, and forward the corresponding commands to the robotic arm controller.

A network switch is used to connect the laptop, Arduino, and PLC through a wired Ethernet network. The switch acts as the central connection point between the networked devices within the system and allows the interconnected devices to establish network communication.

The RoArm-M2-S robotic arm serves as the final execution device within the system. It performs physical movements based on commands received from the Arduino through serial communication. A laptop is used for PLC programming, Arduino programming, monitoring, and testing throughout the project implementation process.

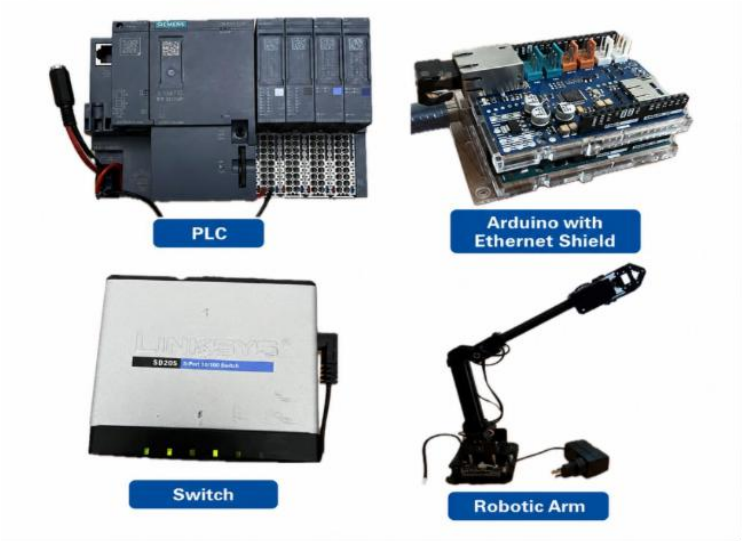


Figure 1. Hardware components used in the project.

3.3 System Structure

The project system architecture defines how the hardware components are organized and interconnected to achieve coordinated robot control. Figure 2 illustrates the overall system architecture and interconnection between hardware components. The architecture integrates the Siemens PLC, Arduino embedded system, network switch, and robotic arm into a single automation structure. Each component performs a specific function within the system, allowing commands and data to flow through a structured control process.

The architecture was designed to provide a clear separation between control, communication, and execution functions. This approach improves system organization and simplifies future modification, maintenance, and expansion of the automation system.

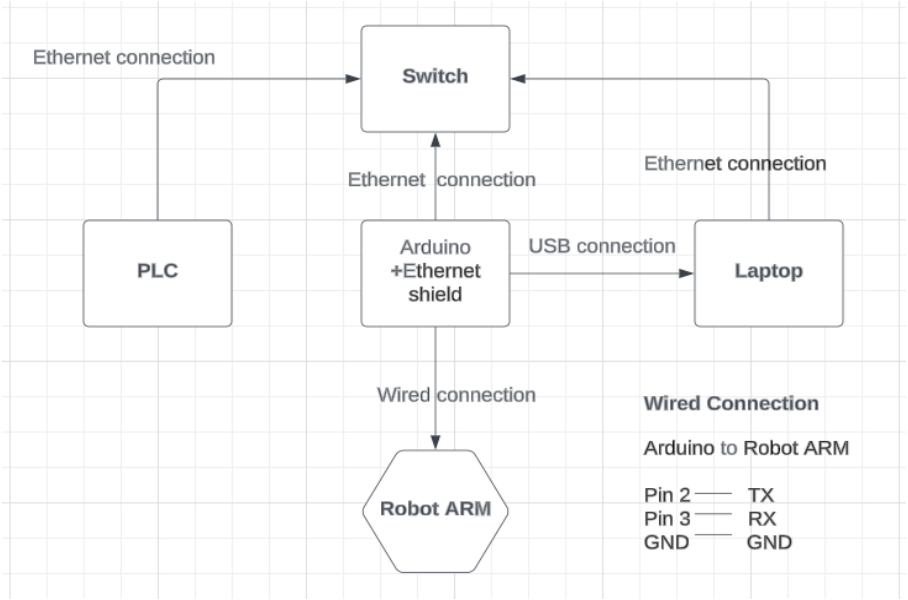


Figure 2. The project's overall system architecture and interconnection between the hardware components.

3.4 Project System Communication Framework

The communication framework was developed to enable data exchange between the Siemens PLC, the Arduino embedded system, and RoArm-M2-S robotic arm. Two communication methods were implemented to support the system's operation. The first communication layer manages data transfer between the PLC and Arduino, while the second communication layer transfers robot control commands from the Arduino to the robotic arm controller.

3.4.1 PLC-Arduino Communication

Communication between the Siemens PLC and Arduino was implemented through the Ethernet network. The PLC transmitted robot control parameters that were received and processed by the Arduino before being prepared for robot execution. Data exchanged between the devices included robot joint angles, movement positions, speed values, and gripper control commands. The communication setup was configured to allow the PLC to continuously update command values during operation. This enabled the Arduino to receive the latest control parameters and prepare corresponding robot movement instructions.

3.4.2 Arduino–Robot Arm Communication

A serial communication interface was implemented as the second communication layer of the system to transfer robot control commands from Arduino to the robotic arm controller. The Arduino acted as an intermediary controller by converting the received PLC data into robot-specific commands required for movement functions.
The communication link was implemented using TX, RX, and GND connections between the Arduino and the robotic arm. Once the command values were received from the PLC, the Arduino generated the appropriate control instructions and transmitted them to the robotic arm controller for movement execution.

The serial communication link transfers movement-related parameters to the robotic arm. By separating robot command transmission from the network communication layer, the system maintains an organized communication structure and facilitates efficient interaction between the control system and the robotic arm.

3.5 Network Infrastructure

The network configuration was implemented in the project to enable communication between hardware devices through an Ethernet network using a network switch to interconnect them. Static IP addressing was assigned to the network devices to ensure consistent device identification during operation. The Siemens PLC and Arduino embedded system were configured within the same Ethernet network to support reliable communication. This configuration ensured stable device connectivity and simplified communication management during system operation and testing.

The use of a static IP address prevents the automatic address changes that may occur with dynamic network allocation methods and ensures that communication settings remain consistent during startup and operation. This configuration simplifies device connectivity and provides reliable communication between the PLC and the embedded system.

3.6 Software and Control System Design

The software implementation for the project was carried out using Siemens Totally Integrated Automation Portal Version 20 (TIA Portal V20) and Arduino IDE. TIA Portal was used to configure the Siemens PLC, develop the control program, and establish communication settings. Arduino IDE was used to develop and upload the embedded system firmware to the Arduino Uno.

The PLC control program was developed using Ladder Logic Diagram (LAD) within TIA Portal. The software environment also provided tools for hardware configuration, communication, diagnostics, and online monitoring during system testing. Arduino IDE was used for firmware development, program uploading, serial monitoring, and debugging operations throughout the project implementation process.

These software platforms provided the development environment required for programming, configuring, testing, and monitoring the hardware components implemented in this project.

4 Configuration and Programming

The configuration and programming of the developed automation system involved the setup of hardware devices, communication parameters, and control programs required for system operation. Device configuration was carried out in TIA Portal and Arduino IDE, while communication settings and program structures were implemented to establish communication between the Siemens PLC, Arduino embedded system, and robotic arm.

4.1 Device Configuration in TIA Portal

The project devices were configured to establish reliable communication and coordinated operation between the Siemens PLC, Arduino embedded system, and the robotic arm. This configuration phase involved Ethernet network setup, hardware installation, software configuration within Arduino IDE and TIA Portal, and serial communication setup. According to Lasi et al. (2014), the distributed control system architecture is commonly utilized in industrial automation processes to enhance system scalability, communication efficiency between devices, and flexibility.

The Siemens PLC was configured in the TIA Portal V20 using the CPU 1512SP F-1 PN controller. Figure 3 shows the PLC IP address configuration and the connection to the PN/IE interface. A static IP address of 192.168.72.199 with a subnet mask of 255.255.255.0 was assigned to the PLC to establish stable Ethernet communication within the industrial network. The PLC was then connected to a network switch through the PROFINET (PN/IE) interface. A static IP address provides reliable and fixed device identification and ensures stable communication between industrial devices connected within the same network (Siemens AG, 2024).

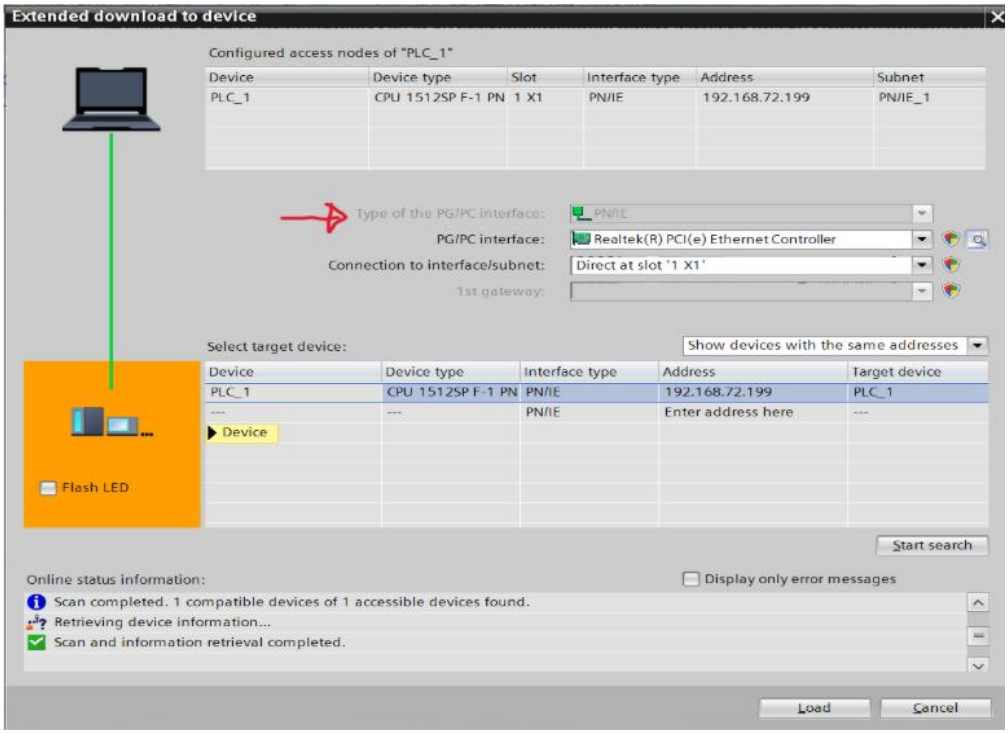


Figure 3. PLC IP address configuration and connection to PN/IE interface.

4.2 Arduino IDE Program Implementation

The Arduino Uno equipped with an Ethernet Shield was configured and programmed using Arduino IDE to operate as an embedded communication gateway between the Siemens PLC and the robotic arm. In this configuration, the PLC transmitted robot joint angle values to the Arduino through Modbus TCP, while the Arduino received the values and converted them into serial commands for the robotic arm controller.

4.2.1 Arduino Libraries and Network Configuration

The first section of the Arduino program includes the software libraries required to support communication between the Siemens PLC, Arduino, and the robotic arm. Each library was selected to perform a specific function within the system. The SPI library facilitates communication between the Arduino Uno and the Ethernet Shield through the Serial Peripheral Interface (SPI) protocol. The Ethernet library provided the networking

functions required to configure the Arduino on the Ethernet network and establish TCP/IP communication. The ModbusTCPSlave library allows the Arduino to function as a Modbus TCP server, allowing it to receive robot joint angle values transmitted by the PLC. In the developed system, the Siemens PLC functions as the Modbus TCP client by initiating communication requests, while the Arduino functions as the Modbus TCP server by receiving the requests and responding with the requested register data. Finally, the SoftwareSerial library was used to establish serial communication between the Arduino and the robotic arm controller, allowing the transmission of movement commands generated from the received PLC data.

```
#include <SPI.h>
#include <Ethernet.h>
#include <ModbusTCPSlave.h>
#include <SoftwareSerial.h>
```

As shown in Figure 4, the Arduino was assigned a static IP address of 192.168.72.20, with the gateway set to 192.168.72.1 and the subnet mask set to 255.255.255.0. These settings ensured that the Arduino and Siemens PLC operate within the same local Ethernet network, enabling communication between the devices. The Modbus TCP communication port was also configured as port 502, allowing the Arduino to function as a Modbus TCP server and receive communication requests from the PLC.

Lastfinal.ino

```
1   #include <SPI.h>
2   #include <Ethernet.h>
3   #include <ModbusTCPSlave.h>
4   #include <SoftwareSerial.h>
5
6   byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
7
8   IPAddress ip(192, 168, 72, 20);
9   IPAddress gateway(192, 168, 72, 1);
10  IPAddress subnet(255, 255, 255, 0);
11
12  EthernetServer ethernetServer(502);
13  ModbusTCPSlave modbus;
14
15  SoftwareSerial robotSerial(2, 3); // RX, TX
16
17  #define NUM_REGISTERS 7
18  uint16_t holdingRegisters[NUM_REGISTERS] = {0, 0, 0, 0, 0, 0, 0};
19
20  bool commandDone = false;
```

Figure 4. Arduino Ethernet network configuration showing the libraries, IP address, gateway, subnet and Modbus TCP server settings.

4.2.2 Serial Communication with the Robotic Arm

As shown in Figure 4, line 15 defines the communication pins used for serial data exchange between the Arduino and the robotic arm controller. In this configuration, digital pin 2 is assigned as the receiver (RX) pin, while digital pin 3 is assigned as the transmitter (TX) pin. These pin assignments establish the serial communication channel through which movement commands are exchanged between the two devices. This ensures that the data is transmitted and received through the correct communication paths between the Arduino and the robotic arm controller. Through this interface, the Arduino can forward robot movement commands generated from the PLC, enabling the robot to execute the desired joint movements.

4.2.3 Modbus Register Configuration and Program Execution

To receive data transmitted from the PLC, seven Modbus holding registers were defined within the Arduino program, as shown in Figure 4, lines 17 and 18. The registers were allocated for the robotic arm joint angle values from J1 to J6, while the seventh register

was reserved for the movement signal. An additional Boolean variable, CommandDone, was implemented to prevent repeated execution of the same movement command while the command signal remained active.

The Arduino setup function was responsible for initializing serial communication, Ethernet communication, and the Modbus TCP server during system startup. The function established communication between the Arduino and the robotic arm controller by initializing both the Serial and robotSerial interfaces at a baud rate of 115200. It also initialized the Ethernet interface using the predefined network parameters and started the Modbus TCP server on port 502. Additionally, the holding registers shown in Figure 5, line 28, were configured to store the robot joint angle values assigned to the Modbus server, enabling the Arduino to receive data from the PLC. Serial Monitor messages, including the configured IP address and communication port, are displayed in the Serial Monitor to verify successful initialization of the communication system.

```
21
22  void setup() {
23    Serial.begin(115200);
24    robotSerial.begin(115200);
25
26    delay(2000);
27
28    Ethernet.begin(mac, ip, gateway, subnet);
29    ethernetServer.begin();
30
31    modbus.configureHoldingRegisters(holdingRegisters, NUM_REGISTERS);
32
33    Serial.println("========================================");
34    Serial.println("ARDUINO READY: PLC -> ARDUINO -> ROBOT");
35    Serial.print("Arduino IP: ");
36    Serial.println(Ethernet.localIP());
37    Serial.println("Port: 502");
38    Serial.println("========================================");
39  }
40
41  void loop() {
42    EthernetClient client = ethernetServer.available();
43
```

Figure 5. The Arduino setup function displaying startup information and communication initialization parameters.

During the program execution, the main program continuously monitored communication requests from the PLC shown in Figure 6. When a Modbus TCP request was received, the transmitted joint angle values were stored in the holding registers, and the command register was evaluated. If a valid movement command was detected, the Arduino displayed the received values and transmitted the corresponding command to the robotic arm. The commandDone variable was used to prevent repeated execution of the same command. The implementation of the main program loop.

```
40
41  void loop() {
42    EthernetClient client = ethernetServer.available();
43
44    if (client) {
45      modbus.poll(client);
46
47      if (holdingRegisters[6] == 1 && commandDone == false) {
48        showPLCValues();
49        moveRobot();
50        commandDone = true;
51      }
52
53      if (holdingRegisters[6] == 0) {
54        commandDone = false;
55      }
56    }
57  }
58
```

Figure 6. Arduino main program loop implementation for receiving PLC data and controlling the robotic arm.

For monitoring and debugging purposes, the received joint angle values were displayed in the Arduino Serial Monitor. This allowed the values received by the Arduino to be compared with those transmitted from the PLC, helping to verify that the communication was functioning correctly. Once the data had been received, the Arduino converted the joint values into a JSON-formatted command suitable for the robotic arm controller. The generated command contained the required movement parameters and was transmitted to the robotic arm through the serial communication interface. The implementation of this function is shown in Figure 7.

```
58
59  void showPLCValues() {
60    Serial.println("======================================");
61    Serial.println("VALUES RECEIVED FROM PLC");
62    Serial.print("J1 = "); Serial.println(holdingRegisters[0]);
63    Serial.print("J2 = "); Serial.println(holdingRegisters[1]);
64    Serial.print("J3 = "); Serial.println(holdingRegisters[2]);
65    Serial.print("J4 = "); Serial.println(holdingRegisters[3]);
66    Serial.print("J5 = "); Serial.println(holdingRegisters[4]);
67    Serial.print("J6 = "); Serial.println(holdingRegisters[5]);
68    Serial.print("Command = "); Serial.println(holdingRegisters[6]);
69  }
70
71  void moveRobot() {
72    int j1 = holdingRegisters[0];
73    int j2 = holdingRegisters[1];
74    int j3 = holdingRegisters[2];
75    int j4 = holdingRegisters[3];
76
77    String robotCommand = "{\"T\":102,\"base\":" + String(j1)
78                          ",\"shoulder\":" + String(j2) +
79                          ",\"elbow\":" + String(j3) +
80                          ",\"hand\":" + String(j4) +
81                          ",\"spd\":30}";
82
83    Serial.println("SENDING TO ROBOT:");
84    Serial.println(robotCommand);
85
86    robotSerial.println(robotCommand);
87
88    Serial.println("Robot command sent.");
89    Serial.println("======================================");
90  }
```

Figure 7. Arduino code part used to display PLC data and generate the robot arm movement commands that enable robot movement.

4.3 TIA Portal Implementation

The development of the PLC control system was carried out within the TIA Portal project environment. Implementation within the TIA Portal began with the creation and configuration of the PLC project used throughout the development of the automation system. The project environment was used to configure Siemens PLC hardware, establish communication settings, and prepare the system for program development and testing.

The physical Siemens PLC layout was established in the TIA Portal project environment to match the physical PLC used in the project as shown in Figure 8. The configured hardware consists of a CPU 1512SP F-1 PN controller with the required digital and analog I/O modules. The similarity between the physical hardware and its configuration in TIA Portal

ensures consistency between the software project and the actual controller during program download, commissioning, and operation.

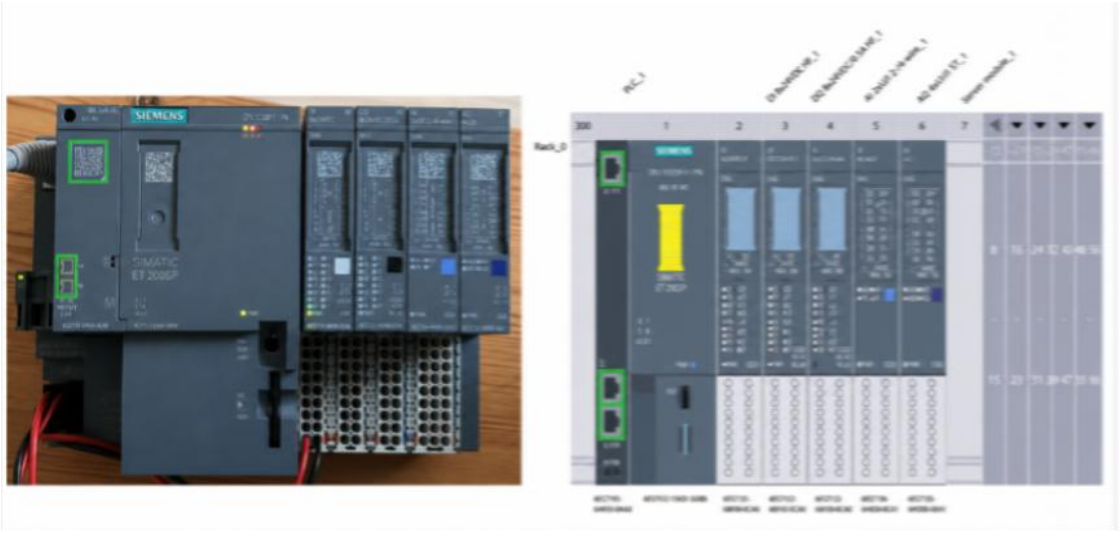


Figure 8. The hardware configuration implemented within the TIA Portal is the same as the physical PLC. Physical Siemens PLC is shown on the left side, and the configured PLC hardware is shown on the right side.

4.4 PLC Programming in TIA Portal

The PLC program was developed in TIA Portal V20 using Ladder Logic Diagram (LAD). The program was designed to implement the control functions, communication routines, and data processing operations required by the system.

To maintain a structured and organized design, the PLC application was developed using the programming block architecture available within the TIA Portal. Several program blocks such as Organization Blocks (OBs), Function Blocks (FBs), Functions (FCs), and Data Blocks (DBs), were implemented to perform specific tasks within the control system. This modular approach improved program organization, simplified troubleshooting activities, and facilitated project's future expansion.

Communication between the PLC and Arduino was implemented using the Modbus TCP protocol through the MB_CLIENT instructions available within the TIA Portal communication library. Communication parameters such as IP addresses, port numbers, connection settings, and Modbus register mappings were defined to establish reliable Ethernet communication between both devices.

4.5 Roles of TIA Programming Blocks

To establish communication between the Siemens PLC, the Arduino, and the robotic arm controller, several program blocks were developed and configured within TIA Portal. Each block was assigned a specific function within the control system architecture, allowing the PLC to generate robotic arm movement commands, manage Modbus TCP communication, and monitor system operation. The coordinated interaction of these blocks enabled reliable transmission of robot joint angle values from the PLC to the Arduino, which subsequently controlled the robotic arm movements.

4.5.1 Organization block OB1 (MAIN)

The Main Organization Block (OB1) serves as the central execution block of the PLC program. It operates continuously whenever the PLC CPU is in RUN mode and continuously executes the programmed instructions. Within this project, OB1 is responsible for coordinating the communication process by calling the MB_CLIENT function block during every scan cycle and managing the transfer of the robotic joint angle values to the Arduino. Figure 9 illustrates the configuration of the MB_CLIENT function block within OB1.

The block is configured to write six robotic joint angle values and one command value from DB_RobotAngles (DB5) to Modbus holding registers starting at address 40001. The communication request is initiated through the Trigger signal (M10.0), while the communication status is monitored using the DONE, BUSY, ERROR, and STATUS outputs in DB_ModbusStatus. The MB_MODE parameter is configured in write mode (1) to transmit the robot joint angle values, while MB_DATA_ADDR, MB_DATA_LEN, and

MB_DATA_PTR define the starting Modbus register address, the number of registers to be transmitted, and the memory location of the data within the DB_RobotAngles data block, respectively. Through this continuous execution, the PLC continuously monitors the communication trigger signal and processes robot movement commands whenever new data becomes available.

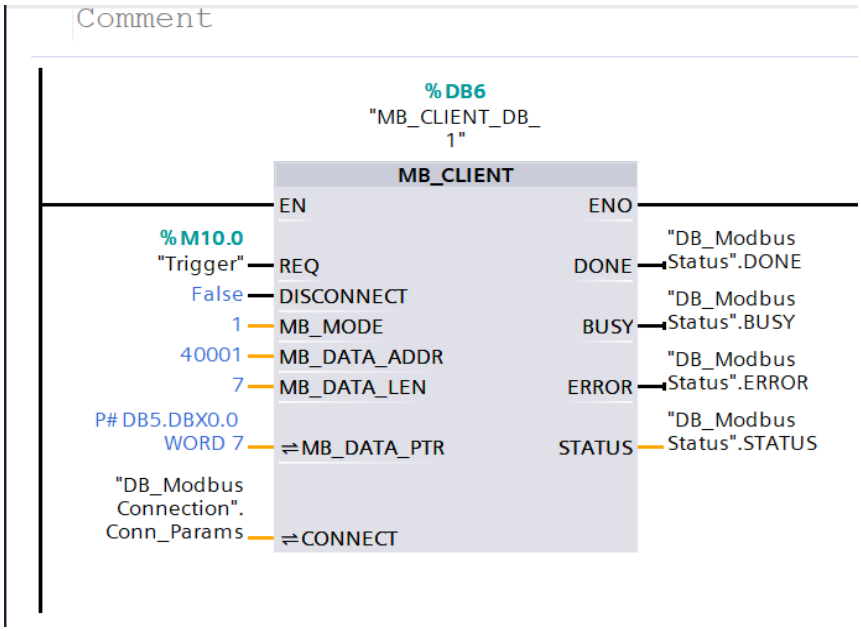


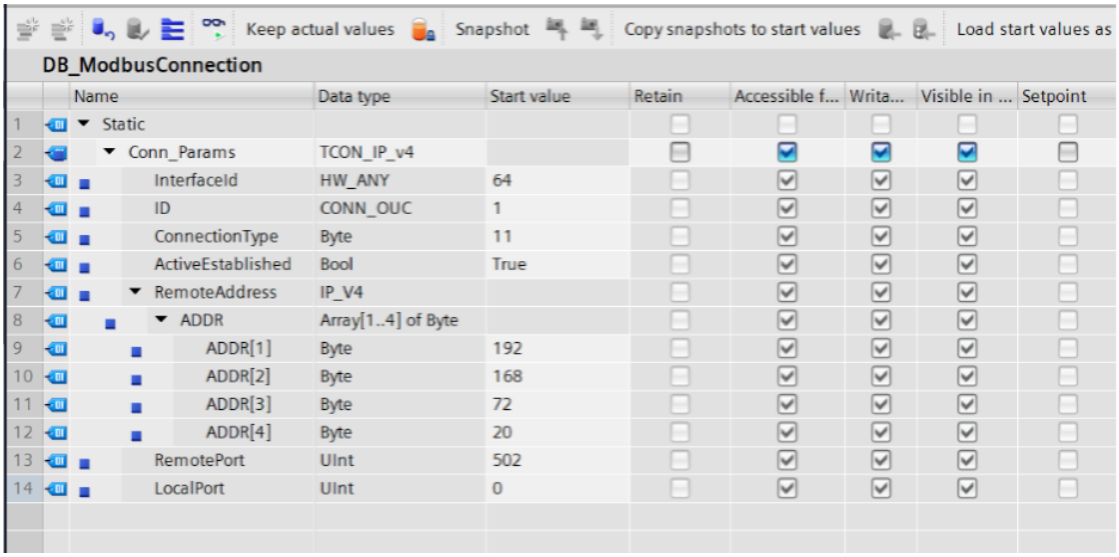
Figure 9. Configuration of the MB_*CLIENT function block within OB1 for Modbus communication between PLC and Arduino.*

In addition to initiating communication requests, OB1 also supervises the overall operation of the system by monitoring the status information returned by the MB_CLIENT block. This allows the PLC to determine whether communication has been completed successfully, whether a request is still being processed, or whether an error has occurred. OB1 functions as the supervisory control block of the application, ensuring that robotic joint angle values are continuously prepared, transmitted, and monitored through system operation.

4.5.2 DB_ModbusConnection [DB3]

The DB_ModbusConnection data block shown in Figure 10 was implemented to store all communication parameters required to establish and maintain the Modbus TCP connection between the PLC and the Arduino. This block contains the TCON_IP_V4 structure used by the MB_CLIENT function block to define the network communication settings. Since the PLC operated as the Modbus TCP client and the Arduino operated as the Modbus TCP server, the connection parameters stored in this data block defined how the PLC initiated communication with the Arduino.

The configuration includes the connection identifier, connection type, remote IP address, communication port, and other network-related parameters necessary for TCP/IP communication. The Arduino was assigned IP address 192.168.72.20 and listened for Modbus requests on the standard Modbus TCP port 502. The parameter ActiveEstablished was configured as True, to allow the PLC to actively initiate and maintain the communication session whenever the request is generated. By storing these parameters in this dedicated block, communication settings are managed separately from the main program logic; this improves the program's organization and simplifies troubleshooting procedures.



	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Static							
2	Conn_Params	TCON_IP_v4						
3	InterfaceId	HW_ANY	64					
4	ID	CONN_OUC	1					
5	ConnectionType	Byte	11					
6	ActiveEstablished	Bool	True					
7	RemoteAddress	IP_V4						
8	ADDR	Array[1..4] of Byte						
9	ADDR[1]	Byte	192					
10	ADDR[2]	Byte	168					
11	ADDR[3]	Byte	72					
12	ADDR[4]	Byte	20					
13	RemotePort	UInt	502					
14	LocalPort	UInt	0					

Figure 10. DB_ModbusConnection data block containing network parameters.

4.5.3 DB_ModbusStatus [DB7]

The DB_ModbusStatus data block was created to monitor and record the communication status of the Modbus TCP connection. It stores the output signals generated by the MB_CLIENT block and provides a convenient way to observe communication activity during system testing and debugging. As shown in Figure 11, this data block contains variables that indicate whether communication is currently in progress and whether an error has occurred. It also stores detailed status codes that provide diagnostic information.

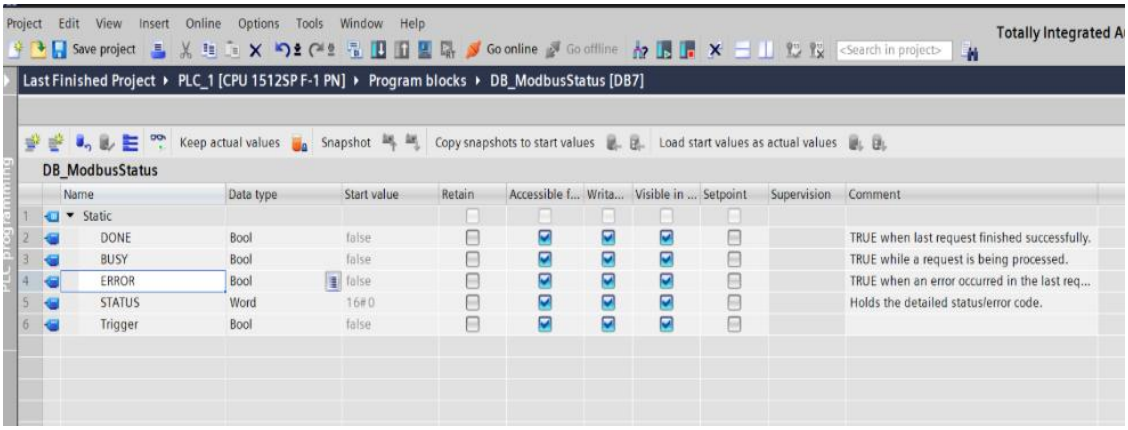


Figure 11. Configured DB_ModbusStatus data block containing communication statuses.

Throughout the development and testing phase of the project, DB_*ModbusStatus played a significant role in identifying communication issues between the PLC and the Arduino. The stored status variables such as DONE, BUSY, ERROR, STATUS, and Trigger are monitored in the MB*_CLIENT block in OB1. They enable real-time observation of communication behavior, indicating how the Modbus block reacts, diagnosing communication states, and verifying whether the Modbus connection has been established successfully.

4.5.4 DB_RobotAngles [DB5]

The DB_RobotAngles data block was created to store the target position values for the robot arm joints before transmission to the Arduino, as shown in Figure 12. This block

acted as the primary communication buffer between the PLC application and the Modbus TCP communication layer. Each variable within the data block represented a specific joint of the robot arm, while an additional command register was included to indicate when a movement operation should be executed.

The stored variables consist of six joint angle values corresponding to the robot base, shoulder, elbow, wrist roll, wrist pitch, and wrist yaw joints, together with a command register used to trigger the movement. These variables are directly mapped to the Modbus holding registers ranging from addresses 40001 to 40007. The Word data type was used to ensure compatibility with the 16-bit Modbus register format, while the data block was configured for non-optimized access to enable pointer-based addressing by the MB_CLIENT instruction.

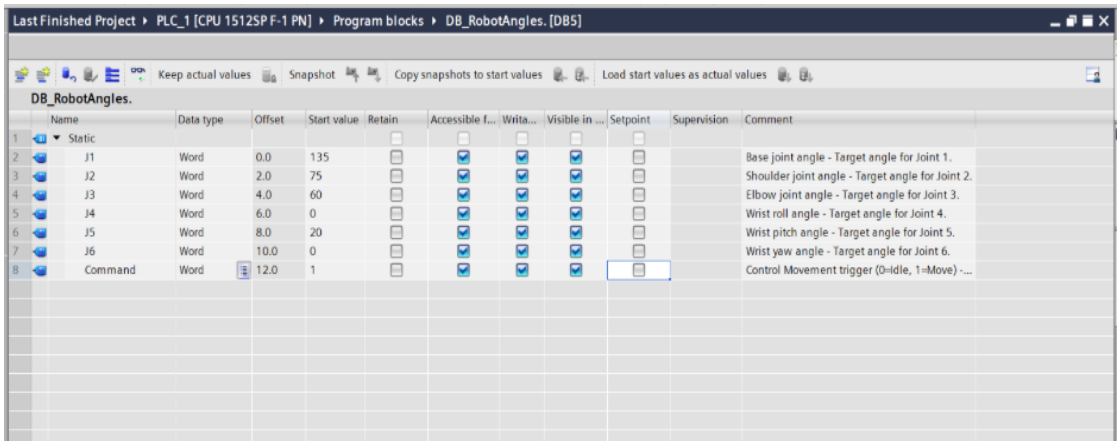


Figure 12. DB_RobotAngles data block containing robotic arm joint angle values.

During operation, the user can enter the desired joint angle values into the data block. The MB_CLIENT function block then accesses these values directly through a memory pointer and transmits them to the Arduino. As a result, the DB_RobotAngles data block serves as the central repository for all robot motion commands generated within the PLC.

5 Experiments and Results

This chapter presents the experiments conducted and the results obtained from the implementation of the PLC-based robotic arm control system. The primary objective was to verify the successful integration of the Siemens ET 200SP PLC, the Arduino Uno as an embedded system, and Waveshare RoArm-M2-S robotic arm through the Modbus TCP communication protocol. The project focused on evaluating PLC program execution, Modbus TCP communication, data transmission reliability, and the resulting robotic arm movements.

Before evaluating the communication performance of the implemented system, the operational status of the Siemens PLC was verified using TIA Portal. As shown in Figure 13, the PLC CPU was operating in RUN mode, indicating that the control program was executing correctly. The green status indicators on the configured hardware and beside the program blocks confirmed that the PLC was online and that the project had been successfully downloaded to the PLC and was operating correctly during system testing.

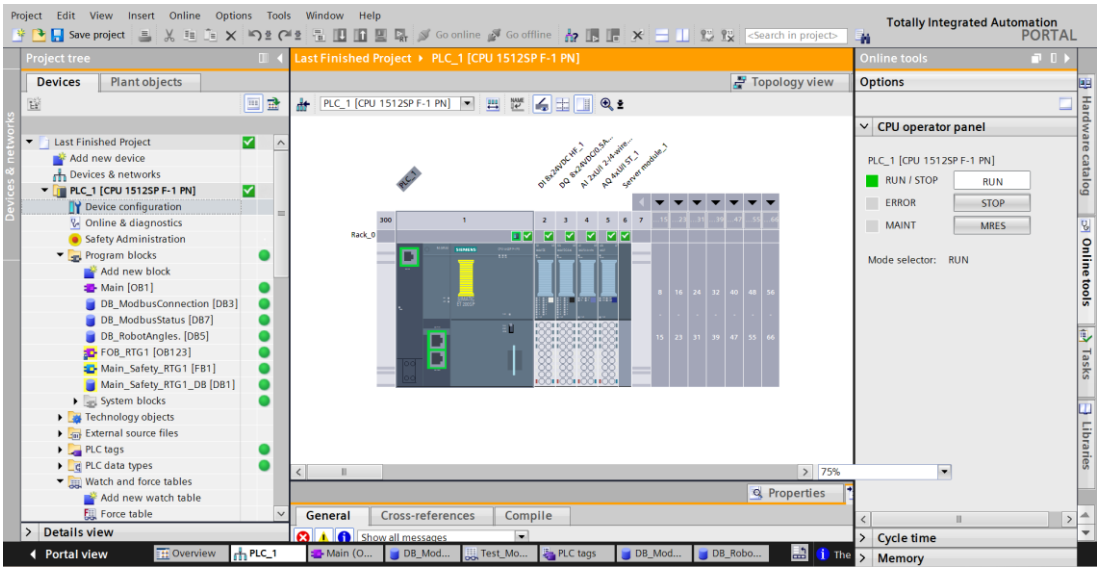


Figure 13. Online status of the Siemens PLC showing the CPU in RUN mode and successfully executed program blocks.

To evaluate the project's system performance, several testing procedures were carried out using TIA Portal online monitoring tools, Modbus communication diagnostics, Arduino Serial Monitoring, and physical observation of the robotic arm. The results obtained from these tests are presented and analyzed in the following sections.

5.1 Verification of PLC Program Execution

The first stage of testing involved verifying the correct execution of the PLC program developed in the TIA Portal. As discussed in Chapter 4, the Main Organization Block (OB1) continuously executed the MB_CLIENT function block to manage the transmission of robot joint angle values and movement commands from the PLC to the Arduino. Figure 13 presents the online monitoring view of the MB_CLIENT function block during system operation.

During online monitoring (Figure 13), the REQ input remained TRUE, indicating that the Trigger signal continuously initiated Modbus TCP communication requests from the PLC to the Arduino. The MB_CLIENT block operated in write mode (MB_MODE = 1), allowing

the PLC to write six robot joint angle values and one movement command from DB_RobotAngles (DB5) to the Arduino's Modbus holding registers, starting at register address 40001.

The communication status further verified correct system operation. The DONE output was observed as TRUE, confirming that the communication request had completed successfully. At the same time, BUSY remained FALSE, indicating that the communication cycle had finished, while ERROR remained FALSE, demonstrating that no communication faults occurred during data transmission. In addition, the STATUS register displayed the value 16#0000, which indicates successful Modbus TCP communication without errors. These observations confirmed that the PLC correctly transmitted the robot joint angle values and movement command to the Arduino through the configured Modbus TCP connection.

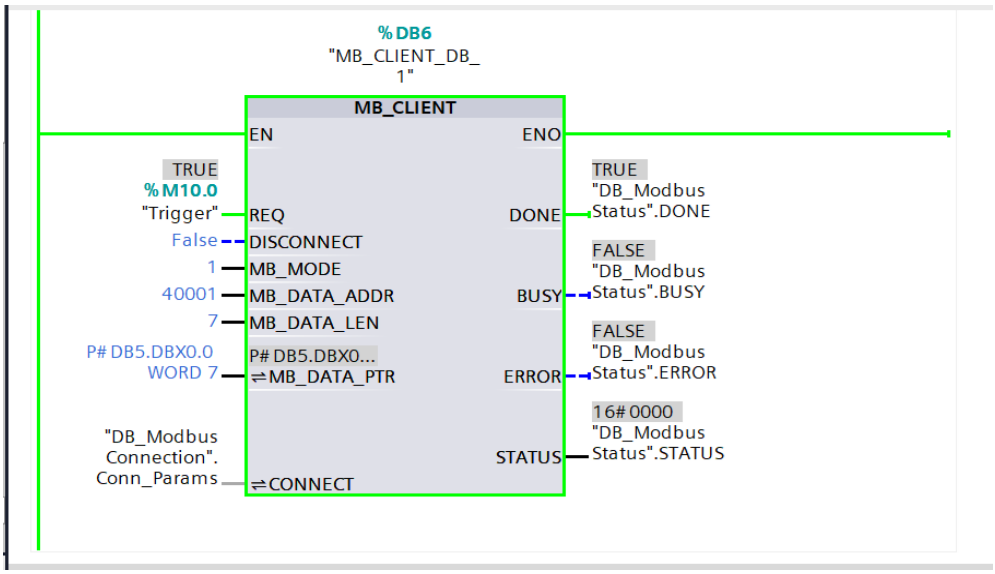


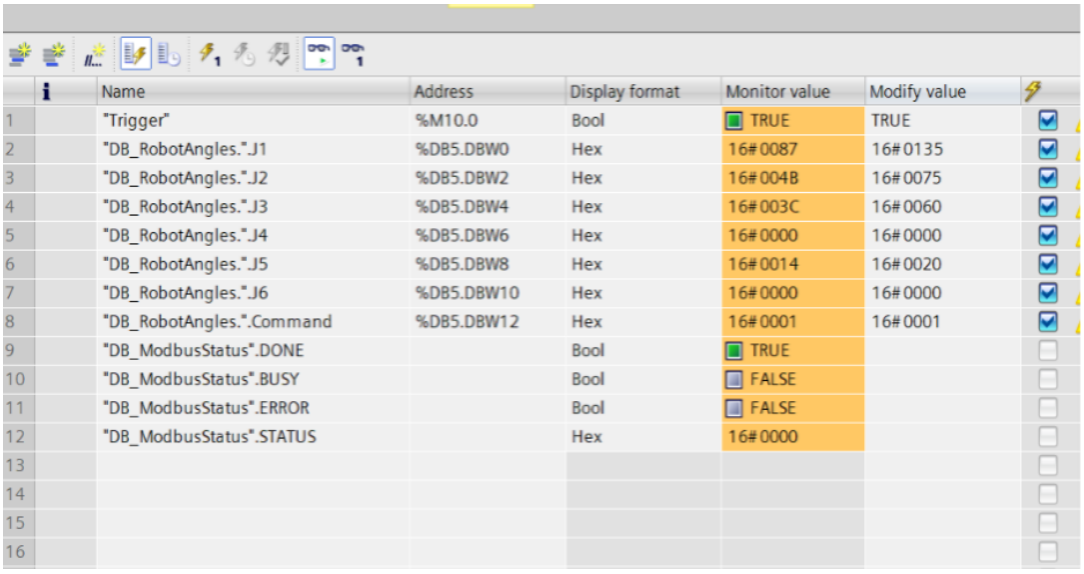
Figure 14. Online monitoring of the MB_CLIENT function block implemented in OB1.

5.2 Monitoring of Communication Status

To further evaluate the system performance, the Test_Modbus watch table was used to monitor the transmission of the robot joint angle command and the communication

status in real-time. Figure 15 shows the status conditions of TRUE or FALSE in the watch table during online operation.

The watch table confirmed that the Trigger signal was active (TRUE) and the values stored in DB_RobotAngles matched those observed within the data block. Furthermore, the communication status variables contained within DB_ModbusStatus provided important information regarding the state of the communication process. During online testing, the DONE status was observed as TRUE, indicating that the communication request had been completed successfully. At the same time, the BUSY monitor output was FALSE, indicating that the PLC was no longer processing a request. Furthermore, the ERROR monitor output remained FALSE, demonstrating that no communication faults were detected during the transmission process. The STATUS register displayed the value 16#0000, which indicates normal operation and the absence of communication errors. This provided strong evidence that the Modbus TCP communication between the PLC and the Arduino embedded system was functioning correctly. Therefore, the Test_Modbus watch table proved to be an effective diagnostic and verification tool for communication status and transmitted robot control data.



	Name	Address	Display format	Monitor value	Modify value	
1	"Trigger"	%M10.0	Bool	TRUE	TRUE	☑
2	"DB_RobotAngles.".J1	%DB5.DBW0	Hex	16#0087	16#0135	☑
3	"DB_RobotAngles.".J2	%DB5.DBW2	Hex	16#004B	16#0075	☑
4	"DB_RobotAngles.".J3	%DB5.DBW4	Hex	16#003C	16#0060	☑
5	"DB_RobotAngles.".J4	%DB5.DBW6	Hex	16#0000	16#0000	☑
6	"DB_RobotAngles.".J5	%DB5.DBW8	Hex	16#0014	16#0020	☑
7	"DB_RobotAngles.".J6	%DB5.DBW10	Hex	16#0000	16#0000	☑
8	"DB_RobotAngles.".Command	%DB5.DBW12	Hex	16#0001	16#0001	☑
9	"DB_ModbusStatus".DONE		Bool	TRUE		☐
10	"DB_ModbusStatus".BUSY		Bool	FALSE		☐
11	"DB_ModbusStatus".ERROR		Bool	FALSE		☐
12	"DB_ModbusStatus".STATUS		Hex	16#0000		☐
13						☐
14						☐
15						☐
16						☐

Figure 15. Online monitoring of the Test_Modbus watch table showing communication status and robot joint angle values.

5.3 Arduino Serial Monitor and Robot Arm Movement Results

After the Arduino program was uploaded and executed, the Serial Monitor displayed system initialization messages followed by the robot joint angle values received from the PLC. During testing, the Serial Monitor was used to verify the reception and processing of robot control data transmitted through the Modbus TCP communication channel. As shown in Figure 16, the displayed values correspond to the robot joint angles transmitted from the PLC and stored within the DB_RobotAngles data block. The display of these values confirmed that the Arduino correctly received and processed the transmitted data through the Modbus TCP network.

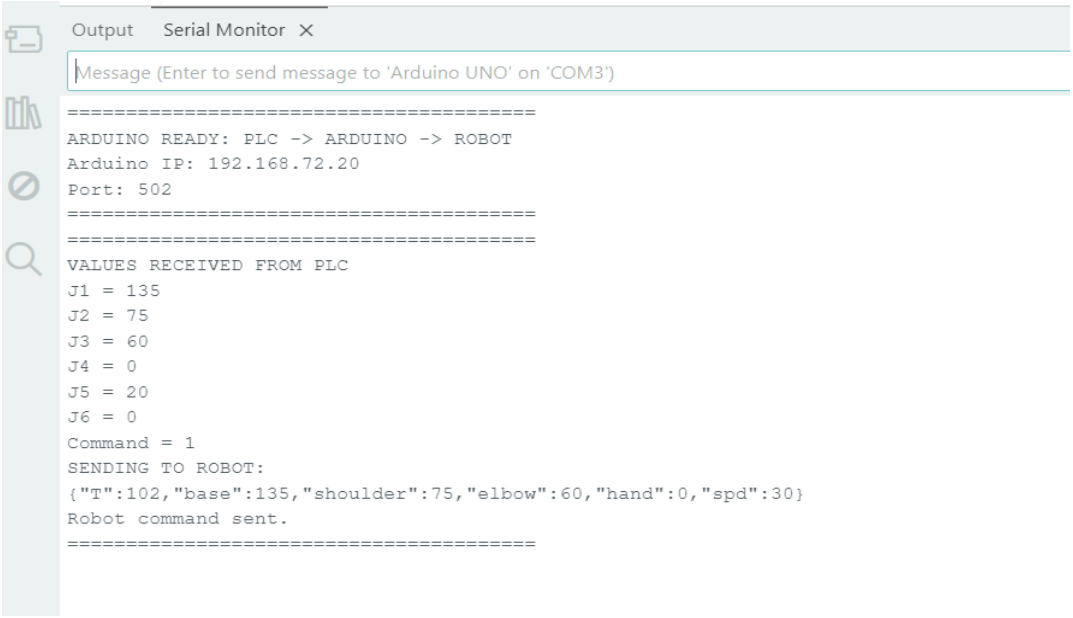


Figure 16. Arduino Serial Monitor displaying the robot joint angle values received from the PLC and the generated robot movement command transmitted to the robotic arm.

5.4 Observation of Robotic Arm Movement

The final stage of the experiment involved evaluating the physical response of the robotic arm to the commands generated by the PLC. As shown in Figure 17, the robotic arm was in its initial position before the PLC command was executed. This position served as the reference configuration for evaluating the robot's response to the transmitted joint angle values.

Figure 17. Initial position of the robotic arm before execution of the PLC command.

After the PLC transmitted the programmed joint angle values through the Arduino embedded system, the robotic arm successfully reached the final position shown in Figure 18. The observed change in the robot configuration confirmed that the transmitted joint angle values were correctly received, processed, and executed by the robotic arm controller. These results provide physical evidence that the developed communication architecture successfully transferred control commands from the PLC to the robotic arm through the Arduino embedded system.

Figure 18. Final position of the robotic arm after execution of PLC-generated joint angle commands.

6 Discussion

The aim of this project was to design, implement, and evaluate the integration of a Siemens PLC and an embedded system for industrial automation applications using Modbus TCP and UART communication technologies. The robotic arm was used as a practical case study to validate the developed communication architecture. The results presented in Chapter 5 demonstrated that the system successfully achieved the intended communication and control objectives.

The physical movement of the robotic arm provided practical validation of the entire system architecture. Comparison of the initial and final robot positions demonstrated that the transmitted joint angle values resulted in the expected robotic arm movements. These observations confirm that the communication chain from the PLC, through the Arduino, and finally to the robotic arm controller functioned correctly. Therefore, the automation system successfully demonstrated the integration of industrial control hardware and embedded systems technology within a single automation architecture.

6.1 Achievement of research objectives and research questions

Based on the experimental results obtained during system testing, the research questions identified in Chapter 1 were successfully addressed. Reliable communication was established between the Siemens PLC and the Arduino embedded system using the Modbus TCP protocol. The Arduino successfully operated as a communication gateway by receiving data transmitted from the PLC, processing the received joint angle values, and transmitting robot control commands to the robotic arm controller through UART serial communication.

The integration of Modbus TCP and UART communication technologies proved effective for transferring robot control data between the different system components. The successful reception of PLC data, generation of robot commands, and execution of robotic arm movements demonstrated the functionality of the developed communication

architecture. Furthermore, the absence of communication errors during testing indicated that the developed system provided reliable communication for industrial automation applications. Therefore, the research objectives were achieved, and the research questions were satisfactorily answered.

6.2 Challenges and Limitations

Although the developed system successfully achieved its objectives, several challenges were encountered during implementation and testing. One of the main challenges was establishing reliable Modbus TCP communication between the Siemens PLC and the Arduino, which required careful configuration of network parameters, Modbus addressing, and connection settings. Additionally, effort was required to ensure that robot control commands generated from the PLC data were correctly formatted and interpreted by the robotic arm controller.

The current implementation is also limited to one-way communication, where commands are transmitted from the PLC to the robotic arm without feedback being returned to the PLC. As a result, the PLC cannot directly monitor the actual robot position or operational status during execution. Furthermore, the system was tested using a single PLC, one Arduino, and one robotic arm. Although the results demonstrated successful communication and control, the performance of the proposed architecture in large industrial automation environments was not investigated within the scope of this project.

6.3 Significance of the Project

Despite the identified limitations, the results of this research demonstrate the viability of integrating industrial PLC technology with embedded systems for robotic control applications. The successful Modbus TCP communication between the Siemens PLC, the Arduino, and the robotic arm controller provides a practical approach for connecting industrial control systems with external robotic hardware. The developed architecture demonstrates how embedded systems can be used as flexible communication gateways

within industrial automation environments. Additionally, the project contributes to the growing adoption of Industry 4.0 concepts, embedded devices, and robotic systems within a unified communication framework.

7 Conclusion

This thesis presented the development, implementation, and evaluation of an integrated industrial automation system that combined a Siemens PLC, an Arduino embedded system, and a robotic arm. The developed system utilized Modbus TCP communication between the PLC and the Arduino and UART serial communication between the Arduino and the robotic arm controller.

The experimental results confirmed that robot joint angle values generated by the PLC were successfully transmitted to the Arduino, processed into robot control commands, and executed by the robotic arm. The results obtained from TIA Portal online monitoring, the Test_Modbus watch table, the Arduino Serial Monitor, and the observed robotic arm movement verified the correct operation of the developed communication architecture.

The findings demonstrate that embedded systems can effectively function as communication gateways between industrial controllers and external devices. Furthermore, the successful integration of Modbus TCP and UART communication technologies highlights their suitability for industrial automation applications.
Overall, the objective of this project was achieved, and the integrated system successfully demonstrated the integration of embedded systems and PLCs for industrial automation applications. The project architecture provides a practical approach for integrating PLCs, embedded systems, and robotic devices using standard industrial communication protocols. In addition, the results obtained from this study contribute to the growing adoption of Industry 4.0 technologies by demonstrating how PLCs, embedded systems, and robotic devices can be integrated into a unified automation framework. The system also provides a foundation for future developments involving feedback control, advanced robotic applications, and large-scale industrial automation systems.

References

Bolton, W. (2015). Programmable Logic Controllers (6th ed.). Newnes.

Christoulakis, F., & Thramboulidis, K. (2016). IoT-based Integration of IEC 61131 Industrial Automation Systems: The case of UML4IoT. In 2016 IEEE 25th International Symposium on Industrial Electronics (ISIE) (pp. 322–327). IEEE.

Craig, J. J. (2018). Introduction to robotics: Mechanics and Control (4th ed.). Pearson.

Folgado, F. J., Calderon, D., Gonzalez, I., & Calderon, A. J. (2024). Review of industry 4.0 from the perspective of automation and supervision systems: Definitions, architectures and recent trends. Electronics, 13(4), 782. https://www.mdpi.com/2079-9292/13/4/782

Hermann, M., Pentek, T., & Otto, B. (2016). Design principles for Industrie 4.0 scenarios. In 2016 49th Hawaii International Conference on System Sciences (HICSS) (pp. 3928–3937). IEEE.

Koondhar, M. A., Kalioi, G., Junejo, A. K., Soomro, A. H., Chandio, S., & Ali, M. (2023). The Role of PLC in Automation, Industry and Education Purpose: A Review. Pakistan Journal of Engineering, Technology and Science, 11(2), 22–31.

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., & Hoffmann, M. (2014). Industry 4.0. Business & Information Systems Engineering, 6(4), 239–242.

Mahalik, N. P. (2007). Fieldbus technology: Industrial network standards for real-time distributed control. Springer.

Minchala, L. I., Peralta, J., Mat-Quevedo, P., & Rojas, J. (2020). An approach to industrial automation based on low-cost embedded platforms and open software. Applied Sciences, 10(14), 4696. https://www.mdpi.com/2076-3417/10/14/4696

Modbus Organization. (2024). Modbus application protocol specification V1. 1b3.

Siciliano, B., & Khatib, O. (Eds.). (2016). Springer handbook of robotics. Springer.

Shylla, D., Shah, P., Sekhar, R., & Murugesan, D. (2023). Embedded systems in industrial automation 4.0. In 2023 14th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1–6). IEEE.

Siemens AG. (2024). SIMATIC STEP 7 and TIA Portal documentation. Siemens industry online support.

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2020). Robot modeling and control (2nd ed.). Wiley.

Thramboulidis, K. (2014). A framework for the implementation of industrial automation systems based on PLCs. https://arxiv.org/pdf/1402.3920

Wolf, M. (2022). Computers as components: Principles of Embedded Computing System Design (5th ed.). Morgan Kaufmann.

Xu, L. D., Xu, E. L., & Li, L. (2018). Industry 4.0: State of the art and future trends. International Journal of Production Research, 56(8), 2941–2962.

Zurawski, R. (Ed.). (2018). Embedded systems handbook. CRC Press.