Tartu University

Faculty of Science and Technology

Institute of Technology

Asier Mandiola Arrizabalaga

Haptic-Based Teleoperation of Robots

Master's thesis (30 EAP)
Robotics and Computer Engineering

Supervisors:

Karl Kruusamäe and Diego Gonzalez

Tartu 2025

Kokkuvõte

Haptilisel tagasisidel põhinev robotite teleoperaatorjuhtimine

Käesolev magistritöö on koostatud Tehniker keskuses, mis on Baskimaal asuv tehnoloogiauuringute keskus, ning see kuulub Autopilot-projekti koosseisu. Projekti eesmärk on luua robotisüsteem kirurgiliste rakenduste toetamiseks. Eesmärgiks on võimaldada ohutu ja intuitiivne teleoperatsioon haptilise tagasiside ja täpse liikumise juhtimise kombinatsiooni abil.

Töö kirjeldab kahepoolse teleoperatsioonisüsteemi arendamist ja juurutamist haptilise tagasisidega, kasutades programmeeritaval loogikakontrolleril (PLC) põhinevat juhtimisarhitektuuri. Süsteem integreerib võtmekomponendid, nagu robotmanipulaator, haptilise liidese seade (Omega.7), jõumõõtmine ja reaalaja suhtlus. Haptilise seadme ja roboti juhtkontrolleri vaheline andmevahetus toimub mitteterministliku suhtlusprotokolli (ADS ja UDP) kaudu PLC abil.

Esimene oluline saavutatud tulemus oli roboti edukas teleoperaatorjuhtimine haptilise seadme abil, mis võimaldas manipulaatori sujuvat ja täpset juhtimist. Pärast seda keskenduti jõutagasiside integreerimisele, et sulgeda kahepoolne juhtimisahel. Eksperimentaalsed tulemused kinnitasid, et jõuandmed omandatakse, töödeldakse ja skaleeritakse korrektselt, et pakkuda operaatorile realistlikku haptilist tunnetust. Kommunikatsiooni latentsus osutus kriitiliseks teguriks süsteemi stabiilsuse seisukohalt ning see leevendati edukalt protokolli optimeerimise ja ülesannete ajastamise kaudu. Üldine süsteem osutus töökindlaks, modulaarseks ja kohandatavaks.

Süsteemi arhitektuur on täielikult valmis füüsilise Omega.7 seadme integreerimiseks ning on kavandatud katsetamiseks realistlikes tingimustes tulevikus, potentsiaalsete rakendustega kirurgilise simulatsiooni või kaugjuhitava roboti manipuleerimise valdkondades.

CERCS: T120 Süsteemitehnika, infotehnoloogia; T125 Automaatika, robootika, juhtimistehnika.

Võtmesõnad: teleoperatsioon, haptilisus, robootika, tööstusarvuti, jõutagasiside, reaalaja juhtimine, determinism, suhtlus.

Abstract

Haptic-Based Teleoperation of Robots

This thesis has been developed at Tekniker, a technological research center located in the Basque Country, as part of the Autopilot project, which is focused on designing a robotic system to assist in surgical applications. The objective is to enable safe and intuitive teleoperation through a combination of force feedback and precise motion control.

The work presents the development and implementation of a bilateral robot teleoperation system with haptic feedback using a PLC-based external control architecture. The system integrates key components such as a robotic manipulator, a simulated haptic device (Omega.7), force sensing, and real-time communication. A non-deterministic communication protocol (ADS and UDP) is used to enable data exchange between the haptic device and the robot controller via a PLC.

The first milestone was the successful teleoperation of the robot using the haptic device simulation as a master interface, enabling smooth and accurate control of the slave manipulator. Once this was achieved, the integration of force feedback was addressed to close the bilateral control loop. Experimental results confirmed that force data is correctly acquired, processed, and scaled to provide realistic haptic sensations to the operator. Communication latency was identified as a critical factor affecting system stability and was successfully mitigated through protocol optimization and task scheduling. The overall system proved to be robust, modular, and adaptable.

The architecture is fully prepared for the integration of the physical Omega.7 device and is intended to be tested in realistic environments in future work, with potential applications in areas such as surgical simulation or remote robotic manipulation

CERCS: T120 Systems engineering, computer technology; T125 Automation, robotics, control engineering.

Keywords: teleoperation, haptics, robotics, industrial PC, force feedback, real-time control, determinism, communication.

Contents

List of Figures

Abbreviations, Constants, and Terms

ACS - Axis Coordinate System

ADS - Automation Device Specification

AMS - TwinCAT Messaging System

API - Application Programming Interface

CPU - Central Processing Unit

DMA - Direct Memory Access

DOF - Degrees of Freedom

EXT - External (Axis or Reference)

I/O - Input/Output

IPC - Industrial PC

MCS - Machine Coordinate System

MIS - Minimally Invasive Surgery

N - Newton (unit of force)

NC - Numerical Control

PC - Personal Computer

PLC - Programmable Logic Controller

RT - Real-Time

SDK - Software Development Kit

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

USB - Universal Serial Bus

XAE - TwinCAT Automation Engineering

XAR - TwinCAT Automation Runtime

1 Introduction

Since ancient times, humans have used tools to extend their manipulative capabilities and facilitate tasks. Examples of this include the use of sticks for fruit picking and blacksmiths' tongs to manipulate incandescent parts, avoiding direct contact with dangerous objects. Today, thanks to technological advances, many of these tasks have been taken over by master-slave teleoperation systems. These systems allow a "secondary" manipulator to replicate the movements of a remotely controlled "primary" (9), improving safety and precision in complex tasks. Since their inception, teleoperation systems have advanced significantly, finding applications in fields ranging from entertainment to disaster response and medical surgery (10). A clear example of teleoperation is the use of a robot in remote surgery, where the surgeon controls the robot's movements remotely, executing precise actions without being physically in the operating room (11).

Robots, essential in teleoperation, enable tasks in hazardous environments, such as space exploration (12) and industrial manipulation (13), increasing efficiency and reducing risks. However, teleoperation depends not only on the operator's skill, but also on a robust architecture that integrates robotic control with real-time communication between the robot and the manipulator. Traditional industrial robot architectures often rely heavily on proprietary control platforms and vendor-specific programming environments (14), which limit scalability, interoperability, and adaptability. As robotic environments demand higher levels of customization, these rigid systems struggle to integrate new functionalities, external devices like sensors or actuators, or coordinate operations across multi-vendor systems (15). Furthermore, limited access to the internal control structure hampers the implementation of advanced strategies and real-time responsiveness, increasingly required in dynamic and collaborative scenarios (16; 17). In this context, advanced external control systems, such as PLCs, enhance overall integration. (18)

Low-level control through external systems like PLCs offers tangible advantages by enabling direct access to motion planning elements such as kinematics and trajectory generation (19). Unlike traditional systems, where these elements are locked within proprietary controllers, PLC-based setups provide granular control over motion execution (20), opening the door to real-time adjustments and custom behaviors.

A key benefit of this approach is the ability to manage position setpoints at the actuator level. Sending commands directly to each axis allows for more refined motion control and tailored robot behavior. Additionally, by providing setpoints prior to internal processing, the system can implement real-time compensation for misalignments or deviations—critical for applications requiring high precision-, such as manufacturing or surgery. In summary, PLC control introduces a more open and adaptable framework for robot operation, offering both detailed trajectory manipulation and improved system responsiveness.

Still, one of the biggest challenges in robot teleoperation is the lack of feedback (21), especially kinesthetic or force feedback, since the human experience of manipulation is directly related to the sensory information the operator receives. Therefore, in teleoperation, the absence of this feedback can make robot manipulation less intuitive and accurate, as the operator cannot know exactly what forces he/she is applying in the operation nor the forces or resistances he is experiencing due to the interaction with the environment (22). This becomes a major challenge in tasks that require high precision.

To address this challenge, haptic devices provide sensory feedback to the operator, reproducing the forces and tactile sensations of manipulation (23). These devices allow the operator to perceive the forces applied during interaction with the robot, improving control accuracy and intuitiveness (24).This is especially useful in tasks that require high precision and responsive real-time feedback.

For enhancing teleoperation performance and experience, the integration of haptic feedback with PLC-based control systems is a powerful solution. The PLC, with its precise control over robot motion, ensures that the robot follows accurate trajectories and responds promptly to commands. Meanwhile, the haptic device transmits real-time feedback to the operator, providing a sense of touch and reaction force. This combination allows for a more intuitive and controlled interaction with the robot, improving both the operator's perception and the precision of robot manipulation. The reliability and speed of communication between the master (haptic device) and slave (robot) are critical to the effectiveness of this system. Any delay or instability in this communication loop could lead to a mismatch between the operator's input and the robot's movement, compromising the overall performance.

In this document, a robot teleoperation system based on haptic technology will be studied and implemented, with the aim of improving the operator's precision and interaction with the robot. Through this system, the operator will be able to control the robot precisely using emulated the haptic device, while receiving real-time sensory feedback, transmitted by the robot's force sensor. This feedback will allow realistic replication of the interactions between the robot and its environment, giving the operator the most authentic possible feeling of the manipulation being performed. The robot will be controlled by a Beckhoff PLC, using TwinCAT3 software, which will communicate with the Staubli RX160 robot via uniVAL Drive. The system is designed to incorporate the OMEGA.7 haptic device, which would transmit motion commands and receive forces measured by the robot's force sensor. Additionally, a force sensor will be used to read the forces applied during interaction, enabling precise and fluid feedback. A key aspect of this setup is the communication between the master and slave devices, as it directly impacts responsiveness and system stability. For this reason, the document will also focus on communication protocols, latencies, and real-time performance.

However, due to project constraints, the actual Omega.7 haptic device was not available in time. To overcome this, a code-based prototype was developed in Visual Studio to emulate the behavior of the Omega.7. This prototype simulates the trajectory data that would be received from the device, as well as the force feedback readings that would typically be sent from the PLC to the Omega.7. Although this is not a full physical simulation, it enables early testing of the communication interface and overall system architecture. By using this virtual environment, we ensure that the software and PLC logic function correctly before integrating the real hardware.

1.1 Objectives and Research Roadmap

The main objective of this master's thesis is to study, develop and implement a robotic teleoperation system based on haptic devices, through which an operator can control a robot remotely and in real time using a haptic device, receiving force feedback to achieve a more realistic and accurate manipulation. Although this study uses the Staubli RX160 robot, the emulated OMEGA.7 haptic device and an external control using a Beckhoff PLC with TwinCAT, the proposed methodology is generalizable and can be adapted to other systems.

The development of this work followed a structured roadmap. First, the state of the art in external robotic control systems was studied, with the goal of designing a modern control architecture aligned with recent advancements in robotics. This was followed by an analysis of the state of the art in haptic devices, focusing on their characteristics, capabilities, and integration potential. Finally, the state of the art in haptic-based robotic teleoperation was reviewed, a field in constant evolution and rich with opportunities for research and innovation.

Next, the development and implementation of the two fundamental parts of the system began. On the one hand, the PLC-Robot integration, which allows the external control of the robot in real time through TwinCAT 3. Once this stage was completed, we proceeded to the implementation of the haptic device side, which involved the development of a desktop application to interact with the OMEGA.7 API and establish communication with the PLC, thus closing the flow of the OMEGA.7-PLC-Robot system.

Finally, the results of the developed system have been obtained and analyzed, drawing conclusions about its performance. This whole process is described in detail in this document.

2 State of the Art

2.1 External Control of Robots

Robotics is undoubtedly one of the technologies that has advanced the most in recent decades, driven by its advantages in handling heavy or hazardous materials and operating in environments that pose risks to humans. In addition to its traditional role in industry—particularly in the automotive and electronics sectors (25)(26)—robotics has recently expanded into fields such as medicine, where precision and safety are critical. Since the 1970s, technological progress has positioned industrial robots as key components of modern smart factories, capable of tasks like manipulation, picking and inspection operations for assembling and palletizing applications (27). However, despite their mechanical versatility, these systems often rely on closed control architectures that limit adaptability.

As manufacturing and, specially, medical applications increasingly demand real-time responsiveness and high accuracy, the complexity of robotic control systems has grown requiring flexible programming and low-latency communication to ensure safe and reliable operation (28). Additionally, these systems must adhere to strict regulations and standards specific to medical applications, which often include requirements related to safety, electromagnetic compatibility, and software reliability (29) (30). Although not detailed here, it is worth mentioning that such systems are expected to meet "medical grade" criteria, which imply rigorous certification processes and compliance with health-related norms.

Industrial robots are traditionally controlled by robust internal systems housed within the robot cabinet. These systems include vendor-specific programming environments and lenguages that handle tasks such as path planning and trajectory generation. The resulting motions are executed by servo drives through closed-loop control, using feedback from encoders to ensure precision and stability. However, in more complex setups—especially those involving multiple subsystems or requiring advanced interaction logic—PLCs are commonly integrated into the architecture (14). PLCs allow for high-level coordination, making it possible to synchronize a robot arm with external elements such as haptic devices and force sensors in real time, as is the case in our application.

However, the traditional rigid method limits robots' performance in modern, reconfigurable systems.The limitations of programming languages arise from the absence of a universal standard, which makes integrating robots from different manufacturers challenging. This lack of standardization results in different programming languages for each manufacturer, increasing complexity and requiring specialized personnel. Additionally, limited motion control in traditional robotic programming restricts the ability to execute complex robot paths, with most systems only supporting basic movements. This constraint significantly affects precision tasks, where intricate paths and continuous motion are often required (1).

For applications that require high precision, robotic control architectures have been developed and studied that differ from traditional ones, which do not provide access to the robot controller for managing axis-level setpoints, kinematics, trajectory generation, or other low-level control aspects. This limitation reduces the operator's ability to optimize the robot's precision. These alternative architectures can be categorized into three groups, as described in (1):

In the "External Coordinator" architecture, an external device (such as a PC or PLC) sends commands written in a standard programming language. The robot controller runs a software interpreter that translates these commands into its native instruction set. The controller then performs trajectory planning internally, similar to the traditional architecture.

In the "External Trajectory Generator" architecture, trajectory planning is handled by an external device (e.g., an industrial PC or real-time PLC). Motion profiles are sent to the robot, and feedback is received through fast-cyclic communication via the I/O module. The robot controller does not handle trajectory generation but may still process logical instructions through its non-real-time components.

Through methodologies such as "Open Controller", allows centralizing the generation of setpoints and trajectories independently from the original robot controller.

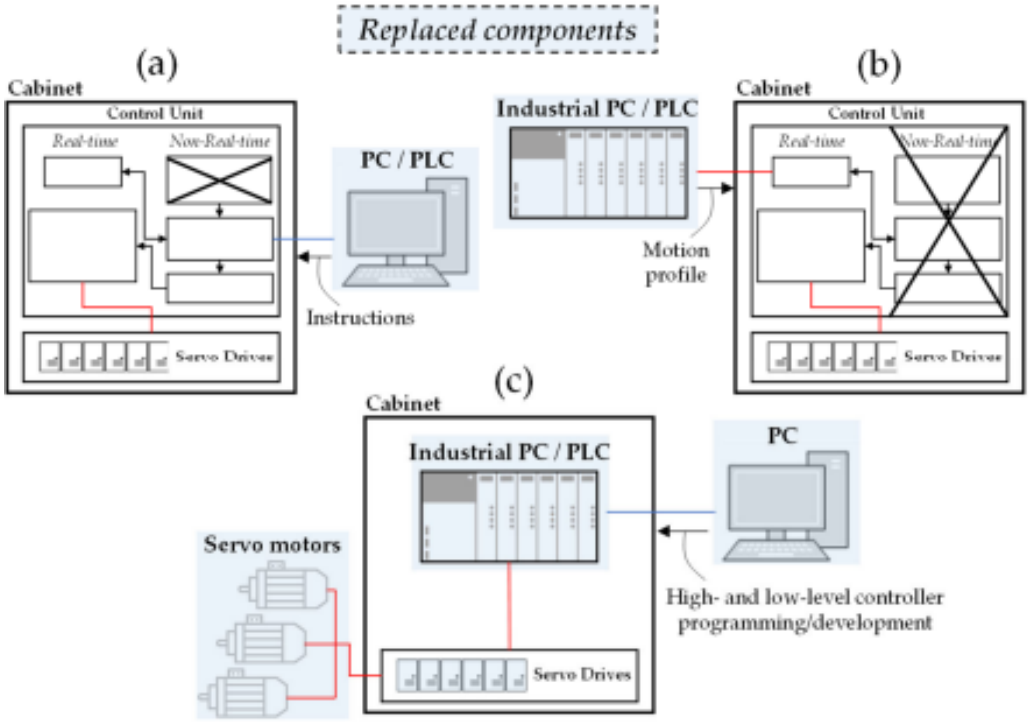

Figure 2.1: Schematic of the innovative robot control approaches. (a) External coordinator, (b) External trajectory generator, and (c) Open controller. (1)

Real-time execution is essential for achieving precise and responsive robot control. This requirement is fulfilled by PLCs that operate on real-time capable kernels. For example, TwinCAT3 runs on top of a real-time extension of the Windows operating system, allowing deterministic task execution, synchronized I/O handling, and integration of complex motion control routines such as kinematics and trajectory interpolation (31). To implement axis-level control of industrial robots, manufacturers provide dedicated real-time interfaces that enable external

systems to communicate with the robot controllers. These interfaces can allow either direct control of the drive units or the transmission of motion setpoints to the internal control loop. Examples include Staubli's uniVAL Drive and KUKA's RSI, which facilitate integration into external PLC-based or PC-based control architectures (20)

In this context, 'axis level' refers to the control level where commands are sent directly to the actuators of the robot's joints. Each joint of a robot arm is typically driven by an actuator (such as a motor), and 'axis level' control allows for the specification of position, velocity, or force setpoints for each individual actuator. This enables the robot's movement to be controlled with a high degree of granularity, directly influencing the physical motion of the joints, rather than relying on higher-level trajectory or motion generation algorithms. 'Real-time' means that the commands sent to the actuators are transmitted using real-time communication protocols, ensuring deterministic communication. This ensures that data packets are sent at precise time intervals, guaranteeing that control signals are delivered with minimal latency and synchronization.

In the context of trajectory planning and generation, NC has been widely adopted in traditional machine tools, particularly in machining centers (32) (33). Due to the large size and high cost of such machines, robots—although generally less precise—are increasingly being considered as viable alternatives. As a result, modern automation software often integrates NC functionalities to manage complex motion tasks. Moreover, in order to enable external control of the robot, it is essential that automation software provides dedicated NC libraries. These libraries allow advanced motion control from external systems, making it possible to execute precise path planning, apply kinematic transformations, or even implement adaptive control strategies outside of the robot's internal controller.

In the article on NC controllers (19), an external controller implementation based on NC technology is presented. This setup allows a 6-degrees-of-freedom industrial manipulator to be driven by an external NC controller, enabling the integration of advanced functionalities such as kinematic transformations and Adaptive Impedance Control. These examples highlight how external axis-level control can significantly enhance the accuracy and performance of robotic systems, particularly in applications requiring high precision.

In this paper (20), the use of laser tracking systems to compensate real-time positioning errors in industrial robots is presented. These systems measure the TCP position in 6 degrees of freedom, allowing for dynamic correction of joint position setpoints. Crucially, this compensation is achieved by interfacing with the robot through an external controller, which accesses the robot's lowest control level to apply precise setpoint adjustments. This architecture enables the implementation of advanced control strategies beyond what the robot's native controller typically allows. As a result, the study demonstrated a significant improvement in positioning accuracy—from 0.6725 mm to 0.0268 mm—representing a 95.7% reduction in error.

The article (18) validates the RTRobMultiAxisControl framework using the DextRobC platform, focused on Industry 4.0. It presents a complex robotic cell with the RoBioSS dexterous hand, designed for fine manipulation and mounted on a 6-axis Stäubli robot, totaling 22 actuated axes. The robot is powered via a uniVAL drive, allowing integration with external controllers. A B&R industrial PC and POWERLINK ensure real-time communication. The framework synchronizes all axes with high precision, sending commands to the fingers every 0.4 ms and to the robot every 2 ms, proving its effectiveness in multi-robot coordination.

2.2 Haptics Technology

Haptics, or haptic technology, refers to a set of technologies that create a sense of touch by applying forces, vibrations, or motions to the user. It is used to simulate virtual objects in computer environments, to interact with those objects, and to enhance the remote control of machines and devices (telerobotics) (34).

In recent years, the rapid advancement of this technology has led to exponential growth in the development of haptic devices (35). As a field with significant room for innovation, haptics has gained increasing importance. One example of its accelerated development is the wide range of applications it supports. These range from the entertainment industry to specialized medical devices, wearable gloves, and surgical procedures (35) 2.2.

Figure 2.2: Haptic applications. HaptX system (2), Industrial exoskeleton (3), Force Dimension haptic device (4).

Haptic devices typically consist of actuators (like motors), interface mechanisms, and sensors. Haptic feedback can be tactile (cutaneous) or kinesthetic (proprioceptive) (22; 36), and sometimes includes thermal feedback (37). Tactile feedback originates from mechanoreceptors on the skin, while kinesthetic feedback arises from sensors in muscles and joints, providing a sense of movement and force. Interaction with a haptic device is inherently bidirectional: the user moves the device, sensors detect motion, and force feedback is sent back via actuators. This closed-loop system enhances telepresence and user interaction in real, remote, or virtual environments (24).

Haptic devices allow users to interact with a wide variety of environments, creating a sense of realism and tangibility. These devices operate based on haptic feedback, which is generated by force or torque responses from objects located in real, teleoperated, or virtual environments. As defined in (38), the field of haptics can be organized into three areas: Human Haptics, Machine Haptics, and Computer Haptics.

The Human Haptic system integrates mechanical structures, sensory input, and motor control. Human hand, with 22 degrees of freedom and various receptors in the skin, muscles, and tendons, detects both contact (tactile information) and movement (kinesthetic information). These signals are quickly processed to adjust actions, such as increasing grip when an object starts to slip.

Machine haptics refers to the use of machines to replicate or augment human touch, either autonomously, via telerobotics, or through haptic interfaces. It involves three core operations: measuring positions or contact forces from the human body, processing this information, and displaying corresponding positions and forces back to the user.

Computer haptics focuses on generating and rendering the sense of touch in virtual environments using algorithms and software architectures. Unlike computer graphics, which provide only visual rendering, computer haptics creates forces, torques, and tactile sensations. Since visual interfaces cannot deliver physical feedback, interaction with virtual environments relies on devices such as joysticks, robotic arms, and actuators. Common feedback types include force, vibrotactile, and electrotactile systems, with most haptic devices primarily using force and vibrotactile feedback. (39)

Among haptic devices, there is a wide variety, each offering different types of feedback and varying degrees of freedom. Examples include the Virtuose™ 3D Desktop (40), which provides haptic force feedback with 3 to 6 degrees of freedom; the Omega 7 (41), which offers 7 degrees of freedom and haptic force feedback; and the SenseGlove Nova 2 (42), a haptic feedback glove with arm exoskeletons that offers 30 degrees of freedom. It uses magnetic friction brakes to simulate the size and stiffness of virtual objects. The glove features four brakes, one for each finger from the thumb to the ring finger, each capable of applying up to 20N of force—approximately the weight of a 2 kg brick per finger—to deliver highly realistic force feedback.

2.3 Haptic-based robot teleoperation

Significant advances in robot safety and precision, driven by different control architectures and improved drives, robotic teleoperation, although a concept that emerged in the 20th century, has gained much more relevance today. A clear example of the application of teleoperation is in the construction sector, where crane operators can manipulate blocks of stone, for example, using devices such as joysticks from the cab, without the need to physically do so.

The improvements brought by the previously mentioned haptic devices, which have made a strong entry into the field of robotic teleoperation (11). Alongside advances in robotics, teleoperation has gained great importance in medical applications, ranging from early developments in prosthetics and assistive devices for people with disabilities to innovative telesurgery procedures (43). This technological niche is one of the most distinctive areas within robotic teleoperation based on haptic devices. One prominent example of robotics in medicine is MIS (44). The main motivation behind the rapid progress of this technology lies in its benefits to patients, including reduced trauma, shorter hospital stays, and faster recovery times. This technology comes from the need to carry out minimally invasive procedures with greater precision and efficiency.

A telerobotic system typically includes a leader robot, a follower robot, and a communication network connecting them. In surgical applications, the leader robot is the surgeon's console, while the follower robot operates next to the patient (45). The leader captures the surgeon's commands—such as instrument positions, orientations, velocities, and camera movements—and transmits them through the network. The follower robot then replicates these motions using manipulator arms equipped with laparoscopic tools and an endoscopic camera, enabling precise remote surgical procedures.

While most current surgical robotic systems improve visual feedback for surgeons, they often lack haptic feedback (46) (47). This is a limitation, as haptic feedback provides essential information about tissue properties such as stiffness, depth, location, size, and texture. It also helps surgeons sense the amount of force applied, which is crucial for safe interactions with delicate tissues, such as when handling tumors. The absence of haptic feedback may increase procedure time, hinder palpation, and increase the risk of tissue damage (48). From a technical standpoint, the lack of force feedback (i.e., kinesthetic haptics), combined with limitations in system response speed and resolution, can significantly reduce the surgeon's ability to perceive subtle interactions with tissue.

One of the most well-known examples in surgical robotics is the da Vinci system, developed by the company Intuitive Surgical 2.3. Up until its latest version, the da Vinci 5, which was approved in March 2024, force feedback was not integrated (5). Therefore, the state of the art in force feedback for surgical robotics is very recent.

Figure 2.3: Da Vinci S robotic system, an earlier version preceding the Da Vinci 5 (5)

These factors directly impact the sensitivity and fidelity of the teleoperation loop, making it difficult to detect variations in tissue stiffness or texture, which are crucial for safe and efficient manipulation. This is why surgeons using robotic systems undergo intensive training to master the required motor skills and adapt to the absence of haptic feedback. They often rely on visual cues—developed through experience—to estimate the force applied, a method known as visual haptics.

Augmented force feedback improves minimally invasive surgery, benefiting surgeons labour. This article (49) quantifies the improvement based in this operation. Although haptic systems are intuitive, a multisensory approach is suggested. In this paper (50), the design and control of the sigma.7 haptic device and the new surgical console of the MiroSurge robotic system, which integrates a force/torque sensor for improved transparency, are presented. The system has been experimentally validated. Another paper addresses the planning and control software of a teleoperation system for minimally invasive robotic surgery research, providing intuitive bilateral

teleoperation within the planned workspace (45).

Apart from the numerous studies on haptic feedback-based robotic teleoperation, there is also research that looks in detail at how this force feedback should be received and what factors should be considered when interpreting forces on laparoscopic instruments. This study examines the design of a laparoscopic simulator, focusing on the parameterization of a force feedback system (22). Also how interactions between organs and surgical instruments influence the surgeon's haptic perception are investigated (51).

Lastly, and by no means less important, in robotic teleoperation applications, reliable and efficient communication between the master and slave devices is critical. Issues such as latency, data transmission delays, and packet loss can significantly impact system performance and stability cite(52). Therefore, careful consideration must be given to the communication protocols and middleware used within the overall system architecture.

This document (18) emphasizes that aspect, highlighting, for example, a study in which a ROS-based architecture was compared to a new setup using an industrial PC running a real-time operating system (RTOS) and hard real-time Ethernet communication via POWERLINK. The latter, based on an Atom CPU with the VxWorks operating system, demonstrated robust communication with no sensor data loss and maintained a consistent 2 ms cycle time. These results meet the temporal determinism requirements essential for axis control in high-precision environments.

3 System Design, Implementation and Integration

The purpose of this section is twofold. First, the subsection titled ”General System Architecture” aims to describe the design of the teleoperation system, including its main components, elements, and characteristics. Second, the subsection titled ”Implementation” explains the two implementations that have been developed: one on the PLC side—i.e., the implementation carried out on the PLC to control the robot—and the other in the C++ program, which enables communication between the haptic device and the robot. Together, these implementations complete the full teleoperation system

3.1 General System Architecture

Due to the size and characteristics of the system to be developed, the overall architecture design is based on two main parts 3.1. The first one, developed initially, is responsible for the external control of the robot by the PLC, i.e., the PLC-Robot architecture. This section will provide a general overview of the architecture, detailing its key elements and their characteristics. It will describe the real-time axis-level control of the Staubli RX160 robot using UniVAL Drive and TwinCAT3 automation software, and explain the integration with the force/torque sensor to use its measurements as feedback in the haptic system as well as the communication protocols that are used. The second part focuses on the Haptic Device architecture, which includes a detailed description of the OMEGA.7 device, its elements, characteristics, and API. This section will also outline the general structure of the C++ program that will act as a bridge between the two architectures.

By structuring the architecture this way, each component can be developed independently before being integrated into a fully operational system.

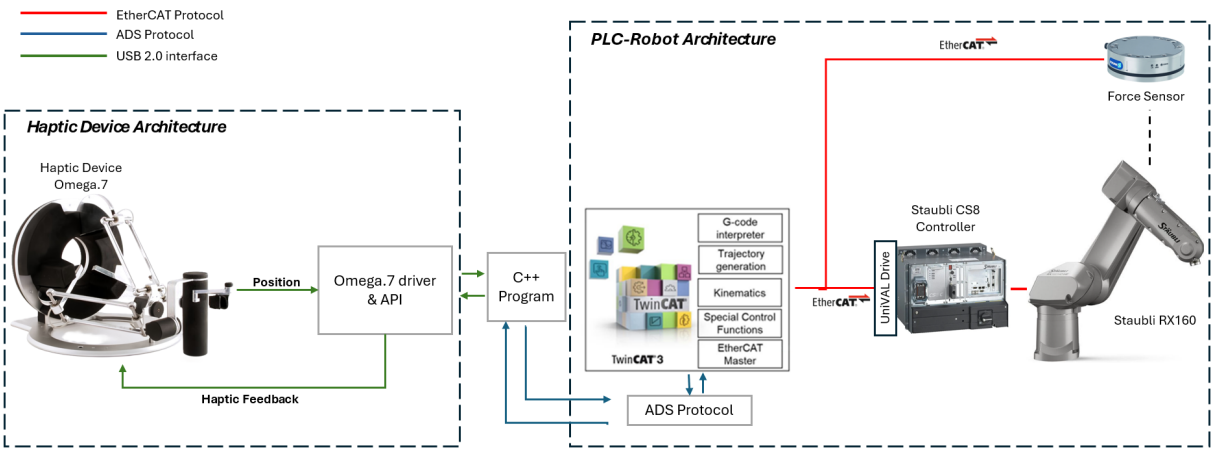

Figure 3.1: General System Architecture Scheme

3.1.1 PLC-Robot Control System

The PLC-ROBOT architecture developed in this Master Thesis is based on two main elements: a PLC centric approach, using a BECKHOFF controller with complementary NC functionalities, and the robot with its corresponding controller 3.2. In an industrial robotic system, the control system is the core, responsible for kinematic calculations, motion planning and transmission of instructions to the actuator.

The accessibility to the control system is key to the overall robot performance. With this PLC–ROBOT architecture, the robot control system is fully integrated into the PLC. Once the robot trajectory is defined, the PLC executes all tasks typically handled in a conventional architecture and sends position setpoints to the robot controller. In this configuration, the PLC becomes responsible for the entire system control: it not only handles robot motion by sending commands to the controller, but also receives sensor readings (e.g., from force/torque sensors) and transmits this data to the haptic device. This configuration offers greater flexibility and customization, leveraging the full capabilities of TwinCAT3.

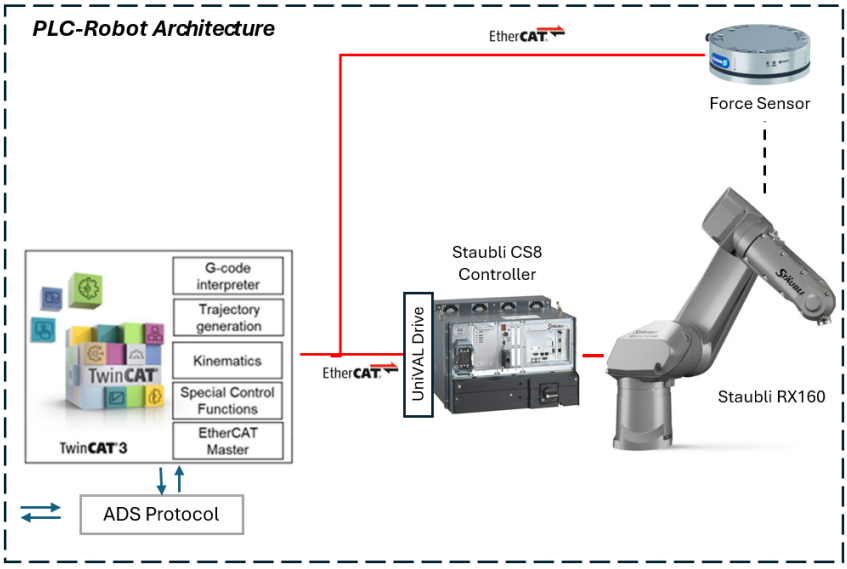

Figure 3.2: PLC-Robot Architecture

To better structure the key components of this architecture in an organized manner, each of the main elements will be described separately. The most important components are the following: the PLC, which handles the entire robotic system control using the TwinCAT3 automation software and the EtherCAT communication protocol; the Staubli RX160 robot and its CS8C controller, which provides the uniVAL Drive interface for axis control; and finally, the sensor system, which enables haptic feedback through force/torque sensors used to close the haptic feedback loop with the haptic device.

PLC and Automation Software Overview

As previously mentioned, the PLC serves as the central control unit in our teleoperation system. It is responsible for receiving cartesian position setpoints sent from the haptic device and converting them into joint position commands for the EtherCAT SubDevice,in this case, the robot controller. Additionally, the PLC reads force data from the force/torque sensor via the EtherCAT bus and transmits this information back to the haptic device, completing the haptic feedback loop.

To enable the external control architecture, a Beckhoff CX2043 Embedded PC was selected. This device integrates an AMD Ryzen™ V1807B CPU operating at 3.35 GHz with four cores. It runs on the Windows 10/11 operating system and includes two independent Gbit-Ethernet interfaces and four USB 3.2 Gen 2 ports, ensuring reliable communication with external devices such as the haptic interface.

For the automation framework, TwinCAT 3 was employed. TwinCAT 3 is an industrial automation suite offering tools for control logic, communication, and robotic applications (53). The application development was performed on a separate engineering workstation using the TwinCAT 3 development environment (XAE). The compiled control logic was then deployed to the embedded PC, where the TwinCAT 3 runtime (XAR) was executed to run the application on the target hardware

The main TwinCAT 3 functionalities employed in this project include several motion control software packages. TwinCAT NCI enables interpolated path movements using three path axes and up to five auxiliary axes. TwinCAT 3 Kinematic Transformation L4 provides control for serial 6-axis kinematics, allowing the computation of forward and inverse kinematics for the robotic system. TwinCAT NC PTP facilitates point-to-point movements by implementing axis control in software, where each axis is represented as an object offering a cyclic interface for components such as the PLC. Each of these axis objects is linked to a physical actuator, corresponding in this case to the robot's controller drives (54). For communication with the haptic interface, the TwinCAT TCP/UDP Realtime and ADS libraries were also utilized (55) (56).

Furthermore, in order to develop the robotic system, it was necessary to establish an EtherCAT communication bus. This setup allows communication between the EtherCAT MainDevice (the PLC) and various EtherCAT SubDevices. Once the I/O configuration is defined—including the PLC, the uniVAL Drive, and the force sensor—data exchange among these components becomes possible, as illustrated in Figure 3.3

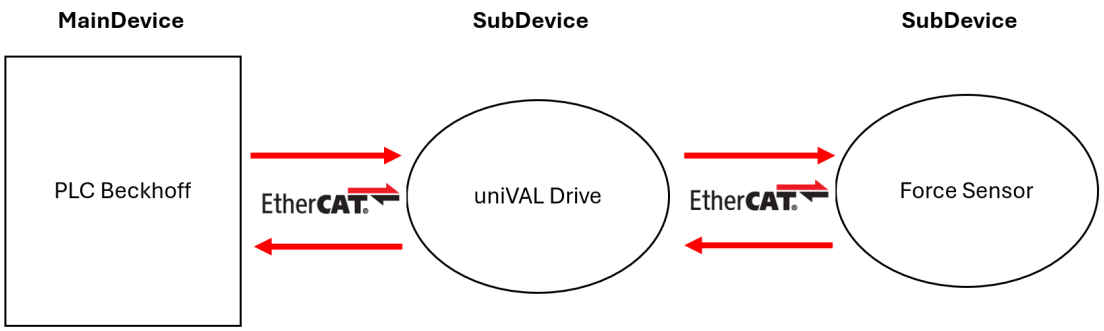

Figure 3.3: EtherCAT Bus

EtherCAT is widely recognized as one of the most commonly used real-time and deterministic communication protocols in industrial automation. In the context of this project, three EtherCAT devices were interconnected to enable real-time data transmission. The EtherCAT bus operated with a cycle time limited to 4 ms, constrained by the update rate provided by the uniVAL Drive interface of the robot controller. EtherCAT functions by using a MainDevice that transmits Ethernet frames sequentially through all connected nodes. SubDevices insert their respective data into these frames on the fly, significantly reducing latency and enhancing communication speed. The protocol relies on standard Ethernet frames, eliminating the overhead of conventional protocol stacks such as TCP/IP, and employs direct memory access (DMA) for efficient and real-time data handling.

Moreover, EtherCAT supports various network topologies, including linear and star configurations. In this implementation, a linear topology was used to interconnect the system components

Robot Control and uniVAL Drive Interface

UniVAL Drive is the interface provided by Stäubli for enabling external control integration—such as that of a PLC—with the CS8C robot controller 3.4. This interface grants direct access to the robot's internal control mechanisms, allowing for high-precision and real-time control of individual joint positions. By supporting major real-time fieldbus protocols (e.g., EtherCAT, POWERLINK, PROFINET, SERCOS), UniVAL Drive enables deterministic transmission of position setpoints from the external controller. Real-time position commands can be issued to each robot axis at a frequency of 250 Hz, corresponding to an update period of 4 ms.

The PLC program is responsible for handling axis configuration, kinematics, and path generation, all of which are implemented directly in the automation software. The CS8C controller, in this setup, manages gravity compensation, axis synchronization, control loops, safety functions, and calibration—ensuring that the position setpoints received for each joint are executed safely and accurately.

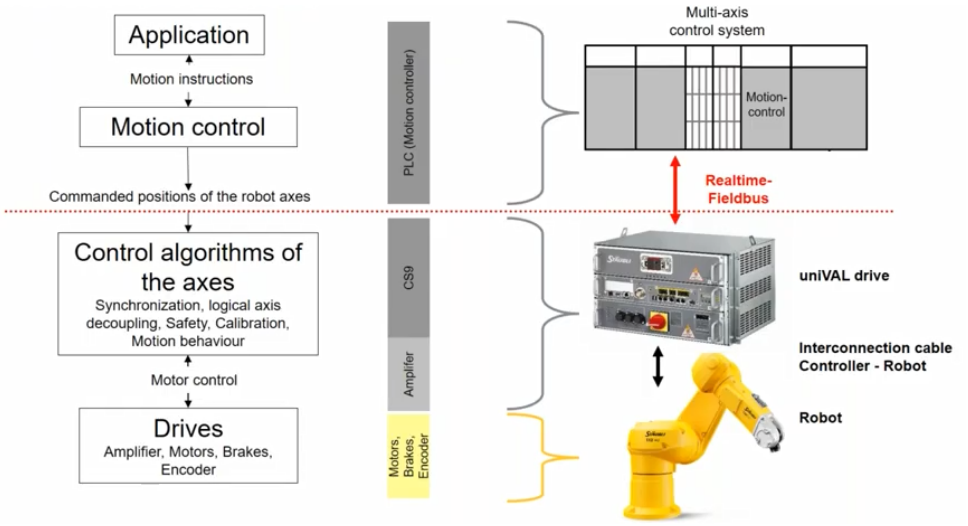

Figure 3.4: uniVALDrive Architecture (6)

With uniVAL Drive, the user gains access to the robot controller up to the level of joint position setpoints, which are generated externally and sent every 4 ms via the EtherCAT bus. This interface allows the user to manage trajectory planning, generation, and kinematics—tasks traditionally handled internally by the robot controller.

From that point onward, low-level control—such as velocity and current loops—is fully managed by the CS8C controller and remains inaccessible to the user. While access to these internal loops would provide finer control over the robot's behavior, this is not currently possible with Stäubli robots. As a result, the robot's dynamic response depends entirely on the controller's internal implementation. If the commanded setpoints exceed the controller's capabilities, issues like tracking errors may occur.

Force Sensing and Feedback

In addition to the SubDevice used to communicate with the robot controller, an additional SubDevice is required in the communication bus to acquire the forces and torques acting on the robot. This is achieved by integrating a force sensor into the system, enabling the PLC to receive real-time measurements transmitted via the EtherCAT bus.

For the teleoperation system to function properly and deliver effective haptic feedback, it is essential to incorporate components capable of accurately measuring interaction forces. These measurements allow the operator to perceive the forces applied and experienced by the robot, resulting in a more immersive and realistic manipulation experience. To this end, a Schunk force sensor has been mounted at the robot's TCP to capture contact force data. Although the forces exerted at both the TCP and the laparoscopic instrument are relevant for achieving highly realistic feedback, the current focus is on reading force measurements at the TCP. Integration of the laparoscopic instruments is planned for future stages of the project to enhance the fidelity and accuracy of the haptic feedback

In this case, as mentioned, a FTE-AXIA80-DUAL SI-75-4/SI-150-8 sensor will be used, which allows measuring the force in the X, Y and Z axes, with a range of ±75 N in X and Y and ±250 N in Z. 3.5 It also measures torque in the Mx, My and Mz axes, with a range of ±4 Nm (7). This sensor offers a resolution between 0.04 N and 0.002 N, which is essential for medical ap-

plications. Communication will be via EtherCAT, although it is also compatible with Ethernet, RS-422 and RS-485. In this work, the EtherCAT protocol will be used.

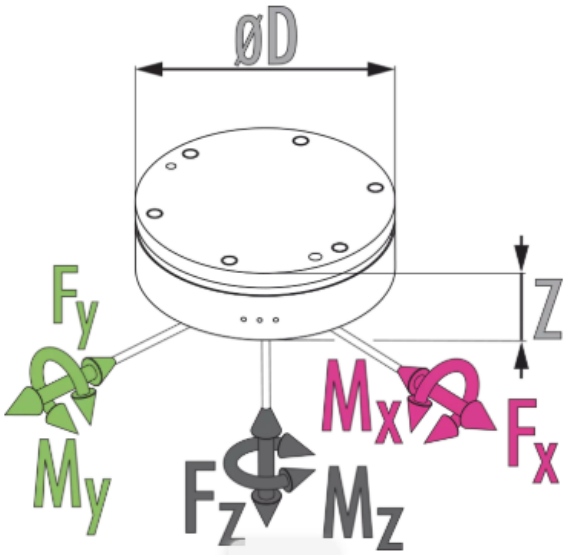

Figure 3.5: Force/Torque Sensor (7)

3.1.2 Haptic-Device Control System

The second architecture designed for the teleoperation system focuses on integrating the haptic device with the robotic system. In this architecture, two fundamental components stand out: on the one hand, the Omega.7 haptic device, which allows the operation of the robot and provides haptic feedback through its API, communicating with the system via a USB connection; on the other hand, the C++ program, which is responsible for implementing the logic through the Omega.7 API and for transferring and receiving information between the PLC and the haptic device. This program enables the transfer of Cartesian position setpoint data from the Omega.7 to the PLC and subsequently to the robot, as well as sending the force readings from the PLC to the Omega.7 so it can close the loop and provide the force feedback in the haptic device 3.6.

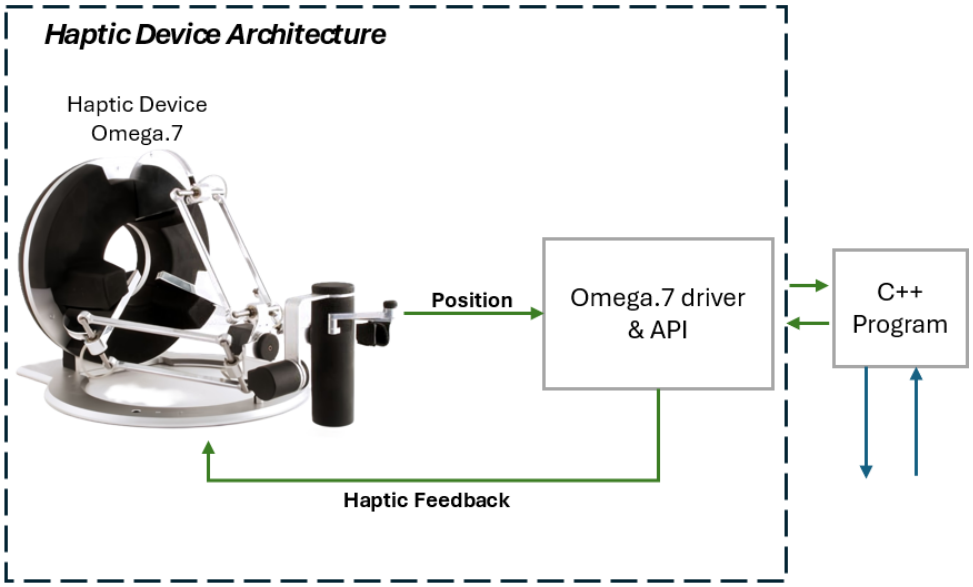

Figure 3.6: Haptic device architecture and the C++ program

Omega.7 Haptic Device

The haptic device for which we have developed this system is the Omega.7, developed by Force Dimension. Its end-effector covers the natural range of motion of the human hand, making it suitable for bi-manual teleoperation console configurations. The device provides three-dimensional force feedback, rotation sensing, and an active grasping mechanism. The Omega.7 is widely used in applications such as medical and space robotics, micro- and nano-manipulation, teleoperation consoles, virtual simulation environments, training systems, and academic research (50) (57).

From a technical perspective, the Omega.7 offers a translational workspace of 160 x 110 mm, a rotational range of 240 x 140 x 180°, and 25 mm for grasping motion. The device is capable of generating forces up to 12.0 N in translation and ±8.0 N in grasping, with a resolution finer than 0.01 mm in translation, 0.09° in rotation, and 0.006 mm in grasping (8). Communication with the host system is established via a USB 2.0 interface, supporting an update rate of up to 4 kHz. The device is compatible with multiple operating systems, including Windows, Linux, macOS, QNX, and VxWorks.

An additional important aspect in the system is the coordinate system used by the haptic device as a reference for positioning. The Omega.7 utilizes a Cartesian coordinate system to represent the position of its end-effector (handle) in metric units (International System of Units). The origin of this coordinate system (0,0,0) is defined at a virtual point located at the center of the device's physical workspace 3.7.

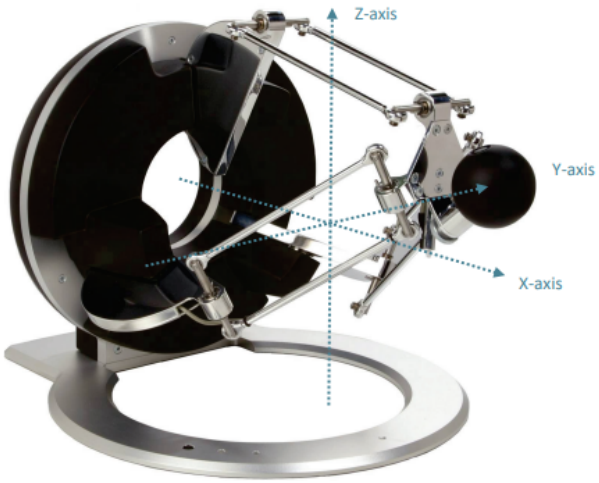

Figure 3.7: Cartesian coordinate system of the omega.x haptic device (8)

In addition to translational positioning, the Omega.7 incorporates a rotational wrist mechanism. Its orientation is defined by a local reference frame (Rwrist), represented numerically by a 3×3 rotation matrix. This matrix is calculated based on angular measurements provided by the joint sensors located on each revolute axis of the wrist.

C++ Program for Haptic Control and PLC Communication

As mentioned earlier, a C++ program has been developed to interact with both the haptic device's API and the PLC. Through the API (SDK), the program is responsible for reading and writing values to the device's driver, allowing control over the haptic device 3.8.

The main objective of the C++ program is to integrate the Omega.7 haptic device with the robotic system via the PLC. The program obtains the position of the end-effector from the Omega.7 using its API. This position is then transformed from the Omega.7's coordinate system to the robot's coordinate system through mathematical transformations. Depending on the system's requirements, the position may be scaled or shifted to fit the robot's working space. Once the position is transformed, it is transmitted to the PLC using either the UDP or ADS communication protocol. Additionally, the program utilizes the Omega.7 API to apply force feedback, which is received from the PLC and delivered to the haptic device, ensuring a closed-loop control system. This setup enables precise control of the robot and provides realistic haptic feedback during teleoperation. Both UDP and ADS protocols have been tested for communication between the PLC and the C++ program, and performance differences will be discussed later.

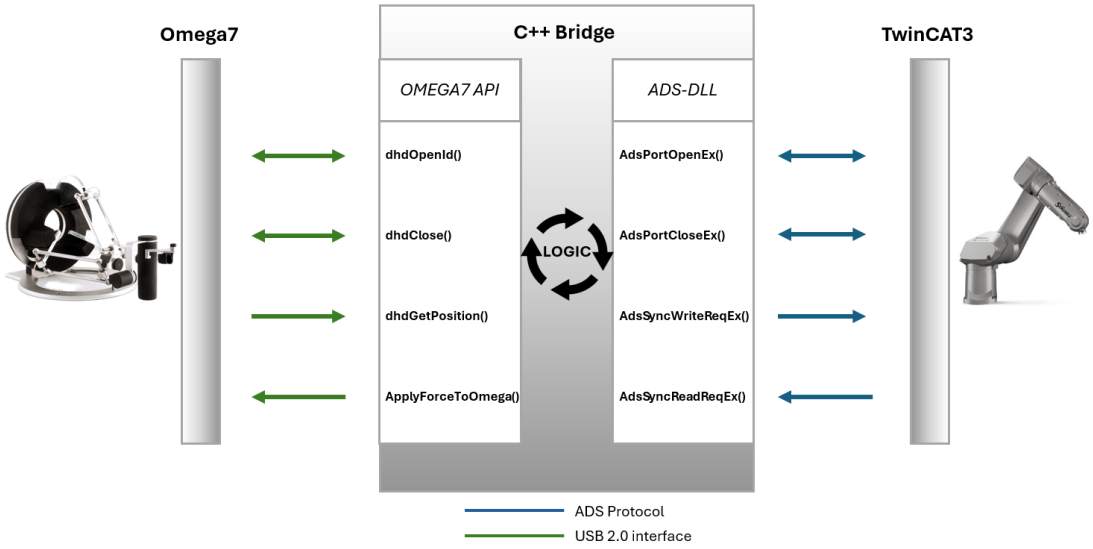

Figure 3.8: C++ Bridge program diagram with ADS or UDP communication protocol data transfer with PLC

To enable communication via ADS, the C++ program uses the TwinCAT ADS library, including the necessary header files and linking the specific ADS library. For UDP communication, the Boost Asio C++ library is used, employing synchronous operations. The Hercules utility was also used for testing and debugging UDP messages, ensuring correct formatting and reliable connectivity with the PLC. To enhance performance and maintainability, the program utilizes the Eigen library for mathematical computations and efficient matrix operations when transforming coordinates between the Omega.7 and the robot's coordinate systems. Additionally, spdlog is used for logging, allowing real-time monitoring and debugging of communication events and system states.

3.2 Implementation

This section describes the practical implementation of the theoretical concepts explained in the previous chapter. First, the development work on the PLC side will be presented, specifically the project created in TwinCAT 3. This will cover the execution tasks, the NC axis system, kinematic groups, trajectory generation, and overall control of the robot's axes. In the second part, the implementation of the C++ code described earlier will be explained. This will include the functions and logic used, leveraging the Omega.7 API and the transformations and scaling of coordinates necessary for communication with the PLC. In summary, this section aims to present the real implementations developed for the theoretical system described so far

3.2.1 PLC programming

In TwinCAT 3, tasks are periodic execution units that organize the control and communication processes within the PLC. For the teleoperation system, three key tasks have been defined to manage the robot's control.

First, the "Main" task is responsible for implementing the core logic of the system, such as enabling the robot axes, generating trajectories, and managing position control. This task has been configured to execute every 4 ms, since the Unival Drive controller limits the system to sending

position setpoints at this frequency. As a result, the reading of feedback from the EtherCAT bus and the sending of new position commands to the robot are synchronized with this 4 ms cycle.

In parallel, a second task is dedicated exclusively to solving the robot's kinematics. Because kinematic computations are among the most execution-intensive processes, this task has been separated to run in parallel with the Main task. The kinematics task is also executed every 4 ms to ensure that at each control cycle, a fresh and updated kinematic solution is available to generate the corresponding position commands.

Finally, a third task is dedicated to communication with the external C++ bridge program. This task is responsible for exchanging data (position and force values) between the PLC and the program. Since two different communication protocols (ADS and UDP) have been tested, the detailed implementation and performance evaluation of this communication task will be explained later in the integration section.

Regarding the axis control logic ofthe project, it is necessary to define the different axis groups that structure the system. We have four fundamental axis groups 3.9.

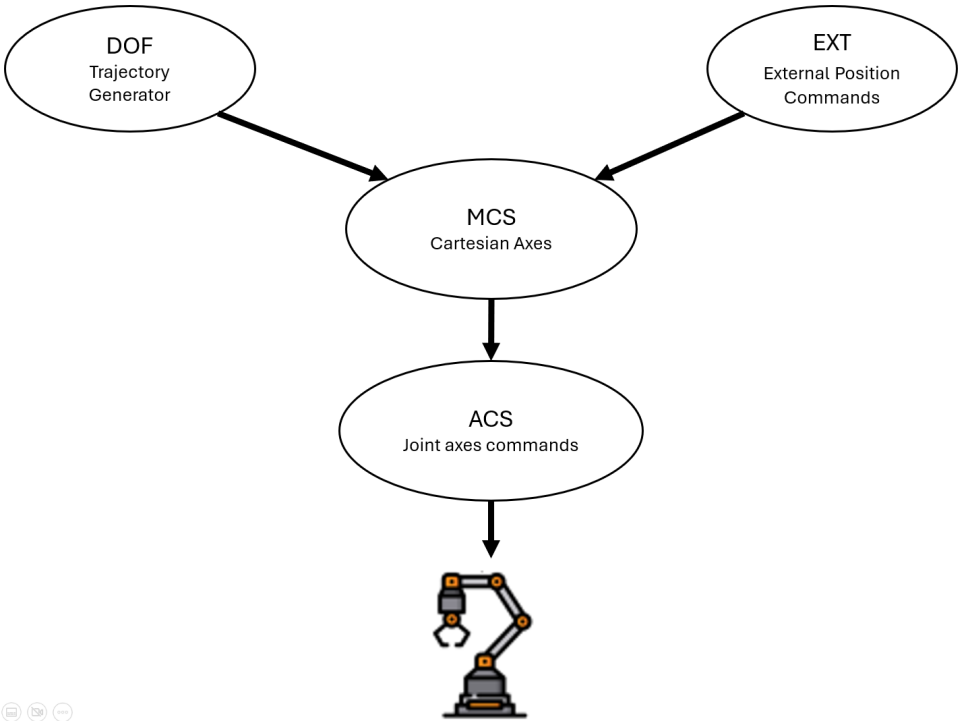

Figure 3.9: NC axis groups of the system

Two groups are responsible for trajectory planning and the management of external position inputs received from the C++ program. These groups are the DOF and EXT (External axes) groups. The DOF group corresponds to the robot's internal trajectory generator, while the EXT group handles external positions provided by the Omega.7 haptic device. External position inputs are added to the robot's original trajectories, enabling teleoperation based on operator commands.

Additionally, two groups manage the execution of robot motion. The MCS group represents

the Cartesian position and orientation of the robot's TCP relative to the robot's base coordinate system. Positions in MCS result from the combination of planned internal trajectories and external inputs. Finally, the ACS group corresponds to the robot's physical joints or actuators. These ACS axes are derived through inverse kinematics based on the MCS positions. The ACS group directly controls the physical robot joints (axes 1 to 6) by sending position setpoints in degrees to the corresponding motors.

In summary, the EXT and DOF groups manage external and internal trajectory references, the MCS group manages Cartesian TCP positioning, and the ACS group handles the physical joint control.

To define the kinematics of the robot, TwinCAT3 provides the option to specify the robot's parameters within the kinematic group configuration 3.10. These parameters are obtained from the manufacturer's robot schematics.

The arm length parameters (L1, L2, L3) and arm offset parameters (D1, D2, D3) define the geometric structure of the robot. L1, L2, and L3 represent the distances between the links, while D1, D2, and D3 correspond to the displacements between the rotational axes. These parameters are fundamental for performing both forward and inverse kinematics calculations. Additionally, the Z3Y2X1_DIN9300 rotation convention is used, following a Cardan sequence where rotations are applied in the order of the Z-axis, Y-axis, and X-axis.

	.Rotation convention	Rotation_Z3Y2X1_DIN9300
	.Spatial reference	00000000
	.Definition direction	toReference
-	TCP extension	
	Tool offset OID	00000000
-	Kinematic	
	Arm length L1	825.0
	Arm length L2	625.0
	Arm length L3	110.0
	Arm offset D1	150.0
	Arm offset D2	0.0
	Arm offset D3	0.0

Figure 3.10: Kinematic parameters and configuration

Once the kinematics configuration is completed, attention is directed to the two groups responsible for generating the robot's trajectory. The first group, DOF axes, represents the robot's position and orientation within Cartesian space, indicating the robot's spatial location and orientation

To control the robot's movement, its axes are configured to accept new setpoint values cyclically. The robot is then coupled with the external system (Omega.7) in a master-slave configuration, where the Omega.7 provides displacement offsets instead of absolute positions. These offsets are added to the robot's original trajectory, allowing the robot to follow the intended path while being influenced in real time by the operator's input.

As an example, if Omega7 sends an offset of 10 mm along the X-axis for the robot's TCP, and the robot's current position is (X=100 mm, Y=50 mm, Z=200 mm), the new position of

the robot would become (X=110 mm, Y=50 mm, Z=200 mm). This ensures that the robot's movements match the displacement from Omega7. Finally, using inverse kinematics, the new positions of each joint are calculated and sent to the robot's controller, completing the motion synchronization between the robot and the external system.

Finally,the I/O components of the project have been implemented. Two SubDevices are present on the EtherCAT bus: the uniVAL Drive controller of the robot and the force sensor. Regarding the uniVAL Drive controller, a bus scan reveals the interface shown in Figure 3.11

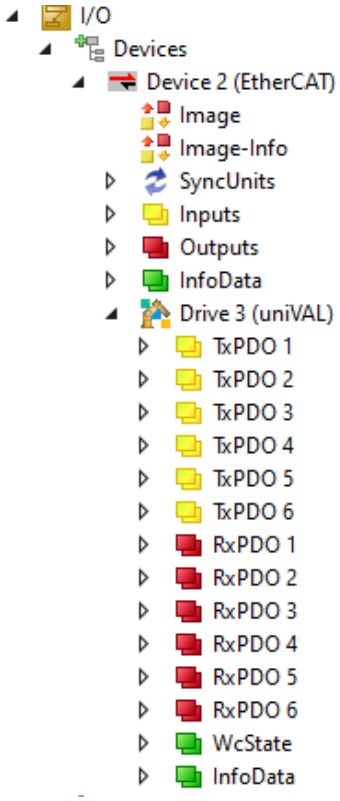

Figure 3.11: uniVAL Drive SubDevice in the EtherCAT bus

In Figure 3.11, the uniVAL Drive SubSystem is shown to provide the controller's inputs and outputs. The six yellow variables represent outputs (Position Actual Value, Velocity Actual Value, and Following Error Actual Value), while the red variables correspond to inputs (Target Position). Within the ACS axis configuration, linking the PLC axis variables to the controller's I/O is required to send the calculated position commands.

3.2.2 C++ Programming

To establish communication between the PLC (and thus the robot) and the emulated Omega.7 haptic device, the API provided by Force Dimension in C++ was utilized. A C++ program was developed to enable communication between both devices (PLC and Omega.7) via this API. The development process was carried out using Visual Studio 2019.

This section describes the development of the program, emphasizing the use of functionalities provided by the Force Dimension API. Additionally, implementation of code to establish communication with the PLC was required alongside the API integration.

Nevertheless, the development of communication with the PLC will not be described in this section. Instead, the focus will be exclusively on the use of the API and the implementation of functions for coordinate transformation and position scaling. The implementation of communication using ADS and UDP protocols with the PLC is addressed in the subsequent section titled ”Integration”.

The C++ program was structured using three classes, each with specific functionalities: HapticsDeviceInterface, HapticsProcessor, and HapticsRobotController

The HapticsDeviceInterface class is responsible for direct and low-level interaction with the Omega.7 through the Force Dimension API 3.12. Its primary function is to abstract device management operations and real-time data acquisition. The class stores the device ID and initializes the connection status as disconnected. The connect() method attempts to open a communication channel with the Omega.7 by invoking dhdOpenID(). If successful, it updates the internal connection flag and logs the event; otherwise, it reports an error. Similarly, the disconnect() method properly closes the device communication using dhdClose(), ensuring that all hardware resources are released safely.

In terms of data acquisition, the class uses methods getPositionAndOrientation() to retrieve the current 3D position and orientation of the device as a vector and a 3x3 matrix, respectively, by calling dhdGetPositionAndOrientationFrame().

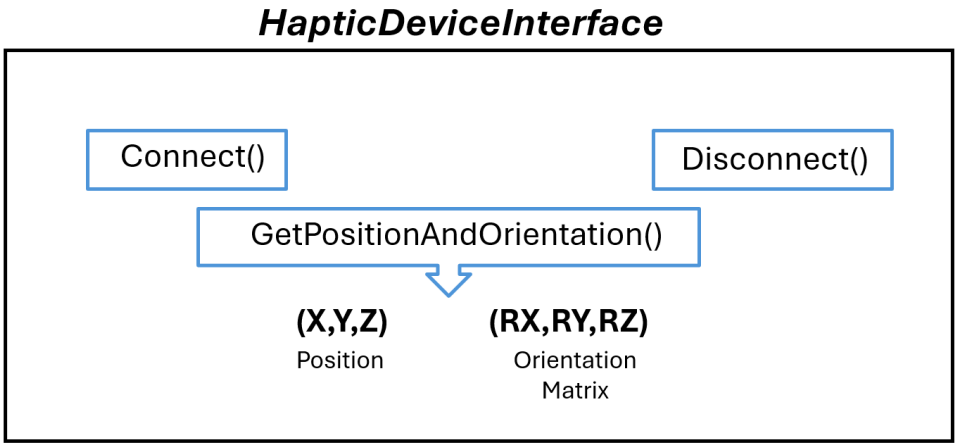

Figure 3.12: HapticDeviceInterface Class methods

The HapticsProcessor class focuses on processing and interpreting the raw data obtained from the Omega.7 3.13. It does not interact with the hardware directly. Its key responsibility begins with the requestInitialPosition() function, which queries the device for its current position and orientation and stores them as reference baselines. These initial conditions are critical because all subsequent movements and rotations are measured relatively to them.

The method getDeltaPositionAndOrientation() calculates how much the device has moved and rotated compared to the stored initial state. The delta position is simply the element-wise difference between the current and initial 3D positions. The delta orientation is more complex: it uses linear algebra operations provided by Eigen’s Matrix3d class to compute a relative rotation matrix. Specifically, the delta orientation is determined by multiplying the transpose of the initial orientation by the current orientation.

This relative data representation (instead of absolute measurements) is crucial for ensuring compatibility with the PLC's expected input formats, which operates in relative motion control as we saw before. The class gracefully handles uninitialized or disconnected states, returning neutral values (zeros and identity matrices) if necessary.

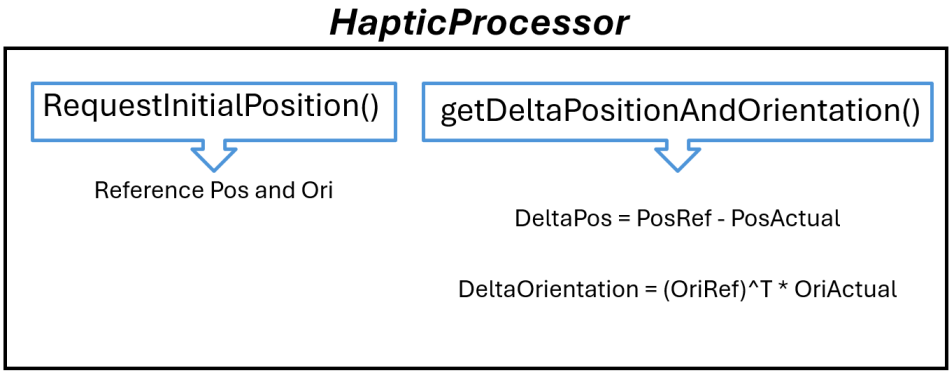

Figure 3.13: HapticProcessor Class methods

The HapticsRobotController class bridges the gap between processed haptic data and the operational requirements of the robot controlled via the PLC 3.14. It operates at a higher level than the other two classes.

Its first major role is coordinate transformation. The applyRotationToRobot() method takes a user-defined rotation matrix (representing, for instance, a reorientation between device and robot frames), applies it to both the delta position and delta orientation, and optionally scales the results by a reduction factor. This factor is useful when it is necessary to scale down the user's hand movements to finer robot motions, thus improving control precision or safety.

The second major role is force and torque feedback. The applyForceAndTorqueToOmega7() method receives force and torque vectors, applies the appropriate rotation transformation to match the Omega.7's reference frame, and then uses dhdSetForceAndTorque() to physically apply these forces and torques back to the haptic device. This creates a realistic force feedback experience for the operator.

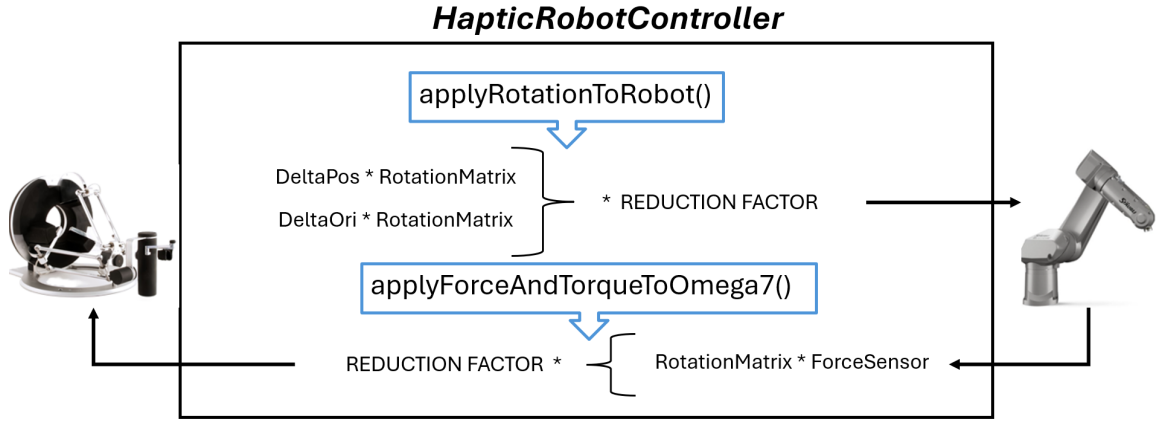

Figure 3.14: HapticRobotController class methods

In summary, the developed C++ application is structured around three core classes — Hap-

ticsDeviceInterface, HapticsProcessor, and HapticsRobotController — each with a specific role. The HapticsDeviceInterface manages the direct connection to the Omega.7, retrieving real-time data such as the device's 3D position and orientation matrix. For example, it can obtain the current position of the Omega.7's end-effector as coordinates (x, y, z) at any moment. The HapticsProcessor takes this raw data and computes relative changes (deltas) between the current and initial position and orientation, which is critical for interpreting movements rather than absolute positions. For instance, if the user moves the device 5 cm to the right, the processor detects and outputs only that displacement. Finally, the HapticsRobotController transforms these deltas into the robot's reference frame, optionally scales the movements if needed (e.g., reducing a large hand movement to a small robotic arm motion), and applies external forces and torques to the Omega.7 to simulate physical interactions.

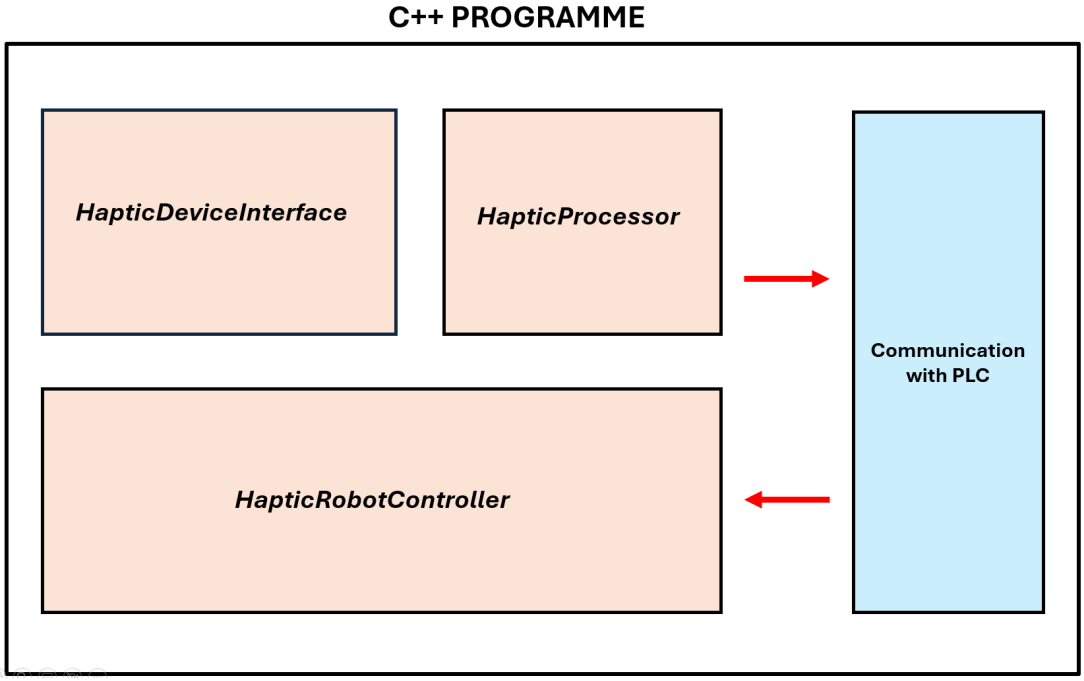

Figure 3.15: C++ program overall scheme

In Figure 3.15, the pink-colored sections represent the code developed using the Force Dimension API, while the blue section corresponds to the communication module implemented to interface with the PLC. The latter will be described in the following section.

3.3 Integration

This section focuses specifically on the integration of communication between the C++ program and the PLC, emphasizing the importance of reliable communication and low latency, as both factors significantly influence the robot's dynamic response. The first communication method considered was the ADS protocol provided by TwinCAT 3. However, after evaluating its performance, the decision was made to switch to the UDP protocol. Both communication approaches are presented in the order in which they were developed—first ADS, followed by UDP. For each method, the implementation of the code on both the C++ side and within the PLC program is described. Additionally, this section outlines the logical flow of the system.

3.3.1 Communication Protocols and Data Exchange

As previously mentioned, a key factor for the effective performance of the teleoperation system is the communication latency between the haptic device and the PLC. This communication is responsible for sending, from the Omega.7 to the PLC, the position commands for the robot's six axes. To achieve this, two different communication protocols were used: initially ADS, and later UDP.

ADS Protocol

The first communication protocol studied was TwinCAT 3's native protocol, the ADS. ADS enables data exchange and control of TwinCAT systems over TCP/IP connections. This means that ADS operates on top of the TCP/IP layer, using it as the transport mechanism to ensure reliable and ordered delivery of messages over a network. ADS follows a client-server communication pattern: the client sends a request (ADS Request), the server receives it (Indication), processes it and sends a response (ADS Response), which the client receives as a confirmation (ADS Confirmation). To implement this protocol, the ADS library functions provided by TwinCAT 3 were used to enable external communication with the PLC.

While EtherCAT operates as a real-time protocol, ADS does not offer deterministic or real-time communication. Although ADS is reliable for many industrial applications, the frequency of data packets may vary. For instance, one packet may arrive within 1 ms, while the next might take 2 ms, resulting in a non-deterministic communication pattern. This behavior must be considered when evaluating its impact on the robot's dynamics, particularly in terms of latency.

As previously explained in the context of the code interfacing with the Omega.7 API, a dedicated class named *PlcAdsClient* was also developed to facilitate communication with the PLC via the ADS protocol. This class provides a high-level abstraction that enables operations such as reading from and writing to PLC variables, in addition to managing the connection lifecycle. The following section describes the key components of the *PlcAdsClient* class.

The class constructor takes an AdsClientConfig object, which holds the connection configuration, including the AMS address and port of the PLC. Each TwinCAT device in the network is identified by an AMS NetId, which consists of six octets. The first four octets can be freely chosen, while the port refers to the TwinCAT 3 runtime system.

Regarding connection management 3.16, the class provides two key methods. The connect() method establishes a connection to the PLC using the provided AMS address and port. To connect to the device, the AdsPortOpenEx() function is used. The disconnect() method closes the connection to the PLC by calling AdsPortCloseEx(). For reading from and writing to PLC variables, the *PlcAdsClient* class provides template methods that support different data types.

On one hand, there is the write(symbol, data) method. This method writes data of type T to the PLC variable specified by symbol. It first checks whether the client is connected and then performs the write operation using the AdsSyncWriteReqEx() function from the TwinCAT ADS library. To obtain the variable's handle from its symbol (i.e., the PLC variable name, for example displacementX : LREAL), a dedicated function has been implemented. Once the handle is retrieved, AdsSyncWriteReqEx() is used to perform the write.

On the other hand, the read(symbol) method reads the value of the PLC variable specified by

symbol and returns it as type T. If the client is not connected, it logs an error. This method uses the AdsSyncReadReqEx2() function to read values from the PLC, ensuring accurate data retrieval.

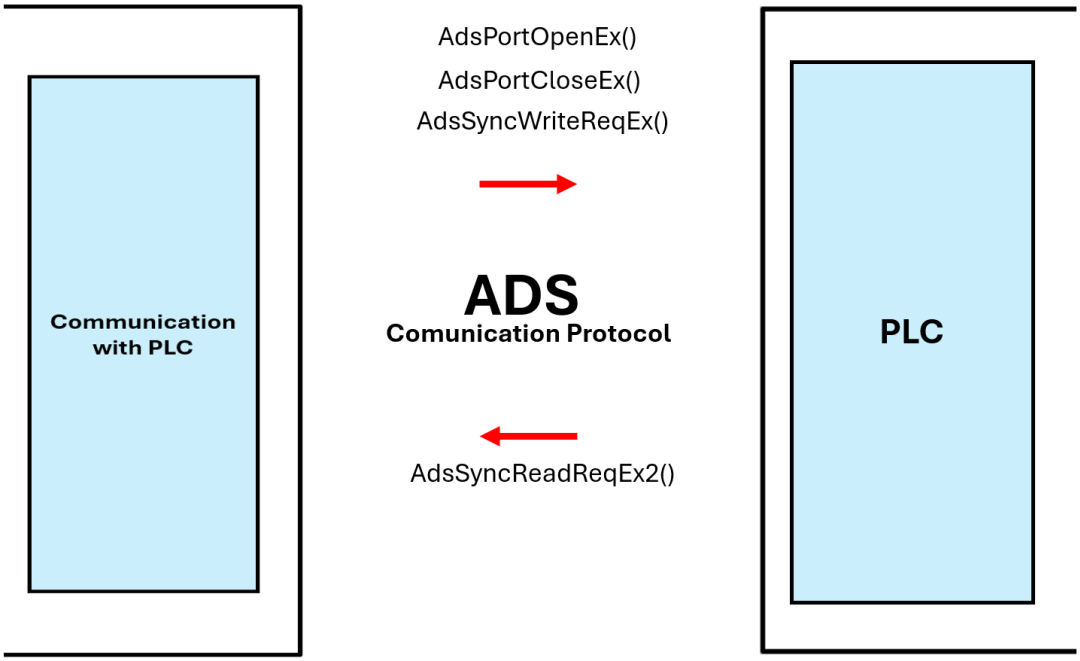

Figure 3.16: C++ program and PLC communication using ADS communication protocol

The class makes use of spdlog to log all communication activities, including successful reads and writes, as well as any errors that occur.This integration helps with debugging and ensures that the communication process is transparent.

To calculate and analyze the latency of the communication between the C++ program and the PLC, we implemented a testing procedure that measures the time taken for each read and write operation in a loop. This test allows us to assess the real-time performance and identify any potential delays in communication, which are critical for teleoperation systems that require precise and responsive control. For calculating the latency, during each iteration, the start time is recorded before reading and writing operations, and the end time is recorded afterward. The elapsed time in microseconds is calculated by subtracting the start time from the end time.

During the analysis of the communication latency obtained with the ADS protocol, it was identified that latency plays a critical role in the dynamic performance of the robot. As previously noted, reading variables from the PLC requires 4 ms. To maintain optimal dynamics, new position values from the Omega.7 must be transmitted to the PLC within this time frame. Consequently, an alternative implementation was developed using the UDP protocol instead of ADS, with the objective of evaluating whether this approach would enhance the performance of the teleoperation system

UDP Protocol

The development of the code to enable communication using the UDP was more complex due to the nature of this protocol. UDP is a layer 4 transport protocol in the OSI model that enables fast and connectionless data transmission. It does not guarantee delivery, ordering, or data in-

tegrity, making it suitable for applications where speed is more critical than reliability. It is also worth mentioning that, like the ADS protocol, UDP is not a real-time protocol.

The goal with this protocol was to ensure that the PLC variable was constantly ”bombarded” with UDP communication so that the position data being read from the PLC would be the most recent, i.e., with the least possible delay. This will be further clarified in the results section, where we will explain the importance of latency and the outcome of both solutions.

Unlike the ADS protocol, which allows direct linking between the C++ application and PLC variables through handles—thus requiring implementation only on the C++ side—the use of UDP necessitated development on both the C++ application and TwinCAT 3 sides. This is due to the inherent characteristics of UDP communication, which can be summarized by the principle: “specify the data to be sent, the IP address, and the port, and it will be transmitted without delivery guarantees.” As a result, additional logic had to be implemented to interpret and convert the received data into meaningful information for the control system

To establish communication between the PC and TwinCAT, a class has been developed in C++ using the Boost Asio library, which enables efficient socket management.The core of the implementation is the UdpSocket class, which encapsulates both sending and receiving data. In its constructor, a UDP socket is opened on the local receiving port, and the remote endpoint to which data will be sent is defined, consisting of an IP address and a destination port. 3.17

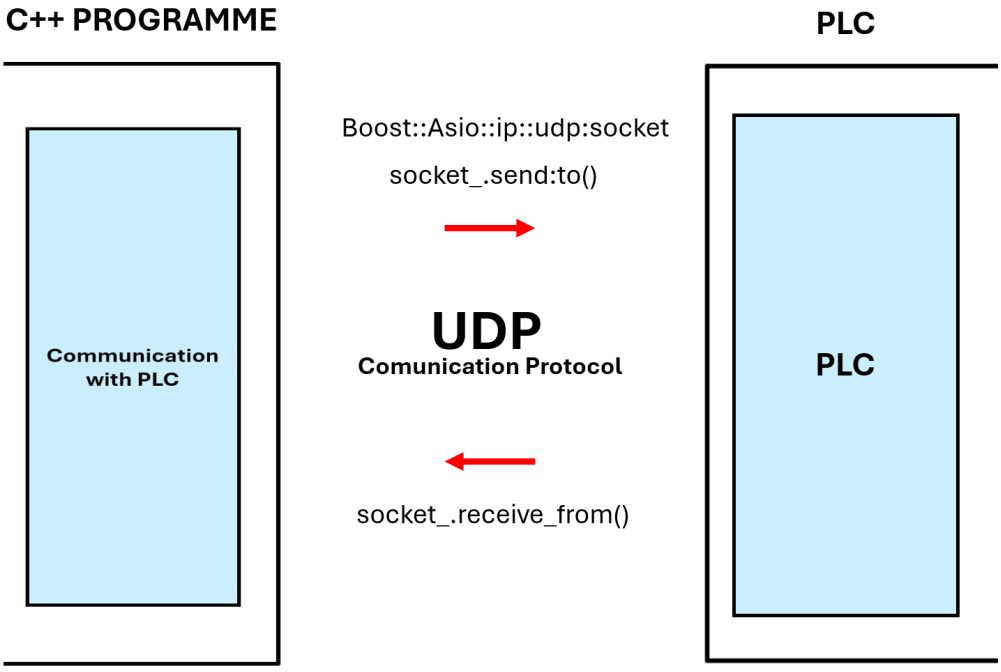

Figure 3.17: C++ program and PLC communication using UDP communication protocol

Data transmission from the C++ program to the PLC is handled by the sendPosition() method. This method constructs a 53-byte packet with the following format: a start byte, a message type byte (used for future extensions, in case more than just position commands are sent), a byte indicating the length of the data payload (48 bytes in this case), the position data itself (6 double values, each 8 bytes), and an end byte 3.18.

StartByte	DataType	Lenght	Position data	EndByte
1 byte	1 byte	1 byte	48 bytes	1 byte

Figure 3.18: Designed UDP packet format for sending data to the PLC

This packet is sent synchronously using the send_to() function provided by Boost Asio. The data is continuously transmitted in a dedicated thread (sendForever()).

To receive the data sent from the C++ program to TwinCAT, the TF6311 TCP/UDP Realtime product was initially used in TwinCAT 3. This library provides direct access to network cards from the real-time environment. The ReceiveData block uses a receive buffer to temporarily store incoming UDP data as raw bytes 3.19.

The process works as follows: first, bytes arrive from the network via the UDP socket. Then, the function block writes these raw bytes into the buffer. The program subsequently scans this buffer to identify a valid message using predefined markers—specifically, the start byte, message type, payload length, and end byte.

Once a valid structure is detected, the 48-byte payload is copied from the buffer into a PLC position variable using the MEMCPY function. This mechanism ensures that only well-formed and complete data packets are processed.

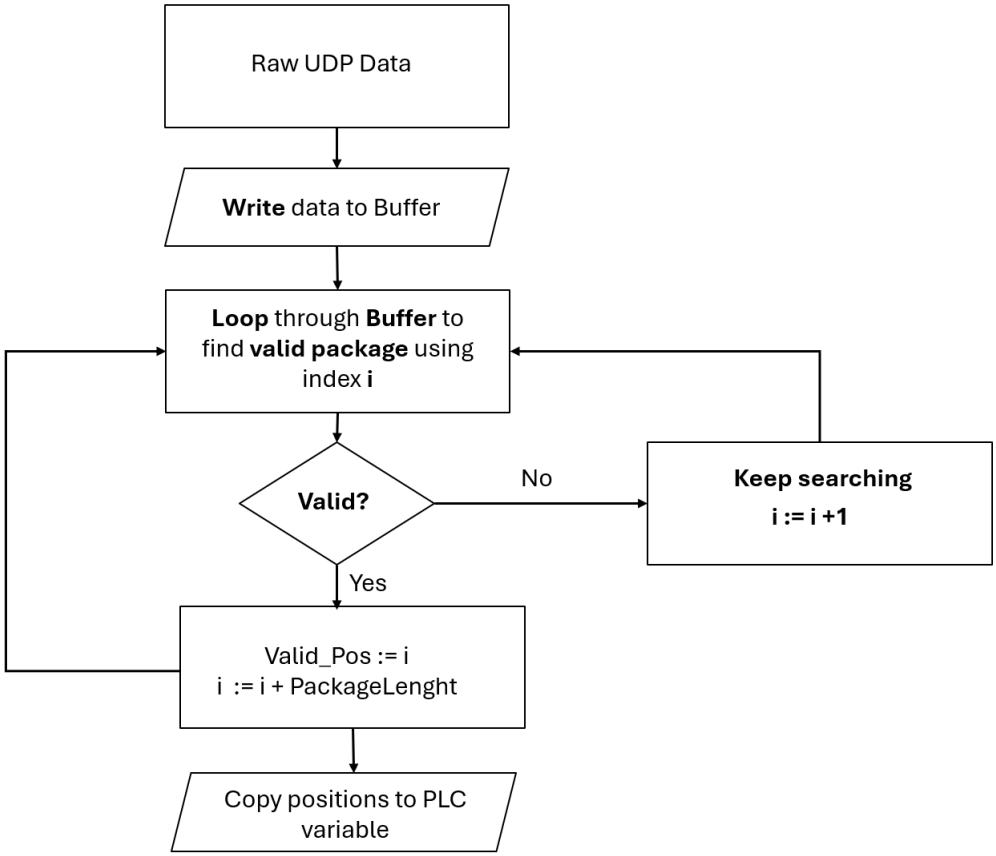

Figure 3.19: Logical flowchart of the TwinCAT 3 program for receiving and validating UDP packets sent from the C++ program

In this way, the packets sent from the C++ program can be written to a PLC variable, allowing the Omega7's displacement to be applied to the robot. The latency with which this variable is updated depends on the cycle time assigned to the data acquisition task, which in TwinCAT 3 can be set as low as 125µs or even lower, depending on the hardware configuration. Despite this, the main task (Main) reads the variable every 4ms, matching the EtherCAT bus cycle time used to update the controller.

Regarding the data sent from the PLC to the C++ program—and ultimately to the Omega7—a similar implementation has been developed in both TwinCAT 3 and the C++ application. Importantly, the same socket used to receive data from the C++ application is also used to send force values back. This reuse of the socket simplifies the communication setup and ensures bidirectional data exchange between the PLC and the external application over a single UDP channel.

As with the position packets, the measured forces are also encapsulated in a specific format 3.20. In this case, the packet consists of a start byte, a byte indicating the data type (to allow future extensibility), a byte specifying the payload length (in this case, three DOUBLE values of 8 bytes each, totaling 24 bytes), and a final byte to mark the end of the packet.

StartByte	DataType	Lenght	Force data	EndByte
1 byte	1 byte	1 byte	24 bytes	1 byte

Figure 3.20: Designed UDP packet format for receiving data from the PLC

On the C++ program side, the same buffer parsing logic has been implemented to read valid packets as was used in TwinCAT 3 for reading position data.

In this way, a robust communication system has been developed to both send position commands (displacements) from the C++ program to the PLC and receive force sensor measurements from the PLC back to the C++ program, so these values can be used to provide haptic feedback to the Omega7.

4 Experimental Results

This section presents the results obtained from the experiments carried out to evaluate the system's performance. The experimental analysis includes measurements of communication latency during teleoperation, as well as the evaluation of force feedback performance.

4.1 Robot Teleoperation Performance using ADS

As previously mentioned, the ADS communication protocol was initially used to communicate the Omega7 with the PLC. Once the entire system was implemented, it was time to run tests using code-generated trajectories (since the haptic device had not yet arrived, we simulated the trajectory that the haptic device would generate using software). To evaluate the teleoperation system's response, a sinusoidal signal was generated to move the robot along a Cartesian axis (e.g., the X-axis). A sine wave was chosen because it does not involve constant velocity, making it easier to observe system dynamics.

Two trajectory tests were performed, and the following describes the first one. During this test, it was observed that the robot occasionally exhibited abnormal vibrations while following the trajectory. To understand the source of these vibrations, the position and velocity signals of the Cartesian positions were analyzed 4.1.

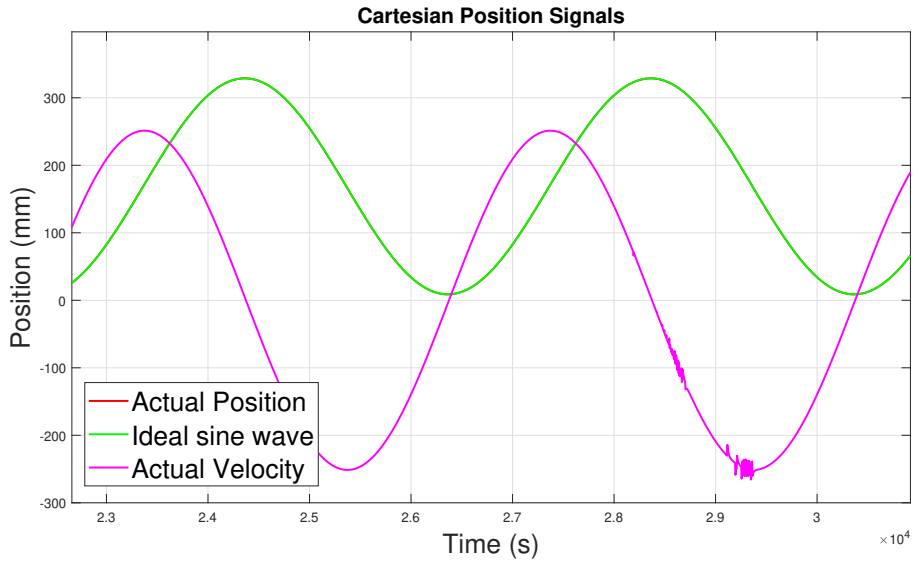

Figure 4.1: Robot First Cartesian Trajectory Following – Position and Velocity Analysis

It can be observed that the robot accurately follows the trajectory sent from the C++ program. The red line represents the robot's position setpoint, which closely tracks the ideal sine

wave shown in green.

Once the velocity vibrations, which are generated due to the variation in the Cartesian setpoint read from ADS, were visualized, the next step was to observe how this variation is reflected at the robot's axis level. To achieve this, the trajectory tracking was plotted 4.2, visualizing the current position, the setpoint (which is already affected due to the Cartesian setpoint variation, meaning it inherits the damaged setpoint), and the current velocity.

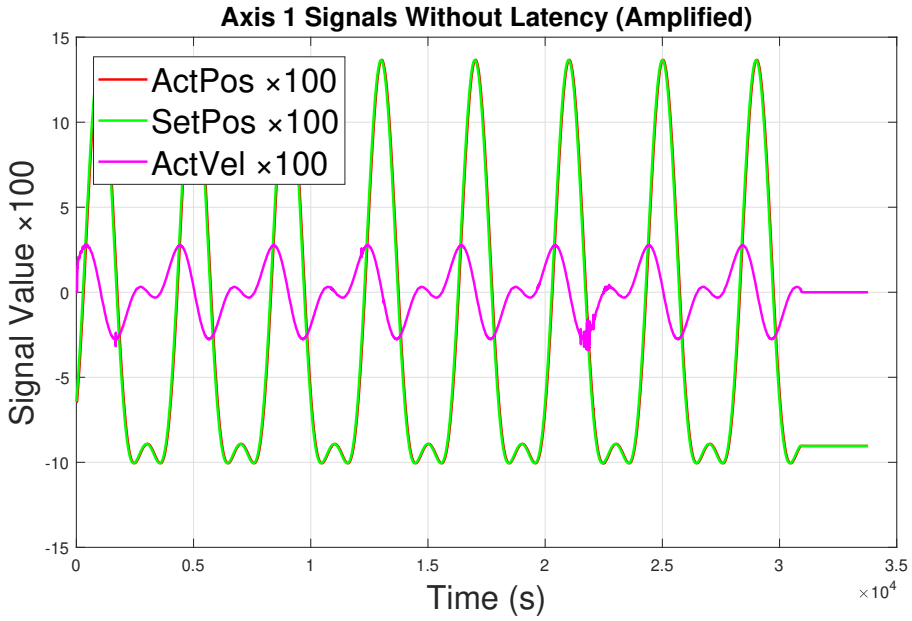

Figure 4.2: Robot first Joint 2 Trajectory Following – Position and Velocity Analysis

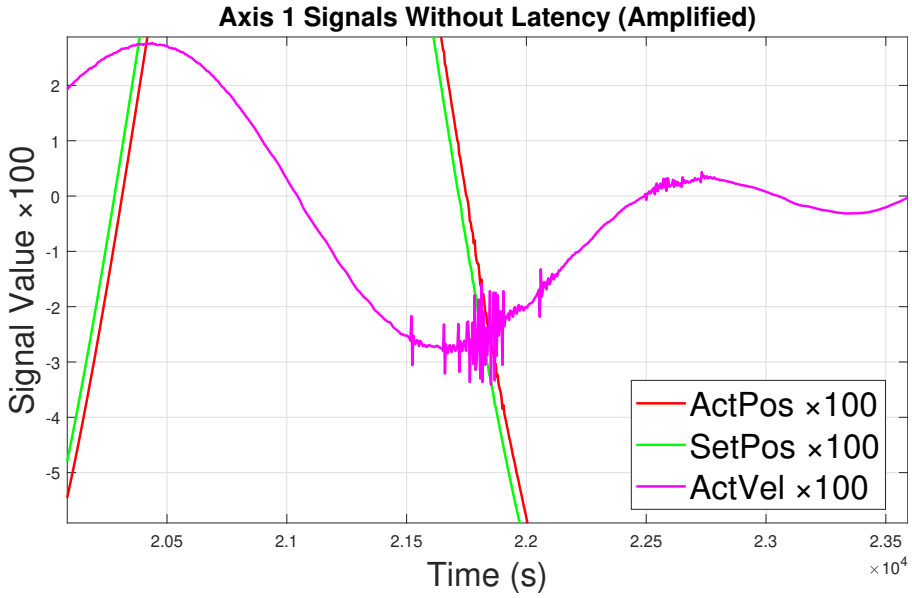

Figure 4.3: Amplified Analysis of joint 2 Velocity Disturbances Induced by SetPoint Variations

From the graph, it is clear that the velocity vibration (and the physical vibration of the robot) is caused by fluctuations in the position SetPoint being sent to the controller. As seen in the image 4.3, both the SetPoint signal (green) and the actual position (red) exhibit variations

that result in abrupt dynamic responses from the controller.

To quantify the absolute positioning error at the robot's joints, the absolute position error of Joint 1 has been plotted 4.4. This allows us to analyze how these vibrations impact the robot's teleoperation performance.

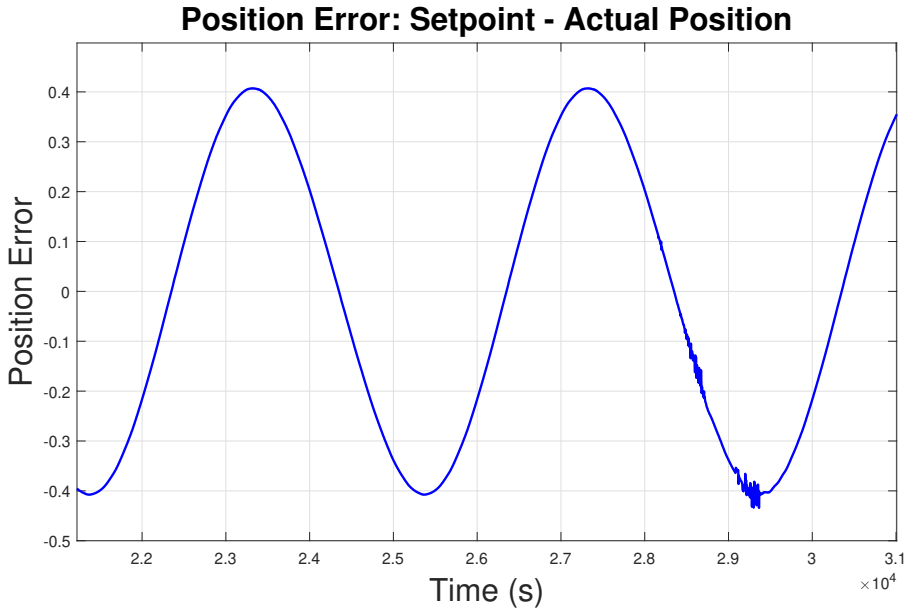

Figure 4.4: Absolute Cartesian Positioning Error (First Trajectory)

To rule out the possibility that the issue was due to the specific trajectory used, the same tests were performed with the robot in a different pose. This resulted in the following graph 4.5.

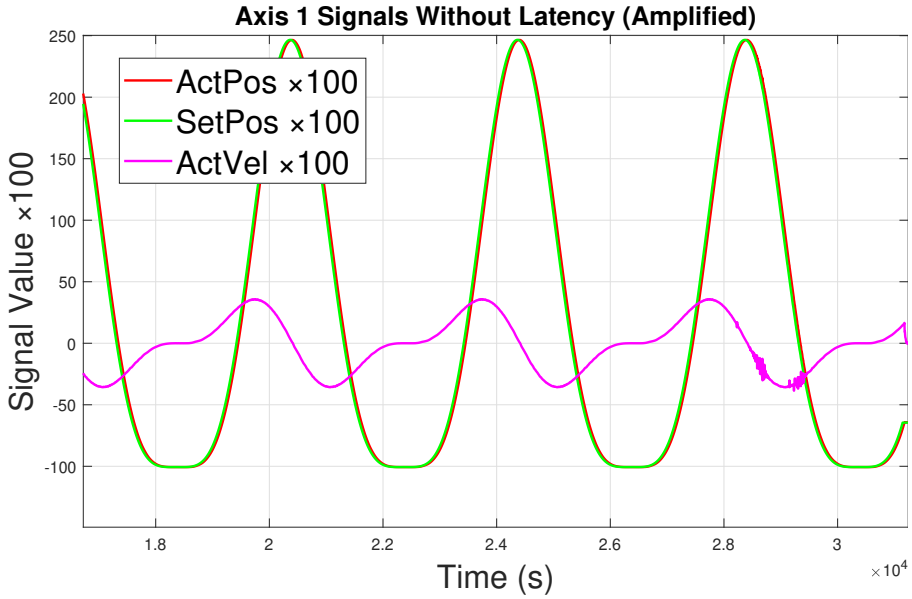

Figure 4.5: Robot Second joint 2 Trajectory Following – Position and Velocity Analysis

In this case as well, abnormal behavior of the robot can be observed during certain stages of the trajectory. As in the previous test, the data was examined in detail to determine whether

the fluctuations in the position setpoint were also responsible for the observed vibrations. As shown in Figure 4.6, this appears to be the case.

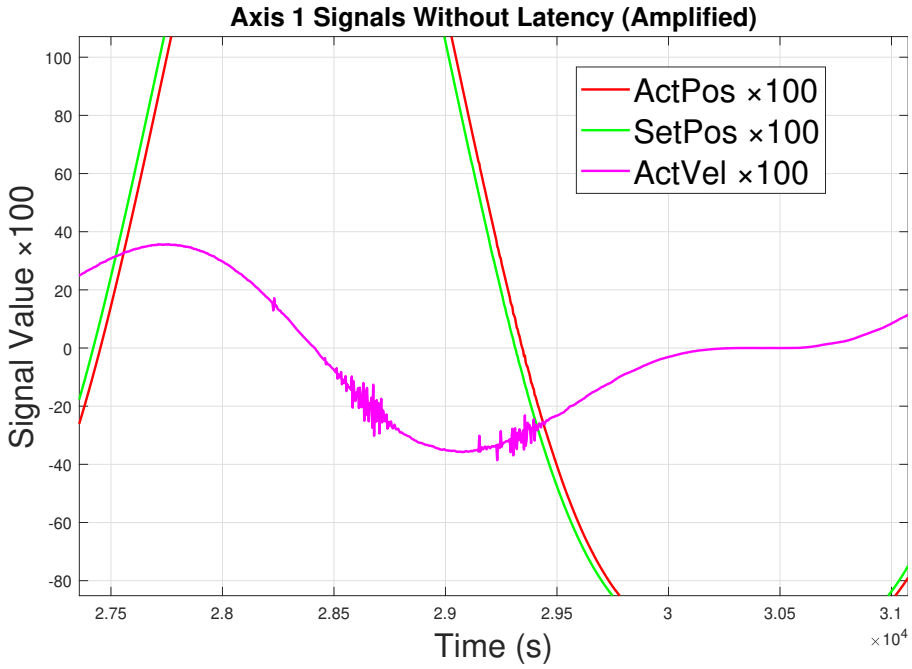

Figure 4.6: Detailed Analysis of joint 2 Velocity Disturbances Induced by SetPoint Variations

The absolute positioning error has also been quantified, as shown in the following figure, where the presence of vibrations is clearly visible 4.4.

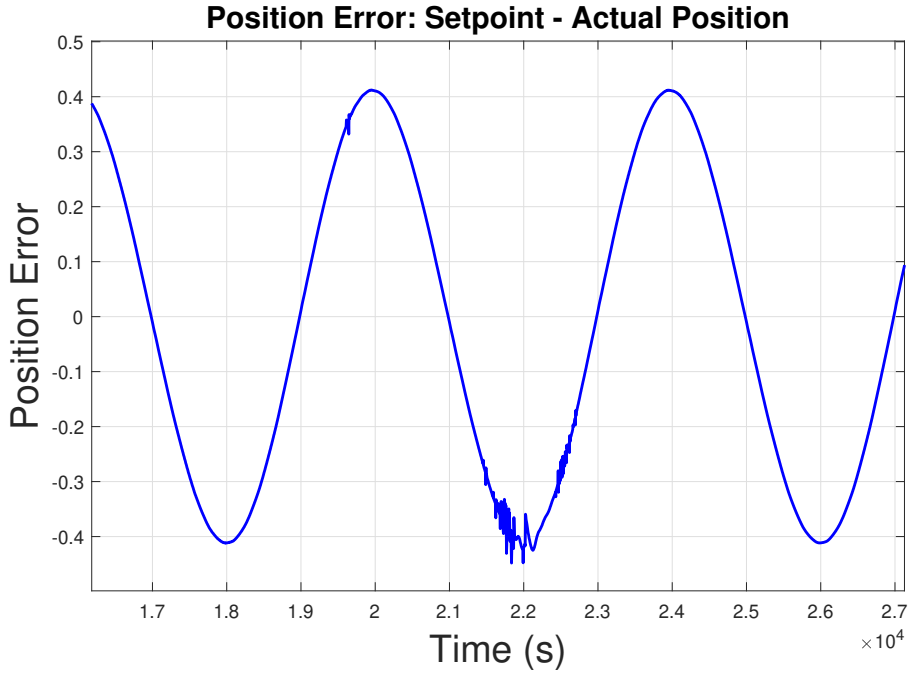

Figure 4.7: Absolute Cartesian Positioning Error (Second Trajectory)

In these graphs, it can be observed that the teleoperation system functions correctly. That is, the displacement generated by the Omega7 (in this case, generated through code) sends

position commands to the robot, and the robot follows the commanded trajectory. The only drawback identified is the vibration observed in the motion, which is caused by the variation in the position setpoint. This variation leads the controller to demand high dynamics from the robot, as it attempts to follow rapidly changing commands. As a result, the robot's controller is pushed closer to its dynamic limits, which can cause instability or visible oscillations in the movement. As observed in the previous plots, a delay in the position commands was identified. To investigate the cause of this behavior, the communication latency with the PLC was analyzed in order to determine whether the vibrations were a consequence of latency issues. The plots shown in Figure 4.8 were generated for both trajectories to support this analysis.

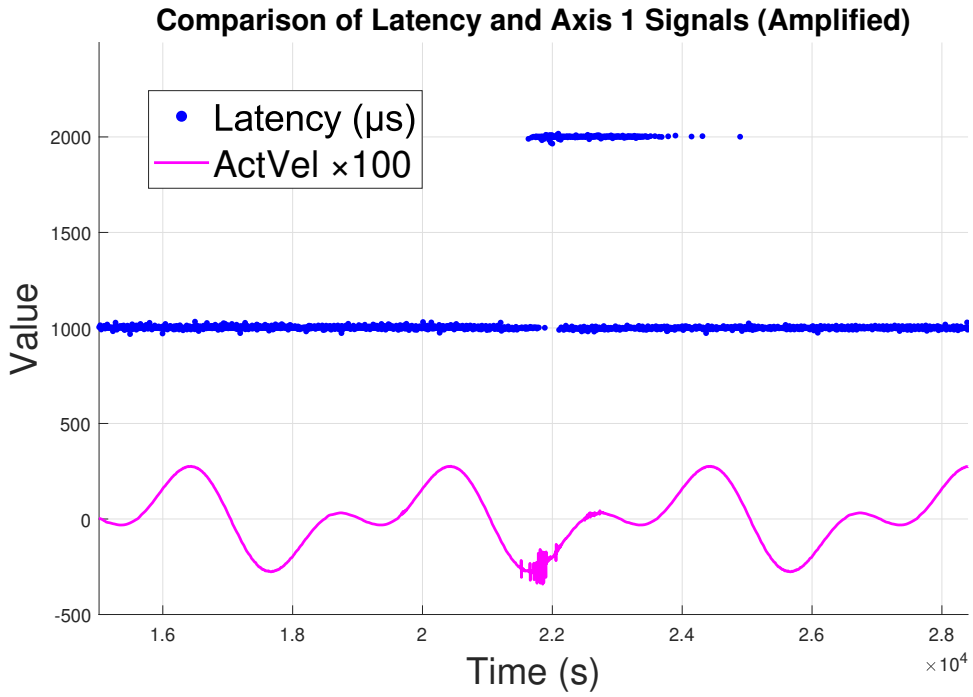

Figure 4.8: Impact of Communication Latency on joint Stability and Robot Dynamics

Figure 4.8 presents the current velocity. Additionally, communication latency with the PLC is depicted in blue dots, representing the interval at which the PLC variable is updated via ADS. It is evident that when latency remains stable around 1 ms, the robot accurately follows the trajectory. However, increases in latency to approximately 2 ms cause abrupt changes in the setpoint, resulting in vibrations within the teleoperation system. A similar effect is observed in the first generated trajectory, as shown in Figure 4.9.

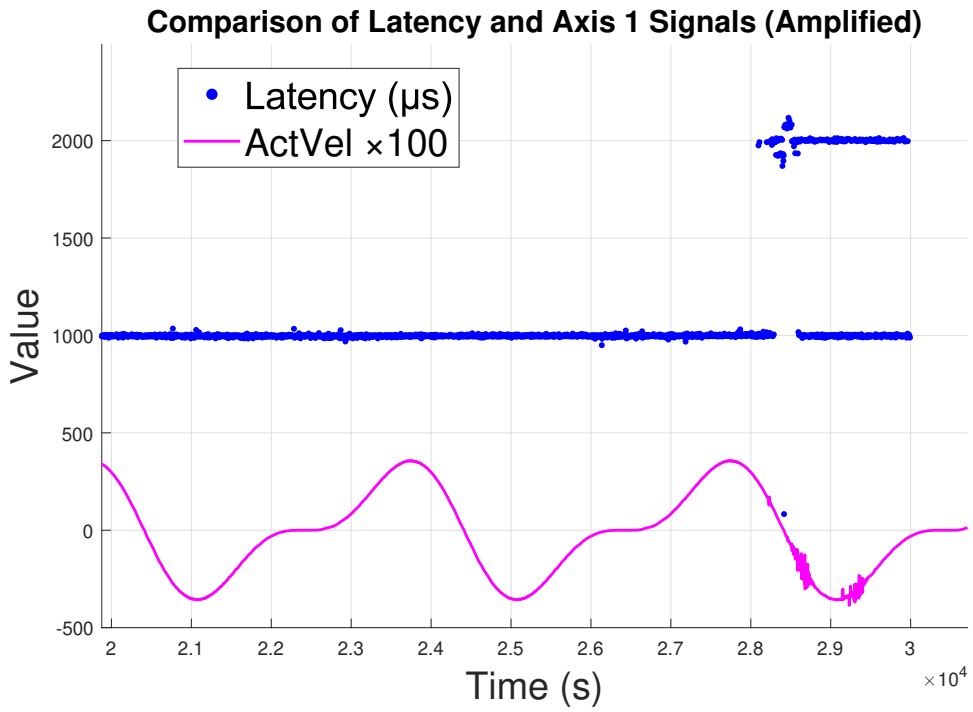

Figure 4.9: Impact of Communication Latency on joint Stability and Robot Dynamics

4.2 Robot Teleoperation Performance using UDP

Following significant issues observed with teleoperation over ADS communication—specifically vibrations caused by latency fluctuations—a protocol with better latency handling was adopted. The objective was to reduce jitter induced by communication delays, ensuring that delayed packets would not overwrite newer data, thereby minimizing the maximum error and delay in position commands. This new implementation resulted in the improved performance presented below.

As with the ADS case, the first teleoperation test was conducted to evaluate the overall behavior of the robot during remote operation. For clarity and comparison, a Cartesian position signal is plotted.4.10.

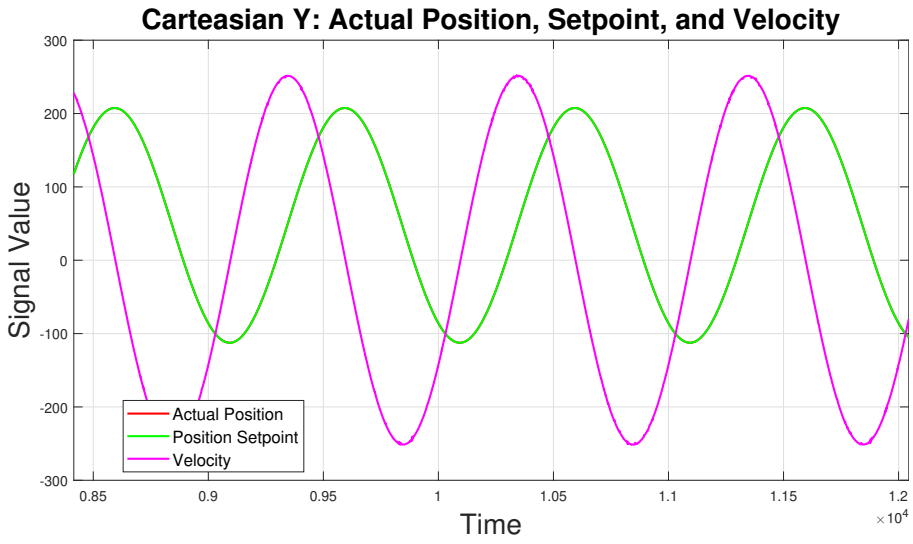

Figure 4.10: Robotic Cartesian Teleoperation Movement with UDP

As seen in the figure, the behavior is significantly improved compared to the ADS-based communication. No unexpected vibrations are present throughout the trajectory execution. To better quantify this improvement, the absolute error between the position setpoint and the actual position is also plotted 4.11.

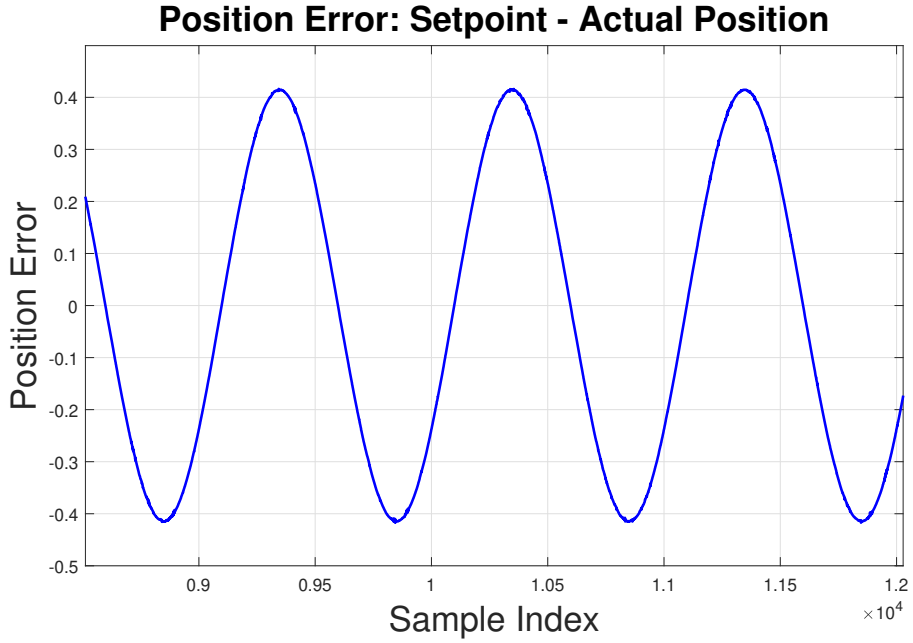

Figure 4.11: Absolute Cartesian Positioning Error

The error plot confirms the absence of undesired vibrations, with the tracking error consistent with the robot's inherent response limitations. To verify that the smooth behavior observed in Cartesian space is also present at the joint level, the joint position setpoint, actual joint position, and joint velocity for the first axis were plotted. This facilitates a more intuitive analysis 4.12.

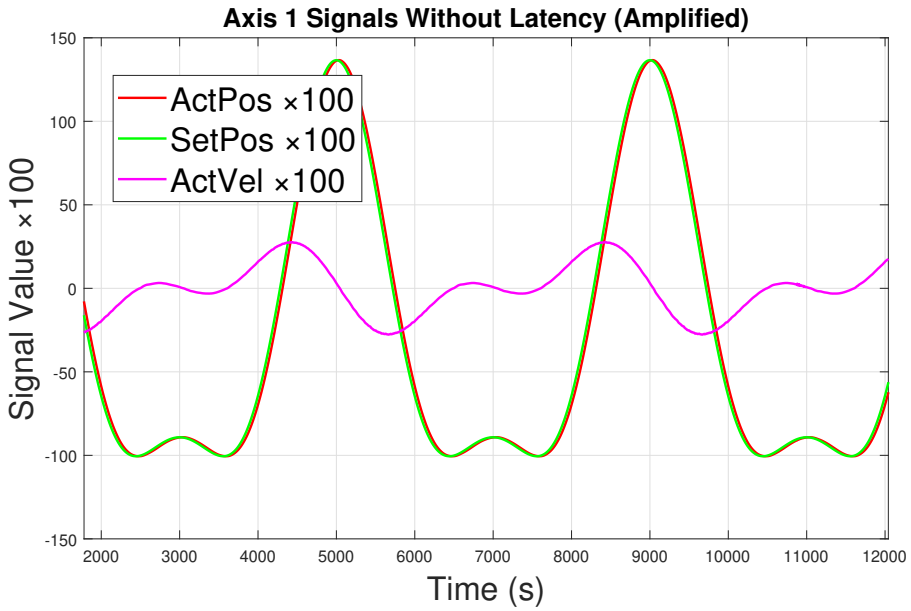

Figure 4.12: Joint 2 – Position Setpoint, Actual Position, and Velocity

It is clearly observable that the performance is greatly improved compared to the ADS case. The joint follows the setpoint without introducing any undesired noise or oscillatory behavior, thus eliminating the erratic vibrations previously observed.

4.3 Force Feedback Performance

This section presents the results obtained from the force measurements provided by the force sensor. As previously indicated, the sensor measures forces along the X, Y, and Z axes. Initially, raw data from the sensor is recorded, as shown in the following figure 4.13.

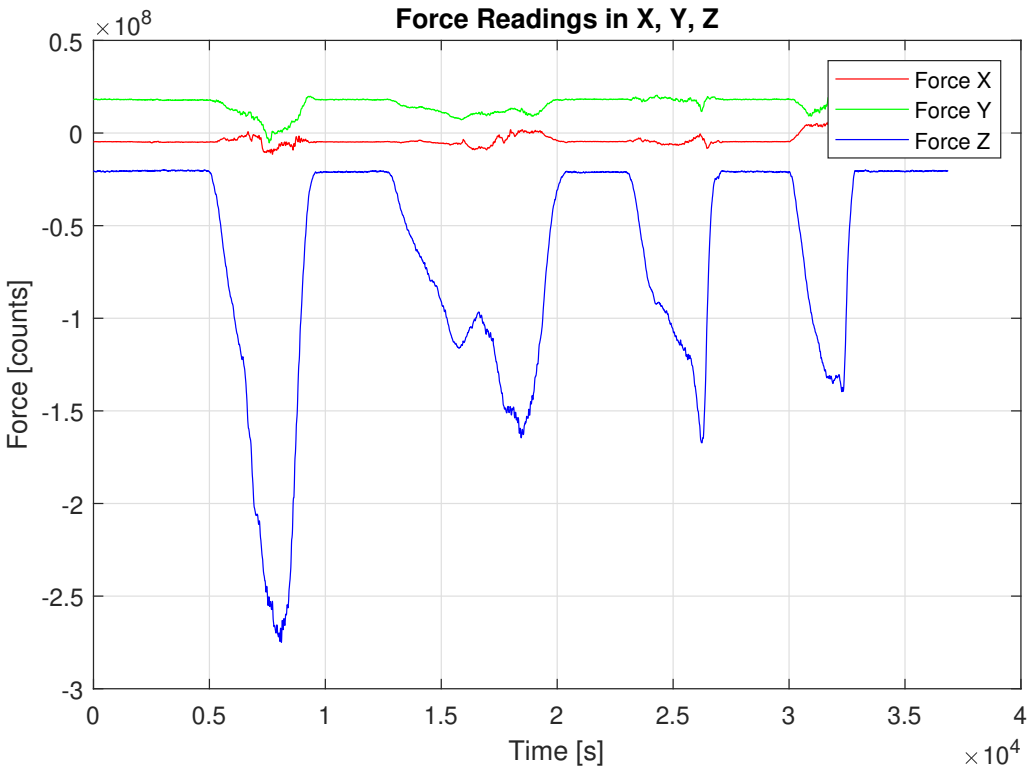

Figure 4.13: Raw Force Reading in the EtherCAT Bus

The Figure shows how the raw signal from the sensor is captured via the EtherCAT bus. These raw counts are then converted into Newtons using the sensor's calibration factors. Once the actual force measurement is obtained, our C++ program transforms the force vector into the coordinate frame of the Omega.7. Finally, a gain is applied to scale the force feedback, depending on the desired haptic experience for the user.

Although the force sensor shares the same EtherCAT bus as the uniVAL Drive, it can be assigned to a faster task—just like the UDP data reception task. This allows the force feedback loop to be closed at the same high frequency, ensuring more responsive and realistic haptic interaction.

5 Analysis and Discussion

This section analyzes the results presented previously, comparing the expected outcomes with the actual findings and discussing possible reasons for any discrepancies.

5.1 Impact of Communication on System Performance

The primary cause of the issues in the teleoperation system is the use of a non-deterministic communication protocol, which does not guarantee the transmission of position commands at fixed time intervals. Consequently, the observed system dynamics are mainly attributed to the latency between the Omega.7 device and the PLC. The main PLC program, Main, reads the displacement variable precisely every 4 ms. However, as shown in the ADS latency graphs, write operations to this variable occur predominantly every 1 ms, with occasional increases to 2 ms.

To analyze the issue in greater detail, an explanation is provided using a specific example. It has been observed that when the ADS communication latency remains at 1 ms, the trajectory follows a smooth and consistent path. The following describes the behavior under these conditions 5.1.

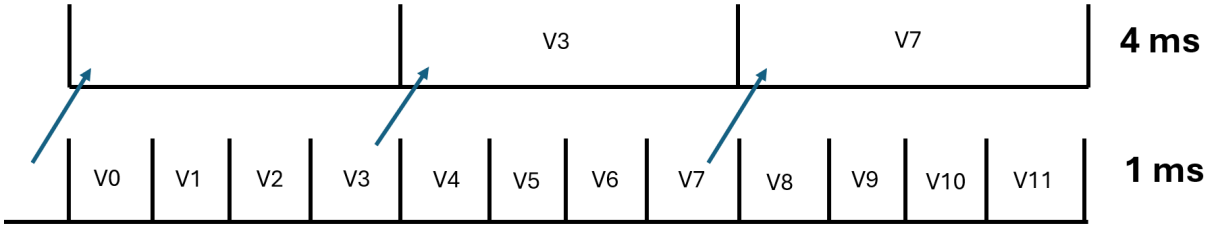

Figure 5.1: Example of an ideal ADS communication where PLC reads position commands (V) with 4ms cycle task

On the other hand, when the latency increases to 2 ms, the PLC may end up reading a position command that is twice as old as when the latency is 1 ms 5.2. This causes a delay in the position command.

This delayed command can have a significant impact on the robot's dynamics. The controller expects to receive real-time position updates to maintain smooth and accurate movement. When the position command is delayed, the controller acts on outdated information, leading to discrepancies between the actual position and the desired trajectory 5.3. Because the setpoint is older, the difference between the current robot position and the delayed target is larger, which results in a larger correction. This forces the controller to apply a higher velocity to reach the

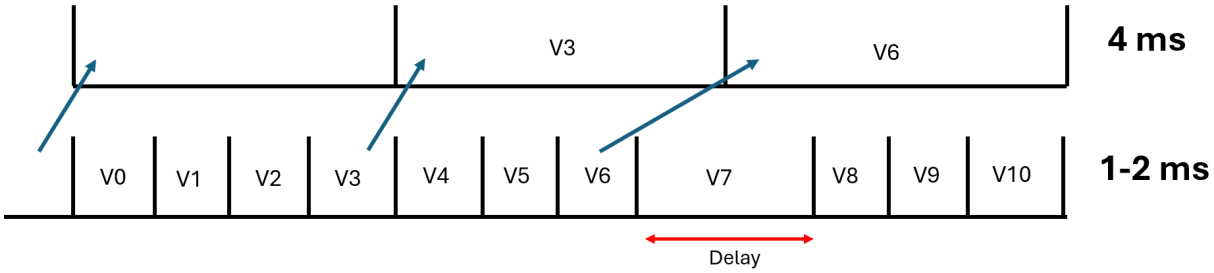

Figure 5.2: Example of a delayed ADS communication where PLC reads position commands (V) with 4ms cycle task

new setpoint in time, potentially exceeding the system's natural dynamics. As a result, the robot may experience sharp movements or oscillations, as seen in the plot 5.4. The higher the latency, the greater the required dynamic response, which can lead to instability or erratic behavior.

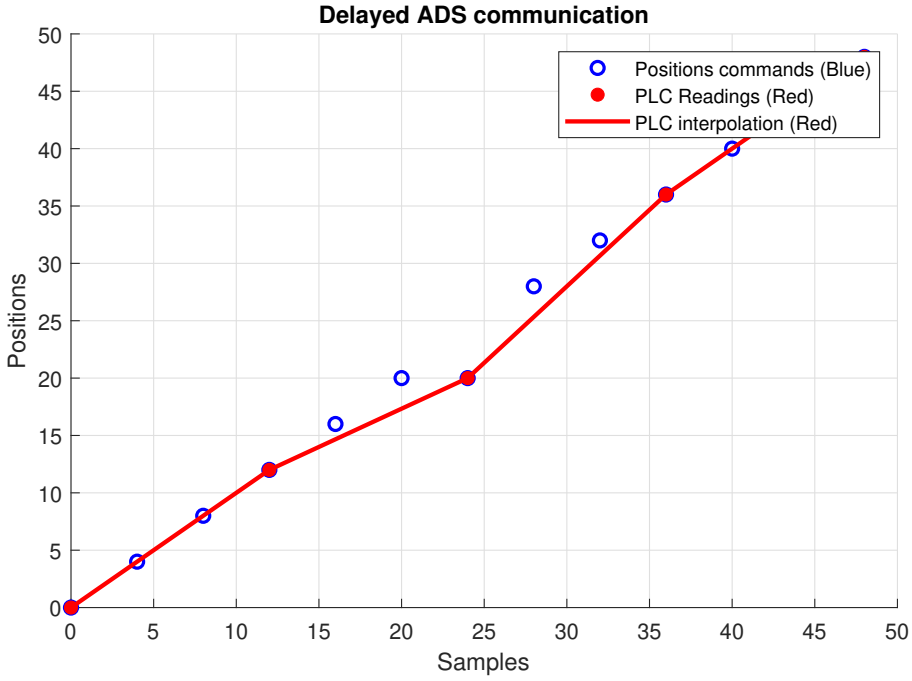

Figure 5.3: Delayed ADS communication

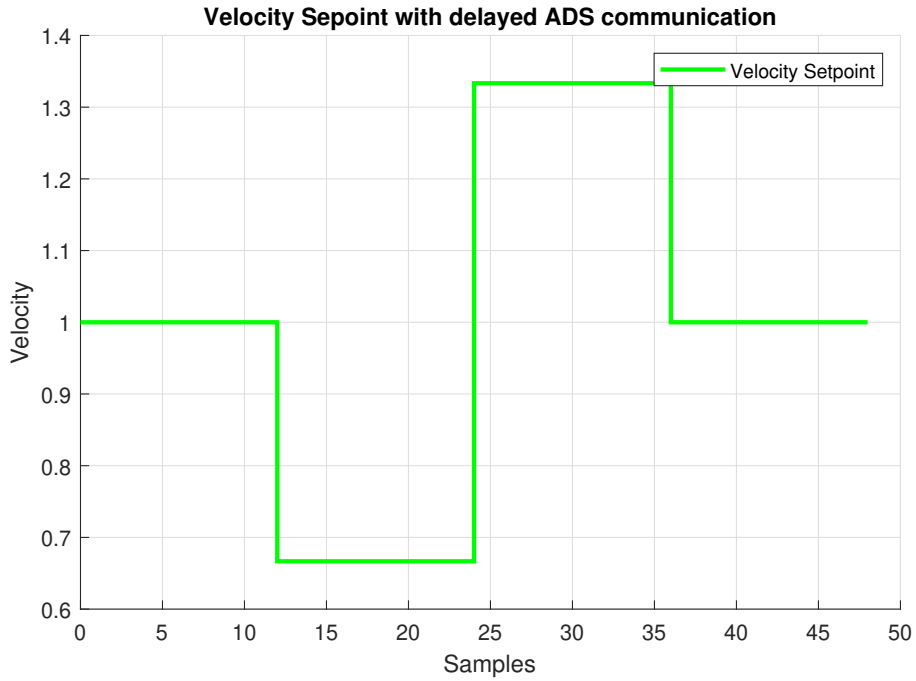

Figure 5.4: Velocity Setpoint with Delayed ADS communication

To address this issue, the proposed solution focuses on reducing latency. Given that some delay will inevitably occur due to the non-deterministic nature of the protocol, the objective is to minimize this delay as much as possible. By doing so, the impact on the setpoint variation is reduced, enabling the robot to manage the system dynamics more effectively.

To achieve this, UDP was chosen as the communication protocol. The idea was to ”flood” the target variable with frequent updates at very low latency, reducing the time between successive position commands. This effectively decreases the variation in the setpoint seen by the PLC during each cycle, making the trajectory smoother and more consistent. With smaller changes between consecutive position values, the internal controller of the robot can better manage the dynamics without being pushed to its limits.

The following analysis was carried out using this approach. It was concluded that increasing the communication frequency of the system—i.e., sending more position setpoints to the PLC within the same time frame—improves the performance of the teleoperation system. An ideal and constant UDP communication, as illustrated below 5.5, offers clear advantages in terms of response time.

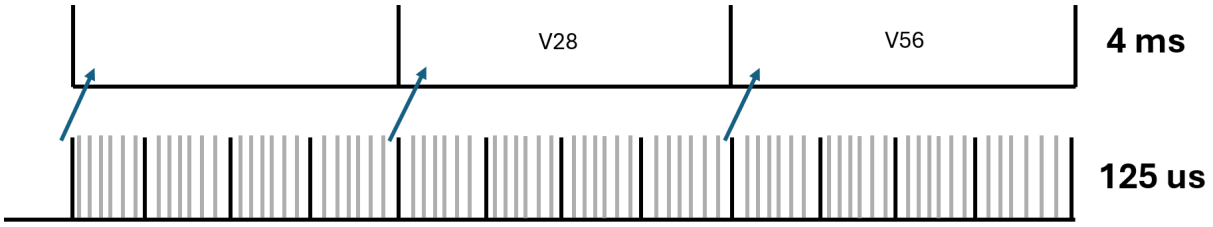

Figure 5.5: Example of an ideal and constant UDP communication

Using this communication protocol, position setpoints can be sent from the C++ program

every 125µs and read by the PLC. The main advantage is that, even when latency is not constant (as the protocol is not deterministic), the effect on the teleoperation system is significantly reduced compared to ADS. The maximum delay that may occur, if a data packet is not received on time, is only 125µs 5.6.

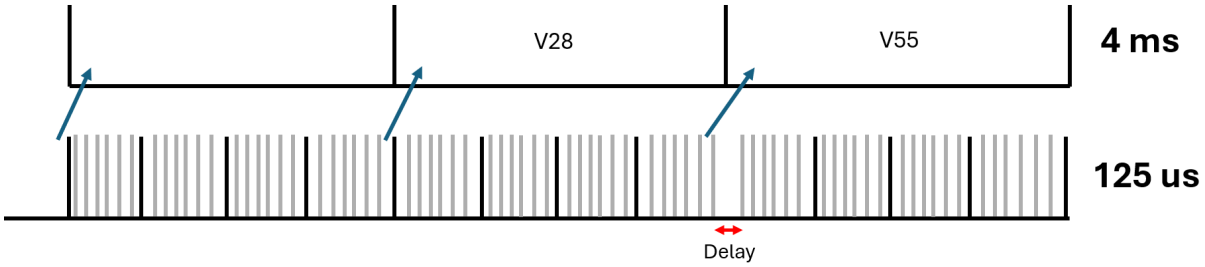

Figure 5.6: Example of a delayed UDP communication

This can be clearly observed in the following figure 5.7, where the position setpoints are shown.

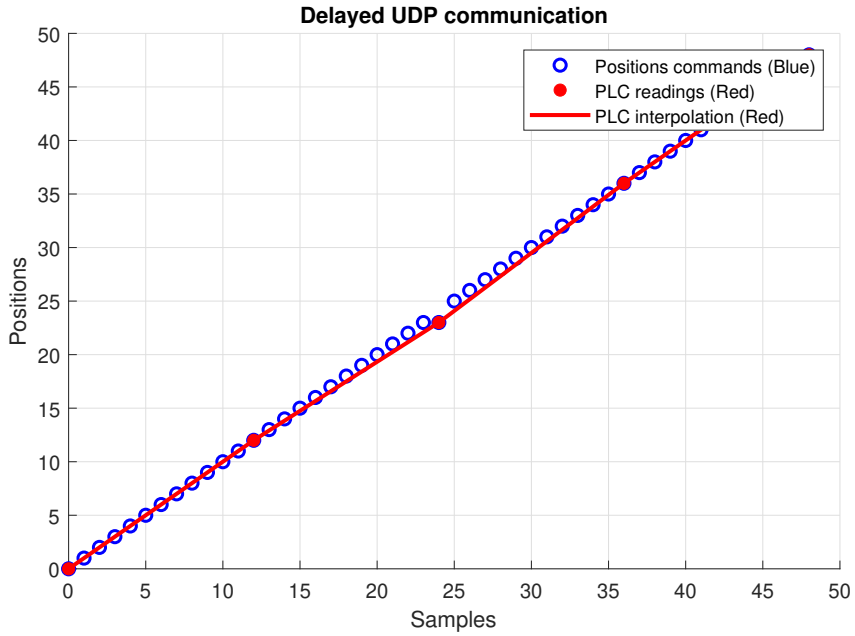

Figure 5.7: Delayed UDP Communicationn

As shown, due to the high communication frequency, the maximum deviation is significantly smaller compared to ADS. Consequently, the dynamic demand on the controller is reduced, as reflected in the velocity setpoint 5.8.

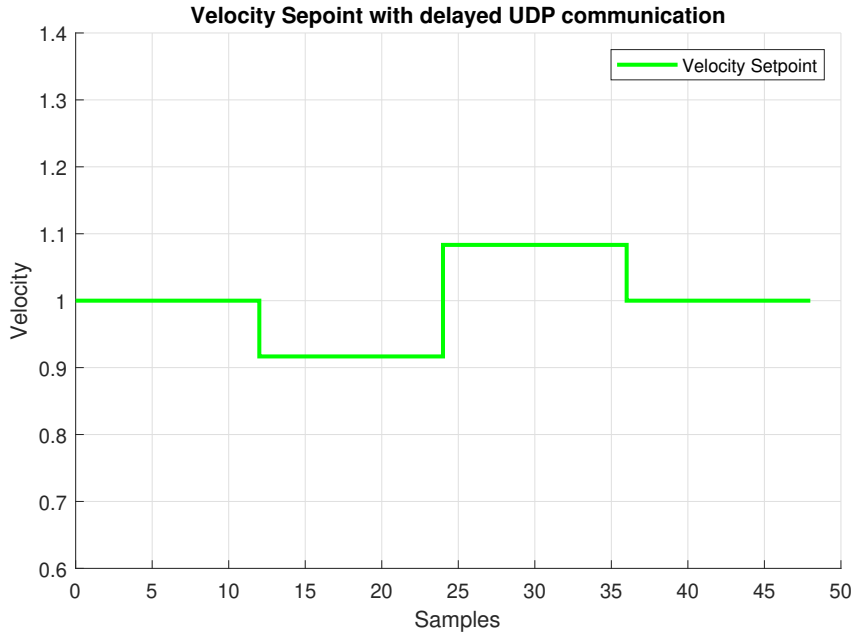

Figure 5.8: Velocity Setpoint with delayed UDP communication

This analysis reveals and quantifies a substantial improvement in the performance of the teleoperation system, particularly in the critical aspect of communication between the two devices.

5.2 Effectiveness of Teleoperation and User Experience

Once the entire system is fully implemented and integrated—including robot control, the position commands generated from the Omega.7 (in our case, through custom code), the communication between the haptic device and the PLC (and consequently, the robot), the force sensor reading and processing, and finally the force feedback applied to the Omega.7—it can be concluded that the system is ready to operate seamlessly once the actual Omega.7 device is available for full integration.

Regarding the improvement of user experience, it is essential that the force feedback aligns with the position commands being sent, as both are closely related. Therefore, it is crucial that the force reading and feedback tasks operate at the same frequency or with similar latency as the process of sending displacement commands from the haptic device. This synchronization is key to ensuring a realistic and responsive teleoperation experience.

6 Conclusion

In this final section of the project, developed over the past four months, the main conclusions are presented. Throughout the development process, a bilateral teleoperation system with haptic feedback was successfully implemented, effectively integrating all the key components: robot control via the PLC, communication with the emulated haptic device, force acquisition and processing, and position command generation.

The resulting architecture enables a user—such as a surgeon—to control the robot through hand movements using an haptic device. The PLC reads these commands and drives the robot accordingly. At the same time, the PLC receives force data from the robot's interaction with the environment. After processing this information, the corresponding force feedback is sent back to the haptic device, effectively closing the force loop. This enhances both the user experience and the safety of the application.

Given the importance of communication between the two systems (the haptic device and the PLC), one of the main challenges identified was the appearance of vibrations during the robot's motion. These vibrations were analyzed in detail and related to latency issues in the communication between the PLC and the haptic system. As demonstrated, the use of two non-deterministic communication protocols—ADS and UDP—can lead to delays in data transmission.

To address this, the correlation between latency and the robot's motion was thoroughly investigated. This analysis made it possible to mitigate vibrations by improving the frequency and stability of communication, resulting in a much smoother and more stable system performance.

It was also verified that the force data is accurately acquired from the sensor and can be appropriately transformed and scaled to provide realistic haptic feedback to the operator. Thanks to the C++ program, the data received from the Omega.7 and the feedback signals sent from the PLC to the device are correctly processed and transmitted, ensuring coherent and synchronized interaction. This capability enables dynamic adjustment of the user experience based on the specific requirements of the environment or task

Finally, it is concluded that the system architecture is fully prepared for the integration of the physical Omega.7 device. The external PLC-based control approach has proven to be robust, flexible, and scalable, supporting its application in more complex environments or multi-robot systems, both in industrial and research contexts.

7 Future Work

Building upon the foundations established in this thesis, several avenues for future work are proposed to further enhance the teleoperation system and its practical applicability.

The next immediate step is to integrate and validate the physical Omega.7 device within the existing control architecture. While the system has been fully developed and tested with simulated inputs, incorporating the real haptic device will allow for a comprehensive evaluation of system behavior under actual operating conditions.

In parallel, efforts should be directed toward fine-tuning the dynamic response and improving system stability when dealing with more complex or rapid movements. Optimizing the control parameters and refining the communication loop could significantly enhance user experience and accuracy.

Additionally, the current framework offers a strong foundation for exploring multi-robot coordination, using the same PLC-based external control strategy. Such expansion could enable collaborative or parallel robotic tasks, relevant in industrial or research contexts.

Lastly, it is essential to evaluate the system's performance in realistic or application-specific scenarios, such as surgical simulation or precision assembly. These tests would help identify practical limitations, validate usability, and guide the system's adaptation to real-world requirements

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my academic supervisor, Karl Kruusamäe, for his continuous guidance, support, and valuable feedback throughout the development of this thesis.

I also wish to sincerely thank my colleagues Mikel Armendia and Bruno Santamaría for their collaboration, encouragement, and helpful insights during the many stages of this project.

Special thanks go to my industrial supervisor, Diego González, whose practical expertise, direction, and involvement have been instrumental in successfully carrying out the work in a real-world context.

Finally, I would like to acknowledge all the individuals who, in one way or another, contributed their time, advice, or support to help me complete this thesis. Your assistance has been greatly appreciated, and this work would not have been possible without you.

Bibliography

[1] P. Bilancia, J. Schmidt, R. Raffaeli, M. Peruzzini, and M. Pellicciari, "An overview of industrial robots control and programming approaches," *Appl. Sci.*, vol. 13, p. 2582, 2023, academic Editors: Paolo Renna, Ana Martins Amaro, Paulo Nobre Balbis dos Reis and Michele Ambrico. [Online]. Available: https://doi.org/10.3390/app13042582

[2] M. News. (2024) Haptx: Million-dollar investment in full-body vr haptics. Accessed: May 13, 2025. [Online]. Available: https://mixed-news.com/en/haptx-million-dollar-investment-in-full-body-vr-haptics/

[3] Automate.org. (2023) How exoskeletons are being leveraged for more than healthcare. Accessed: May 13, 2025. [Online]. Available: https://www.automate.org/robotics/blogs/how-exoskeletons-are-being-leveraged-for-more-than-healthcare

[4] F. Dimension. (2024) Haptic devices. Accessed: May 13, 2025. [Online]. Available: https://www.forcedimension.com/products

[5] Intuitive Surgical, Inc., "da vinci® 5 surgical system," https://www.intuitive.com/en-us/products-and-services/da-vinci/5, Intuitive Surgical, Inc., 2024, accessed: 2025-05-15.

[6] S. Robotics, "unival drive - controlador de ejes universal para robots industriales," https://www.staubli.com/es/es/robotics/productos/robot-software/uniVAL-drive.html, 2024, accedido el 13 de mayo de 2025.

[7] S. G. . C. KG, "Fte-axia80-dual si-75-4/si-150-8," 2025, accessed: 2025-04-25. [Online]. Available: https://schunk.com/de/en/automation-technology/force/torque-sensors/ft-axia/fte-axia80-dual-si-75-4/si-150-8/p/000000000001392577

[8] Force Dimension, "omega.7 haptic device – force feedback interface," 2017, accessed: April 25, 2025. [Online]. Available: https://www.forcedimension.com/images/doc/specsheet_-_omega7.pdf

[9] M. A. Mashagbeh and M. B. Khamesee, "Unilateral teleoperated master-slave system for medical applications," *IFAC-PapersOnLine*, vol. 48-3, pp. 784–787, 2015.

[10] S. Lichiardopol, "A survey on teleoperation," Technische Universiteit Eindhoven, Eindhoven, Tech. Rep. DCT rapporten; Vol. 2007.155, December 2007.

[11] R. V. Patel, S. F. Atashzar, and M. Tavakoli, "Haptic feedback and force-based teleoperation in surgical robotics," *Proceedings of the IEEE*, vol. 110, no. 7, pp. 1012–1027, 2022.

[12] P. Kazanzides, B. P. Vagvolgyi, W. Pryor, A. Deguet, S. Leonard, and L. L. Whitcomb, "Teleoperation and visualization interfaces for remote intervention in space," *Frontiers in Robotics and AI*, vol. 8, 2021. [Online]. Available: https://doi.org/10.3389/frobt.2021.747917

[13] K. Wan, C. Li, F.-S. Lo, and P. Zheng, "A virtual reality-based immersive teleoperation system for remote human-robot collaborative manufacturing," *Manufacturing Letters*, vol. 41, pp. 43–50, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2213846324000658

[14] R. Raffaeli, P. Bilancia, F. Neri, M. Peruzzini, and M. Pellicciari, "Engineering method and tool for the complete virtual commissioning of robotic cells," *Applied Sciences*, vol. 12, no. 6, p. 3164, 2022. [Online]. Available: https://doi.org/10.3390/app12063164

[15] E. Estévez, A. Sánchez-García, J. Gámez-García, J. Gómez-Ortega, and S. Satorres-Martínez, "A novel model-driven approach to support development cycle of robotic systems," *International Journal of Advanced Manufacturing Technology*, vol. 82, pp. 737–751, 2016. [Online]. Available: https://doi.org/10.1007/s00170-015-7396-4

[16] M. Wojtynek, J. J. Steil, and S. Wrede, "Plug, plan and produce as enabler for easy workcell setup and collaborative robot programming in smart factories," *KI - Künstliche Intelligenz*, vol. 33, pp. 151–161, 2019. [Online]. Available: https://doi.org/10.1007/s13218-019-00595-0

[17] Z. Salcic, U. D. Atmojo, H. Park, A. T.-Y. Chen, and K. I.-K. Wang, "Designing dynamic and collaborative automation and robotics software systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 540–549, 2019.

[18] H. Fischer, M. Vulliez, P. Laguillaumie, P. Vulliez, and J. P. Gazeau, "Rtrobmultiaxiscontrol: A framework for real-time multiaxis and multirobot control," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 3, pp. 1205–1217, 2019.

[19] D. Gonzalez and M. Armendia, "Nc controlled robot for adaptive and constant force 3d polishing," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022, pp. 1–7.

[20] A. Mandiola, M. Armendia, D. Gonzalez, and J. Eguskiza, "Dynamic accuracy optimization for nc controlled industrial robots," in *Proceedings of the IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*. Sinaia, Romania: IEEE, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10275340

[21] H. Ji, S. Li, J. Wang, and Z. Ruan, "Improving teleoperation through human-aware haptic feedback: A distinguishable and interpretable physical interaction based on the contact state," *IEEE Transactions on Human-Machine Systems*, vol. 53, no. 1, pp. 24–34, 2023.

[22] G. Picod, A. C. Jambon, D. Vinatier, and P. Dubois, "What can the operator actually feel when performing a laparoscopy?" *International Journal of Computer Assisted Radiology and Surgery*, vol. 1, no. 1, pp. 45–50, 2005.

[23] U. Seibold, B. Kubler, and G. Hirzinger, "Prototype of instrument for minimally invasive surgery with 6-axis force sensing capability," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 496–501.

[24] J. Singh, A. R. Srinivasan, G. Neumann, and A. Kucukyilmaz, "Haptic-guided teleoperation of a 7-dof collaborative robot arm with an identical twin master," *IEEE Transactions on Haptics*, vol. 13, no. 1, pp. 246–252, 2020.

[25] H. Cheng, R. Jia, D. Li, and H. Li, "The rise of robots in china," *Journal of Economic Perspectives*, vol. 33, no. 2, p. 71–88, May 2019. [Online]. Available: https://www.aeaweb.org/articles?id=10.1257/jep.33.2.71

[26] E. Oztemel and S. Gursev, "Literature review of industry 4.0 and related technologies," *Journal of Intelligent Manufacturing*, vol. 31, 01 2020.

[27] A. D. Pham and H.-J. Ahn, "Rigid precision reducers for machining industrial robots," *International Journal of Precision Engineering and Manufacturing*, vol. 22, p. 1469–1486, 08 2021.

[28] "Addressing regulatory considerations for medical robotic devices," UL Solutions, Tech. Rep., 2017, accessed: 2025-04-24. [Online]. Available: https://code-authorities.ul.com/wp-content/uploads/sites/40/2017/08/BNG-UL17-Medical-Robots-White-Paper-080117-1.pdf

[29] F. De Micco, S. Grassi, L. Tomassini, G. Di Palma, G. Ricchezze, and R. Scendoni, "Robotics and ai into healthcare from the perspective of european regulation: who is responsible for medical malpractice?" *Frontiers in Medicine*, vol. 11, p. 1428504, Sep. 2024.

[30] "Safety testing in healthcare robotics," UL Solutions, Tech. Rep., 2019, accessed: 2025-04-24. [Online]. Available: https://collateral-library-production.s3.amazonaws.com/uploads/asset_file/attachment/17911/BNG-UL19-Robotics-WP-112219.pdf

[31] M. Bonfe, M. Vignali, and M. Fiorini, "Plc-based control of a robot manipulator with closed kinematic chain," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 1262–1267.

[32] M. Fatriyana, "Cnc program and programming of cnc machine," *Journal of Mechanical Science and Engineering*, 2020.

[33] M. Hasan, "Computer numerical control machines: An account of programming methods and techniques," *Journal of Material Science and Mechanical Engineering*, vol. 2, pp. 14–17, 08 2015.

[34] Wikipedia contributors, "Haptic technology," https://en.wikipedia.org/wiki/Haptic_technology, 2025, accessed: 2025-04-23.

[35] S. Laycock and A. Day, "Recent developments and applications of haptic devices," *Computer Graphics Forum*, vol. 22, pp. 117 – 132, 06 2003.

[36] B. Hannaford, "A design framework for teleoperators with kinesthetic feedback," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 4, pp. 426–434, 1989.

[37] A. Nasser, K. Zheng, and Z. Kening, "Thermearhook: Investigating spatial thermal haptic feedback on the auricular skin area," 10 2021, pp. 662–672.

[38] M. A. Srinivasan, "What is haptics?" http://touchlab.mit.edu, director, Laboratory for Human and Machine Haptics: The Touch Lab, Massachusetts Institute of Technology.

[39] P. Kemigisha, "Navigating the gender dynamics of food and nutrition security in south-western uganda," *Academia Nutrition and Dietetics*, vol. 2, no. 2, 2025.

[40] Haption, "Virtuose 6d – high performance force-feedback devices," 2025, accessed: 2025-04-23. [Online]. Available: https://www.haption.com/product/virtuose6d

[41] Force Dimension, "Omega haptic devices," https://www.forcedimension.com/products/omega, accessed: 2025-04-23.

[42] SenseGlove, "Senseglove nova 2 - how it works," 2025, accessed: 2025-04-23. [Online]. Available: https://www.senseglove.com/product/nova-2/#howitworks

[43] S. Sood, *Telesurgery*, 04 2006.

[44] A. Ayme, J. Suárez, M. Ortega, G. Gualoto, S. Lima, A. Campoverde, A. Ticona, C. Vergara, and G. Serrano, "Advancements in minimally invasive surgical techniques: A comprehensive review," *Salud, Ciencia y Tecnología*, vol. 3, p. 745, 02 2024.

[45] A. Tobergte, R. Konietschke, and G. Hirzinger, "Planning and control of a teleoperation system for research in minimally invasive robotic surgery," *2009 IEEE International Conference on Robotics and Automation*, pp. 4225–4232, 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:15085084

[46] C. R. Wottawa, B. Genovese, B. N. Nowroozi, S. D. Hart, J. W. Bisley, W. S. Grundfest, and E. P. Dutson, "Evaluating tactile feedback in robotic surgery for potential clinical application using an animal model," *Surgical Endoscopy*, vol. 30, no. 8, pp. 3198–3209, Aug. 2016, epub 2015 Oct 30.

[47] C. Freschi, V. Ferrari, F. Melfi, M. Ferrari, F. Mosca, and A. Cuschieri, "Technical review of the da vinci surgical telemanipulator," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 9, no. 4, pp. 396–406, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rcs.1468

[48] M. Kitagawa, D. Dokko, A. M. Okamura, and D. D. Yuh, "Effect of sensory substitution on suture-manipulation forces for robotic surgical systems," *The Journal of Thoracic and Cardiovascular Surgery*, vol. 129, no. 1, pp. 151–158, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022522304008876

[49] E. Westebring, R. Goossens, J. Jakimowicz, and J. Dankelman, "Haptics in minimally invasive surgery - a review," *Minimally invasive therapy allied technologies : MITAT : official journal of the Society for Minimally Invasive Therapy*, vol. 17, pp. 3–16, 02 2008.

[50] A. Tobergte, P. Helmer, U. Hagn, P. Rouiller, S. Thielmann, S. Grange, A. Albu-Schaffer, F. Conti, and G. Hirzinger, "The sigma.7 haptic interface for mirosurge: A new bi-manual surgical console," 09 2011.

[51] J. Rosen, M. MacFarlane, C. Richards, B. Hannaford, and M. Sinanan, "Surgeon-tool force/torque signatures - evaluation of surgical skills in minimally invasive surgery," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 774–783, 2001, department of Electrical Engineering, University of Washington.

[52] D. G. Black, D. Andjelic, and S. E. Salcudean, "Evaluation of communication and human response latency for (human) teleoperation," *IEEE Transactions on Medical Robotics and Bionics*, vol. 6, no. 1, pp. 53–63, 2024.

[53] Beckhoff Automation GmbH & Co. KG, *TwinCAT — Automation software*, 2025, accessed: 2025-04-24. [Online]. Available: https://www.beckhoff.com/en-us/products/automation/twincat/

[54] ——, *TwinCAT 3 Motion - TF5xxx Function Libraries*, 2024, accessed: 2025-04-24. [Online]. Available: https://www.beckhoff.com/en-us/products/automation/twincat/tfxxxx-twincat-3-functions/tf5xxx-motion/

[55] ——, *TF6310 — TwinCAT 3 TCP/UDP Realtime*, 2025, accessed: 2025-04-24. [Online]. Available: https://www.beckhoff.com/en-us/products/automation/twincat/tfxxxx-twincat-3-functions/tf6xxx-connectivity/tf6311.html

[56] ——, *TC1000 — TwinCAT 3 ADS*, 2025, accessed: 2025-04-24. [Online]. Available: https://www.beckhoff.com/en-us/products/automation/twincat/tc1xxx-twincat-3-base/tc1000.html

[57] K. van Teeffelen, "Intuitive impedance modulation in haptic control using electromyography," Master's thesis, University of Twente, Enschede, The Netherlands, January 2018, mSc Thesis, Robotics and Mechatronics Group, EE-Math-CS. [Online]. Available: https://essay.utwente.nl/74440/

Non-exclusive licence to reproduce thesis and make thesis public

I, Asier Mandiola Arrizabalaga

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   "Haptic-Based Teleoperation of Robots"

   supervised by Karl Kruusamäe and Diego Gonzalez.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Asier Mandiola Arrizabalaga
15.05.2025