Mixed Reality Teleoperation Platform for Experiential Manufacturing Education

Dr. Israa Azzam, Milwaukee School of Engineering

Dr. Azzam is an Assistant Professor of Mechanical Engineering at the Milwaukee School of Engineering (MSOE). Her teaching philosophy centers on creating an inclusive, learner-focused environment that emphasizes experiential learning, teamwork, and innovation. Dr. Azzam's goal is to integrate multiple teaching techniques and state-of-the-art methods to help students succeed. A major part of her approach is using advanced technologies such as Extended Reality (XR) in lectures and labs. She aims to bridge the gap between theory and practice, equipping students with essential STEM skills to thrive as innovators and future leaders. Dr. Azzam earned her Ph.D. in Mechanical Engineering Technology from Purdue University in 2025, where she specialized in digital technologies and control systems. Her doctoral research focused on industrial teleoperations, XR technology, and engineering education, and has been recognized with multiple awards. Before pursuing her Ph.D., Dr. Azzam completed her B.S. in Mechanical Engineering at Beirut Arab University (2019) and her M.E. in Mechanical Engineering at the American University of Beirut (2021). Her graduate work at AUB provided a strong theoretical foundation in control systems, which she expanded into applied research at Purdue, where she collaborated with industry partners on simulation-based and XR-integrated modules for engineering education.

Khalid Bello, University of Louisville

Khalid Bello is a graduate student at University of Louisville.

Dr. Farid El Breidi, Purdue University - Purdue Polytechnic Institute – West Lafayette

Dr. Farid El Breidi is an Assistant Professor in the School of Engineering Technology at Purdue University. Dr. El Breidi is the director of the Digital Technologies Lab and is actively engaged in research encompassing digital fluid power, digital reality, and engineering pedagogy. He serves as the Principal Investigator for numerous internally and externally funded research projects. These projects have received support from agencies and associations such as the National Science Foundation and the National Fluid Power Association.

Dr. Faisal Aqlan, University of Louisville

Dr. Faisal Aqlan is an Associate Professor of Industrial Engineering at The University of Louisville. He received his Ph.D. in Industrial and Systems Engineering form The State University of New York at Binghamton.

Mixed Reality- Enabled Teleoperation Platform for Experiential Manufacturing Education

Abstract

Mixed Reality (MR) technologies are redefining the way humans interact with physical systems by merging real and digital environments. Research shows that human operators perform tasks up to 66% faster in immersive 3D environments such as MR compared to conventional 2D interfaces. Translating this capability into manufacturing education presents an opportunity to enhance experiential learning. When applied to educational contexts, these immersive environments can improve students' learning experiences, deepen conceptual understanding, and provide hands-on practice in safe and accessible conditions.

Building on this potential, the present work leverages industrial teleoperation technologies within an MR-enabled digital twin framework to develop an interactive training platform for manufacturing education. The proposed platform enables real-time communication between a physical six-degree-of-freedom (6-DOF) robotic arm and its virtual counterpart to create an interactive MR environment for hands-on learning and control in manufacturing education.

This study focuses on the development of the MR-Enabled Teleoperation platform and on the design and integration of advanced control algorithms to minimize communication latency and ensure real-time synchronization between the physical and virtual systems. Different control strategies (i.e., event-driven, adaptive smoothing, and predictive control) were designed and tested, with the best strategy achieving an average latency below 10 milliseconds. Compared to prior work reporting latencies between 50-100 ms, the achieved 10 ms delay represents an improvement for control accuracy and system stability. These control protocols enabled a responsive digital twin environment that accurately replicates real-world robotic behavior.

Beyond its technical foundation, this work demonstrates the platform's significant educational potential in manufacturing training. It discusses the platform's future potential to enable students to visualize, control, and analyze robotic operations in a safe, immersive MR environment, gaining hands-on experience with industrial systems without requiring direct physical access. The system supports interactive instruction in manufacturing processes, control theory, and human-robot collaboration. Further, it highlights the future potential of MR-enabled teleoperation systems for supporting manufacturing education by providing experiential learning opportunities that bridge theory and practice, enhance systems-level understanding, and prepare students for Industry 4.0 and smart manufacturing applications.

Keywords: Mixed Reality (MR), Teleoperation, Digital Twin, Robotics, Manufacturing Education, Industry 4.0

1. Introduction

Extended Reality (XR) technologies, including Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR), have been shown to enhance human-system interaction by improving spatial awareness, perceptual feedback, and task performance [1], [2]. Studies and applications of XR technologies across multiple domains have further demonstrated that MR offers unique

advantages over VR and AR, particularly in applications that require real-time interaction between physical systems and digital representations [3]. While VR provides fully immersive virtual environments and AR overlays digital content onto the real world, MR enables the coexistence and interaction of real and virtual elements within a shared spatial environment [4], [5]. Unlike VR, which isolates users from their surroundings, and AR, which provides limited interaction with virtual content, MR allows digital objects to be spatially anchored, context-aware, and responsive to physical environments and user actions [6]. These characteristics make MR suitable for applications that require continuous interaction between humans and physical systems.

With all these distinguishing features, including spatial mapping, environmental awareness, gesture-based interaction, and real-time rendering of virtual objects within physical space, MR technology began to be used across a range of industrial fields. [7]. MR has been increasingly applied in manufacturing, construction, maintenance, and engineering design to support tasks such as equipment assembly, process visualization, remote inspection, and operator training [8], [9]. Among these applications, MR has played a critical role in advancing human-robot interaction (HRI), particularly in teleoperation and digital twin-based systems. Teleoperation involves the remote control of physical systems, often in environments that are unsafe, inaccessible, or impractical for direct human presence [10]. Digital twins extend this concept by providing a virtual replica of a physical system that reflects its structure, state, and behavior in real time [11], [12]. MR reinforces these two frameworks by merging real and digital environments, which enables operators to interact with virtual representations of physical systems as if they were co-located. This integration enhances operator intuition, reduces cognitive load, and supports more natural control strategies compared to conventional 2D interfaces.

Research has demonstrated that immersive MR can improve human performance in teleoperation and manipulation tasks. Experimental results reported in prior work indicate that human operators can complete tasks up to 66% faster when using immersive 3D environments (MR-based interfaces), compared to traditional 2D screen-based systems [13], [14], [15]. These performance improvements are attributed to MR's ability to enhance alignment between operator intent and system response, enabling more intuitive and accurate interaction with physical systems. However, despite these advantages, many of the existing MR-enabled teleoperation platforms exhibit technical limitations, including communication delays and system latency. Recent studies have reported end-to-end latencies on the order of 50 ms, and in some applications exceeding 100 ms, which can affect control accuracy, system stability, and user experience [16], [17], [18].

These latency-induced challenges motivate the need for more responsive and reliable MR-based teleoperation systems. The main goal of this work is to develop an MR-enabled teleoperation platform that integrates advanced control strategies designed to mitigate communication delays and synchronization errors between physical and virtual systems. Beyond industrial applications, this work also aims to translate the platform's capabilities into manufacturing education by taking the first step toward developing smart manufacturing laboratories aligned with modern teleoperation practices.

To this end, this study investigates the following research questions:
RQ1: How to design an MR-enabled teleoperation platform integrating advanced control strategies to minimize communication latency while maintaining precise synchronization between the physical system and its digital twin?

RQ2: How can the MR-enabled teleoperation platform serve as a foundation for experiential learning in smart manufacturing laboratories?

The rest of this paper is organized as follows. Section 2 reviews existing smart industrial manufacturing laboratories, highlighting their advantages over traditional laboratories and discussing some technical and instructional challenges. Section 3 presents the development stages of the MR-enabled teleoperation platform, from system architecture to final implementation. Section 4 describes the design and integration of advanced control protocols to mitigate communication latency and improve system performance. Sections 5 and 6 present the experimental results and corresponding discussion. Finally, Section 7 concludes the paper and outlines directions for future work.

2. Smart Manufacturing Laboratories for Experiential Learning

Smart manufacturing laboratories have emerged to support experiential learning in manufacturing and industrial education. These laboratories integrate digital technologies, such as advanced visualization, automation, and cyber–physical system integration, to mimic real-world manufacturing environments within controlled instructional settings [19]. Unlike traditional laboratories that often emphasize isolated experiments or predefined procedures, smart manufacturing laboratories enable students to interact with interconnected systems, fostering a deeper understanding of manufacturing processes and control system analysis & design [20]. The following subsections present the educational potential of smart manufacturing laboratories and discuss the existing technical and instructional challenges that limit their effectiveness.

2.1. Educational Potential of Smart Manufacturing Laboratories

From an educational perspective, such environments can enhance student learning by improving conceptual understanding, system-level thinking, and hands-on engagement. Immersive and interactive platforms allow students to visualize abstract concepts, observe cause-and-effect relationships in real time, and experiment with system behavior under varying conditions [21]. Moreover, smart manufacturing laboratories offer safe and accessible learning environments, where complex or potentially hazardous industrial operations can be explored without exposing students to physical risks or requiring continuous access to costly equipment [22].

Given these benefits, several academic institutions have begun developing smart manufacturing laboratories, with many initiatives relying primarily on VR-based simulations or standalone digital training platforms [20], [23], [24], [25], [26], [27]. For example, Purdue University's Smart Learning Factory emphasizes Industry 4.0 instruction through interconnected automation systems, digitalized material flow, and data-driven monitoring, providing students with hands-on exposure to cyber–physical production concepts. Similarly, TU Darmstadt's Learning Factory has been extended with digital and virtual representations of production systems, enabling learners to explore factory layout planning, process optimization, and control strategies within simulated environments that closely mirror real industrial scenarios. Arizona State University has

developed VR-based manufacturing and workforce-training environments in which students and trainees practice complex industrial procedures, such as equipment setup and operation, within immersive scenarios that can be safely repeated at low cost. Also, Virginia Tech's Learning Factory integrates advanced automation and digital technologies, exposing students to robotics, control, and instrumentation while supporting project-based learning on realistic manufacturing systems [28].

2.2. Existing Challenges Facing Smart Manufacturing Labs

Despite the educational value of smart manufacturing labs, many existing implementations exhibit technical limitations [29], [30]. For instance, VR-based and simulation-only platforms often lack real-time interaction with physical systems, resulting in weak coupling between virtual models and real-world behavior. Communication delays, system latency, and the absence of synchronized feedback loops have been reported as recurring challenges, especially when limited forms of hardware integration are attempted. These issues reduce system responsiveness, compromise realism, and limit the applicability of such platforms for instruction in teleoperation, real-time control, and digital twin technologies. As a result, current approaches may fall short in preparing students for modern manufacturing environments that increasingly rely on integrated cyber–physical systems [31].

Thus, this work aims to address some of these technical limitations by developing an MR-enabled teleoperation platform that integrates advanced control strategies and can be extended as an interactive training platform for manufacturing education. The platform combines a physics-based virtual replica, real-time communication between Unity and ROS, and predictive closed-loop control to enable responsive teleoperation in an immersive MR environment. The proposed system establishes a technical foundation that supports experiential learning. It bridges theoretical instruction with practical system interaction, preparing students for Industry 4.0 and future smart manufacturing systems.

3. MR-Enabled Teleoperation Platform Development

The development of the platform was summarized into three stages: ***(1) Teleoperation Architecture***, ***(2) Platform Design and Implementation,*** and ***(3) System Debugging and Final Configuration.*** The following subsections describe each development stage.

3.1. Teleoperation Architecture

This section presents a *comparative analysis* for selecting a suitable robotic arm for the proof-of-concept platform development, an *overview of the selected robotic system*, and a *brief kinematics and numerical dynamics computation*. It also presents an *overview of the software stack* and *communication protocol* for the platform development.

Comparative Analysis for the Teleoperation System (Robotic Arm) Selection: The development of the MR-enabled teleoperation platform required selecting a robust robotic arm to allow for the proof-of-concept platform implementation. The selected arm needed to be compatible and accessible for integration and testing. Several options (up to 9 systems) were evaluated based on criteria, including affordability, technical capability, software compatibility, size, and support for education and research applications. Table 1 presents a comparative overview of the robotic arm systems evaluated before leading to the final selection of the Niryo Ned2.

Table 1. Comparative analysis of the robotic arms selection for platform development

Robotic Arm	DOF	Price ($)	Use	Software Compatibility & Working Environment	Description
Baxter Rethink Robot	7	~40,000–45,000	Industrial, Research	• ROS • Intera (Customized Package)	Baxter is a dual-arm robot with two 7-DOF arms, cameras, and force sensors, enabling it to adapt to its environment and respond to changes.
Quanser QArm	4	~22,000	Educational, Research	• MATLAB/Simulink • ROS • Python • QUARC (Customized Package)	Serial manipulator with tendon-based two-stage gripper and RGBD camera.
Quanser Joint Control Robot	4	~18,000–22,000 (based on bundled educational kits)	Educational, Research	• MATLAB/Simulink • ROS • QUARC (Customized Package)	Serial manipulator with a two-stage gripper and an RGBD camera.
Quanser 3DOF Hover System	3	~18,000–22,000 (based on bundled educational kits)	Educational, Research	• MATLAB/Simulink • LabVIEW • QUARC (Customized Package)	Platform for examining flight dynamics and control of vertical lift-off vehicles.
UFACTORY xArm 7	7	~21,000	Industrial, Educational, Research	• ROS • Python • xArm Studio (Customized Package)	Robotic arm with six-axis force-torque sensors, enabling precise measurement of applied forces and torques.
Kinovarobotics Gen3	6-7	~10,000-13,000 (based on bundled educational kits)	Educational, Light Industrial Tasks, Research	• Arduino • MATLAB • ROS • Python • Gazebo • MoveIt • C++ • Kinova Kortex (Customized Package)	Robot offers high precision, modularity, and real-time control tasks.
Dobot Magician & Magician E6	4-6	~6,995	Educational, Light Industrial Tasks	• Arduino • ROS • Python • DobotStudio (Customized Package)	A desktop robotic arm for hands-on training, supporting 3D printing, laser engraving, and related tasks.
MyCobot 320 PI	6	~2,700	Educational, Research, Light Industrial Tasks	• MyBlocky • Python • ROS • C++	Light robot (3.36 kg) with integrated Raspberry Pi for flexible programming and control.
Niryo Ned2	6	~4,000	Educational, Research, Industry 4.0	• ROS • Python • Blocky • Gazebo • Niryo Studio (Customized Package)	The robot supports open-source software and ROS, making it ideal for automation, control, and programming in academic and prototyping settings.

The selection of the robotic arm for developing the XR-enabled teleoperation system was based on a decision matrix incorporating five evaluation criteria. Each criterion was rated on a scale from 1 (lowest) to 5 (highest), and the total score was used to guide the selection process. Table 2 presents the decision matrix used in the evaluation. Based on the decision matrix, the Niryo Ned2 robotic arm was selected for developing the MR-enabled teleoperation platform, as it achieved the highest overall score (21). This selection was primarily driven by its affordability, technical compatibility, and suitability for academic research. The Niryo Ned2 is a 6-DOF robotic arm priced at approximately $4,000, which falls within the project's budget constraint of $5,000 for the robotic system. It is designed for education, research, and Industry 4.0 prototyping and is supported by extensive documentation, tutorials, and an active user community. From a technical perspective, the Niryo Ned2 supports ROS and Python, ensuring seamless integration with the MR development framework. It is also compatible with simulation and motion-planning tools such as Gazebo and MoveIt, which are essential for modeling, testing, and validation. In addition, its moderate size and light weight (approximately 7 kg) make it practical for laboratory use and easier to integrate into an MR environment compared to larger industrial systems.

Alternative platforms, including Baxter, Quanser, and xArm 7, offer industrial-grade capabilities but exceed the project's budget, with costs ranging from $18,000 to over $40,000. Other options, such as the Kinova Gen3 and Dobot Magician E6, were more affordable but presented limitations related to cost-effectiveness, size, or integration complexity.

Thus, the Niryo Ned2 provides a suitable platform for developing an MR-enabled teleoperation system. Its open-source ecosystem, software compatibility, and educational focus align well with the objectives of this research. The primary goal is to develop an MR-enabled teleoperation platform that integrates advanced control strategies to mitigate communication delays and synchronization errors between physical and virtual systems. Following platform development and validation, the system is intended to be translated into manufacturing education as an initial step toward smart manufacturing laboratories aligned with modern teleoperation practices. Thus, the educational orientation of the Niryo Ned2 supports the application of the MR-enabled platform in experiential manufacturing education.

Table 2. Decision matrix for robotic arm selection

Robot Platform	**Affordability**	**Research Application**	**Software Compatibility**	**Physical Suitability**	**Available Resources**	**Sum**
Baxter Rethink Robot	1	5	3	2	1	12
Quanser QArm	2	4	3	3	3	15
Quanser Joint Control	2	3	3	3	3	14
Quanser 3DOF Hover	2	1	3	3	3	12
UFACTORY xArm 7	2	4	3	3	2	14
Kinovarobotics Gen3	3	3	4	4	2	16
Dobot Magician E6	4	3	4	5	2	18
MyCobot 320 PI	5	2	2	5	1	15
Niryo Ned2	**5**	**4**	**4**	**5**	**3**	**21**

Overview of the Selected Niryo Ned2 Robotic Arm: The information presented in this section is based on official documentation provided by Niryo [32], [33], [34]. The Niryo Ned2 is a serial 6-degree-of-freedom (6-DOF) robotic manipulator designed for education, research, and Industry 4.0 prototyping. It has six revolute joints, corresponding to base rotation, shoulder pitch, elbow pitch, forearm pitch, wrist yaw, and wrist roll, enabling a wide range of manipulation tasks such as picking, placing, rotating, and aligning components (see Figure 1).

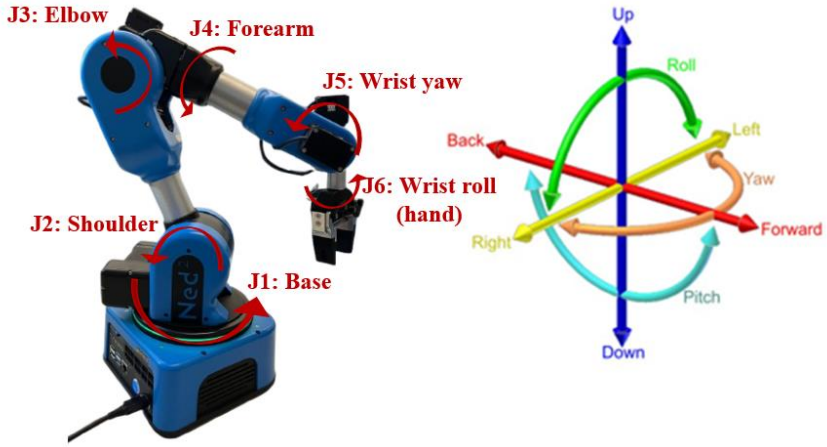

Figure 1. Joint configuration of the Niryo Ned2 robotic arm and the corresponding rotational axes (yaw, pitch, and roll).

From a hardware perspective, the robot weighs approximately 7 kg, has a reach of 440 mm, and supports a payload of up to 300 g. It offers a repeatability of ±0.5 mm and supports a Tool Center Point (TCP) speed of up to 0.468 m/s. The system is constructed for tabletop operation and supports modular end-effectors, including customizable grippers, vacuum tools, electromagnets, and vision-based attachments, which can be exchanged using a magnetic tool-mounting mechanism. The robot is controlled by a Raspberry Pi 4 platform and provides multiple wired and wireless communication interfaces, including Ethernet, Wi-Fi, and USB, along with digital and analog I/O for real-time data exchange.

In terms of software and communication protocols, the Niryo Ned2 supports multiple programming and control interfaces, including *Niryo Studio, PyNiryo (Python), ROS (Melodic and Noetic), Blockly, MATLAB, and Modbus TCP*. These interfaces enable flexible integration with simulation, control, and digital twin environments, making the robot suitable for our MR-enabled teleoperation and control experiments. Additional features such as onboard control buttons, visual and auditory status indicators, collision detection, and extensive educational resources further support its use in instructional and research settings. For additional details on the communication protocols and user interface, readers are referred [35]to the official Niryo documentation [32], [33].

Kinematics and Numerical Dynamics: Forward kinematics were done to determine the position and orientation of the end-effector relative to the base frame as a function of the joint variables, enabling accurate mapping and synchronization between the physical robot and its virtual representation. The Denavit–Hartenberg (DH) convention was applied to define the robot's kinematic structure and compute the homogeneous transformation matrices associated with each

joint. The resulting forward kinematics are represented by the overall homogeneous transformation matrix from the base frame to the end-effector frame, expressed as:

$$H_6^0 = A_1.A_2.A_3.A_4.A_5.A_6 =$$
$$\begin{bmatrix} c_{\theta_1} & 0 & s_{\theta_1} & 0 \\ s_{\theta_1} & 0 & -c_{\theta_1} & 0 \\ 0 & 1 & 0 & 171.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -s_{\theta_2} & -c_{\theta_2} & 0 & -221s_{\theta_2} \\ c_{\theta_2} & -s_{\theta_2} & 0 & 221c_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c_{\theta_3} & 0 & s_{\theta_3} & 0 \\ s_{\theta_3} & 0 & -c_{\theta_3} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$
$$\begin{bmatrix} c_{\theta_4} & 0 & -s_{\theta_4} & 0 \\ s_{\theta_4} & 0 & c_{\theta_4} & 0 \\ 0 & -1 & 0 & 235 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c_{\theta_5} & 0 & s_{\theta_5} & 0 \\ s_{\theta_5} & 0 & -c_{\theta_5} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c_{\theta_6} & -s_{\theta_6} & 0 & 0 \\ s_{\theta_6} & c_{\theta_6} & 0 & 0 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $\mathbf{A_i}$ denotes the homogeneous transformation matrix corresponding to the $\boldsymbol{i}$-th joint. The explicit form of $\boldsymbol{H_6^0}$ defines the end-effector position and orientation in three-dimensional space and serves as the basis for virtual–physical alignment in the teleoperation platform. The kinematic model was validated through numerical simulation, allowing interactive visualization of joint motion and end-effector pose.

Inverse kinematics were also computed numerically using Mathematica to determine joint configurations corresponding to desired end-effector poses. In addition to kinematic modeling, the robot's nonlinear dynamics were computed numerically to support controller development and system-level analysis. The dynamic model was evaluated using MATLAB's Robotics System Toolbox to obtain the inertia, Coriolis, and gravity terms governing joint motion. Then, the dynamic analysis allowed the generation and import of the robot's URDF file for implementing the MR-enabled teleoperation platform. The detailed derivation of the forward and inverse kinematics, DH parameterization, and dynamic equations of motion is outside the scope of this paper and is presented in the author's Ph.D. dissertation. [35]

Overview of the Software Stack: This section presents a brief overview of the selected software stack. For detailed information and resources, readers are referred to [35]. The development and deployment of the MR-enabled teleoperation platform required the selection and integration of an appropriate software stack. Multiple software tools, packages, and application programming interfaces (APIs) were explored to enable reliable communication between the Niryo Ned2 robotic arm and the MR environment. The MR environment was developed using Unity (version 2021.3.16), which served as the main development platform for real-time visualization, interaction, and control. Several Unity-compatible packages were then integrated to support system functionality. The required integrated packages were as follows:

- **Photon Unity Networking (PUN)** was employed to enable multi-user interaction by managing networking, synchronization, and shared experiences within the MR environment.
- **URDF Importer Package** was used to import the kinematic structure of the Niryo Ned2 robotic arm into Unity by extracting the URDF file and generating a corresponding virtual representation of the robot's links, joints, and kinematics.

- **ROS for Unity**, a rendering package compatible with Unity's Universal Render Pipeline, was explored to support communication with ROS-based systems.
- **Mixed Reality Toolkit (MRTK 2),** developed by Microsoft, was selected to support MR application development.

Besides Unity-based tools, additional software was incorporated to enable communication between the MR environment and the physical robotic system. Niryo's PyNiryo, a Python-based API, was used to support direct robot control through ROS. **Niryo Studio** (v1.7.1), the official software provided by Niryo, was utilized for robot calibration, testing, and preliminary motion validation. The system was deployed on **Ubuntu**, an open-source Linux-based operating system, which served as the middleware platform for running ROS and managing communication between the MR virtual replica and the physical Niryo Ned2 robot. Ubuntu provided the required environment for data exchange, enabling bidirectional communication between Unity, ROS, and PyNiryo.

Communication Protocol Selection: Figure 2 illustrates the communication between the MR replica in Unity and the physical system (Niryo Ned2). The communication is summarized in two phases: *Phase One: The communication between the MR interface (developed in Unity) and the ROS system (running on Ubuntu) & Phase Two: The communication between the ROS system and the physical robot (Niryo Ned2).*

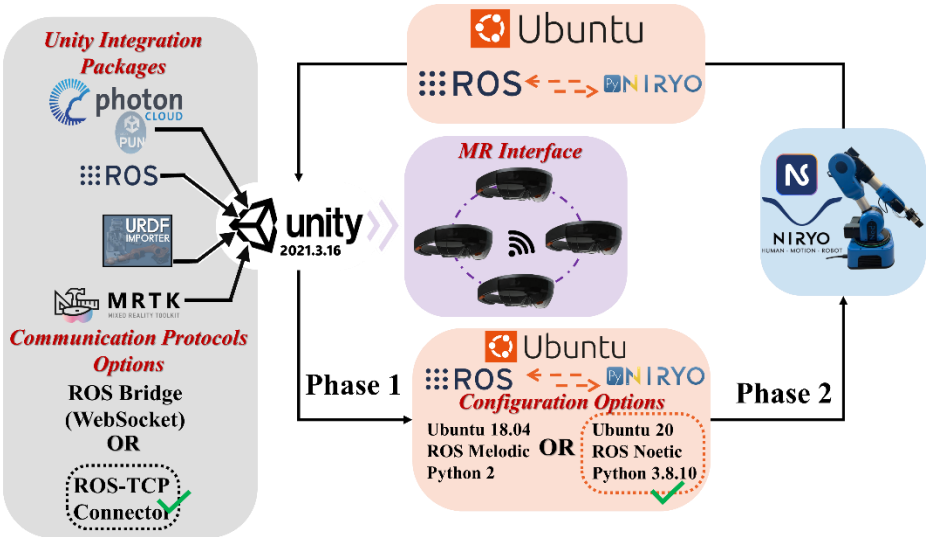

Figure 2. Diagram illustrating the software stack and communication protocol for the MR-enabled teleoperation platform.

In phase one, the real-time data exchange between Unity and the ROS system was tested through the following two communication protocols (ROS Bridge and ROS-TCP Connector). ROS Bridge, developed by the Siemens ROS-Sharp library, uses WebSocket-based communication to transmit data between Unity and ROS. ROS Bridge is used in applications built on .NET frameworks, including Unity3. However, ROS-TCP Connector is developed and maintained by Unity Technologies to support TCP/IP communication with ROS [36]. The ROS-TCP Connector

allows for the integration with Unity and is designed for performance-targeted robotics applications. Table 3 provides a comparison between the two communication protocols explored for enabling data exchange between Unity and the ROS system.

Table 3. Comparison of communication protocols for Unity-ROS integration

Communication Protocol	**ROS Bridge (WebSocket)**	**ROS-TCP Connector**
Company Developer	ROS-Sharp Library by Siemens	Unity Technologies
Latency (based on testing and supported by documentation) [37]	Higher Latency due to Websocket: **2s** per image	**0.17 s** per image
Real-Time Communication	Does not work for heavy data streams.	Works for heavy data streams, i.e., clouds, images, points, etc.
ROS Messages	Requires **manual setup**.	**Auto-generates** messages for unity.

The comparison between ROS Bridge and ROS-TCP Connector was based on three factors: latency, real-time communication capabilities, and the type of message generation in Unity. Based on testing and supported documentation, **ROS-TCP Connector** was selected due to its lower latency (0.17 per image), better performance for real-time data streams, and integration with Unity through auto-generated message types.

Phase two aimed at establishing communication between ROS, running on Ubuntu, and the physical Niryo Ned2 robot. Phase two of the communication enabled command execution, retrieving sensor data, and maintaining real-time synchronization between the MR environment and the physical system. The communication between ROS and the physical robot was established through PyNiryo 1.2.0, also running on Ubuntu. PyNiryo allowed communication with Niryo Ned2 by converting ROS services and topics into Python functions. PyNiryo v1.2.0 was selected, given its compatibility with Python 3.8.10 and ROS-TCP integration within Unity. Establishing this communication required selecting an appropriate ROS environment that supports PyNiryo v1.2.0, provides stability during real-time operations, and maintains compatibility with Unity's selected protocol (ROS-TCP Connector). Thus, two ROS configurations were tested (see Table 4).

Table 4. Two ROS environments

Configuration Option	**Operating system**	**ROS Version** [38]	**Python**
1	Ubuntu 18.04: Either installed as a desktop image using a bootable USB **OR** from Microsoft store running on WSL-2	ROS Melodic	Python 2
2	Ubuntu 20 running on WSL-2	ROS Noetic	Python 3.8.10

Ubuntu 18.04 with ROS Melodic, supported officially by Niryo, had compatibility problems since it was built upon Python 2, which is not supported anymore for Niryo Ned2. However, Ubuntu 20.04 with ROS Noetic supported the development and compatibility with ROS-TCP features, including custom message handling and real-time performance enhancement. Ubuntu 20 was installed via Windows Subsystem for Linux 2 (WSL-2), allowing for the running of Linux-based ROS infrastructure in a development environment hosted on Windows. The selected setup eliminated the need for dual booting or external installations, reducing system complexity. Thus, the final configuration selected for Phase two was:

- Operating System: Ubuntu 20.04 (via WSL-2)
- ROS Version: ROS Noetic
- Python Version: 3.8.10
- API: PyNiryo to control and observe the Niryo Ned2 robot

3.2. Platform Design and Implementation

After selecting and preparing the required software/ hardware components, as described in Section 3.1 (Teleoperation Architecture), the MR-enabled teleoperation platform was developed by integrating the selected components within the teleoperation architecture. The development process was done in three Phases: (1) development of the MR environment, (2) preparation for ROS-TCP communication, and (3) the integration of ROS communication with the physical Niryo Ned2 robotic arm (see Figure 3). The following subsection provides an overview of the development process; additional details can be found in [35].

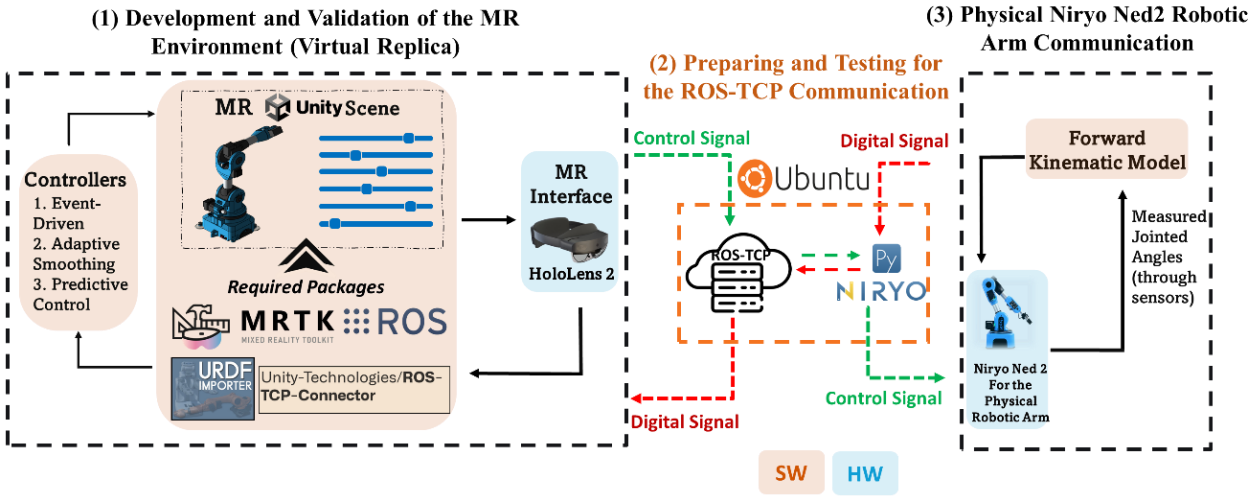

Figure 3. Architecture of the XR-enabled teleoperation platform integrating Unity, ROS-TCP communication, and the physical Niryo Ned2 robotic arm.

Phase 1: Development and Validation of the MR Environment

The MR development involved setting up the MR scene, implementing the virtual replica of the Niryo Ned2 robotic arm into Unity, and integrating and mapping UI controls for joint actuation.

MR Scene: The MR scene was set up on **Unity 2021.3** with the **MRTK 2** through the **MR Feature Tool Kit** configuration process [39], in a similar way to the MR development illustrated in [1]. The **MR Feature Tool Kit** enabled the integration of standard MRTK packages, including **MRTK Foundations**, **MRTK Extensions**, **MRTK Test Utilities**, and **MRTK Tools**. Additionally, platform-specific packages like the **MR OpenXR Plugin** and **MR Moving Platform SDK**, as well as **Azure MR Services** packages such as **Microsoft Azure Object Anchor** and **Azure Spatial Anchor for Windows**, were added. The MR playspace was then configured by switching Unity to the UWP, enabling the OpenXR plugin.

The publishing settings were then manually adjusted to enable **InternetClient**, **InternetClientServer**, and **PrivateNetworkClientServer** capabilities. The **InternetClient** was required to access the internet (outbound traffic), allowing Unity to communicate with ROS via network. The **InternetClientServer** allows inbound and outbound internet traffic, acting as both client and server in network communications. The **PrivateNetworkClientServer** allows communication on a private network, which was required because HoloLens 2 and the PC running ROS were on the same Wi-Fi. Thus, the integration of these capabilities ensured that the deployed application would establish bi-directional TCP communication with ROS over the private network, necessary for ROS-TCP Connector functionality during teleoperation tasks. Unity's integration with the HoloLens 2 was then tested and validated through a build deployment test of a cube.

Virtual Replica: The **URDF Importer** package was then installed from Unity's Package Manager, following the instructions in the Unity-Robotics-Hub [40]. The robot's kinematic and dynamic properties, outlined in Section 3.1, were implemented by referencing the Niryo Ned2 URDF files downloaded from the ned_ros GitHub repository [41]. Since the robot files were originally in Xacro format, they were manually converted to URDF format using ROS Noetic running in the Windows Subsystem for Linux (WSL) Ubuntu 20.04 environment, as Unity does not support Xacro files. For converting the files from Xacro to URDF, ROS Noetic [41] compatible with Ubuntu 20.04, was installed on WSL-2. The MR scene setup was based on the communication protocol selection resulting from the analysis illustrated in Section 3.1. Then, a ROS workspace (catkin_ws) was created to facilitate the conversion using the `rosrun xacro xacro` command for each Xacro file variant, including the main robot model and gripper configurations. The files were generated and validated through `check_urdf`. The virtual replica of the Niryo Ned2 robotic arm was imported into Unity using the **URDF Importer**, incorporating the coupled linkages, six articulated joints, and physics attributes such as mass, inertia, and collision geometries, thus establishing the robot's physics-based model. The imported physics-based model was validated for stability by setting the Ring Link "led_ring_link" and the Base Link "base_link" as immovable within Unity's inspector.

Virtual UI Controls Mapping: Six virtual sliders were added into the MR environment utilizing the PinchSlider UI elements from the MRTK2 to interact with the virtual physics-based model of the robot. Each of the six PinchSlider UI elements were linked to control the corresponding articulation body of the robotic joints, as illustrated in Figure 4.

Each slider output a normalized value [0 to 1], which was mapped to the allowable range of motion [lower and upper limits] for the respective joint's `xDrive target` parameter. A custom C# script, "MRSliderMapping", was written to map slider movements to joint target positions, adjusting Unity's articulation drive parameters (stiffness, damping, force limits) to ensure stable, continuous, and physically realistic interaction. The script continuously read the normalized output value of each slider, mapped it to the physical joint's allowable motion range based on its articulation drive setting.

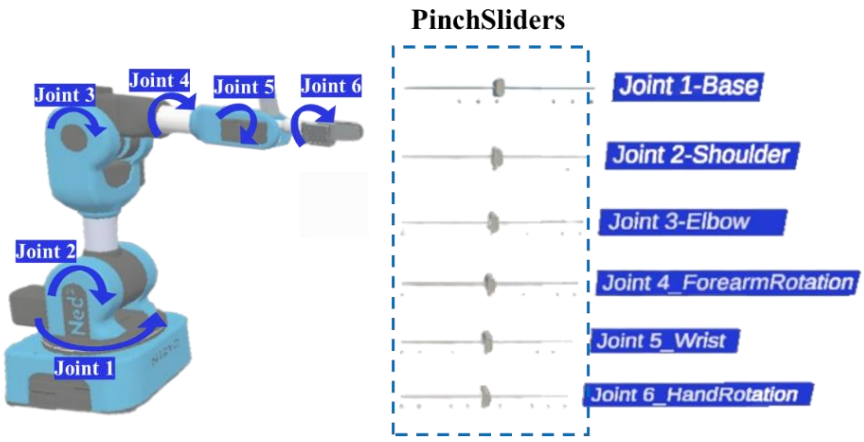

Figure 4. Virtual physics-based model of the Niryo Ned2 robotic arm, integrated with six PinchSlider UI controls for joint manipulation.

Phase 2: Preparing and Testing the ROS-TCP Communication

The real-time interaction between the MR environment and the physical Niryo Ned2 robotic arm was enabled through a bidirectional communication framework that was established using ROS–TCP integration (see Figure 5). This communication allowed the MR interface to function as both a command source and a feedback receiver within the ROS network.

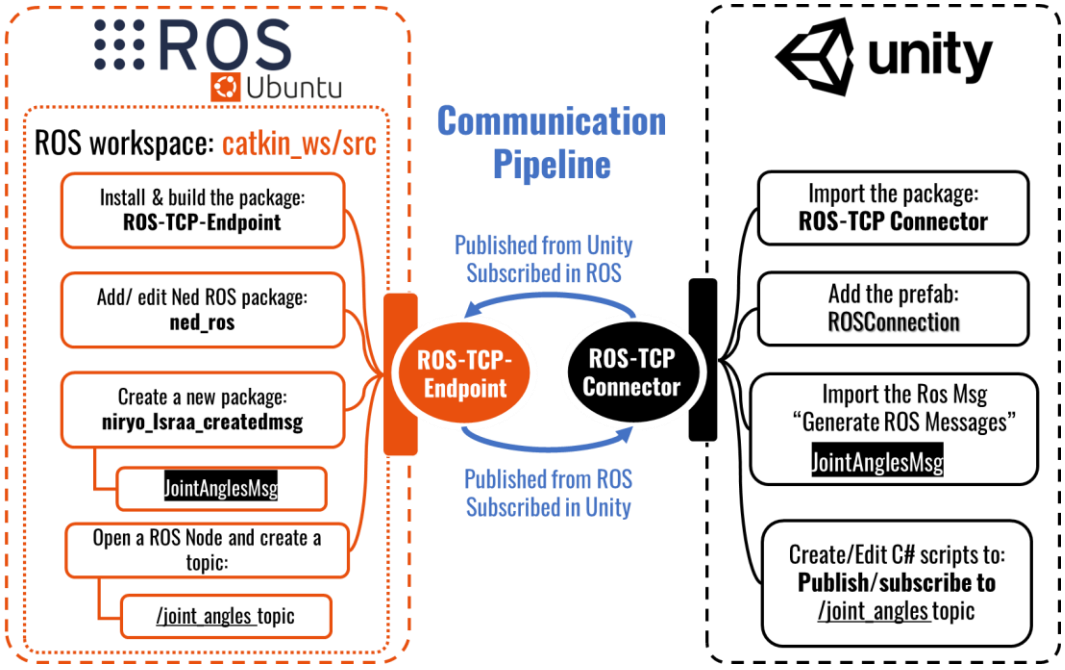

Figure 5. Unity–ROS bi-directional communication setup.

The ROS–TCP communication was implemented by deploying the ROS–TCP Endpoint within a ROS workspace running on Ubuntu 20.04. The corresponding ROS–TCP Connector was

integrated into the Unity environment to facilitate message exchange between Unity and ROS. Initial testing focused on validating basic connectivity, ensuring correct network configuration, and verifying compatibility between Unity, Python, and ROS. Live testing was performed using **Holographic Remoting on the HoloLens 2**, enabling real-time streaming of the MR application from the Unity Editor while maintaining active ROS communication.

For the closed-loop control, two communication pathways were designed and validated: (1) publishing joint angle commands from Unity to ROS, and (2) subscribing to joint angle updates from ROS back into Unity. The two pathways enable synchronized control and visualization between the virtual and physical robotic systems.

In the first pathway, user interaction within the MR environment, specifically through virtual sliders controlling the robot joints, was mapped to joint angle commands and transmitted to ROS in real time. A custom ROS message type was defined to represent the six joint angles of the robotic arm. This message was generated within the ROS workspace and made accessible to Unity through the ROS–TCP Connector. On the Unity side, a custom control script linked the MR user interface elements to the virtual robot joints and published the corresponding joint angle data to a designated ROS topic. Real-time monitoring confirmed that joint commands generated in the MR environment were successfully received by ROS, validating Unity's role as an effective teleoperation interface.

The second communication pathway enabled Unity to receive joint angle data from ROS, allowing the MR environment to update the posture of the virtual robot based on externally generated commands. A subscriber script was implemented in Unity to listen to the same joint angle topic and apply incoming data to the virtual robot's articulation drives. This functionality supports bidirectional communication and lays the foundation for closed-loop control and feedback-driven synchronization between the physical robot and its digital twin. Validation tests confirmed that Unity correctly responded to joint commands published from ROS, updating the virtual robot's configuration in real time.

The complete communication architecture was verified using ROS graph visualization tools, which illustrated the interaction between Unity, ROS nodes, and the defined communication topics. These results confirmed reliable data flow in both directions and demonstrated that the MR environment could operate as both a controller and a receiver within the ROS ecosystem.

The implemented Unity–ROS communication framework enabled bidirectional data exchange between the MR environment and the physical robotic system. This integration allows real-time command transmission, synchronized visualization, and extensibility toward the advanced control strategies (to be discussed in the coming section) and closed-loop operation.

Phase 3: Network Communication: Synchronization between the Physical Niryo Ned2 Robotic Arm and its Virtual Physics-Based Model

Following the development of the MR environment and the establishment of communication between Unity and ROS, the third stage focused on enabling synchronized communication between the MR interface and the physical Niryo Ned2 robotic arm.

Robot Configuration and Baseline Communication Validation: The Niryo Ned2 robotic arm was initially configured using the official Niryo Studio software to establish baseline connectivity and verify correct system operation. The robot firmware and control software were configured following official documentation, enabling control through Python and Blockly interfaces. Preliminary pick-and-place tasks were executed using the built-in 2D interface to validate robot functionality and network stability prior to integration with ROS and the MR environment.

To assess communication performance, network latency between the control computer and the robotic arm was evaluated using round-trip time (RTT) measurements. The observed latency remained stable and within acceptable bounds for real-time teleoperation, with an average RTT of approximately 5.7 ms. The RTT values under 10 ms are regarded as optimal for achieving responsive teleoperation for local connections (same room) [42], [43]. So, the RTT latency for the existing 2D interface was classified as “Very Good”, meeting the performance requirements for real-time robotic control.

ROS–Python Hardware Bridge and MR Integration: To enable direct control of the physical robot from the MR environment, a hardware bridge was implemented using a custom ROS–Python node. This node subscribed to joint angle commands published from Unity and translated them into executable motion commands for the physical robot using the PyNiryo API. The ROS launch configuration initialized the required drivers, controllers, and robot description, ensuring that the robotic system was fully operational and ready to receive actuation commands.

On the Unity side, the control scripts were modified to ensure compatibility with ROS communication standards. Joint angle data generated through MR-based user interaction were converted from degrees to radians and published using standardized ROS message types. A default joint configuration was applied during system initialization to synchronize the virtual robot’s posture with the physical robot’s starting pose, ensuring consistent alignment between the two systems.

Real-Time Synchronization and Validation: The integrated system achieved real-time synchronization between the physical Niryo Ned2 robotic arm and its virtual physics-based replica within the MR environment, as shown in the proof-of-concept in Figure 6. User interactions within the MR interface, such as adjusting virtual sliders, resulted in immediate actuation of the physical robot, while maintaining consistent visual alignment between the virtual and physical systems. This synchronization validated the effectiveness of the communication architecture and demonstrated the feasibility of using the MR environment as a teleoperation interface for physical robotic systems.

Thus, reliable network communication and real-time synchronization were successfully established between the MR environment and the physical robotic arm. The implemented framework supported low-latency command transmission, consistent virtual–physical alignment, and extensibility toward advanced control strategies.

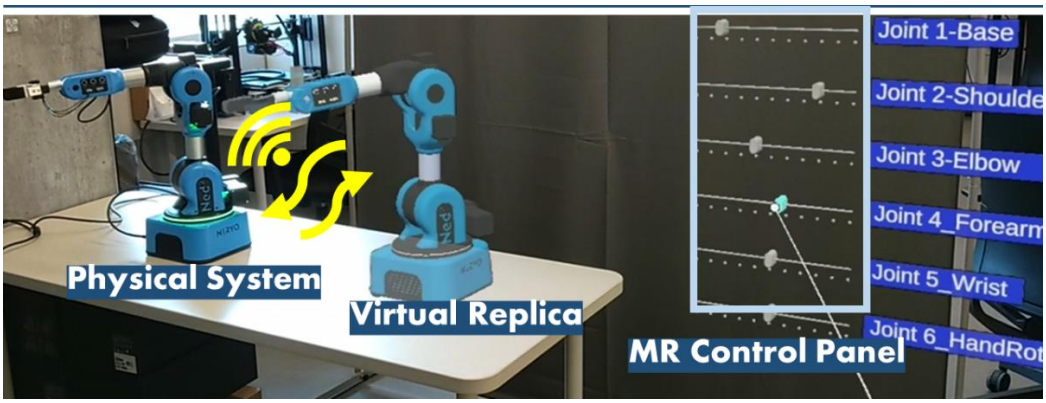

Figure 6. The synchronization between the physical Niryo Ned2 robotic arm and its virtual replica in the proof-of-concept XR-enabled teleoperation platform.

3.3. System Debugging, Troubleshooting, and Final Configuration

Debugging and final configuration were conducted to ensure the reliability and stable operation of the MR-enabled teleoperation platform before the integration of advanced control protocols. Given the multi-layered software architecture and hardware components involved, this phase required systematic testing across Unity, ROS running on Ubuntu, and the Niryo Ned2 robotic arm. System-level testing was performed through sequential execution of each software module to verify connectivity, message exchange, and robot actuation.

Four system-level issues were identified during initial testing:

- Incomplete execution of robot driver launch files due to unresolved dependencies and workspace configuration issues.
- Instability in robot visualization caused by missing semantic and kinematic parameters.
- ROS node failures related to hardware-specific dependencies incompatible with the host environment.
- Unstable robot motion resulting from high-frequency command updates generated by the MR slider interface.

These issues were resolved through configuration and software-level adjustments. Workspace organization and launch file configurations were corrected to restore proper robot driver execution and model stability. Hardware-specific dependencies were disabled to ensure compatibility with the selected operating environment. To address command instability, input handling in the MR interface was optimized by introducing buffering and stabilization logic, ensuring that only steady joint commands were transmitted to the physical robot.

Following these refinements, system-level validation confirmed reliable communication between Unity, ROS, and the Niryo Ned2 robotic arm. Joint command transmission was verified through ROS topic monitoring, and real-time robot actuation in response to MR-based user input was successfully demonstrated. While this stage focused on unidirectional teleoperation, the finalized

architecture was verified to support future extensions toward bidirectional feedback and closed-loop control.

4. Advanced Control Design and Integration

Following the development and validation of the MR-enabled teleoperation platform, advanced control strategies were designed and integrated to address control latency and system responsiveness, constituting a key technical contribution of this work. The primary objective of incorporating these controllers was to enhance system stability and enable synchronized closed-loop teleoperation between the physical robot and its virtual representation.

Three control strategies were developed and integrated into the platform, starting from an event-driven open-loop batch controller to a trajectory-based controller and, finally, to a closed-loop predictive control strategy. This section focuses on the design and integration of the closed-loop predictive controller, detailing both the control logic and the associated control law.

4.1. Overview of the Closed-Loop Predictive Control

The Closed-Loop Predictive Control is utilized in robotics and automation systems to enhance trajectory tracking performance, improve stability, and minimize response delays through the use of real-time feedback [44]. This controller is based on the Model Predictive Control (MPC) frameworks, which continuously adjust the control input based on the last system states, anticipating future system behavior to optimize performance [45]. The controller allows for robustness to disturbances and modeling uncertainties, making it useful for systems requiring precision and synchronization, such as teleoperated robotic platforms, by incorporating feedback into the control synthesis.

The predictive control strategies are categorized into two main types: classical predictive control and advanced multi-step MPC [46]. Classical predictive control, such as Generalized Predictive Control (GPC) and Dynamic Matrix Control (DMC), rely on a single-step or limited-horizon prediction mechanism that adjusts control actions based on the immediate future state, allowing for lower computation and faster execution [47], [48]. The advanced MPC schemes involve optimizing over a multi-step prediction horizon using a cost function that balances tracking performance, control effort, and constraint satisfaction, at the expense of increased computational complexity.

For this work, the classical form of closed-loop predictive control was developed by building on the GPC framework to address the challenges of latency, synchronization, and real-time feedback integration within the MR-enabled teleoperation platform. The proposed controller utilizes a single-step predictive adjustment based on the latest feedback from the physical robot to correct deviations from the desired trajectory instead of employing a full multi-step optimization horizon. The adopted approach provides a balance between computational tractability and dynamic responsiveness, allowing for both real-time performance and synchronization between physical and virtual systems. The proposed controller predicts and smooths joint commands in real time using feedback from the Niryo Ned2 physical robot, while simultaneously updating the Unity digital twin (virtual replica).

4.2. Control Logic

The control logic of the closed-loop predictive control is illustrated in Figure 7. As shown in the diagram, the control logic architecture consists of the three interconnected environments: *Unity XR Environment*, *ROS Environment*, and the *Niryo Physical Environment*.

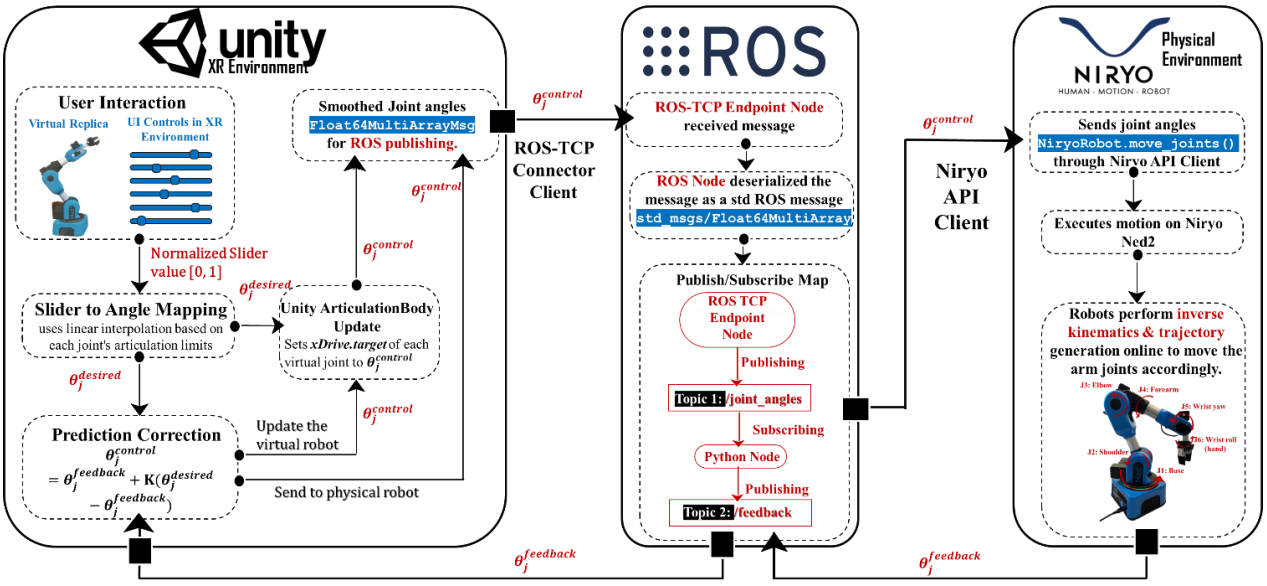

Figure 7. Control logic architecture of the closed-loop predictive control.

Unity XR Environment: The users in the XR environment interact with the virtual replica of the Niryo Ned2 robotic arm through UI sliders representing each of the six joints, virtual joysticks, or PinchSliders, each mapped to a revolute joint of the robot's virtual replica. The sliders generate normalized values [0,1], which are then linearly mapped to each joint's actual articulation limits to produce the initial desired joint angles $\boldsymbol{\theta_j^{desired}}$. The generated desired angles are fed as an input to the predictive controller block, which allows for real-time feedback received from the physical robot via ROS.

The predictive correction adjusts each joint angle based on the discrepancy between the desired $\theta_j^{desired}$ and current joint states, generating a control command $\theta_j^{control}$ that predicts the future state of the system. The $\theta_j^{control}$ control command is fed to the Unity ArticulationBody block to update the angles of the virtual replica, thus maintaining an up-to-date digital twin. Also, the control command is fed to the smoothed joint angle block to be stored as a Float64MultiArrayMsg and published to ROS via the /joint_angles topic for physical execution.

ROS Environment: The $\boldsymbol{\theta_j^{control}}$ control command is transmitted from the Unity XR environment to ROS through the ROS-TCP Connector Client. Upon receiving the control message, the ROS node deserializes it and passes it to a Python-based subscriber node, which invokes the move_joints() function via the Niryo API.

Niryo Ned 2 Robot in the Physical Environment: The physical Niryo Ned2 robot then performs the commanded motion by internally resolving inverse kinematics and trajectory planning routines. The same Python node publishes the resulting joint positions back to Unity through the /feedback topic. The closed-loop allows for feedback, enabling Unity to access real-time joint angle values $\boldsymbol{\theta_j^{feedback}}$ which are then used to refine future predictions in the next control loop.

4.3.Control Law

The control synthesis for the Closed-loop Predictive Control is based on a single-step predictive adjustment model that aims to minimize the tracking error between the desired and actual joint positions in real time. The control synthesis is based on the Model Predictive Control (MPC) principles. However, the prediction horizon is simplified given the single-step projection due to system constraints on computational latency and communication in real-time robotic teleoperation.

The control law is defined through the predictive update of the joint angle at time step *t*, denoted as $\theta_j^{predict}[t] = \theta^{predict}[t][j]$, and is expressed as in the following equation:

$$\theta_j^{predict}[t] = \theta_j^{current} + K(\theta_j^{desired} - \theta_j^{current}),$$

where *K* denoted the tunable proportional gain that modulates the rate of convergence of the predictive correction. $\theta_j^{current} = \theta_j^{feedback}$ is the latest feedback value of the joint angle.

The controller leverages the latest feedback value of the joint angle, $\theta_j^{current} = \theta_j^{feedback}$ received from the physical system to estimate the control command required to reduce the tracking error.

The prediction horizon would then iterate over *H* steps to optimize the desired trajectory. The predictive model, although not fully implemented in this controller, would be represented as follows:

$$\theta_j^{predict}[t+1] = \theta_j^{predict}[t] + K(\theta_j^{desired} - \theta_j^{predict}[t]),$$

with the following cost function that minimizes the cumulative tracking error over all joints and time steps:

$$Total\ Error = \sum_{t=1}^{H}\sum_{j=1}^{N}(\theta_j^{desired} - \theta_j^{predict}[t])^2,$$

where *N=6* represents the number of joints and *H* denotes the prediction horizon length.

Then, in our current control implementation, the control input signal is computed at $t=1$ and directly applied as the command sent to both Unity XR environment and Niryo Ned2 physical environment:

$$\theta_j^{control} = \theta_j^{predict} [1]$$

The implemented control law allows for low-latency execution (below 50 ms) while ensuring continuous real-time control. The system dynamically corrects tracking errors by continuously adjusting $\theta_j^{control}$ based on the latest $\theta_j^{feedback}$, ensuring that both virtual and physical arms stay synchronized during operation. This predictive closed-loop formulation balances responsiveness and stability while remaining computationally tractable within a mixed-reality teleoperation system.

5. Results

The Closed-Loop Predictive Control strategy was integrated into the MR-enabled teleoperation, enabling real-time feedback and reducing the latency. The performance of the controller was evaluated to assess the system's responsiveness and synchronization between the physical Niryo Ned2 robotic arm and its MR virtual replica. Two main performance indicators were used: (1) synchronization accuracy along with end-to-end latency of the command transmission pipeline and (2) accuracy of the end-effector trajectory. The results are reported in Figure 8, Figure 9, and Figure 10.

Synchronization and Latency: Figure 8 shows three subplots: (1) the virtual joint angles, (2) the actual joint angles executed by the physical Niryo Ned2 robotic arm, and (3) the corresponding end-to-end latency measurements. The latency subplot (3) shows that the system maintained an average latency below 0.01s (10 ms) throughout the session, indicating a stable control pipeline. The joint subplots (1) and (3) show that the physical joint angles track the MR input angles, with deviations below 0.1 radians. The results demonstrate the controller's ability to anticipate deviations and adjust the actuation trajectory accordingly, improving overall alignment between virtual commands and physical execution.

End-Effector Trajectory: Figure 9 illustrates the end-effector position tracking along the X, Y, and Z axes. The figure shows that the end-effector of the MR replica tracks the physical end-effector, as the three positions (X, Y, and Z) exhibit coupled movement profiles. Using the Euclidean distance equation, the red dashed line (physical) tracks the blue line (MR) with minimal lag (below 10 mm), confirming effective closed-loop control. Thus, the predictive adjustment mechanism of Controller Three constrained the deviation to within 5–10 mm during dynamic transitions. Recalling that the Niryo Ned2 robot's specified positional uncertainty is ±0.5 mm, all reported deviations were interpreted within this uncertainty margin. Thus, the synchronization was tight during longer movement intervals, where the predictive correction allows for steady convergence.

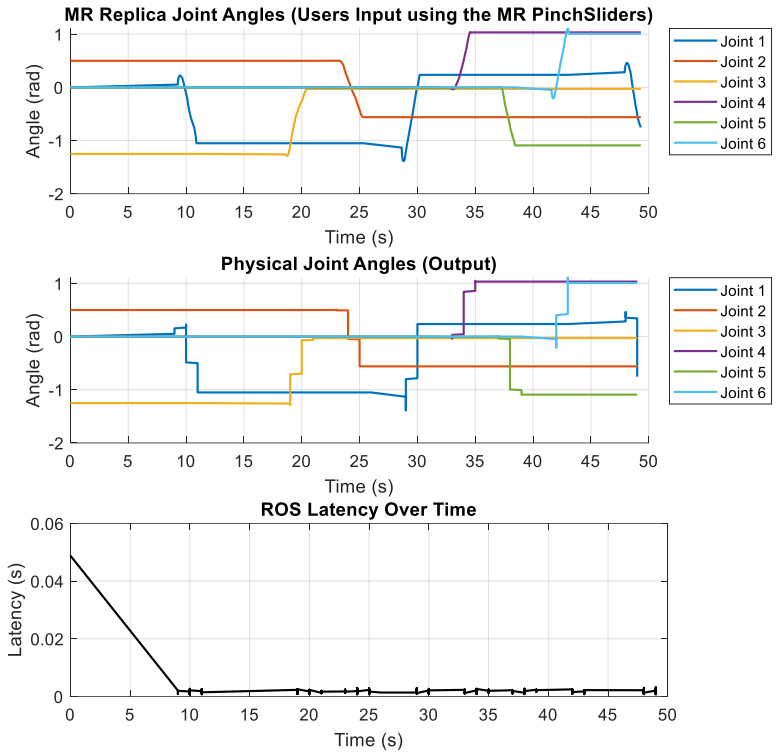

Figure 8. The MR input joint angles versus the actual physical output joint angles with the end-to-end latency resulting from Controller Three.

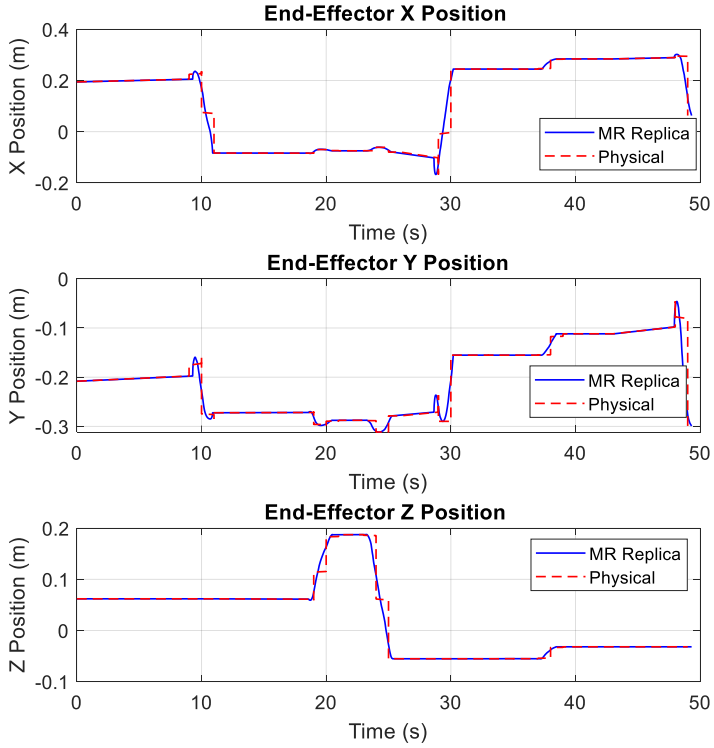

Figure 9. The end-effector cartesian positions over time in X, Y, and Z directions for the MR replica versus the physical Niryo Ned2 robot resulting from Controller Three.

Figure 10 presents the 3D end-effector trajectory using time as the vertical axis. The MR trajectory (dashed blue) and the physical trajectory (solid red) appear aligned, with slight variations (approximately 5–10 mm) during sharp transitions or directional changes.

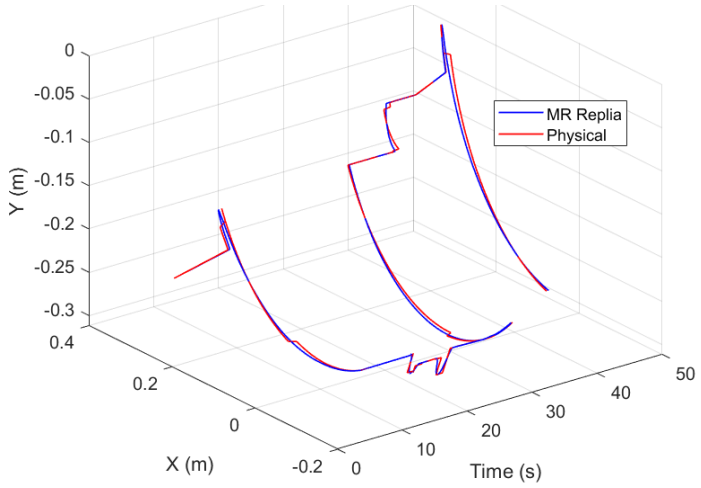

Figure 10. End-effector 3d trajectory with time as z-axis: MR vs physical resulting from Controller Three.

6. Discussion & Mapping to Manufacturing Education

The closed-loop predictive controller enabled bidirectional communication and improved performance relative to the other two controllers (out of scope for this paper). The integrated control pipeline achieved an average end-to-end latency of < 10 ms while maintaining synchronization between the physical system and its virtual replica, with joint-angle deviations below 0.1 rad and end-effector positional differences of 5–10 mm during fast transitions. These results indicate that the proposed architecture can support responsive MR teleoperation under local networking conditions. Prior work in the literature reports end-to-end delays above 50 ms 50 ms, and in some applications exceeding 100 ms, which can degrade teleoperation stability and interaction quality [42], [43]. Thus, reducing the delay below 10 ms enables a more “continuous” interaction loop, showing a significant improvement. This directly addresses RQ1 by demonstrating that an MR-enabled teleoperation platform, combined with feedback-based predictive correction, can minimize latency while preserving synchronization.

From a control perspective, the observed improvement aligns with the role of feedback in maintaining tracking performance under time delays and sampling effects. When command updates occur at high frequency, delays can reduce stability margins and introduce high overshoots. By updating the command using the latest measured joint state, the proposed controller reduces cycle-to-cycle tracking error and limits drift between the physical robot and its MR digital twin. Also, the reported 5–10 mm end-effector difference reflects system-level synchronization and mapping error (communication, control, and digital-twin update), not the robot’s intrinsic mechanical repeatability.

Beyond technical performance, the results show the potential for extending the platform into manufacturing education. Many smart manufacturing laboratory implementations rely on simulation-only or VR-only environments, which often lack real-time coupling with physical systems. This is due to the fact that the physical integration causes delays. Achieving <10 ms average latency with consistent synchronization supports the use of MR as an interactive training interface in which learners can observe and reason about cause–effect relationships between commands, robot motion, and system dynamics in real time. Therefore, the proposed platform represents a first step toward smart manufacturing laboratories that integrate MR, digital twins, and real hardware in a single instructional environment, addressing RQ2 from a technical viability standpoint.

7. Conclusion and Future Work

This paper presented the development and validation of an MR-enabled teleoperation platform that integrates a physical 6-DOF robotic arm with its digital shadow (replica) to enable real-time interaction and synchronized operation. The platform combines MR visualization, Unity-based interaction, ROS communication, and hardware execution to support teleoperation within a shared virtual–physical environment. A key technical contribution is the integration of a closed-loop predictive controller that leverages real-time feedback to mitigate delay and reduce divergence between the virtual and physical systems. The resulting system achieved <10 ms average end-to-end latency, <0.1 rad joint deviations, and 5–10 mm end-effector differences during dynamic motion, supporting responsive MR-based digital-twin teleoperation and directly addressing RQ1.

This work also establishes a technical foundation for smart manufacturing laboratories by enabling tightly coupled interaction with real hardware through an MR digital-twin interface. Such coupling can support experiential instruction in teleoperation, real-time control, and cyber–physical systems by allowing learners to observe immediate cause–effect relationships between control inputs and robot behavior. Future work will investigate multi-step predictive control for improved robustness and trajectory optimization, incorporate additional feedback modalities (e.g., force/haptic cues), and expand the architecture to support multi-user collaborative MR teleoperation. Finally, controlled classroom studies will be conducted to evaluate learning outcomes and inform deployment of the platform as a core component of next-generation smart manufacturing laboratories.

References

[1] I. Azzam, K. Bello, F. El Breidi, and F. Aqlan, “Analysis of user experience in extended reality: a comparative study of VR and MR for manufacturing training,” *Manuf Lett*, vol. 46, pp. 77–82, Dec. 2025, doi: 10.1016/J.MFGLET.2025.10.003.

[2] I. Azzam, K. Bello, F. El Breidi, and F. Aqlan, “Collaborative problem-solving in mixed reality manufacturing environments,” *Manuf Lett*, vol. 46, pp. 118–122, Dec. 2025, doi: 10.1016/J.MFGLET.2025.10.018.

[3] I. Azzam, K. Pate, F. Breidi, M. Choi, Y. Jiang, and C. Mousas, “Mixed Reality: A Tool for Investigating the Complex Design and Mechanisms of a Mechanically Actuated

Digital Pump," *Actuators* , vol. 12, no. 11, p. 419, Nov. 2023, doi: 10.3390/ACT12110419.

[4] I. Azzam, K. El Breidi, F. Breidi, and C. Mousas, "Virtual Reality in Fluid Power Education: Impact on Students' Perceived Learning Experience and Engagement," *Education Sciences* , vol. 14, no. 7, p. 764, Jul. 2024, doi: 10.3390/EDUCSCI14070764.

[5] I. Azzam, F. Breidi, and F. Aqlan, "Teaching Manufacturing Assembly Processes Using Immersive Mixed Reality," in *2024 ASEE Annual Conference & Exposition Proceedings*, Portland, Oregon: ASEE Conferences, Jun. 2024. doi: 10.18260/1-2--48063.

[6] M. J. Liberatore and W. P. Wagner, "Virtual, mixed, and augmented reality: a systematic review for immersive systems research," *Virtual Reality 2021 25:3*, vol. 25, no. 3, pp. 773–799, Jan. 2021, doi: 10.1007/S10055-020-00492-0.

[7] M. Speicher, B. D. Hall, and M. Nebeling, "What is mixed reality?," in *Conference on Human Factors in Computing Systems - Proceedings*, Association for Computing Machinery, May 2019. doi: 10.1145/3290605.3300767.

[8] S. Palihawadana, "Cognitive Overload in Mixed-Reality Interactions: A Qualitative Analysis," UMEA Universitet, 2023. Accessed: Nov. 29, 2024. [Online]. Available: https://umu.diva-portal.org/smash/get/diva2:1783578/FULLTEXT01.pdf

[9] I. Azzam, S. Das, G. Nanda, J. G. Bravo, and F. Breidi, "Investigating Safety Awareness in Assembly Operations via Mixed Reality Technology," *IISE Trans Occup Ergon Hum Factors*, pp. 1–17, Nov. 2024, doi: 10.1080/24725838.2024.2431112.

[10] Z. Ju, C. Yang, Z. Li, L. Cheng, and H. Ma, "Teleoperation of humanoid baxter robot using haptic feedback," in *Proceedings of 2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems, MFI 2014*, Institute of Electrical and Electronics Engineers Inc., Dec. 2014. doi: 10.1109/MFI.2014.6997721.

[11] M. Singh, E. Fuenmayor, E. P. Hinchy, Y. Qiao, N. Murray, and D. Devine, "Digital Twin: Origin to Future," *Applied System Innovation*, vol. 4, no. 2, p. 36, May 2021, doi: 10.3390/ASI4020036.

[12] F. Tao and Q. Qi, "Make more digital twins," *Nature* , vol. 573, no. 7775, pp. 490–491, Sep. 2019, doi: 10.1038/d41586-019-02849-1.

[13] Z. Gharaybeh, H. Chizeck, and A. Stewart, "Telerobotic control in virtual reality," in *OCEANS 2019 MTS/IEEE Seattle,* Institute of Electrical and Electronics Engineers Inc., Oct. 2019. doi: 10.23919/OCEANS40490.2019.8962616.

[14] R. Hetrick, N. Amerson, B. Kim, E. Rosen, E. J. D. Visser, and E. Phillips, "Comparing Virtual Reality Interfaces for the Teleoperation of Robots," in *2020 Systems and Information Engineering Design Symposium, SIEDS 2020*, Institute of Electrical and Electronics Engineers Inc., Apr. 2020. doi: 10.1109/SIEDS49339.2020.9106630.

[15] D. Whitney, E. Rosen, E. Phillips, G. Konidaris, and S. Tellex, "Comparing Robot Grasping Teleoperation Across Desktop and Virtual Reality with ROS Reality," in *In Robotics Research: The 18th International Symposium ISRR Cham: Springer International Publishing.*, Springer Science and Business Media B.V., 2020, pp. 335–350. doi: 10.1007/978-3-030-28619-4_28/TABLES/1.

[16] E. Duff, C. Caris, A. Bonchis, K. Taylor, C. Gunn, and M. Adcock, "The development of a telerobotic rock breaker," *Springer Tracts in Advanced Robotics*, vol. 62, pp. 411–420, 2010, Accessed: Sep. 05, 2023. [Online]. Available: https://doi.org/10.1007/978-3-642-13408-1_37

[17] D. Sun, A. Kiselev, Q. Liao, T. Stoyanov, and A. Loutfi, "A New Mixed-Reality-Based Teleoperation System for Telepresence and Maneuverability Enhancement," *IEEE Trans Hum Mach Syst*, vol. 50, no. 1, pp. 55–67, Feb. 2020, doi: 10.1109/THMS.2019.2960676.

[18] A. Rukangu, A. Tuttle, and K. Johnsen, "Virtual reality for remote controlled robotics in engineering education," in *Proceedings - 2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops, VRW 2021*, Institute of Electrical and Electronics Engineers Inc., Mar. 2021, pp. 751–752. doi: 10.1109/VRW52623.2021.00258.

[19] R. Cioffi *et al.*, "Smart Manufacturing Systems and Applied Industrial Technologies for a Sustainable Industry: A Systematic Literature Review," *Applied Sciences* , vol. 10, no. 8, Apr. 2020, doi: 10.3390/APP10082897.

[20] W. J. Yun, S. Hong, K. W. Lee, D. Kim, H. Kim, and S. H. Ahn, "SmartLab: Flexible and interoperable manufacturing laboratory system for remote education and research using a mobile manipulator," *J Comput Des Eng*, vol. 12, no. 4, pp. 221–235, Mar. 2025, doi: 10.1093/JCDE/QWAF034.

[21] R. Salehzadeh, G. Galvani, A. Zargarani, N. Jalili, and D. J. Fonseca, "Educating the Workforce of the 21st Century through Smart Manufacturing Systems in the Classrooms," *ASEE Annual Conference and Exposition, Conference Proceedings*, Jun. 2023, doi: 10.18260/1-2--43222.

[22] T. Ruppert, A. Darányi, T. Medvegy, D. Csereklei, and J. Abonyi, "Demonstration Laboratory of Industry 4.0 Retrofitting and Operator 4.0 Solutions: Education towards Industry 5.0," *Sensors* , vol. 23, no. 1, p. 283, Jan. 2022, doi: 10.3390/S23010283.

[23] L. Yang, Y. Wu, Z. Li, G. Zhang, and H. Tian, "Research on Practical Teaching System of Intelligent Manufacturing Engineering: A Case Study of Major Design," *Proceedings of 2024 International Conference on Artificial Intelligence and Future Education, AIFE 2024*, pp. 83–88, Jan. 2025, doi: 10.1145/3708394.3708410.

[24] "Purdue Polytechnic Smart Learning Factory to integrate advanced AutoStore warehouse system through Element Logic partnership | Purdue Polytechnic News," Newsroom. Accessed: Dec. 26, 2025. [Online]. Available:

https://polytechnic.purdue.edu/newsroom/purdue-polytechnic-smart-learning-factory-integrate-advanced-autostore-warehouse-system

[25] "Human-Centered Intelligent Systems for Operational Excellence," Aqlan Lab. Accessed: Dec. 26, 2025. [Online]. Available: https://www.aqlanlab.org/

[26] "Learning Factory | Grado Department of Industrial and Systems Engineering | Virginia Tech," ISE. Accessed: Dec. 26, 2025. [Online]. Available: https://www.ise.vt.edu/research/labs/Learning-Factory.html

[27] "Leveling up manufacturing with virtual reality and AI," News. Accessed: Dec. 26, 2025. [Online]. Available: https://news.engineering.asu.edu/2025/07/leveling-up-manufacturing-with-virtual-reality-and-ai/

[28] R. Begum and F. Aqlan, "MAKER: A Collaborative Virtual Learning Factory for Manufacturing Education," *ASEE*, Jun. 2025, doi: 10.18260/1-2--55554.

[29] "Manufacturing 4.0 Part 2: Transition Challenges and the Role of Technology - LabVantage," Labvantage. Accessed: Dec. 26, 2025. [Online]. Available: https://www.labvantage.com/blog/manufacturing-4-0-part-2-transition-challenges-and-the-role-of-technology/

[30] S. Liu, P. Li, J. Wang, and P. Liu, "Toward industry 5.0: Challenges and enablers of intelligent manufacturing technology implementation under the perspective of sustainability," *Heliyon*, vol. 10, no. 15, p. e35162, Aug. 2024, doi: 10.1016/J.HELIYON.2024.E35162.

[31] P. M. Goutham, R. Y. R, and T. S. Nanjundeswaraswamy, "A Review on Smart Manufacturing, Technologies and Challenges," *International Research Journal of Engineering and Technology (IRJET)*, 2022, Accessed: Dec. 26, 2025. [Online]. Available: www.irjet.net

[32] NiryoDocs, "The Ned2." [Online]. Available: https://docs.niryo.com/robots/ned2/article/

[33] Niryo, "Discover Ned2," *Niryo*. [Online]. Available: https://niryo.com/6-axis-robot/

[34] Niryo, "Join the new robotic era," 2025.

[35] I. Azzam, "Transforming Teleoperations: A Synchronized Extended Reality-Enabled Platform for Remote Control and Monitoring of Industrial Robotic Systems," *Purdue University Graduate School. Thesis*, Jun. 2025, doi: 10.25394/PGS.29218886.V1.

[36] Github, "ROS-TCP-Endpoint," 2021. [Online]. Available: https://github.com/Unity-Technologies/ROS-TCP-Endpoint. Accessed: June. 10, 2025.

[37] Discussion.unity, "Delay problem of subscribing image topic from ROS in Unity," 2024. [Online]. Available: https://discussions.unity.com/t/delay-problem-of-subscribing-image-topic-from-ros-in-unity/891334. Accessed: June. 10, 2025.

[38] Packages.ROS, "Open Source Lab," 2025. [Online]. Available: http://packages.ros.org/ros/ubuntu/. Accessed: June. 10, 2025.

[39] MicrosoftMRTeam, "Welcome to the Mixed Reality Feature Tool - Mixed Reality," Microsoft Learn. Accessed: Apr. 23, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/welcome-to-mr-feature-tool

[40] Github, "Installing the Unity Robotics packages," 2021. [Online]. Available: https://github.com/Unity-Technologies/Unity-Robotics-Hub/blob/main/tutorials/quick_setup.md. Accessed: June. 10, 2025.

[41] Github, "Niryo Robot Description," 2024. [Online] https://github.com/NiryoRobotics/ned_ros/tree/master/niryo_robot_description/urdf/ned2 Accessed: June. 10, 2025.

[42] X. Xu, Q. Liu, and E. Steinbach, "Toward QoE-driven dynamic control scheme switching for time-delayed teleoperation systems: A dedicated case Study," *HAVE 2017 - IEEE International Symposium on Haptic, Audio-Visual Environments and Games, Proceedings*, vol. 2017-December, pp. 1–6, Dec. 2017, doi: 10.1109/HAVE.2017.8240352.

[43] M. Condoluci, T. Mahmoodi, E. Steinbach, and M. Dohler, "Soft Resource Reservation for Low-Delayed Teleoperation over Mobile Networks," *IEEE Access*, vol. 5, pp. 10445–10455, 2017, doi: 10.1109/ACCESS.2017.2707319.

[44] J. B. Rawlings, D. Q. Mayne, and M. Diehl, "Model predictive control: theory, computation, and design," *Nob Hill Publishing*, vol. 2, 2017.

[45] K. Basil and C. Mark, *Model predictive control*. Switzerland: Springer International Publishing, 2016.

[46] Y. Xu, Y. Sun, and Y. Hou, "Multi-step model predictive current control of permanent-magnet synchronous motor," *Journal of Power Electronics*, vol. 20, no. 1, pp. 176–187, Jan. 2020, doi: 10.1007/S43236-019-00024-3/TABLES/3.

[47] M. J. Grimble and A. W. Ordys, "Predictive control for industrial applications," *Annu Rev Control*, vol. 25, pp. 13–24, Jan. 2001, doi: 10.1016/S1367-5788(01)00003-7.

[48] A. Ramdani and S. Grouni, "Dynamic matrix control and generalized predictive control, comparison study with IMC-PID," *Int J Hydrogen Energy*, vol. 42, no. 28, pp. 17561–17570, Jul. 2017, doi: 10.1016/J.IJHYDENE.2017.04.015.