Article

A Path to Industry 5.0 Digital Twins for Human–Robot Collaboration by Bridging NEP+ and ROS

Enrique Coronado *, Toshio Ueshiba and Ixchel G. Ramirez-Alpizar

Industrial Cyber-Physical Systems Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Tokyo 135-0064, Japan; t.ueshiba@aist.go.jp (T.U.); ixchel-ramirezalpizar@aist.go.jp (I.G.R.-A.)
* Correspondence: coronadozuniga.enrique@aist.go.jp

Abstract: The integration of heterogeneous hardware and software components to construct human-centered systems for Industry 5.0, particularly human digital twins, presents considerable complexity. Our research addresses this challenge by pioneering a novel approach that harmonizes the techno-centered focus of the Robot Operating System (ROS) with the cross-platform advantages inherent in NEP+ (a human-centered development framework intended to assist users and developers with diverse backgrounds and resources in constructing interactive human–machine systems). We introduce the nep2ros ROS package, aiming to bridge these frameworks and foster a more interconnected and adaptable approach. This initiative can be used to facilitate diverse development scenarios beyond conventional robotics, underpinning a transformative shift in Industry 5.0 applications. Our assessment of NEP+ capabilities includes an evaluation of communication performance utilizing serialization formats like JavaScript Object Notation (JSON) and MessagePack. Additionally, we present a comparative analysis between the nep2ros package and existing solutions, illustrating its efficacy in linking the simulation environment (Unity) and ROS. Moreover, our research demonstrates NEP+'s applicability through an immersive human-in-the-loop collaborative assembly. These findings offer promising prospects for innovative integration possibilities across a broad spectrum of applications, transcending specific platforms or disciplines.

Keywords: Human–Robot Collaboration; Industry 5.0; internet of robotic things; virtual reality; digital twins

check for updates

Citation: Coronado, E.; Ueshiba, T.; Ramirez-Alpizar, I.G. A Path to Industry 5.0 Digital Twins for Human–Robot Collaboration by Bridging NEP+ and ROS. *Robotics* **2024**, *13*, 28. https://doi.org/10.3390/robotics13020028

Academic Editor: Chris Lytridis

Received: 19 December 2023
Revised: 29 January 2024
Accepted: 30 January 2024
Published: 1 February 2024

1. Introduction

In the rapidly advancing landscape of technology, the vision of Industry 5.0, as endorsed by the European Commission, is gaining traction. This concept represents a paradigm shift in the industry, placing humanity at the center by respecting individuals' diversity, roles, talents, needs, and rights [1,2]. In essence, Industry 5.0 calls for the adoption of technologies that empower and celebrate the diverse talents and needs of industrial workers, establishing sustainable production processes that adhere to planetary boundaries, and creating industries that are more agile, adaptable, and flexible [1,3,4].

In the context of Industry 5.0, digital twins (DTs) play a crucial role [5]. It is imperative, however, to differentiate DTs from other digital entities. Traditional digital representations are manually constructed models of reality that remain static when real-world conditions change. Additionally, digital shadows automatically adapt the virtual model to reflect environmental changes, yet they lack a real-time feedback loop [6]. In contrast, DTs, as explained in [7], establish a bidirectional data exchange between the physical system and its virtual counterpart. In this context, digital representations of robotics systems—denoted in this article as robot digital twins (RDTs)—are often deployed for simulating, testing, and refining industrial processes [8]. As a critical method for realizing human-centricity, human digital twins (HDTs) (i.e., digital representations of humans and their interaction environment) have emerged as pivotal components of Industry 5.0 [5]. HDTs are often designed to enhance human capabilities, productivity, and overall well-being [9]. The convergence of RDT and HDT can enable the creation of holistic systems that bridge the gap between humans and technology by complementing human and robot strengths and enhancing human interaction and well-being. However, creating accurate digital twins requires expertise from various domains, such as engineering for physical modeling, computer science for software development, data science for analysis, and domain-specific specialists [9]. Furthermore, the system architectures of human digital twins (HDT) can involve the integration of numerous components in a scalable manner, encompassing smart devices, collaborative robots, augmented reality (AR), mixed reality (MR), virtual reality (VR) devices, and human–machine interfaces (HMIs) [5]. As detailed in [10], employing immersive devices and advanced visual sensors facilitates two-way interaction between humans and their environment. This interaction serves to mitigate potential risks while simultaneously enhancing operational efficiency and optimizing resource utilization. In this context, some examples have been reported in [11–13] in the area of teleoperation of robotics platforms and [14–16] in Human–Robot Collaboration. Hence, fostering cross-disciplinary collaboration is a pivotal factor in ensuring the precision and applicability of these digital twins. Furthermore, the imperative for cross-platform development arises when these digital replicas must seamlessly function across diverse hardware and software landscapes. This approach not only amplifies accessibility but also augments their overall utility.

NEP+ [17] is a framework under development, aiming to simplify the construction of interactive human–machine systems and enhance compatibility across diverse technologies and user groups, all while reducing development efforts. NEP+ is an IoT technology designed to facilitate the development of cyber-physical systems. For example, it can integrate the necessary elements to create hand-guiding collaborative robots, where robots work alongside humans under their guidance [10]. Therefore, this article primarily discusses the IoT capabilities inherent in NEP+. As a bridge between different platforms and disciplines, NEP+ facilitates smoother interactions and collaboration through a platform-independent middleware layer and user-friendly interfaces. NEP+ is Windows and macOS first, but it is also designed to be fully compatible with popular Linux-based systems like Ubuntu and Raspberry Pi OS. NEP+ tools are built on top of ZeroMQ [18]. This open-source universal and high-performance messaging library can connect code in almost all languages on many platforms. These tools empower ZeroMQ by providing serialization options, command lines, and user interfaces for network management and service discovery [17]. By embracing an inclusive and cross-platform approach, NEP+ tools aim to foster seamless integration, grant users autonomy, and fuel collaborative research endeavors spanning multiple disciplines. This includes the integration of interactive applications involving both physical and digital agents. Contributions of NEP+ to the Industry 5.0 principles are summarized below:

- Sustainable: NEP+ is designed to operate across various platforms, including Windows, MacOS, Linux, and mobile operating systems, enabling seamless connections. This flexibility empowers users to work with their existing systems, reducing the need for new hardware or OS switches. Consequently, NEP+ can be used to minimize electronic waste and optimize resource utilization in academic, industrial, or service environments.
- Human-Centered: NEP+ can be used to foster a human-centered Industry 5.0 by facilitating connectivity among digital twin components, enabling comprehensive interactions between physical and digital entities. Its diverse array of tools accommodates developers' varying expertise levels. Examples showcasing how NEP+ addresses integration challenges within multidisciplinary projects to construct human-centered systems are delineated in [17]. This inclusivity empowers developers across skill spectrums, enhancing effective collaboration within the system. Furthermore, NEP+ prioritizes user-centric design principles, emphasizing the development of user-friendly interfaces that enhance productivity, minimize errors, and drive user adoption.

- Resilient: NEP+ could potentially bolster industry resilience by enhancing the compatibility of cutting-edge technologies that might lack direct support in traditional platforms. This adaptability enables NEP+ to integrate and interact with emerging technologies, providing a flexible framework that harmonizes diverse hardware and software components. NEP+ ensures compatibility, fostering continued innovation in evolving technological landscapes.

2. Related Work and Contributions

The Robot Operating System (ROS) [19] has been widely used in robotics development, offering valuable tools and libraries. However, ROS encounters a fundamental limitation when creating applications for a broader audience, and expanding its usage beyond the realm of robotics presents notable challenges. This challenge primarily stems from ROS' exceptional performance in Linux-based environments, which inherently restricts its adoption outside this operating system. ROS2 [20], the next iteration of ROS, currently has varying levels of support across different platforms. However, its Windows support lags behind the comprehensive support available for Linux. Therefore, installing, resolving issues, and configuring ROS2 on a Windows system can be more complex than a Linux-based setup [17]. These factors collectively contribute to ROS' limitations when attempting to cater to a broader public, particularly those outside the traditional robotics sphere. Furthermore, many VR/AR platforms, such as the Oculus Pro and Microsoft Mixed Reality, are Windows-centric. This is because Windows is the world's most popular PC operating system, with over 70% market share. This gives VR/AR platform developers a larger potential product audience. In this context, the ROS Bridge Suite [21] was created to facilitate communication between ROS and non-ROS systems, aiming to tackle interoperability challenges. Solutions based on the ROS Bridge Suite, such as ROS-Sharp [22], have been widely used to build digital twins using the Unity game engine [8]. Recent examples using ROS-Sharp solutions include [23,24]. However, the ROS Bridge Suite encounters communication performance issues, particularly in handling complex and data-intensive applications (e.g., images and point-cloud data) [25]. Such limitations can impact the real-time interactions that are essential to digital twins and immersive environments. To enhance communication speed for the real-time transfer of camera images, an alternative approach presented in [25] is to replace the JavaScript Object Notation (JSON) format traditionally used in the ROS Bridge Suite with BSON (Binary JSON) format, and use TCP/IP instead of WebSockets. BSON is a binary-encoded serialization format that closely mirrors JSON. This strategy also incorporates a specialized server known as sigverse_rosbridge_server to handle the data exchange between Unity and ROS. When a Unity script advertises ROS messages, the primary thread of the sigverse_rosbridge_server generates a new thread for each topic, creating a communication channel. A similar strategy has been adopted in the Unity Robotics Hub, specifically in the ROS-TCP-Connector [26] and ROS-TCP-Endpoint [27] packages. The ROS-TCP-Connector, available as a Unity package, serves as a means to transmit ROS messages through conventional TCP-IP sockets to a specialized server known as ROS-TCP-Endpoint. Recent works, including those by Li et al. (2022) [28] and Pryor et al. (2023) [29], have successfully integrated the TCP-Connector and ROS-TCP-Endpoint packages. Nonetheless, these solutions are tailored specifically for Unity and robotics applications. This underscores the necessity for more versatile options like NEP+.

Building upon the solutions proposed in [25] and implemented in the ROS-TCP-Connector and ROS-TCP-Endpoint, we present an alternative solution to facilitate connectivity between ROS and non-ROS environments. Nevertheless, the flexibility features of NEP+ enable the exploration of new implementation possibilities. In this context, we demonstrate NEP+'s capability to streamline the integration of digital twins (DTs) using diverse programming environments, such as Python, ROS, and Unity. This multifaceted approach underscores NEP+'s suitability for various scenarios, extending beyond traditional robotics applications. The main contributions of this article are:

- The development of the nep2ros package and the conceptual design of HDTs using NEP+ and ROS (Section 3): The nep2ros package serves as a bridge between ROS and non-ROS environments through NEP+. This integration facilitator empowers the exchange of diverse sensory data types essential for effective Human–Robot Collaboration. This advancement plays a pivotal role in supporting the Industry 5.0 vision by enabling enhanced interoperability and communication between humans and robots. Furthermore, the architecture presented provides a general overview and a pathway towards Industry 5.0 applications using NEP+ and ROS.
- Experimental evaluation (Section 4): The evaluation process provides essential insights into communication performance within the NEP+ framework, thereby enhancing the understanding of serialization options. This knowledge is fundamental for optimizing data transmission, which is paramount for efficient and effective Human–Robot Collaboration.
- Implementation with real robotics systems (Section 6): The practical implementation showcased in this section offers a tangible demonstration of human-in-the-loop collaborative assembly using NEP+ and ROS tools. This dummy scenario using an industrial robot exemplifies the practical application of these technological advancements in facilitating collaborative human–robot interactions within an Industry 5.0 framework. Despite the simplicity of the designed demonstration, it serves as an illustrative example, effectively showcasing the potential impact of NEP+ in developing human-centric systems tailored for Industry 5.0 settings.

3. System Architecture of Human Digital Twin Using NEP+ and ROS

Wang et al. [5] proposed a comprehensive four-layer system architecture model outlining the elements governing HDTs. These layers include an agent layer, a data layer, an inference layer, and an implementation layer. The agent layer comprises human and physical entities, such as robots and manipulation objects. The data layer is responsible for extracting, transforming, transferring, storing, and managing access to data and metadata. Conversely, the inference layer handles processing, calculation, simulation, prediction, reasoning, and decision-making. Finally, the implementation layer categorizes inference data into specific service modules, delivering consumptive services to end-users. The human-centric focus of this model is particularly evident in the dependence of components on human entities. Notable examples include wearable devices that facilitate human factors analysis, ergonomics, usability, user experience techniques, and the integration of multi-modal and heterogeneous data sources describing the status of the human–machine interaction, equipment attributes, faults, or disturbances in the system. Furthermore, certain devices, including smartphones, smartwatches, AR/MR/VR headsets, and HMI, can effectively enhance human integration into a cyber system [30]. The data generated from these sources is utilized by both the system and humans to formulate intelligent reactions conveyed through feedback instructions or commands. Within the manufacturing system, components in these layers operate independently, generating substantial real-time data. These data are transmitted into cyberspace through the communication network for continuous analysis. IoT technologies, illustrated by NEP+, facilitate seamless interconnection. Employing quasi-standard, language-agnostic, and flexible formats like JSON for data transfer can enhance system cohesiveness, simplifying raw data consolidation and promoting collaboration across disciplines. However, specialized messaging formats may be favored for efficiency and real-time communication. An illustrative example is ROS2, which has been developed to optimize communication and control within robotic systems.

This article highlights the potential benefits of NEP+ for fostering seamless integration among diverse components outlined in the HDT model from [5]. The example proposed in this article encompasses five fundamental elements obtained from the model proposed in [5]: (1) data sensing, (2) data connectivity, (3) data processing, (4) simulation, and (5) interaction. Figure 1 illustrates a simplified system architecture for HDTs, demonstrating the interconnection of various elements through NEP+ and ROS. For instance, data sensing could involve an image acquisition module within an Android Studio-based smartphone application transmitting data to a data processing component. Similarly, an RGB-D camera could relay information from a computer running Ubuntu to the DT-enabled computer on Windows that needs real-time updates of a generated point cloud. Additional examples might encompass robot joint angle values, input and position data from VR controllers, and smartwatches' inertial values and heart rate information. In contrast, data processing can be represented by any AI-enabled framework or algorithm, such as deep learning methods using TensorFlow, face, hands, and skeleton detection using Mediapipe, OpenAI's deep reinforcement learning, and natural language processing tools. Data processing components transform raw data collected by sensors into high-level human-related representations that can be used to produce human–robot and human–machine interaction. The simulation of reality is realized through immersive virtual environments crafted within game engines like Unity and Unreal. Furthermore, various frameworks, such as the Software Development Kit (SDK) and third-party tools (e.g., Android Studio, Node.js, and AR Core), can be used to design user interfaces operated on displays or smart devices, enabling different interactive experiences. The depicted architecture represents a general model not designed for specific applications. The evaluation performed in Section 4 aims to simulate the typical data exchanged within components of the presented HDT system architecture using NEP+ and ROS bridge alternatives. Conversely, Section 6 combines simple sensing, data connectivity, processing, simulation, and interaction elements to showcase a practical, real-world demonstration, specifically an immersive, human-in-the-loop collaborative assembly application.

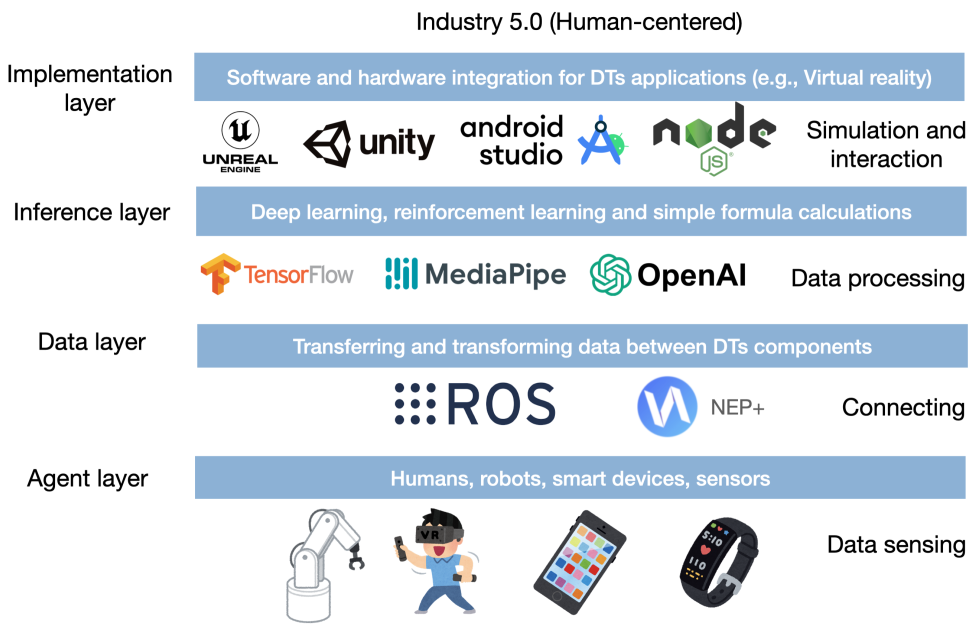
Industry 5.0 (Human-centered)
Implementation layer
Software and hardware integration for DTs applications (e.g., Virtual reality)
UNREAL ENGINE
unity
android studio
Simulation and interaction
Inference layer
Deep learning, reinforcement learning and simple formula calculations
TensorFlow
MediaPipe
OpenAI
Data processing
Data layer
Transferring and transforming data between DTs components
ROS
NEP+
Connecting
Agent layer
Humans, robots, smart devices, sensors
Data sensing

Figure 1. Conceptual system architecture for human digital twins (HDTs) using NEP+ and ROS as tools enabling connecting components. This architecture is inspired by the general architecture for Industry 5.0 presented in [5] and we adapted it for Human–Robot Collaboration (HRC).

NEP+ Tools and the nep2ros Package

This section outlines a prototype system architecture for constructing human digital twins (HDTs) using both ROS and NEP+ for connecting components. However, these two frameworks espouse distinct visions and cater to different user types and applications. Additionally, their socket libraries are incompatible. To bridge this gap, this article introduces an initial version of the nep2ros package. This contribution stands out as one of the main novelties of this article, as no previous work has presented a solution that seamlessly connects ROS and NEP+ before.

The NEP+ application and middleware layers, outlined in [17], consist primarily of the NEP+ App, NEP-CLI command-line interface, and the NEP libraries. NEP+ utilizes ZeroMQ, an open-source messaging library, to establish high-performance and cross-platform socket-based communication. ZeroMQ, using the ZeroMQ Message Transport Protocol (ZMTP), ensures adaptability across various programming languages and operating systems, making it ideal for building IoT and robotics modules. NEP+ tools simplify ZeroMQ's usage, enabling the smooth development of interactive applications. The NEP+ App serves as the central hub, enabling distributed system connections through ZeroMQ sockets and supporting functionalities like service discovery and message monitoring. On the other hand, NEP-CLI, developed with Node.js, offers a command-line interface. It provides an alternative to the NEP+ App for developers familiar with conventional command-line tools. Users have the option to utilize either the NEP+ App or NEP-CLI as a central hub to facilitate communication between components within a local network, such as computers interconnected via Wi-Fi or Ethernet. These tools utilize NEP libraries to establish connections and route messages to their designated destinations. In both the NEP+ App and NEP-CLI, each communication channel, identified by unique topics, is managed separately. This individual handling ensures that the performance remains optimized, particularly when numerous communication channels are established simultaneously. This approach prevents potential performance degradation when managing multiple communication channels concurrently. Furthermore, the NEP libraries offer a high-level abstraction of the underlying ZeroMQ socket implementations. This abstraction allows for smooth and efficient inter-process communication among diverse modern programming languages. This article introduces and evaluates the initial version of the nep2ros ROS package, a novel component within the NEP+ middleware layer. Similar to the ROS Bridge server, the nep2ros package facilitates the conversion of ROS messages to JSON and vice versa, effectively bridging the gap between ROS and non-ROS environments. However, rather than Websockets or traditional TCP/IP socket libraries, NEP+ implements ZeroMQ sockets. As NEP+ can function independently, a service discovery mechanism akin to ROS becomes necessary. This capability, illustrated in Figure 2, is facilitated through interfaces like the NEP-CLI or the NEP+ App. While this approach needs an extra component, it streamlines the connectivity among various agents (e.g., robots, smart devices) developed using diverse environments, encompassing but not limited to ROS, as illustrated in Figure 3. The nep2ros package will be licensed under the Mozilla Public License 2.0. Detailed information and support on using and installing NEP+ tools can be accessed from the official documentation page [31].

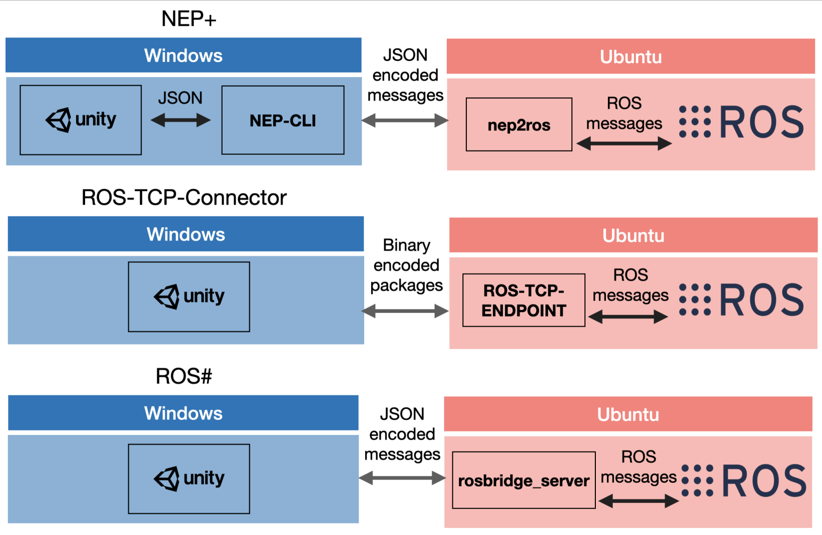

Figure 2. Differences in system architectures: NEP+, ROS-TCP-Connector, and ROS# facilitating connectivity between ROS nodes in Ubuntu and a Unity-based simulation on a Windows platform.

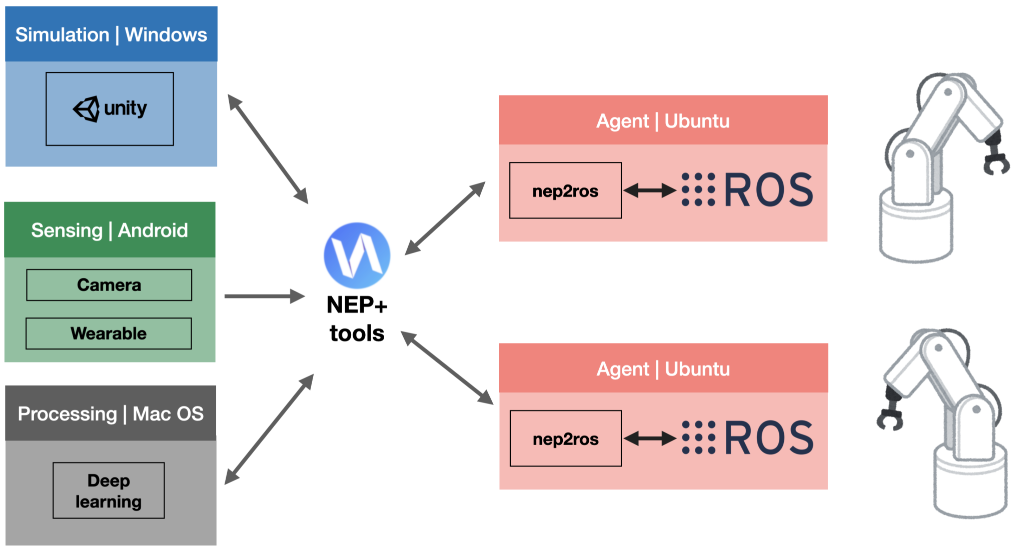

Figure 3. Example of an HDT system architecture potentially enabled using NEP+ tools for connecting components from different environments. Either the NEP-CLI or NEP+ App interfaces represent NEP+ tools.

4. Assessing Communication Performance in NEP+ for Common Human Digital Twin Scenarios

Our study consists of two stages. During stage 1, we evaluate communication performance by analyzing various formats and implementations within NEP+. Moving to stage 2, we conduct a comparative analysis involving the initial implementation of the nep2ros bridge solution. This evaluation contrasts the effectiveness of bridge solutions connecting ROS and Unity. Our evaluation revolves around latency, a cornerstone metric in robotics and control systems, that signifies the time taken for a data packet's journey from sender to receiver. We specifically focus on round-trip time (RTT) within a publisher–subscriber system. RTT assumes paramount importance when synchronized clocks are absent between the publisher and subscriber or when a comprehensive understanding of message transit time is necessary. This metric encapsulates the duration for a message to traverse from the publisher to the subscriber and the acknowledgment (i.e., the subscriber's receipt of the message) to travel back to the publisher. Figure 4 illustrates the traditional method for calculating RTT. This method determines RTT by subtracting the sending time ($T1$) from the receipt time ($T2$) at the publisher node. Subsequently, the one-way latency can be estimated as half of the RTT.

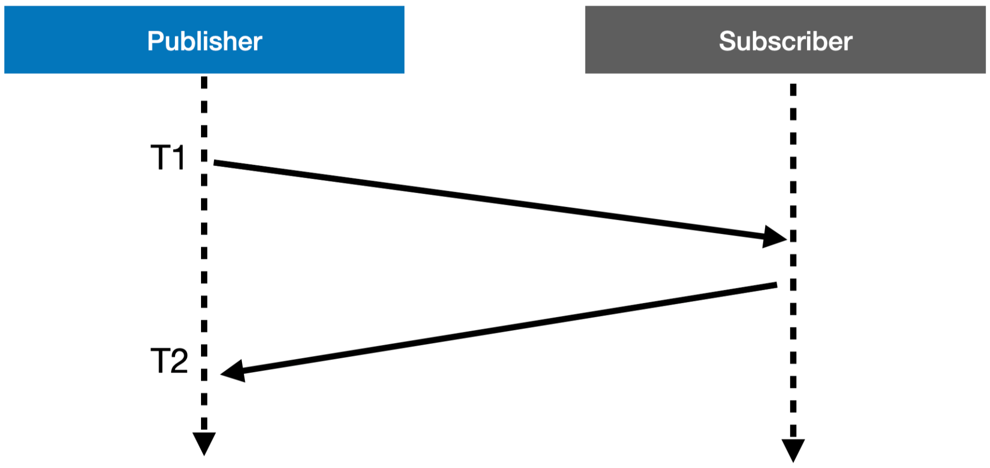

Figure 4. Publish–subscribe latency measurement using round-trip time.

Our evaluation encompassed message sizes of 100 KB, 1 MB, and 10 MB to analyze the impact of different data volumes on performance. For instance, a 100 KB message could contain serialized sensor readings from various sensors, such as ultrasonic and IMU, which are crucial in robotics. In artificial intelligence, such a message might encapsulate MediaPipe skeleton data depicting human poses. Likewise, a 100 KB message could compile a package of environmental sensor data in IoT applications, such as temperature, humidity, and air quality metrics. Expanding to larger message sizes of 1 MB and 10 MB: In these contexts, a 1 MB message might hold low-resolution camera images and point cloud data. A 10 MB message, on the other hand, could represent dense point cloud data or high-resolution camera images. These varying data sizes are pertinent to robotics, artificial intelligence, and IoT applications. The connection between computers in the experimental scenarios was established using Ethernet, ensuring a direct and stable link between them. It is important to note that during the tests, no additional network nodes or concurrent processes were active, simulating an optimal testing environment. We acknowledge that real-world conditions, such as varying network conditions or hardware loads, can impact latency measurements. However, for the purposes of this article, our evaluation focused on examining the baseline performance of NEP+ in a controlled and nearly ideal scenario. We aimed to isolate the communication performance between ROS and Unity through NEP+ in a stable setting to provide a fundamental understanding of its capabilities.

In this experiment, we conducted continuous message transfers at 30 frames per second (FPS) for 100 KB and 1 MB data sizes. However, transmitting a 10 MB (Megabyte) message at a consistent rate of 30 FPS significantly strains a Gigabit Ethernet connection. To illustrate this, such continuous transmission at this rate demands a bandwidth calculated as 10 MB $*$ 30 FPS = 300 MB/s (Megabytes per second). Comparatively, Gigabit Ethernet offers a theoretical maximum bandwidth of 1 Gbps (Gigabit per second), equivalent to approximately 125 MB/s. Consequently, the sustained data transfer rate required by the 10 MB message considerably surpasses the capacity of Gigabit Ethernet. Attempting such transmission rates could lead to network saturation, causing potential congestion and performance degradation. To mitigate this issue, adjustments were made by reducing the publisher frame rate to 5 FPS for the 10 MB messages. This adjustment reduced the required bandwidth to 10 MB $*$ 5 FPS = 50 MB/s (Megabytes per second), effectively averting network congestion and ensuring smoother transmission.

Experiments were conducted over Ubuntu 20 and ROS Noetic on a Galleria Workstation with an Intel i9-13900KF processor. Unity was executed on an Asus ROG Zephyrus G15 with an AMD Ryzen 9 5900HS processor using Windows 11 Pro as the operating system.

4.1. Communication Performance of NEP+ Using Different Serialization Formats

In this study stage, we evaluate NEP+'s communication performance across various serialization formats through a comprehensive series of tests detailed below. We consider the JSON and MessagePack [32] for this evaluation. JSON is a widely used lightweight data-interchange format known for its human-readable nature and ease of parsing across multiple platforms and programming languages. Its simplicity and broad support make it a preferred choice. However, JSON can lack efficiency, especially with complex data structures, resulting in larger payloads and slower transmission speeds due to its verbosity. In contrast, MessagePack is a binary-based serialization format offering a more compact representation than JSON. Designed for rapid serialization and deserialization, MessagePack's key advantage lies in its compactness, resulting in smaller message sizes and reduced network bandwidth usage. However, this efficiency comes at the cost of human-readability, making direct inspection and debugging less straightforward than JSON.

Figures 5–7 show the latency and throughput results when using JSON and MessagePack as serialization formats. As anticipated, employing MessagePack results in lower latencies compared to JSON. Nonetheless, MessagePack is solely integrated into the Python version of NEP+ libraries. This observation hints at the potential enhancement of communication performance by incorporating MessagePack into other NEP+ implementations, like Java and C#.

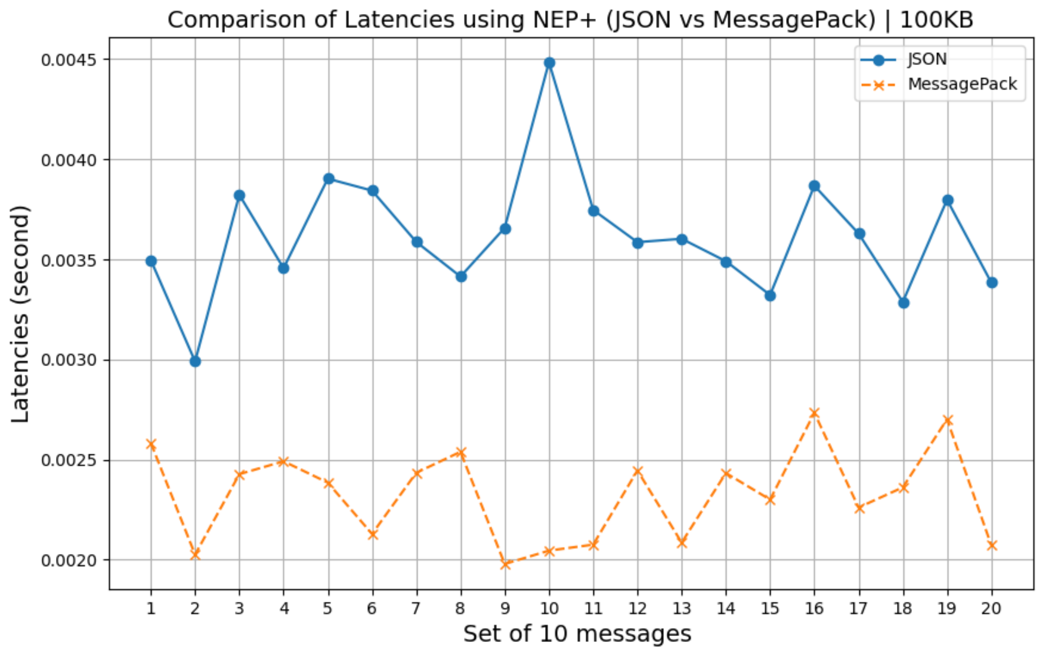

Figure 5. Average latency results for sets of 10 messages utilizing string data of 100 KB size.

4.2. Performance Comparison with State-of-the-Art Solutions

In this comparative study, we evaluate the communication performance of the NEP+ architecture and the nep2ros bridge solution proposed in this article against two popular ROS-Unity communication solutions (ROS-Sharp and ROS-TCPConnector). Our framework selection was informed by an extensive literature review [8], emphasizing their prevalent usage and acceptance within the ROS community.

Figures 8–10 exhibit the latency and throughput results obtained from nep2ros, ROS#, and ROS-TCPConnector across different message sizes (100 KB, 1 MB, and 10 MB). Notably, for 100 KB messages, NEP+ demonstrates slightly lower latency values compared to ROS-TCPConnector and significantly lower values than ROS-Sharp. However, for 1 MB and 10 MB messages, ROS-TCPConnector exhibits marginally reduced latency results compared to NEP+. These findings potentially arise from the current implementation of nep2ros utilizing the JSON format for serialization, while ROS-TCPConnector employs a binary format. As highlighted in Section 4.1, the adoption of an efficient binary serialization format like MessagePack could potentially enhance NEP+'s latency in future implementations. Both NEP+ and ROS-TCPConnector exhibit a superior performance in comparison with ROS# for 1 MB and 10 MB, rendering ROS# less suitable for transmitting large sized messages, such as images and point cloud data.

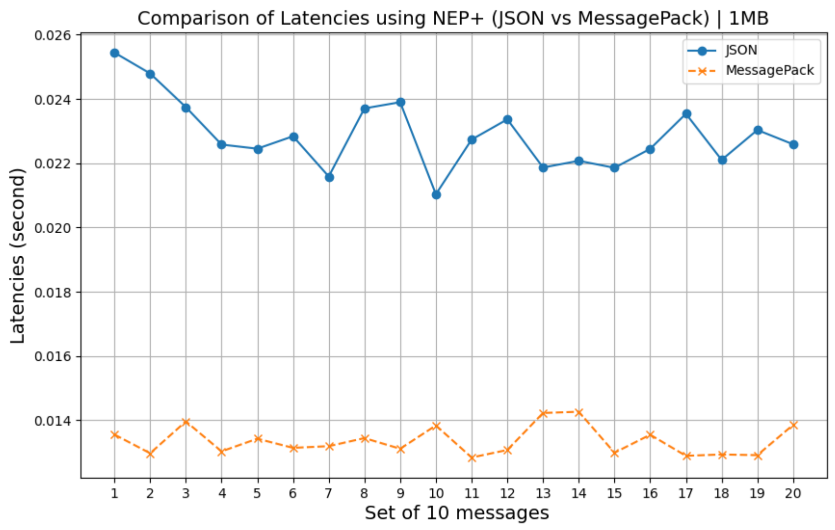

Figure 6. Average latency results for sets of 10 messages utilizing string data of 1 MB size.

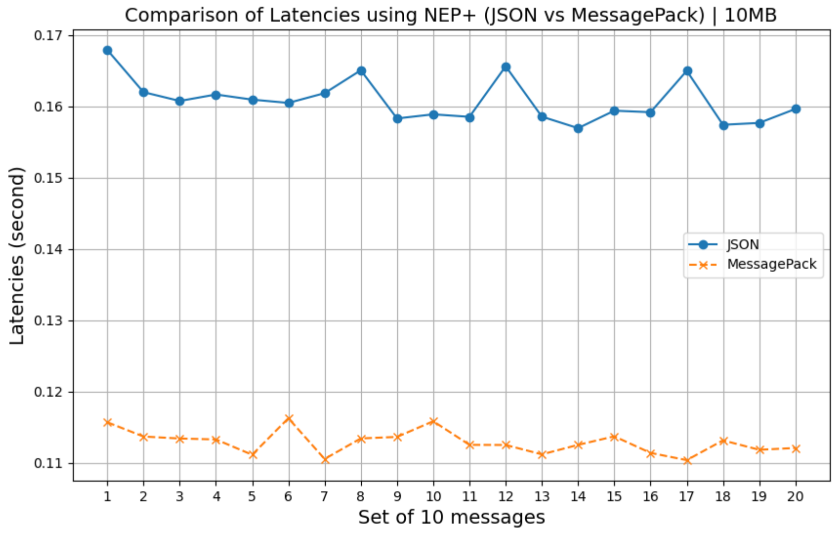

Figure 7. Average latency results for sets of 10 messages utilizing string data of 10 MB size.

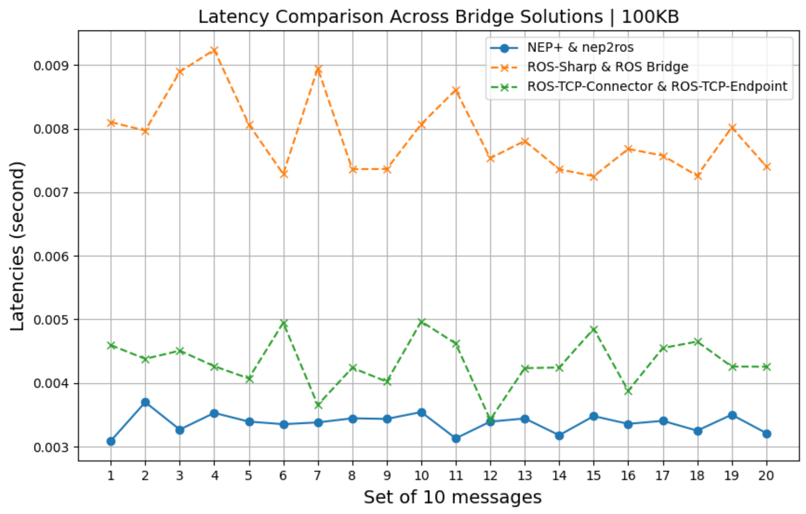

Figure 8. Comparison of bridge solutions between Unity and ROS: average latency results for sets of 10 messages using string data with a size of 100 KB.

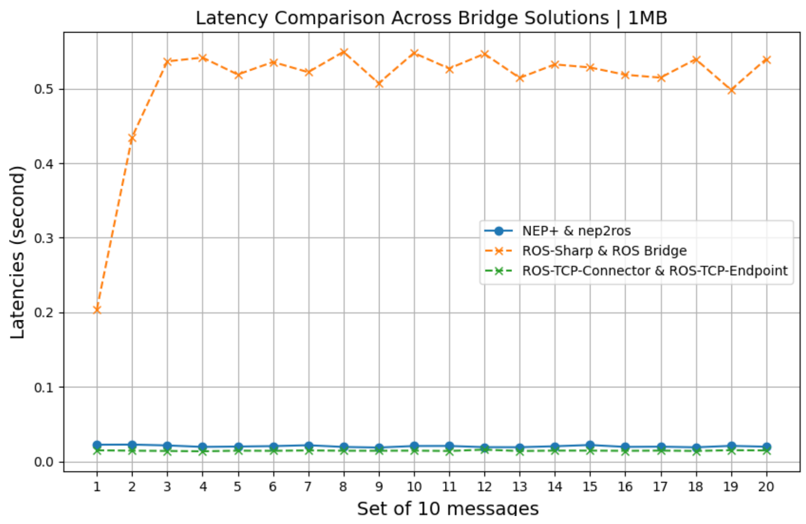

Figure 9. Comparison of bridge solutions between Unity and ROS: average latency results for sets of 10 messages using string data with a size of 1 MB.

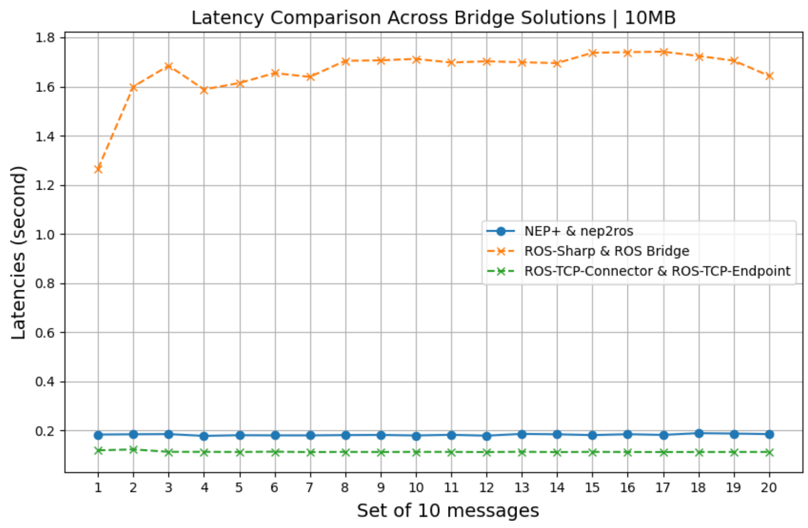

Figure 10. Comparison of bridge solutions between Unity and ROS: average latency results for sets of 10 messages using string data with a size of 10 MB.

5. Validating NEP+ through a Dummy Example of Immersive Human-Robot Collaboration for Peg-in-Hole Assembly

This section presents an initial implementation of an HDT system utilizing NEP+ to facilitate integration between Unity and ROS. This demonstration aims to validate the compatibility of the communication tools detailed in this article with real robotics systems and immersive devices, facilitating hand-guided Human–Robot Collaboration (HRC). The implementation of complex real-world assembly applications requires analyzing diverse human factors, real-time specifications, performance metrics, and uncertain variables. However, these aspects are beyond the scope of this article. Examples of articles discussing many of these aspects can be found in [2,33]. Challenges related to the use of immersive devices for HRC are described in [8,34,35]. The demonstration's primary goal is to enable remote collaboration with a UR5e robot in a product assembly scenario. Leveraging the teleportation capabilities inherent in industrial robots, this system allows humans to remotely assist robot operations in environments where human presence might be restricted, unsafe, or impractical [36]. The integration aims to foster a collaborative environment by merging human and robotic capabilities to enhance the safety and efficiency of manufacturing operations.

Figure 11 depicts the proposed HDT architecture, building upon the general architecture outlined in Section 4. Within this architecture, NEP+ connects the simulation environment and the robot sensing and processing data modules operated in ROS. Additionally, it links to an image acquisition system managed through a Node.js interface. The headset employed in this demonstration is the Oculus Quest 2, renowned for its primary compatibility with Windows. Unity is running on an Asus ROG Zephyrus G15 equipped with an AMD Ryzen 9 5900HS processor featuring Radeon Graphics at 3.30 GHz, an Nvidia RTX 3080 GPU, and Windows 11 Pro. The robot control algorithms are executed on a desktop computer with Ubuntu 20 and ROS Noetic. The camera used is 4K Ultra HD Webcam Logicool BRIO C1000eR.

The robot sensing and processing system employs a reinforcement learning-based force-controlled mechanism to enable peg-in-hole insertions with low clearance as presented in [37]. The RL-based system uses the soft actor critic (SAC) algorithm and, given the pose of the hole as input, it is capable of autonomously inserting pegs without using excessive forces to prevent undesired emergency stops by using a reward function based on a desired insertion force and collision penalties. The policy learns the insertion path from the starting pose and the position-force control parameters to control the robot. The details of the implementation can be found in [37]. In this setup, the simulation incorporates joint state data from the UR5e robot to update the virtual model, force-torque data from the sensor placed at the robot's end-effector, and video images for providing feedback on the insertion's status to the user (Figure 12). These visual cues, complemented by image values displayed in the VR system, enable humans to ascertain the successful completion of the robot's insertion task. The user will push the "Success" or "Failure" buttons located at the center of the VR screen (see Figure 12); this information is sent to the robot as a bool value (true for a successful insertion and false otherwise), for the robot to try the insertion again or to move to the next task. If the robot needs to try again, it will ask the user for feedback on which direction to modify the hole pose in to be able to succeed with the insertion. The user will provide information on how much pose error it had on each of the axes through the slide bars located at the top right corner of the VR screen (see Figure 12), then by pushing the "Try again" button, this information will be sent to robot for it to try again. The robot will use this information as a new input for the RL policy to try to insert the peg again; it should be noted that the RL policy is not being updated, the user just changed the input to the policy. This loop will be repeated until the insertion is successful. Like this, the human will help the robot to find the hole and will check whether the insertion was successful or not (Figure 13).

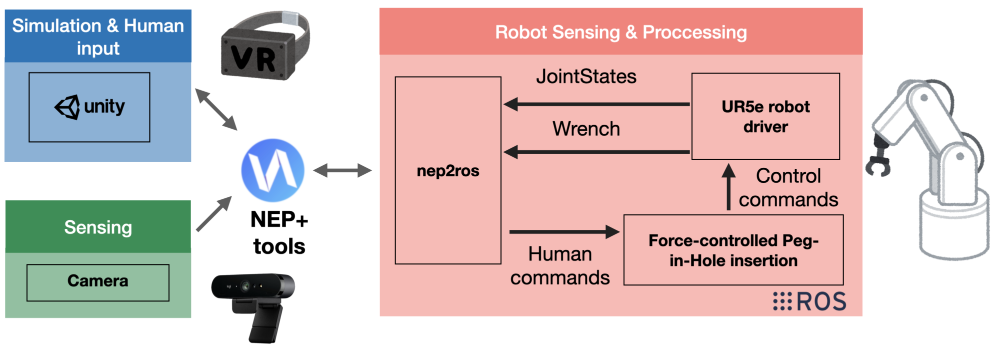

Figure 11. Simple HDTs architecture enabling remote and immersive human-in-the-loop collaborative assembly used in the proposed demonstration.

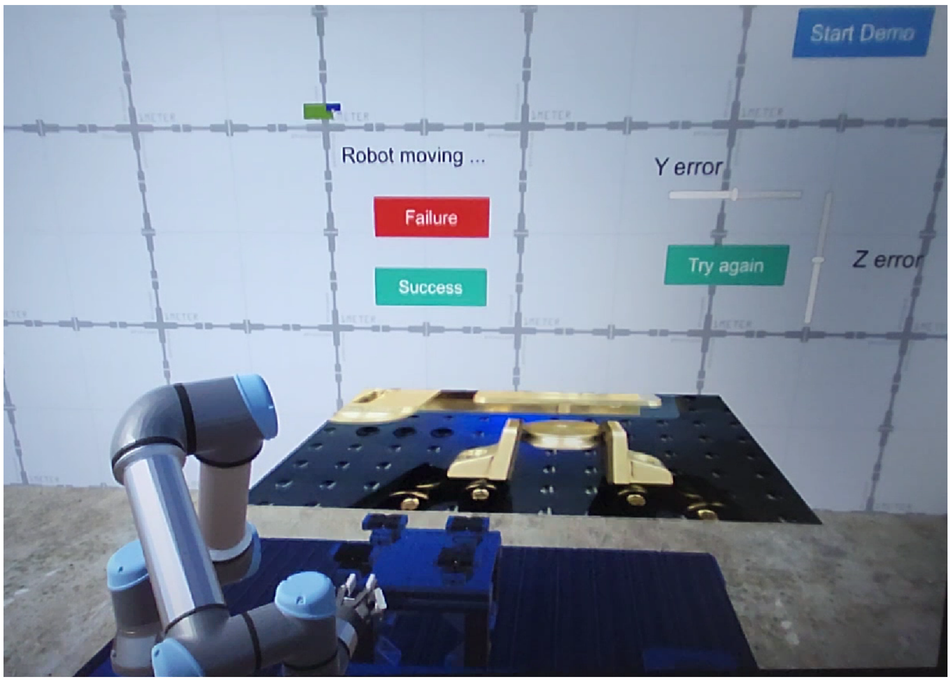

Figure 12. Digital twin executed in Unity and connected to ROS using NEP+.

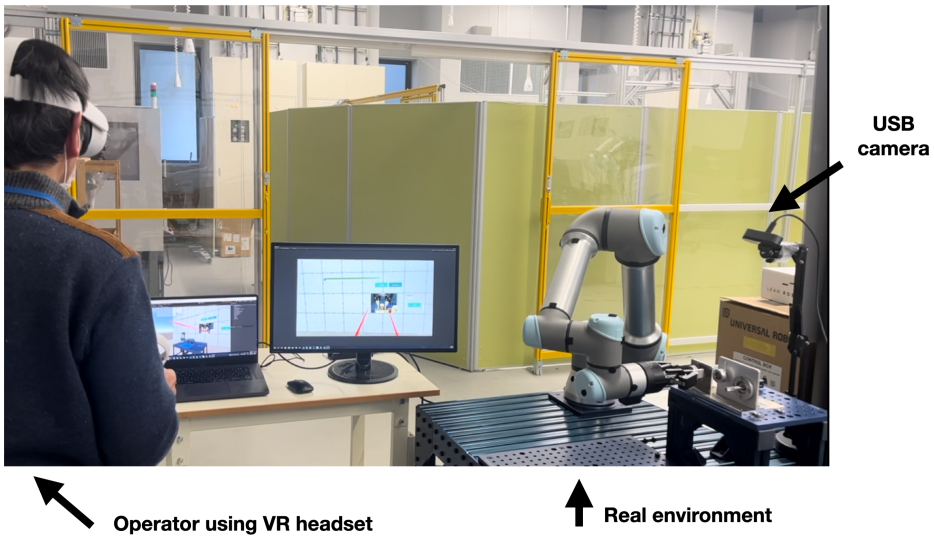

Figure 13. Real user testing the proposed demonstration.

6. Conclusions and Future Work

This article presents a developmental solution for integrating the diverse elements within HDTs. The results showcase that utilizing NEP+ libraries and tools leads to lower latencies compared to the widely used ROS Bridge suite, while demonstrating similar performance to ROS-TCP-Connector when connecting Unity and ROS. Moreover, the real-world demonstration of a human-in-the-loop DT validated the technological suitability of NEP+ in integrating heterogeneous software architectures.

Although this article primarily highlights Unity and ROS, NEP+'s defining characteristic lies in its adaptability, allowing for expansion to various development frameworks. As part of our forthcoming efforts, we intend to implement and evaluate NEP+ tools aimed at establishing connections with alternative simulation platforms, including the Unreal Engine and Nvidia Omniverse. Moreover, integrating ROS with platforms like Unity involves resolving disparities in their coordinate systems and addressing other complexities. However, enabling communication between components is the initial step towards facilitating such integration. Additionally, the adaptable nature of NEP+ can accommodate solutions based on binary encodings, such as MessagePack. As evidenced in this article, this approach can be used to enhance the performance of the nep2ros package, which is currently limited to JSON. Our plans also include the development and testing of a version of nep2ros tailored specifically for ROS2.

In HDT applications, the communication latency between a digital twin and the robot control system can play a relevant role in shaping the user experience of human–robot interaction (HRI). Increased latency can negatively impact the realism and immersion, diminishing the intuitiveness and responsiveness of the system. Moreover, in scenarios demanding precise control, like teleoperation, communication delays may result in unexpected movements or actions. While variations in network conditions or hardware loads can affect latency, assessing these factors comprehensively is beyond the objectives of this article. Therefore, we prioritized evaluating the intrinsic communication performance facilitated by NEP+ between the specified systems. Future research endeavors could explore the impact of diverse environmental conditions on NEP+'s latency in complex settings. Examples can involve the testing of NEP+ in scenarios comprising many publishers and subscribers from diverse devices, operating systems, and agents like robots. This exploration can be used to assess NEP+'s scalability, performance, and interoperability in handling many data streams from various sources. For this, we plan to expand the real-world demonstration introduced in this article by adding different data processing sources for human understanding and multi-agent interaction.

Also, HDTs could be used for the continual learning of the robotic system. In the demonstration shown in Section 6, the human will help the robot to find the precise location of the hole; this information could be used to continue to train the RL-based system on the robot side, so the robot can reduce its failed attempts.

Author Contributions: Conceptualization, E.C., T.U. and I.G.R.-A. methodology, E.C.; formal analysis, E.C.; investigation, E.C. and I.G.R.-A.; resources, E.C., T.U. and I.G.R.-A.; data curation, E.C.; writing—original draft preparation, E.C.; writing—review and editing, E.C., T.U. and I.G.R.-A.; visualization, E.C.; supervision, E.C. and I.G.R.-A.; project administration, I.G.R.-A.; funding acquisition, I.G.R.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Augmented Reality	AR
Digital Twin	DT
Frames per second	FPS
Human Digital Twin	HDT
Human–Machine Interfaces	HMI
Human–Robot Collaboration	HRC
JavaScript Object Notation	JSON
Mixed Reality	MR
Robot Digital Twin	RDT
Robot Operating System	ROS
Round-Trip Time	RTT
Software Development Kit	SDK
Virtual Reality	VR
ZeroMQ Message Transport Protocol	ZMTP

References

1. De Nul, L.; Breque, M.; Petridis, A. *Industry 5.0, towards a Sustainable, Human-Centric and Resilient European Industry*; Technical Report; General for Research and Innovation (European Commission): Brussels, Belgium, 2021.
2. Coronado, E.; Kiyokawa, T.; Ricardez, G.A.G.; Ramirez-Alpizar, I.G.; Venture, G.; Yamanobe, N. Evaluating quality in human–robot interaction: A systematic search and classification of performance and human-centered factors, measures and metrics towards an industry 5.0. *J. Manuf. Syst.* **2022**, *63*, 392–410. [CrossRef]
3. Xu, X.; Lu, Y.; Vogel-Heuser, B.; Wang, L. Industry 4.0 and Industry 5.0—Inception, conception and perception. *J. Manuf. Syst.* **2021**, *61*, 530–535. . [CrossRef]
4. Nahavandi, S. Industry 5.0—A human-centric solution. *Sustainability* **2019**, *11*, 4371. . [CrossRef]
5. Wang, B.; Zhou, H.; Li, X.; Yang, G.; Zheng, P.; Song, C.; Yuan, Y.; Wuest, T.; Yang, H.; Wang, L. Human Digital Twin in the context of Industry 5.0. *Robot.-Comput.-Integr. Manuf.* **2024**, *85*, 102626. [CrossRef]
6. van der Aalst, W.M.; Hinz, O.; Weinhardt, C. Resilient digital twins. *Bus. Inf. Syst. Eng.* **2021**, *63*, 615–619. [CrossRef]
7. Sepasgozar, S.M. Differentiating digital twin from digital shadow: Elucidating a paradigm shift to expedite a smart, sustainable built environment. *Buildings* **2021**, *11*, 151. [CrossRef]
8. Coronado, E.; Itadera, S.; Ramirez-Alpizar, I.G. Integrating Virtual, Mixed, and Augmented Reality to Human–Robot Interaction Applications Using Game Engines: A Brief Review of Accessible Software Tools and Frameworks. *Appl. Sci.* **2023**, *13*, 1292. [CrossRef]
9. Wang, B.; Zheng, P.; Yin, Y.; Shih, A.; Wang, L. Toward human-centric smart manufacturing: A human-cyber-physical systems (HCPS) perspective. *J. Manuf. Syst.* **2022**, *63*, 471–490. [CrossRef]
10. Inkulu, A.K.; Bahubalendruni, M.R.; Dara, A.; SankaranarayanaSamy, K. Challenges and opportunities in human robot collaboration context of Industry 4.0—A state of the art review. *Ind. Robot. Int. J. Robot. Res. Appl.* **2021**, *49*, 226–239. [CrossRef]
11. Whitney, D.; Rosen, E.; Ullman, D.; Phillips, E.; Tellex, S. Ros reality: A virtual reality framework using consumer-grade hardware for ros-enabled robots. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–9.
12. Togias, T.; Gkournelos, C.; Angelakis, P.; Michalos, G.; Makris, S. Virtual reality environment for industrial robot control and path design. *Procedia CIRP* **2021**, *100*, 133–138. [CrossRef]
13. Zhou, T.; Zhu, Q.; Du, J. Intuitive robot teleoperation for civil engineering operations with virtual reality and deep learning scene reconstruction. *Adv. Eng. Inform.* **2020**, *46*, 101170. [CrossRef]
14. Weber, D.; Kasneci, E.; Zell, A. Exploiting Augmented Reality for Extrinsic Robot Calibration and Eye-based Human-Robot Collaboration. In Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction, Sapporo, Japan, 7–10 March 2022; pp. 284–293.
15. Tuli, T.B.; Manns, M.; Zeller, S. Human motion quality and accuracy measuring method for human–robot physical interactions. *Intell. Serv. Robot.* **2022**, *15*, 503–512. [CrossRef]
16. Dimitropoulos, N.; Togias, T.; Michalos, G.; Makris, S. Operator support in human–robot collaborative environments using AI enhanced wearable devices. *Procedia CIRP* **2021**, *97*, 464–469. [CrossRef]
17. Coronado, E.; Yamanobe, N.; Venture, G. NEP+: A Human-Centered Framework for Inclusive Human-Machine Interaction Development. *Sensors* **2023**, *23*, 9136. . [CrossRef]
18. ZeroMQ Community. ZeroMQ: The Asynchronous Messaging Library. 2023. Available online: https://www.zeromq.org/ (accessed on 31 October 2023).
19. Koubâa, A. (Ed.) *Robot Operating System (ROS)*; Springer: Berlin/Heidelberg, Germany, 2017; Volume 1.
20. Maruyama, Y.; Kato, S.; Azumi, T. Exploring the performance of ROS2. In Proceedings of the 13th International Conference on Embedded Software, Pittsburgh PA, USA, 1–7 October 2016; pp. 1–10.
21. ROS Community. ROS Bridge Suite. 2023. Available online: https://wiki.ros.org/rosbridge_suite (accessed on 31 October 2023).
22. Siemens AG. ROS Sharp. 2023. Available online: https://github.com/siemens/ros-sharp (accessed on 31 October 2023).
23. Galarza, B.R.; Ayala, P.; Manzano, S.; Garcia, M.V. Virtual Reality Teleoperation System for Mobile Robot Manipulation. *Robotics* **2023**, *12*, 163. [CrossRef]
24. Shamaine, C.X.E.; Qiao, Y.; Henry, J.; McNevin, K.; Murray, N. RoSTAR: ROS-based telerobotic control via augmented reality. In Proceedings of the 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP), Tampere, Finland, 21–24 September 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
25. Inamura, T.; Mizuchi, Y. SIGVerse: A cloud-based VR platform for research on multimodal human–robot interaction. *Front. Robot. AI* **2021**, *8*, 549360. [CrossRef] [PubMed]
26. Unity Technologies. ROS-TCP-Connector. 2023. Available online: https://github.com/Unity-Technologies/ROS-TCP-Connector (accessed on 31 October 2023).
27. Unity Technologies. ROS-TCP-Endpoint. 2023. Available online: https://github.com/Unity-Technologies/ROS-TCP-Endpoint (accessed on 31 October 2023).
28. Li, J.; Liu, M.; Wang, W.; Hu, C. Inspection Robot Based on Offline Digital Twin Synchronization Architecture. *IEEE J. Radio Freq. Identif.* **2022**, *6*, 943–947. [CrossRef]

29. Pryor, W.; Wang, L.J.; Chatterjee, A.; Vagvolgyi, B.P.; Deguet, A.; Leonard, S.; Whitcomb, L.L.; Kazanzides, P. A Virtual Reality Planning Environment for High-Risk, High-Latency Teleoperation. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 11619–11625.
30. Nunes, D.S.; Zhang, P.; Silva, J.S. A survey on human-in-the-loop applications towards an internet of all. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 944–965. [CrossRef]
31. Developers, N. NEP+ Documentation. 2023. Available online: https://enrique-coronado.gitbook.io/nep-docs/ (accessed on 31 October 2023).
32. MessagePack Community. MessagePack. 2023. Available online: https://msgpack.org/ (accessed on 31 October 2023).
33. Eswaran, M.; kumar Inkulu, A.; Tamilarasan, K.; Bahubalendruni, M.R.; Jaideep, R.; Faris, M.S.; Jacob, N. Optimal layout planning for human robot collaborative assembly systems and visualization through immersive technologies. *Expert Syst. Appl.* **2024**, *241*, 122465. [CrossRef]
34. Eswaran, M.; Gulivindala, A.K.; Inkulu, A.K.; Raju Bahubalendruni, M. Augmented Reality-Based Guidance in Product Assembly and Maintenance/Repair Perspective: A State of the Art Review on Challenges and Opportunities. *Expert Syst. Appl.* **2023**, *213*, 118983. . [CrossRef]
35. Badia, S.B.i.; Silva, P.A.; Branco, D.; Pinto, A.; Carvalho, C.; Menezes, P.; Almeida, J.; Pilacinski, A. Virtual reality for safe testing and development in collaborative robotics: Challenges and perspectives. *Electronics* **2022**, *11*, 1726. [CrossRef]
36. Li, C.; Zheng, P.; Li, S.; Pang, Y.; Lee, C.K. AR-assisted digital twin-enabled robot collaborative manufacturing system with human-in-the-loop. *Robot.-Comput.-Integr. Manuf.* **2022**, *76*, 102321. [CrossRef]
37. Beltran-Hernandez, C.C.; Petit, D.; Ramirez-Alpizar, I.G.; Harada, K. Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach. *Appl. Sci.* **2020**, *10*, 6923. [CrossRef]