Wireless Robotic Motion Controller: System Architecture and Latency Assessment

Jan Petershans[2], Florian Lehn[2], Jan Herbst[2], and Hans D. Schotten[1,2]

[1]University of Kaiserslautern-Landau (RPTU), Germany, `{lastname}@rptu.de`
[2]German Research Center for Artificial Intelligence (DFKI), Germany
`{firstname}.{lastname}@dfki.de`

Abstract

Telerobotics has the potential to transform robotic operations across various domains, including industrial automation, medical robotics, and the chemical sector, where accurate and timely interventions are paramount. Wireless communication is pivotal in expanding the practical applications of telerobotic systems, enabling untethered, and flexible operation. To achieve precise teleoperation, low-latency motion controllers are required that enable seamless and intuitive human-machine interaction. This work presents the design and implementation of a wireless motion controller device for accurate telerobotic operation within the Robot Operating System (ROS) framework. By combining optical-active position tracking using a high-precision Motion Capture (MoCap) system and low-latency wireless communication, the setup facilitates a responsive and interactive human-robot interface. Experimental evaluations of the integrated system demonstrate accurate teleoperation control of a 7-axis collaborative robot (cobot). Additionally the impact of the different system components on the end-to-end latency is analyzed with robotic actuation and processing time of the MoCap system identified as the primary contributors. The results confirm the viability of wireless teleoperation and highlight opportunities for future research and optimization in telerobotic systems.

Please note: This is a preprint of a paper which has been accepted for publication at 21st IEEE International Conference on Factory Communication Systems (WFCS 2025).

Index Terms

Wireless Teleoperation; Human-Machine Interaction; Robot Motion Controller; Motion Capture; ESP-NOW; Wireless Communication; ROS

I. Introduction

Robotics is leading a technological transformation, rapidly reshaping industries and redefining task execution across various sectors. Among these developments, teleoperated robotic systems have gained particular significance, particularly in applications requiring seamless human-robot collaboration, or in environments where direct human involvement is hazardous, or less effective [1].

A fundamental element of modern teleoperated systems is the integration of wireless communication, enhancing flexibility, mobility, and scalability. Unlike wired setups, wireless solutions enable teleoperation in dynamic and unstructured environments where physical cabling is impractical or limiting. In such tasks demanding high precision and deterministic behavior, like in the medical or chemical sectors, the latency and reliability of the communication channel are critical factors influencing overall system responsiveness. The increasing demand for precision and real-time responsiveness in teleoperation is accelerating the development of advanced telerobotic systems aiming to minimize system latency [2].

Achieving this requires a thorough analysis of not only the employed communication technology, but of the entire control pipeline. Identifying and addressing bottlenecks that affect overall performance is essential for ensuring reliable and responsive operation. A key aspect of this optimization is the design of efficient and low-latency control interfaces that enable seamless and intuitive teleoperation, enhancing both system performance and user experience.

This work presents a novel handheld wireless motion controller for robotic teleoperation that combines optical-active position tracking with low-latency wireless communication. The position tracking is enabled by a Vicon Motion Capture (MoCap) system, providing sub-millimeter accuracy. For reliable low-latency communication, the system employs ESP-NOW, a protocol designed by Espressif for low-energy and connectionless data exchange based on the IEEE 802.11 standard. Unlike conventional use cases where the MoCap systems serves as reference tool for evaluation, it is directly integrated into the control loop. The robotic arm is controlled in it's Cartesian space by mirroring the motion controller's trajectory within the mapped workspace, enabling precise and intuitive teleoperation. As central integration hub for the system the Robot Operating System (ROS) framework is employed, facilitating seamless data acquisition, processing and distribution among the system components. Despite its limitations in real-time data processing [3], ROS provides a flexible and structured environment for managing complex robotic applications.

A comprehensive evaluation of the system is realized by teleoperating a 7-axis Franka Research 3 (Franka Robotics, Munich, Germany) collaborative robot (cobot). This assessment validates the feasibility of the proposed approach and analyzes the impact

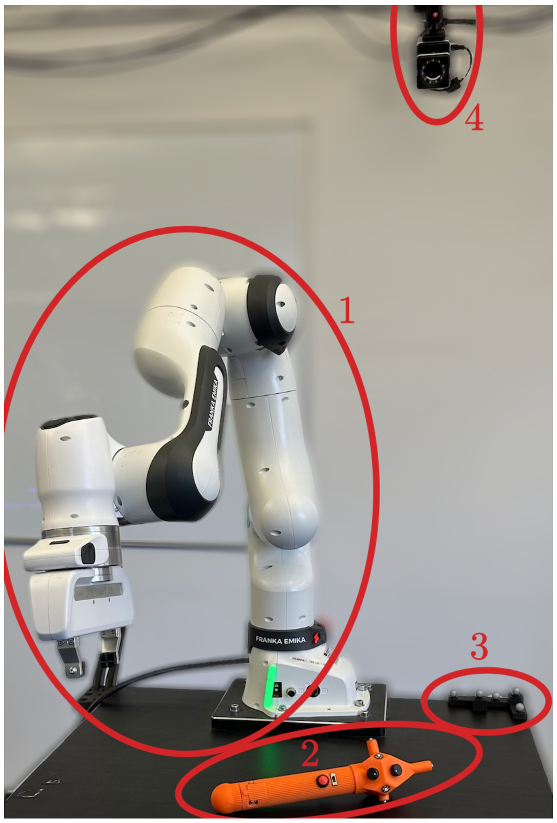


Fig. 1: Overview of the main subsystems. Teleoperation of the Franka Research 3 cobot (1) is facilitated by the wireless motion control device (2). A reference object (3) provides an offset position using passive markers. Position tracking is achieved by a MoCap system with eight cameras (4) capturing the marker objects.

of the different control pipeline stages on the overall system performance. By examining latency sources, system responsiveness, and the accuracy of trajectory mirroring, this evaluation also highlights potential for future optimization in such systems.

To investigate the real-world applicability of a fully wireless teleoperation scenario, evaluations of the system are conducted with a Wi-Fi Direct link established between the MoCap system and the robotic control unit. Building on the previous observations, the performance of this wireless setup is assessed, focusing on its impact on the end-to-end system latency.

The main subsystems of the setup are depicted in Fig. 1: 1) the cobot, 2) the optical-active motion control device, 3) a reference position object and 4) a camera of the infrared-based MoCap system.

The remainder of this work is organized as follows: In Section II, a short introduction to the field of teleoperation using the ESP-NOW protocol and optical motion tracking is given. Section III provides an overview of the system architecture, detailing the integrated components as well as the design and implementation of the motion controller. A detailed analysis of the system, including latency assessment, is described in Section IV, followed by concluding the work in Section V.

II. Related Work

Teleoperation is an interdisciplinary domain, necessitating effective integration of motion tracking and communication technologies to achieve responsive and precise remote robotic control. Reliable, low-latency data exchange between the controlling system and the robot is essential for maintaining synchronization, system stability and overall operational efficiency.

In this context, ESP-NOW offers a promising solution. While its application in robotics is still emerging, the protocol's ability for low-latency and connectionless communication is well-suited for short data transmissions with minimal latency requirements, showcased by *Eridani, Rochim and Cesara* [4]. Although ESP-NOW has a lower default transmission speed compared to Wi-Fi, it can be optimized for higher-frequency applications, making it suitable for robotics. *Grimminger et al.* successfully demonstrated the application of this protocol for direct communication between an external control computer and the Open Dynamic Robot Initiative's robot, 'Solo', with achieving the required 1 kHz update rate. Their study highlights the potential of ESP-NOW for low-latency robotic applications [5]. In comparison, the teleoperation system presented in this work does not require such a high communication frequency, as transmitting the control commands and feedback signals operates at lower frequencies and is partially event-driven, allowing for reliable and efficient communication without pushing the protocol to its maximum throughput limits.

Complementing low-latency and reliable wireless communication in telerobotics, MoCap systems enhance system performance. These systems encompass a range of technologies, each with distinct characteristics and advantages suited for different applications with varying requirements. One notable category within this spectrum is commercial Virtual Reality (VR) systems, such as HTC VIVE with their laser-based Lighthouse tracking technology. This system is capable of capturing controller position and orientation, enabling the acquisition of 6-Degree-of-Freedom (DoF) target pose data. For example, *Whitney et al.* demonstrated an interactive VR robotic teleoperation interface using HTC Vive controllers as part of their ROS Reality package [6]. Presenting a detailed review of contemporary advancements and methodologies in industrial robotics, *Dzedzickis et al.* introduced the Norbo Mimic Kit as interactive teleoperation solution, also relying on the HTC VIVE system [7]. However, Franka cobots are not supported with this interface.

Comparing the accuracy of HTC VIVE to a high-precision Vicon MoCap system, *Merker et al.* conclude that while using Vive Tracker is adequate for robotic visualization in VR environments, its precision is insufficient for detailed mechanical analysis, especially at higher controller velocities [8]. This limitation applies to teleoperation scenarios as well, which is why a high-precision MoCap system is utilized in this work.

Such high-end optical MoCap systems are typically employed as reference measurement tools for evaluating accuracy and reliability of other tracking technologies. For instance, *Zhu et al.* used a Vicon system to validate the performance of their Inertial Measurement Unit (IMU)-based teleoperation solution [9]. However, the direct use of high-accuracy MoCap systems as control interfaces in teleoperated robotics remains relatively uncommon. The study by *Dajles and Siles* illustrates an example by controlling an NAO humanoid robot with translated pose data from human motion captured by an Optitrack system [10]. Similarily, *Lin, Krishnan and Li* used a Vicon system for mapping target pose data derived from an operator's whole-body movements to achieve assistive robotic tasks [11]. Unlike these approaches, the system proposed in this work directly leverages the coordinates of the wireless motion controller device captured by a Vicon MoCap system. These target positions are transmitted to the robotic system without complex motion mapping, allowing for precise control of the cobot's end effector position within its Cartesian space.

To ensure efficient communication, ESP-NOW is integrated, enabling bidirectional exchange of control commands and data between the motion controller and the robotic system. By integrating these components, this approach facilitates responsive, compliant and intuitive teleoperation.

III. System Architecture

The design of an efficient and responsive teleoperation system requires a well-structured architecture. First, a general system overview is presented, outlining the integrated components. Then a detailed description of the design and implementation of the motion control device is provided, followed by a discussion of the ROS-based control software architecture.

A. Architecture Overview

The cobot is mounted on a mobile machinery table, including the external control computer. This cobot station is located within the mapped volume of the MoCap system with a base area of (4×3) m.

In Fig. 2 the overall system architecture is illustrated, subdivided into three main areas and the communication between each other: the optical tracking system, the motion control, and the robotic system. For motion control, a reference object with passive markers is placed on the cobot station, allowing the MoCap system to track the cobot's position. This enables the transformation of the motion controller's coordinates into the cobot's Cartesian space. This approach allows for a dynamic adaption when the cobot station is repositioned or the motion controller is activated from another location within the tracking volume.

The infrared-based tracking is performed using eight high-precision cameras at their maximum supported frame rate of 330 Hz. The coordinates of marker objects are processed on a Windows 11 computer running Vicon Tracker 3.10 and transmitted to the robotic system through the Vicon DataStream Software Development Kit (SDK) v1.12., supporting data transfer via both Ethernet and Wi-Fi Direct.

For real-time control, the motion controller communicates with the robotic system via ESP-NOW, enabling continuous bidirectional exchange of control commands and haptic feedback data. To facilitate this communication, an Intel AX200 Wi-Fi interface is utilized, as it supports monitoring mode and data packet injection. For low-level access the Linux-ESPNOW library is employed, enabling interoperability between ESP-NOW and Linux-based systems. The computer controlling the cobot operates on Debian 12.8 with preemptive real-time kernel v6.1.0.27-rt, while the control software with the ROS processes runs within an Ubuntu 20.04-based Docker container.

In the wireless scenario, all individual sub-systems operate without physical connections, with the MoCap system communicating with the robotic system via Wi-Fi Direct utilizing consumer grade USB Wi-Fi adapters. The cobot control computer features a Qualcomm Atheros AR9271 adapter, configured as an access point, while the MoCap system establishes a connection to it using a D-Link DWA-121 Wireless N 150 adapter.

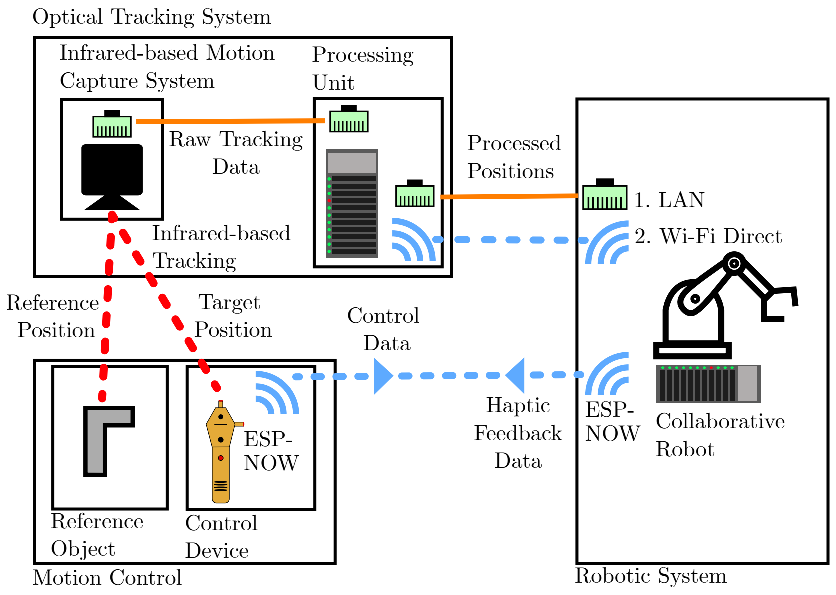


Fig. 2: System Overview: The cobot is operated via control commands from the motion control device via ESP-NOW. Additionally feedback data is transmitted from the robotic system, for instance for haptic feedback. A high-precision optical tracking system captures the target and reference positions, which are then transmitted to the robotic system via Ethernet or Wi-Fi Direct. The target position is derived using optical-active markers on the control device and mapped into the cobot's Cartesian coordinate system.

B. Motion Controller Design

In Fig. 3 a skeleton view of the motion controller's 3D model is illustrated. The housing is 3D printed and has a unique shape, creating a characteristic trackable object for the MoCap system. Eight infrared LEDs (2) with 850 nm centroid wavelength and a beam angle of 120° are used to enable optical-active tracking. Additionally, eight green LEDs provide visual feedback to the operator about the motion controller's current state. Two potentiometers allow parameter adjustments at runtime. One modifies the scaling factor between the control device's movement and the cobot's motion (4), while the other adjusts gripper parameters (6). When the gripper is idle, the grasping force is changed, whereas during a grasping action, the gripper width is modified. Initiating a grasping action is performed via a button in the front of the motion controller (5). The device is powered by a 3000 mAh lithium-ion battery (8).

An ESP32-S3 (1) serves as the processing unit, selected for its built-in low-latency wireless communication capabilities and robust real-time performance. A button on the rear side of the controller (3) initiates the control sequence by transmitting a beacon to the control software via ESP-NOW and concurrently activating the LEDs. Haptic feedback is provided by a rumble motor (7) at the end of the handle, reflecting external force applied to the end effector, provided by the cobot's torque sensors. This enables the operator for immediate reaction to collisions, which is critical in object manipulation tasks to protect the integrity of object and cobot.

As operational environment FreeRTOS is selected due to its efficiency in real-time task management, and deterministic behavior. Its preemptive multitasking capabilities allow precise scheduling, ensuring that time-sensitive operations, such as wireless communication, are prioritized in execution. Thus, FreeRTOS prevents interference from lower-priority tasks, maintaining responsiveness of the system and reliability of data transmission. The control device's software operates with a tick rate of 1 ms, providing the timing resolution needed for responsive task scheduling. To avoid race conditions between FreeRTOS operations and user-defined tasks, binary semaphores are employed for safe and synchronized access to shared resources. The communication between the motion control device and the ROS-based control software is handled via ESP-NOW, which directly interfaces with the Wi-Fi driver. User-defined tasks are assigned significantly lower priorities than the core system tasks, ensuring that time-critical operations such as wireless data handling and interrupt processing are not be preempted by controller tasks. Within the user-defined communication architecture two main components are defined. The first is immediate transmission, which handles time-critical user data, while the second is periodic communication, which is further divided into separate tasks for data reception and transmission at defined intervals.

The immediate transmission handles event-based control data generated by Interrupt Service Routines (ISRs) triggered by button presses. These events are placed into a queue for wireless transmission with only the most recent entry retained to avoid outdated inputs. Since this mechanism controls activation and deactivation of the control pipeline, it is assigned the highest priority among all user tasks. The data reception task holds the second highest priority of user tasks and manages incoming data from the ROS-based control system, such as haptic feedback data. The periodic transmission, operating at 10 Hz, is assigned

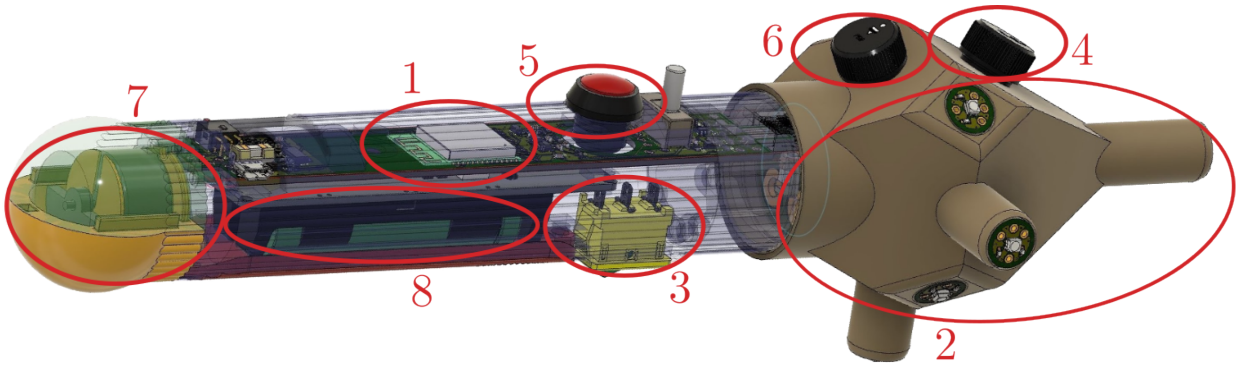


Fig. 3: Model of the motion controller. The device employs an ESP32-S3 (1) using the ESP-NOW protocol for bidirectional communication with the control system. Eight infrared LEDs (2) at the top are used for optical-active tracking, activated by the rear button (3). A potentiometer (4) allows adjustment of the scaling between the controller's movement and the cobot's motion. The gripper is operated by the front button (5), with its parameters dynamically controlled by a separate potentiometer (6). Haptic feedback is provided by a rumble motor (7) at the end of the handle. The controller is powered by a 3000 mAh lithium-ion battery (8).

a slightly lower priority and is responsible for transmitting continuous data to the control software, such as potentiometer readings.

ESP-NOW operates on a best-effort delivery model, meaning it does not guarantee packet delivery or provide built-in mechanisms for retransmission or acknowledgment. Therefore, timeouts are implemented to detect dropped connections or delays in receiving data on the control device. By carefully synchronizing the user-defined tasks and assigning lower priority to computationally more intensive operations, the system maintains robust performance under variable processing loads.

C. Control Software Architecture

To effectively manage the teleoperation tasks, the control software is implemented using the ROS framework. This architecture encompasses multiple nodes exchanging data and interacting with the components of the system, enabling precise and responsive coordination of controlling the cobot and communication with the peripherals. An overview of the architecture is illustrated in Fig. 4.

At the core of the software, the *Control Device Interface* communicates with the motion controller via ESP-NOW. It handles the reception of the beacon for control initiation, processes incoming control commands, and transmits data to the control device such as external end effector force for haptic feedback. The 6-DoF position of the motion controller transmitted from the MoCap system is received by the *Motion Capture Interface*. This position is transformed into the cobot's Cartesian coordinate system utilizing the position of the reference object and forwarded to the *Cobot Controller*. This controller implements a Cartesian impedance control strategy to achieve trajectory mirroring. By adjusting control parameters, the system can be balanced between precision and compliance, influencing both end-to-end latency and motion lag characteristics. The *Gripper Controller* is responsible for managing the gripper, including initiating grasping actions and adjusting its parameters based on control commands from the motion controller.

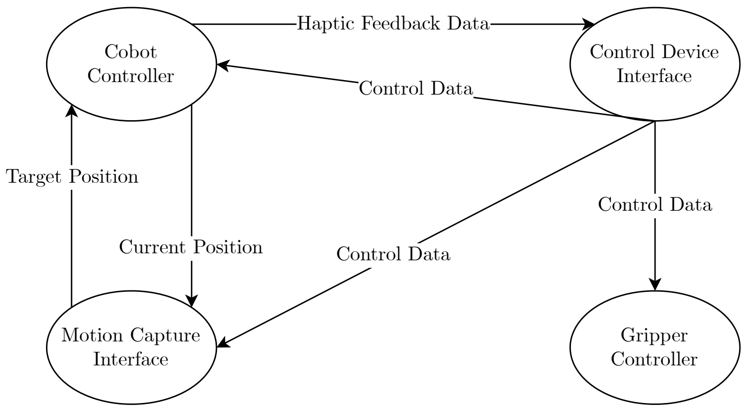


Fig. 4: ROS-based control software architecture. The *Motion Capture Interface* provides the target position to the *Cobot Controller* at 330 Hz with the current end effector position retransmitted to it. The *Control Device Interface* enables bidirectional communication with the motion controller via the ESP-NOW protocol and provides transmitted commands to the software. The *Gripper Controller* manages the end effector.

The haptic feedback data is transmitted from the control software to the control device at a frequency of 100 Hz. Similar to the architecture on the control device, timeouts are implemented within the control software to monitor system connectivity and detect communication failures and transmission delays.

IV. System Evaluation

To assess the performance of the teleoperation system, a comprehensive evaluation is conducted. First, a functional verification is performed to validate its ability to mirror trajectories and highlight the effect of control parameters on trajectory lag and thus end-to-end latency. Next, a detailed analysis examines the latency composition, identifying key contributors in the control pipeline from command initiation to the robot's motion response. Finally, the focus shifts to an evaluation of delays within the MoCap system, specifically examining the impact of Wi-Fi Direct data transmission to the robotic system in order to demonstrate a fully wireless teleoperation scenario.

A. Position Tracking Performance

For teleoperation tasks it is critical that the trajectories of the controlling device and the following robot are closely aligned. To analyze any deviation between these trajectories, the MoCap system is connected to the cobot system via Ethernet, minimizing latency and jitter during transmission. For trajectory recording, the cobots end effector is equipped with passive markers, representing a unique trackable object to the MoCap system.

This analysis examines the alignment of translational motion. To assess its influence on the overall system performance, three different values are employed for the translational stiffness parameter of the Cartesian impedance controller: 200 N/m, 600 N/m, 3000 N/m. As an example, the results for the x-axis are depicted in Fig. 6. It is important to note, that direct comparison of the graphs is not the objective, rather the focus is on a qualitative analysis, specifically evaluating the similarity between the trajectories of *Controller* and *End Effector* in the individual graphs. Within this context, increasing the translational stiffness parameter improves trajectory alignment between the wireless motion controller and the end effector without affecting stability, indicated by an increasing correlation. While the mean positional offset between the target and achieved position at 200 N/m is 32.51 mm with a standard deviation of σ = 24.30 mm, it decreases to 10.24 mm (σ = 6.34 mm) at 3000 N/m. The better synchronized trajectories are also highlighted by increasing Pearson Correlation Coefficients, being r = 0.89 at 200 N/m and r = 0.98 at 3000 N/m. Additionally, the graphs indicate that the trajectory alignment is better at lower trajectory velocities.

The observed lag and velocity dependency are likely due to the cobot's inertia and the chosen parameter settings, rather than latency in the communication links. While tuning of these parameters may improve the systems performance, it would exceed the scope of this work. It is important to note that this lag measurement does not provide an exact empirical analysis of system latency. Instead it serves as indicator of improved system responsiveness associated with adjustments in control parameters.

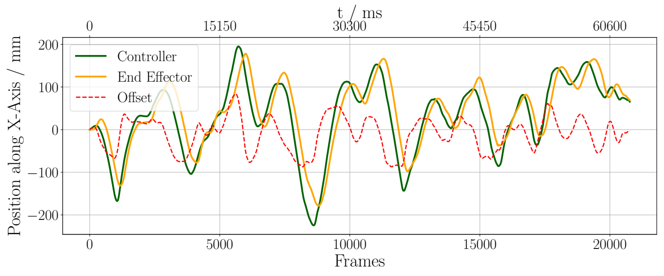
B. Control Pipeline Latency Composition

After the system's ability of accurate position tracking is confirmed, the delay between controller motion and the cobot's response is measured using the end effector's position derived from both its sensor-based estimated pose and the MoCap system's tracking data. To enhance reproducibility, this evaluation is performed at 3000 N/m for translational stiffness with a modified Cartesian impedance controller: upon receiving the ESP-NOW beacon, the end effector moves 0.1 mm along the z-axis in Cartesian space from a fixed starting position. This ensures comparable results by achieving maximum angular velocity at highest joint torque rate while maintaining the repeatability and accuracy of both the cobot and the MoCap system.

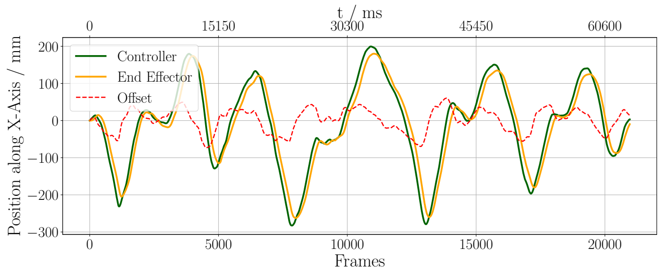
Timestamps are recorded within the *Motion Capture Interface* ROS-based control software to ensure synchronization and enable accurate latency measurements at different stages of the control pipeline. This allows for precise analysis of system performance and targeted optimizations. The first timestamp is logged when the control software receives the ESP-NOW beacon upon activation of the wireless motion controller. The second is captured when the MoCap system detects the controller object, representing its initial position. The third timestamp is recorded when the cobot's end effector moved by the specified distance of 0.1 mm, and the fourth is logged when the MoCap system detects the same displacement. To minimize the risk of measurement distortion, the timestamps are saved to a file upon ROS termination. The measurement is repeated 185 times with Ethernet connecting the MoCap system and the cobot controlling computer.

Additionally, the system is evaluated using a high-speed camera capturing images at 1000 Hz. By visually inspecting the frames, the delay between LED activation (concurrently with transmitting the beacon) and the end effector moving the previous specified distance from its stationary position can be analyzed. Due to the manual nature of this method, the measurement was conducted only ten times. Although visual analysis introduces a higher margin of error, it provides as a valuable reference for cross-verification of the latency results.

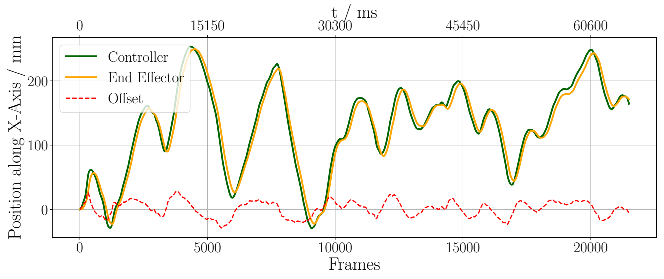
As depicted in Fig. 7, the ESP-NOW beacon for control initiation is received by the control system consistently in advance of the detection of the control device object by the MoCap system. This systematic lead time suggests that ESP-NOW operates with a consistently low transmission latency and is not primarily affecting the end-to-end delay of the demonstrated architecture.



(a) Translational Stiffness = 200 N/m, Mean Offset = 32.51 mm, r = 0.89



(b) Translational Stiffness = 600 N/m, Mean Offset = 26.36 mm, r = 0.96



(c) Translational Stiffness = 3000 N/m, Mean Offset = 10.24 mm, r = 0.98

Fig. 6: Comparison of translational trajectories of the motion controller and end effector along the x-axis of the cobot's Cartesian space for varying values of translational stiffness. Both trajectories are recorded with the MoCap system at a frame rate of 330 Hz. A qualitative analysis of the individual graphs reveals that trajectory alignment improves with increasing stiffness value.

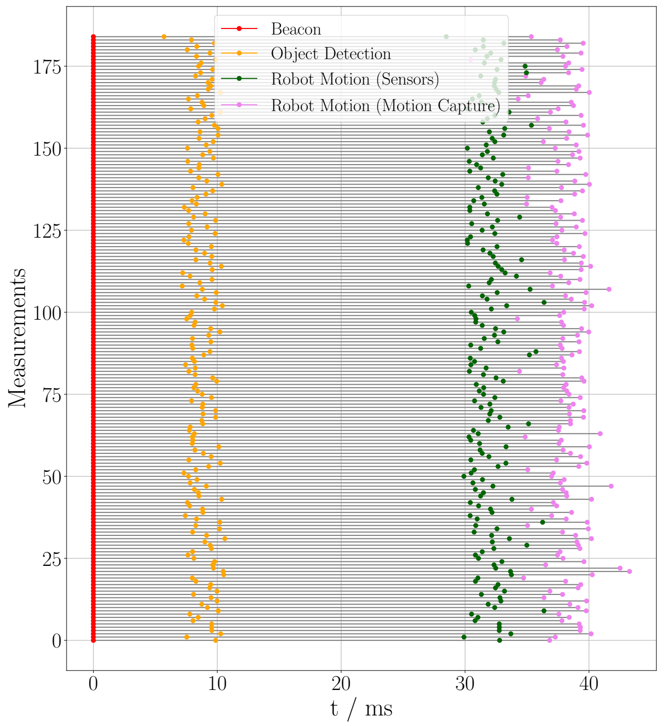


Fig. 7: Timestamps for the different stages in the control pipeline, recorded in the *Motion Capture Interface* of the control software. The graph is normalized to ESP-NOW beacon reception to visualize the system's latency composition.

Thus, the delays are normalized to this timestamp to facilitate the analysis of the latencies exhibited by the various system components.

The mean values and standard deviations for these latencies are summarized in Tab. I. Between receiving the ESP-NOW beacon and the initial position of the control device object captured by the MoCap system, there is a delay of 8.79 ms with a standard deviation of 0.91 ms. Since the beacon and the activation of the infrared LEDs occur concurrently, this indicates that the MoCap system requires some time to detect, process, and transmit the position of a marker object. The delay between a detected motion command—signalized by the detection of the control device object by the MoCap system—and the cobot completing the 0.1 mm movement along the z-axis in Cartesian space is 23.21 ms ($\sigma = 0.96$ ms). This delay may be influenced by the controller parameter settings, the cobot's physical constraints, and the need to counteract high initial inertia as the cobot moves out of its stationary position.

Consistent with previous observations, the MoCap system requires time to detect the positional changes. The interval between a detected motion command and completion of the respective movement is 29.4 ms with standard deviation of 1.33 ms. This results in a delay between an actual movement and its detection by the MoCap system with an average duration of 6.19 ms and a standard deviation of 1.55 ms. This highlights the inherent operational latency within the MoCap system. A primary factor to this delay is likely the frame rate of 330 Hz, which imposes a minimum processing delay of 3.03 ms to the overall system. In combination with the previously noted difference in recognized motion stops, this suggests that the MoCap system operates on basis of a frame-to-frame comparison rather than relying on absolute position estimation. When comparing the standard deviations of the measurement, the MoCap system-based latencies demonstrate greater temporal variability compared to those obtained from joint sensor messages. This is likely the result of frame rate limitations, which introduce an inherent measurement error, and the MoCap systems processing overhead.

In comparison, the high-speed camera measurement captures the latency of the entire pipeline from ESP-NOW beacon transmission to the start of cobot motion. By visually analyzing frame by frame, the average duration between LED activation and the onset of end effector movement is determined to be 36.1 ms (σ = 0.74 ms). However, due to the nature of the visual frame-by-frame analysis, this method inherently introduces a greater degree of uncertainty compared to automated timestamp-based measurements, as it is subject to frame rate limitations, human interpretation errors, and potential inconsistencies in frame alignment. Although the accuracy of the timestamp measurements cannot be replicated in this experiment, the results still indicate that ESP-NOW communication contributes only minimally to overall system delay compared to the other components. Additionally, this measurement provides an upper bound for the systems end-to-end latency, though further automated or high-precision methods would be required to fully quantify the impact of procedural uncertainties on the reported values.

C. *Wireless Motion Capture Latency*

To assess the system performance in a wireless teleoperation scenario, measurements are conducted with the MoCap system connected to the cobot control computer via Wi-Fi Direct, and the results are compared to those obtained using an Ethernet connection. This evaluation is performed by analyzing the timing characteristics of the MoCap system using the Vicon DataStream SDK, which is capable to measure the time required for the various stages of the MoCap pipeline. The time required for communication is estimated based on TCP transmission analysis. To quantify the MoCap system's operational delay, 80,000 frames are recorded, with the results summarized in Tab. II.

The transmission time from the cameras to the processing stage averages to 3.078 ms (σ = 0.071 ms), which is slightly longer than the camera sampling time, but remains acceptable due to the transmission to the capturing software. The data processing stage, which includes image synchronization, extraction of the objects' coordinates, and delivering this data to the stream, accounts to 0.778 ms with a standard deviation of 0.046 ms. Consequently, the minimum operational delay of the MoCap system is 3.856 ms, regardless the communication technology used for interfacing with other systems. Furthermore, the low standard deviations observed across these latency components suggest a highly stable and consistent performance of the MoCap system. Comparing the total duration from received image data to the transmission of processed data—3.999 ms for Ethernet transmission—with the 6.19 ms reported in the previous analysis (see Tab. I), reveals a noticeable discrepancy. This difference is likely attributable to the computational overhead within the ROS framework. Additionally, the measurement error introduced by the minimal operational time of the MoCap system is to be considered in the evaluation of end-to-end latency in the previous section.

The transmission of the processed data to the robotic system occurs within 0.143 ms (σ = 0.001 ms) when using Ethernet, whereas communication via Wi-Fi Direct exhibits a mean duration of 1.208 ms and a standard deviation of σ = 0.344 ms. Building on the previous evaluations, it can be concluded that the latency introduced by the transmission of positional data has only minimal impact on the overall system latency, even when using Wi-Fi Direct. Specifically, this delay accounts for only 3.82 % of the end-to-end latency for the wireless communication between the two computers, when considering the high-speed camera analysis. However, the increase in transmission time and jitter when using Wi-Fi Direct, may still be a relevant factor in applications requiring ultra-reliable low-latency responses in wireless communication.

Tab. I: Mean latencies of different stages in the control pipeline, measured by time stamping within the ROS-based control software. Additionally, the end-to-end latency of the system is visually assessed using a high-speed camera.

Description	**Mean / ms**	**σ / ms**
Beacon $\rightarrow$ Object Detection	8.79	0.91
Object Detection $\rightarrow$ Robot Motion (Sensors)	23.21	0.96
Object Detection $\rightarrow$ Robot Motion (MoCap)	29.40	1.33
Beacon $\rightarrow$ Robot Motion (Sensors)	32.00	1.35
Beacon $\rightarrow$ Robot Motion (MoCap)	38.19	1.61
High-Speed Camera End-to-End Latency	36.10	0.74

Tab. II: Mean delays of different stages in the MoCap system, measured using the Vicon DataStream SDK. The durations for data transmission are estimated based on TCP communication analysis.

Description	**Mean / ms**	**σ / ms**
Image Data Transmitted	3.078	0.071
Data Processing	0.778	0.046
Transmission via Ethernet	0.143	0.001
Transmission via Wi-Fi Direct	1.208	0.344
Total Duration Ethernet	3.999	0.082
Total Duration Wi-Fi direct	5.064	0.353

V. Conclusion and Outlook

This work presents the design and implementation of a custom wireless motion controller for robotic teleoperation based on low-latency wireless communication and optical-active motion tracking. A high-precision Vicon Motion Capture (MoCap) system tracks the controller's position, while the ESP-NOW communication protocol enables transmission of control commands to the robotic system and reception of data for haptic feedback. For fully wireless teleoperation, the MoCap system communicates via Wi-Fi Direct with the robot control unit. The system is seamlessly integrated with a Franka Research 3 collaborative robot (cobot) within the Robot Operating System (ROS) framework utilizing a Cartesian impedance control strategy achieving accurate teleoperation, indicated by strong trajectory correlation.

A comprehensive system evaluation demonstrates that the end-to-end latency in the control pipeline is primarily affected by physical constraints of the cobot's actuation mechanism and the processing time of the MoCap system, posing inherent constraints to the overall performance. Adjusting the translational stiffness parameter in the Cartesian impedance controller enhances trajectory alignment and reduces lag, suggesting that further parameter tuning potentially enhances system performance.

In contrast to these fundamental constraints, the transmission of positional data for trajectory mirroring via Wi-Fi Direct introduces only a minor additional delay. While its duration is longer compared to Ethernet, the difference remains negligible in the context of the total system latency. This reinforces the viability of wireless communication for teleoperation without substantial performance trade-offs. However, while the transmission delay itself is minimal and the two subsystems have a close spatial proximity, real-world applications of wireless teleoperation systems must account for factors such as network stability, interference, and potential packet loss. These challenges become particularly relevant in dynamic environments and when operating over extended distances, where maintaining a reliable and low-latency connection is critical for ensuring seamless system performance.

Acknowledgment

This work has been supported by the Federal Ministry of Education and Research of the Federal Republic of Germany (Förderkennzeichen 16KISK003K, Open6GHub). The authors alone are responsible for the content of the paper.

References

[1] H. Liu and L. Wang, "Remote human–robot collaboration: A cyber–physical system application for hazard manufacturing environment," *Journal of Manufacturing Systems*, vol. 54, pp. 24–34, 2020. DOI: doi.org/10.1016/j.jmsy.2019.11.001.

[2] V. M. Oliveira, P. Morais, B. Oliveira, J. L. Vilaca, and A. H. J. Moreira, "Exploring current communication frameworks for medical teleoperation," in *2021 IEEE 9th International Conference on Serious Games and Applications for Health(SeGAH)*, Dubai, United Arab Emirates: IEEE, Aug. 4, 2021, pp. 1–6. DOI: 10.1109/SEGAH52098.2021.9551888.

[3] J. Petershans, J. Herbst, M. Rüb, E. Mittag, and H. D. Schotten, "Robotic Teleoperation: A Real-World Test Environment for 6G Communications," in *28. ITG-Fachtagung Mobilkommunikation*, VDE-Verlag, May 2024, pp. 106–111.

[4] D. Eridani, A. F. Rochim, and F. N. Cesara, "Comparative Performance Study of ESP-NOW, Wi-Fi, Bluetooth Protocols based on Range, Transmission Speed, Latency, Energy Usage and Barrier Resistance," in *2021 International Seminar on Application for Technology of Information and Communication (iSemantic)*, Semarangin, Indonesia: IEEE, Sep. 18, 2021, pp. 322–328. DOI: 10.1109/iSemantic52711.2021.9573246.

[5] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wuthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Sprowitz, and L. Righetti, "An Open Torque-Controlled Modular Robot Architecture for Legged Locomotion Research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, Apr. 2020. DOI: 10.1109/LRA.2020.2976639.

[6] D. Whitney, E. Rosen, D. Ullman, E. Phillips, and S. Tellex, "ROS Reality: A Virtual Reality Framework Using Consumer-Grade Hardware for ROS-Enabled Robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid: IEEE, Oct. 2018, pp. 1–9. DOI: 10.1109/IROS.2018.8593513.

[7] A. Dzedzickis, J. Subačiūtė-Žemaitienė, E. Šutinys, U. Samukaitė-Bubnienė, and V. Bučinskas, "Advanced Applications of Industrial Robotics: New Trends and Possibilities," *Applied Sciences*, vol. 12, no. 1, p. 135, Dec. 23, 2021. DOI: 10.3390/app12010135.

[8] S. Merker, S. Pastel, D. Bürger, A. Schwadtke, and K. Witte, "Measurement Accuracy of the HTC VIVE Tracker 3.0 Compared to Vicon System for Generating Valid Positional Feedback in Virtual Reality," *Sensors*, vol. 23, no. 17, p. 7371, Aug. 24, 2023. DOI: 10.3390/s23177371.

[9] H. Zhu, X. Li, L. Wang, Z. Chen, Y. Shi, S. Zheng, and M. Li, "IMU Motion Capture Method with Adaptive Tremor Attenuation in Teleoperation Robot System," *Sensors*, vol. 22, no. 9, p. 3353, Apr. 27, 2022. DOI: 10.3390/s22093353.

[10] L. N. M., D. Dajles, and F. Siles, "Teleoperation of a Humanoid Robot Using an Optical Motion Capture System," in *2018 IEEE International Work Conference on Bioinspired Intelligence (IWOBI)*, San Carlos: IEEE, Jul. 2018, pp. 1–8. DOI: 10.1109/IWOBI.2018.8464136.

[11] T.-C. Lin, A. U. Krishnan, and Z. Li, "Physical Fatigue Analysis of Assistive Robot Teleoperation via Whole-body Motion Mapping," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China: IEEE, Nov. 2019, pp. 2240–2245. DOI: 10.1109/IROS40897.2019.8968544.