engineering proceedings

MDPI

Proceeding Paper

Design and Implementation of a Cloud-Based Digital Twin for Real-Time Monitoring and Bidirectional Control of a Robotic Arm †

Minh Hang Phan Vu, Nam Khanh Vu, Thanh Loan Nguyen Thi, Tri Dung Dang * and Tri Cuong Do

Institute of Intelligent and Interactive Technologies, University of Economics Ho Chi Minh City (UEH), Ho Chi Minh City 700000, Vietnam; hangphan.31241021506@st.ueh.edu.vn (M.H.P.V.); khanhvu.31241020368@st.ueh.edu.vn (N.K.V.); loannguyen.31241020850@st.ueh.edu.vn (T.L.N.T.); cuongdt298@ueh.edu.vn (T.C.D.)

* Correspondence: dungdt@ueh.edu.vn

† Presented at the 2026 International Conference on Energy, Infrastructure and Environmental Research (EIER 2026), Ho Chi Minh City, Vietnam, 15–17 July 2026.

Abstract

Six-degree-of-freedom robotic arms are widely used in industrial automation but face motion inaccuracies caused by model uncertainties, environmental disturbances, and payload variations. Existing Digital Twin solutions often rely on resource-intensive platforms such as ROS/Gazebo, limiting their accessibility for broader deployment. This paper proposes a lightweight Digital Twin system for a 6-DOF robotic manipulator using Unity 3D, integrated with a physical ESP32-controlled robot through Firebase Realtime Database as a cloud-based synchronization middleware. The system supports bidirectional communication across two operating modes with conflict-free handover via a manual toggle mechanism. Forward and inverse kinematics are derived using the Denavit–Hartenberg convention, and exponential smoothing is applied on the ESP32 to ensure gradual and safe servo actuation. Experimental results demonstrate a mean Firebase communication latency of 91.7 ms with a standard deviation of 25.5 ms over 215 measurement cycles, confirming near-real-time responsiveness. Synchronization accuracy experiments yield a mean absolute error of 6.88° for Joint 2 and 5.20° for Joint 3 during dynamic motion, with zero steady-state error when joint positions are held constant. The proposed framework offers an accessible and scalable alternative to existing Digital Twin platforms, requiring no specialized operating system expertise or high-end hardware.

Keywords: 6-DOF robotic arm; digital twin; ESP32; Firebase; real-time synchronization; Unity 3D

Check for updates

Academic Editor: Nguyen-Vu Truong

Published: 12 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

In the context of Industry 4.0, 6-degree-of-freedom (6-DOF) robotic manipulators are essential for complex, high-precision tasks [1]. However, accurate control in dynamic environments remains challenging due to modeling errors, external disturbances, and payload variations [2,3]. Digital Twin (DT) technology addresses these issues by establishing a real-time virtual replica for continuous monitoring and control [4]. Despite its potential, the widespread deployment of DT systems is often hindered by a reliance on resource-intensive platforms (e.g., MATLAB/Simscape) or complex, tightly coupled local infrastructures (e.g., ROS/Gazebo) that demand advanced hardware and Linux expertise [5–9]. Furthermore, existing lightweight DT architectures frequently lack explicitly designed conflict-free

 https://doi.org/10.3390/engproc2026164009

mode-switching mechanisms for bidirectional control, and systematic empirical evaluations of latency in cloud-integrated setups are scarce. To bridge these gaps, this paper proposes a lightweight, cloud-based DT framework for a 6-DOF manipulator using Unity 3D and an ESP32 microcontroller, synchronized via the Firebase Realtime Database. This approach provides an accessible, hardware-efficient alternative to traditional simulation tools while ensuring reliable real-time bidirectional communication. The main contributions of this study are threefold: (1) designing a unified, lightweight physical-cloud-virtual DT architecture that utilizes an event-driven Firebase Stream to eliminate continuous polling overhead; (2) implementing a conflict-free dual-mode control strategy with a manual toggle to prevent unintended command interference under network latency variability; and (3) conducting comprehensive empirical evaluations—including latency analysis over 215 cycles and trajectory-based tracking accuracy.

2. Related Work

Recent advancements have increasingly integrated Digital Twin technology into robotic control to meet Industry 4.0 demands. Existing studies have demonstrated the effectiveness of Digital Twins in kinematic simulation [10], reinforcement learning-based control [11], and goal-oriented communication to optimize data transmission [12]. Despite these successes, practical deployment often remains constrained. Most state-of-the-art solutions heavily rely on resource-intensive software like MATLAB/Simscape [10] or complex platforms such as ROS/Gazebo, requiring high-end hardware and specialized Linux expertise. Furthermore, many current architectures primarily focus on one-way state monitoring or lack comprehensive empirical evaluations of latency during bidirectional control. To address these gaps, this paper proposes a lightweight, easy-to-deploy Digital Twin framework for 6-DOF manipulators that achieves real-time bidirectional synchronization without demanding complex local infrastructures.

3. Materials and Methods

3.1. Robot Model

3.1.1. Forward Kinematics

To synchronize the motion between the real robotic arm and the virtual model in Unity, it is necessary to build an accurate kinematic model to determine the position and orientation of the end-effector from the measured joint angles. In this study, the Denavit–Hartenberg (DH) convention is used to systematically and consistently model both the forward and inverse kinematics of the robotic arm. The robot's kinematic chain is modeled using the standard Denavit–Hartenberg convention. The DH parameters for each link are presented in Table 1. The 6-DOF robotic arm model is illustrated in Figure 1.

Table 1. The modified D-H parameters.

i	a_i	α_i	d_i	θ_i
1	l_1	0	0	0
2	l_2	$\frac{\pi}{2}$	$-l_3$	$\frac{\pi}{2}+\theta_1$
3	0	0	l_4	θ_2
4	0	0	l_5	θ_3
5	l_7	$\frac{\pi}{2}$	l_6	$\theta_4=-\theta_2-\theta_3$
6	l_9	0	l_8	$\theta_5=-\theta_1$

https://doi.org/10.3390/engproc2026164009

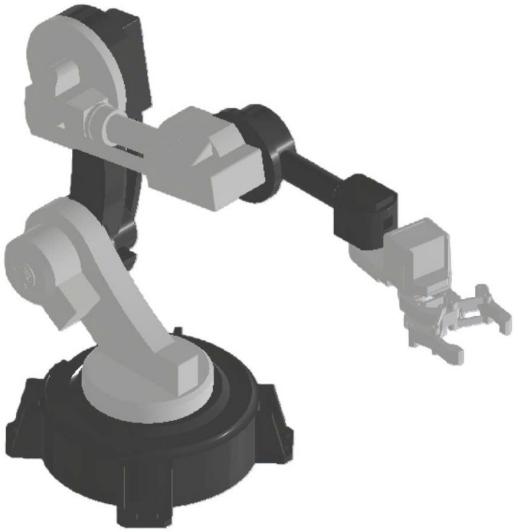


Figure 1. The 6-DOF robotic arm model used for forward kinematics analysis.

Based on the Denavit–Hartenberg (DH) parameters given in Table 1, the forward kinematics of the 6-degree-of-freedom robotic manipulator are established using the standard DH convention.

By explicitly expanding the DH-based transformation matrices, both the orientation $R_6^0 \in \mathbb{R}^{3\times3}$ and the position vector ${}^0P_{EE}$ of the end-effector with respect to the base frame are obtained, where the position is given by:

$$ {}^0P_{EE} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}. \tag{1} $$

3.1.2. Inverse Kinematic

The inverse kinematics problem aims to determine the joint variables $\theta_1, \theta_2, \theta_3$ corresponding to a desired end-effector position $P = [P_x, P_y]^T$. The analytical solution is obtained as follows:

$$ \theta_1 = \text{atan2}(P_y - l_8,\ P_x) \tag{2} $$

$$ \theta_3 = \text{atan2}\left(\pm\sqrt{1 - \cos^2\theta_3},\ \cos\theta_3 \right) \tag{3} $$

$$ \theta_2 = \text{atan2}(h, d) - \text{atan2}(l_5 \sin\theta_3,\ l_4 + l_5 \cos\theta_3) \tag{4} $$

3.2. Unity-Based Digital Twin Framework

3.2.1. Digital Twin System Architecture

The proposed Digital Twin system adopts a three-layer architecture, consisting of a physical layer (real robot and ESP32), a cloud-based communication layer (Firebase Realtime Database) and a virtual simulation layer (Unity 3D). The architecture of the proposed Digital Twin system is illustrated in Figure 2.

At the physical layer, a 6-degree-of-freedom robotic manipulator is controlled by an ESP32 microcontroller, with actuators mounted at each joint. The ESP32 performs local control based on two command sources: control commands transmitted from Unity via Firebase, or commands entered directly through the Serial Monitor. The ESP32 continuously executes motion commands from either source and periodically reports joint states to Firebase regardless of the active command source.

https://doi.org/10.3390/engproc2026164009

The Firebase Realtime Database functions as a bidirectional communication layer between the physical robot and Unity 3D. The database is structured into two primary nodes: /robot/command (used by Unity to transmit target angles to the ESP32) and /robot/feedback (enabling the ESP32 to send current joint states back to Unity). Both nodes contain identical sub-fields for the five joint angles (θ_1 to θ_5), the gripper state, the active control mode, and a timestamp, ensuring seamless and symmetric data mapping.

The real-world implementation of the Digital Twin system is shown in Figure 3.

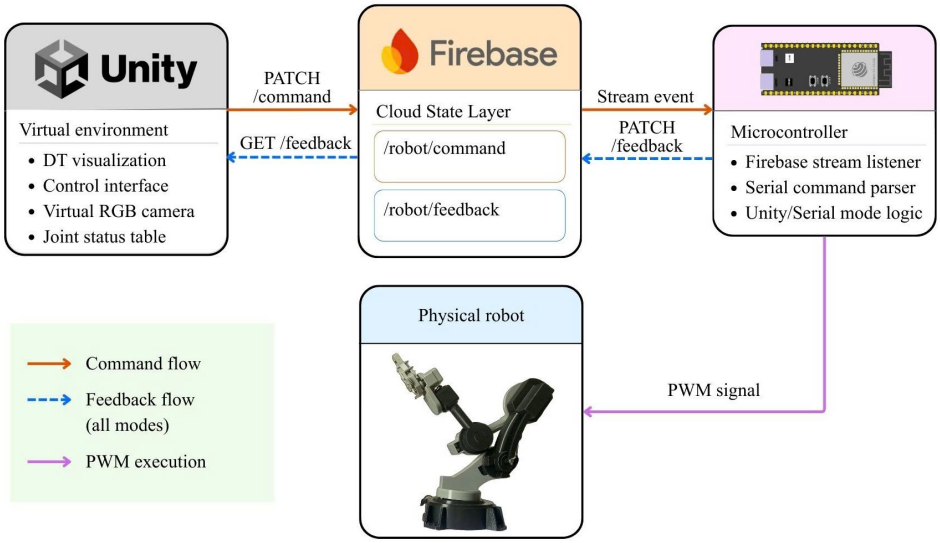


Figure 2. Architecture of the proposed Digital Twin system.

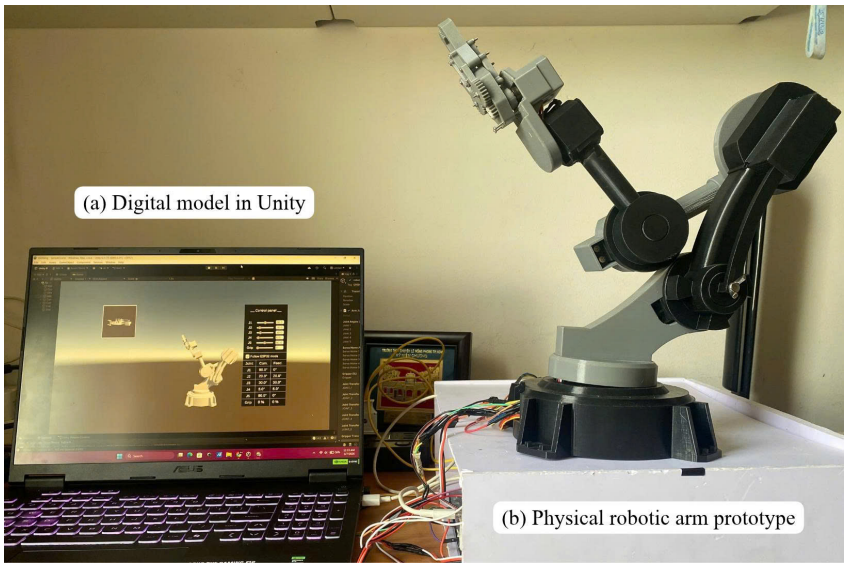


Figure 3. Experimental setup of the Digital Twin system: (**a**) Unity-based digital model and (**b**) physical robotic arm prototype.

Unity 3D is employed to develop the Digital Twin of the robotic manipulator, enabling motion visualization and real-time monitoring of the system's operational state. The system supports bidirectional control: Unity can transmit control commands to the physical robot, while simultaneously updating the virtual model based on the robot's actual joint states when the robot operates independently.

3.2.2. Unity-Based 3D Robot Modeling and Visualization

The 3D model of the 6-degree-of-freedom robotic manipulator is developed in Unity 3D, with a geometric structure corresponding to the physical system. Each robot joint is modeled as an independent object and organized using a hierarchical parent–child structure, enabling accurate representation of the kinematic relationships among the joints.

The joint angle values retrieved from Firebase are directly mapped to the rotational transformation of each joint in Unity according to the following formulation:

$$\theta_i^{Unity} = j_i^{Init} + (joint_i - servoHome_i) \tag{5}$$

where j_i^{Init} denotes the initial rotation of the joint when the model is in the home configuration, $joint_i$ represents the current absolute joint angle, and $servoHome_i$ is the reference home angle of the corresponding servo motor.

This mapping strategy ensures that the virtual model accurately reflects the motion of the physical robot at the command level, regardless of the initial pose of the 3D mesh.

The control interface is implemented directly using the Unity UI system, consisting of joint angle sliders (0°–180°) and a real-time status panel that compares the commanded values (*Command*) with the feedback values received from the physical robot (*Feedback*). Additionally, to enhance remote monitoring capabilities, the visualization system incorporates supplementary virtual cameras (e.g., a top-down perspective) and Unity's Render Texture feature. This setup provides operators with comprehensive, multi-angle visual perception of the robot's real-time operation directly within the dashboard. Figure 4 illustrates the Unity interface.

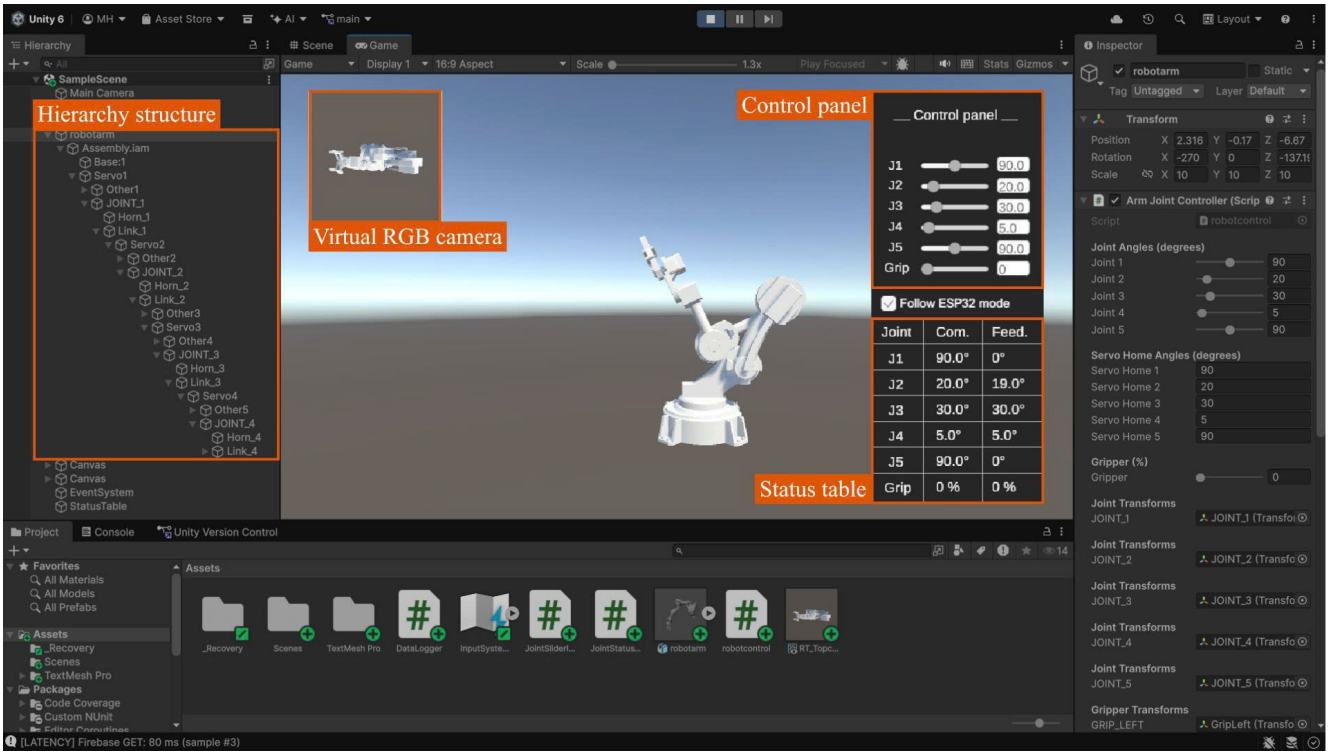


Figure 4. Unity interface.

3.3. Real-Time State Synchronization Between Physical and Virtual Robots

The Digital Twin system employs a real-time bidirectional synchronization mechanism governed by a manual 'Follow Mode' toggle to prevent command conflicts. The synchronization logic strictly follows two mutually exclusive paths. (1) In Unity Control Mode (Follow Mode Disabled), Unity acts as the master controller. Joint angles adjusted via UI

sliders are transmitted to the /robot/command Firebase node via HTTP PATCH, while the ESP32 continuously publishes its actual joint states to /robot/feedback every 200 ms. (2) In Serial/Follow Mode (Follow Mode Enabled), the physical robot is controlled locally via Serial input, and Unity acts as a state-follower by continuously fetching the 200 ms feedback data to update the virtual model. To prevent an infinite feedback loop during this mode, a boolean flag (updatingSliders) is utilized. When feedback data arrives, this flag is set to true to bypass the MarkDirty() invocation in the SetJoint() functions, ensuring incoming telemetry only updates the visualization without re-transmitting redundant control commands.

Control Strategy

The control strategy of the proposed system is structured as a distributed architecture based on an event-driven control paradigm, tightly integrating the virtual simulation environment with the physical mechanical system. Unity serves as the high-level command generation module, while the ESP32 operates as the real-time execution layer, directly actuating the servo motors. The Firebase Realtime Database functions as a synchronization middleware, ensuring consistent command transmission between the virtual and physical domains.

Instead of adopting a continuous polling approach, the ESP32 establishes a real-time Firebase Stream connection to the /robot/command branch and processes data only upon the occurrence of update events from Unity. When Unity writes data to Firebase, a stream event is triggered, allowing the ESP32 to receive the updated command immediately without issuing repetitive requests. This event-driven mechanism reduces unnecessary communication overhead and enhances the responsiveness of the overall system.

Synchronization is performed at the command level through a structured JSON data format. Each command packet generated by Unity contains the target joint angles for all actuated joints. The system operates at a high abstraction level: rather than transmitting low-level control signals such as PWM or torque commands, only target joint position values are exchanged. This design significantly simplifies the overall control architecture while maintaining effective coordination between the Digital Twin and the physical robot.

At the physical execution layer, a motion smoothing mechanism based on exponential filtering is implemented. Instead of instantaneously jumping to the target angle, the current position of each joint is updated every 20 ms according to the following relation:

$$cur_i \leftarrow cur_i + \alpha(target_i - cur_i) \tag{6}$$

where $\alpha = 0.15$ is the smoothing coefficient.

This approach produces continuous and gradual motion, mitigates abrupt torque variations, and reduces mechanical stress on the robot structure. Since no external encoders are employed to verify the actual joint positions, the overall control strategy at the system level is classified as open-loop control.

In the event of connection loss or absence of new commands, the robot maintains its last stable state because the target values remain unchanged. This fallback behavior enhances mechanical safety and prevents unintended motion under unstable network conditions. The flowchart of this process is shown in Figure 5.

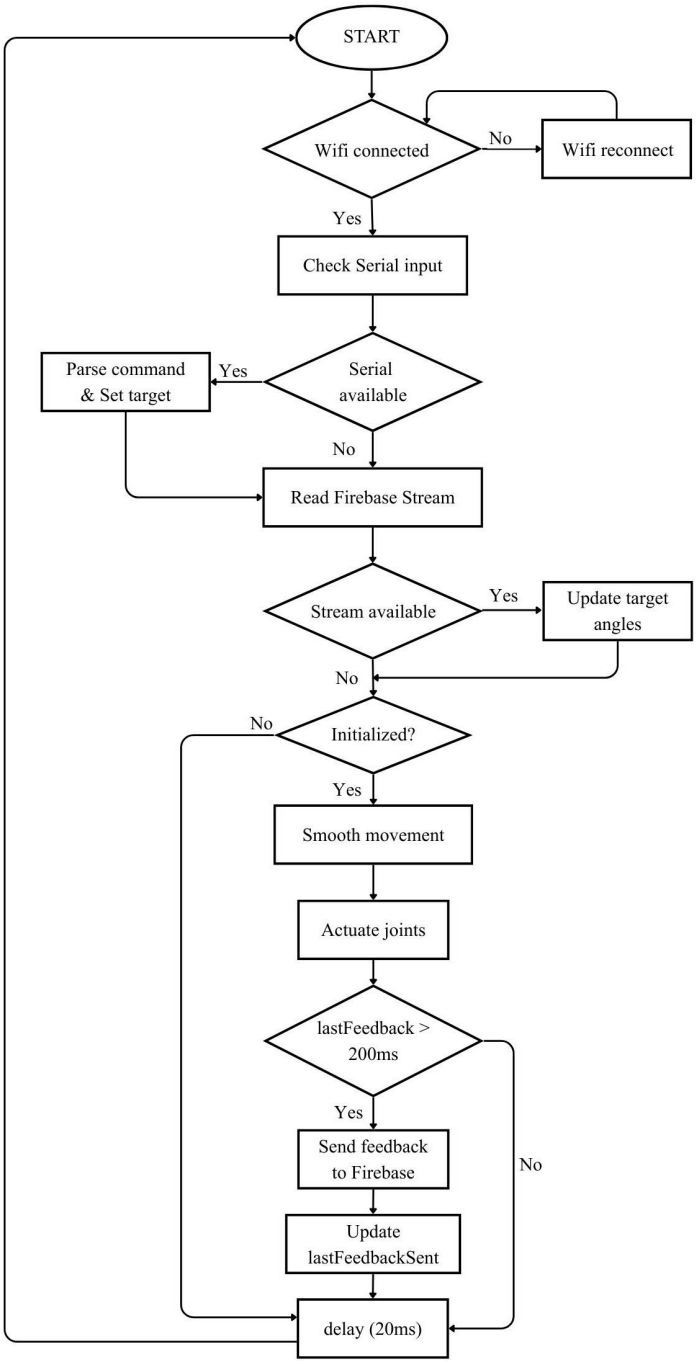


Figure 5. Flowchart of the ESP32 event-driven control loop for bidirectional synchronization between the physical robot and the Digital Twin.

4. Results and Discussion

4.1. Communication Latency Analysis

To evaluate the communication performance of the Firebase-based middleware layer, a latency measurement experiment was conducted over 215 command-response cycles. For each cycle, the time elapsed from the moment Unity initiated an HTTP GET request to the Firebase Realtime Database until a complete response was received was recorded. This metric reflects the effective communication delay experienced by the Digital Twin when polling the physical robot's state.

The results are presented in Table 2 and illustrated in Figure 6.

Table 2. Communication latency statistics ($n = 215$).

Metric	Value
Mean latency	91.7 ms
Minimum	46.5 ms
Maximum	296.9 ms
Std. deviation	25.5 ms

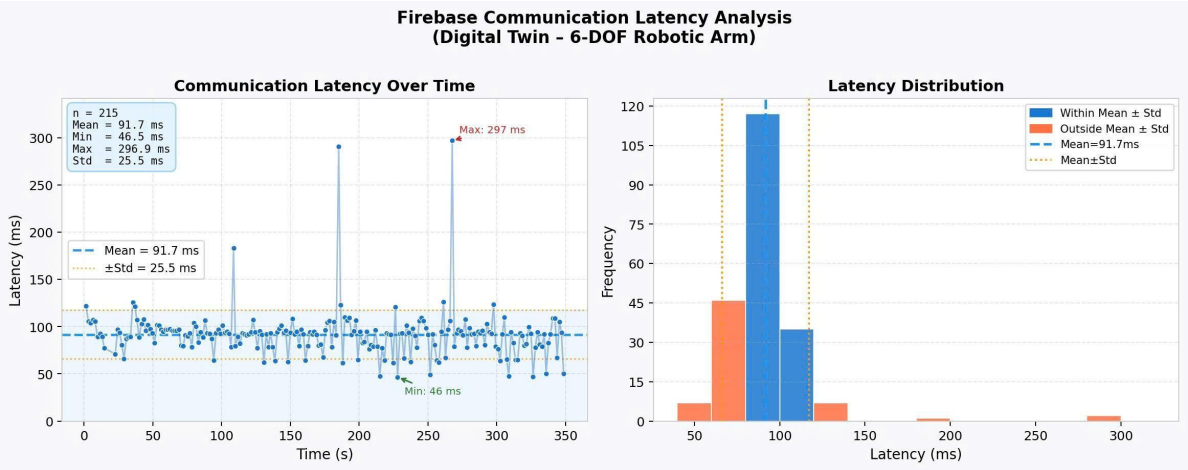


Figure 6. Firebase communication latency analysis.

The mean latency of 91.7 ms indicates that the system is capable of near-real-time state synchronization, which is acceptable for motion monitoring applications that do not require hard real-time guarantees. The minimum recorded latency of 46.5 ms demonstrates that under favorable network conditions, the communication pipeline introduces minimal delay. The maximum latency of 296.9 ms represents an outlier likely caused by transient network congestion or Firebase server scheduling variability. The standard deviation of 25.5 ms reflects moderate variability inherent to cloud-based communication, consistent with prior studies on IoT-based Digital Twin frameworks [11,12].

As shown in Figure 6, approximately 70% of all latency samples fell within the range of 46–120 ms, indicating that the system maintains consistently low latency during normal operation. The latency distribution is right-skewed, with the majority of measurements clustering near the mean and a small number of outliers toward the upper bound. Notably, the larger sample size of 215 cycles yields a lower standard deviation compared to preliminary measurements, confirming the stability of the Firebase communication channel under sustained operation.

While the observed mean latency of 91.7 ms is sufficient for supervisory monitoring and interactive control, it may not satisfy strict hard real-time requirements typically demanded by high-speed closed-loop robotic control systems. Prior studies on IoT communication protocols for Digital Twin applications have reported that cloud-based HTTP architectures inherently introduce higher latency compared to local broker configurations due to server-side routing and network variability [13], representing a known trade-off that the proposed system accepts in exchange for simplified deployment, elimination of local server configuration, and global accessibility. Therefore, the proposed framework is particularly suitable for remote monitoring, educational platforms, and cloud-integrated robotic experimentation, rather than time-critical industrial motion control.

https://doi.org/10.3390/engproc2026164009

4.2. Synchronization Accuracy Evaluation

To assess the synchronization accuracy between the Unity Digital Twin and the physical robot, a trajectory tracking experiment was conducted in Unity Control Mode. The operator adjusted the joint angle sliders for Joint 2 and Joint 3 continuously within the range of their physical limits, while the ESP32 simultaneously reported the actual achieved joint positions back to Firebase. The commanded angles from Unity (Command) and the actual angles reported by the ESP32 (Feedback) were logged at 200 ms intervals over a period of approximately 200 s.

The synchronization error for each joint is defined as

$$Error_i(t) = Command_i(t) - Feedback_i(t) \quad (7)$$

The quantitative results are summarized in Table 3, and the time-domain comparison is illustrated in Figure 7.

Table 3. Synchronization accuracy in Unity Control Mode.

Joint	MAE (°)	RMSE (°)	Max Error (°)
Joint 2	6.88	10.77	41.50
Joint 3	5.20	8.72	29.25

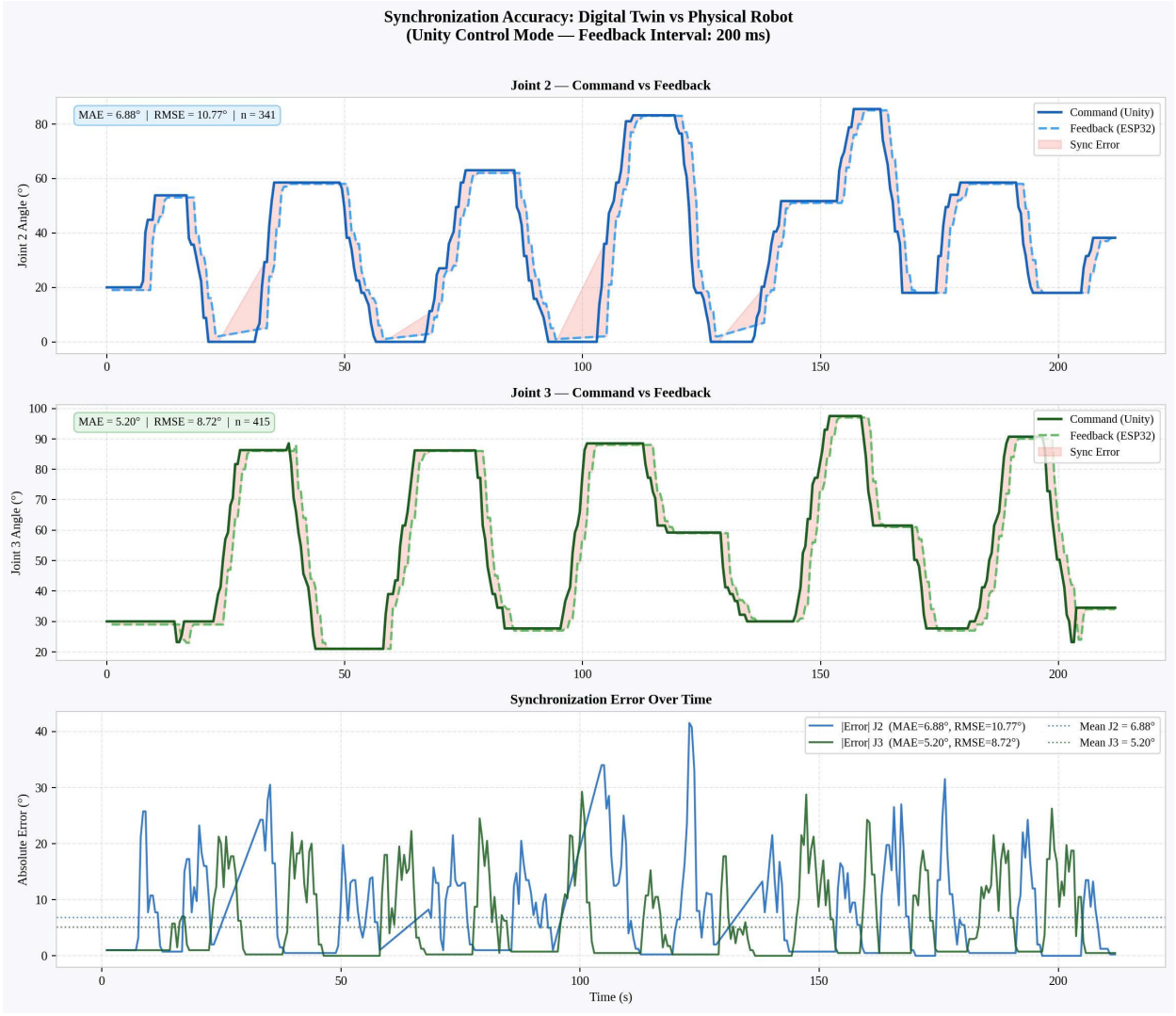


Figure 7. Synchronization accuracy: Digital Twin vs physical robot (Unity Control Mode).

The synchronization errors observed during dynamic motion are primarily attributable to the exponential smoothing filter implemented on the ESP32, which introduces a deliberate lag in servo actuation to prevent abrupt mechanical stress. With a smoothing coefficient of $\alpha = 0.15$ and a control loop period of 20 ms, the servo position converges asymptotically toward the commanded angle, resulting in a transient tracking error during rapid slider movements. Importantly, when the commanded angle is held constant, the feedback value converges to the command value with zero steady-state error, confirming that the system achieves accurate static synchronization.

The higher error observed in Joint 2 compared to Joint 3 is consistent with the faster and wider slider movements applied to Joint 2 during the experiment, which amplifies the transient lag effect of the smoothing filter. These results indicate that the Digital Twin accurately reflects the robot's steady-state position, while transient errors during fast motion are an expected consequence of the motion smoothing design choice.

5. Conclusions

This paper presented the design and implementation of a cloud-based Digital Twin framework for a 6-DOF robotic manipulator, integrating a Unity 3D virtual environment with an ESP32-controlled physical robot via the Firebase Realtime Database. The proposed system enables bidirectional synchronization, interactive joint control, and explicit dual-mode switching between Unity Control Mode and Serial Control Mode.

Experimental results demonstrated that the Firebase-based communication layer achieves near-real-time performance suitable for supervisory monitoring applications. Synchronization accuracy analysis confirmed that steady-state alignment between the Digital Twin and the physical robot is achieved, while transient tracking errors during dynamic motion are primarily attributable to the implemented exponential smoothing filter rather than control instability.

Compared with conventional ROS- or MATLAB R2023a-based Digital Twin platforms, the proposed architecture provides a lightweight and accessible deployment pathway without requiring high-end hardware or advanced system administration expertise.

Despite these contributions, the current system operates in an open-loop configuration without external encoder feedback, and cloud-based communication introduces latency that may limit strict real-time control applications.

Future work will focus on integrating physical feedback sensors for closed-loop control, adopting lower-latency communication protocols such as MQTT or WebSocket-based architectures, and extending the framework toward predictive simulation and multi-robot scalability.

Author Contributions: Conceptualization, T.D.D.; methodology, T.D.D. and N.K.V.; software, M.H.P.V.; validation, T.L.N.T.; writing—original draft preparation, M.H.P.V.; writing—review and editing, T.D.D., N.K.V. and T.C.D.; supervision, T.C.D.; project administration, T.D.D.; funding acquisition, T.C.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the University of Economics Ho Chi Minh City (UEH University), Vietnam.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data generated or analyzed during this study are included in this published article.

Acknowledgments: The authors would also like to thank the University of Economics Ho Chi Minh City–UEH University in Vietnam for providing financial support and study facilities. During the

preparation of this manuscript, the authors used Claude (Anthropic, Claude 5.5 Sonnet) for the purposes of language translation and grammar editing. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

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

References

1. Iqbal, J.; Islam, R.; Khan, H. Modeling and analysis of a 6 DOF robotic arm manipulator. *Can. J. Electr. Electron. Eng.* **2012**, *3*, 300–306.
2. Thakur, S.; Barai, R.K. Addressing model uncertainties and external disturbances in optimal robust control for vibration reduction in a flexible link manipulator. *J. Sci. Ind. Res.* **2023**, *82*, 700–710. [CrossRef]
3. Ibrahim, I.A.; Baballe, M.A. The ups and downs of robotic arms: Navigating the challenges. *Glob. J. Res. Eng. Comput. Sci.* **2024**, *4*, 78–82. [CrossRef]
4. Mazumder, A.; Sahed, M.F.; Tasneem, Z.; Das, P.; Badal, F.R.; Ali, M.F.; Ahamed, M.H.; Abhi, S.H.; Sarker, S.K.; Das, S.K.; et al. Towards next generation digital twin in robotics: Trends, scopes, challenges, and future. *Heliyon* **2023**, *9*, e13359. [CrossRef]
5. Genel, Ö.E.; La Regina, R.; Pappalardo, C.M. Performance comparison of the Simscape Multibody solvers for articulated mechanical systems. In *New Technologies, Development and Application VI*; Karabegović, I., Kovačević, A., Mandžuka, S., Eds.; Lecture Notes in Networks and Systems; Springer: Cham, Switzerland, 2023; Volume 687, pp. 321–328. [CrossRef]
6. Kam, H.R.; Lee, S.-H.; Park, T.; Kim, C.-H. RViz: A toolkit for real domain data visualization. *Telecommun. Syst.* **2015**, *60*, 337–345. [CrossRef]
7. Görner, M.; Haschke, R.; Ritter, H.; Zhang, J. MoveIt! Task Constructor for task-level motion planning. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 190–196. [CrossRef]
8. Koenig, N.; Howard, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan, 28 September–2 October 2004; pp. 2149–2154. [CrossRef]
9. Mai, H.T.; Nguyen, T.S.; Hoang, H.; Nguyen, T.K.; Tran, D.T. A MATLAB/Simulink–Gazebo co-simulation framework for environmental and hazardous applications: A 4-DOF manipulator case study. *IOP Conf. Ser. Earth Environ. Sci.* **2025**, *1565*, 012023. [CrossRef]
10. Nguyen, T.N.; Nguyen, H.T.A.; Dao, N.A.T.; Mai, N.T. Application of digital twin technology in real–virtual simulation of a coaxial spherical parallel mechanism. *TNU J. Sci. Technol.* **2025**, *230*, 276–283.
11. Ali, M.; Giri, S.; Liu, S.; Yang, Q. Digital twin synchronization: Bridging the sim-RL agent to a real-time robotic additive manufacturing control. *arXiv* **2025**, arXiv:2501.18016.
12. Chen, S.; Spyrakos-Papastavridis, E.; Jin, Y.; Deng, Y. Goal-oriented semantic communication for robot arm reconstruction in digital twin: Feature and temporal selections. *arXiv* **2024**, arXiv:2411.08835.
13. Amirkhanov, B.; Amirkhanova, G.; Kunelbayev, M.; Adilzhanova, S.; Tokhtassyn, M. Evaluating HTTP, MQTT over TCP and MQTT over WebSocket for digital twin applications: A comparative analysis on latency, stability, and integration. *Int. J. Innov. Res. Sci. Stud.* **2025**, *8*, 679–694. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.