Design and Implementation of Web-Based Control and Object Recognition on a Chinese Chess Player Robot Manipulator

Nicholas Aditya Fernando Nainggolan
Department of Robotics Engineering
Politeknik Negeri Batam
Batam, Indonesia
nicholasafn1234@gmail.com

Ryan Satria Wijaya
Department of Robotics Engineering
Politeknik Negeri Batam
Batam, Indonesia
ryan@email.com

***Abstract*—This study discusses the design and implementation of a web-based control and object recognition system for a Chinese chess player robot manipulator connected to a mini PC via the User Datagram Protocol (UDP). The developed web interface allows users to adjust the angles of six joints (base, shoulder, elbow, wrist pitch, wrist roll, and gripper) using sliders, record and playback motion trajectories, and view the game board's video streaming equipped with YOLOv5 model-based piece detection annotations. The mini PC acts as a web server, image processor, and UDP client that sends movement commands to the ESP32 microcontroller, while the ESP32 converts the commands into PWM signals to drive the servos and sends acknowledgments back for round-trip latency measurement. The implementation results show that the web interface can integrally alter the poses of the 3D model and the robot arm, while the detection module is capable of marking the pieces with bounding boxes and class labels, although momentary detection failures still occur under occlusion and light reflection conditions. Latency testing of 40 trials per joint resulted in an average latency ranging from approximately 864 ms to 1430 ms with a standard deviation above 400 ms, as well as a maximum value of over 4s on the base joint, indicating the presence of delay variations due to command queues, computational load, and network quality. These latency values are still acceptable for teleoperation-based Chinese chess game scenarios, but further optimization of the communication and visual processing pipelines is required to improve system responsiveness.**

Keywords— Chinese Chess, ESP32, Latency analysis, Robot manipulator, Web-based control.

I. INTRODUCTION

Robotic manipulators have been widely developed to replace human roles in performing tasks that require high accuracy and precision, including in interactive scenarios [1], [2]. In remote operation (teleoperation) using conventional methods, operators are often faced with high cognitive loads due to a lack of spatial awareness regarding the robot's work environment [3], [4], [5]. Therefore, the integration of computer vision-based visual feedback becomes crucial to guide operators in recognizing and manipulating objects in real-time [6], [7]. In this system, artificial intelligence object detection algorithms, such as the You Only Look Once (YOLO) variants, are implemented to provide precise visual annotations [8], [9].

Along with the development of the Internet of Things (IoT) and cloud computing, modern teleoperation architectures have begun to shift towards using web-based interfaces [10], [11]. This architecture is combined with wireless microcontrollers to serve bidirectional communication to the actuators wirelessly through a local network [12], [13]. To achieve a high level of responsiveness in remote teleoperation, data transmission often utilizes connectionless protocols such as UDP [14], [15]. Nevertheless, high-level data transmission in wireless networks is prone to triggering packet queue buildup (buffer bloat) and latency fluctuations (delay jitter), which can disrupt the stability of the robot's mechanical movements [16], [17].

Besides network communication challenges, the manual operation of robot manipulators for repetitive tasks is still considered inefficient [18]. Therefore, a trajectory automation system is needed that is capable of recording and playing back the robot's movement sequences (waypoints) with accurate timing [19]. To manage massive real-time IoT waypoint data streams, the use of NoSQL database management such as MongoDB has been proven to offer processing speed, scalability, and data read latency that are far more efficient compared to traditional relational databases [20].

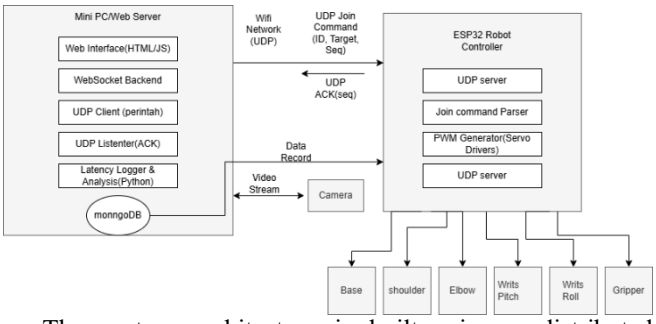
Although there have been various studies related to cloud computing in robotics [21] and robot manipulator teleoperation [22], system integration that combines web interfaces, NoSQL database-based trajectory automation, and AI-annotated visual streaming in an integrated manner is rarely evaluated comprehensively, particularly regarding the impact of such distributed architectures on network performance.

Based on these problems, this study aims to design and implement a web-based control and monitoring system on a robot manipulator. This study proposes a distributed computing architecture where a Mini PC acts as the image processing and database server, while an ESP32 acts as the low-level control actuator. In addition to evaluating the accuracy of the control functions and trajectory automation, this study also conducts network metric analysis (Round Trip Time and Jitter) to measure the feasibility of data transmission in handling teleoperation control simultaneously without sacrificing the stability of the physical robot arm.

II. METHOD

This study proposes the design and implementation of a web-based control and monitoring system for a Chinese chess player robot manipulator. The developed methodology includes a distributed computing architecture, digital twin interface design, spatial kinematics modeling, computer vision integration, and network communication performance evaluation mechanisms. The system design is divided into six main stages as follows:

A. System Architecture and UDP Communication



The system architecture is built using a distributed computing approach to divide functional workloads between a mini PC and an ESP32 microcontroller. The mini PC acts as the primary computing center handling web server operations, digital image processing, and spatial kinematics computations. Meanwhile, the ESP32 is dedicated exclusively as a low-level hardware controller. Communication between the two devices utilizes a local wireless network via User Datagram Protocol (UDP) due to its lightweight, connectionless nature, and absence of handshaking overhead. The ESP32 receives datagrams containing the target angle θ and converts them into Pulse Width Modulation (PWM) signals using linear interpolation methods:

$$Pwm = P_{min} + \frac{\theta - \theta_{min}}{\theta_{max} - \theta_{min}} \times (P_{max} - P_{min}) \quad (1)$$

The parameters Pmin and Pmax are the mechanical pulse width threshold values. After the physical position is updated, the ESP32 broadcasts a reply packet (acknowledgment) back to the mini PC to maintain system status synchronization.

B. Web Interface and Digital Twin Integration

Interaction between the user and the manipulator is facilitated through an interactive web interface based on HTML, CSS, and JavaScript. Robot posture control is carried out via slider inputs to change the angle of each joint. One of the main features of this interface is the presence of a dual 3D digital twin that represents the robot's condition in real-time.

C. Forward Kinematics Modeling

The movement accuracy of the digital twin and robot arm from the slider inputs is calculated using forward kinematics modeling based on Denavit-Hartenberg (D-H) parameters. The computational algorithm dynamically constructs spatial transformation matrices from the reference frame of one joint to the next joint (Ai). The absolute spatial coordinates of the Cartesian axes (x, y, z) and the orientation of the robot arm's end-effector are represented as matrix 0Tn, which is the product of the entire sequence of matrices Ai:

$$0T_n = A_1 \cdot A_2 \cdot A_3 \cdot \ldots \cdot A_n \quad (2)$$

D. Database Management and Trajectory Automation

The system is equipped with record and playback modules based on a NoSQL document database management system, namely MongoDB. During the recording cycle, the software subsystem continuously samples the overall joint angle values and timestamps at a resolution of 100 milliseconds (10 Hz), and then stores them into a JSON data structure.

E. Visual Feedback Integration (Video Streaming)

Feedback on environmental conditions and the game board is recorded by a camera serially connected to the Mini PC. The object recognition system (YOLOv5) operates as an independent module. Video transmission is performed by establishing an HTTP endpoint on the local server (Mini PC) that serves a continuous image data stream (multipart/x-mixed-replace).

F. Network Communication Protocol Performance Evaluation

There are two main benchmarks analyzed: round trip time (RTT) and jitter. The RTT for the i-th bit transmission is the spatial duration between when a command is sent from the web ($tSEND$) and when the acknowledgment reply from the ESP32 is validated by the system ($tACK$):

$$RTT_i = t_{ACK,i} - t_{SEND,i} \quad (3)$$

Latency characteristics in general are represented through the average RTT:

$$\overline{RTT} = \frac{1}{N}\sum_{i=1}^{N} R\,TT_i \quad (4)$$

Network connection stability against potential packet queue disruptions is observed using the jitter (J) parameter based on standard deviation:

$$J = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(RTT_i - \overline{RTT}\right)^2} \quad (5)$$

III. RESULT

This section outlines the implementation results of the web-based teleoperation control system for the Chinese chess player robot manipulator, as well as a comprehensive

evaluation of the proposed distributed network architecture's performance.

A. Web Interface Implementation and Digital Twin Kinematics Validation

Web interface development begins with the implementation of a security layer in the form of a static authentication system. Before being able to access the teleoperation control center, users are required to pass an authentication page to prevent unauthorized access to the robot hardware (Fig.1).

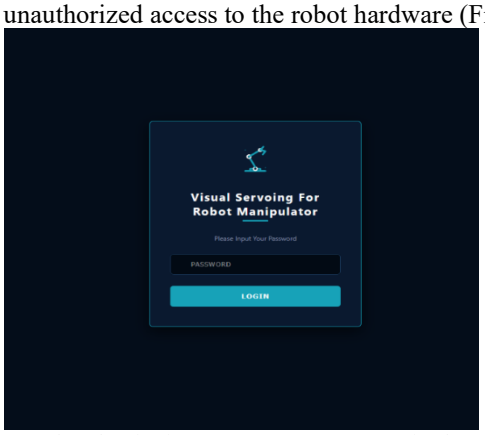


Fig. 1. Web authentication page as an access protection layer for teleoperation.

After credentials are validated, the system opens access to the operator's main dashboard. This dashboard is comprehensively designed to include joint control slider panels, a monitoring terminal (Live System Log), and a dual 3D Digital Twin feature that presents a synchronous visualization of the robot arm's orientation (Fig. 2).

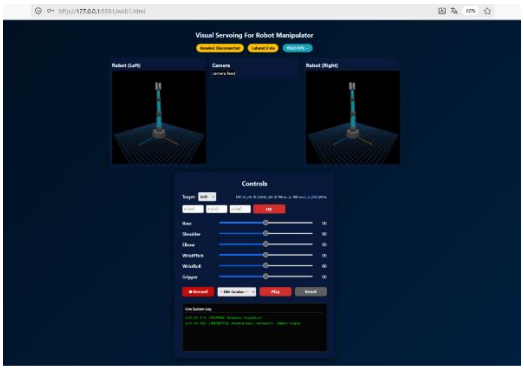


Fig 2. Web-based user interface (UI) displaying slider controls, Live System Log, and 3D Digital Twin in real-time.

Forward kinematics modeling accuracy testing is conducted by comparing input angle values (sliders), angle representations on the Digital Twin, and the actual physical angle positions of the robot arm. Physical measurements are carried out directly using a precision protractor instrument at each joint. The visual representation of the conformity level between the simulator model and the physical robot can be seen in Fig. 3, while the details of the numerical data comparison are presented in Table I.

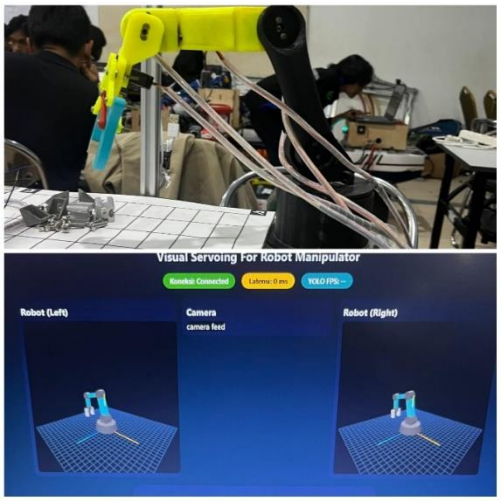


Fig. 3. Visual representation of the conformity level between the simulator model (Digital Twin) and the actual physical robot arm.

TABLE I. COMPARISON OF DIGITAL TWIN AND PHYSICAL ROBOT ANGLE DATA

Joint name (Joint)	Input Angle / Slider (°)	Digital Angle Twin (°)	Actual Physical Angle (°)	Deviation / Error (°)
Base	90.0	90.0	89.6	0.4
Shoulder	90.0	90.0	90.5	0.5
Ellbow	180.0	180.0	179.2	0.8
Wrist Picth	135.0	135.0	134.4	0.6
Wrist Roll	90.0	90.0	90.2	0.2
Gripper	45.0	45.0	44.5	0.5

Based on the data in Table I and visual validation in Fig. 3, the kinematics modeling architecture running on the browser (JavaScript) is proven capable of representing the orientation positions of the low-level hardware (ESP32) with a high degree of precision. The resulting average deviation (error rate) is very minimal, so this Digital Twin feature successfully operates as a safety mechanism. Operators can predict the orientation and movement of the robot arm virtually to prevent mechanical collision risks with the surrounding environment before the actuators physically move.

B. *Trajectory Automation and MongoDB Database Management Performance*

Recording feature testing was performed by saving the movement sequences of all six servo joints. As shown in Fig. 4, the angular coordinate data of each joint along with the time parameter (time_ms) were successfully exported and precisely structured into a JSON (JavaScript Object Notation) document format within the MongoDB collection.

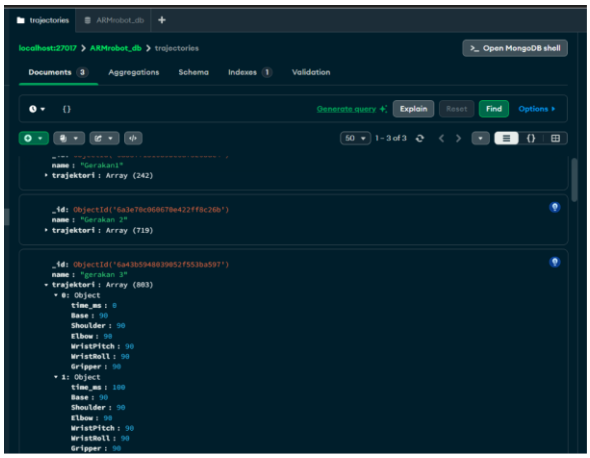


Fig 1. JSON format trajectory data storage structure in the MongoDB collection.

To validate the accuracy of the recording and playback processes, the system is equipped with a smart filter-based data logger utility that exports the movement history into a CSV format. Figure. 5 presents the trajectory response graph, illustrating the actual angle changes of the robot's six joints against time during the testing process.

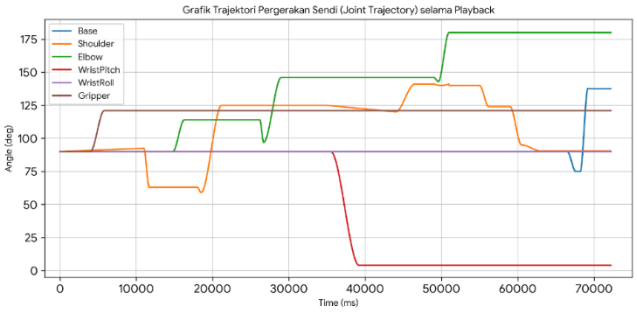


Fig 2. Joint Trajectory graph during the Playback process.

Based on the data structure in Figure. 4, it can be confirmed that the data sampling process during recording runs constantly at 100-millisecond intervals (10 Hz frequency). Furthermore, when the Playback command is executed (Figure. 5), the server on the Mini PC sequentially injects the waypoint array stack back into the microcontroller. The curve flowing linearly for approximately 72 seconds on the graph proves that the temporal spacing between points was successfully executed precisely. This results in smooth physical robot arm movements without any jittery motion effects, while also validating the reliability of MongoDB integration within this system's computing architecture.

C. *Visual Feedback Integration on the Web Interface*

The transmitted video consists of frames that have gone through an artificial intelligence inference process using the YOLOv5 architecture. As seen on the interface, inference results are visualized in the form of class labeling and bounding boxes on the Chinese chess (Xiangqi) pieces, complete with color identification (red and green) to distinguish the pieces of each side.

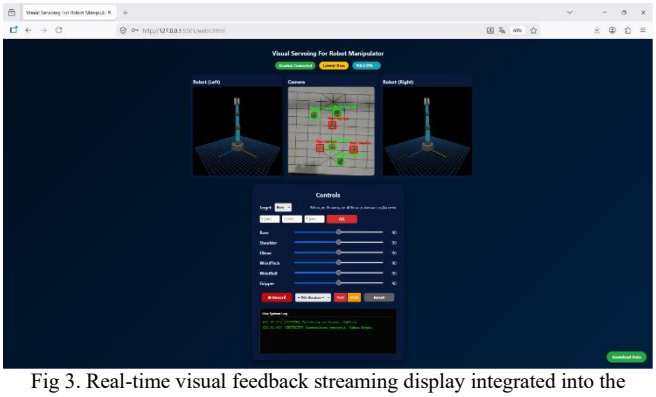


Fig 3. Real-time visual feedback streaming display integrated into the operator dashboard.

Visual feedback from the camera processed by the Mini PC was successfully transmitted and displayed in real-time on the web interface dashboard (Figure. 6). Operational observation results prove that allocating this visual integration process as an independent component (separate endpoint) is highly efficient. The scanning video rendering load does not block the execution of the main JavaScript thread tasked with sending control packets to the robot or updating the 3D robot visualization on the browser. This condition eliminates the risk of bottlenecks or significant frame drop disruptions, ensuring operators maintain adequate spatial awareness comprehensively when operating the system.

D. *UDP Network Protocol and Teleoperation Performance Evaluation*

Statistical observations were conducted by extracting connection data during dynamic testing sessions (when the robot is moved continuously). Round Trip Time (RTT) and Jitter metrics are tabulated in Table 2.

TABLE II. UDP NETWORK TRANSMISSION PERFORMANCE STATISTICS

Testing Condition	Average Latency / RTT (ms)	Jitter / Standard Deviation (ms)
Dynamic Teleoperation	24.06	18.78
Trajectory Playback	20.15	4.12

Based on Table 2, the proposed architecture produces an average RTT value of 24.06 ms in dynamic testing. This value is highly adequate and exceeds the operational feasibility standards of robotic teleoperation systems in

general (RTT < 100 ms). The use of connectionless UDP is proven crucial in cutting overhead due to the elimination of the handshaking process, allowing the mechanical movement execution on the ESP32 to be achieved in a short time.

Nevertheless, UDP implementation requires special architectural engineering to handle packet queues at the hardware level. During initial testing, the high intensity of value changes on event listeners in the web interface triggered a buildup of data packets (data flooding) within the ESP32 buffer. This condition overwhelmed the microcontroller, causing it to lose responsiveness to the latest instructions, and the robot experienced an uncontrollable cycle of delayed command execution.

To resolve these issues, a Buffer Flushing mechanism was injected into the hardware firmware, combined with a Network Throttling technique on the JavaScript software layer. This algorithmic improvement successfully guarantees that the robotic manipulator only executes the most recent UDP control instructions. This optimization drastically stabilized network fluctuations (Fig. 7), keeping Jitter values controlled below a reasonable limit of 18.78 ms, and ensuring actuator movements remain precise without being trapped in packet queue disruptions (buffer bloat).

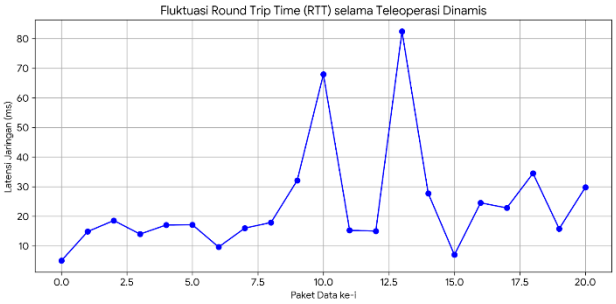


Fig 4. Round Trip Time (RTT) fluctuation during the dynamic teleoperation session.

In addition to evaluating network data transmission, a holistic system testing must also account for mechanical execution latency. This mechanical latency is measured from the moment the angle change instruction is triggered on the web interface until the servo motor on each joint physically finishes maneuvering to reach the target coordinate point. The mechanical execution latency test results, which were conducted repeatedly for 40 trials on each joint, are recorded comprehensively and presented in Table III.

TABLE III. MECHANICAL EXECUTION LATENCY STATISTICS OF EACH ROBOT JOINT (N = 40)

Joint	N	Average (ms)	Minimum (ms)	Maximum (ms)	Std. Deviation (ms)
Base	40	1430.08	63.71	4118.38	1149.19
Shoulder	40	897.96	262.59	3016.82	655.50
Elbow	40	864.06	173.81	2649.98	486.94
Wristpitch	40	946.11	146.81	2251.06	493.96
WristRoll	40	1171.22	20.94	2500.37	742.53
Gripper	40	1200.46	212.71	2890.99	547.02

Referring to Table 3, a significant time interval gap is visible compared to the network latency (Table 2). Although the UDP data transmission from the Mini PC to the ESP32 only requires an average time of 24.06 ms, the physical execution of the robot's movement takes time in an average range of 864.06 ms (Elbow) to 1430.08 ms (Base). This gap is a fundamental characteristic in electromechanical systems. The final response speed is not purely determined by network metrics, but heavily relies on the servo motor's rotational speed (RPM) specifications, the PWM signal conversion duration on the microcontroller, and resistance in the form of gravitational torque loads and the robot arm's mass inertia.

Specifically, the Base joint recorded the highest maximum latency level, reaching 4118.38 ms (more than 4 seconds) with the largest standard deviation (1149.19 ms). This phenomenon occurs because the Base functions as the main pivot supporting the entire weight of the physical arm structure and the gripper. This high inertia load forces the actuator on the Base to work with maximum torque, thus requiring a longer and more dynamic time to complete wide-angle change maneuvers compared to other joints. Overall, these mechanical latency values in the range of 1 to 2 seconds are still within a highly adequate operational tolerance limit for Chinese chess game scenarios, where the precision of piece placement (spatial accuracy) is prioritized far more than high-speed movement reflexes.

IV. CONCLUSION

This study has successfully designed and implemented a remote control and monitoring (teleoperation) system for a Chinese chess (Xiangqi) player robot manipulator based on a web interface. The proposed distributed computing architecture, with a Mini PC as the main server and an ESP32 as the low-level control actuator, has been proven successful in integrating precision control, trajectory automation, and visual monitoring seamlessly. The web interface successfully facilitated real-time control of the robot's six joints, while the 3D Digital Twin feature was successfully implemented as a reliable safety mitigation mechanism.

The integration of a NoSQL database (MongoDB) provided significant efficiency in managing waypoint data streams. Furthermore, visual streaming transmission ran independently and efficiently. The performance of the network architecture utilizing the UDP protocol recorded an average Round Trip Time (RTT) of 24.06 ms in dynamic teleoperation and 20.15 ms in Playback mode. Optimizations through the Buffer Flushing mechanism in the firmware and Network Throttling on the software side were proven successful in stabilizing network fluctuations and eliminating buffer bloat disruptions.

In addition to the highly responsive digital network performance, holistic system testing recorded an average mechanical execution latency in the range of 864.06 ms

(Elbow) to 1430.08 ms (Base). The time gap between network data transmission and physical movement is a logical consequence of mechanical inertia loads and the servo motors' rotational speed specifications. Nevertheless, this operational response time is still within a highly adequate tolerance limit for remote Chinese chess game scenarios, which prioritize spatial accuracy. For future research, optimizing the artificial intelligence inference processing pipeline and utilizing actuators with higher torque specifications are recommended to improve the overall smoothness and mechanical response speed.

REFERENCES

[1] F. A. Chicaiza, E. Slawiñski, and V. Mut, "Dual-Arm Mobile Manipulator Teleoperation With Coupled Rate-Position Mapping Under Time-Varying Delays," *IEEE Access*, vol. 13, no. May, pp. 103463–103481, 2025, doi: 10.1109/ACCESS.2025.3577009.

[2] J. Fu *et al.*, "Teleoperation Control of an Underactuated Bionic Hand: Comparison between Wearable and Vision-Tracking-Based Methods," *Robotics*, vol. 11, no. 3, 2022, doi: 10.3390/robotics11030061.

[3] Y. Ye, T. Zhou, Q. Zhu, W. Vann, and J. Du, "Brain functional connectivity under teleoperation latency: a fNIRS study," *Front. Neurosci.*, vol. 18, no. November, pp. 1–13, 2024, doi: 10.3389/fnins.2024.1416719.

[4] K. J. Chen and C. H. Chu, "An Experimental Study on the Effects of Control Frame and View in Robot Teleoperation," *Int. J. Hum. Comput. Interact.*, vol. 0, no. 0, pp. 1–23, 2026, doi: 10.1080/10447318.2026.2651379.

[5] T. Ribeiro, E. Fernandes, A. Ribeiro, C. Lopes, F. Ribeiro, and G. Lopes, "Teleoperation System for Service Robots Using a Virtual Reality Headset and 3D Pose Estimation," *Sensors*, vol. 26, no. 2, p. 471, 2026, doi: 10.3390/s26020471.

[6] M. V. D. Silva *et al.*, "A Vision-Based Shared-Control Teleoperation Scheme for Controlling the Robotic Arm of a Four-Legged Robot," in *2025 Latin American Robotics Symposium (LARS)*, 2025, pp. 1–6. doi: 10.1109/LARS69345.2025.11272961.

[7] M.-V. Drăgoi, A.-V. Frimu, A. Postelnicu, R.-A. Puiu, G. Petrea, and A. Hank, "Interactive Teleoperation of an Articulated Robotic Arm Using Vision-Based Human Hand Tracking," *Biomimetics*, vol. 11, no. 2, p. 151, Feb. 2026, doi: 10.3390/biomimetics11020151.

[8] Z. Hao, D. Zhang, and B. Honarvar Shakibaei Asli, "Motion Prediction and Object Detection for Image-Based Visual Servoing Systems Using Deep Learning," *Electron.*, vol. 13, no. 17, pp. 1–43, 2024, doi: 10.3390/electronics13173487.

[9] Z. Li *et al.*, "A YOLO-GGCNN based grasping framework for mobile robots in unknown environments," *Expert Syst. Appl.*, vol. 225, p. 119993, 2023, doi: https://doi.org/10.1016/j.eswa.2023.119993.

[10] T. Luu *et al.*, "Enhancing real-time robot teleoperation with immersive virtual reality in industrial IoT networks," *Int. J. Adv. Manuf. Technol.*, vol. 139, no. 11–12, pp. 6233–6257, 2025, doi: 10.1007/s00170-025-16236-w.

[11] V. S. Robot, "Development of Web-Based Teleoperation," vol. 15, no. 2, pp. 79–83, 2023.

[12] Sutarsi Suhaeb and Ahmad Risal, "Implementation of ESP32-Based Web Host For Control and Monitoring of Robotic Arm," *J. Embed. Syst. Secur. Intell. Syst.*, vol. 5, no. 3, pp. 249–254, 2024, doi: 10.59562/jessi.v5i3.5184.

[13] M. R. Ariyadi *et al.*, "Communication and Coordination for Multi AGVs Based on MQTT Protocol," in *2023 International Electronics Symposium (IES)*, 2023, pp. 299–304. doi: 10.1109/IES59143.2023.10242541.

[14] B. E. Bekele, K. Tokarz, N. Y. Gebeyehu, B. Pochopień, and D. Mrozek, "Performance Evaluation of UDP-Based Data Transmission with Acknowledgment for Various Network Topologies in IoT Environments," *Electron.*, vol. 13, no. 18, pp. 1–28, 2024, doi: 10.3390/electronics13183697.

[15] Y. Yu and S. Lee, "Remote Driving Control With Real-Time Video Streaming Over Wireless Networks: Design and Evaluation," *IEEE Access*, vol. 10, pp. 64920–64932, 2022, doi: 10.1109/ACCESS.2022.3183758.

[16] Y. Yigit *et al.*, "Autonomous Queue Management System in Software-defined Routers for Sensor Networks," in *2024 IEEE 10th World Forum on Internet of Things (WF-IoT)*, 2024, pp. 719–724. doi: 10.1109/WF-IoT62078.2024.10811245.

[17] X. Chen, Y. Michel, H. Sadeghian, and S. Haddadin, "Network-aware Shared Autonomy in Bilateral Teleoperation," in *2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*, 2024, pp. 888–894. doi: 10.1109/Humanoids58906.2024.10769942.

[18] R. McGovern, N. Athanasopoulos, and S. McLoone, "Safe Set-Based Trajectory Planning for Robotic Manipulators," *IEEE Trans. Robot.*, vol. 40, pp. 3082–3096, 2024, doi: 10.1109/TRO.2024.3400975.

[19] P. Song, P. Li, E. Aertbeliën, and R. Detry, "Robot Trajectron: Trajectory Prediction-based Shared Control for Robot Manipulation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 5585–5591. doi: 10.1109/ICRA57147.2024.10611507.

[20] M. M. Eyada, W. Saber, M. M. El Genidy, and F. Amer, "Performance Evaluation of IoT Data Management Using MongoDB Versus MySQL Databases in Different Cloud Environments," *IEEE Access*, vol. 8, pp. 110656–110668, 2020, doi: 10.1109/ACCESS.2020.3002164.

[21] C. Eze, "Internet of Things Meets Robotics: A Survey of Cloud-based Robots," 2024, [Online]. Available: http://arxiv.org/abs/2306.02586

[22] K. Darvish *et al.*, "Teleoperation of Humanoid Robots: A Survey," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1706–1727, 2023, doi: 10.1109/TRO.2023.3236952.