

# Advanced Frameworks for Low-latency Teleoperation and LLM-driven Autonomous Exploration in Omnidirectional Mobile Robotics

Umar Ali Khan, Ahmad Ilyas, and Ekram Khan\*

Cite <https://doi.org/10.64589/juri/231349>

Submitted: March 11, 2026 Revised: April 08, 2026 Accepted: April 13, 2026

## ABSTRACT

This study presents the design and implementation of a robust low-latency teleoperation system that integrates extended reality (XR) with embedded robotics. This system bridges the gap between high-level XR development and low-level embedded control by interfacing an AjnaXR headset with a custom-built 4-wheel drive Mecanum robot. This framework is important because inherent latency and network instability in wireless human-robot interaction historically cause cognitive dissonance and severe limitations in operational safety. To address these challenges, the solution utilizes a centralized star network topology employing a multicast domain name system for reliable device discovery without relying on static Internet protocol addressing. The robot architecture was decoupled into two independent ESP32-S3 microcontroller nodes: a control node handling asynchronous motor commands and a vision node dedicated to streaming the quarter video graphics array video. The system implements a direct input-mapping paradigm, bypassing traditional ray-casting user-interface interactions, to minimize the input-to-actuation delay. The experimental results demonstrated a control latency of approximately 50 ms and a stable first-person view video stream at 20–25 frames per second. In addition, this study explores the integration of a large-language-model-driven agentic exploration framework that enables natural-language-guided autonomous navigation using semantic mapping and finite-state machine orchestration.

**Keywords:** teleoperation, extended reality, human-robot interaction, large-language model, semantic mapping, agentic exploration

## 1. INTRODUCTION

The integration of extended reality (XR) technologies into mobile robotics has led to a profound paradigm shift in human-robot interaction (HRI). Teleoperation, the actuation and control of a robotic entity from a remote location, has historically been encumbered by two critical bottlenecks: high latency within the communication link and high cognitive load imposed on the human operator by unintuitive user interfaces<sup>1</sup>. As the demand for remote intervention in hazardous, industrial, and specialized environments accelerates, bridging the gap between high-level immersive interfaces and low-level embedded hardware has become an important engineering challenge.

Traditional teleoperation systems rely on two-dimensional (2D) screens and conventional input devices, such as keyboards or standard joysticks. XR, which encompasses both virtual reality (VR) and augmented reality, addresses this fundamental limitation by immersing the operator in a three-dimensional representation of the remote environment. However, the synthesis of XR and robotics introduces profound systemic complexities: an XR headset must render high-resolution environments at high frame rates while simultaneously transmitting control signals and

receiving high-bandwidth video telemetry over volatile wireless networks.

When the temporal gap between an operator's physical action and the robot's visual feedback exceeded approximately 200 ms, cognitive dissonance and task failure manifested rapidly. Research into the Tactile Internet confirms that traditional wireless networks introduce critical latency bottlenecks, causing severe cognitive dissonance and operational safety risks in real-time teleoperation<sup>2</sup>. Mitigating this latency is fundamental to operator ergonomics and mission success, current frameworks remain limited by the computational overhead of coupled hardware and indirect user interfaces. To address this knowledge gap, the primary aim of this study is to bridge the critical latency bottleneck in human-robot interaction and establish a safe, deterministic architectural framework for generative AI in physical environments. The novelty of this research lies in combining a physically decoupled, dual-node edge architecture with an OpenXR direct input mapping paradigm. By isolating asynchronous motor control from video telemetry and bypassing ray-casting bottlenecks, this study introduces an original solution that effectively minimizes control latency while paving the way for autonomous, large-language-model-driven exploration.

## 2. LITERATURE REVIEW/RELATED WORK

Modern human-robot interaction (HRI) prioritizes direct, low-cognitive-load inputs over traditional raycasting, which introduces nondeterministic latency in time-critical tasks<sup>3</sup>. To support this rapid interaction, asynchronous IoT servers (e.g., ESPAsyncWebServer) and mDNS provide robust, non-blocking network discovery, drastically outperforming synchronous protocols<sup>4</sup>. Within extended reality (XR), migrating to unified frameworks like OpenXR eliminates the vendor-specific component conflicts commonly found in legacy toolkits<sup>5,6</sup>. For autonomous navigation, recent architectures have demonstrated the viability of using Large Language Models (LLMs) as zero-shot planners; these systems parse symbolic semantic maps built via AprilTags and lightweight object detectors into discrete commands, bypassing the computational overhead of end-to-end neural policies<sup>7-9</sup>.

## 3. METHODOLOGY

**3.1. System Architecture.** The system architecture is designed around a centralized star topology combined with functional decomposition of the robot's tasks. The XR headset (client) and robot (server) connected wirelessly to the central Wi-Fi access point (mobile hotspot). A direct peer-to-peer (software-enabled access point) connection was rejected because Android devices acting as hotspots often randomize IP addresses and fail to provide reliable dynamic host configuration protocol services to connected clients. The central hotspot acts as a stable authority for the mDNS broadcasts, ensuring reliable discovery. The complete system architecture, illustrating the centralized star topology and dual microcontroller nodes, is provided in Supplementary Material S1.

The robot's functions are split between two independent microcontrollers: (1) the control node (ESP32-S3 wireless room (WROOM)), dedicated to listening for hypertext transfer protocol (HTTP) motor commands and generating pulse-width modulation (PWM) signals, and (2) the vision node (XIAO ESP32S3 Sense), dedicated to capturing, compressing, and streaming videos. This separation is a critical design decision; if both tasks are placed on a single chip, the heavy interrupt load of the camera interface (vertical synchronization, peripheral clock) introduces jitter into the motor PWM signals, resulting in erratic movement. Table 1 summarizes the system's nodal configurations.

**3.2. Microcontroller Framework: ESP32-S3.** ESP32-S3 features a dual-core Xtensa LX7 central processing unit running at 240 MHz, augmented with single instruction, multiple data vector instructions that accelerate mathematical operations. The Wi-Fi protocol stack was strictly pinned to Core 0 using the free real-time operating system (FreeRTOS) xTaskCreatePinnedToCore application programming interface (API), whereas the application logic for evaluating incoming HTTP strings and generating precise hardware PWM signals was pinned to Core 1. This asymmetric multiprocessing prevents background network management tasks from pre-empting time-critical motor actuation<sup>10</sup>.

**3.3. Kinematic Modeling and Omnidirectional Control.** The physical robot relied on a 4-wheel drive Mecanum drivetrain. Mecanum wheels feature passive rollers angled at

45° relative to the main axis of the wheel, enabling true omnidirectional movements, including forward, lateral (strafing), diagonal, and rotational trajectories, without altering the robot's physical heading<sup>11</sup>. The standard inverse kinematic Jacobian mapping and vector force analysis utilized for the base chassis omnidirectional control are detailed in Supplementary Material S2<sup>12</sup>. Figure 1 shows the assembled robotic platform

**3.4. Hardware Integration and Mechatronics.** Legacy bipolar junction transistor (BJT) motor drivers, such as the L298N, suffer from internal voltage drops in the range of 2.0–2.5 V, necessitating bulky aluminum heatsinks. The 7Semi Vikram-453R6Q Quad Channel Motor Driver utilizes the DRV8871 integrated chip with an advanced n-type metal-oxide semiconductor H-bridge topology and a drain-to-source resistance of  $R_{DS(on)} \approx 360 \text{ m}\Omega$ , resulting in a voltage drop below 0.7 V. The complete power wiring schematic and a detailed architectural comparison of the motor drivers are provided in Supplementary Material S3.

**3.4.1. Chassis design and material selection.** The structural framework employs a multimaterial fused deposition modeling additive manufacturing strategy: polylactic acid for the main double-deck chassis (high rigidity), thermoplastic polyurethane for the motor mounts (elastic vibration damping acting as a mechanical low-pass filter against Mecanum roller oscillations), and polyethylene terephthalate glycol for the structural couplers and camera mounts (high thermal resistance). A stacked mass-centralization layout was adopted comprising the Vikram motor driver, four direct-current motors, and a control node mounted on the lowest deck to lower the center of gravity and prevent chassis rolling during high-speed lateral strafing. The 3S lithium-ion polymer battery, pan-tilt servos, and vision nodes were elevated to the upper deck.

**3.5. Embedded Software and Asynchronous Networking.** The firmware was developed using an Arduino IDE (v2.3.2) with the ESP32 microcontroller board support package. The control node firmware was built on the ESPAsyncWebServer library, which leverages FreeRTOS to handle incoming network traffic asynchronously via lightweight interrupt-like callbacks. When the XR headset transmits a movement command via a stateless uniform resource identifier (URI) (for example, HTTP GET /move?dir=forward), the server immediately executes the corresponding motor function and queues the HTTP 200 response in the background. Experimental evaluations demonstrate that asynchronous HTTP and message queuing telemetry transport protocols provide considerably lower mean latency (~11 ms vs. ~41 ms) than synchronous WebSockets in resource-constrained IoT environments.

Upon booting, both nodes register their host names using the ESPmDNSh library: the control node as bot.local and the vision node as camera.local. The vision node employs a client-pull HTTP snapshot model via the capture endpoint rather than pushing frames in a persistent motion joint photographic experts group (JPEG) stream (which causes backlog buildup during channel degradation), and the XR headset explicitly requests a single JPEG frame via HTTP GET. The camera was configured to a quarter video graphics array resolution (320 × 240), JPEG quality index equal to 10, and a double-buffered frame pipeline (fb\_count = 2) using 8 MB pseudostatic random access memory

**Table 1.** Communication roles, protocols, and network identifiers for the primary system components

| Component                    | Hardware                               | Primary function                                                                                  | Protocol                               | Multicast domain name system (mDNS) identity (ID) |
|------------------------------|----------------------------------------|---------------------------------------------------------------------------------------------------|----------------------------------------|---------------------------------------------------|
| Extended reality (XR) client | AjnaXR headset                         | Sends movement commands; fetches video frames                                                     | Hypertext transfer protocol (HTTP) GET | Not applicable (N/A)                              |
| Control server               | ESP32-S3 microcontroller wireless room | Listens for commands; controls motors via pulse-width modulation                                  | HTTP GET (port 80)                     | <i>bot.local</i>                                  |
| Vision server                | Seeed XIAO ESP32S3                     | Captures and streams Joint Photographic Experts Group (JPEG) snapshots of First Person View (FPV) | HTTP GET (/capture)                    | <i>camera.local</i>                               |
| Network hub                  | Wi-Fi access point                     | Routes traffic; facilitates mDNS broadcasts                                                       | 802.11/Internet Protocol (IPv4)        | N/A                                               |

(RAM), yielding a stable 20–25 frames per second (FPS) stream with a per-frame payload of a few kilobytes. During continuous streaming, the OV2640 sensor can exceed 70 °C; passive copper heatsinks applied to the camera module's electromagnetic interference shield (EMI) shield can prevent thermal throttling and image artifacts.

**3.6. Extended Reality Application Layer.** The client-side interface was developed within the Unity Engine (2022.3 LTS) and deployed in the AjnaXR headset, which is a standalone Android-based device powered by the Qualcomm XR2+ Gen 1 platform with 6-degree-of-freedom inside-out tracking and Wi-Fi 6 connectivity. Initial attempts to use the MRTK generated phantom inputs caused unpredictable polling behavior due to conflicts between MRTK's abstracted input layers and the AjnaLens native Android drivers. The system was migrated to the XRI built strictly upon the OpenXR standard, providing a device-agnostic API that standardizes controller state reporting and eliminates vendor-specific SDK conflicts<sup>5,6</sup>.

The most critical design decision is the rejection of unity UI ray casting interactions in favor of direct input mapping. A custom C# script (DirectInputHandler.cs) was incorporated directly into the OpenXR Input Action Asset. When the operator presses the physical trigger, the hardware interrupts the fire, the OpenXR driver registers the state change (action is started), and the script immediately executes the network request to ESP32. The system also implements a deadman's switch: the instant the operator releases any control input (action canceled), an overriding stop command is transmitted to the robot, ensuring an immediate halt in the event of distraction or network failure.

A technical challenge specific to Android-based standalone VR headsets is that the Android OS does not resolve local domains using standard HTTP request libraries. The solution integrates a native Java Native Interface bridge to directly access the Android NsdManager, which resolves the raw IPv4 addresses of *bot.local* and *camera.local* and passes them back to the Unity C# environment for dynamic URI construction. For memory management, the video-handling script invokes `Destroy(oldTexture)` before each new texture is applied, bypassing the laundry C# Garbage Collector and maintaining a flat memory profile during extended

sessions. To validate this system stability, a custom C# profiling script was integrated into the Unity application to continuously monitor real-time RAM usage. Over a 10-min active teleoperation session, the total RAM footprint on the AjnaXR headset remained strictly bounded between 420 and 480 MB. This stable memory allocation confirms the absence of texture memory leaks and ensures reliable and uninterrupted rendering performance on standalone Android hardware. The complete system program flowchart detailing the logic loop from the Unity application through the network hub to the ESP32 firmware is provided in Supplementary Material S5.

**3.7. LLM-based Agentic Exploration.** For autonomous operation, the robot incrementally builds a lightweight semantic map by detecting visual cues at junctions using a You Only Look Once version 11 (YOLO11)-small object detector<sup>13</sup> to identify signboards, directional arrows, and store icons, whereas AprilTags serve as reliable geometric anchors<sup>14</sup> that complement simultaneous localization and mapping-oriented fast and rotated brief (ORB-SLAM3) global pose tracking<sup>15</sup>. The decision layer was powered by the Qwen2.5-VL-7B-Instruct vision-language model<sup>16</sup>, which acted as a discrete reasoning engine over a semantic graph. To facilitate this task reasoning, a dedicated host machine ran Python-based middleware that parses natural language user instructions and executes Fireworks artificial intelligence (AI) API calls to the vision-language models for high-level path planning.

At each junction, perception nodes append a symbolic record to a persistent JSON semantic map. This JSON structure is explicitly formatted to include a unique junction identifier, the estimated 2D pose (*x*, *y*, *yaw*) of the robot at the time of observation, and a list of directional relations mapping the available branches (for example, left, straight, and right) to specific points of interest (POIs) or store icons. The LLM is prompted with this JSON map, the remaining task order, and a historical log of visited junctions, outputting a constrained command in the format `<DIRECTION>|||<STORE_ACTION>`. To minimize inference latency, the framework implements a payload optimization strategy rather than querying the model for every microstep;

**Table 2.** Summary of performance metrics, including communication latency and video frame rates

| Metric                  | Observed value          | Systemic implication                                                                               |
|-------------------------|-------------------------|----------------------------------------------------------------------------------------------------|
| Control command latency | ~50 ms                  | Near-instantaneous actuation; eliminates overcorrection and operator oscillation                   |
| Video streaming latency | ~150 ms                 | Well below the critical 200 ms cognitive dissonance threshold, enabling safe high-speed navigation |
| Video frame rate        | 20–25 frames per second | Smooth visual flow; minimizes virtual-reality-induced motion sickness                              |
| Input mapping overhead  | <5 ms                   | Direct OpenXR bindings drastically outperform graphical user interface raycasting pipelines        |

a single API call generates a batched sequence of navigation commands that govern several seconds of autonomous movement.

The main controller's finite-state machine executes these commands and orchestrates motion primitives, transitioning between states such as WALL\_AVOID, APRILTAG\_APPROACH, ENTER\_STORE, and grasping routines. By restricting the LLM to high-level semantic decisions and relying on classical low-latency ESP32 controllers for physical execution, the system ensures that the LLM hallucinations<sup>8</sup> do not result in dangerous physical motor commands.

## 4. RESULTS AND DISCUSSION

**4.1. End-to-End Latency Metrics.** The system was tested to measure the end-to-end latency and video stability. The latency was calculated as the temporal differential between the transmission time of a command and the actuation time at the destination. Table 2 summarizes the observed performance metrics.

A control latency of ~50 ms is exceptionally low for a Wi-Fi-based microcontroller system utilizing standard transmission control protocol (TCP)/HTTP protocols. This rapid response is attributable to the combination of the ESPAsyncWebServer handling requests outside the main loop and direct input mapping, bypassing the Unity Physics and graphical user interface event systems. A video latency of ~150 ms represents the aggregate time for the OV2640 sensor exposure, ESP32-S3 JPEG compression, network transmission, and AjnaXR headset decoding and rendering. Although a centralized star topology utilizing a mobile hotspot yields an optimal control latency of ~50 ms

in controlled settings, its performance in congested radio frequency environments presents scalable challenges. The current architecture relies on the ESP32-S3 WROOM and Seeed XIAO ESP32S3 Sense modules, both of which operate exclusively within the 2.4 GHz spectrum. Experimental testing in a heavily congested laboratory environment utilizing a shared institutional router with multiple connected devices at increased physical distances demonstrated a measurable degradation in network determinism. This congestion results in increased mDNS resolution times and elevated command latency. To mitigate these 2.4 GHz spectrum limitations and maintain sub-100 ms latency in industrial deployments, future hardware iterations must be scaled to 5 GHz-capable microcontrollers to ensure isolated, high-bandwidth channels for robust teleoperation.

**4.2. Hardware and Thermal Reliability.** The physical separation of the control and vision nodes proved highly successful during extended operational testing. The control node maintained absolute determinism in its PWM generation; no motor jitter or stuttering was observed even when the vision node was subjected to the maximum network polling stress. The 7Semi Vikram-453R6Q driver remained within the safe operating temperatures without passive or active cooling, even under conditions wherein the peak current was drawn from the four Johnson motors. The OV2640 camera sensor exhibited considerable thermal gain during continuous streaming, reaching unmitigated temperatures in excess of 70 °C. Because the specified operating range for the Seeed XIAO ESP32S3 Sense is strictly rated between –20 and 65 °C, this ambient heat transfer exceeded hardware limits and subtly degraded the vision node's processing efficiency, resulting in frame rate fluctuations. To restore optimal thermal conditions, the module was upgraded by installing a single passive copper heat sink directly onto the thermal pad. This modification effectively arrested the severe thermal climb and brought the system back within its safe operating bounds, ensuring that the overall video telemetry remained stable at a target frame rate level of 20–25 FPS. Because the passive heat sink successfully prevented thermal throttling, additional active cooling mechanisms (such as microfans) were not considered, thereby preserving the power budget and mechanical simplicity of the robot. The empirical thermal profile of the sensor during a 15 min operational window, demonstrating the effectiveness of the copper heat sink, is provided in Supplementary Material S6.

**4.3. Input System Robustness.** The shift to direct input mapping eliminates the synchronization errors found in ray casting. The deadman's switch logic stopping motors on button release functioned reliably, ensuring that the robot did not drift when the operator lost the connection or released the controller.



**Figure 1.** Top-down view of the assembled 4-wheel drive Mecanum robot platform showing the vision node, pan-tilt camera mount, and motor layout

Migration from MRTK to OpenXR/XRI resolved all phantom input events and ensured cross-platform portability.

While mDNS reliably handles zero-configuration device discovery, maintaining robustness in nonlaboratory environments with increased 2.4 GHz spectrum congestion or packet loss requires careful protocol selection. The system mitigates these environmental networking challenges using the client-pull HTTP snapshot model. Rather than relying on a persistent MJPEG stream, which can suffer from backlog buildup and severe desynchronization during channel degradation, the XR headset explicitly requests individual JPEG frames. In the event of packet loss, the dropped frame does not crash the stream, and the client simply issues the next HTTP GET request. Furthermore, the asynchronous nature of ESPAsyncWebServer ensures that the dropped control packets do not block the microcontroller's main loop, allowing the robot to safely execute the deadman's switch protocol and halt actuation if sustained congestion interrupts the command link.

#### 4.4. Autonomous Exploration Performance Metrics.

The teleoperation framework requires strict real-time execution; however, adding an LLM-based autonomous layer naturally introduces processing overhead. To measure the impact of this extension, we evaluated the system based on the inference latency, mapping success rates, and overall task completion.

**4.4.1. Inference latency.** Running the YOLO11-small detector for signboard and POI recognition required ~28 ms per frame on the hardware used (NVIDIA GTX 1650). High-level reasoning handled by Qwen2.5-VL-7B-Instruct via the Fireworks AI API introduced a larger network delay. Based on the round-trip time between sending a payload and receiving a response, the LLM requires an average of 2.4 s to process the JSON semantic map and output a decision at each junction. To prevent this from stalling the progress of the robot, the system requests multiple sequential commands per API call. In addition, the MainController shifts the robot to a safe WAIT\_NEXT\_ACTION state while waiting, ensuring that the network delay does not interfere with the real-time obstacle avoidance loop.

**4.4.2. Semantic mapping reliability.** The metric SLAM (such as ORB-SLAM3) often suffers from tracking drift in featureless corridors. Our lightweight JSON mapping approach bypasses this by entirely depending on discrete visual markers. During testing in diverse corridor setups, the system successfully logged the junction-to-POI relationships 94% of the time. The use of AprilTags as geometric anchors proved to be highly effective for lining up approach maneuvers. Mapping or alignment failure was almost exclusively due to the camera losing sight of the tag, which is usually caused by an extreme approach angle or physical occlusion near the doorway.

**4.4.3. Task execution and instruction adherence.** To evaluate the zero-shot planning capabilities of the LLM in a physical environment, we conducted 20 real-world navigation trials by directing the robot to different target positions within the laboratory. The model extracts the correct items from the language requests and converts them into valid JSON orders with 95% accuracy. Over the course of the trials, the navigation logic successfully guided the robot to the correct unvisited target locations

and back to the pick-up zone 80% of the time. Maintaining a continuous history log and strictly formatting the system prompt were key factors in reducing errors, as these constraints prevented the LLM from outputting invalid directions or looping back to areas it had already visited.

## 5. CONCLUSIONS

This study successfully designed and validated a robust, low-latency XR teleoperation system for an omnidirectional Mecanum robot, extended with an LLM-driven autonomous exploration layer. By migrating to a standardized OpenXR framework and implementing a direct input-mapping paradigm, the system bypassed raycasting latency to achieve a highly responsive ~50 ms control loop. The use of mDNS solved dynamic IP assignment issues, while the physically decoupled ESP32-S3 architecture effectively isolated deterministic motor actuation from interrupt-heavy video streaming. Furthermore, the LLM-driven exploration framework provided a viable pathway for natural-language navigation, maintaining physical safety by restricting the AI to high-level semantic decisions. Democratizing this low-cost architecture opens pathways for critical civilian applications, such as teleoperated hospital wheelchairs, shifting these capabilities out of defense-dominated sectors. Industrially, our findings underscore that crowded 2.4 GHz spectrums are no longer viable for real-time robotics, requiring facilities to adopt dedicated 5 GHz or private 5G infrastructure to guarantee determinism. Future iterations will resolve current 2.4 GHz and thermal constraints by migrating to WebRTC for sub-100 ms video streaming, integrating a 2D LiDAR sensor for holographic SLAM overlays, and implementing bidirectional FPV telemetry alongside edge-deployable vision-language models for direct speech-based control.

## AFFILIATIONS AND AUTHOR DETAILS

### Undergraduate Authors

**Umar Ali Khan** – Department of Electronics Engineering, Aligarh Muslim University, Aligarh 202002, India;  0009-0001-1020-2509  
Email: samkhan705279@gmail.com

**Ahmad Ilyas** – Department of Electronics Engineering, Aligarh Muslim University, Aligarh 202002, India;  0009-0007-6752-6639  
Email: ahmad7.ilyas@gmail.com

### Corresponding Author

**Ekram Khan** – Research Mentor, Professor, Department of Electronics Engineering, Aligarh Muslim University, Aligarh 202002, India;  0000-0001-8983-7584  
Email: ekhan.el@amu.ac.in

## ACKNOWLEDGEMENTS

The authors gratefully acknowledge the guidance of Professor Ekram Khan of the Department of Electronics Engineering, Zakir Husain College of Engineering & Technology, Aligarh Muslim

University, throughout the course of this project. The authors also acknowledge the support and 5G XR technology facilities provided by the 5G Lab, Department of Electronics Engineering. No external funding was received for this study.

## REFERENCES

- (1) Aguilar, I. A., Suomalainen, M., LaValle, S. M., Ojala, T. & Riecke, B. E. From “skype on wheels” to embodied telepresence: a holistic approach to improving the user experience of telepresence robots. *Virtual Real.* **29**, 161 (2025).
- (2) Aijaz, A., Dohler, M., Aghvami, A. H., Friderikos, V. & Frodigh, M. Realizing the Tactile Internet: Haptic Communications over Next Generation 5G Cellular Networks. *IEEE Wirel. Commun.* **24**, 82–89 (2017).
- (3) Chen, J. Y. C., Haas, E. C. & Barnes, M. J. Human Performance Issues and User Interface Design for Teleoperated Robots. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **37**, 1231–1245 (2007).
- (4) Maier, A., Sharp, A. & Vagapov, Y. Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. in *2017 Internet Technologies and Applications (ITA)* 143–148 (IEEE, Wrexham, 2017). doi:10.1109/ITECHA.2017.8101926.
- (5) Zhao, Z., Yu, L., Jing, K. & Yang, N. XRoboToolkit: A Cross-Platform Framework for Robot Teleoperation. Preprint at <https://doi.org/10.48550/ARXIV.2508.00097> (2025).
- (6) Unity Technologies. XR Interaction Toolkit Documentation. (2023).
- (7) Huang, W., Abbeel, P., Pathak, D. & Mordatch, I. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. Preprint at <https://doi.org/10.48550/ARXIV.2201.07207> (2022).
- (8) Ahn, M. *et al.* Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. Preprint at <https://doi.org/10.48550/ARXIV.2204.01691> (2022).
- (9) Driess, D. *et al.* PaLM-E: An Embodied Multimodal Language Model. Preprint at <https://doi.org/10.48550/ARXIV.2303.03378> (2023).
- (10) Raymundo Belleza, R. & De Freitas Pignaton, E. Performance study of real-time operating systems for internet of things devices. *IET Softw.* **12**, 176–182 (2018).
- (11) Hernández, J. C. O., Almeida, D. I. R., Carrillo, J. A. R. & Carmona, R. R. Mitigating Heading Deviations in Mecanum-Wheeled Mobile Robots Using Active Disturbance Rejection Control: A Practical Alternative to PID. *IEEE Access* **13**, 213038–213051 (2025).
- (12) Taheri, H., Qiao, B. & Ghaeminezhad, N. Kinematic Model of a Four Mecanum Wheeled Mobile Robot. *Int. J. Comput. Appl.* **113**, 6–9 (2015).
- (13) Jocher, G. & Qiu, J. Ultralytics YOLO11. (2024).
- (14) Olson, E. AprilTag: A robust and flexible visual fiducial system. in *2011 IEEE International Conference on Robotics and Automation* 3400–3407 (IEEE, Shanghai, China, 2011). doi:10.1109/ICRA.2011.5979561.
- (15) Campos, C., Elvira, R., Rodriguez, J. J. G., M. Montiel, J. M. & D. Tardos, J. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM. *IEEE Trans. Robot.* **37**, 1874–1890 (2021).
- (16) Bai, S. *et al.* Qwen2.5-VL Technical Report. Preprint at <https://doi.org/10.48550/ARXIV.2502.13923> (2025).