A Framework for Low-Latency, LLM-Driven Multimodal Interaction on the Pepper Robot

Erich Studerus
erich.studerus@fhnw.ch
University of Applied Sciences and Arts Northwestern Switzerland
Institute for Information Systems
Basel Switzerland

Vivienne Jia Zhong
viviennejia.zhong@fhnw.ch
University of Applied Sciences and Arts Northwestern Switzerland
Institute for Information Systems
Basel Switzerland

Stephan Vonschallen
stephan.vonschallen@zhaw.ch
Zurich University of Applied Sciences
School of Management and Law
Winterthur Switzerland

Abstract

Despite recent advances in integrating Large Language Models (LLMs) into social robotics, two weaknesses persist. First, existing implementations on platforms like Pepper often rely on cascaded Speech-to-Text (STT)→LLM→Text-to-Speech (TTS) pipelines, resulting in high latency and the loss of paralinguistic information. Second, most implementations fail to fully leverage the LLM's capabilities for multimodal perception and agentic control. We present an open-source Android framework for the Pepper robot that addresses these limitations through two key innovations. First, we integrate end-to-end Speech-to-Speech (S2S) models to achieve low-latency interaction while preserving paralinguistic cues and enabling adaptive intonation. Second, we implement extensive Function Calling capabilities that elevate the LLM to an agentic planner, orchestrating robot actions (navigation, gaze control, tablet interaction) and integrating diverse multimodal feedback (vision, touch, system state). The framework runs on the robot's tablet but can also be built to run on regular Android smartphones or tablets, decoupling development from robot hardware. This work provides the HRI community with a practical, extensible platform for exploring advanced LLM-driven embodied interaction.

CCS Concepts

• **Computer systems organization** → **Embedded and cyber-physical systems**; • **Human-centered computing** → **Interaction design**.

Keywords

Human-Robot Interaction, Large Language Models, Function Calling, Multimodality, Pepper Robot, Open-Source Framework

ACM Reference Format:
Erich Studerus, Vivienne Jia Zhong, and Stephan Vonschallen. 2026. A Framework for Low-Latency, LLM-Driven Multimodal Interaction on the Pepper Robot. In *Proceedings of the 21st ACM/IEEE International Conference on Human-Robot Interaction (HRI '26), March 16–19, 2026, Edinburgh, Scotland, UK*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3757279.3788808

HRI '26, Edinburgh, Scotland, UK

ACM ISBN 979-8-4007-2128-1/2026/03
https://doi.org/10.1145/3757279.3788808

1 Background and Summary

Recent advancements in Large Language Models (LLMs) have fundamentally transformed Human-Robot Interaction (HRI) [13], enabling open-domain dialogue that replaces scripted systems with natural, context-aware conversation [21, 33]. Several recent studies have integrated LLMs into social robots like Pepper [2–6, 15, 21, 27]. While early implementations primarily used LLMs for dialogue generation, more recent work explores their potential for task planning, gesture generation, and multimodal behavior control [22, 23].

However, two major barriers prevent these implementations from fully realizing the LLM's potential for multimodal, embodied interaction. First, most existing implementations rely on cascaded pipelines that sequentially process speech through separate Speech-to-Text (STT), LLM response generation, and Text-to-Speech (TTS) stages. This architecture introduces two critical problems: Each processing stage adds incremental delays, resulting in accumulated latency that prevents fluid real-time dialogue [7, 24]. Consequently, recent LLM implementations on platforms like the Pepper robot report system response times ranging from 3.84 to 8.96 seconds [10, 21], far exceeding the 1–2 second threshold established as acceptable for natural conversation flow [11, 20, 26, 30]. Additionally, the conversion of audio to text and back to audio discards paralinguistic information, such as prosody, intonation, and emotional cues, that are crucial for natural, empathetic human-robot interaction [7, 21].

Second, despite recent progress, most implementations still fail to fully leverage the LLM's capabilities for multimodal perception and agentic control. Achieving embodied AI requires the LLM to convert abstract natural language instructions into concrete, executable actions and integrate diverse sensor feedback in real-time [1, 12, 23]. This requires mechanisms like Function Calling (also known as "Tool Use") [3, 22, 23], where the LLM autonomously selects and invokes predefined functions with appropriate parameters. Through Function Calling, the LLM can orchestrate robot actions (navigation, gaze control, tablet interaction) and actively direct its perception by triggering targeted sensor data acquisition (e.g., camera images) [1, 25]. Additionally, the system must integrate asynchronous, event-driven inputs from haptic sensors [29] and device interfaces that provide contextual information about physical interactions and system state. The LLM's native multimodal capabilities then process these diverse data streams to achieve embodied grounding in the physical environment [18, 19].

This submission presents an open-source Android framework for the Pepper robot that overcomes these barriers through two key innovations (see Figure 1): (1) Integration of end-to-end S2S

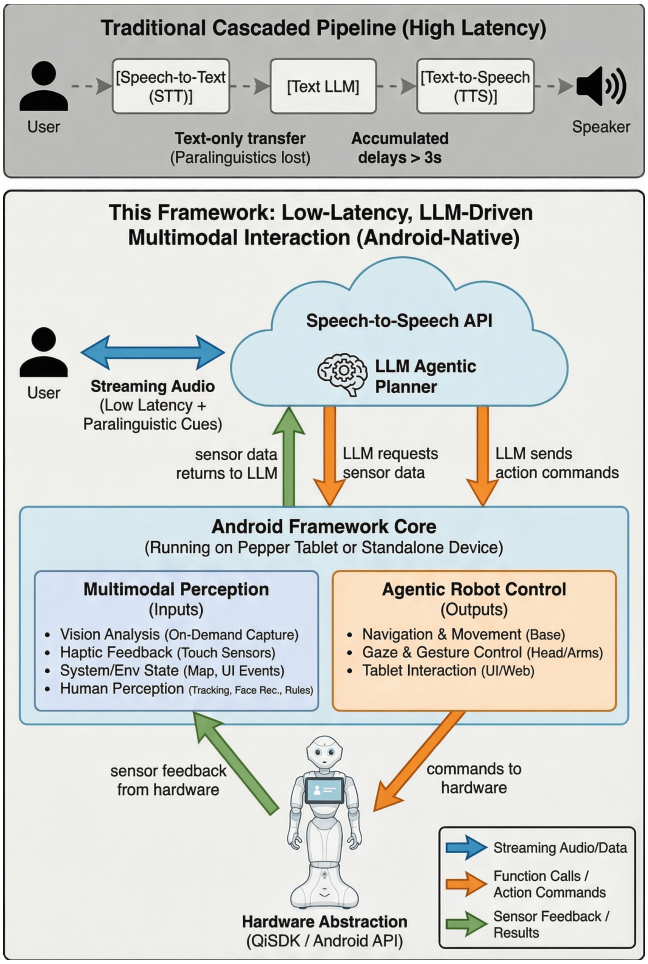

Figure 1: System architecture comparison showing traditional cascaded pipelines versus the proposed low-latency, multimodal framework.

models [8, 9, 18, 19] to achieve low-latency streaming interaction while preserving paralinguistic cues, and (2) extensive Function Calling capabilities that enable the LLM to orchestrate robot actions (navigation, gaze control, tablet interaction) and integrate diverse multimodal feedback (vision analysis, haptic feedback, system state). The framework provides a modular, easily deployable reference implementation that facilitates wider adoption of modern LLM features in embodied platforms. Its design is informed by insights from a prior field study investigating communication dynamics between older adults and an LLM-powered robot [34].

2 Purpose

The primary purpose of this framework is to provide the HRI community with a reusable, open-source framework that addresses the latency and limited agentic control challenges outlined above. It establishes a practical blueprint for using an LLM as an agentic orchestrator for multimodal robot behavior—an architectural approach that can be adapted to other robotic platforms. The framework is particularly valuable given the practical challenges facing Pepper researchers: following Aldebaran's court-ordered liquidation on June 2, 2025 [14], official support for Pepper has become uncertain, and the platform's Android/Java ecosystem and aging hardware (Android 6.0, API level 23) [28] pose non-trivial integration barriers. Furthermore, a key purpose of this contribution is to simplify deployment for HRI research. The framework is designed to run both on the Pepper robot and on standard Android smartphones or tablets, decoupling the development cycle from physical robot hardware. This enables researchers to develop, test, and prototype interaction logic without requiring access to expensive robotic platforms.

3 Characteristics

3.1 Architectural Advantage

Android-Native Execution: The entire application runs natively on Android devices—either on the Pepper robot's onboard tablet (controlling the robot's physical body via QiSDK) or on standard Android smartphones and tablets. This self-contained architecture eliminates the need for external computers or bridging software common in previous LLM-HRI integrations [2, 4, 5], simplifying deployment and use.

Dual Build-Flavor System: The framework can be deployed either on the Pepper robot or on standard Android smartphones and tablets through two build configurations. Robot-specific functions adapt automatically: The `analyze_vision` tool, for instance, accesses the robot's camera system (via QiSDK) when running on Pepper, but uses the device's front camera when running on a smartphone or tablet. Navigation and movement commands control the physical robot or are simulated accordingly. All other tools, including internet search, weather queries, games, and conversational features, operate identically regardless of the deployment platform.

3.2 Core Interaction Engine: Speech-to-Speech Models

The framework's low-latency capabilities are powered by end-to-end S2S models from multiple providers (OpenAI [18, 19], Azure OpenAI, x.ai [32], Google [9]). A central `TurnManager` orchestrates the robot's state (listening, thinking, speaking). Key advantages include:

End-to-End Speech-to-Speech (S2S) Model: By processing audio directly, the system bypasses the cumulative latency of separate STT/TTS stages. Furthermore, unlike cascaded systems, where information is lost during text conversion, the S2S model can process paralinguistic cues like user intonation. Consequently, its generated speech is not merely text being read aloud; the model adapts its own intonation, tone, and pacing to match the content. While traditional TTS pipelines can also achieve prosody control through explicit SSML markup, this requires complex prompt engineering and rule definitions. The S2S model, on the other hand, generates contextually appropriate prosody intrinsically.

Duplex Communication: While S2S APIs are designed to process user input while simultaneously streaming back responses, our implementation adapts to hardware constraints. Due to the lack of

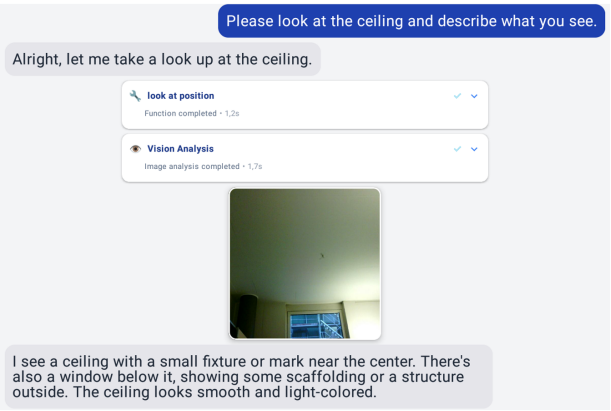

Figure 2: User query triggers multi-step tool sequencing with expandable UI cards showing execution details.

echo cancellation on the Pepper robot, the microphone is intentionally muted during speech output. To compensate, a key usability feature is the ability for the user to tap the status capsule (the interactive state indicator at the bottom of the screen) to instantly interrupt the robot's speech, ensuring a user-controlled conversational experience. The system's duplex nature is still leveraged to process asynchronous, non-auditory inputs (e.g., touch sensor data, tool results) during speech output.

Advanced Language and Voice Flexibility: The models support multiple built-in voices and can respond in a wide range of languages with on-the-fly language switching; a user can change their spoken language mid-conversation, and the model will understand and adapt without requiring configuration changes.

3.3 LLM as an Agentic Planner: Tool Library

The framework elevates the LLM to an agentic planner through a rich library of function calls. The LLM can autonomously sequence these tools to fulfill complex requests. For instance, the query "What do you see at the ceiling?" triggers a multi-step sequence (see Figure 2): First, the LLM invokes `look_at_position` with calculated 3D coordinates to direct the robot's gaze upward. Then, it calls `analyze_vision`, which captures an image with the robot's front camera and sends it to the S2S backend. The model's native multimodal capabilities process the image directly, enabling it to describe what it sees. Available tools include navigation (movement, floor mapping), multimodal interaction (vision analysis, gaze control), information retrieval (web search, weather, date/time), and interactive entertainment (quizzes, tic-tac-toe, memory game, jokes, drawing game, melody playback, YouTube videos). All tool executions are rendered in the User Interface (UI) as expandable function cards for full transparency.

3.4 Multimodal Perception and Context

The framework integrates various sensory inputs to provide the LLM with contextual awareness of its environment.

Visual Perception: Rather than continuous video streaming, the system uses an event-driven architecture where the `analyze_vision` function captures images on demand—triggered by user requests, the model's reasoning, or physical events like obstacle detection. The function sends a camera image as input to the S2S model, which natively supports multimodal input, enabling direct analysis of visual data without requiring separate vision APIs.

Haptic Perception: The robot responds to physical interaction through its various touch sensors on its head, hands, and bumpers. A touch event sends a contextual message (e.g., `[User touched my right hand]`) directly to the LLM, allowing it to generate a socially appropriate and context-aware response to physical contact.

Human Perception: A custom head-based perception system runs locally on Pepper's head computer, providing stable person tracking and face recognition via WebSocket streaming. This replaces the unreliable QiSDK PeoplePerception metadata with a dedicated YuNet/SFace-based solution [31] that processes biometric data entirely on-device. An event-based rule system allows researchers to define custom rules that automatically trigger AI responses based on perception events (e.g., person recognized, person approached within interaction distance). Rules can be configured with conditions, cooldowns, and different action types (e.g., trigger an immediate response, update the context of the LLM without requesting a response).

System and Environmental State: The LLM is grounded in its physical context by receiving information about its state and environment. This includes map locations, game interactions, hardware status (e.g., an open charging flap), and error or success messages from tool calls. For example, a movement blocked by an obstacle automatically triggers an `analyze_vision` call, allowing the LLM to reason about the cause of the failure.

3.5 High Configurability and User Control

An in-app settings panel allows for runtime adjustments of the entire interaction pipeline. Users can switch between S2S models across four providers: OpenAI (gpt-realtime, gpt-realtime-mini, gpt-4o-realtime-preview, gpt-4o-mini-realtime-preview), Azure OpenAI, x.ai (Grok Voice Agent API [32]), and Google (Gemini Live API [9]). Azure OpenAI additionally provides network-level isolation and customer-managed encryption keys [16] for enterprise compliance requirements. Users can also choose between two audio input modes: Direct Audio or STT (speech is first transcribed to text via Azure Speech Services and then sent to the S2S backend). The latter can perform better for some regional dialects, such as Swiss German, and provides confidence scores [17] that inform the LLM when transcription quality may be uncertain.

3.6 Dependencies, Installation, and Usage

The framework is built on a modern Android toolchain (API level 35, Java 17) and notably works without the deprecated Pepper SDK plugin, ensuring compatibility with the latest Android Studio versions and future-proofing the codebase. Key dependencies include the SoftBank Robotics QiSDK and OkHttp, all managed via Gradle.

The setup process is straightforward and requires minimal configuration. Only a single API key from one of the supported providers is needed to enable the core conversational features. All other API keys are optional and enable supplementary services: Tavily for internet search, OpenWeatherMap for weather

information, Azure Speech for alternative speech recognition, and YouTube Data API for video playback. A comprehensive guide in the README.md details the setup process.

3.7 Availability

The framework is provided as an open-source artifact under the MIT License at https://github.com/studerus/pepper-android-realtime-chat. The repository includes comprehensive documentation and setup instructions.

4 Code/Software

4.1 Code Structure and Key Components

The software is an Android application written entirely in Kotlin, built with Gradle, and architected around a build flavor system (as described in Section 3.1) to separate hardware-dependent code from the core application logic. The codebase leverages modern Kotlin features including coroutines for structured concurrency, data classes, and Hilt for dependency injection. This results in a highly modular and extensible structure divided into three primary source sets:

(1) `app/src/main/` (Shared Core): Contains most code shared between `pepper` and `standalone` flavors. Key components include `ChatActivity.kt` (UI, lifecycle, and service orchestration), `TurnManager.kt` (conversational state machine for listening, thinking, speaking), `RealtimeSessionManager.kt` (WebSocket communication with S2S backends), `ToolRegistry.kt` (agentic system core for registering, validating, and executing function calls), and the `robot/` sub-package with abstraction interfaces (`RobotController`, `RobotLifecycleBridge`) that define the contract for robot interaction, allowing the main module to remain hardware-agnostic.
(2) `app/src/pepper/` (Pepper Flavor): Contains Pepper-specific implementations using QiSDK. For example, `MovementController.kt` translates abstract movement commands into QiSDK actions.
(3) `app/src/standalone/` (Standalone Flavor): Provides implementations for standard Android devices—either stubs (e.g., `MovementController.kt` logs actions) or adapted implementations (e.g., `VisionService.kt` uses Camera2 API).

4.2 Configuration and Data Formats

Build configuration is managed via `app/build.gradle`, which injects API keys from `local.properties` (excluded from version control) into `BuildConfig` at compile time. A template file (`local.properties.example`) guides key setup.

4.3 Code Maintenance and Supplemental Documentation

Contributions via GitHub Issues for bug reports and feature requests, as well as Pull Requests for enhancements, are welcome and will be reviewed. The primary source of documentation is the comprehensive README.md file, which provides step-by-step installation instructions, detailed feature explanations, API key setup guidance, and troubleshooting tips. Additional documentation includes inline code comments and architectural diagrams.

5 Usage Notes

5.1 Utility and Extensibility for HRI Research

The framework is designed for extensibility and adaptation. Researchers can extend its capabilities by implementing new tools that conform to the Tool interface, modify robot personalities through the system prompt, or integrate additional sensor modalities. While implemented on Pepper's Android platform (QiSDK), the architectural principles, particularly the abstraction layer separating core logic from hardware-specific implementations, are transferable to other Android-based humanoid robots. Beyond Android, the conceptual approach (S2S model integration, Function Calling for robot control, multimodal grounding) provides a blueprint adaptable to other robotic platforms. For advanced use cases on Pepper, the framework establishes an SSH connection at startup, enabling access to low-level NAOqi functions not exposed by QiSDK. This connection is also used to launch the head-based people perception system on the robot's head computer.

5.2 Ethical Considerations and Responsible Use

Our implementation may expose researchers to data security issues. In particular, the current method of storing API keys in `BuildConfig` is suitable for development but is not secure for production deployments. Furthermore, user data, including audio and images, is transmitted to third-party cloud services like OpenAI and Azure when their respective features are active. A detailed summary of these considerations, including instructions on how to disable specific data-transmitting features, is provided in the "Security & Privacy" section of the README.md file.

5.3 Limitations and Outlook

Several limitations should be acknowledged: First, Pepper lacks articulated lips, so lip-syncing—relevant for robots like Furhat—is not addressed. Second, while the architectural principles are designed to be transferable, the current implementation is validated only on the Pepper platform. Third, the reliance on cloud-based APIs introduces dependency on external services and potential latency variations based on network conditions. Finally, developers using speech-to-speech models have reduced control over specific phrasings compared to text-based pipelines.

Despite these limitations, this framework represents a significant step towards fluid, low-latency, and agentic human-robot interaction. By open-sourcing this tool, we aim to accelerate the adoption of multimodal, agentic LLM capabilities in social robotics, encouraging the community to extend these principles to new platforms and interaction scenarios.

References

[1] Giulio Antonio Abbo and Tony Belpaeme. 2025. I Was Blind but Now I See: Implementing Vision-Enabled Dialogue in Social Robots. In *2025 20th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, Melbourne, Australia, 1176–1180. doi:10.1109/HRI61500.2025.10973830
[2] Tahsin Tariq Banna, Sejuti Rahman, and Mohammad Tareq. 2025. Beyond Words: Integrating Personality Traits and Context-Driven Gestures in Human-Robot Interactions. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*. 242–251.

[3] Luccas Rojas Becerra, Juan Andrés Romero Colmenares, and Rubén Manrique. 2024. Improving Autonomy and Natural Interaction of Pepper Robot via Large Language Models. doi:10.21203/rs.3.rs-3997840/v1

[4] Francesca Bertacchini, Francesco Demarco, Carmelo Scuro, Pietro Pantano, and Eleonora Bilotta. 2023. A social robot connected with chatGPT to improve cognitive functioning in ASD subjects. *Frontiers in Psychology* 14 (2023), 1232177. doi:10.3389/fpsyg.2023.1232177

[5] Erik Billing, Julia Rosén, and Maurice Lamb. 2023. Language Models for Human-Robot Interaction. In *Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, Stockholm, Sweden, 905–906. doi:10.1145/3568294.3580040

[6] Xiaohui Chen, Katherine Luo, Trevor Gee, and Mahla Nejati. 2024. Does ChatGPT and Whisper Make Humanoid Robots More Relatable? doi:10.48550/arXiv.2402.07095

[7] Yuhao Du, Qianwei Huang, Guo Zhu, Zhanchen Dai, Shunian Chen, Qiming Zhu, Le Pan, Minghao Chen, Yuhao Zhang, Li Zhou, Benyou Wang, and Haizhou Li. 2025. MTalk-Bench: Evaluating Speech-to-Speech Models in Multi-Turn Dialogues via Arena-style and Rubrics Protocols. doi:10.48550/arXiv.2508.18240

[8] Mengxue Fu, Zhonghao Shi, Minyu Huang, Siqi Liu, Mina Kian, Yirui Song, and Maja J. Matarić. 2026. Personalized Socially Assistive Robots with End-to-End Speech-Language Models for Well-Being Support. In *Social Robotics + AI (ICSR+AI 2025) (Lecture Notes in Computer Science, Vol. 16132)*. Springer, 192–206. doi:10.1007/978-981-95-2382-5_14

[9] Google. 2025. *Get started with Live API*. https://ai.google.dev/gemini-api/docs/live Google AI for Developers.

[10] Leon Hanschmann, Ulrich Gnewuch, Carolin Kaiser, and Alexander Mädche. 2025. Designing Adaptive LLM-Based Social Robots for Retail Sales Consultations. In *International Conference on Information Systems*.

[11] Koji Inoue, Yuki Okafuji, Jun Baba, Yoshiki Ohira, Katsuya Hyodo, and Tatsuya Kawahara. 2025. A Noise-Robust Turn-Taking System for Real-World Dialogue Robots: A Field Experiment. doi:10.48550/arXiv.2503.06241

[12] Ruben Janssens and Tony Belpaeme. 2025. Towards Multimodal Social Conversations with Robots: Using Vision-Language Models. doi:10.48550/arXiv.2507.19196

[13] Yeseung Kim, Dohyun Kim, Jieun Choi, Jisang Park, Nayoung Oh, and Daehyung Park. 2024. A Survey on Integration of Large Language Models with Intelligent Robots. *Intelligent Service Robotics* 17, 5 (2024), 1091–1107. doi:10.1007/s11370-024-00550-5

[14] Le Monde. 2025. Aldebaran, la vedette de la robotique française, placée en liquidation judiciaire. *Le Monde* (2025). https://www.lemonde.fr/economie/article/2025/06/02/aldebaran-la-vedette-de-la-robotique-francaise-placee-en-liquidation-judiciaire_6610246_3234.html

[15] JongYoon Lim, Inkyu Sa, Bruce MacDonald, and Ho Seok Ahn. 2023. A Sign Language Recognition System with Pepper, Lightweight-Transformer, and LLM. doi:10.48550/arXiv.2309.16898

[16] Microsoft. 2024. *Data, privacy, and security for Azure OpenAI Service*. https://learn.microsoft.com/en-us/legal/cognitive-services/openai/data-privacy Microsoft Learn.

[17] Microsoft. 2025. *Get speech recognition results*. https://learn.microsoft.com/en-us/azure/ai-services/speech-service/get-speech-recognition-results Microsoft Learn.

[18] OpenAI. 2024. *Introducing the Realtime API*. https://openai.com/index/introducing-the-realtime-api/

[19] OpenAI. 2025. *Introducing gpt-realtime and Realtime API updates for production voice agents*. https://openai.com/index/introducing-gpt-realtime/

[20] Hannah Pelikan and Emily Hofstetter. 2023. Managing Delays in Human-Robot Interaction. *ACM Transactions on Computer-Human Interaction* 30, 4 (2023), 1–42. doi:10.1145/3569890

[21] Maria Pinto-Bernal, Matthijs Biondina, and Tony Belpaeme. 2025. Designing Social Robots with LLMs for Engaging Human Interaction. *Applied Sciences* 15, 11 (2025), 6377. doi:10.3390/app15116377

[22] Emmanuel K. Raptis, Athanasios Ch. Kapoutsis, and Elias B. Kosmatopoulos. 2025. Agentic LLM-based robotic systems for real-world applications: a review on their agenticness and ethics. *Frontiers in Robotics and AI* 12 (2025), 1605405. doi:10.3389/frobt.2025.1605405

[23] Sahar Salimpour, Lei Fu, Farhad Keramat, Leonardo Militano, Giovanni Toffetti, Harry Edelman, and Jorge Peña Queralta. 2025. Towards Embodied Agentic AI: Review and Classification of LLM- and VLM-Driven Robot Autonomy and Interaction. doi:10.48550/arXiv.2508.05294

[24] Mohammad Sarim, Saim Shakeel, Laeeba Javed, Jamaluddin, and Mohammad Nadeem. 2025. Direct Speech to Speech Translation: A Review. doi:10.48550/arXiv.2503.04799

[25] Zhonghao Shi, Enyu Zhao, Nathaniel Dennler, Jingzhen Wang, Xinyang Xu, Kaleen Shrestha, Mengxue Fu, Daniel Seita, and Maja Matarić. 2025. HRIBench: Benchmarking Vision-Language Models for Real-Time Human Perception in Human-Robot Interaction. doi:10.48550/arXiv.2506.20566

[26] Toshiyuki Shiwa, Takayuki Kanda, Michita Imai, Hiroshi Ishiguro, and Norihiro Hagita. 2008. How Quickly Should Communication Robots Respond?. In *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction*. 153–160. doi:10.1145/1349822.1349843

[27] Thomas Sievers and Nele Russwinkel. 2025. Retrieving Memory Content from a Cognitive Architecture by Impressions from Language Models for Use in a Social Robot. *Applied Sciences* 15, 10 (2025), 5778. doi:10.3390/app15105778

[28] SoftBank Robotics. 2019. *QiSDK Documentation: Creating a Robot Application*. https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch1_gettingstarted/starting_project.html Pepper SDK requires minimum API level 23 (Android 6.0).

[29] Christiana Tsirka, Anna-Maria Velentza, and Nikolaos Fachantidis. 2025. Touch in Human Social Robot Interaction: Systematic Literature Review with PRISMA Method. *International Journal of Social Robotics* 17 (Nov. 2025), 2803–2825. doi:10.1007/s12369-025-01319-1

[30] Ya-Ling Wang and Chi-Wen Lo. 2025. The effects of response time on older and young adults' interaction experience with Chatbot. *BMC Psychology* 13, 1 (2025), 150. doi:10.1186/s40359-025-02459-9

[31] Wei Wu, Hanyang Peng, and Shiqi Yu. 2023. YuNet: A Tiny Millisecond-level Face Detector. *Machine Intelligence Research* 20, 5 (2023), 656–665. doi:10.1007/s11633-023-1423-y

[32] xAI. 2025. *Grok Voice Agent API*. https://x.ai/news/grok-voice-agent-api

[33] Ceng Zhang, Junxin Chen, Jiatong Li, Yanhong Peng, and Zebing Mao. 2023. Large language models for human–robot interaction: A review. *Biomimetic Intelligence and Robotics* 3, 4 (2023), 100131. doi:10.1016/j.birob.2023.100131

[34] Vivienne Jia Zhong, Erich Studerus, and Stephan Vonschallen. 2025. Integrating LLM into a Socially Assistive Robot for Social Dialogue: An Exploratory Study in a Nursing Home. In *2025 34th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE. doi:10.1109/RO-MAN63969.2025.11217629

Received 2025-10-07; accepted 2025-12-09