Flexible LLM-Based Voice Assistance for Mobile Robots

Söhnke Benedikt Fischedick
Technische Universität Ilmenau
Ilmenau Germany
soehnke.fischedick@tu-ilmenau.de

Robin Schmidt
Technische Universität Ilmenau
Ilmenau Germany
robin.schmidt@tu-ilmenau.de

Benedict Stephan
Technische Universität Ilmenau
Ilmenau Germany
benedict.stephan@tu-ilmenau.de

Horst-Michael Gross
Technische Universität Ilmenau
Ilmenau Germany
horst-michael.gross@tu-ilmenau.de

Abstract

Voice-based interaction offers an intuitive way for untrained users to control mobile robots, but existing speech interfaces often rely on intent maps or robot-specific pipelines that are difficult to transfer across robots, backends, and applications. Recent multimodal large language models (LLMs) can process audio and produce structured function calls, enabling a more flexible form of voice interaction.

This late-breaking report proposes a vendor-independent integration pattern (cloud, edge server, or local) that exposes robot capabilities as Model Context Protocol (MCP) tools and maps them to existing middleware interfaces such as remote procedure calls (RPCs). Continuous sensor streams remain in the middleware and are accessed through a snapshot mechanism that returns the most recent buffered value on demand. We demonstrate the approach on a mobile co-presence robot using a lightweight audio pipeline built around wake word detection (WWD), voice activity detection (VAD), multimodal LLM inference, and text-to-speech (TTS). MCP tools trigger capabilities such as navigation, communication, and projector control. The architecture provides a general pattern for robots and middlewares, enabling flexible voice interaction without rewriting intent logic.

CCS Concepts

• **Human-centered computing** → **Human computer interaction (HCI)**; • **Computing methodologies** → **Natural language processing**; • **Software and its engineering** → *Middleware*.

Keywords

voice assistance, large language models, robotics, middleware

ACM Reference Format:
Söhnke Benedikt Fischedick, Robin Schmidt, Benedict Stephan, and Horst-Michael Gross. 2026. Flexible LLM-Based Voice Assistance for Mobile Robots. In *Companion Proceedings of the 21st ACM/IEEE International Conference on Human-Robot Interaction (HRI Companion '26), March 16–19, 2026, Edinburgh, Scotland UK*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3776734.3794452

HRI Companion '26, Edinburgh, Scotland UK

ACM ISBN 979-8-4007-2321-6/2026/03
https://doi.org/10.1145/3776734.3794452

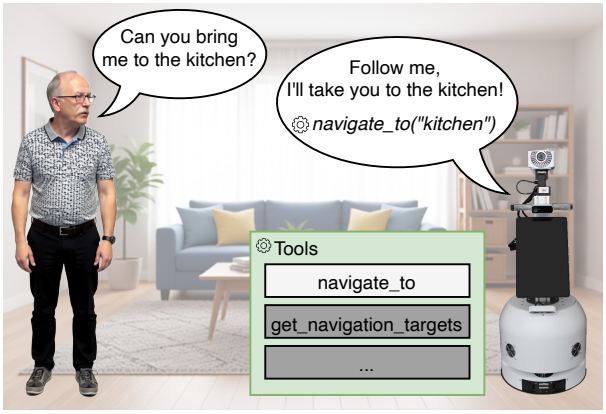

Figure 1: The proposed approach enables voice assistance for mobile robots with integrated function calling. In this example, the user's request to go to the kitchen is processed by an LLM, which triggers a corresponding tool call that the middleware executes. Because capabilities are exposed through a standardized tool interface, the same interaction pattern can be applied across different tools and robot platforms.

1 Introduction

Speech-based interaction allows untrained users to communicate with robots naturally and without relying on specialized interfaces. Furthermore, speech also supports hands-free interaction, which is particularly useful in shared spaces such as homes or care facilities. In our ongoing research project CO-HUMANICS [3, 5, 8], a mobile robot with telepresence features is deployed in an older adult's home and can be accessed remotely by family and friends, where simple and natural voice interaction is essential for lowering barriers for everyday use. For example, as shown in Fig. 1, a user might instruct a mobile robot to drive to a specified location. Many current HRI systems use cascaded pipelines consisting of speech-to-text (STT), a text-based large language model (LLM) or natural language understanding (NLU), and text-to-speech (TTS) components. These setups can perform well, but they often depend on robot-specific intent logic or prompt engineering, which limits reuse across platforms and tasks [4, 6, 20, 28, 29].

Modern multimodal LLMs extend this pipeline by supporting audio and vision inputs and by producing function calls. Such models can reduce the need for custom NLU components, but integrating them into robots presents two challenges. First, integration should be flexible, as it must work with different middlewares, different

robots, and different LLM backends, including cloud, edge, and fully local deployments. Second, audio-based interaction should remain easy to deploy even when some robot-specific audio handling is required, which calls for keeping these components decoupled from the LLM backend. Although end-to-end audio-to-audio LLMs are emerging, they remain limited, and early models struggled to combine audio understanding and reliable tool calling [10, 21, 25, 27].

Existing LLM integrations fall into two categories. LLM-centric systems have the model produce low-abstraction action steps. These steps are executed directly on the robot and therefore are not checked or coordinated by the middleware's planning and safety modules. Middleware-centric systems restrict the LLM to selecting high-level capabilities, while the middleware performs planning, control, and safety handling. We adopt the middleware-centric design as it keeps safety and control in the established middleware and treats the LLM purely as a reasoning component.

The contribution of this report is a middleware-aligned integration pattern that exposes robot capabilities as Model Context Protocol (MCP) tools and maps them to services, actions, or remote procedure calls (RPCs) in the underlying middleware. Continuous sensor streams remain within the middleware and are accessed through a snapshot mechanism that provides explicit, on-demand retrieval of the latest sensor value. This reduces coupling between the LLM and the robot software. It also allows different LLM backends to be substituted without modifying the capability surface. In our practical application, a lightweight audio pipeline combines WWD, VAD, LLM inference, and TTS, but the integration pattern is independent of any specific audio configuration.

Section 2 of this report reviews related work on robotic middleware, multimodal LLMs, and function calling. Section 3 presents the general architecture, the tool schemas, and the snapshot mechanism. Section 4 describes a practical deployment, including the audio pipeline. Section 5 discusses HRI implications.

2 Related Work

Modern HRI systems combine speech interfaces, robot middleware, and increasingly powerful LLM backends [4, 9, 22]. In this section, we summarize the prior work most relevant to our integration approach regarding robotic middlewares, recent advances in multimodal LLMs, and function-calling interfaces.

2.1 Robotic Middleware and Capability Interfaces

Robotic middleware provides the communication and coordination backbone of many mobile robots. Typically the middleware is based on *nodes*, which implement different functionalities, share data over *topics* using a publish-subscribe pattern and use *RPCs* to trigger actions or use implemented methods of other nodes. Examples for these approaches are ROS/ROS2 [14, 19] and MIRA [7], which may use different terminology but are based on the same principles.

Because middleware already defines how robot capabilities and data flows are structured, LLM-based interaction should align with these interfaces rather than bypass them in a middleware-centric approach. Prior work has shown that LLMs can operate on top of ROS to guide navigation or manipulation, but many such integrations rely on custom bridges or robot-specific intent logic, limiting generality and portability. [15, 20]

Although names for communication interfaces differ across frameworks, these interfaces serve the same purpose: asynchronous sensing and synchronous actuation. This makes them ideal candidates for exposing them to LLMs through function calling.

Table 1: Overview of multimodal LLMs, including their modality and native function calling support.

Model	Input Modalities	Output Modalities	Function Calling
GPT-5[17]	Text, Image	Text	✓
Claude Sonnet 4.5[2]	Text, Image	Text	✓
Gemma3n[23]	Text, Audio, Image	Text	✗
Voxtral[13]	Text, Audio	Text	✓
Qwen2.5-Omni [26]	Text, Audio, Image, Video	Text, Audio	✗
Qwen3-Omni[27]	Text, Audio, Image, Video	Text, Audio	✓
GPT-4o Audio[16]	Text, Audio	Text, Audio	✓
LongCat-Flash -Omni [24]	Text, Audio, Image, Video	Text, Audio	✗

2.2 Multimodal Large Language Models

LLMs were initially text-only models, and robotics systems integrated them by placing them between STT and TTS components. This cascaded approach remains common in deployed systems [4, 11]. As shown in Tab. 1, recent developments extend text-only LLMs with vision and audio modalities, enabling a single model to process spoken commands and to emit structured function calls. Early audio-capable models often struggled with robustness and reliable function invocation, but newer systems such as multimodal variants of Qwen and GPT have improved audio input handling [16, 27]. Despite this, audio-to-audio LLMs remain experimental in common LLM inference solutions [18] which therefore often support audio only as an input[1,2,3]. The current state in development therefore motivates a modular pipeline that splits audio processing into separate modules for LLM-based audio input handling and speech synthesis via TTS, which can be combined in the future when audio output is supported in the inference solutions. There may be additional modules for WWD and VAD to further control the audio input, e.g., to save processing power and allow explicit activation by a user.

2.3 Function Calling and Application Integration for LLMs

Integrating LLMs with external software typically requires function calling. Early systems relied on ad hoc JSON schemes or custom wrappers that tied the LLM to the surrounding application and were

[1] https://github.com/sgl-project/sglang – accessed on December 2
[2] https://github.com/ggml-org/llama.cpp – accessed on December 2
[3] https://github.com/vllm-project/vllm/ – accessed on December 2

difficult to reuse [4, 22]. The Model Context Protocol (MCP) was introduced as a way to standardize how external context, such as tools or other resources, e.g., external data sources or prompt templates, are exposed to LLM(-Applications) [1]. To achieve this, it proposes a client-server architecture in which context is defined on the servers and retrieved via clients contained in the LLM-Application (MCP host). Tools specifically are functions defined and registered on an MCP server. Once an MCP client (that is connected to an LLM) connects to a server, it may retrieve available tools, including their schema (signature, descriptions, etc.), or call specific tools.

Additionally, the MCP server can also use client-side features, such as requesting LLM generation or user information.However, while these capabilities are part of the protocol, they are not used in our integration and thus not discussed.

This communication happens via JSON-RPC messages. Although this is required by MCP, various SDKs have already implemented the protocol and abstracted it for ease of use. As a result, MCP is a versatile protocol that is independent of programming language and applications (including LLM backends) that enables a plug-and-play-like integration of external context. Although most MCP usage has focused on non-robotic applications, the concept aligns well with robot middleware, where robot capabilities are already exposed as callable units via RPCs.

3 Generic Integration Approach

Our approach connects function-calling LLMs to mobile robots in a way that remains compatible with existing middleware and does not depend on robot-specific intent logic. The central idea is that the middleware continues to handle perception, planning, and actuation, while the LLM is restricted to selecting high-level capabilities exposed as tools via MCP. This keeps the interaction predictable and allows the same integration strategy to be reused across robots, compatible middleware frameworks, and different LLM backends.

Fig. 2 shows the architecture: The robot hosts an MCP server, allowing the LLM to access its capabilities by using an MCP client. This keeps the middleware separated from the LLM. In the following, we describe how tools and continuous sensor data are accessed and examine the backend flexibility of our approach.

3.1 Exposing Robot Capabilities as MCP Tools

Robot capabilities are exposed to the LLM as MCP tools that mirror callable middleware RPCs. Each tool describes a single capability

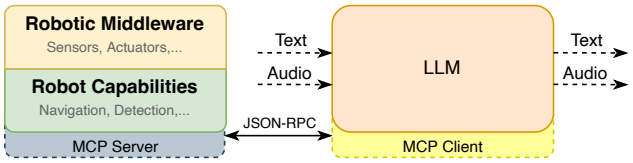

Figure 2: Architecture overview. Robot capabilities provided by the middleware are exposed as MCP tools via an MCP server, which the LLM accesses through an MCP client. This allows the LLM to issue structured tool calls based on audio input for voice-based robot control.

with named parameters and short guidance for the LLM, while continuous data remains on topics inside the middleware. This avoids intent maps and keeps feasibility and safety checks inside the robot stack. As a concrete example, a navigation capability can be exposed as:

```
{
  "name": "navigate_to",
   "description": "Moves the robot to a named location.",
  "input_schema": {
    "type": "object",
    "properties": {
      "target": {
        "type": "string",
        "description": "Name of the waypoint or location."
      }
    },
    "required": ["target"]
  }
}
```

In practice, each MCP tool corresponds to exactly one callable middleware function. The mapping preserves type information, enforces parameter validation, and allows the middleware to reject invalid or unsafe requests. For example, the *navigate_to* schema could expose a navigation tool that accepts a location string that maps to a navigation RPC. It can be implemented in a node responsible for navigation or related capabilities, which checks whether the location is known, reachable, and safe. Errors propagate back to the LLM as structured failure messages, which makes it easier for the system to recover or request clarification from the user. Furthermore, when a requested location is invalid, the LLM can call a companion tool, such as *get_navigation_targets*, to list valid destinations first, enabling the selection of alternatives or asking the user to choose from known locations. This explicit and typed mapping avoids the ambiguity often associated with free-form natural language plans and helps maintain predictable system behavior. Both tools map directly to existing middleware functions. We expose a single MCP server as one middleware node. Each node provides a small manifest (i.e. *mcp.json*) that is indexed by the server and declares which RPCs are exposed as tools, optionally remapping internal names to LLM-facing identifiers while omitting safety-critical calls. Manifests may also define default arguments to simplify tool usage. This manifest-based discovery automatically updates the tool surface when components change, while the middleware retains responsibility for feasibility checks, execution, and device state.

3.2 Snapshot Access to Sensor Data

Continuous streams such as camera images or localization updates should not be forwarded directly into an LLM because they quickly exceed context window limits, increase latency, and obscure which data point a decision depends on. For speech-based interaction, the LLM typically needs only occasional access to the most recent value. We therefore introduce a snapshot mechanism, in which a lightweight component subscribes to relevant topics and continuously buffers the latest message. When the LLM calls a snapshot tool, the component returns the buffered value. If a sensor becomes unavailable or the buffered value exceeds a configurable age threshold, the tool returns a null or flagged stale value. In practice, the snapshot mechanism is realized by a dedicated node that exposes

generic RPCs. For each subscribed topic, it offers a snapshot call that returns the most recent buffered message. This approach keeps high-rate data inside the middleware while giving the LLM explicit and predictable access to this data.

3.3 Backend Flexibility

Because all communication passes through MCP, the LLM backend is independent of robot capabilities and can be swapped between cloud, edge, or local deployments. All communicate with the robot through the same tool definitions, which simplifies experimentation with different models and accommodates both provider-agnostic deployments and privacy-sensitive scenarios. This separation allows improvements in LLM capabilities to be adopted without redesigning robot-side logic and enables reuse on other robots that expose callable middleware functions.

4 Practical Application on a Co-Presence Robot

We applied the proposed integration approach to a mobile co-presence robot that supports conversational voice interaction for tasks such as navigation, projector control, and video-based communication. The robot runs a standard robotic middleware stack with MIRA while the LLM executes on an external compute node. This setup reflects a typical deployment where on-robot resources are limited, and speech interaction benefits from a more powerful model running on an edge server, here equipped with an NVIDIA RTX 5090, while keeping audio processing local for privacy.

4.1 Audio Interaction Pipeline

On the co-presence robot, we implement a lightweight audio pipeline combining wake word detection (WWD), voice activity detection (VAD), LLM inference, and text-to-speech (TTS), as shown in Fig. 3. We use openWakeWord[4] for WWD, SileroVAD[5] for VAD, vLLM[12] with Qwen3-Omni for LLM inference, and Piper[6] for TTS. The WWD component ensures that the robot only processes speech when explicitly addressed. VAD is used to gate the input so that the LLM receives complete speech segments rather than continuous, potentially irrelevant audio, which also reduces unnecessary compute. The LLM produces either tool calls or verbal responses, and TTS synthesizes the spoken output. To reduce perceived latency, LLM output is streamed and spoken sentence-wise, so the robot can respond before the full output is generated. A fallback mode informs the user if the LLM backend is unavailable, for example due to a bad network connection.

4.2 Prototype Deployment

The system was tested in a laboratory co-presence scenario aimed at illustrating feasibility rather than evaluating long-term interaction performance. The separation between audio processing, LLM inference, and middleware execution proved practical. The robot remains responsive even when the LLM is under load, and the middleware continues to enforce feasibility and safety constraints.

In the tested scenario, Qwen3-Omni reliably transformed spoken requests into structured tool calls, particularly when using its

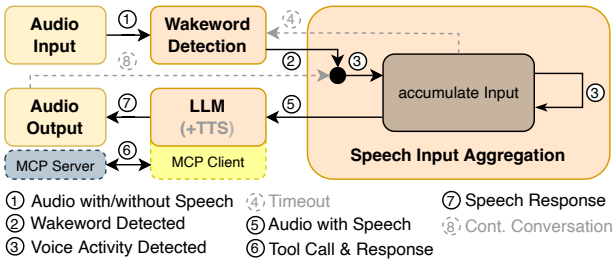

Figure 3: Routine overview. After the wake word is detected, input audio is segmented by VAD and passed to the LLM, which produces either a tool call or a verbal response. The result is spoken through TTS.

reasoning-capable variant. Most failures were caused by ambiguous commands rather than incorrect tool invocation. The MCP-based interface allows the LLM backend to be replaced without changing robot-side logic, so improvements in audio-capable and tool-calling models can be adopted directly [10, 21]. Furthermore, the architecture allows privacy-sensitive deployments to move inference entirely on-device if needed.

Overall, the deployment demonstrates that a middleware-aligned, tool-based integration can support natural voice interaction on a mobile co-presence robot without requiring extensive custom logic. As a Late Breaking Report, this work focuses on the integration pattern and on prototype feasibility. Systematic user studies with participants are planned as future work.

5 Discussion

The proposed integration pattern combines structured tool calling through an MCP-enabled LLM with middleware-centered execution, ensuring that robot actions remain predictable and traceable. At the same time, the separation between LLM reasoning and middleware execution keeps the architecture flexible, allowing different models or deployment backends to be swapped without changing robot-side logic. Because the LLM issues explicit tool calls rather than free-form plans as in LLM-centric approaches, every action can be validated, logged, or rejected by the middleware. The snapshot mechanism further contributes to clarity by making sensor access explicit and preventing model behavior from relying on uncontrolled continuous streams. Potential failure modes include misinterpretation of user intent, ambiguous commands, and hallucinated tool calls. While ambiguity and misinterpretation primarily depend on the capabilities of the chosen LLM and affect the interaction itself, incorrect tool calls are constrained by the explicit tool surface and middleware-side validation.

From an HRI perspective, the middleware-aligned interface supports transparency and turn-taking while decoupling robot capabilities from specific models, enabling simplifications as audio-to-audio LLMs become more robust.

Acknowledgments

This work has received funding from the Carl-Zeiss-Stiftung for the project *Co-Presence of Humans and Interactive Companions for Seniors* (CO-HUMANICS, 06/2021–05/2026), www.co-humanics.de.

[4] https://github.com/dscripka/openWakeWord – accessed December 2, 2025
[5] https://github.com/snakers4/silero-vad – accessed December 2, 2025
[6] https://github.com/OHF-Voice/piper1-gpl – accessed December 2, 2025

References

[1] Anthropic. 2024. Introducing the Model Context Protocol — anthropic.com. https://www.anthropic.com/news/model-context-protocol. [Accessed 24-11-2025].

[2] Anthropic. 2025. Claude Sonnet 4.5. (2025). https://www.anthropic.com/news/claude-sonnet-4-5

[3] Stephanie Arévalo Arboleda, Söhnke Fischedick, Chenayo Diao, et al. 2024. An exploratory study on the impact of varying levels of robot control on presence in robot-mediated communication. In *Proc. of RO-MAN*. 83–88. doi:10.1109/RO-MAN60168.2024.10731330

[4] 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*. 905–906. doi:10.1145/3568294.3580040

[5] Melisa Conde, Veronika Mikhailova, and Nicola Döring. 2024. "I have the Feeling that the Person is Here": Older Adults' Attitudes, Usage Intentions, and Requirements for a Telepresence Robot. *Int. J. Soc. Robot.* 16, 7 (July 2024), 1619–1639. doi:10.1007/s12369-024-01143-z

[6] Ana Davila, Jacinto Colan, and Yasuhisa Hasegawa. 2024. Voice control interface for surgical robot assistants. In *2024 International Symposium on Micro-NanoMehatronics and Human Science (MHS)*. IEEE, 1–5. doi:10.1109/mhs63891.2024.10856297

[7] Erik Einhorn, Tim Langner, Ronny Stricker, et al. 2012. MIRA – Middleware for Robotic Applications. In *Proc. of IROS*. doi:10.1109/IROS.2012.6385959

[8] Söhnke Benedikt Fischedick, Kay Richter, Tim Wengefeld, et al. 2023. Bridging Distance with a Collaborative Telepresence Robot for Older Adults – Report on Progress in the CO-HUMANICS Project. In *Proc. of ISR*.

[9] Khashayar Ghamati, Maryam Banitalebi Dehkordi, et al. 2025. Towards AI-Powered Applications: The Development of a Personalised LLM for HRI and HCI. *Sensors* 25, 7 (2025), 2024. doi:10.3390/s25072024

[10] Dhruv Jain, Harshit Shukla, Gautam Rajeev, Ashish Kulkarni, Chandra Khatri, and Shubham Agarwal. 2025. VoiceAgentBench: Are Voice Assistants ready for agentic tasks? *arXiv preprint arXiv:2510.07978* (2025).

[11] Anis Koubaa, Adel Ammar, and Wadii Boulila. 2025. Next-generation human-robot interaction with ChatGPT and robot operating system. *Software: Practice and Experience* 55, 2 (2025), 355–382. doi:10.1002/spe.3377

[12] Woosuk Kwon, Zhuohan Li, and Siyuan Zhuang. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*. doi:10.1145/3600006.3613165

[13] Alexander H. Liu, Andy Ehrenberg, Andy Lo, et al. 2025. Voxtral. *arXiv preprint arXiv:2507.13264* (2025).

[14] Steven Macenski, Tully Foote, and Brian Gerkey. 2022. Robot operating system 2: Design, architecture, and uses in the wild. *Science robotics* 7, 66 (2022), eabm6074. doi:10.1126/scirobotics.abm6074

[15] Christopher E Mower, Yuhui Wan, Hongzhan Yu, Antoine Grosnit, Jonas Gonzalez-Billandon, Matthieu Zimmer, Jinlong Wang, Xinyu Zhang, Yao Zhao, Anbang Zhai, et al. 2024. ROS-LLM: A ROS Framework for Embodied AI with Task Feedback and Structured Reasoning. *arXiv preprint arXiv:2406.19741* (2024).

[16] OpenAI. 2024. GPT-4o System Card. arXiv:2410.21276 [cs.CL] https://arxiv.org/abs/2410.21276

[17] OpenAI. 2025. GPT-5 System Card. https://cdn.openai.com/gpt-5-system-card.pdf. [Accessed 01-12-2025].

[18] Sihyeong Park, Sungryeol Jeon, Chaelyn Lee, et al. 2025. A Survey on Inference Engines for Large Language Models: Perspectives on Optimization and Efficiency. *arXiv preprint arXiv:2505.01658* (2025).

[19] Morgan Quigley, Ken Conley, and Brian Gerkey. 2009. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*.

[20] Stanislau Stankevich and Wojciech Dudek. 2024. Interpreting and learning voice commands with a Large Language Model for a robot system. arXiv:2407.21512 [cs.RO] https://arxiv.org/abs/2407.21512

[21] Sidharth Surapaneni, Hoang Nguyen, Jash Mehta, Aman Tiwari, Oluwanifemi Bamgbose, Akshay Kalkunte, Sai Rajeswar, and Sathwik Tejaswi Madhusudhan. 2025. AU-Harness: An Open-Source Toolkit for Holistic Evaluation of Audio LLMs. *arXiv preprint arXiv:2509.08031* (2025).

[22] Daniel Tanneberg, Felix Ocker, Stephan Hasler, et al. 2024. To help or not to help: Llm-based attentive support for human-robot group interactions. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 9130–9137.

[23] Gemma Team. 2025. Gemma 3n. (2025). https://ai.google.dev/gemma/docs/gemma-3n

[24] Meituan LongCat Team, Bairui Wang, Bayan, and Bin Xiao. 2025. LongCat-Flash-Omni Technical Report. arXiv:2511.00279 [cs.MM] https://arxiv.org/abs/2511.00279

[25] Gijs Wijngaard, Elia Formisano, Michel Dumontier, et al. 2025. AudioToolAgent: An Agentic Framework for Audio-Language Models. arXiv:2510.02995 [cs.SD] https://arxiv.org/abs/2510.02995

[26] Jin Xu, Zhifang Guo, Jinzheng He, et al. 2025. Qwen2.5-omni technical report. *arXiv preprint arXiv:2503.20215* (2025).

[27] Jin Xu, Zhifang Guo, Hangrui Hu, et al. 2025. Qwen3-Omni: An Audio-Visual Language Model with Function Calling. *arXiv preprint arXiv:2509.17765* (2025).

[28] Jessie Yuan, Janavi Gupta, Akhil Padmanabha, Zulekha Karachiwalla, Carmel Majidi, Henny Admoni, and Zackory Erickson. 2024. Towards an LLM-Based Speech Interface for Robot-Assisted Feeding. In *The 37th Annual ACM Symposium on User Interface Software and Technology (UIST '24)*. ACM, 1–4. doi:10.1145/3672539.3686759

[29] Yuchong Zhang, Bastian Orthmann, and Shichen Ji. 2025. Multimodal "Puppeteer": An Exploration of Robot Teleoperation Via Virtual Counterpart with LLM-Driven Voice and Gesture Interaction in Augmented Reality. arXiv:2506.13189 [cs.HC] https://arxiv.org/abs/2506.13189