Empowering NAO: Overcoming Humanoid Robot Constraints via Large Language Models

Valerio Colonnese[1,*], Gilda Manfredi[1], Nicola Capece[1], Ugo Erra[1] and Anthony Rivelli[1]

[1]Università degli Studi della Basilicata, Via dell'Ateneo Lucano 10, Potenza, 85100, Italy

Abstract

Humanoid robots such as NAO are widely used in educational and research settings, yet their interaction capabilities remain constrained by limited onboard computing power and rigid, script-based control architectures. This work presents a fully local, LLM-driven interaction framework that augments the NAO humanoid robot by decoupling low-level embodiment from high-level language understanding and reasoning. The proposed system follows a client–server architecture in which NAO acts as an embodied interface, while speech processing, request interpretation, and behavior synthesis are delegated to locally executed Large Language Models (LLMs). Spoken user input is transcribed via a speech-to-text pipeline and dynamically routed through prompt-engineered agents that either generate natural language responses or synthesize executable Python code compliant with the NAOqi framework. To improve reliability and prevent invalid robot actions, the framework integrates Retrieval-Augmented Generation (RAG) grounded in structured, robot-specific knowledge. The system is evaluated under both conversational and action-oriented interaction scenarios, comparing two code-oriented LLMs, CodeLlama 13B and Qwen2.5-Coder 14B. Experimental results show that locally deployed LLMs can effectively support adaptive human–robot interaction, while highlighting the critical role of model selection in latency, request-routing reliability, and task success rate. Overall, the findings demonstrate that retrieval-augmented, locally executed LLMs represent a viable and scalable approach for extending the functional lifespan of legacy humanoid robotic platforms without modifying their native control stack or relying on cloud-based services.

Keywords

Large Language Models (LLMs), Embodied Artificial Intelligence, Humanoid Robotics, Code Generation for Robot Control, Retrieval-Augmented Generation (RAG), Legacy Robotic Platforms, Human-Robot Interaction

1. Introduction

Humanoid robots are increasingly adopted in educational, research, and social environments due to their embodied nature and their potential to support intuitive human–robot interaction. Among existing platforms, the NAO humanoid robot (Figure 1) remains widely used thanks to its accessibility, modular software ecosystem, and rich sensorimotor capabilities [1]. Despite its diffusion, however, NAO's interaction paradigm is still largely constrained by predefined behaviors, rigid dialogue structures, and limited onboard computational resources, which hinder flexible and context-aware interaction. In parallel, recent advances in Large Language Models (LLMs), driven by Transformer-based architectures [2], have significantly expanded the capabilities of natural language understanding, reasoning, and code generation. These models have demonstrated strong performance in conversational agents, instruction following, and program synthesis, making them promising candidates for enhancing human–robot interaction. However, integrating LLMs into physically embodied robotic platforms remains challenging. Legacy humanoid robots such as NAO were not designed to host large-scale neural models or complex reasoning pipelines, which often motivates solutions based on cloud services or simplified dialogue logic. While cloud-based approaches provide access to powerful models, they introduce limitations in terms of latency, reliability, privacy, and deployment robustness, which are particularly critical for real-time, speech-based interaction [3]. Moreover, transferring language-driven program synthesis from virtual

Joint Proceedings of the ACM Intelligent User Interfaces (IUI) Workshops 2026, March 23-26, 2026, Paphos, Cyprus

*Corresponding author.

valerio.colonnese@unibas.it (V. Colonnese); gilda.manfredi@unibas.it (G. Manfredi); nicola.capece@unibas.it (N. Capece); ugo.erra@unibas.it (U. Erra); anthony.rivelli@studenti.unibas.it (A. Rivelli)

0009-0001-8176-1228 (V. Colonnese); 0000-0003-0633-862X (G. Manfredi); 0000-0002-1544-3977 (N. Capece); 0000-0003-2942-7131 (U. Erra)

Figure 1: NAO humanoid robot used in the experiments. The robot serves as the physical interface for speech input and output and for motor execution within the proposed local LLM-driven interaction framework.

agents to physical robots imposes additional constraints, including strict API compatibility requirements, real-time execution demands, and the need to prevent invalid or unsafe actions. Although recent work in human-robot interaction has explored the use of LLMs for conversational interaction, task planning, and instruction interpretation, reliably grounding language-driven action synthesis in robot-specific capabilities and execution constraints remains an open challenge. To address these limitations, this work proposes a fully local, LLM-driven interaction framework designed to augment the NAO humanoid robot without modifying its native control stack. The system adopts a client-server architecture in which NAO acts as an embodied interface for perception and actuation, while speech processing, language understanding, and behavior synthesis are delegated to locally executed LLMs accessed via Open WebUI [4] and Ollama [5]. Spoken user input is transcribed through a speech-to-text pipeline and dynamically routed through prompt-engineered agents that generate either natural language responses or executable Python code compliant with the NAOqi framework. To improve reliability and constrain model outputs to the robot's actual capabilities, the framework integrates Retrieval-Augmented Generation (RAG) [6] grounded in structured, robot-specific knowledge. Accordingly, the proposed framework is evaluated with respect to system-level reliability, latency, and execution correctness, rather than user-centered or immersive interaction metrics, through a comparative study of two code-oriented LLMs, CodeLlama 13B and Qwen2.5-Coder 14B.

2. Related Work

Recent research has increasingly investigated the integration of LLMs into humanoid robotic platforms to mitigate the intrinsic limitations of legacy systems such as NAO, whose onboard computational resources and native dialogue mechanisms are not designed to support open-ended language understanding or adaptive reasoning. A substantial body of work focuses on enhancing conversational capabilities by interfacing NAO with external generative language services. Campilii *et al.* [7] integrate NAO with OpenAI's ChatGPT through a Python-based architecture that addresses incompatibilities with the robot's Python 2.7 environment, evaluating response time, robustness, and performance under acoustic noise. Similarly, Andreeva *et al.* [8] propose a cloud-based conversational architecture for therapeutic scenarios involving children with speech and language disorders, combining speech recognition, LLMs, and multilingual text-to-speech services orchestrated via Node-RED, while also addressing ethical and regulatory concerns. While these approaches demonstrate the feasibility of LLM-enhanced dialogue, they primarily emphasize conversational quality, with limited attention to grounding language-driven interaction in executable robot behaviors. Beyond purely conversational systems, several studies have explored the use of LLMs as intermediaries between natural-language instructions and robot actions. Abpeikar *et al.* [9] present a framework in which an LLM determines whether a user request should result in a verbal response or a physical action, representing the latter through behavior trees that are subsequently translated into NAO actions. Complementary to this line of work, Reforgiato Recupero and Boi [10] propose an ontology-driven approach in which an LLM handles question answering while robot actions are selected through semantic matching against an action ontology to verify posture compatibility and execution feasibility. These methods introduce structured decision layers that reduce the risk of invalid or unsafe actions, but they typically rely on predefined behavior sets and do not fully address open-ended interaction scenarios that require the runtime synthesis of executable robot control code. More recently, LLMs have also been investigated for motion and gesture generation in humanoid robots. Catalini *et al.* [11] treat LLMs as high-level motion planners that translate natural language descriptions into sequences of joint configurations for NAO, evaluating motion quality and recognizability through a large-scale user study. While this work highlights the potential of LLMs to bridge linguistic intent and embodied motion at the kinematic level, it focuses on motion generation fidelity and human perception rather than on end-to-end spoken interaction, execution constraints, or dialogue management. In contrast to these approaches, the present work targets a fully local interaction framework designed to operate within the strict constraints of the NAOqi ecosystem. By combining a client–server architecture with NAO acting as an embodied interface, prompt-engineered routing between conversational and action-oriented requests, synthesis of NAOqi-compliant executable code, and retrieval-augmented grounding over structured robot-specific knowledge, our framework emphasizes reliable execution and practical deployability on legacy humanoid platforms. Furthermore, by systematically comparing code-oriented LLMs under a unified evaluation protocol with respect to end-to-end latency and task success rate, this work addresses a practical reliability dimension that remains underexplored in existing studies on LLM integration for the NAO robot.

3. Materials and Methods

3.1. System Architecture and Experimental Setup

The proposed framework adopts a fully local client-server architecture [12] designed to extend the interaction capabilities of the NAO humanoid robot while preserving its native software stack. In this configuration, NAO operates as a lightweight embodied client responsible for audio acquisition, speech output, and motor execution, whereas computationally intensive processes, including speech transcription, language understanding, reasoning, and response generation, are delegated to an external local server. This architectural separation enables the integration of LLMs despite the computational and software constraints imposed by the NAOqi ecosystem [13, 14]. An overview of the system architecture and communication flow is shown in Figure 2. Spoken user input is captured through NAO's onboard

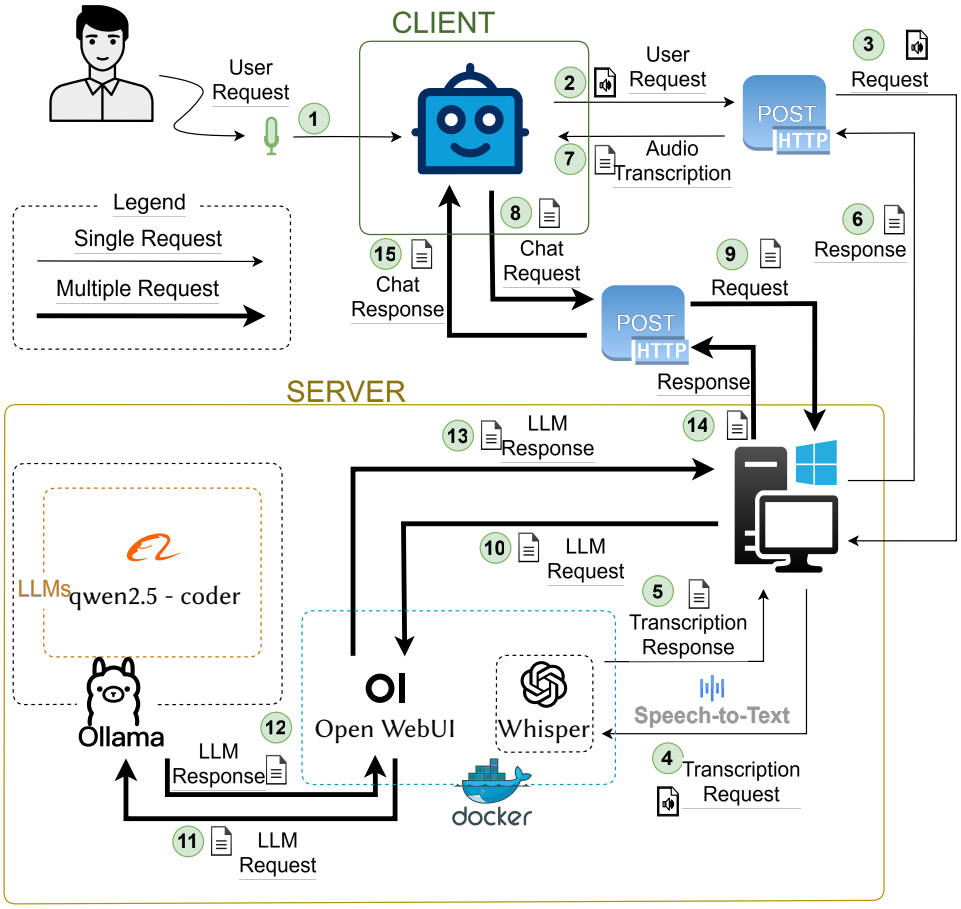


Figure 2: Client–server architecture of the proposed system. The NAO robot captures user speech and executes actions, while a local server performs speech transcription, request routing, language model inference, and response generation. Arrows indicate the direction of data flow between components.

microphones and transmitted to the server via authenticated HTTP requests, with audio recording automatically terminated through silence detection to ensure consistent speech segmentation. On the server side, speech-to-text transcription is performed using the Faster Whisper large-v3 model [15, 16], selected for its robustness and favorable trade-off between accuracy and inference latency. The resulting transcription constitutes the input to the language-driven interaction pipeline. Following transcription, user requests are routed by a prompt-engineered LLM to either a conversational branch, which produces a natural-language response, or an action-oriented branch, which produces executable robot-control code. The internal control flow implemented within the Choregraphe application, covering speech acquisition, request routing, response generation, execution, and feedback, is summarized in Figure 3. Responses generated by the server are returned to the robot in textual form and are vocalized using NAO's text-to-speech module or executed through the NAOqi framework, enabling seamless alternation between conversational behavior and embodied action.

3.2. Hardware and Software Stack

The physical interaction platform used in the experiments is a NAO v6 humanoid robot, which provides a rich set of sensors and actuators for speech-based and embodied interaction. However, NAO operates under strict software constraints, most notably its reliance on Python 2.7 and legacy NAOqi APIs [13]. The NAOqi runtime environment is tightly coupled to this legacy Python version, lacks support for modern machine learning libraries, and offers limited computational resources, making the direct deployment of large neural models or advanced reasoning pipelines infeasible. As a result, approaches based on onboard inference, fine-tuning, or direct integration of contemporary LLM frameworks are

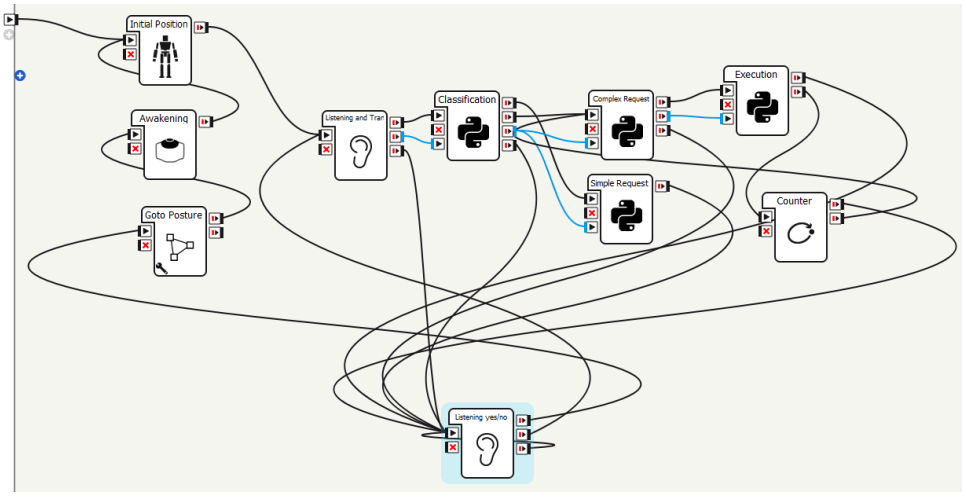


Figure 3: Control flow implemented in the Choregraphe application. The workflow comprises speech acquisition, request routing between conversational and action-oriented processing, response or executable code generation, execution on the robot, and feedback handling.

not viable on the NAO platform. These constraints motivate the architectural separation adopted in this work, in which low-level embodiment remains on the robot while all computationally intensive processes are offloaded to an external local server, preserving full compatibility with the native NAOqi control stack. The server infrastructure used in all experiments consists of a high-performance local workstation configured for low-latency LLM inference. The system is equipped with an Intel Core i9-13900KF CPU, 32 GB of DDR5 system RAM, and an NVIDIA GeForce RTX 4090 GPU with 24 GB of dedicated VRAM. This configuration enables LLMs with a medium to large number of parameters to be fully resident in GPU VRAM, minimizing data transfers between system RAM and VRAM and thereby reducing inference latency. A large VRAM budget is particularly important for interactive scenarios, as it enables both the speech-to-text model and the selected LLM to remain in memory during operation. The server hosts the LLM runtime, along with an orchestration layer that manages prompts, controls inference, and supports optional knowledge retrieval. On the robot side, behavior is implemented using Choregraphe [17], which serves as both the development environment and runtime controller for NAO applications. Custom Python scripts embedded within Choregraphe handle audio acquisition, network communication, and the execution of server responses. On the server side, LLMs are executed locally using Ollama [5], while Open WebUI provides a unified interface for model interaction, prompt configuration, and retrieval-augmented inference [4]. Communication between the robot and the server follows REST principles [18] and employs token-based authentication compatible with OAuth 2.0 [19], ensuring stateless and modular interactions.

3.3. LLM Integration, Prompt Design, and RAG

At the core of the proposed framework lies the integration of an LLM as a cognitive mediator between human intent and robotic execution. Rather than relying on model fine-tuning, the system adopts a prompt-based control strategy, enabling a general-purpose LLM to adapt to the constraints of the NAO robotic platform through carefully designed prompts. This choice ensures reproducibility, flexibility, and independence from retraining procedures, which is particularly important given the legacy nature of the NAOqi ecosystem. Given the dual requirement of generating both natural language responses and executable robot control code, the framework employs code-oriented LLMs optimized for structured output generation. Model selection was guided by a trade-off between reasoning capability and inference latency, leading to the evaluation of models in the 13B-14B parameter range. This size allows the models to be fully resident in GPU memory while preserving sufficient capacity for instruction decomposition, control-flow reasoning, and code synthesis. Prompt engineering plays a central methodological role in constraining model behavior and ensuring reliable interaction [20]. Distinct system prompts are

Table 1
Outcome-based routing reliability over 15 trials per condition. Simple requests correspond to speech-only responses, while complex requests require code generation and execution of robot actions.

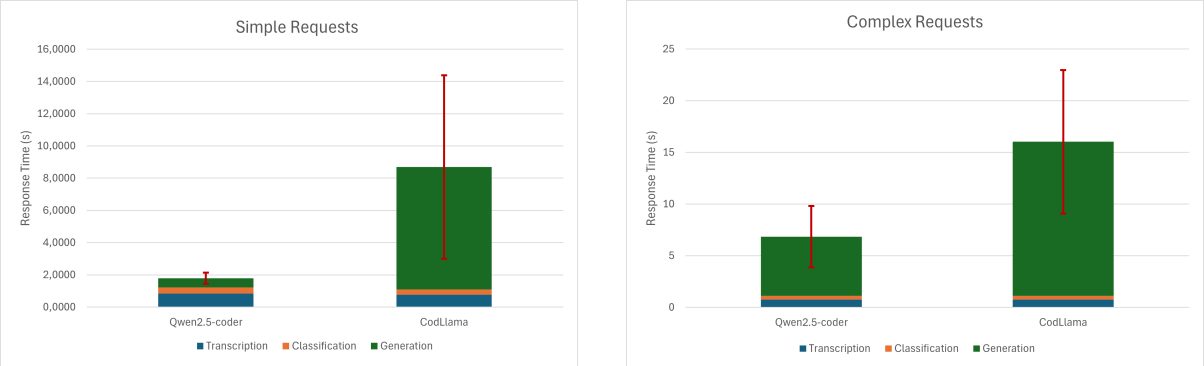
Model	Category	Successful	Failed	Success rate
Qwen2.5-Coder	Simple (*say*)	14	1	93,33%
Qwen2.5-Coder	Complex (*do*)	11	4	73,33%
CodeLlama	Simple (*say*)	4	11	26,67%
CodeLlama	Complex (*do*)	5	10	33,33%

used for request routing, conversational response generation, and code synthesis. The routing stage implements a binary distinction between *saying* and *doing*, determining whether a request should trigger speech generation or robot action. For action-oriented requests, prompts enforce strict compatibility with Python 2.7 and the NAOqi APIs, constraining the output to a predefined code template to reduce syntactic and semantic errors. To further improve robustness to articulated or multi-step commands, the system incorporates a Chain-of-Thought prompting strategy [21], which encourages the internal decomposition of complex instructions into simpler sub-actions. Although intermediate reasoning steps are not exposed to the user, their effect is reflected in more consistent and interpretable robot behavior. In addition to prompt-based constraints, the framework integrates a RAG mechanism to ground model outputs in robot-specific knowledge and mitigate hallucinations [22]. During inference, the LLM is augmented with an external structured knowledge base describing available actions, motion primitives, and sensor interfaces supported by the NAO platform. Robot-specific knowledge is encoded in lightweight JSON documents and organized into small, semantically focused files, improving retrieval precision and simplifying maintenance. Only the most relevant knowledge fragments are dynamically injected into the prompt at inference time, ensuring alignment between generated outputs and the robot's actual capabilities while reducing the risk of invalid API calls or unsupported behaviors.

4. Results and Evaluation

4.1. Evaluation Protocol and Metrics

The proposed framework was evaluated to quantify its responsiveness and functional reliability under spoken human-robot interaction. Experiments were conducted using a NAO v6 humanoid robot acting as the client, connected via a local network to a dedicated server hosting the LLMs and orchestration layer. Speech transcription was performed using the Faster Whisper `large-v3` model [15, 16] to ensure consistent speech-to-text conversion across all trials. Two code-oriented LLMs were evaluated under identical conditions: CodeLlama 13B and Qwen2.5-Coder 14B [23, 24]. Both models were executed locally and configured with task-specific prompts and the same retrieval-augmented knowledge base. User interactions were grouped into two predefined categories corresponding to conversational (simple) and action-oriented (complex) requests, as described in Section 3. A predetermined set of 15 simple and 15 complex requests was established. Each request was run once per model, yielding 30 trials per model. Requests were assigned to categories based on the expected system behavior rather than linguistic form. The primary performance metric is end-to-end latency, defined as the time elapsed from the end of the user's spoken input to the completion of the robot's response. Latency was decomposed into four stages: speech transcription, request routing, LLM inference, and action execution or speech synthesis. In addition, the task success rate was measured as the percentage of interactions that resulted in correct system behavior. A trial was considered successful if the system correctly routed the request and, in the case of complex interactions, generated and executed a syntactically valid, semantically correct NAOqi-compatible Python script that produced the intended robot action. Failures were categorized as request-routing errors, code-generation errors, or execution-time failures. All measurements were collected automatically by logging timestamps at each stage of the interaction pipeline.



(a) Simple requests

(b) Complex requests

Figure 4: Average end-to-end response time for simple and complex requests using CodeLlama 13B and Qwen2.5-Coder 14B. Bars show mean response time over 15 trials, and error bars indicate one standard deviation. Response time comprises speech transcription, request routing, language-model inference, and robot execution or speech synthesis.

4.2. Quantitative Results

Figure 4 reports the average end-to-end response time for simple and complex requests using CodeLlama 13B and Qwen2.5-Coder 14B. The comparison between models is based on numerical values reported in the figure and does not rely on color alone to distinguish conditions. For simple requests, Qwen2.5-Coder achieves an average response time of 1.79 ± 0.35 s, compared to 8.69 ± 5.70 s for CodeLlama. The difference becomes more pronounced for complex requests, where Qwen2.5-Coder reaches an average response time of 6.84 ± 2.96 s, while CodeLlama requires 16.02 ± 6.95 s on average. This gap is primarily attributable to differences in LLM inference time, whereas speech transcription and robot execution times remain comparable across models. Task success rates further highlight differences in reliability between the evaluated models. Qwen2.5-Coder successfully handles 93.3% of simple requests and 73.3% of complex requests, while CodeLlama achieves success rates of 26.7% and 33.3%, respectively. Table 1 summarizes successful and failed trials for each model and request category, providing an outcome-based characterization of routing reliability within the end-to-end interaction pipeline. Routing performance was evaluated at the system level rather than through isolated classification metrics. A routing decision was considered correct only if it resulted in appropriate system behavior, namely speech generation for simple requests and code generation followed by execution for complex requests. Per-trial routing labels were not explicitly logged; therefore, confusion matrices or per-class precision-recall metrics are not reported. Instead, routing reliability is reflected indirectly through outcome-based success and failure rates. An analysis of failure cases indicates that errors are primarily associated with incorrect request routing or with syntactic and semantic inconsistencies in the generated code. The integration of retrieval-augmented knowledge helps reduce invalid API calls and unsupported actions, thereby improving overall system stability, particularly in complex action-oriented interactions.

5. Discussion

The presented results highlight both the potential and the current limitations of augmenting legacy humanoid robots with locally deployed LLMs. By decoupling low-level embodiment from high-level cognitive processing, the proposed framework demonstrates that robots such as NAO can move beyond rigid, script-based interaction paradigms and support more flexible, language-driven behaviors without modifying their native control architecture. This architectural separation is a key enabler of integrating advanced reasoning and generation capabilities despite the severe computational and software constraints imposed by the NAOqi ecosystem. A central outcome of the evaluation concerns

the role of the LLM as a mediator between user intent and robot execution. While both evaluated models support the proposed interaction pipeline, the results indicate that model selection substantially affects system behavior, particularly in request interpretation and control-flow reasoning. Errors in intermediate decisions, such as request routing, propagate downstream, affecting latency, response quality, and overall interaction reliability. These findings suggest that, in embodied interaction scenarios, the quality of intermediate control decisions can be as critical as the correctness of the final generated output, and that code-oriented LLMs with stronger structural reasoning capabilities are better suited for mixed conversational and action-driven tasks. Although no explicit ablation study was conducted, qualitative inspection of failure cases indicates that grounding the LLM with structured, robot-specific knowledge helps limit unsupported API calls and action hallucinations, particularly in articulated or parameterized commands. This grounding mechanism is especially relevant in physically embodied systems, where erroneous outputs may not only degrade interaction quality but also lead to potentially unsafe behaviors. The use of multiple small, semantically focused knowledge fragments further enhances retrieval precision, thereby supporting more consistent alignment between user intent, generated code, and the robot's actual capabilities [22]. These observations reinforce the view that parametric knowledge alone is insufficient for reliable robot control, and that external structured knowledge sources play an important role in LLM-based robotic systems. Despite these advantages, several limitations emerge from the evaluation. Although end-to-end latency remains acceptable for many interaction scenarios, it is influenced by the cumulative cost of speech transcription, retrieval, and LLM inference, and becomes more noticeable in complex requests involving multi-step actions. Moreover, reliance on Python 2.7 and legacy NAOqi APIs constrains code expressiveness and limits the complexity of behaviors that can be safely generated and executed. These constraints emphasize that the goal of the proposed framework is not to modernize the native robotic software ecosystem, but rather to extend the functional lifespan of legacy humanoid platforms by introducing advanced cognitive capabilities externally while respecting their architectural limitations. Finally, the current evaluation focuses on system-level performance metrics such as latency and task success rate, without incorporating user-centered measures of interaction quality. While this choice aligns with the technical scope of the work, human-centered evaluations represent an important direction for future investigation, as they could provide additional insight into perceived responsiveness, naturalness, and trust. Overall, the results demonstrate that locally deployed, retrieval-augmented LLMs constitute a viable and scalable strategy for empowering existing humanoid robots, while highlighting critical design trade-offs that must be carefully managed in real-world interactive deployments.

6. Conclusions and Future Work

This work demonstrated that LLMs can be effectively employed to augment the interaction capabilities of legacy humanoid robots through a fully local, language-driven control architecture. By integrating locally executed LLMs with speech-based interaction, prompt-engineered request routing, dynamic code generation, and retrieval-augmented knowledge grounding, the proposed framework enables the NAO robot to move beyond rigid, script-based behaviors and operate as an adaptive, language-enabled embodied agent. The experimental evaluation confirms that the proposed approach can reliably support both conversational and action-oriented interactions while maintaining acceptable latency and execution robustness. In particular, the results highlight the critical role of LLM selection in embodied AI scenarios, showing that code-oriented models with stronger structural reasoning capabilities significantly improve both task success rate and interaction quality. These findings suggest that, even under strict computational and software constraints, locally deployed LLMs can provide a practical and scalable pathway for extending the functional lifespan of existing humanoid robotic platforms without modifying their native control stack or relying on cloud-based services. Building on these results, several research directions emerge as natural extensions of this work. A promising pathway involves decomposing complex user requests into sequences of simpler sub-tasks, enabling incremental execution and more fine-grained error recovery, with the potential to reduce both failure

rates and perceived interaction latency. Another direction concerns the integration of visual perception modules [25], such as object detection and scene understanding, which would allow the robot to ground linguistic instructions in its physical environment and support more context-aware behaviors. While the present study focuses on system-level integration of LLMs with a physically embodied humanoid robot, the proposed architecture could be extended to mixed reality (MR)- mediated human-robot interaction. For instance, an MR interface could provide an additional interaction layer, allowing users to replicate gestures or movements, visualize robot intent, or interact with shared virtual objects whose manipulation is grounded in the robot's physical capabilities. Finally, further optimization of the interaction pipeline, together with the exploration of more efficient or multimodal LLMs, may enable smoother, more natural, and more robust human–robot interaction in real-world settings. While the current work serves as a preliminary study validating the proposed framework, a systematic ablation study isolating the contribution of the retrieval-augmented component, by comparing LLM performance with and without external knowledge grounding, represents an important direction for future work.

Declaration on Generative AI

During the preparation of this work, the author(s) used generative AI tools (including LLMs) for the following purposes, according to the CEUR-WS activity taxonomy: grammar and spelling checking, language refinement, and stylistic revision of the manuscript. The author(s) also used generative AI systems during the development and testing of the proposed framework, as described in the paper, in particular for reasoning support and code generation within the experimental setup. All generated content was critically reviewed, validated, and, where necessary, edited by the author(s). The author(s) take full responsibility for the content of the publication.

References

[1] A. Robotics, NAO V6 Developer Technical Documentation, 2025. http://doc.aldebaran.com/2-8/family/nao_technical/index_dev_naov6.html, last accessed on 2025-06-03.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[3] L. Seymour, B. Kutukcu, S. Baidya, Large language models on small resource-constrained systems: Performance characterization, analysis and trade-offs, arXiv preprint arXiv:2412.15352 (2024).

[4] Designing an open-source llm interface and social platforms for collectively driven llm evaluation and auditing, Open WebUI (2024). https://openwebui.com/assets/files/whitepaper.pdf.

[5] Ollama, Ollama, 2025. https://github.com/ollama/ollama, last accessed on 2025-06-03.

[6] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive nlp tasks, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Curran Associates Inc., Red Hook, NY, USA, 2020.

[7] E. Campilii, D. Perpetuini, A. Merla, Enhancing human-robot interaction using generative artificial intelligence: The case of the nao robot, in: International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA), 2025. doi:10.1109/ACDSA65407.2025.11165865.

[8] A. T. Andreeva, A. K. Lekova, P. T. Tsvetkova, M. I. Simonska, Expanding the capabilities of robot nao to enable human-like communication with children with speech and language disorders, in: International Conference on Computer Systems and Technologies (CompSysTech), 2024. doi:10.1145/3674912.3674919.

[9] S. Abpeikar, M. Garratt, K. Kasmarik, S. Anavatti, Integrating large language models for task

planning in robots – a case study with nao, in: International Conference on Control and Robotics (ICCR), 2024. doi:10.1109/ICCR64365.2024.10927585.

[10] D. Reforgiato Recupero, L. Boi, Integrating action robot ontology for enhanced human-robot interaction: A nao robot case study, Springer-Verlag, Berlin, Heidelberg, 2025, p. 306–310. URL: https://doi.org/10.1007/978-3-031-78952-6_47. doi:10.1007/978-3-031-78952-6_47.

[11] R. Catalini, G. Salici, F. Biagi, G. Borghi, L. Biagiotti, R. Vezzani, Llms as nao robot 3d motion planners, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, 2025, pp. 2455–2465.

[12] V. Colonnese, G. Manfredi, N. Capece, U. Erra, S. Ungaro, M. Rinaldi, A large language model-driven architecture for intuitive interaction in virtual reality environments, in: 2025 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE), 2025, pp. 812–817. doi:10.1109/MetroXRAINE66377.2025.11340278.

[13] A. Robotics, NAOqi OS - Developer Documentation, 2025. http://doc.aldebaran.com/2-8/dev/tools/opennao.html#term-naoqi-os, last accessed on 2025-06-03.

[14] A. Robotics, NAOqi Framework Documentation, 2025. http://doc.aldebaran.com/2-8/naoqi/index.html, last accessed on 2025-06-03.

[15] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, I. Sutskever, Robust speech recognition via large-scale weak supervision, in: Proceedings of the 40th International Conference on Machine Learning, ICML'23, JMLR.org, 2023.

[16] SYSTRAN, faster-whisper: Faster Whisper transcription with CTranslate2, 2025. https://github.com/SYSTRAN/faster-whisper?tab=readme-ov-file, last accessed on 2025-06-03.

[17] A. Robotics, Choregraphe Software Overview, 2025. http://doc.aldebaran.com/2-8/software/choregraphe/choregraphe_overview.html, last accessed on 2025-06-03.

[18] R. T. Fielding, Architectural Styles and the Design of Network-based Software Architectures, University of California, Irvine, 2000. https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm, last accessed on 2025-06-03.

[19] D. Hardt, The OAuth 2.0 Authorization Framework, 2012. https://www.rfc-editor.org/info/rfc6749, last accessed on 2025-06-03.

[20] Q. Ye, M. Ahmed, R. Pryzant, F. Khani, Prompt engineering a prompt engineer, in: L.-W. Ku, A. Martins, V. Srikumar (Eds.), Findings of the Association for Computational Linguistics: ACL 2024, Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 355–385. URL: https://aclanthology.org/2024.findings-acl.21/. doi:10.18653/v1/2024.findings-acl.21.

[21] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Curran Associates Inc., Red Hook, NY, USA, 2022.

[22] F. Moiseev, Z. Dong, E. Alfonseca, M. Jaggi, SKILL: Structured knowledge infusion for large language models, in: M. Carpuat, M.-C. de Marneffe, I. V. Meza Ruiz (Eds.), Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Seattle, United States, 2022, pp. 1581–1588. URL: https://aclanthology.org/2022.naacl-main.113/. doi:10.18653/v1/2022.naacl-main.113.

[23] B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Lu, et al., Qwen2.5-coder technical report, arXiv preprint arXiv:2409.12186 (2024).

[24] Meta, Code Llama, 2023. https://github.com/meta-llama/codellama, last accessed on 2025-06-03.

[25] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection , in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2016, pp. 779–788. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.91. doi:10.1109/CVPR.2016.91.