Simposio CEA de Robótica, Bioingeniería, Visión Artificial y Automática Marina 2025

Interaction with a humanoid robot through a conversational interface using DeepSeek

Łukawski, B.[*], Victores, J. G., Balaguer, C., Jardón, A.

RoboticsLab, Department of Systems Engineering and Automation, Universidad Carlos III de Madrid, Avda. Universidad, 30, 28911, Leganés, Spain.

Resumen

Se presenta una aplicación para entablar conversaciones con un agente de chat ejecutándose sobre un robot humanoide dotado con el equipo audio correspondiente (micrófono, altavoz). Se ha diseñado considerando dos limitaciones: todos los componentes están respaldados por software libre y abierto, y los cálculos se deben ejecutan fuera de línea, bien localmente, o bien en un servidor dedicado a bordo del propio robot. El intermediario y entorno de trabajo YARP conforma la base de una arquitectura distribuida de módulos desarrollados en el marco de este trabajo: síntesis de voz (implementada con eSpeak y Piper), reconocimiento de voz (usando Kaldi por medio de Vosk), detección de palabras de activación (mediante openWakeWord) e inferencia sobre modelos de lenguaje extensos (con llama.cpp). La aplicación ha sido probada sobre el robot humanoide TEO usando una variante destilada del modelo de lenguaje DeepSeek-R1. Los resultados muestran que es posible adoptar un agente conversacional sin acceso a red y de baja latencia para tareas de interacción humano-robot.

Palabras clave: interacción humano-robot, conversación automatizada, modelo extenso de lenguaje, síntesis de voz, reconocimiento de voz, computación en la frontera, robótica humanoide.

Abstract

This work presents an application that enables users to hold a conversation with a chat agent, aimed to be launched aboard a humanoid robot equipped with audio sink and source hardware. It was designed to fulfill two constraints: all components are backed up by free and open-source software, and all computations must be carried out offline, either locally or offloaded to a dedicated edge server on the robot itself. The YARP robotics middleware and framework was leveraged to be at the foundation of a distributed architecture of newly developed modules: text-to-speech (implemented with eSpeak and Piper), automatic speech recognition (powered by Kaldi, wrapped by Vosk), wake word detection (via openWakeWord) and inference on large language models (with llama.cpp). The application was tested on the humanoid robot TEO using a distilled variant of the DeepSeek-R1 language model. Results show that a fully offline low-latency conversational agent can be adopted to achieve human-robot interaction tasks.

Keywords: human-robot interaction, chat completion, large language model, text-to-speech, automatic speech recognition, edge computing, humanoid robotics.

1. Introduction

This work is motivated by the desire to acquire conversational capabilities on the robot TEO depicted in Figure 1. This full-sized humanoid features 28 degrees of freedom on four limbs, torso and head, force-torque and inertial sensors, color and depth cameras, and a speaker and microphone. These components are managed by three on-board computers to perform a wide range of tasks and functionalities, including manipulation, locomotion, vision, force feedback, etc. It is sought to explore other ways of interacting with TEO through a fully offline conversation interface using the integrated audio hardware.

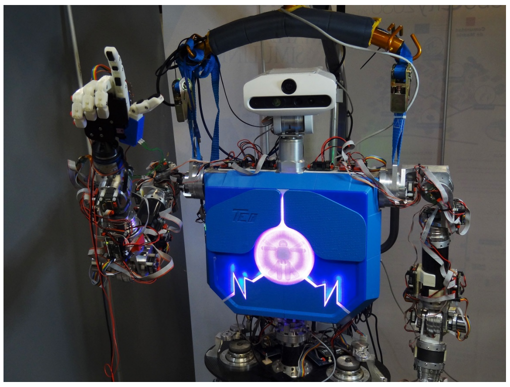


Figure 1: TEO humanoid robot (Pérez Martínez et al., 2010).

[*]Corresponding author: blukawsk@ing.uc3m.es

2. Background

In this study, it has been deemed interesting to focus on the main components of the audio system, i.e. speech synthesis and speech recognition, and analyze the application on human-robot interaction (HRI) through different means, including speech.

Regarding speech synthesis, text-to-speech (TTS) is devised as a means of automatically producing new sentences from text, as opposed to simply reproducing it, e.g. from a cassette, or concatenating isolated words or parts of sentences. Therefore, TTS can be described as the "grapheme-to-phoneme transcription of the sentences to utter". Digital signal processing (DSP) along with natural language processing (NLP) were highlighted as successful tools prior to the wide adoption of machine learning. Early implementations struggled to account for variability, speaker and speaking style effects to derive natural sounding intonation (Dutoit, 1997). Over the years, focus was shifted from formant synthesis and articulatory synthesis, although these techniques required less memory, to unit selection synthesis, thus seeking generation of speech that resembled a more natural voice. Hidden Markov models (HMM) were deemed a more suitable tool to achieve that (Rashad et al., 2010). With the advent of deep learning, however, naturalness of speech has been greatly improved and the focus is shifting towards the development of lightweight deep neural network (DNN) architectures that enable automatic synthesis from text (Khanam et al., 2022).

Automatic speech recognition (ASR), on the other hand, allows computers to convert voice into written text. To achieve high quality synthesis, traditional techniques include HMM and mel-frequency cepstrum coefficients (MFCCs) (Radha and Vimala, 2012). More recent approaches delve into the introduction of machine learning, yielding much better results in speech recognition than in other areas that use this technique. While MFCCs is still used for feature extraction, hybrid models mixing DNN with HMM and Gaussian mixture models (GMM) provide more accurate output. Recurrent neural networks (RNN) such as long short term memory (LSTM) gained traction in new developments (Nassif et al., 2019). More recently, large language models (LMM) helped centralize the cognitive competencies of computers and robotic agents through multimodal conversation capabilities, acting as a central text-based coordinating unit (Allgeuer et al., 2024).

Both areas have been merged to seek and study social interaction between humans and robots. A line of research involving the Robovie and Robovie-II humanoid robots investigated the different ways of them for initiating an conversation and getting the attention of human participants (Kanda et al., 2004; Hayashi et al., 2008; Shi et al., 2015) Other studies stressed on the ability of the robot for sensing people in its vicinity and knowing their location, towards the interaction with multiple persons (Bennewitz et al., 2005). Existing systems such as WikiTalk, a dialogue system, were integrated with the Nao humanoid robot to extend the robot's interaction capabilities along with motor functions such as nodding, gesturing, face-tracking, etc. (Csapo et al., 2012). Erica, a robot with a clearly human resemblance, was provided with a dialogue engine that allowed it to proactively produce coherent responses, accounting for focused words detected in the user's utterance (Milhorat et al., 2019).

3. Proposed Architecture

The implementation presented in this work relies on the middleware and robotics framework YARP (Yet Another Robot Platform) for leveraging communications and programming tools focused on developing robotic applications (Metta et al., 2006). Thanks to its modular approach, a distributed architecture can be introduced in which individual blocks ("devices") implemented as dynamically loaded libraries ("plugins") perform an individual task, e.g. data acquisition from a sensor, PID control of a motor, kinematic computations as a service, etc. A communications layer wraps each of these devices, thus exposing their inputs and outputs over a local network of interconnected computers. Computationally intensive tasks, especially those involving the usage of neural networks, can be offloaded to stations with graphics processing units (GPU). Previous works showcase these tools, for instance, to teleoperate a robotic manipulator arm using a range of low-cost peripherals (Łukawski et al., 2023). Although the core of YARP was written in C++, bindings are provided for a variety of programming languages, most notably Python.

The proposed architecture introduces a collection of YARP devices written in C++ and orchestrated by a central Python script serving the role of a chat application. Each device fulfills a specific task in the setup outlined in Figure 2:

- Test-to-Speech (TTS): synthesizes audio from text.
- Automatic Speech Recognition (ASR): transcribes audio to text.
- Large Language Model (LLM): generates a text response from text input using a language model run on the GPU.
- Wake word: ASR module trained to detect a specific short audio sentence.
- Player: reproduces audio frames using a speaker.
- Recorder: grabs audio frames from a microphone.

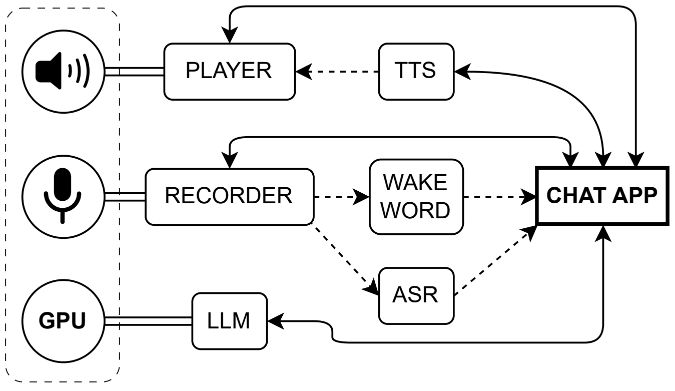
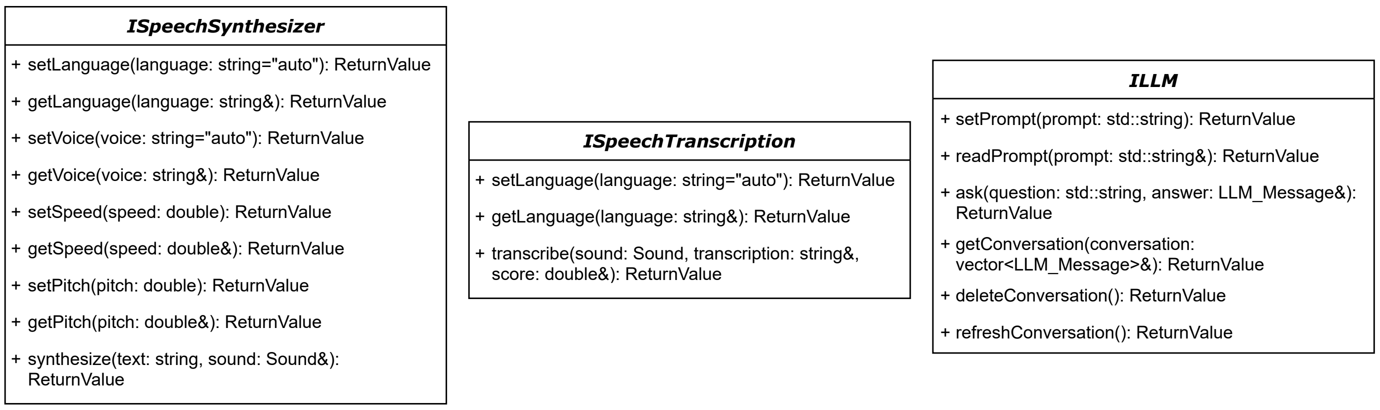


Figure 2: Proposed architecture. Solid lines indicate two-way communication via remote procedure calls (RPC, involving request-acknowledge messages). Dashed lines indicate one-way streaming communication.

Figure 3 describes the Application Programming Interface (API) of the TTS, ASR (including wake word) and LLM modules using the Unified Modeling Language (UML). Thanks to the language bindings, these methods can be called both from C++ and Python code.



```
ISpeechSynthesizer
+ setLanguage(language: string="auto"): ReturnValue
+ getLanguage(language: string&): ReturnValue
+ setVoice(voice: string="auto"): ReturnValue
+ getVoice(voice: string&): ReturnValue
+ setSpeed(speed: double): ReturnValue
+ getSpeed(speed: double&): ReturnValue
+ setPitch(pitch: double): ReturnValue
+ getPitch(pitch: double&): ReturnValue
+ synthesize(text: string, sound: Sound&):
  ReturnValue
```

```
ISpeechTranscription
+ setLanguage(language: string="auto"): ReturnValue
+ getLanguage(language: string&): ReturnValue
+ transcribe(sound: Sound, transcription: string&,
  score: double&): ReturnValue
```

```
ILLM
+ setPrompt(prompt: std::string): ReturnValue
+ readPrompt(prompt: std::string&): ReturnValue
+ ask(question: std::string, answer: LLM_Message&):
  ReturnValue
+ getConversation(conversation:
  vector<LLM_Message>&): ReturnValue
+ deleteConversation(): ReturnValue
+ refreshConversation(): ReturnValue
```

(a) Text-to-speech (TTS).   (b) Automatic Speech Recognition (ASR).   (c) Large Language Model (LLM).

Figure 3: Unified Modeling Language (UML) diagrams of selected YARP C++ interfaces.

4. Software Components

To leverage the distributed and modularized plugin approach, YARP allows splitting the logic of the devices that perform the heavy-lifting of their designated task, and the communications layer that exposes their API over the YARP network. The latter is managed by a separate collection of device pairs that implement the client-server approach, referred to as network wrapper servers (NWS) and network wrapper clients (NWC) in YARP jargon. Data flows through ports handled by these special devices, while the programmer only needs to care about their correct instantiation and connection.

The devices presented next belong to the former group. All internal implementations rely on free and open-source libraries.

4.1. Audio-Related Hardware Devices

Already bundled with the YARP library, two devices are devoted to communicating with the hardware: to reproduce sounds using a speaker, and to record audio from a microphone; see the Player and Recorder blocks in Figure 2, respectively. The PortAudio library is at the backbone of their implementation. Another pair of devices is also available for playing audio files stored on disk, and for recording audio to a file.

4.2. Speech Synthesizer

The TTS devices generate audio from the “robot” side of the conversation during two phases: to signalize the user that their prompt was heard and the robot is ready to accept the question, and to announce the result of the LLM block. A first device was implemented using the eSpeak engine, which features a phonemizer that helps reducing the size of voice models. At the downside, the resulting speech is neither natural nor smooth. To overcome this, an alternative device was developed using the Piper library, whose models are trained with the ONNX inference engine based on human speech recordings. A fork was derived from the original repository to wrap Piper’s API in a convenient C++ library (Hansen, 2025).

4.3. Speech Transcription

As soon as the chat application reports readiness, the ASR device starts transcribing the user’s voice into text to be fed into the LLM module. It is implemented using the Vosk toolkit powered by Kaldi (Povey et al., 2011).

4.4. Wake Word Module

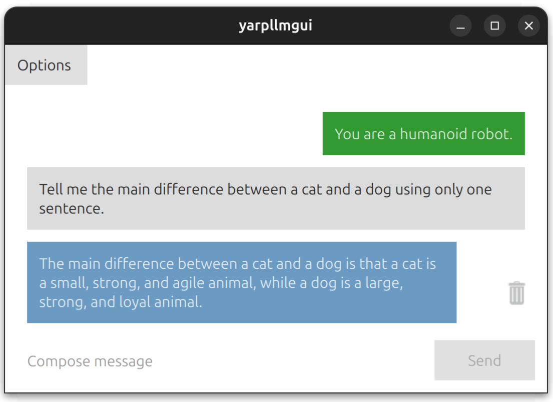
This specialized ASR devices implements the openWakeWord framework to perform inference through ONNX for detecting short voice sentences. For the purpose of this work, the “hey, TEO” phrase was trained from a sample synthesized by Piper. The wake word module initiates the execution flow of the main application, signaling to the following phases that the user wants to start a conversation.

4.5. Large Language Model

The conversational agent (or chatbot) has been implemented using the llama.cpp project by Georgi Gerganov, an inference framework for various large language models. It is optimized for on-device (i.e. at the edge) transformer model inference, thus allowing to offload most of the computing power to a dedicated edge server aboard the robot. In the proposed application, this device accepts the result of the ASR voice detection module, and forwards the text response to the TTS device.

At the time of writing, the llama.cpp framework supports a range of models, including variants of LLaMa (Llama), Mistral, GPT and DeepSeek, among others. Distilled versions of DeepSeek models with a reduced number of parameters have been targeted in this work.

Figure 4 depicts an example conversation through this device carried out with the *yarpllmgui* graphical interface installed alongside YARP.



You are a humanoid robot.

Tell me the main difference between a cat and a dog using only one sentence.

The main difference between a cat and a dog is that a cat is a small, strong, and agile animal, while a dog is a large, strong, and loyal animal.

Figure 4: Chat completion GUI app.

5. Chat Completion App

The execution flow of the chat application is depicted in Figure 5. The user and the "robot", i.e. the application, are the conversation actors. It is always initiated by the former, and conveniently acknowledged by the latter during the dialogue.

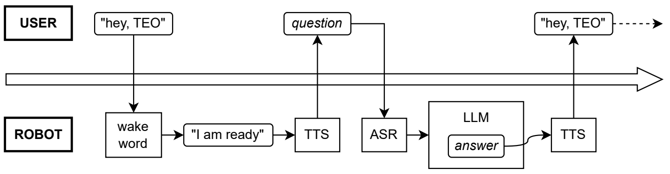


Figure 5: Execution flow of the proposed application.

The task is split in several phases detailed next:

1. A message is sent by the application to the Recorder module to initiate audio recording from the microphone. To avoid interference between TTS and ASR, another message is sent concurrently to the Player module to disable audio playback on the speaker.
2. The execution is blocked until the wake word device reports that the user has articulated the target phrase (e.g. "hey, TEO").
3. Audio playback is enabled, audio recording is disabled.
4. The robot acknowledges the reception of the wake word through the synthesizing of a pre-configured phrase (e.g. "I am ready to listen").
5. Audio playback is disabled, audio recording is enabled.
6. The ASR module performs transcribes the question asked by the user and stores it as text.
7. Audio playback and recording are disabled.
8. The transcribed question is forwarded to the LLM module and a text response is produced.
9. Audio playback is enabled, audio recording is disabled.
10. The text response is synthesized by TTS.

The algorithm is looped indefinitely until the user finalizes the program. Its implementation has been tested and made publicly available on the GitHub platform (RoboticsLab, 2025).

6. Results and Conclusions

The application has been tested using free models:

- TTS: medium quality "lessac" American English model phonemized with eSpeak and curated for Piper.
- ASR: lightweight American English model "vosk-model-small-en-us-0.15" for Android and Raspberry Pi.
- LLM: DeepSeek-R1 distilled from Qwen with 1.5 billion parameters, quantized using the bfloat16 floating-point format for better compatibility with llama.cpp.

Exact model versions used in the experiments are listed in the GitHub repository sources. It has been deemed convenient to post-process the output of the LLM module since DeepSeek-R1 includes a usually lengthy chunk of text before the actual result, solely devoted to reasoning about the asked question.

While it is possible to hold a conversation, it has been concluded that a dedicated edge server with high amounts of RAM memory would allow to run heavier LLM models, thus greatly increasing the quality of the chat.

Acknowledgments

The research leading to these results received funding from "iRoboCity2030-CM" (TEC-2024/TEC-62) of Comunidad de Madrid, Dirección General de Investigación e Innovación Tecnológica, 5696/2024; "iREHAB" (DTS22/00105) of Instituto de Salud Carlos III; and EU structural funds.

References

Allgeuer, P., Ali, H., Wermter, S., 2024. When robots get chatty: Grounding multimodal human-robot conversation and collaboration. In: International Conference on Artificial Neural Networks. Springer, pp. 306–321.

Bennewitz, M., Faber, F., Joho, D., Schreiber, M., Behnke, S., 2005. Multimodal conversation between a humanoid robot and multiple persons. In: Proc. of the Workshop on Modular Construction of Humanlike Intelligence at the 20th National Conferences on Artificial Intelligence (AAAI). pp. 1–8.

Csapo, A., Gilmartin, E., Grizou, J., Han, J., Meena, R., Anastasiou, D., Jokinen, K., Wilcock, G., 2012. Multimodal conversational interaction with a humanoid robot. In: 2012 IEEE 3rd Int. Conf. on Cognitive Infocommunications (CogInfoCom). pp. 667–672.
DOI: `10.1109/CogInfoCom.2012.6421935`

Dutoit, T., 1997. High-quality text-to-speech synthesis: An overview. Journal of Electricaland Electronics Engineering Australia 17 (1), 25–36.

Hansen, M., 2025. A fast, local neural text to speech system. https://github.com/roboticslab-uc3m/piper.

Hayashi, K., Kanda, T., Miyashita, T., Ishiguro, H., Hagita, N., 2008. Robot manzai: Robot conversation as a passive–social medium. Int. Journal of Humanoid Robotics 5 (01), 67–86.
DOI: `10.1142/S0219843608001315`

Kanda, T., Ishiguro, H., Imai, M., Ono, T., 2004. Development and evaluation of interactive humanoid robots. Proc. of the IEEE 92 (11), 1839–1850.
DOI: `10.1109/JPROC.2004.835359`

Khanam, F., Munmun, F. A., Ritu, N. A., Saha, A. K., Firoz, M., 2022. Text to speech synthesis: A systematic review, deep learning based architecture and future research direction. J. of Advances in Information Technology 13 (5).
DOI: `10.12720/jait.13.5.398-412`

Łukawski, B., Victores, J. G., Balaguer, C., 2023. A generic controller for teleoperation on robotic manipulators using low-cost devices. XLIV Jornadas de Automática, 785–788.
DOI: `10.17979/spudc.9788497498609.785`

Metta, G., Fitzpatrick, P., Natale, L., 2006. YARP: yet another robot platform. International Journal of Advanced Robotic Systems 3 (1), 43–48.
DOI: `10.5772/5761`

Milhorat, P., Lala, D., Inoue, K., Zhao, T., Ishida, M., Takanashi, K., Nakamura, S., Kawahara, T., 2019. A Conversational Dialogue Manager for the Humanoid Robot ERICA. Springer Int. Publishing, Cham, pp. 119–131.
DOI: `10.1007/978-3-319-92108-2_14`

Nassif, A. B., Shahin, I., Attili, I., Azzeh, M., Shaalan, K., 2019. Speech recognition using deep neural networks: A systematic review. IEEE access 7, 19143–19165.
DOI: `10.1109/ACCESS.2019.2896880`

Pérez Martínez, C., Pierro, P., Martinez, S., Pabon, L., Arbulú, M., Balaguer, C., 2010. RH-2: an upgraded full-size humanoid platform. In: Mobile Robotics: Solutions and Challenges. World Scientific, pp. 471–478.
DOI: `10.1142/9789814291279_0058`

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., Vesely, K., Dec. 2011. The Kaldi Speech Recognition Toolkit. In: IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. IEEE Signal Processing Society, iEEE Catalog No.: CFP11SRW-USB.

Radha, V., Vimala, C., 2012. A review on speech recognition challenges and approaches. World of Computer Science and Information Technology Journal (WCSIT) 2 (1), 1–7.

Rashad, M., El-Bakry, H. M., Isma'il, I. R., Mastorakis, N., 2010. An overview of text-to-speech synthesis techniques. Latest trends on communications and information technology, 84–89.

RoboticsLab, 2025. Text To Speech (TTS) and Automatic Speech Recognition (ASR). https://github.com/roboticslab-uc3m/speech.

Shi, C., Shiomi, M., Kanda, T., Ishiguro, H., Hagita, N., 2015. Measuring communication participation to initiate conversation in human–robot interaction. International Journal of Social Robotics 7, 889–910.