

End-User Personalisation of Humanoid Robot Behaviour Through Vocal Interaction

Simone Gallo^{1,2,*}, Giacomo Vaiani^{1,2,†} and Fabio Paternò¹

¹CNR – ISTI, Via G. Moruzzi 1, 56127 Pisa, Italy

²University of Pisa – Computer Science Dept., Largo B. Pontecorvo 3, 56127 Pisa, Italy

Abstract

This study explores the integration of Large Language Models with social robots to facilitate End-User Development through natural language interactions. The paper presents a prototype system embodied in a Pepper robot that allows non-expert users to customise robot behaviours by defining personalisation rules via vocal commands. This system employs trigger-action programming, enabling users to create automations based on specific triggers and actions without requiring in-depth technical knowledge. Through an example scenario, we show how users can program the robot by employing voice commands to execute actions when an event occurs. The created automations can also involve available IoT objects. The study investigates the potential of natural language interaction to improve the usability and flexibility of robot programming, offering new possibilities for personalised interactions in various settings.

Keywords

End-User Development, Human-Robot Interaction, Smart Spaces, CEUR-WS

1. Introduction

Over recent years, technological advancements have resulted in the development of increasingly sophisticated robots that are more closely integrated into human daily activities. This evolution is especially noticeable in the realm of social robots. These robots are designed to interact with humans in various social contexts, assisting with a range of tasks, including children’s language education [1], older adults’ cognitive training [2], and smart device management [3]. Moreover, robots could offer advantages over traditional voice assistants, particularly in routine activity detection and support for individuals with functional limitations [4][5][6]. Several end-user development tools have been introduced to facilitate user engagement and customisation of these robots’ behaviour. These tools utilise various paradigms, such as block-based [7] and natural language programming [8], enabling users to compose personalisation rules (for instance, having the robot say something when a person is in front of it or perform specific actions based on vocal commands).

Recent advancements in artificial intelligence, particularly with Large Language Models (LLMs), have the potential to enhance robots’ communicative and operational capabilities. This enables interactions that can resemble human-like conversations and dynamically adapt to end-user requests [9]. Within this context, trigger-action programming emerges as an effective approach to End-User Development (EUD) in robotic systems. It allows users to define robot behaviours in response to specific events or conditions, offering a user-friendly way to customise robot functionalities without the need for deep technical knowledge.

This paper presents a prototype of a conversational agent embodied in a Pepper robot that utilises an LLM to assist non-expert users in creating personalisation rules in trigger-

action format. Through vocal conversations, users can naturally express their preferences for how they want the robot to act when a specific event occurs. These events can be triggered by interacting with the robot itself (e.g., when the robot recognises a person), or by the surrounding smart environment (e.g., the change of temperature in a room, opening or closing windows/doors). In the following sections, we first introduce related work in the field of trigger-action programming for robot end-user development, and then we delve into the architecture of this prototype and an application scenario. Finally, we discuss the next step for this work.

2. Related Work

Various research studies have explored the possibilities of end-user programming of robot behaviour using the trigger-action paradigm, employing diverse interaction modalities and approaches. Leonardi et al. [10] exploited a graphical web-based wizard interface to enable users without programming skills to define personalisation rules by specifying events and/or conditions (*triggers*) that, once met, initiate the execution of defined actions. The tool allows the user to select triggers and actions related to smart devices (e.g., the motion detected by a sensor, turning on smart bulbs) and a Pepper robot (e.g., a touch on the robot’s head). Thus, it was possible to create automations, such as having Pepper say “Hey, how are you?” when someone entering the room is detected, by combining Internet of Things devices with Pepper. In the present study, we propose the development of automations through direct interaction with the robotic system, as opposed to the utilisation of a separate web-based wizard.

Another contribution by Porfirio et al. [11] presents Tabula, a multimodal end-user development system for programming service robots for personal use in domestic and workplace environments. In this case, the system enables users without programming skills to script tasks, defining humanoid robots’ behaviour (a Pepper one) using trigger-action programming and combining natural language with sketches on a visual interface to define the automation. In particular, users can utilise natural language commands (via voice) to define triggers and actions. The resulting automation is visualised on a two-dimensional map, displaying the

Workshop Robots for Humans 2024, Advanced Visual Interfaces, Arenzano, June 3rd, 2024

*Corresponding author.

†These authors contributed equally.

✉ simone.gallo@isti.cnr.it (S. Gallo); giacomo.vaiani@phd.unipi.it (G. Vaiani); fabio.paterno@isti.cnr.it (F. Paternò)

ORCID 0000-0003-3412-1639 (S. Gallo); 0009-0002-0910-2284 (G. Vaiani); 0000-0001-8355-6909 (F. Paternò)



© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

current environment (e.g., the user’s house) and the defined path or actions of the robot. This setup allows users to modify or refine the automation or to implement more complex logic that is difficult to express verbally. The implemented prototype encompasses a set of five actions (e.g., moving to a position, saying something) and two events (e.g., a person approaching or speaking to the robot), enabling the creation of automations like “when the user arrives home, the robot goes to the entrance”. Although users have considered the approach promising, the system faces challenges in processing natural language input due to difficulties in understanding complex or ambiguous commands, which leads to errors in automation and user frustration.

Finally, a recent work proposed by Karli et al. [12], developed a system integrating ChatGPT to enable end-users to define robot programs (e.g., for defining the movement of robotic harm) using natural language instructions. The system interface presents a chat from which the user sends the inputs, a console showing the generated code and a view of the robot simulator. Beginning with the description of the desired robot behaviour, the user engages in a collaborative process that iteratively defines and debugs the specifications with the system to address the request. While the natural language interaction is effective, the study emphasises several critical points regarding the use of LLMs in this context. It highlights the necessity of enhancing the reliability of LLM-generated code through accurate code verification processes, crafting more effective prompts and adjusting prompts dynamically to better fit the context.

In general, LLM approaches open new possibilities for applications across a broad range of settings in end-user development, highlighting the potential to enhance the usability and flexibility of robot programming.

3. Our Approach

In this proposal, we introduce the combined use of Pepper with an LLM agent, aiming to define the robot’s behaviour through specific trigger-action personalisation rules (also called automations) expressed by voice. By integrating an LLM as a natural language processing module, users can communicate with Pepper in a more intuitive and conversational way. This enhancement enables Pepper to process complex commands and questions, significantly improving its usability and interactive capabilities. Furthermore, this system design lets users create automations verbally, eliminating the need for any programming skills. Figure 3 illustrates the architecture of the designed system.

Specifically, this prototype aims to create automations that include triggers and actions related to both a smart environment (e.g. a smart home) and the robot. Through these automations, it is possible to define events, conditions, and actions related to both sensors and smart objects (e.g., motion sensors, smart light bulbs, smart thermostats) and Pepper (e.g., recognise a person, display something on its tablet, say something). In this way, the robot becomes part of a smart ecosystem in which it can perform actions in response to triggers related to the environment or be itself the trigger of events for the execution of actions by smart objects. On a technical level, Pepper can be considered a system entity at the same level as sensors and smart objects and can thus be integrated into control systems of smart environments. The automations created are then executed

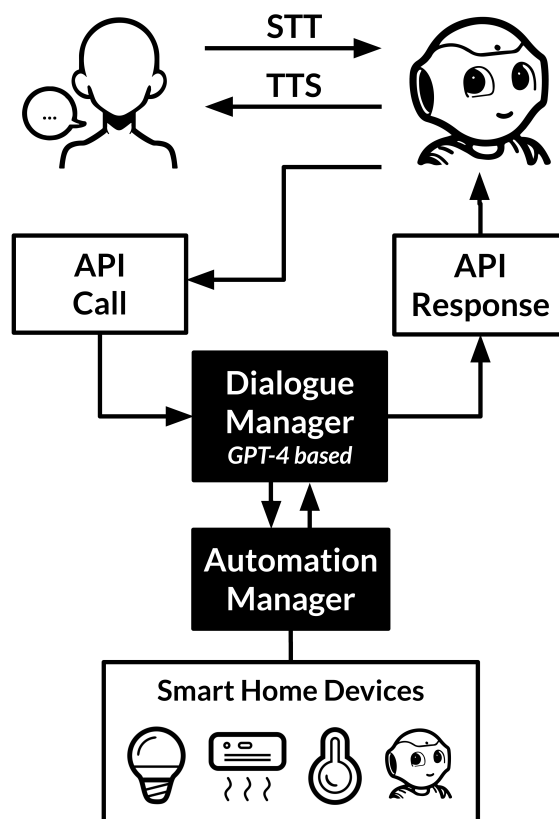


Figure 1: System Architecture

through an Automation Manager (we use Home Assistant¹ as Automation Manager because it is open-source, robust, and widely used with an active community). In particular, we consider the following triggers and actions involving Pepper:

- **Triggers:** chest button press, head touch, hand touch (both right and left hand), face recognition, emotion recognition, and speech recognition.
- **Actions:** speak, hand movement, run animation, position change, video camera activation, LEDs state change, show something on display.

When the user speaks, the robot utilises the Google Speech API to identify the spoken sentence. The system then makes an API call to the Dialogue Manager to process the user message. The API response, which is delivered using Pepper’s voice, contains the response message for the user. Once the automation is complete it is saved in a database and executed through the automations manager.

More generally, our approach opens the possibility of creating automations that involve robots and surrounding connected sensors and objects. Indeed, the created automations can involve both triggers and actions associated with the robot (e.g., when Pepper recognises a specific person, says “Hello [name]” and does a greeting animation), but it is also possible to have triggers activated by external sensors or objects and actions executed by the robot. Vice versa, triggers can be generated by the robot, with actions executed by surrounding objects (e.g., When Pepper detects a

¹<https://www.home-assistant.io/>

negative emotion on the user’s face, soft lights turn on, and relaxing music plays.).

3.1. Interacting with Pepper

The application integrates voice and text messaging functionalities, as well as message display editing. The vocal interaction approach utilises Google Cloud’s Text-to-Speech (TTS) and Speech-to-Text (STT) APIs. This approach offers a comprehensive solution for adding vocal input and output capabilities, thereby enhancing the user experience by making interactions more natural and engaging.

Users can activate voice recognition in the robot by pressing a designated button on the interface or touching a part of the robot’s body, such as the hand or head. This service converts vocal input into text. Initially, the service starts recording audio via the robot’s microphone, visually notifying the user of the recording status with a change in button colour to indicate when the recording is active. Next, the service sends the audio stream to the Google STT service. The speech service promptly notifies the robot application when receiving the input converted into text format, allowing for almost instantaneous processing of the transcribed message. The obtained text can be used for various purposes within the robot application, such as displaying it for the user, sending it to other backend services for further responses or actions, or triggering specific commands based on keywords.

The robot application’s audio output is generated by a TTS functionality that handles authentication to the Google Cloud service, manages synthesis requests, and ultimately plays the resulting audio using a media player. The system saves the synthesised audio to a local file, readying it for playback. This conversion occurs after the message has been processed and sent to the list of displayed messages, ensuring that the user can both see and hear the bot’s response. A useful feature introduced is the toggle functionality that allows users to choose between viewing all messages exchanged in the chat and viewing only the last two messages (the application’s default view). The toggle is triggered by a method that flips a Boolean value, based on which the adapter decides which view to adopt. This mechanism helps keep the user interface tidy, showing only a part of the messages during vocal interaction but allowing users to view the entire conversation, if desired. This decision was made under the assumption that during a real-time vocal conversation, users do not always need a textual representation of the entire chat.

Finally, the interface also features a reset function, allowing users to activate a series of actions through a dedicated button to stop any ongoing activity, such as vocal recording or the playback of animations and vocal synthesis. Moreover, this functionality serves to clear the user interface by deleting the message history and returning any input fields or selections to their original state. This reset feature ensures a smooth and intuitive user experience by allowing users to easily reset the application without the need to navigate through complex menus or restart the app entirely.

3.2. Dialogue Manager

The Dialogue Manager serves as the core component for processing natural language inputs and managing conversational flow. When a user engages with the robot, Pepper transcribes the user’s speech into text and then sends it to

the Dialogue Manager via an HTTP request. Upon receiving the text, the Dialogue Manager (a Flask Python server) forwards the message to the GPT-4 model through the OpenAI API. Guided by the instructions in the defined prompt, the model determines the next step in the conversation. It can either execute a function to perform a specific task or directly generate a textual response to the user’s query.

The constructed prompt begins with a description of the role the model is expected to adopt (e.g., *“You are Pepper, a humanoid robot...”*), which is succeeded by an explanation of the task (e.g., *“Your task is to help users create automations...”*) and some general guidelines to be adhered to when interacting with the user (e.g., *“call the user by the name”* or *“keep the response short and simple...”*). Then, the prompt introduces the functions the model can use, along with instructions on when and how to use these (e.g., *“always use the verify_automation function before saving the automation”*). The function calling functionality, provided by the OpenAI API, enables the LLM to interface with external resources and tools. This is possible by supplying the model with a set of function descriptions and the required parameters for their execution. In particular, we include a function for retrieving the list of possible triggers and actions for defining an automation, a function to verify the correctness of the defined automation, and a function for saving the created automation in a database. When a function is invoked, its output is fed back into the model. This becomes the basis for GPT-4 to generate an appropriate response. For example, the *“save automation”* functions provide an output message containing the unique automation ID as well as a confirmation message if the operation was successful or an error message otherwise. The model utilises this output to generate the user’s response. This response is then dispatched to Pepper, closing the loop of the initial HTTP request. Pepper then verbalises the response, providing the user with an audible answer.

Example Scenario. Let’s consider a usage scenario in which the user talks with Pepper and defines one automation by saying something like: *“When I come back home if I’m in a bad mood, say something comforting”*. Consequently, Pepper will send this sentence to the Dialogue Manager that retrieves the list of possible triggers and actions and proposes the users an initial automation: *“We can detect when you are home using the location of your smartphone, and I can detect your mood by your facial expressions. If you are sad or angry, I can put on relaxing music and say something comforting like...”*.

At this point, the user can continue talking with Pepper to refine the proposed automation. Once the user is satisfied with the defined triggers and actions, the Dialogue Manager uses the *“verify automation”* function to check that the automation contains only available triggers and actions. If an automation is correctly verified, Pepper asks the user if the created automation can be saved. After saving, a confirmation message resumes the created automation along with the assigned ID in the database. Once automation is saved, the Automation Manager module executes the defined actions when the chosen event is triggered, and the defined conditions are eventually met.

4. Conclusions and Future Work

This research explores the integration of Large Language Models into the domain of End-User Development for social robots, focusing on enabling users to customise robot behaviours through intuitive, natural language vocal interactions. By implementing a prototype conversational agent embodied in a Pepper robot, we facilitate non-expert users in creating automation for personalising the robot behaviour based on events in a smart environment (e.g., presence detection in a room), or on events on the robot itself (e.g., human face recognition). This approach leverages trigger-action programming, presenting a user-friendly method for customising robot functionalities without requiring technical knowledge. Our proposal contributes to the field by illustrating the practical application of LLMs in enhancing robot usability and flexibility, suggesting a promising avenue for future research and development in social robotics and user-centric automation. For future work, we plan to initially conduct user tests in a controlled environment (e.g., laboratory setting) to evaluate the strengths and weaknesses of our solution in comparison with existing tools based on visual interfaces. User tests in real-world scenarios will follow, addressing the need for realistic, extended evaluations in robotics. Given the focus on the End-User Development approach, it is important to test with users without programming skills and home automation experience.

References

- [1] J. de Wit, T. Schodde, B. Willemsen, K. Bergmann, M. de Haas, S. Kopp, E. Kraemer, P. Vogt, The Effect of a Robot's Gestures and Adaptive Tutoring on Children's Acquisition of Second Language Vocabularies, in: Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, HRI '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 50–58. URL: <https://dl.acm.org/doi/10.1145/3171221.3171277>. doi:10.1145/3171221.3171277.
- [2] M. Manca, F. Paternò, C. Santoro, E. Zedda, C. Braschi, R. Franco, A. Sale, The impact of serious games with humanoid robots on mild cognitive impairment older adults, *International Journal of Human-Computer Studies* 145 (2021) 102509. URL: <https://www.sciencedirect.com/science/article/pii/S1071581920301117>. doi:10.1016/j.ijhcs.2020.102509.
- [3] H.-D. Bui, N. Y. Chong, An Integrated Approach to Human-Robot-Smart Environment Interaction Interface for Ambient Assisted Living, in: 2018 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO), 2018, pp. 32–37. URL: <https://ieeexplore.ieee.org/document/8625821>. doi:10.1109/ARSO.2018.8625821, iSSN: 2162-7576.
- [4] N. Ramoly, A. Bouzeghoub, B. Finance, A Framework for Service Robots in Smart Home: An Efficient Solution for Domestic Healthcare, *IRBM* 39 (2018) 413–420. URL: <https://www.sciencedirect.com/science/article/pii/S1959031818302793>. doi:10.1016/j.irbm.2018.10.010.
- [5] E. Toscano, M. Spitale, F. Garzotto, Socially Assistive Robots in Smart Homes: Design Factors that Influence the User Perception, in: 2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2022, pp. 1075–1079. URL: <https://ieeexplore.ieee.org/document/9889467>. doi:10.1109/HRI53351.2022.9889467.
- [6] G. Wilson, C. Pereyda, N. Raghunath, G. de la Cruz, S. Goel, S. Nesaei, B. Minor, M. Schmitter-Edgecombe, M. E. Taylor, D. J. Cook, Robot-enabled support of daily activities in smart home environments, *Cognitive Systems Research* 54 (2019) 258–272. URL: <https://www.sciencedirect.com/science/article/pii/S1389041718302651>. doi:10.1016/j.cogsys.2018.10.032.
- [7] Towards a Modular and Distributed End-User Development Framework for Human-Robot Interaction | IEEE Journals & Magazine | IEEE Xplore, ??? URL: <https://ieeexplore.ieee.org/document/9323043>.
- [8] S. Beschi, D. Fogli, F. Tampalini, CAPIRCI: A Multi-modal System for Collaborative Robot Programming, volume 11553, Springer International Publishing, Cham, 2019, pp. 51–66. URL: http://link.springer.com/10.1007/978-3-030-24781-2_4. doi:10.1007/978-3-030-24781-2_4, book Title: End-User Development Series Title: Lecture Notes in Computer Science.
- [9] S. Vemprala, R. Bonatti, A. Bucker, A. Kapoor, ChatGPT for Robotics: Design Principles and Model Abilities, 2023. URL: <http://arxiv.org/abs/2306.17582>. doi:10.48550/arXiv.2306.17582, arXiv:2306.17582 [cs].
- [10] N. Leonardi, M. Manca, F. Paternò, C. Santoro, Trigger-Action Programming for Personalising Humanoid Robot Behaviour, in: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1–13. URL: <https://dl.acm.org/doi/10.1145/3290605.3300675>. doi:10.1145/3290605.3300675.
- [11] D. Porfirio, L. Stegner, M. Cakmak, A. Sauppé, A. Albarghouthi, B. Mutlu, Sketching Robot Programs On the Fly, in: Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction, HRI '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 584–593. URL: <https://dl.acm.org/doi/10.1145/3568162.3576991>. doi:10.1145/3568162.3576991.
- [12] U. B. Karli, J.-T. Chen, V. N. Antony, C.-M. Huang, Alchemist: LLM-Aided End-User Development of Robot Applications, in: Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction, HRI '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 361–370. URL: <https://dl.acm.org/doi/10.1145/3610977.3634969>. doi:10.1145/3610977.3634969.