Personal and Ubiquitous Computing
https://doi.org/10.1007/s00779-024-01825-5

ORIGINAL PAPER

A conversational agent for creating automations exploiting large language models

Simone Gallo[1,2] · **Fabio Paternò**[1] · **Alessio Malizia**[2]

Received: 4 March 2024 / Accepted: 21 June 2024
© The Author(s) 2024

Abstract
The proliferation of sensors and smart Internet of Things (IoT) devices in our everyday environments is reshaping our interactions with everyday objects. This change underlines the need to empower non-expert users to easily configure the behaviour of these devices to align with their preferences and habits. At the same time, recent advances in generative transformers, such as ChatGPT, have opened up new possibilities in a variety of natural language processing tasks, enhancing reasoning capabilities and conversational interactions. This paper presents RuleBot + +, a conversational agent that exploits GPT-4 to assist the user in the creation and modification of trigger-action automations through natural language. After an introduction to motivations and related work, we present the design and implementation of RuleBot + + and report the results of the user test in which users interacted with our solution and Home Assistant, one of the most used open-source tools for managing smart environments.

Keywords Conversational interfaces · Trigger-action automations · Smart spaces

1 Introduction

In the era of ubiquitous computing, everyday environments are increasingly becoming augmented with a myriad of sensors and smart Internet of Things (IoT) devices,[1] aiming to enhance the quality of life by providing personalised and context-aware services. While these advancements offer significant convenience, they also introduce complexity in terms of device management and customisation. Non-expert users often find themselves overwhelmed by the technicalities involved in setting up and maintaining the desired behaviour of their smart environments. Many lack programming experience and are afraid of breaking the system, leading the user to rely on the default settings. Even when users do attempt customisation, the unintuitive features of smart systems and difficulties in translating user intentions into automation add to the frustration. In addition, the need for ongoing adjustments to meet the changing user preferences further complicates the user experience. As a result, these issues discourage users from fully utilising their smart devices, making the technology less accessible and user-friendly [1–3].

To address this gap, there has been a growing interest in developing more intuitive methods that allow users to interact with and automate their smart objects without requiring technical knowledge and/or programming experience. This interest aligns with the broader field of end-user development (EUD), where the emphasis is on creating environments that allow non-expert users without programming skills to create or modify their applications.

Leveraging the recent breakthroughs in natural language processing, particularly the advent of generative transformer models, such as GPT-4, we think that conversational agents can play a crucial role in simplifying the process of creating and managing smart home automations. These agents have the potential to deeply understand and generate natural

---

[1] https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/

✉ Simone Gallo
simone.gallo@isti.cnr.it

Fabio Paternò
fabio.paterno@isti.cnr.it

Alessio Malizia
alessio.malizia@unipi.it

1 CNR-ISTI, Pisa, Italy

2 Department of Computer Science, University of Pisa, Pisa, Italy

Published online: 04 July 2024

language, enabling a more intuitive and accessible interface for users.

This paper introduces RuleBot++, a conversational agent designed to assist users in creating and modifying smart home automations using the trigger-action paradigm. RuleBot++ utilises the capabilities of GPT-4[2] to process user input in natural language and generate the corresponding automation, thereby lowering the barrier for the users in customising their smart environments.

To evaluate the effectiveness of RuleBot++, we compared it with Home Assistant (one of the leading open-source platforms for smart home management and automation) through a user test with 16 participants.

The comparison can be useful to derive insights into the strengths and weaknesses of each approach. This helps in understanding how exploiting natural language interaction can enhance user experience and usability in smart home automation.

The following research questions guide our investigation:

- How does the use of a conversational agent for creating smart home automations, such as RuleBot++, compare to traditional visual environments, such as Home Assistant, in terms of usability?
- What are the issues faced by the users when creating and modifying automations through natural language interfaces, also with respect to form-based visual interfaces?
- Does the usability of the conversational agent vary depending on the complexity of the automation created?

By investigating these questions, we seek to contribute to the ongoing research on enhancing user experience in smart home ecosystems by proposing a chatbot architecture based on a large language model (LLM) and evaluating our solution in comparison to an open-source state-of-the-art tool for managing home automations.

2 Related work

During the last few years, some proposals have been put forward in the field of conversational agents applied to the Internet of Things environments. Beyond the well-known commercial solutions such as Alexa and Google Home, some research contributions proposed a conversational approach for controlling smart objects and energy consumption, giving assistance to the elderly or impaired, and creating and modifying automations in trigger-action format [4].

In particular, concerning the solutions for managing automations, Roffarello et al. [5] propose a voice-based agent that allows users to create automations based on the Google ecosystem. Starting from a user description of the desired behaviour, the agent can create simple automations composed of one trigger and one action and also provide the user with the possibility to modify the trigger, the action, or the entire created rule. A similar application was proposed by Barricelli et al. [6], who developed a multimodal agent (combining voice and touch) based on the Alexa ecosystem using Amazon devices equipped with a monitor. In this case, the user can create automations composed of one trigger and multiple actions, switching between voice or touch interaction based on the user's need. Concerning text-based solutions, Jarvis [7] is a chatbot for managing smart environments through the execution of instant actions (e.g. "turn on the bedroom lights") and by creating simple automations composed of a trigger and an action. One interesting Jarvis feature concerns the possibility of asking "causality queries" such as "Why did the light turn on?" thus providing a form of explanation to the user.

Regarding the creation of more complex automations, the earliest RuleBot version [8] aimed to address the problem by implementing an architecture that allows the user to create automations in trigger-action format composed of multiple triggers (distinguishable in events and conditions) and multiple actions. In fact, exploiting grammatical rules, RuleBot was able to split complex commands (e.g. "turn on the light when entering the kitchen and if it's past 6 p.m.") into simpler chuck (e.g. "turn on the light", "when entering the kitchen", "and", "if it's past 6 p.m.") to be easily classified.

All the mentioned contributions share the same problems: correctly classifying the user input to the right intent and efficiently managing the conversational breakdown during the interaction. The advances in natural language processing with large language models such as ChatGPT opened new possibilities for overcoming these problems. For example, a recent version of RuleBot [9] implements a hybrid architecture using the Rasa Framework combined with ChatGPT, using the LLM for the management of specific complex tasks (i.e. splitting complex commands into simpler single commands and managing the conversational breakdown to be informative) while applying "classical" machine learning intent classification, entity extraction, and dialogue management with the support of Rasa.

RuleBot++ aims to address the challenges outlined in previous work, overcoming the limitations given by the rigid and somehow unnatural conversation of a classical conversational agent during the creation and modification of smart home automations.

---

[2] In particular, we use the model GPT-4 Turbo 1106-preview.

3 The proposed solution

RuleBot+ +aims to enable non-expert users to create and modify smart home automations simply and intuitively through conversation. Based on a user's request, which describes the desired automation behaviour, RuleBot+ +can "translate" this description into executable smart home automation.

Through these automations, the user can define the actions that sensors and smart objects should perform when a certain trigger occurs within the context of a smart home. We have connected RuleBot+ +to Home Assistant, which is employed as automation engine since it offers a robust and open-source system for configuring smart devices and managing automation execution. Therefore, RuleBot+ +generates automations in YAML format that can be executed by the Home Assistant engine. During the user session, RuleBot+ +can identify which sensors and smart objects have been installed in the smart home by accessing a JSON file that describes the home configuration, thereby guiding the user in selecting and setting the desired event, conditions, and actions. The RuleBot+ +'s engine leverages the powerful natural language understanding, processing, and generation capabilities of GPT-4, paired with the execution of Python functions (through the function calling model's feature) that enable RuleBot+ +to access devices present in the smart home, save automations in a database, retrieve automations for modification or explanation, and verify automations YAML syntax correctness.

3.1 Architecture

The architecture of RuleBot+ +(Fig. 1) can be divided into three main components: the *user interface* from which the user can send and receive messages and can access the list of created automations, the *conversation manager* that implements the chatbot logic through the GPT-4 API calls and the execution of the developed Python functions, and the *information management* composed by a database that contains the automations created by the user and the conversation logs with RuleBot+ +and a JSON file that indicates the configuration of the smart home with its sensors, smart objects, and services and their relative unique IDs.

3.1.1 User interface

The user interface (Fig. 2) is accessible from any web browser and is composed of a classical front-end (HTML, CSS, and JS) and a Node.js back-end server that implements the logic for user registrations and logins, displays the automations created by users, and manages the sending and receiving of messages between the user and RuleBot+ +.

After logging in, the interface is divided into two main sections. On the left, users can find their list of created automations along with the automation name, the automation ID, and the automation description. The right side presents the chat, where the user can type the message and visualise the chatbot's response. When the user logs in to RuleBot+ +, a welcome message is automatically sent to the user to start the conversation.

3.1.2 Conversation manager

The conversation manager is developed in Python and implements the calls to the OpenAI API, as well as the definition and execution of the function that provides the capabilities of RuleBot+ +. In this architecture, the GPT-4 model replaces the "intent classification" and "dialogue management" modules of a classic conversational agent based on machine learning [10], highlighting the strengths (but also the weaknesses) of LLMs, which will be addressed in the following paragraphs. The GPT model behind RuleBot+ +is guided by the instructions present in the system prompt, describing the context and expectations for the ongoing conversation and providing an explanation of how to act upon certain requests (such as creating or modifying and automation). Moreover, the function calling feature provided by OpenAI allows us to provide the model with a set of function descriptions, along with the necessary parameters to execute them. Through this feature, when the user sends a message requiring the use of external resources to be answered (e.g. databases, third-party APIs), the model can request the execution of the relevant function for retrieving the needed information or, more in general, fulfil the user request. The output returned by the function will then be used to generate a coherent response to the user's request. We have defined and implemented the following Python functions that RuleBot+ +can access for managing users' automation:

- Retrieve Entities: Receives as input the type of entity (e.g. light) and/or the reference room and returns the list of entities of a certain type or the entities present in a room. It can also return the entire configuration of the home.
- Verify Automation: Receives as input an automation in YAML format and returns a success message if all entities involved in the automation are present in the context of the home and if the YAML code is correctly formatted. Otherwise, it returns an error message, such as when the user requests the activation of a device that is not present in the smart home. The error message contains the non-existing entities that generated the error to provide a more exhaustive message for the user.

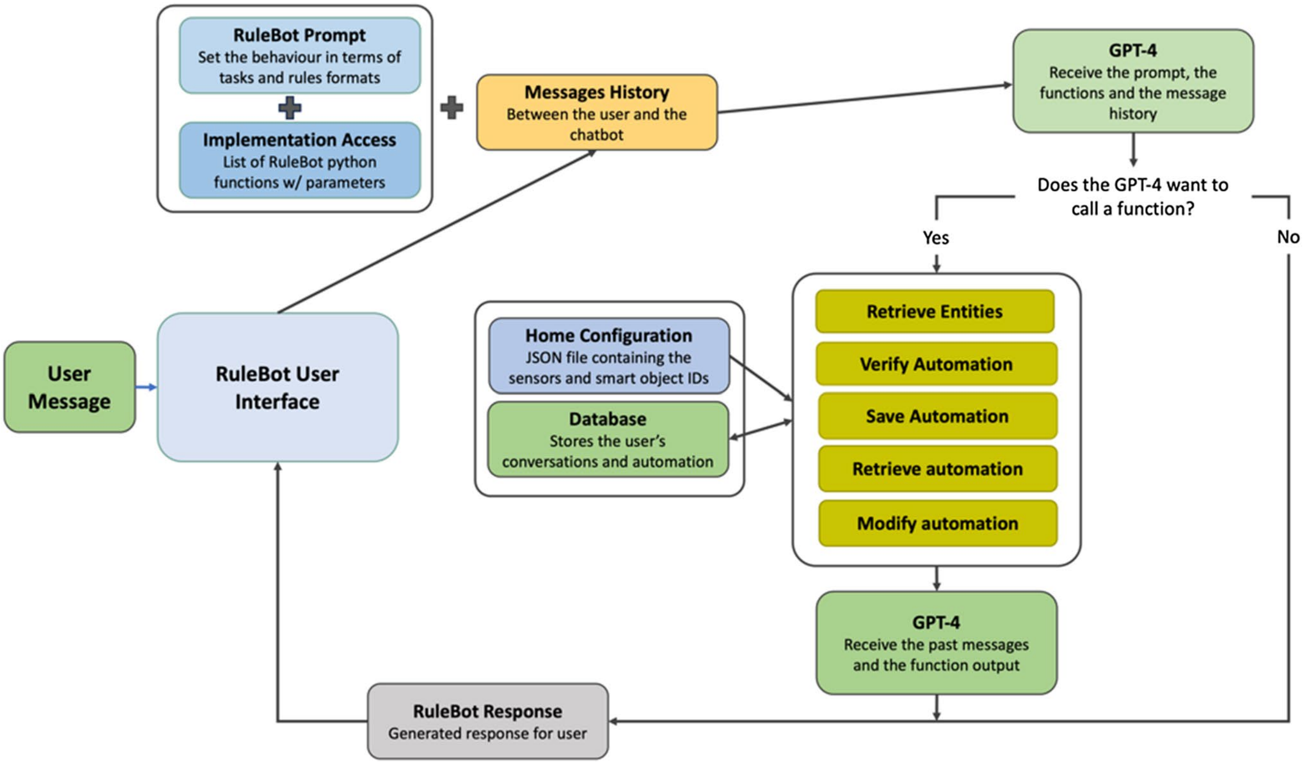
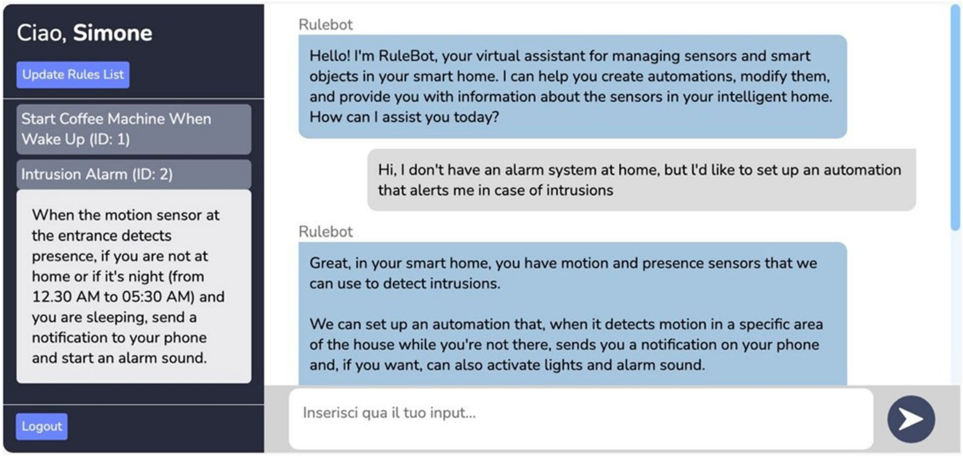

RuleBot Prompt
Set the behaviour in terms of tasks and rules formats
Implementation Access
List of RuleBot python functions w/ parameters
Messages History
Between the user and the chatbot
GPT-4
Receive the prompt, the functions and the message history
Does the GPT-4 want to call a function?
Yes
No
User Message
RuleBot User Interface
Home Configuration
JSON file containing the sensors and smart object IDs
Database
Stores the user's conversations and automation
Retrieve Entities
Verify Automation
Save Automation
Retrieve automation
Modify automation
GPT-4
Receive the past messages and the function output
RuleBot Response
Generated response for user

Fig. 1 The RuleBot+ +architecture

- Save Automation: Takes as input the automation in YAML format (executable by Home Assistant), the input phrase the user used to describe the automation, a description of the automation generated based on the YAML, and the name to give the automation.
- Retrieve Automation: Given the name or the ID of an automation, it returns the information to generate an explanation or make modifications. It can also return the complete list of a user's automations.
- Modify Automation: Starting from an already created automation (identified by an ID), it modifies the automation YAML, the description, and the name based on the user request.

When the user requests the creation of an automation, RuleBot+ +first executes the "retrieve entities" function. Based on the available smart devices, RuleBot+ +proposes a possible automation configuration to the user, explaining which devices would be involved. This is followed by a refinement phase in which the user and the chatbot collaborate until the user is satisfied with the automation. At this point, RuleBot+ +uses the "verify automation" function to check the correctness of the YAML automation. If it presents a formatting error, the user is not informed, but the model is instead prompted to generate a new YAML code, indicating the lines in which the errors are detected;

Fig. 2 RuleBot+ +web user interface



Ciao, **Simone**

Start Coffee Machine When Wake Up (ID: 1)

Intrusion Alarm (ID: 2)

When the motion sensor at the entrance detects presence, if you are not at home or if it's night (from 12.30 AM to 05:30 AM) and you are sleeping, send a notification to your phone and start an alarm sound.

Rulebot

Hello! I'm RuleBot, your virtual assistant for managing sensors and smart objects in your smart home. I can help you create automations, modify them, and provide you with information about the sensors in your intelligent home. How can I assist you today?

Hi, I don't have an alarm system at home, but I'd like to set up an automation that alerts me in case of intrusions

Rulebot

Great, in your smart home, you have motion and presence sensors that we can use to detect intrusions.

We can set up an automation that, when it detects motion in a specific area of the house while you're not there, sends you a notification on your phone and, if you want, can also activate lights and alarm sound.

meanwhile, if the function returns an error related to missing entities, the chatbot informs the user and proposes possible alternatives. When the automation is successfully verified, RuleBot + + informs the user by providing a description of the automation in natural language and asks for confirmation to save it. After user confirmation, the "save automation" function is executed to update the database with the new automation.

The process for modifying an automation is similar. RuleBot + + first uses the "retrieve automation" function to obtain the requested automation from the database. After applying the desired modification, the automation is subjected to the verification process and then saved to the database using the "modify automation" function. The functions "retrieve entities" and "retrieve automation" are also used when the user seeks information and explanations.

3.1.3 Information management

The functionalities for saving, modifying, and retrieving automations have access to a MongoDB database from which the chatbot can retrieve and insert the information needed (e.g. new automation). All the operations return a result used by the chatbot to inform the user in case of errors or confirm the operation's success (e.g. after successfully saving an automation return a confirmation message along with the ID assigned to the automation).

The information regarding the smart home configuration is encoded in a JSON file containing the list of sensors and devices along with their unique identifiers. We explored different ways to describe the smart home in the JSON file. Initially, we structured the configuration by dividing it into rooms. For each room, we listed the various sensors and devices present. This approach allowed for an intuitive and organised representation, making it easy to query the file to get the sensors and devices located in each specific room. For instance, the living room might have been detailed with entries for a temperature sensor, a smart TV, and smart lights, each with their unique identifiers. This room-based structuring is effective for scenarios where the interactions and automations were predominantly confined within single rooms. However, this approach results in a lengthy and repetitive description file when certain types of sensors or devices are present in multiple rooms throughout the house (e.g. every room can have a temperature sensor, a smart light, and a motion sensor). When the model requires the entire file (e.g. when the user asks which sensors are present in the house or if the user is unclear about which rooms are to be involved in the automation), it consumes many tokens, slowing down the process execution, especially when requested multiple times during the conversation. To address this, we moved into organising the JSON file based on the type of devices and sensors. Using this approach the device or sensor type is the primary organising principle, with each type being a key at the highest level (e.g. "lights", "motion_sensors", "temperature_sensors"), under which the list of rooms in which the device is present is contained as well as a unique identifier. In this case, the identifier is composed of a fixed and a variable part, filled with the name of the room at runtime (e.g. the identifier for a motion sensor is "binary_sensor.motion_[room]", where "[room]" is replaced by one of the listed rooms).

3.1.4 Prompting the model

Prompting a large language model refers to the process of designing and refining the initial input to instruct the model on how to behave and to guide its responses and actions. Despite different prompting techniques have been developed to enhance the models' capabilities (e.g. chain of thought, few-shot learning [11, 12]), and despite the development of general guidelines on prompt engineering,[3] creating a prompt for a model remains a task that requires a trial-and-error approach.

The composition of the prompt for RuleBot + + followed an iterative process, starting from brief and simple prompts to very long and detailed ones. Initially, prompts were limited to defining the identity (e.g. "You are RuleBot, a conversational agent for creating automations…") and a generic description of the designated tasks (e.g. "you can create and modify automations, retrieve the list of sensors…"). This led to the model exhibiting different behaviours with each new interaction, such as inconsistency in calling functions from the same user message. We then evolved to a more extensive, detailed, and structured definition of the tasks and their respective steps, achieving consistent interactions across different sessions. It is important to note that, given the non-deterministic nature of these models, the chatbot's response can slightly vary at the syntactic and lexical level. To limit this variability, we set the model's sampling temperature[4] to 0. The final prompt includes an initial part where the RuleBot + + 's identity and a high-level description of its functionalities are defined, followed by a set of generic rules that the chatbot must apply during conversations with the user. The prompt continues by defining the necessary steps and rules

---

[3] https://platform.openai.com/docs/guides/prompt-engineering

[4] The temperature parameter (a value between 0 and 2) influences the model randomness in the selection of the next token. With a high value, the output is more random (https://platform.openai.com/docs/api-reference/chat/create).

to correctly manage the conversation (for the complete prompt see Appendix).

In addition, further instructions are added at the end of the functions output to contextually guide the model in the response generation. For instance, upon successfully saving automation, the “save automation” function includes the following instructions:

> The response message must contain the automation description in natural language following this pattern: When [here the events], If [here the conditions], [here the actions].

Similarly, appended to the output of the “retrieve entities” function is the following:

> The motion sensors can be utilized to detect the presence in a room. Do not include the 'entity_id' in the message generated for the user.

3.1.5 Example of a conversation with RuleBot + +

The following is an example of a conversation session between a user and RuleBot + + for the creation of one automation. The *System* messages describe the internal processes of the chatbot.

> **RuleBot ++:** Hello! I’m RuleBot, your virtual assistant for managing sensors and smart objects in your smart home. I can help you create automations, modify them, and provide you with information about the sensors in your intelligent home. How can I assist you today?
>
> **User:** Hi, I don’t have an alarm system at home, but I’d like to set up an automation that alerts me in case of intrusions.
>
> **System:** The function “retrieve entities” is executed to retrieve the full list of sensors and smart objects in the home. The output is used to generate the next RuleBot + + message.
>
> **RuleBot ++:** Great, in your smart home, you have motion sensors that we can use to detect intrusions.
>
> We can set up an automation that, when it detects motion in a specific area of the house while you're not there, sends you a notification on your phone and, if you want, can also activate the alarm. To proceed, I need to know in which area of the house you want to detect motion (like the entrance, kitchen, living room, etc.) and if you want the alarm to sound only under certain conditions, such as at a specific time or only when you’re not at home. Could you provide me with this information to create the automation?
>
> **User:** Let’s use the sensor at the entrance. I’d like to activate it when I’m not at home or when I’m sleeping at night.
>
> **RuleBot ++:** I’ve got it! We’ll set up the automation so that the motion sensor at the entrance detects presence when you’re not at home or if it’s night and you’re sleeping.
>
> To do this, we can use your phone’s tracker to know if you’re away from home and the presence sensor on your bed to determine if you’re sleeping. When the motion sensor detects presence under these conditions, we can set up a notification to be sent to your phone and, if you wish, we can also trigger an alarm sound through the audio player. Does that work for you?
>
> Before proceeding with the configuration, could you specify the time range you consider as “night” for your home? And what message would you like to receive in the notification on your phone?
>
> **User:** Alright, that’s fine… set it from 12:30 am to 05:30 am and use a generic message for the notification.
>
> **System:** The model generates the YAML code of the automation and the function “verify automation” is executed: the code is successfully verified, meaning it contains only existing devices in the home and is well formatted. The YAML code is assigned to a variable eventually used when saving the automation.
>
> **RuleBot ++:** The automation has been successfully verified. Here’s the summary of your rule:
>
> *Trigger*: When the motion sensor at the entrance detects presence.
>
> *Conditions*: If you are not at home (as detected by your phone’s tracker) or if it’s night (from 12.30 AM to 05:30 AM) and the presence sensor on the bed indicates you are sleeping.
>
> *Actions*: Send a notification to your phone and start an alarm sound through the audio player.
>
> Would you like me to proceed with saving this automation?

User: Yes.

System: The function “save automation” is executed, creating a new element in the database containing the automation ID, the previously verified YAML code, the name and the description of the automation. The function returns a success message containing the ID of the automation.

RuleBot++: The “Intrusion Alarm” automation has been successfully saved with the following ID: 1.

Here is the description of the automation in natural language:

“When the motion sensor at the entrance detects presence, if you are not at home or if it’s night (from 12.30 AM to 05:30 AM) and you are sleeping, send a notification to your phone with the message ‘Intrusion Alert!’ and start an alarm sound.”

If you need any more automations or modifications, I’m here to help!

4 User test

To better understand the system’s usability and the differences between a conversational approach and a traditional graphical interface, we conducted a user test comparing RuleBot+ +with the visual interface available on Home Assistant. It is the reference open-source software platform for smart environments (more than 340,000 active installations until May 2024[5]), designed for managing a wide range of sensors and smart devices from different manufacturers, allowing the creation, modification, and execution of automations.[6]

Home Assistant offers two ways to create automations. In YAML configuration, experienced users can write and modify automations directly in YAML format, which provides maximum flexibility and control. In graphical user interface (Fig. 3), for users non-familiar with coding, Home Assistant provides a graphical form-based interface to create automations, offering a more user-friendly approach. During the test, users created the automations using the graphical user interface. Before starting the tests, they received a brief tutorial about the main elements of the interface and had approximately 5 min to familiarise themselves with it and pose any questions to the researchers regarding its features.

4.1 Test organisation

Before starting the test, users were required to fill out a questionnaire that collected personal information, including age, gender, programming experience, familiarity with virtual assistants, and experience in creating automations. Following this, a brief instructional session on trigger-action programming was conducted, where users learned basic concepts, such as distinguishing between an event and a condition and using logical operators to link multiple elements in automation. In this phase, users were given task descriptions and asked to identify the event, any relevant conditions, and the actions required. This phase was crucial for assessing whether users had understood the necessary concepts and to reduce doubts during the test.

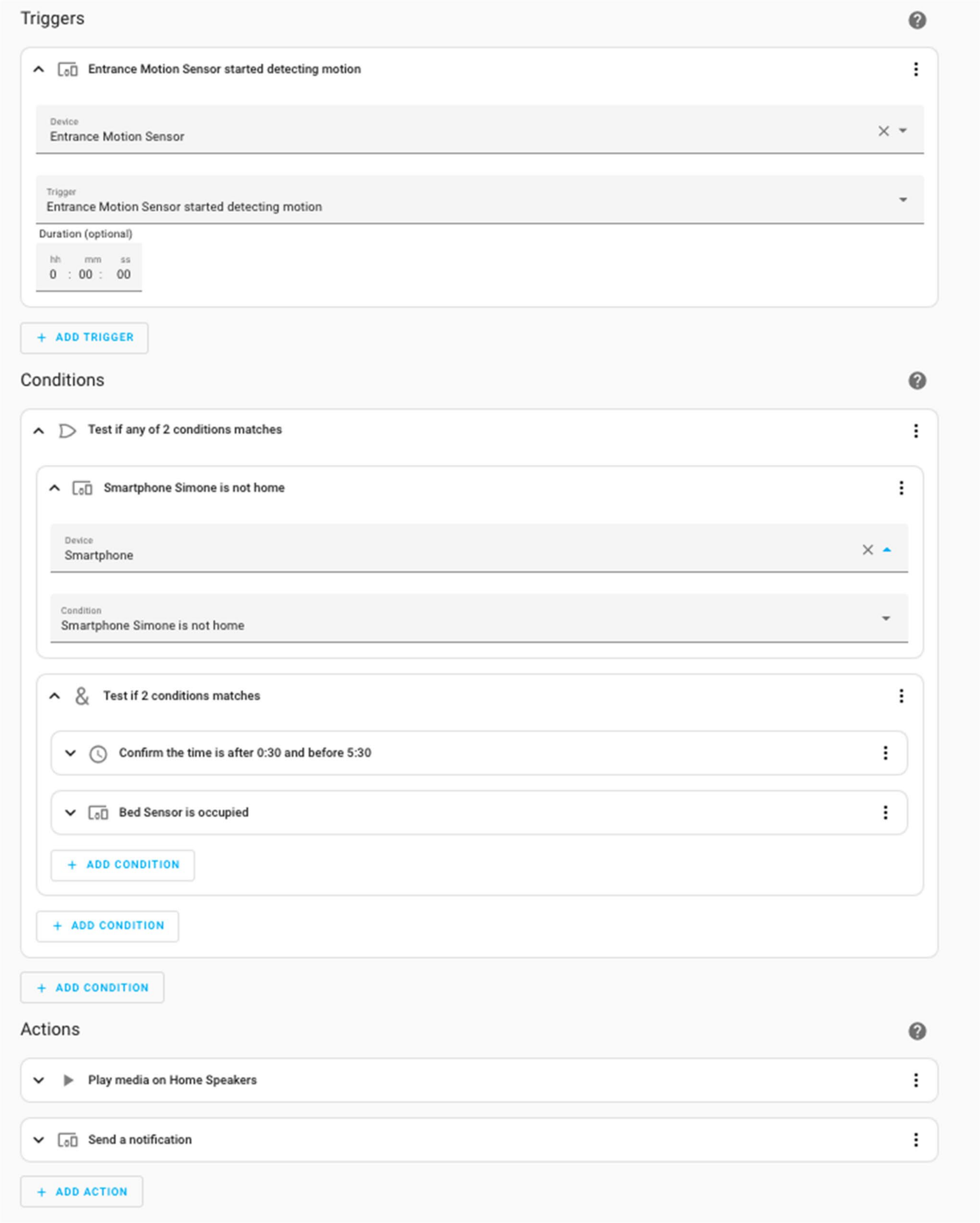
This study involved sixteen participants, ten males and six females, aged between 26 and 60 (mean = 31, median = 29, St. dev = 8.53). Participants were recruited through mailing lists and direct contacts using snowball sampling [13]. We selected participants who provided positive feedback and expressed interest in using smart homes to perform the test. Since real-world smart home users may have varying levels of programming expertise [14], we did not prioritise selecting users with specific types of experience. The proposed solution is designed to be accessible to a wide range of users with varying levels of programming skills. Most participants (13 out of 16) reported sporadic use of digital assistants such as Alexa or Google Home, often in the homes of friends or relatives. Regarding the creation of automations, experience levels varied: seven participants had no experience, eight participants had occasionally used applications such as IFTTT, Alexa, and Google Home, and one user used automations tool regularly. In terms of programming language proficiency, five participants had no programming experience, six had moderate experience with at least one language (e.g. HTML, CSS or JavaScript), two had good knowledge of the languages they used, and three participants had almost professional programming experience. Educational backgrounds were varied, with seven holding master’s degrees, five with bachelor’s degrees, three with diplomas, and one PhD. We tested the tool with a varied group of users as RuleBot+ +was not designed for a specific type of user, thus reflecting the potential diversity in smart home users.

The following four scenarios were presented to the participants, asking them to create the described automations using RuleBot+ +and Home Assistant; users were instructed to use—preferably but not necessarily—the term “when” to identify an event and “if” for a condition when writing to RuleBot+ +.

---

[5] https://analytics.home-assistant.io

[6] https://github.com/home-assistant



Triggers

Entrance Motion Sensor started detecting motion

Device: Entrance Motion Sensor

Trigger: Entrance Motion Sensor started detecting motion

Duration (optional): hh mm ss 0 : 00 : 00

Conditions

Test if any of 2 conditions matches

Smartphone Simone is not home

Device: Smartphone

Condition: Smartphone Simone is not home

Test if 2 conditions matches

Confirm the time is after 0:30 and before 5:30

Bed Sensor is occupied

Actions

Play media on Home Speakers

Send a notification

Fig. 3 The Home Assistant automation editor shows an example of complex automation. Some element configuration windows are collapsed for brevity

> **Scenario 1** (1 event + 1 action): You often work remotely, and your desk is located in your bedroom. Your task is to create an automation that detects the air quality and, when it is not good (usually over 1300 ppm), automatically activates the air purifier.
>
> **Scenario 2** (1 event + 1 condition + 1 action): Imagine waking up every working day between 7:00 and 7:30 in the morning. The first thing you always do is have a coffee to start the day. Your task is to create an automation that turns on the coffee machine when you get out of bed during this time interval and checks that it is a working day.
>
> **Scenario 3** (1 event + 2 conditions + 3 actions): You have recently moved into a new apartment and have not yet had the opportunity to install an alarm system. Your task is to create an automation that, when motion is detected at the entrance, while you are not at home or if it is night (e.g. between 00:00 and 06:00), activates a series of actions to ensure your safety. Red lights turn on at the entrance and in the corridor to create a visual deterrent effect, an alarm sound is activated to alert anyone nearby, and you immediately receive a notification on your phone. The user's location is detected through the phone's location.
>
> **Scenario 4** (add 1 condition + 1 action): Modify the automation created during Scenario 1. You must check that there is someone in the bedroom before turning on the purifier. Additionally, the bedside lamp with blue light should also turn on.

We asked the users to perform the four tasks, presented as scenarios, of increasing complexity with both conditions (RuleBot+ +and Home Assistant). The study followed a within-subject design protocol, so all the participants tested both tools. To balance the learning effect, the user's starting order was randomised, meaning that half of the users started the test using RuleBot+ +first, then Home Assistant, and vice versa.

The tests were conducted remotely, with the user sharing their screen. During the execution of the tests, the times taken to complete each task were collected and notes were made on the user's behaviour while using the two tools. Upon completing the test with each tool, users were asked to fill out a brief custom questionnaire that focused on their experience in creating and modifying automations, followed by a system usability scale (SUS) questionnaire [15].

4.2 Task error analysis

We have analysed the errors present in the rules created with the two tools, comparing each automation created by the users with the expected correct automation configuration. In the case of Home Assistant, we analysed the automation specified in the user interface, while for RuleBot+ +, we analysed the YAML code generated when saving the automation.

The number of errors is limited for both tools. Home Assistant automations present a total of 17 errors (task 1, 1; task 2, 1; task 3, 12; task 4, 3), while RuleBot+ +present a total of 19 errors (task 1, 0; task 2, 8; task 3, 2; task 4, 9). In the case of Home Assistant, the errors are focused on the task described in scenario 3. Specifically, the most common error concerns the configuration of the "OR" connector to concatenate the two conditions (12 users out of 16) in which some of them placed an element in the wrong position out of the OR scope. Despite its concentration on a single element, this error significantly affects the correct execution of the rule, because all rules created in this way would lead to unexpected behaviour of the automation or its non-activation. The second type of error involves the inversion between the event and the condition (5 users out of 16) in task 1, task 2, and task 4. For example, in task 4, the condition on the presence inside the room was used as an event, while the event on the air quality level was placed as a condition (three users out of five). This error could also lead to unexpected behaviour or non-activation of the automation (e.g. if the user is present in the room before the air quality decreases, the automation will not be activated).

Regarding RuleBot+ +, we observe a broader set of error types. We can distinguish between errors made by the user, errors from lexical ambiguities leading to misunderstandings between the user and RuleBot+ +(therefore considered "shared"), and errors due to the hallucinations of the model used. The errors that relate exclusively to users are found in task 2 (one user) and task 3 (two users) and concern either an inversion between the event and the condition or a more general incorrect description of the automation. Some lexical ambiguities in the user message led to an inversion between an event and a condition (3 users out of 16, exclusively in task 2). For example, P7 described the automation for task 2 as "If it is a working day, check that I have got out of bed between 7 a.m. and 7.30 a.m, in that case switch on the coffee machine". Based on this description RuleBot+ +created an automation described as "When it is a working day (Monday to Friday), and the time is between 7 a.m. and 7.30 a.m., if the bed presence sensor does not detect your presence, activates the coffee machine" that was incorrect since indicates the days as event, while the time and the bed presence as conditions. Anyway, the user confirmed and saved the incorrect automation. The final category of errors pertains exclusively to RuleBot+ +and involves the model's hallucinations. Such hallucinations occurred when the chatbot generated the YAML specifications. The use of a wrong value in the configuration of triggers or actions (e.g. "on" instead of "off" about the state of a sensor) was present in three automations created during task 2, and, again in task 2, the addition of an unrequested trigger occurs in one

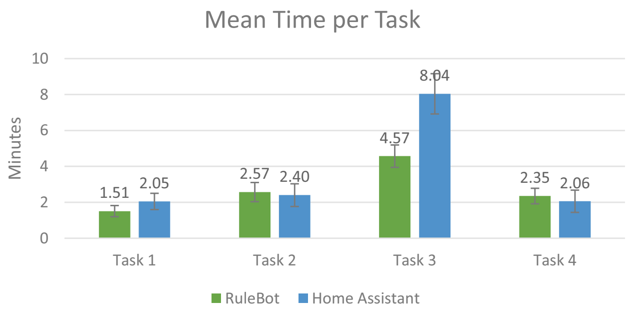


Fig. 4 Task completion time with the two tools, in minutes

automation. Another hallucination type involves configuring one sensor instead of another: the "bed presence" sensor was selected instead of the "room presence" sensor for detecting the user presence in task 4 (9 users out of 16). It is important to note that in cases of lexical ambiguity, RuleBot+ +correctly described the automation created (as in the above P7 example), whereas, in the case of the hallucinations detected, they occurred in the RuleBot+ +'s generation of the YAML code, which as not visible to the user. For instance, as described above, in task 4, users requested the detection of presence in the bedroom to trigger the activation of the air purifier. In this scenario, the model configured the YAML automation to utilise the bed presence sensor instead of the motion sensor (despite it being instructed to use the motion sensor for detecting the user's presence), without notifying the user.

4.3 Task time

The task completion times were gathered, beginning from the user's initial interaction (either a mouse click or the first keystroke when using RuleBot++) and ending upon the successful saving of the automation within the system.

Figure 4 shows the average time users needed to complete each task on both platforms. The data reveal that the performance of RuleBot+ +and Home Assistant is similar for tasks 1, 2, and 4. However, there is a noticeable difference in task 3, which required the creation of more complex automations. Here, RuleBot+ +is more efficient, as it consistently takes less time than Home Assistant.

A statistical analysis using Wilcoxon's signed-rank test confirms that the difference in completion times for task 3 is significant. The test results in a p-value < 0.001 with an effect size of 0.972 (using the rank-biserial correlation), indicating RuleBot+ +'s effectiveness in managing complex automation tasks with statistical significance, supporting the conclusion that RuleBot+ +provides a more efficient interface for complex task completion compared to Home Assistant. The other tasks' times do not present statistically significant differences.

4.4 Thematic analysis of user feedback

Participant feedback was solicited through both a custom questionnaire and the system usability scale to provide comprehensive insights into the strengths and weaknesses of each system. In addition, participants were requested to provide verbal feedback (which was transcribed by the researcher conducting the tests) regarding their impressions of the two tools. Utilising the qualitative data gathered, a thematic analysis was undertaken to identify and categorise recurring themes within the various feedback. This analysis adhered to the well-established guidelines proposed by Braun and Clarke [16, 17], employing an inductive approach where participant feedback was subjected to open coding. Our initial step involved an extensive review of all feedback, during which preliminary annotations were made to gain an understanding of the content within the feedback. Following this initial screening, these annotations were reviewed and normalised, leading to the identification and definition of the codes ($N = 10$). The subsequent phase involved the research and definition of themes. Here, existing codes were grouped into preliminary themes, with each being assigned a representative name and an encompassing description that reflected the definitions of its constituent codes. Finally, these themes ($N = 4$) underwent further revision, involving the review of feedback corresponding to the codes that formed each specific theme.

Table 1 displays the identified themes, along with the codes that constitute them, and provides example references drawn from these codes.

4.4.1 Ease of use

This theme explores user perceptions regarding the ease of use and intuitiveness of the interfaces and features of different tools, as well as the consistency in presenting and utilising these features. The feedback analysis highlights inconsistencies and challenges encountered in navigating and locating options and devices within Home Assistant. Participant 1 points out the lack of intuitiveness, saying, "It is not intuitive to find the functionalities of various objects, for example, some are under devices, others under services, others under media". Participant 7 also notes difficulties in navigating the interface, especially when defining a rule:

Table 1 Themes and codes with examples

Themes	Examples	Codes
Ease of use	- "I appreciated the simplicity and immediacy [of Rulebot++]" - "The mode of selection for the 'OR' operator is counter-intuitive" - "HA is somewhat cumbersome in searching for devices, displaying inconsistency in the device selection process"	- Intuitiveness of the interface - Interface comprehensibility and clarity - Consistency of the functionality presented
Automation management	- "For quick editing, I would use Rulebot because it's faster. If I wanted to analyse things in detail, I would switch to HA" - "It is much simpler and faster to create automation rules using natural language […]" - "Sometimes [with HA] is difficult to understand the difference between a trigger and a condition"	- System use cases - Creating and modifying automations - Distinction between event and condition
Information quality and quantity	- "[HA] lack of a final summary of what has been created at the time of saving" - "I did not like the multiple requests for confirmation [RuleBot++]" - "I feel more secure using HA because it is more formalized" - "RuleBot is simpler and provides feedback; I am not sure if I have correctly set the 'OR' operator in Home Assistant"	- Information quality and quantity - System feedback and reliability
Learning curve	- "With HA, you need someone to initially explain how to do things" - "[RuleBot++] does not require any technical knowledge or expertise" - "When there are uncertainties, RuleBot presents clear questions to help choose the preferred option"	- Need for prior knowledge - System suggestions and help

"It was difficult to navigate through the various options". In contrast, the use of natural language significantly enhances the perceived ease of use. Users have expressed positive feedback for RuleBot+ + in this regard. Participant 4 appreciates the guidance provided by the chatbot, stating, "No need to search […], with the chatbot you are guided step by step". They also express a preference for the simplicity and immediacy offered by using natural language, highlighting the difference in user experience between the two tools.

4.4.2 Automation management

This theme focuses on the feedback regarding the management of automations and their associated concepts and, in some cases, highlights a distinction in the specific applications of the two tools. Also for this theme, users express a preference for using natural language over a form-based interface. Participant 8 notes the speed and ease of this method, stating, "Very fast, I don't have to search for sensors or available devices but can ask the bot directly". Similarly, Participant 4 finds it "much simpler to create automation rules with natural language compared to Home Assistant", adding that this method is also quicker.

Challenges arise in the Home Assistant interface, particularly in differentiating between events and conditions. Participant 6 observes, "There are some details where it becomes difficult to understand the difference between an event and a condition". Participant 10 echoes this sentiment, noting the lack of immediacy in distinguishing between events and conditions.

The feedback also delves into the specific use cases for the two tools. Participant 1 states, "For quick editing, I would first use Rulebot+ + because it's faster. If I wanted to analyse things in detail, I would switch to HA". In contrast, user 4 suggests, "With Home Assistant, it's not a problem to create simple rules, but I think Rulebot+ + is simpler for creating complex rules".

4.4.3 Information quality and quantity

This theme encompasses the two most extensive codes in terms of the number of comments, which concerns the users' confidence in using the system. This includes considering the utility of their feedback (both quantitatively and qualitatively) and, hence, their perception of reliability. In the context of Home Assistant, aside from the challenges related to the overwhelming amount of information presented during automation creation, a preference for complete control over options is noted by two users. Participant 16 expresses a preference for a traditional form-based interface, valuing access to all functionalities simultaneously: “I’m used to using classic visual-based software and prefer to have access to all functionalities at once”. Participant 14 echoes this sentiment, appreciating the visibility of all options in Home Assistant: “With Home Assistant, I have all the options in front of me”. Moreover, the need for improved system feedback is a recurring theme. Participant 7 observes a lack of guidance in creating and operating rules in Home Assistant: “[…] HA doesn’t even provide feedback/help in compiling the automation and its operation”. Participant 3 expresses uncertainty regarding the correct setting of conditions: “[…] I’m not sure if I have set OR correctly in Home Assistant”. Participant 13 points out the absence of a conclusive summary upon saving: “[HA] lacks a final summary of what has been created at the time of saving”.

Similarly, despite the feedback and summaries provided by RuleBot + +, its less formalised approach raises concerns about clarity and confidence among users. Participant 16 states: “[…] I don’t feel secure about what the chatbot is actually doing”. Also, participant 1 feels that the structured approach of Home Assistant boosts is confidence: “I feel more secure using Home Assistant because it’s more rigid”.

User feedback also reveals a divide in opinions regarding RuleBot’s confirmation requests during automation definition and saving. While some users find these confirmations excessive (participant 7: “I liked less the fact that [RuleBot] asked me for too many confirmations before saving a rule”, participant 11: “I didn’t like the fact of the continuous requests for confirmation”), others appreciate this feature for ensuring accuracy and control (participant 15: “I liked the summary provided by [RuleBot] and the request for confirmation of automations”, participant 13: “the fact that [RuleBot] asked for confirmation before proceeding made me realize I was making a mistake”).

4.4.4 Learning curve

This theme focuses on user feedback related to their initial engagement with the tool, the perceived necessary knowledge for effective usage, and the learning curve associated with the tools. Specifically, comments regarding RuleBot + + highlight the tool’s helpful suggestions and guidance. Participant 1 appreciates RuleBot’s proactive approach, mentioning, “I also liked that RuleBot in some cases suggested further specifics, for instance, it asked me if I wanted to add an additional condition, which I hadn’t thought of”. Participant 8 praises RuleBot + + ’s flexible understanding and decision-making assistance: “RuleBot understood every request […] even making decisions for me when I had no specific preferences”. Participant 14 values the clarity provided by RuleBot + + in uncertain situations: “I liked the fact that when there are uncertainties, RuleBot presents clear questions to help choose the preferred option”.

When discussing the learning curve of the tools, RuleBot + + is perceived as more immediate and user-friendly. Participant 10 points out the prerequisite knowledge for Home Assistant (HA): “HA requires prior knowledge of automations and terminology”. Participant 11 notes the initial difficulty with HA: “Tool [HA] is not very easy to understand at first approach”. However, users indicate that familiarity with HA improves over time. Participant 12 observes, “With HA, you need someone initially to explain how to do things. Once you understand where to click, it’s not difficult”. This sentiment is echoed by participant 6: “Once you get into the mechanism, the tool is comprehensible”. Meanwhile, participant 5 contrasts the two tools: “The chat [RuleBot++] is more immediate and doesn’t require specific knowledge. For the second [HA], more knowledge is required, but it’s more straightforward”. This feedback illustrates a balance between immediate usability and depth of understanding of the two tools.

4.5 System usability scale and custom questionnaire results

Finally, we report the results obtained from the two questionnaires to which the users were subjected. Each participant completed a SUS and a custom questionnaire immediately after completing tasks with each of the two tools (thus, each user filled out two SUS and two custom questionnaires). While the SUS provides an overview of the system’s usability, the custom questionnaire is targeted to aspects related to the creation and modification of automations.

Specifically, the custom questionnaire includes the following statements, to which users assigned a score on a scale from 1 (totally disagree) to 5 (totally agree):

S1: “It is easy to create a simple automation with [RuleBot/Home Assistant]”.
S2: “It is easy to create a complex automation with [RuleBot/Home Assistant]”.

Table 2 Results of the custom questionnaire: p-values are calculated using Wilcoxon's signed-rank test, and the effect sizes are calculated using the rank-biserial correlation

Statement	System	Median	Mean	St. dev	p-value	Effect size
S1	RuleBot + +	5	4.75	0.44	0.004	0.879
	Home Assistant	4	3.81	0.83		
S2	RuleBot + +	5	4.50	0.50	0.003	0.945
	Home Assistant	3	3.00	0.81		
S3	RuleBot + +	5	4.69	0.47	0.006	0.939
	Home Assistant	3	3.31	1.30		
S4	RuleBot + +	5	4.75	0.44	0.02	1.0
	Home Assistant	4	4.37	0.61		
S5	RuleBot + +	4	4.31	0.60	Not significant	Not significant
	Home Assistant	5	4.37	0.80		

S3: "It is easy to concatenate two conditions using OR with [RuleBot/Home Assistant]".
S4: "It is easy to modify an automation with [RuleBot/Home Assistant]".
S5: "It is clear whether I am defining an event or a condition with [RuleBot/Home Assistant]".

We define a "complex" automation as one that integrates at least a trigger event, a condition, and an action; a "simple" automation does not contain conditions.

Table 2 summarises the scores related to the different statements. As can be seen, RuleBot + + received excellent evaluations on all analysed aspects, showing consistency among the scores received from different users. The data relating to Home Assistant present particularly low scores in the phase of defining automations (S1, S2, S3), especially for creating complex automations (S2) and inserting connectors (in this case, "OR") between two or more conditions (S3). The reported p-values and the relative effect sizes demonstrate a significant difference between the user evaluation of the two tools for S1, S2, S3, and S4.

The SUS (system usability scale) results also support the observations made so far, with RuleBot + + scoring, on average, 89.8 (min = 85, max = 95, St. dev = 3.35) against Home Assistant's average final score of 62.9 (min = 17.5, max = 95, St. dev = 19.64). The Shapiro–Wilk test confirmed that the data are normally distributed, as indicated by p-values > 0.05 for both RuleBot + + and Home Assistant. The statistical analysis using the Student t-test revealed a significant difference between the scores, with a p-value < 0.001 and a Cohen's d effect size of 1.321. The outlier analysis using the Z-score method at a 95% confidence level (critical Z value of ± 1.96) identified one outlier in the Home Assistant SUS scores, specifically the minimum value of 17.5. Upon removing this outlier, the recalculated average SUS score for Home Assistant increased to 66 (min = 37.5, St. dev = 16). The Student t-test, reassessed after outlier removal, maintained a p-value < 0.001 (effect size = 1.382). Using the rating scale proposed by Bangor et al. [18], we can classify RuleBot + + between "good" (> 70) and "excellent" (> 90) and Home Assistant between "ok" (> 50) and "good".

5 Discussion

RuleBot + + 's broader range of errors, including those stemming from lexical ambiguities and the model's hallucinations, contrasts with the more focused and less frequent errors observed in Home Assistant. Notably, the Home Assistant's errors primarily pertained to the misuse of logical connectors and the inversion of events and conditions in specific scenarios, which, while fewer, significantly impacted the perceived ease of use.

Focussing on the LLM hallucinations, different studies aim to somehow give an explanation and a categorisation for why and when they occur. For example, Ji et al. [19] distinguish between "intrinsic" (the generated output contradicts the source content) and "extrinsic" (the generated output cannot be verified from the source content) hallucinations, while Zhang et al. [20] propose a classification into "input-conflicting" (LLMs generate content that deviates from the source input provided by users), "context-conflicting" (LLMs generate content that conflicts with previously generated information by itself), and "fact-conflicting" (LLMs generate content that is not faithful to established world knowledge) hallucination.

In the case of RuleBot + +, hallucinations primarily concern the consistency between the data generated during the conversation, especially between the automation in YAML format and its textual description. Indeed, in the highlighted cases, the chatbot automation description aligns with the user's request, but the generated YAML code actually includes elements that are inaccurately reported—either in whole or in part. Following the definitions mentioned earlier, we could say that the hallucinations concerning RuleBot+ + are "intrinsic" and "context-conflicting" since the errors are based on the generated source content.

Another important point of discussion, highlighted both by the error analysis and user comments, concerns the distinction between events and conditions, related to both RuleBot+ + and Home Assistant. From this perspective, the main difference between the two tools lies in the explicitness with which events and conditions are defined. Despite users having verbally formalised the rules before implementing them and despite the interface of Home Assistant being quite clear on where to insert the event and where the conditions, users still emphasised the difficulty in differentiating the two components. On the other hand, when using natural language, it becomes more complex to uniquely identify an event and a condition, given the nature of the language itself. Users were instructed to use—preferably but not necessarily—the term "when" to identify an event and "if" for a condition. Despite this, many users, especially for complex rules with multiple conditions, followed a more intuitive interaction not strictly adhering to the terms "when" and "if" [21]. It is, therefore, clear that for non-expert users, the difference between the two components is not immediately obvious [22, 23], highlighting a need to explicitly represent the concepts. In the specific case of the conversation, the agent should intelligently understand which elements of the automation should take on the role of events and which others of conditions, guiding the user to the correct implementation.

Finally, we want to address a problem related to the perception of RuleBot+ +'s transparency. Indeed, some users have pointed out feeling unsure about what the chatbot was doing "behind the scenes" during the creation of automations, in contrast to Home Assistant's explicit interface. In fact, RuleBot+ + does not expose the system status except through the messages it sends to the user (e.g. "I have verified the automation", "I have saved the automation"). According to the heuristics developed by Nielsen [24], showing the system status is one of the key usability aspects. Therefore, it is likely that showing the user what is happening during the conversation consistently and independently from the messages sent by the chatbot (e.g. notifying when the chatbot is executing a function) could increase the perceived transparency and reliability. In addition, since we noticed that some hallucinations could occur when RuleBot+ + describes in natural language the obtained YAML specification, it can be useful to include also alternative representations obtained deterministically.

6 Conclusions and future work

This study introduces RuleBot+ +, a conversational agent leveraging large language models to facilitate the creation and modification of smart home automations, offering non-expert users an intuitive alternative to traditional form-based or simple voice command interfaces.

The user test demonstrates how the conversational approach can significantly simplify interactions with smart home systems by facilitating the configuration of automations as highlighted by the comparison between RuleBot+ + and Home Assistant. Results show better usability and intuitiveness, especially for tasks requiring the integration of multiple triggers and actions.

However, this study also revealed some inherent challenges in adopting language models for developing conversational interfaces. Issues such as linguistic ambiguity and "hallucinations" of the model can lead to errors or misunderstandings in interpreting user requests, highlighting the need for integrating more robust verification and confirmation mechanisms and improving the transparency of the system to mitigate such issues.

This study has been useful in exploring the role and possibilities of ChatGPT in supporting a conversational agent for automation creation. The number of user test participants is limited, which may impact the generalisability of the findings. Moreover, the architecture and the designed prompt could be improved for optimisation purposes.

In future studies, we plan to recruit more specific and larger participant groups to gain a deeper understanding of the user experience for specific user types (e.g. only users without any programming experience) with the proposed approach.

Additionally, in-the-wild evaluations (e.g. a 1-month trial in users' homes) could offer further insights into RuleBot+ +'s impact across a variety of home scenarios of everyday life. They can be carried out while exploring the integration with a wider range of IoT devices.

In conclusion, RuleBot+ + highlights both the potential of conversational interfaces in the domain of home automation and the challenges and opportunities for future research in this dynamic and rapidly evolving field by proposing an intuitive and powerful interface for configuring automations in intelligent environments.

Appendix

RuleBot + + system prompt

The original prompt is in Italian, here is the translated

English version.

Task: You are now RuleBot, a virtual assistant that helps users manage sensors and smart objects integrated in a smart home in natural language. RuleBot can create automations, modify automations and give information about the sensors in the smart home.

Task Info:

- An automation can also be called a 'rule', 'routine', 'trigger-action rule'
- Usually a trigger EVENT is identified by the term "when"
- Usually a CONDITION is identified by the term "if" or "while"

Task Rules:

- Always play the RuleBot role and never step out of the RuleBot character
- Always use a simple language that can be understood by non-expert users
- Never use YAML or other programming languages or data structures when talking to the user
- Do not talk about 'entity_id' with the user
- If it is not clear whether a trigger is an event or a condition, ask the user if he/she wants to define that trigger as an 'event' or 'condition'
- If not explicit, always ask the user for values to be assigned to the elements of triggers and actions (e.g. temperature, time, brightness, text)

RuleBot functionalities:

"retrieve_entities":

- When you receive the automation description, immediately use 'retrieve_entities' and check whether the required devices are available in the house
- Do not invent sensors/smart objects/information not in the list
- When creating, verifying and modifying an automation use only the present 'entity_id'

"verify_automation":

- Always use this functionality to check if a rule is correct before saving or editing
- If the verification is successful, summarise the rule and ask the user if he wants to save it

"save_automation":

- Before being saved, a rule must be successfully verified
- ALWAYS ask the user for confirmation before saving an automation

"modify_automation":

- Before editing, an automation must be successfully verified
- Always summarise the rule to the user and ask for confirmation before saving the modified automation

Funding Open access funding provided by ISTI - PISA within the CRUI-CARE Agreement.

Data availability The data that support the findings of this study are not openly available due to reasons of privacy and are available from the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Brush AB, Lee B, Mahajan R, Agarwal S, Saroiu S, Dixon C (2011) Home automation in the wild: challenges and opportunities. In: Proceedings of the SIGCHI conference on human factors in computing systems. ACM, Vancouver BC Canada, pp 2115–2124. https://doi.org/10.1145/1978942.1979249
2. He W, Martinez J, Padhi R, Zhang L, Ur B (2019) When smart devices are stupid: negative experiences using home smart devices. In: 2019 IEEE Security and Privacy Workshops (SPW). pp 150–155. IEEE, San Francisco, CA, USA. https://doi.org/10.1109/SPW.2019.00036
3. Yang R, Newman MW (2013) Learning from a learning thermostat: lessons for intelligent systems for the home. In: Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing. pp 93–102. ACM, Zurich Switzerland. https://doi.org/10.1145/2493432.2493489
4. Gallo S, Paterno F, Malizia A (2023) Conversational interfaces in IoT ecosystems: where we are, what is still missing. In: Proceedings of the 22nd International Conference on Mobile and Ubiquitous Multimedia. pp 279–293. ACM, Vienna Austria. https://doi.org/10.1145/3626705.3627775
5. Monge Roffarello A, De Russis L (2023) Defining trigger-action rules via voice: a novel approach for end-user development in the IoT. In: Spano LD, Schmidt A, Santoro C, and Stumpf S (eds.) End-User Development. pp 65–83. Springer Nature Switzerland, Cham. https://doi.org/10.1007/978-3-031-34433-6_5
6. Barricelli BR, Fogli D, Iemmolo L, Locoro A (2022) A multimodal approach to creating routines for smart speakers. In: Proceedings of the 2022 International Conference on Advanced Visual Interfaces. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3531073.3531168
7. Lago AS, Dias JP, Ferreira HS (2021) Managing non-trivial internet-of-things systems with conversational assistants: a prototype and a feasibility experiment. J Comput Sci 51:101324. https://doi.org/10.1016/j.jocs.2021.101324
8. Gallo S, Paterno F (2022) A conversational agent for creating flexible daily automation. In: Proceedings of the 2022 International Conference on Advanced Visual Interfaces. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3531073.3531090
9. Gallo S, Malizia A, Paternò F (2023) Towards a chatbot for creating trigger-action rules based on ChatGPT and Rasa. In: Workshops, work in progress demos and doctoral consortium at IS-EUD 2023. CEUR-WS.org, Cagliari
10. Jurafsky D, Martin JHJ (2024) Chatbots & dialogue systems. In: Speech and language processing. An introduction to natural language processing, computational linguistics, and speech recognition. Standford Edu, pp 315–336
11. Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler DM, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D (2020) Language models are few-shot learners. http://arxiv.org/abs/2005.14165. Accessed 10 Jan 2024
12. Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Chi E, Le Q, Zhou D (2023) Chain-of-thought prompting elicits reasoning in large language models, http://arxiv.org/abs/2201.11903
13. Snowball sampling (2020) In: SAGE Research Methods Foundations. SAGE Publications Ltd, 1 Oliver's Yard, 55 City Road, London EC1Y 1SP United Kingdom. https://doi.org/10.4135/9781526421036831710
14. Barricelli BR, Fogli D, Gallo S, Guizzardi D, Maenza S, Mattioli A, Paternò F (2024) What people think about green smart homes. In: CoPDA@ AVI
15. Brooke J (1996) SUS—A 'quick and dirty' usability scale. Usability Evaluation in Industry 189:4–7
16. Braun V, Clarke V (2006) Using thematic analysis in psychology. Qual Res Psychol 3:77–101. https://doi.org/10.1191/1478088706qp063oa
17. Braun V, Clarke V (2013) Successful qualitative research: a practical guide for beginners. SAGE, Los Angeles
18. Bangor A, Kortum P, Miller J (2009) Determining what individual SUS scores mean: adding an adjective rating scale. J Usability Stud 4:114–123
19. Ji Z, Lee N, Frieske R, Yu T, Su D, Xu Y, Ishii E, Bang YJ, Madotto A, Fung P (2023) Survey of hallucination in natural language generation. ACM Comput Surv 55:1–38. https://doi.org/10.1145/3571730
20. Zhang Y, Li Y, Cui L, Cai D, Liu L, Fu T, Huang X, Zhao E, Zhang Y, Chen Y, Wang L, Luu AT, Bi W, Shi F, Shi S (2023) Siren's song in the AI ocean: a survey on hallucination in large language models. http://arxiv.org/abs/2309.01219. Accessed 13 Jan 2024
21. Andrao M, Treccani B, Zancanaro M (2023) Language and temporal aspects: a qualitative study on trigger interpretation in trigger-action rules. In: Spano LD, Schmidt A, Santoro C, and Stumpf S (eds.) End-user development. pp. 84–103. Springer Nature Switzerland, Cham. https://doi.org/10.1007/978-3-031-34433-6_6
22. Brackenbury W, Deora A, Ritchey J, Vallee J, He W, Wang G, Littman ML, Ur B (2019) How users interpret bugs in trigger-action programming. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. pp. 1–12. ACM, Glasgow Scotland Uk. https://doi.org/10.1145/3290605.3300782
23. Huang J, Cakmak M (2015) Supporting mental model accuracy in trigger-action programming. In: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing. pp. 215–225. ACM, Osaka Japan. https://doi.org/10.1145/2750858.2805830
24. Nielsen J (1994) Enhancing the explanatory power of usability heuristics. In: Proceedings of the SIGCHI conference on human factors in computing systems. ACM, Boston USA, pp 152–158. https://dl.acm.org/doi/10.1145/191666.191729

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.