Constructing Scheduled Home Automation with LLM Agent for Supporting Elderly Living Alone

Ryota Murate[1], Akinori Matsukawa[1], Takuya Nakata[1], Sinan Chen[1], Kiyoshi Yasuda[1], and Masahide Nakamura[1,2]

[1] Kobe University, 1-1 Rokkodai-cho, Nada-ku, Kobe, 657-8501, Japan
rmurate@es4.eedept.kobe-u.ac.jp
matuaki@es4.eedept.kobe-u.ac.jp
tnakata@bear.kobe-u.ac.jp
chensinan@gold.kobe-u.ac.jp
fwkk5911@gmail.com
masa-n@cmds.kobe-u.ac.jp
[2] RIKEN AIP, 1-4-1 Nihonbashi, Chuo-ku, Tokyo 103-0027, Japan

Abstract. With the rise of elderly individuals living alone, IoT and AI-based life support technologies have gained significant attention. However, complex interfaces often exacerbate the digital divide, while excessive automation can undermine user autonomy. This paper proposes a Large Language Model (LLM)-based home automation system that generates control schedules adapted to user needs through natural language dialogue. By extracting "6W1H" requirements from dialogue logs and utilizing iterative reasoning to create API execution plans, the system fosters independence. Evaluations demonstrate that our approach achieves higher task coverage and respects user preferences for manual control more effectively than simple batch generation.

Keywords: Elderly living alone · Home automation · LLM · Life support · Dialogue systems.

1 Introduction

In recent years, the development of smart homes and monitoring services utilizing IoT (Internet of Things) and AI technologies to support the lives of the elderly has been actively pursued [1]. These technologies are expected to contribute to maintaining and improving the Quality of Life (QoL) of the elderly and reducing the burden on caregivers.

However, existing smart home devices often exacerbate the digital divide due to complex setups. Conversely, excessive automation risks depriving the elderly of their autonomy, leading to learned helplessness. Therefore, a system must adaptively balance support and user independence.

To address this issue, Large Language Models (LLMs), which have been developing rapidly in recent years, are promising as interfaces that pick up ambiguous user requests through natural language dialogue and convert them into

commands for the system. If elderly people can interact with the system like daily conversation, conveying their life rhythm and what they want to do themselves or want done for them, and the complex IoT device settings are completed in the background, there is a possibility of solving the above problems. However, sufficient verification is needed on whether accurate and safe control schedules for IoT devices, which require strict operation, can be generated using LLMs that operate probabilistically.

Therefore, this study proposes a system that extracts life support requirements from dialogue and generates specific control schedules, aiming to realize a "Personalized Home Automation Service" for elderly people living alone. This study focuses particularly on accurate and comprehensive generation, the effectiveness of the reasoning process, and suitability for independence support.

This system extracts structured data on when (When), what (What), and how (How) to support from the dialogue logs of the user and the agent, and converts it into an executable API schedule for controlling existing smart home appliances (SwitchBot [6], etc.) and voice notification systems. As a feature of the proposed method, instead of the LLM generating all schedules at once, it takes an approach of iteratively refining the schedule while considering the context of the dialogue and user constraints (e.g., "I want to turn off the lights myself") via an intermediate representation (6W1H). Through this, we aim to automate support settings that respect the user's independence.

In this paper, we implement a prototype of the proposed system and conduct an evaluation experiment based on scenarios assuming elderly people living alone. In the experiment, we compare a simple conversion pipeline with a stepwise generation method considering context, verifying the coverage of tasks and the quality of generated messages. As a result, we show that by using the proposed method, it is possible to generate schedules that successfully maintain logical consistency from ambiguous user utterances, and to set appropriate support ranges that do not hinder the user's abilities.

2 Preliminaries

2.1 Dilemma in Life Support Systems

In modern society, where the aging population is rapidly increasing, life support systems utilizing AI and IoT technologies play an important role in maintaining the QoL (Quality of Life) of the elderly [2] and reducing the burden of care. Existing smart homes have complex settings, creating a high barrier to entry for the elderly. Another issue is the harmful effects of excessive automation. As a solution to reduce the burden of operation, full automation using sensors etc. can be cited. However, excessive automation that does not intervene with the user's intention leads to depriving the user of opportunities to think and act on their own. This risks inducing a state similar to learned helplessness [9] in psychology and inviting a decline in physical and cognitive functions.

Therefore, an ideal support system is not one that simply automates all tasks, but a system that can adjust the balance between support and independence according to the user's state and intention.

2.2 Related Work and Potential of LLMs

Conventionally, rule-based approaches using ontologies and logic programming have been adopted for context recognition and control of smart homes.

In recent years, with the development of Large Language Models (LLMs) [3], attempts have been made to generate programs and actions from instructions in natural language. If LLMs are used, there is a possibility that elderly people can configure the system through natural dialogue. However, since LLMs perform probabilistic output, sufficient verification is necessary regarding whether ambiguous user instructions can be accurately converted into strict execution schedules (date and time, parameters) and whether their logical consistency is maintained.

2.3 Focused Issues and Approach

In this research, as a final goal, we aim to build an "Adaptive Life Support System" that performs automatic support tailored to the individual's lifestyle and cognitive level through dialogue between the user and the agent [8]. It is an environment where the agent acts as a proxy for complex settings of IoT devices, but instead of automating everything, the user continues to be involved in their own life management through dialogue.

As a first step to realize this major goal, this paper focuses on the issue of "how to generate IoT control schedules that require strict execution without omission while not impairing the user's autonomy using LLM". If one tries to generate a schedule from a long dialogue log with a single prompt, there is a technical difficulty that missing information or ignoring detailed user constraints (requests regarding independence support) occurs. The ability to elicit user needs through dialogue, combine them with existing IoT toolsets, and break them down into executable schedules is the core function of system realization.

In this paper, as a preliminary study for the realization of an adaptive support system, we aim to verify the feasibility of a "dialogue-to-schedule generation/execution framework" using LLMs and to clarify the effectiveness of an iterative reasoning approach for improving the accuracy and comprehensiveness of generation.

Specifically, taking dialogue logs regarding support needs between a user and a dialogue agent and the definition of devices and functions operable by the system (tool catalog) as inputs, we investigate how the LLM schedules tasks.

As an approach, we construct a pipeline that extracts 6W1H (When, Where, Who, What, Why, Whom, How) as structured data from natural language dialogue [4] and converts it into an API execution schedule. The scope of verification in this paper is to confirm whether this conversion process functions logically

without breakdown and whether the generated schedule operates as intended. Through this, we establish the technical foundation for a future "adaptive system coexisting with the elderly".

3 Proposed Method

This section describes the overall architecture of the life support system to solve the issues mentioned in the previous section.

In this study, we propose an architecture that links three components: Conversation to Schedule (C2S), Scheduler, and Executor, to realize needs extraction through dialogue, schedule generation, and environmental device control.

3.1 Overall Architecture

The overall configuration of the proposed system is shown in Fig. 1. The system adopts a microservices configuration that loosely couples on the server side, except for the interface with the user and the physical control endpoints.

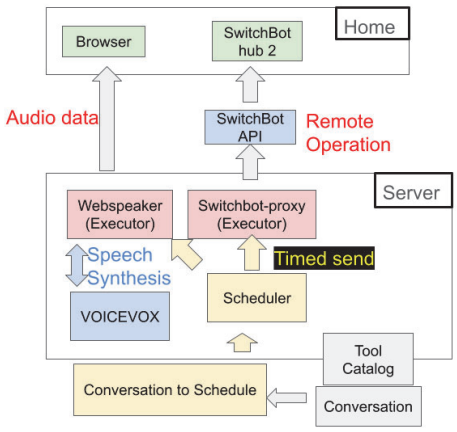

Fig. 1. Overall architecture of the proposed system.

The roles of each component are as follows.

- **Conversation to Schedule (C2S):** Contains the LLM, receives dialogue logs with the user and the "tool catalog" operable by the system as input, and outputs executable schedule data.
- **Scheduler:** Holds the schedule generated by C2S in a database and performs time management. When a predetermined time is reached, it triggers the corresponding job and sends an execution order to the Executor in the subsequent stage.
- **Executor:** Receives the execution order from the Scheduler and executes the actual device operation.

3.2 Conversation to Schedule (C2S): Context Understanding and Schedule Generation

The C2S component is responsible for converting unstructured conversation data into a structured, system-interpretable schedule. Receiving dialogue logs and tool definitions as input, the generation process first constructs a prompt by dynamically embedding the current time, user information, and tool catalog. Next, the LLM performs inference to output a needs extraction result in a 6W1H format alongside an execution schedule in JSON format. To realize programmable and flexible repetition settings, the execution timing within this generated schedule data adopts standard Cron expressions.

In converting these extracted needs into actual execution plans, we evaluate the effectiveness of information structuring and process division in the LLM's inference by comparing three pipelines. Under Pipeline 1 (Direct Conversion or One-Shot Mapping), the LLM directly outputs the schedule from the conversation logs and tool catalog in a single inference step without an intermediate representation. Under Pipeline 2 (Conversion via 6W1H or Two-Stage Generation), the system creates an intermediate file in 6W1H format before generating the schedule based on that organized information. Finally, under Pipeline 3 (Individual Consideration by Loop Processing or Iterative Elaboration), the system extracts all support requirements in 6W1H format and iteratively inputs each requirement into the LLM one by one to sequentially construct the final schedule.

3.3 Scheduler: Execution Management Logic

The Scheduler is the heart of the system, responsible for schedule persistence and monitoring of execution timing.

The Scheduler saves job data with fields for cron, endpoint, HTTP method, request body, and header in the database. To monitor and execute these schedules, it activates an internal task every minute, extracting jobs whose `cron` definition matches the current time. Each extracted job is then immediately passed to an asynchronous process, sending an execution request to the specified `endpoint`.

3.4 Executor: Abstraction and Control via Proxy

The Executor serves as a proxy interface that receives abstract requests from the Scheduler and acts on the physical world, specifically handling device authentication and voice notifications.

When operating physical appliances like lighting and air conditioners, the data generated by C2S and passed to the Scheduler contains only the device ID. Within the Executor's database, these device IDs are securely linked to their corresponding API tokens required for cloud access. When the Executor receives a request, it retrieves the necessary token using the device ID as a key, appends the standard authentication information, and relays the command to the external cloud API. This proxy configuration ensures that confidential information is never included in the LLM's prompt or generation output, reliably eliminating

the risk of token leakage via prompt injection or the transmission of incorrect credentials due to hallucinations. Furthermore, restricting the LLM to outputting only concise device IDs significantly reduces output token consumption.

In addition to device control, the Executor manages voice notifications that directly interact with the user. To ensure real-time performance, it cooperates with a speech synthesis engine and employs a mechanism to immediately push the generated audio data to the client via WebSocket.

Control of Voice Notification For voice notifications that call out to the user, in order to ensure real-time performance, the Executor cooperates with a speech synthesis engine and adopts a mechanism to immediately push the generated audio data to the client via WebSocket.

4 Implementation

This section briefly describes the implementation environment of the prototype system based on the architecture proposed in the previous section. Note that the main purpose of this study is to verify the logic of deriving a schedule from dialogue content, so in this prototype, we focused on the implementation of the backend processing of C2S, Scheduler, and Executor. The real-time voice dialogue function with the user is outside the scope of this paper's implementation, and the input to the system was configured to directly provide dialogue logs in text format. In the implementation, flexible Python was adopted for AI processing, and Java (Spring Boot) was adopted for scheduling and DB operations which require robustness, linking them via REST API and WebSocket. As the LLM, we adopted GPT-5.2 [5], the latest model from OpenAI as of December 11, 2025, which excels in inference settings and deep understanding of long contexts. The generated JSON data is saved as a log in storage. Implemented using Python. The Scheduler and SwitchBot-proxy were implemented using Spring Boot. For the speech synthesis of WebSpeaker, the VOICEVOX engine was used.

5 Evaluation Experiment

This section describes the experiment to verify the effectiveness of the proposed system and the three schedule generation pipelines. The experiment used a dialogue scenario with a fictional elderly user, and comparatively evaluated the accuracy, completeness, and generation time of the actually generated schedules.

5.1 Experimental Setup

To evaluate the proposed system and the three schedule generation pipelines, we established a physical experimental environment in the author's residence,

equipping it with a SwitchBot Hub 2 [7], a WebSpeaker, and actual appliances including an air conditioner, smart light, and TV.

The evaluation was based on a scenario assuming "Sato," an 80-year-old male living alone. While Mr. Sato maintains an independent lifestyle, he struggles with unfamiliar device operations, anxiety about forgetting to turn off appliances before leaving home, and medication management. To quantitatively assess the accuracy of schedule generation without the unpredictability of human trials at this early stage, we utilized predefined "mock dialogue logs" rather than interacting with actual elderly subjects. These logs simulate conversations where the agent interviews the user to determine support needs across four specific daily scenes: waking up, taking medication, going out, and going to bed. Table 1 outlines the ground-truth requirements that the C2S component was expected to extract from these dialogues.

Table 1. Support requirements to be extracted from dialogue.

Scene	Time Condition	Expected Action
Waking up	Mon/Thu 05:00, Others 05:30	TV, Light, Heater ON, Voice notification
Medication	Daily 07:30, 18:30	Voice notification only
Going out	Mon/Thu 08:30, Wed 09:00	TV, Light, Heater OFF, Voice notification
Bedtime	Daily 22:00	TV, Heater OFF, Voice notification

5.2 Experimental Results

Table 2 shows a comparison of the generation results by the three pipelines. Here, the definitions of each index are as follows. Task coverage rate is the percentage of items for which the system was able to generate a schedule without omission among all 16 support requirements defined in Table 1. Generation time is the total processing duration—including API round-trip time—from when the dialogue log and tool definition are input until the final JSON schedule is output. Description detail is a qualitative evaluation of whether the generated voice notification messages are not just a list of facts, but contain contextual consideration for the user.

Table 2. Comparison of evaluation results by pipeline.

Performance Index	Pipeline 1	Pipeline 2	Pipeline 3
Task Coverage	75.0% (12/16)	100% (16/16)	100% (16/16)
Generation Time	26.4 sec	46.2 sec	57.5 sec
Description Detail	Low	Low	High

Comparing the three approaches, Pipeline 1 (Direct Conversion) yielded the fastest generation time at 26.4 seconds. However, it lacked reliability for complex life support; while it generated schedules up until Wednesday, it completely missed the bedtime schedule. This omission is likely due to the LLM's scattered attention when processing long conversations and complex tool definitions simultaneously, or due to output token constraints. Pipeline 2 (Conversion via 6W1H) took 46.2 seconds but successfully covered all requirements, including the missing bedtime task. Notably, it correctly handled the user's request to "turn off the lights myself" by only scheduling the TV and air conditioner to turn off. By utilizing an intermediate representation (6W1H), the LLM was able to organize information and prevent task omissions. Finally, Pipeline 3 (Individual Consideration by Loop Processing) had the longest generation time at 57.5 seconds but delivered the highest quality results. While matching Pipeline 2's task coverage and constraint compliance, Pipeline 3 significantly improved the generated voice notification messages. Instead of simply listing facts like in Pipelines 1 and 2 (e.g., "Turned on the TV and lights"), Pipeline 3 generated context-aware messages such as:

> *"Good morning. I turned on the TV, lights, and air conditioner heating. Please be careful of your steps as you get up." "Confirmation before going out. ... Please turn off the lights yourself as needed."*

By processing tasks individually, Pipeline 3 minimized the impact of output token constraints, allowing the model to fully focus its inference capabilities on context understanding and detailed message generation for each specific task.

Following the generation phase, the JSON data produced by Pipeline 3 was registered in the Scheduler to verify its operation in the physical environment. We confirmed that requests to the SwitchBot API and WebSpeaker were successfully triggered at the set Cron times (e.g., 05:00, 22:00), executing the intended home appliance operations and voice notifications. Fig. 2 illustrates the system during active operation: the TV is powered on by the system, while the WebSpeaker plays the contextual morning greeting.

Fig. 2. Operation verification of the proposed system with actual equipment.

5.3 Discussion

From the experimental results, it was confirmed that there is a trade-off between processing speed and generation quality in schedule generation for life support systems.

The improvement in message quality in Pipeline 3 is presumed to be due to the efficiency of token allocation as mentioned above. In batch generation like Pipeline 1 and Pipeline 2, resources are allocated to maintaining the structure of the entire schedule, but by performing individual generation like Pipeline 3, there is room to delve deeper into the user's intention behind each task. In this research where accuracy and kindness of settings are required rather than a few seconds of waiting time, this characteristic is a major advantage.

A point worth noting is the system's judgment regarding lighting operation. The user said, "I want to turn off the lights myself at bedtime," and the system complied with this constraint. Furthermore, even in the going-out scenario, the system did not automatically turn off the lights but stopped at a notification saying "Please turn off the lights yourself as needed." This can be interpreted as a result of the system inferring from the bedtime statement that "the user has the physical and cognitive operational ability to operate the light switch." The judgment to entrust what the user can do to the user instead of automating everything can be highly evaluated as a form of support that does not hinder independence, which this research aims for.

From the above, it is concluded that Pipeline 3, which ensures comprehensiveness and safety and enables adaptive behavior according to user ability, is most suitable as a life support system in a real environment.

6 Advantages and Limitations

This study demonstrated several key advantages of the proposed dialogue-to-schedule generation framework. By utilizing an iterative reasoning approach with an intermediate 6W1H representation (Pipeline 3), the system achieved high task coverage and generated context-aware notifications. Most importantly, the system successfully balanced life support with user autonomy—such as allowing users to manually operate lighting—thereby preventing excessive automation that could lead to learned helplessness.

Despite these advantages, as a preliminary study verifying the feasibility of LLM-based home automation, several issues remain for actual operation.

First, this research is at the stage of feasibility verification using a prototype system, and the experiment is also limited based on specific scenarios. In this evaluation experiment, simulation dialogue logs created based on standard scenarios were used to simplify input to the system. However, in an actual operating environment, it is necessary for the elderly user and the system to interact in real time. Therefore, robustness against speech recognition errors and implementation of a function for the agent to ask back when the user's utterance intention is unclear are indispensable. In the future, it is necessary to implement

a voice dialogue interface and verify acceptance and usability in long-term use through field experiments targeting actual elderly subjects. Integrating a robust conversational interface will significantly lower the adoption barrier for elderly individuals with limited digital literacy, allowing them to naturally communicate their needs without the anxiety of operating complex devices.

Second is the response to life changes and sudden events. In the current implementation, the schedule (Cron setting) generated by the initial dialogue is static and is not automatically changed once set. However, in actual life, dynamic changes frequently occur, such as changes in wake-up/bedtime due to seasonal changes, review of life rhythm due to changes in physical condition, and one-time schedule changes.

In the current system, in order to respond to these changes, it is necessary to start dialogue from scratch again and regenerate the schedule, which is a heavy burden for the user. Therefore, as a future prospect, we plan to add a function that allows easy modification and correction of generated schedules later. Through this, we aim to realize a flexible support system that accommodates the fluctuations of daily life, rather than rigid automation. Ultimately, by overcoming these current limitations, the proposed system will evolve into a personalized AI assistant that continuously adapts to the user's aging and daily changes. Providing such sustained, long-term support will not only extend the period during which the elderly can safely and comfortably live independently, but also drastically reduce the mental and physical monitoring burden on families and caregivers, thereby contributing to the realization of a more resilient aging society.

7 Conclusion

In this paper, toward the realization of a personalized home automation service that supports the independent life of elderly people living alone, we proposed a system that automatically generates device control schedules from dialogue using a large language model. The proposed system has the function of extracting support needs (6W1H) from natural dialogue logs with the user and converting them into specific API schedules specifically controlling physical devices and voice notifications.

As a result of evaluation experiments using a prototype, it was confirmed that the approach of iteratively performing generation via an intermediate representation (Pipeline 3) is most effective in terms of task coverage and consideration for the user, rather than simple batch conversion. In particular, the behavior of accurately picking up the user's intention of "wanting to operate it myself" and providing only necessary support while suppressing excessive automation is an important characteristic for maintaining elderly people's abilities and protecting their dignity. In the future, we will verify the acceptance of the system through the implementation of the dynamic schedule change function mentioned in the previous chapter and long-term experiments in actual fields, contributing to the

realization of a smart society where the elderly can live with peace of mind while enjoying the benefits of technology.

Acknowledgments. This research was partially supported by JSPS KAKENHI Grant Numbers JP25H01167, JP25K02946, JP25K24389, JP24K02765, JP24K02774, JP23K17006, JP23K28091, JP23K28383.

References

1. Cabinet Office, G.o.J.: The 5th science and technology basic plan. https://www8.cao.go.jp/cstp/kihonkeikaku/5honbun.pdf (2016), accessed 2026-02-03
2. Cella, D.F.: Quality of life: concepts and definition. Journal of pain and symptom management **9**(3), 186–192 (1994)
3. Dam, S.K., Hong, C.S., Qiao, Y., Zhang, C.: A complete survey on llm-based ai chatbots. arXiv preprint arXiv:2406.16937 (2024)
4. Nakata, T., Nakamura, M., Chen, S., Saiki, S.: Needs companion: A novel approach to continuous user needs sensing using virtual agents and large language models. Sensors **24**(21: 6814) (October 2024), doi.org/10.3390/s24216814
5. OpenAI: Introducing gpt-5.2. https://openai.com/index/introducing-gpt-5-2-en/ (2025), accessed 2026-02-14
6. SwitchBot: Switchbot. https://switchbot.jp/, accessed 2026-02-03
7. SwitchBot: Switchbot hub 2. https://www.switchbot.jp/products/switchbot-hub2, accessed 2026-02-03
8. Tokunaga, S., Tamamizu, K., Saiki, S., Nakamura, M., Yasuda, K.: VirtualCare-Giver: Personalized smart elderly care. International Journal of Software Innovation (IJSI) **5**(1), 30–43 (October 2017), dOI: 10.4018/IJSI.2017010103
9. Zheng, Q., Wang, W.: The relationship between the digital divide and the well-being of older adults: The mediating role of learned helplessness and the moderating role of growth mindset. Current Psychology **43**(25), 21547–21556 (2024)