一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
KBSE2025-57(2026-03)

独居高齢者を支援する個人適応型ホームオートメーションサービス

対話による要求抽出と制御スケジュールの生成

村手　亮太†　松川　晃徳†　中田　匠哉†　陳　思楠†　安田　清†
中村　匡秀†

† 神戸大学 〒657-8501 神戸市灘区六甲台町 1-1
E-mail: †{rmurate,matsuaki}@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

あらまし　近年，独居高齢者の増加に伴い，IoT や AI を活用した生活支援技術が注目されている．しかし，既存のスマートホーム機器は操作が複雑であり，デジタルデバイドを助長する一因となっている．また，ユーザの意図を介さない過度な自動化は，高齢者の自立性を損ない，心身機能の低下を招く恐れがある．そこで本稿では，大規模言語モデル（LLM）を活用し，自然言語対話を通じてユーザの要望や能力に適応した制御スケジュールを生成するホームオートメーションシステムを提案する．提案手法では，対話ログから支援ニーズ（6W1H）を抽出し，反復的な推論プロセスを経て具体的な API 実行計画へと変換する．プロトタイプを用いた評価実験の結果，提案手法は単純な一括生成と比較してタスクの網羅性が高く，かつ「自分で操作したい」というユーザの拘束条件を遵守した自立支援型のスケジュール生成が可能であることを示した．

キーワード　独居高齢者，ホームオートメーション，大規模言語モデル，LLM，生活支援，対話システム

Ryouta MURATE†, Akinori MATUKAWA†, Takuya NAKATA†, Sinan CHEN†, Kiyoshi YASUDA†, and Masahide NAKAMURA†

† Kobe University, 1-1 Rokkodai-cho, Nada-ku, Kobe, 657-8501, Japan
E-mail: †{rmurate,matsuaki}@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

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.

Key words　Elderly living alone, Home automation, Large language models, LLM, Life support, Dialogue systems

1. はじめに

近年，我が国における少子高齢化の進行は著しく，介護人材の不足や独居高齢者の増加が深刻な社会課題となっている．これに対し，IoT（Internet of Things）や AI 技術を活用して高齢者の生活を支援するスマートホームや見守りサービスの開発が活発に行われている [1]．これらの技術は，高齢者の QoL（Quality of Life）の維持・向上や，介護者の負担軽減に寄与すると期待されている．

しかし，既存の多くのスマートホーム機器やサービスは，デジタル機器の操作に不慣れな高齢者にとって，導入と設定のハードルが高いという課題がある．スマートフォンやタブレットを用いた複雑なアプリ操作は，認知機能や視力が低下した高齢者にとって大きな負担となり，技術の恩恵を受けられないデジタルデバイドを生む要因となっている．一方で，操作負担を解消するために全てを自動化することは，必ずしも正解ではない．利用者の意図を介さない過度な自動化は，高齢者が自ら考え行動する機会を奪い，身体的・認知的機能の低下や学習性無力感を招くリスクがあるからである．したがって，真に高齢者を支援するシステムには，ユーザの意図や能力に合わせて支援と自立のバランスを適応的に調整できる機能が求められる．

この課題に対し，近年急速に発展している大規模言語モデル（LLM）は，自然言語による対話を通じてユーザの曖昧な要望を汲み取り，システムへの命令に変換するインターフェースとして有望である．もし，高齢者が日常会話のようにシステムと対話し，生活のリズムや自分でやりたいこと・やってほしいことを伝えるだけで，背後で複雑な IoT 機器の設定が完了するならば，上記の問題を解決できる可能性がある．しかし，確率的に動作する LLM を用いて，厳密な動作が求められる IoT 機器の制御スケジュールを正確かつ安全に生成できるかについては，十分な検証が必要である．

そこで本研究では，独居高齢者を対象とした「個人適応型ホームオートメーションサービス」の実現に向け，対話から生活支援要求を抽出し，具体的な制御スケジュールを生成するシステムを提案する．本研究では特に，以下の 3 点に注目する．

- 正確かつ網羅的な生成: ユーザの曖昧な日常会話から，情報の欠落なく論理的整合性の取れた IoT 制御スケジュールを生成できるか．
- 推論プロセスの有効性: 6W1H（中間表現）の抽出と反復的な推論プロセスを組み合わせることで，生成の質が向上するか．
- 自立支援の適合性: ユーザの「自分でやりたい」という拘束条件を遵守し，過度な自動化を抑えた適応的な支援が可能か．

本システムは，ユーザとエージェントの対話ログから，いつ（When），何を（What），どのように（How）支援すべきかという構造化データを抽出し，既存のスマート家電（SwitchBot [2] 等）や音声通知システムを制御するための実行可能な API スケジュールへ変換する．提案手法の特徴として，LLM が一度に全てのスケジュールを生成するのではなく，中間表現（6W1H）を経由し，対話の文脈やユーザの拘束条件（例：「照明は自分で消したい」等）を考慮しながら反復的にスケジュールを精緻化するアプローチをとる．これにより，ユーザの自立性を尊重した支援設定の自動化を目指す．

本稿では，提案システムのプロトタイプを実装し，独居高齢者を想定したシナリオに基づく評価実験を行う．実験では，単純な変換手法と，文脈を考慮した段階的な生成手法を比較し，タスクの網羅性や生成されるメッセージの質について検証する．その結果，提案手法を用いることで，ユーザの曖昧な発話から論理的に整合性の取れたスケジュールを生成できること，およびユーザの能力を阻害しない適切な支援範囲の設定が可能であることを示す．

2. 準　　備

2.1 超スマート社会と生活支援の現状

内閣府の第 5 期科学技術基本計画 [1] 等で提唱される Society 5.0（超スマート社会）では，年齢や性別，地域に関わらず，あらゆる人が質の高いサービスを受け，快適に暮らす社会の実現が目指されている．特に，急速な少子高齢化が進む現代において，AI や IoT 技術を活用した生活支援システムは，高齢者の QoL（Quality of Life）維持 [3] や介護負担の軽減において重要な役割を担う．現在，第 4 次産業革命 [4] の流れの中で数多くの支援サービスが登場しているが，これらを実際の高齢者の生活に導入するには，まだ大きな障壁が存在する．

2.2 生活支援システムにおけるジレンマ

既存のスマートホームや生活支援システムには，相反する二つの課題が存在する．

第一に，設定と操作の複雑性である．現在のスマートサービスは，スマートフォンや PC 上の個別アプリケーションを介して提供されるのが一般的である．しかし，機器ごとに異なる操作の学習や，細かな設定作業は，デジタル機器に不慣れな高齢者にとって大きな認知負荷となる．結果として，有用な機能であっても利用が敬遠されるデジタルデバイドの問題が生じている．

第二に，過度な自動化による弊害である．操作負担を減らすための解決策として，センサー等を用いた完全自動化が挙げられる．しかし，ユーザの意図を介さない過度な自動化は，利用者が自ら思考し行動する機会を奪うことにもつながる．これは心理学における学習性無力感 [5] に類する状態を誘発し，身体的・認知的機能の低下を招くリスクがある．

したがって，理想的な支援システムとは，単に全てのタスクを自動代行するのではなく，ユーザの状態や意図に合わせて支援と自立のバランスを調整できるシステムである．

2.3 関連研究と LLM の可能性

従来，スマートホームのコンテキスト認識や制御には，オントロジーや論理プログラミングを用いたルールベースのアプローチが採用されてきた．例えば Alirezaie らは，異種センサーデータを統合し，活動認識を行うフレームワークを提案している [6]．しかし，こうした手法は専門家による知識定義（Knowledge Engineering）を必要とし，エンドユーザ自身が，自身の生活変化に合わせてシステムを柔軟に変更・設定するこ

とは極めて困難である．

近年，大規模言語モデル（LLM）[7] の発展により，自然言語による指示からプログラムやアクションを生成する試みが行われている．LLM を用いれば，高齢者が自然な対話を通じてシステムの設定を行える可能性がある．しかし，LLM は確率的な出力を行うため，ユーザの曖昧な指示を，厳密な実行スケジュール（日時やパラメータ）に正確に変換できるか，またその論理的整合性が保たれるかについては，十分な検証が必要である．

2.4 着目する課題

本研究では，最終的な目標として，ユーザとエージェント [8] の対話を通じて，個人の生活様式や認知レベルに合わせた自動支援を行う「適応的生活支援システム」の構築を目指している．IoT 機器の複雑な設定をエージェントが代行しつつ，すべてを自動化するのではなく，対話を挟むことでユーザが自身の生活管理に関与し続ける環境である．

この大きな目標を実現するための第一歩として，本論文では「LLM を用いて，ユーザの自立性を損なわず，かつ厳密な実行が求められる IoT 制御スケジュールを，いかに欠落なく生成するか」という課題に着目する．単一のプロンプトで長い対話ログからスケジュールを生成しようとすると，情報の欠落やユーザの細かな拘束条件（自立支援に関する要望）の無視が発生するという技術的な困難さがある．対話によってユーザのニーズを引き出し，それを既存の IoT ツールセットと組み合わせて，実行可能なスケジュールへと落とし込む能力は，システム実現の核となる機能である．特に，曖昧な日常会話から，正確な API コールのパラメータと実行タイミングを導出できるかは自明ではない．

2.5 研究目的とアプローチ

本稿では，適応的支援システムの実現に向けた予備的検討として，LLM を用いた「対話からのスケジュール生成・実行フレームワーク」の実現可能性を検証するとともに、生成の正確性と網羅性を高めるための反復的推論アプローチの有効性を明らかにすることを目的とする．

具体的には，ユーザと対話エージェントが行う支援ニーズに関する対話ログとシステムが操作可能なデバイスや機能の定義 (ツールカタログ) の 2 点を入力とし，LLM がどのようにタスクをスケジュールするかを調査する．

アプローチとして，自然言語対話から 6W1H (When, Where, Who, What, Why, Whom, How) を構造データとして抽出し [9]，それを API 実行スケジュールへ変換するパイプラインを構築する．本稿での検証範囲は，この変換プロセスが論理的に破綻なく機能するか，および生成されたスケジュールが意図通りに動作するかを確認することである．これにより，将来的な「高齢者と共生する適応的システム」の技術的基盤を確立する．

3. 提 案 手 法

本章では，前章で述べた課題を解決するための生活支援システムの全体アーキテクチャについて述べる．本システムは，大枠として以下の 3 つの概念モジュールから構成される．

(M1) **対話によるニーズ抽出:** ユーザの自然言語対話から支援要件を理解する．

(M2) **スケジュール生成:** 抽出した要件をシステム実行可能なスケジュールへ変換する．

(M3) **環境デバイス制御:** 物理デバイスや通知サービスを介して実際の支援を行う．

本研究では，これらを実現するために，Conversation to Schedule (C2S)，Scheduler，Executor という 3 つのコンポーネントを連携させるアーキテクチャを提案する．

3.1 全体アーキテクチャ

提案システムの全体構成を図 1 に示す．システムは，ユーザとのインターフェースおよび物理的制御のエンドポイントを除き，サーバーサイドで疎結合に連携するマイクロサービス構成をとる．

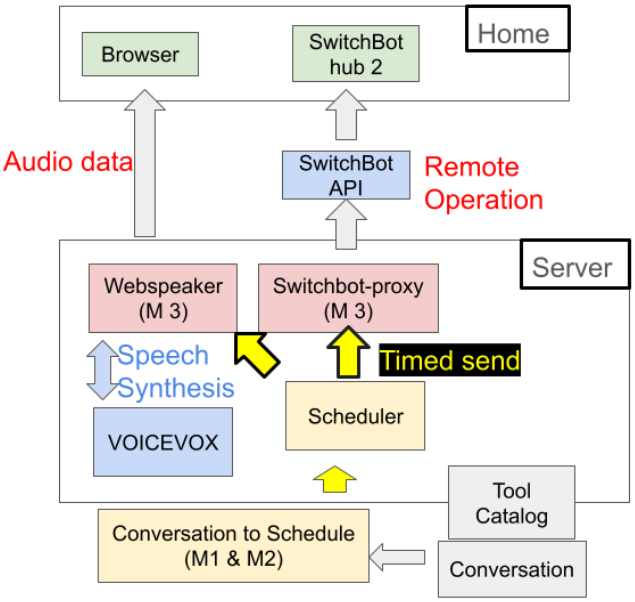

図 1 提案システムの全体アーキテクチャ

各コンポーネントの役割は以下の通りである．

- **Conversation to Schedule (C2S):** 概念モジュール (M1) および (M2) を担当する．LLM を内包し，ユーザとの対話ログと，システムが操作可能な「ツールカタログ」を入力として受け取り，実行可能なスケジュールデータを出力する．
- **Scheduler:** C2S によって生成されたスケジュールをデータベースに保持し，時間管理を行う．所定の時刻に到達した際，対応するジョブをトリガーし，後段の Executor へ実行命令を送信する．
- **Executor:** 概念モジュール (M3) を担当する．Scheduler からの実行命令を受け取り，実際のデバイス操作を実行する．

3.2 Conversation to Schedule (C2S): コンテキスト理解とスケジュール生成

C2S は，非構造データである会話を，システムが解釈可能な構造化スケジュールへ変換する役割を担う．

3.2.1 生成プロセスとデータ構造

本コンポーネントは，対話ログとツール定義を入力として受け取り，以下の手順で処理を行う．

（1） **プロンプト構築:** システムプロンプトに対し，現在時刻，ユーザ情報，およびツールカタログを動的に埋め込む．

（2） **推論と JSON 生成:** LLM に対してリクエストを送信し，ニーズ抽出結果（6W1H 形式）および実行スケジュールを JSON フォーマットで出力させる．

生成されるスケジュールデータの実行タイミング定義には標準的な Cron 式を採用し，LLM に対して「毎朝 7 時」を ０ ０ ７ * * * のように変換させるよう指示を与える．これにより，プログラマブルかつ柔軟な繰り返し設定を実現する．

3.2.2 スケジュール生成のアプローチ

抽出されたニーズを実際の実行計画に変換するにあたり，LLM の推論プロセスにおける「情報の構造化」および「処理の分割」の有効性を段階的に評価するため，以下の 3 つの手法を比較・検討可能な設計とした．

手法 1: 直接変換 (One-Shot Mapping)
会話ログとツールカタログを入力とし，中間表現を経由せず，一度の推論で直接スケジュールを出力する手法．

手法 2: 6W1H 経由の変換 (Two-Stage Generation)
一度 6W1H 形式の中間ファイルを出力し，整理された情報を元にスケジュールを生成する 2 段階の手法．

手法 3: ループ処理による個別検討 (Iterative Elaboration)
手法 2 と同様に対話ログから全ての支援要件を 6W1H 形式で抽出した後，各要求に対して個別にスケジュール生成を行う手法．抽出された 6W1H のリストから一つ（例：起床時の支援要件）を取り出し LLM に送信する．LLM はその入力に対し，必要な具体的なアクション群（例：呼びかけ「おはようございます」，照明 ON，テレビ ON，エアコン ON の 4 動作）を生成する．次に出かける項目があればその 6W1H のみを送信し同様に処理するなど，この手順を全ての要件に対して反復的に適用することで，最終的なスケジュールを構築する．

3.3 Scheduler: 実行管理ロジック

Scheduler は，システムの心臓部としてスケジュールの永続化と実行タイミングの監視を担う．

3.3.1 スケジュールの定義

C2S から受け取るスケジュール登録リクエストは，以下のフィールドを持つクラスとして定義されている．

- `apiName`: ジョブの名称
- `cron`: 実行タイミングを示す Cron 式
- `endpoint`: 実行対象となる Executor の URL
- `method`: HTTP メソッド（GET/POST 等）
- `requestBody`: 送信するペイロード
- `headers`: 必要な HTTP ヘッダ情報

3.3.2 監視と実行

Scheduler は 1 分ごとに内部タスクを起動し，データベースに保存されたジョブの中から，現在時刻が `cron` 定義と一致するものを抽出する．抽出されたジョブは即座に非同期プロセスへ回され，指定された `endpoint` に対して実行リクエストを送信する．

3.4 Executor: プロキシによる抽象化と制御

Executor は，Scheduler からの抽象的なリクエストを受け，物理世界への作用を行うインターフェース群である．本研究では，セキュリティと生成コストの観点から，物理デバイスを直接 LLM に操作させるのではなく，プロキシ（代理）サーバーを介する構成を提案する．

3.4.1 デバイス ID と API トークンの管理

照明やエアコン等の操作において，Scheduler および C2S が生成するデータは，各家電のデバイス ID のみを保持する．Executor 側のデータベースには，この論理 ID に対し，実デバイスのデバイス ID およびクラウド API へのアクセスに必要な「API トークン」が紐付けて管理されている．Executor はリクエストを受け取ると，論理 ID をキーとして DB を検索し，物理 ID への置換と正規の認証情報付与を行って外部クラウド API へコマンドを中継する．

この構成には以下の二つの利点がある．

（1） **セキュリティの担保:** LLM のプロンプトおよび生成出力に API トークン等の機密情報を含めないことで，プロンプトインジェクションによる漏洩リスクや，ハルシネーションによる誤った認証情報の送信を確実に排除する．

（2） **生成コストの削減:** 文字列が長くなるトークンをクライアントシークレットを LLM に扱わせず，短い論理 ID のみを出力させることで，出力トークン数を大幅に削減する．

3.4.2 音声通知の制御

ユーザへの呼びかけを行う音声通知においても，リアルタイム性を確保するため，Executor が音声合成エンジンと連携し，生成された音声データを WebSocket を通じてクライアントへ即座にプッシュ送信する仕組みを採用する．

4. 実　　装

本章では，前章で提案したアーキテクチャに基づくプロトタイプシステムの実装環境について簡潔に述べる．なお，本研究の主目的は対話内容からスケジュールを導出する論理の検証にあるため，本プロトタイプでは C2S，Scheduler，Executor のバックエンド処理の実装に注力した．ユーザとのリアルタイムな音声対話機能（音声認識や対話 UI 等のフロントエンド）は本稿の実装範囲外とし，システムへの入力はテキスト形式の対話ログを直接与える構成とした．本システムは，各コンポーネントを Docker コンテナとして独立させ，docker-compose を用いてオーケストレーションを行うマイクロサービス構成で実装した．

4.1 開発環境と技術スタック

表 1 に，本システムの実装に用いた主な技術スタックを示す．AI 処理には柔軟な Python を，堅牢性が求められるスケジューリングと DB 操作には Java (Spring Boot) を採用し，REST API および WebSocket を通じて連携させた．

4.2 各コンポーネントの実装詳細

4.2.1 C2S の実装

LLM として推論設定と長文コンテキストの深い理解に長けた

表 1　実装環境と技術スタック

Component	Technology Stack
C2S	Python, OpenAI API (GPT-5.2)
Scheduler	Java, Spring Boot, MySQL
SwitchBot-proxy	Java, Spring Boot, MySQL
WebSpeaker (Back)	Python, FastAPI, VOICEVOX
WebSpeaker (Front)	TypeScript, Vite, Bootstrap
Infrastructure	Docker, Docker Compose

OpenAI 社の 2025 年 12 月 11 日時点での最新モデル GPT-5.2 [10] を採用した．生成された JSON データは，ストレージにログとして保存する．

4.2.2　Scheduler および Executor の実装

Scheduler および SwitchBot-proxy は Spring Boot や JPA を用いて実装し，定期実行と DB 操作の信頼性を確保した．特に Scheduler は，Cron 式の解釈と，Spring Framework の`@Scheduled` アノテーションを活用して実装した．WebSpeaker の音声合成には VOICEVOX エンジンを使用し，フロントエンドは Web Audio API を用いてブラウザ上での再生を実現し，字幕は Bootstrap を用いて実装したブラウザベースの UI に表示する構成とした．

5. 評価実験

本章では，提案システムおよび 3 つのスケジュール生成手法の有効性を検証するための実験について述べる．実験は，架空の高齢者ユーザとの対話シナリオを用い，実際に生成されたスケジュールの正確性，網羅性，および生成時間を比較評価した．

5.1 実験設定

5.1.1　シナリオと被験者設定

実験の対象として，80 歳代の男性・独居高齢者「佐藤さん」を想定した．佐藤さんは基本的な自立生活を送っているが，機器操作への不慣れや，外出時の消し忘れ不安，服薬管理の課題を抱えている．

なお，本実験では提案手法によるスケジュール生成の正確性を定量的に評価するため，実際の高齢者を被験者とした対話実験は行わず，想定されるシナリオに基づきあらかじめ作成した「模擬対話ログ」を入力データとして用いた．具体的には，起床，服薬，外出，就寝の 4 つの生活シーンについて，エージェントがヒアリングを行い，支援内容を決定する流れを会話ログとして記述した．表 2 に C2S によって抽出されるべき要件の正解データを示す．

表 2　対話から抽出されるべき支援要件

シーン	時刻・条件	期待されるアクション
起床	月・木 05:00, 他 05:30	TV, 照明, 暖房 ON, 音声通知
服薬	毎日 07:30, 18:30	音声通知のみ
外出	月・木 08:30, 水 09:00	TV, 照明, 暖房 OFF, 音声通知
就寝	毎日 22:00	TV, 暖房 OFF, 音声通知

5.1.2　実験環境

著者の住環境において, SwitchBot Hub 2 [11], WebSpeaker, およびエアコン・照明・テレビの実機を設置し，提案システムの各コンポーネントを稼働させた．

5.2 実験結果

表 3 に，3 つの手法による生成結果の比較を示す．ここで，各指標の定義は以下の通りである．タスク網羅率とは，表 2 で定義した全支援要件（計 16 項目）のうち，システムが欠落なくスケジュールを生成できた項目の割合である．生成時間とは，対話ログとツール定義を入力としてから，最終的なスケジュール（JSON データ）が出力されるまでの処理時間である．API 往復時間を含む全体の処理時間を計測した．記述詳細度とは，生成された音声通知メッセージ等が，単なる事実の羅列ではなく，対話の文脈やユーザへの配慮を含んだ内容になっているかどうかの定性的な評価である．

表 3　手法ごとの評価結果比較

評価項目	手法 1	手法 2	手法 3
タスク網羅率	75.0% (12/16)	100% (16/16)	100% (16/16)
生成時間	26.4 秒	46.2 秒	57.5 秒
記述詳細度	低	低	高

5.2.1　手法 1: 直接変換 (One-Shot Mapping)

生成時間は 26.4 秒と最も高速であった．しかし，出力された JSON を確認すると，水曜までの起床・服薬・外出のスケジュールは生成されていたものの，会話の最後にある就寝時のスケジュールが完全に欠落していた．これは，LLM が一度の推論で長文の会話と複雑なツール定義を同時に処理する際，コンテキスト後半への注意が散漫になる，あるいは出力トークン長の制約により生成が打ち切られた可能性が考えられる．この結果から，複雑な生活支援においては単純な E2E アプローチは信頼性に欠けることが示された．

5.2.2　手法 2: 6W1H 経由の変換 (Two-Stage Generation)

生成時間は 46.2 秒であった．手法 1 で欠落した就寝時のタスクを含め，全ての要件を網羅することに成功した．特に，就寝時の「照明は自分で消す」というユーザの要望に対し，テレビとエアコンの OFF のみをスケジュールし，照明の OFF コマンドを含めなかった点は正しく評価できる．中間表現（6W1H）を経由することで，LLM が一度情報を整理・保持でき，タスクの抜け漏れを防げたと考えられる．

5.2.3　手法 3: ループ処理による個別検討 (Iterative Elaboration)

生成時間は 57.5 秒と最も長くなったが，質的な面で最も優れた結果となった．手法 2 と同様にタスク網羅と制約遵守を達成した上で，生成された音声通知のメッセージの内容に顕著な違いが見られた．

手法 1・2 のメッセージが「テレビと照明をつけました」といった事実の列挙に留まったのに対し，手法 3 では以下のように文脈を汲み取った生成が行われた．

> 〃おはようございます．テレビと照明，エアコンの暖房をつけました．足元に気をつけて起きてください．〃

〃外出前の確認です．… 照明は必要に応じてご自身で消してください．〃

これは，タスクを個別に処理することで，出力トークン数の制約による影響を受けにくくなり，推論能力をタスクごとの文脈理解やメッセージ生成へ十分に注力できたためと考えられる．

5.2.4 実機による動作検証

手法 3 で生成された JSON データを本システムの Scheduler に登録し，動作検証を行った．その結果，設定された Cron 時刻（05:00, 22:00 等）において SwitchBot API および WebSpeaker へのリクエストが正常にトリガーされ，実際の家電操作と音声通知が行われることを確認した．図 2 は，実際の動作中の様子を撮影したものである．テレビはシステムにより起動され，WebSpeaker からは「おはようございます．テレビと照明，エアコンの暖房をつけました．足元に気をつけて起きてください．」という音声が再生されている．

図 2 提案システムの実機動作検証．

5.3 考　　察

実験結果より，生活支援システムのスケジュール生成において，処理速度と生成品質の間にはトレードオフが存在することが確認された．

5.3.1 生成プロセスの違いによる品質への影響

手法 3 におけるメッセージ品質の向上は，前述の通りトークン配分の効率化によるものと推察される．手法 1 及び手法 2 のような一括での生成ではスケジュール全体の構造維持にリソースが割かれるが，手法 3 のように個別生成を行うことで，各タスクの背景にあるユーザの意図を深掘りする余裕が生まれる．数秒の待機時間よりも設定の正確さと親切さが求められる本研究において，この特性は大きな利点である．

5.3.2 ユーザ能力の推定と自立支援

特筆すべき点は，照明操作に関するシステムの判断である．ユーザは「就寝時の照明は自分で消したい」と発言しており，システムはこの制約を遵守した．さらに，外出時のシナリオにおいても，システムは照明を自動消灯せず，「必要に応じてご自身で消してください」という通知に留めた．これは，就寝時の発言から「ユーザには照明スイッチを操作する身体的・認知的操作能力がある」とシステムが推論した結果であると解釈できる．全てを自動化するのではなく，ユーザができることはユーザに委ねるという判断は，本研究が目指す自立を妨げない支援の形として高く評価できる．

以上より，実環境における生活支援システムとしては，網羅性と安全性を担保し，かつユーザの能力に応じた適応的な振る舞いが可能な手法 3 が最も適していると結論付ける．

6. 研究の限界と今後の展望

本稿では，LLM を用いた対話型ホームオートメーションの実現可能性を検証するための予備的検討を行った．提案手法により，ユーザの曖昧な発話から論理的な制御スケジュールを生成できることが示されたが，実運用に向けてはいくつかの課題が残されている．本章では，本研究の限界と今後の展望について述べる．

6.1 実験の限定性と対話インターフェース

第一に，本研究はプロトタイプシステムを用いた実現可能性検証の段階にあり，実験も特定のシナリオに基づいた限定的なものである．今回の評価実験では，システムへの入力を単純化するため，標準的なシナリオに基づき作成されたシミュレーション対話ログを使用した．しかし，実際の運用環境では，高齢者ユーザとシステムがリアルタイムに対話を行う必要がある．そのため，音声認識の誤りに対する堅牢性や，ユーザの発話意図が不明確な場合にエージェント側から聞き返しを行う機能の実装が不可欠である．今後は，音声対話インターフェースを実装し，実際の高齢者被験者を対象としたフィールド実験を通じて，長期的な利用における受容性やユーザビリティを検証する必要がある．

6.2 スケジュールの動的な変更と適応

第二に，生活の変化や突発的なイベントへの対応である．現在の実装では，初期対話によって生成されたスケジュール（Cron 設定）は静的なものであり，一度設定されると自動的には変更されない．しかし，実際の生活では季節の移り変わりによる起床・就寝時間の変化や，体調の変化による生活リズムの見直し，日限りの予定変更など，動的な変化が頻繁に発生する．

現状のシステムでは，これらの変化に対応するためには，再度ゼロから対話を行いスケジュールを再生成する必要があるが，これはユーザにとって大きな負担となる．したがって，今後の展望として，生成済みのスケジュールに対して，あとから変更・修正を容易に行える機能の追加を予定している．具体的には，ユーザが「今日はご飯を食べて帰るよ」と話しかけるだけで，その日の夕食時の音声通知や家電制御のみを一時的にスキップする例外処理や，「最近は寒くなったから暖房を早めにつけて」といった対話から，既存の Cron 設定を部分的に更新する差分適用ロジックを C2S モジュールに組み込む．これにより，硬直的な自動化ではなく，生活の揺らぎに寄り添う柔軟な支援システムの実現を目指す．

7. ま と め

本稿では，独居高齢者の自立した生活を支援する個人適応型ホームオートメーションサービスの実現に向け，大規模言語モデルを用いて対話から機器制御スケジュールを自動生成するシ

ステムを提案した．提案システムは，ユーザとの自然な対話ログから支援ニーズ（6W1H）を抽出し，物理デバイスや音声通知を制御する具体的な API スケジュールへと変換する機能を持つ．

プロトタイプを用いた評価実験の結果，単純な一括変換よりも，中間表現を経由して反復的に生成を行うアプローチ（手法 3）が，タスクの網羅性とユーザへの配慮において最も有効であることを確認した．特に，ユーザの「自分で操作したい」という意図を正確に汲み取り，過度な自動化を抑制しつつ必要な支援のみを提供する挙動は，高齢者の能力維持と尊厳を守る上で重要な特性である．今後は，前章で述べた動的なスケジュール変更機能の実装や，実フィールドでの長期実験を通じてシステムの受容性を検証し，高齢者がテクノロジーの恩恵を享受しながら安心して暮らせるスマート社会の実現に貢献する．

謝辞 本研究の一部は JSPS 科研費 JP25H01167, JP25K02946, JP25K24389, JP24K02765, JP24K02774, JP23K17006, JP23K28091, JP23K28383 の研究助成を受けて行われている．

文　献

[1] 内閣府，“第 5 期科学技術基本計画，” https://www8.cao.go.jp/cstp/kihonkeikaku/5honbun.pdf，2016. accessed 2026-02-03.

[2] siwtchbot, “Switchbot，” https://switchbot.jp/. accessed 2026-02-03.

[3] D.F. Cella, “Quality of life: concepts and definition，” Journal of pain and symptom management, vol.9, no.3, pp.186–192, 1994.

[4] acatech National Academy ofScience and Engineering, “Recommendations for implementing the strategic initiative industrie 4.0，” https://www.acatech.de/publikation/recommendations-for-implementing-the-strategic-initiative-industrie-4-0-final-report-of-the-industrie-4-0-working-group/, 2013. accessed 2026-02-03.

[5] Q. Zheng and W. Wang, “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, vol.43, no.25, pp.21547–21556, 2024.

[6] M. Alirezaie, J. Renoux, U. Köckemann, A. Kristoffersson, L. Karlsson, E. Blomqvist, N. Tsiftes, T. Voigt, and A. Loutfi, “An ontology-based context-aware system for smart homes: E-care@ home，” Sensors, vol.17, no.7, p.1586, 2017.

[7] S.K. Dam, C.S. Hong, Y. Qiao, and C. Zhang, “A complete survey on llm-based ai chatbots，” arXiv preprint arXiv:2406.16937, pp.••–••, 2024.

[8] S. Tokunaga, K. Tamamizu, S. Saiki, M. Nakamura, and K. Yasuda, “VirtualCareGiver: Personalized smart elderly care，” International Journal of Software Innovation (IJSI), vol.5, no.1, pp.30–43, Oct. 2016. DOI: 10.4018/IJSI.2017010103, http://www.igi-global.com/journals/abstract-announcement/158780.

[9] T. Nakata, M. Nakamura, S. Chen, and S. Saiki, “Needs companion: A novel approach to continuous user needs sensing using virtual agents and large language models，” Sensors, vol.24, no.21: 6814, pp.••–••, Oct. 2024. doi.org/10.3390/s24216814.

[10] openai, “Gpt-5.2 が登場，” https://openai.com/ja-JP/index/introducing-gpt-5-2/，2025. accessed 2026-02-14.

[11] siwtchbot, “Switchbot hub 2，” https://www.switchbot.jp/products/switchbot-hub2?srsltid=AfmBOoojudjM9rPpsWw29vFAA6FfRf4HbgWc0lGxt_3IUYWjxQPd8ZDP. accessed 2026-02-03.