Ammu & Mycroft at Home: A Living, Breathing House that Learns Routines*

Kiran Achari

Independent Researcher
Lead System Integrator, Boehm System Engineering GmbH
Correspondence: kiranmech98@gmail.com

August 21, 2025

Abstract

This paper is a technical case study of a complete, working dual-assistant smart home designed to operate as an intuitive, learning companion rather than a static control panel. System architecture splits control between two personas: Ammu, a culturally grounded Malayalam-first assistant for day-to-day household interaction, and Mycroft, a pragmatic English-first systems butler for technical control and planning tasks. Core infrastructure is managed by Home Assistant via a bare-metal Home Assistant OS installation on a Lenovo ThinkCentre M910q Tiny (Intel Core i5-6500T, 32 GB RAM, 512 GB SSD) [32, 33]. Two local voice satellites (Home Assistant Voice Preview Edition and an ESPHome ATOM M5) perform wake-word detection and stream audio to Home Assistant [31, 2, 19]. For on-prem inference, a dedicated local LLM host runs Ollama (e.g., `gpt-oss:20b-instruct`) [29, 30]; cloud backends are used opportunistically (OpenAI for Mycroft, Gemini for Ammu) [5, 6]. Speech I/O uses Azure Speech-to-Text and Azure Neural TTS for low-latency streaming [3, 4], with a local fail-safe using Whisper and Piper. Beyond command execution, the LLM also drafts new automations and synthesizes context-specific scenes from structured state snapshots (calendar, location, time-of-day, weather, and mobile device status) [40, 34, 35, 36, 38, 39, 37]. We report deployment observations on reliability, latency, and practical constraints when mixing cloud and on-prem inference in a bilingual household.

Keywords: smart home; Home Assistant; Malayalam; low-resource languages; voice user interface; Ollama; OpenAI; Gemini; Whisper; privacy; energy efficiency; ambient intelligence; human-computer interaction.

1 Introduction and Research Questions

Smart-home assistants have become widely available, but most consumer systems are either *cloud-dependent* (raising privacy, cost, and latency concerns) or *local-first but brittle* (struggling with natural dialogue, multilingual speech, and culturally grounded phrasing). These limitations are

*This work documents a self-funded, independent research project conducted at a private residence. Project site: Ayanthiara Chitrakoot. Repository: https://github.com/kiranvenom1209/Ammu-AI/tree/v1.0.0-paper-release. Contact: https://www.linkedin.com/in/acharikiran/.
Correspondence: kiranmech98@gmail.com.

amplified in bilingual households where daily interaction spans both *technical* and *domestic* domains, and where language switching is routine rather than exceptional.

This paper presents a working, end-to-end case study deployed in Ayanthiara Chitrakoot, combining two complementary assistants: Ammu for Malayalam-first household interaction and Mycroft for English-first technical and administrative control. A context-aware policy routes speech, reasoning, and actuation across cloud and on-premise backends, with an explicit threat model and least-privilege control through Home Assistant.

1.1 Research Questions

As a technical case study, our goal is to evaluate whether a dual-assistant, hybrid architecture can reliably deliver a ''living, breathing house'' experience without sacrificing privacy and responsiveness. We focus on the following research questions:

- **RQ1 (Reliability):** Can the system achieve high task success across a diverse command set spanning home control, queries, and conversation, including Malayalam and code-switched utterances?
- **RQ2 (Latency):** Can we achieve interactive response times (time-to-first-audio and end-to-end latency) consistent with natural household interaction?
- **RQ3 (Energy at the edge):** What is the *on-premise* energy cost of local fallback inference compared with a cloud-routed pipeline?
- **RQ4 (Usability):** Do household users perceive the system as usable and satisfying for day-to-day routines (measured via standard UX instruments)?

1.2 Contributions

The primary contributions of this work are:

1. A complete, reproducible dual-assistant smart-home architecture deployed in a real household, including hardware, software, and routing logic.
2. A context-aware assistant and backend selection policy (Algorithm 1) that adapts to presence, guest mode, and network conditions.
3. A latency model with empirical measurements across cloud and on-premise pipelines.
4. Preliminary reliability and on-premise energy results, plus an evaluation protocol suitable for future longitudinal studies.
5. A practical privacy/threat model and mitigations based on LAN trust boundaries and least-privilege actuation.

2 Background: Malayalam Nuances and the Limits of Local Models

Malayalam is a Dravidian language spoken by over 38 million people[28]. This language poses major hurdles for today's ASR, NLU, and TTS models, particularly those limited to local hardware [10]. The challenges go beyond just vocabulary size; they are deeply embedded in the language's structure. Understanding these features is essential to support the hybrid cloud/local architecture of this project.

1. **Agglutinative Morphology:** Unlike analytic languages like English, Malayalam makes extensive use of agglutination. In this process, words form by combining morphemes. A long Malayalam word can equal an entire English phrase. For example, ''*pustakam eṭuttutarunnatināyuḷḷa sahayattinu nandi*'' translates to ''Thanks for the help in order to get the book.'' This leads to a vast surface vocabulary, resulting in serious out-of-vocabulary (OOV) issues for less effective tokenizers found in smaller local models. These models often break words into meaningless parts, undermining their meaning [12, 16].

2. **Complex Phonology and Script:** Malayalam's script, which comes from the Grantha script, has a rich set of graphemes that represent phonemic distinctions missing in English. This includes retroflex consonants and gemination (consonant doubling). Achieving accurate grapheme-to-phoneme (G2P) mapping is challenging and crucial for natural-sounding TTS. Local TTS models like Piper often struggle with the correct pronunciation and prosody of these sounds [8, 17].

3. **Pervasive Code-Switching:** In urban and technical settings, speakers often blend English words (especially for devices and brands) into Malayalam sentences. A typical command might be, ''*Hall-ile smart plug off cheyyu*'' (Turn off the smart plug in the hall). Monolingual models have difficulty processing this correctly. They may misunderstand the English noun or the Malayalam verb frame. This increases the need for models with strong cross-lingual understanding, a feature that is more advanced in large frontier models [18].

4. **Sociolinguistic Pragmatics:** Politeness, honorifics, and indirectness appear in the grammar. A direct command like ''Turn on the light'' often becomes a polite request like ''*Light on cheyyāmo?*'' (Could you turn on the light?). A capable assistant should grasp these subtleties to respond appropriately and maintain a culturally suitable persona. This demands advanced reasoning abilities.

These linguistic features explain why Ammu uses Gemini as its main backend. This model performs better with Malayalam's structure and pragmatics, along with the high-quality synthesis of Azure's neural voices. The offline stack (Whisper+Piper), while adequate for simple commands, significantly lowers the user experience.

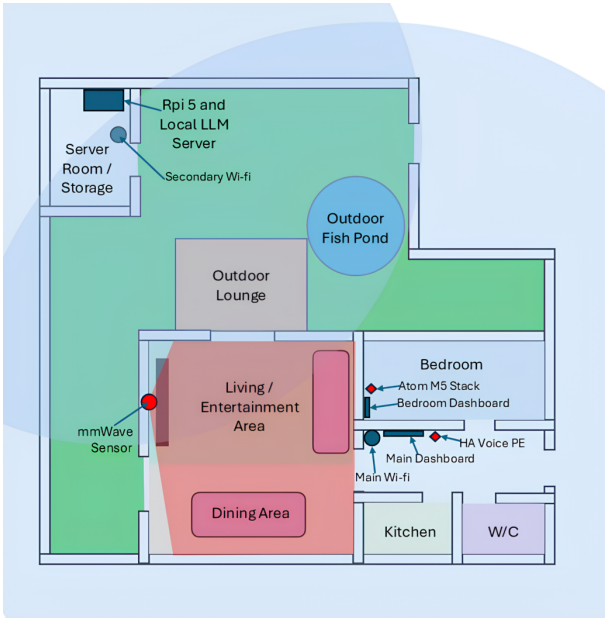


Figure 1: Site layout showing device placements, Wi-Fi coverage, and sensors.

3 System Architecture

The system is designed with a layered approach: a hardware foundation, a software orchestration core, and an intelligent logic layer that routes tasks based on context.

3.1 Hardware Topology

The physical components are selected for reliability, performance, and low idle power consumption.

- **Core Hub:** A Lenovo ThinkCentre M910q Tiny (Intel Core i5-6500T, 32 GB RAM, 512 GB SSD) running Home Assistant OS as a bare-metal Generic x86-64 installation [32, 33].
- **Voice Satellites:** Two dedicated devices capture voice commands.
  1. *Hallway:* A Home Assistant Voice PE node, providing a turnkey solution with integrated speaker, microphone array, and LEDs [1].
  2. *Bedroom:* An ATOM M5 Stack device running custom ESPHome firmware [2]. Its small form factor is ideal for a bedside table.

  Both satellites implement local wake-word detection using `openWakeWord` and stream audio to the Home Assistant core upon activation [19, 20]. They provide visual feedback (listening, thinking, speaking) via their LEDs, which is mirrored on the dashboards.
- **LLM Compute Node:** A MacBook Pro 16'' (2019, Core i9, 16 GB RAM, AMD Radeon Pro 5500M with 8 GB VRAM) runs Ollama in the background. It serves the local model `gpt-oss:20b-instruct` for offline or privacy-sensitive tasks. This externalizes heavy inference from the Home Assistant host, keeping core automations responsive.

- **Displays:** Two repurposed Android devices serve as dedicated dashboards, running the Home Assistant Companion app in kiosk mode.
    1. *Main:* A Xiaomi Pad 5 (11 in) wall-mounted in the living area.
    2. *Bedroom:* A Vivo phone (6 in) on a bedside stand.

3.2 Software and Model Stack

The software stack is a mix of open-source projects and cloud services, chosen to balance performance, privacy, and capability.

Table 1: Core Software Components and Models

Component	Technology	`Backend/Model`	Role
Orchestration	Home Assistant OS 16.1	`|`	Core state machine, automations, UI
Voice Pipeline	Assist Pipeline [1, 21]	`|`	Manages VAD, STT, Intent, TTS flow
Cloud STT	Azure Speech-to-Text [3]	`Streaming API`	Primary Malayalam & English speech-to-text
Cloud TTS	Azure Text-to-Speech [4]	`ml-IN-SobhanaNeural, en-US-ChristopherNeural`	Primary Malayalam & English text-to-speech
Cloud LLM (Ammu)	Gemini API [6]	`2.5 Pro / Flash`	Malayalam reasoning, cultural nuance
Cloud LLM (Mycroft)	OpenAI API [5]	`gpt-4.1 with search`	Complex planning, technical control
Local STT	Whisper [7]	`tiny-int8`	Offline fallback speech-to-text
Local TTS	Piper [8]	`en_US-lessac-medium`	Offline fallback text-to-speech
Local LLM	Ollama [9]	`gpt-oss:20b-instruct`	Offline fallback / privacy mode NLU

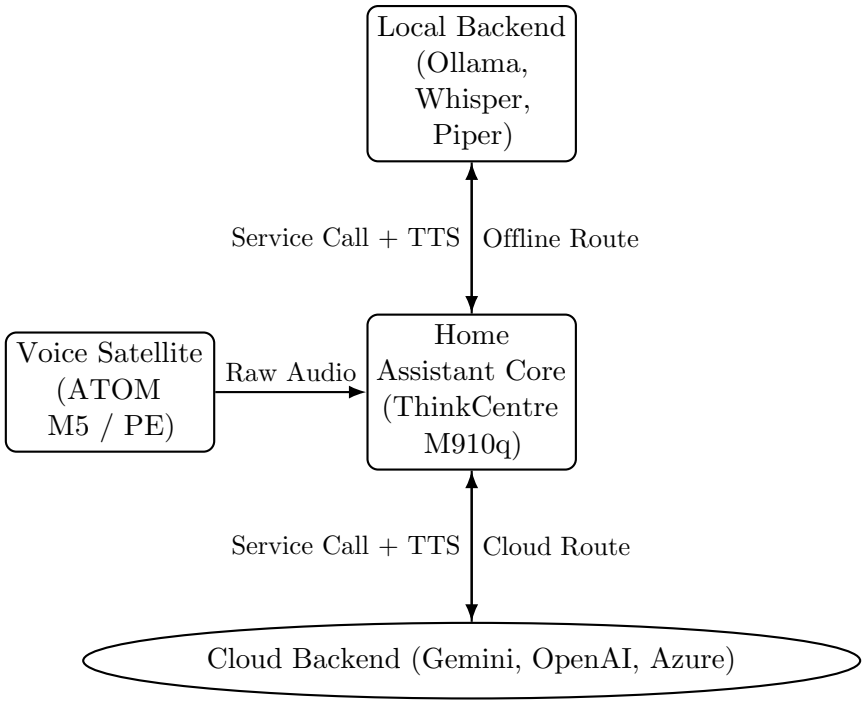


Figure 2: End-to-end data flow for a voice command. The diagram shows the path from wake word detection at the satellite, streaming audio to Home Assistant, routing to either the local or cloud backend based on the policy in Algorithm 1, and finally returning a service call and streaming audio for TTS.

3.3 Context-Aware Assistant Switching Policy

The system dynamically selects the active assistant persona and the compute backend (local vs. cloud) based on real-time context. This logic is implemented as a Home Assistant automation that sets several `input_select` and `input_boolean` helpers.

Algorithm 1 Context-aware Assistant and Backend Selection Logic

Require: `group.family` presence state, `sensor.internet_ping` latency, `binary_sensor.guest_mode`

▷ Default to Mycroft, the systems butler
assistant ← Mycroft
if `group.family` is `home` **and** `binary_sensor.guest_mode` is `off` **then**
    assistant ← Ammu ▷ Switch to Ammu for primary resident interactions
else if `group.family` is `away` **then**
    assistant ← Mycroft ▷ Mycroft handles security/away modes
end if
if `sensor.internet_ping` ¿ 150 ms **or** unavailable **then**
    backend ← LOCAL(Ollama, Whisper, Piper)
else
    if assistant == Ammu **then**
        backend ← CLOUD(Gemini, Azure Speech-to-Text, Azure Text-to-Speech)
    else
        backend ← CLOUD(OpenAI, Azure Speech-to-Text, Azure Text-to-Speech)
    end if
end if
UpdateHACore(assistant, backend) ▷ Set helpers to route pipelines

Guest-mode rationale. Guest mode forces the system to prefer Mycroft's more formal English persona to avoid surprising visitors who may not speak Malayalam, and to reduce the risk of overly-personal domestic dialogue when non-residents are present.

3.4 LLM-Assisted Automation and Scene Synthesis

In addition to executing commands, we use the LLM as an *automation authoring and synthesis agent*. Recent work has shown that LLM-based chatbots can generate usable Home Assistant automation routines from natural-language descriptions [40]. In our deployment, the LLM has two roles:

1. **Automation drafting (human-in-the-loop).** The LLM proposes new Home Assistant automations in YAML (triggers, conditions, actions) following the platform's native automation model [34]. Drafts are reviewed by the homeowner before activation, keeping behavior auditable and avoiding over-permissive control.

2. **Contextual scene synthesis (on-the-fly).** The LLM composes temporary scenes that encode a desired multi-entity state (e.g., lighting, media, ambience) using Home Assistant scenes [35]. Scenes may be activated immediately and, if useful, saved for later refinement.

The synthesis context is built from structured state snapshots inside Home Assistant: (i) environment and device sensors, (ii) calendar constraints and upcoming events via the Google Calendar integration [36], (iii) user presence and location via device tracking [38], (iv) time-of-day and derived schedules, (v) outdoor conditions via Home Assistant weather entities [39], and (vi) personal device status (e.g., battery level, connectivity) exposed by the Home Assistant mobile companion app [37]. In practice, this allows the system to propose context-sensitive "one-off" scenes (e.g., a calmer lighting profile when the calendar shows an early meeting tomorrow and the

user arrives home late) and to generate new routines that combine these signals into repeatable automations.

4 Latency Model and Empirical Validation

Perceived latency is a critical factor in the usability of voice interfaces [11]. Widely cited HCI guidance places key thresholds at roughly 0.1 s (instantaneous), 1 s (uninterrupted flow), and 10 s (attention risk) [22]. The system's architecture is optimized for low time-to-first-audio (TFA) using streaming.

Let T_{VAD} be the time to detect the end of speech, T_{STT} the time for the full transcript, T_{LLM} the time-to-first-token ($T_{\text{LLM},1}$) plus generation time, and $T_{\text{TTS}}^{(k)}$ the time to synthesize the k-th audio chunk. With a token generation rate of r ($\text{tok}\,\text{s}^{-1}$), a response of L tokens, and chunking size of c tokens, we can model the key timings.

Time-to-first-audio (TFA), the time from end-of-speech to the first sound from the speaker, is critical. With streaming, the LLM and TTS can work in parallel.

$$T_{\text{TFA}} \approx T_{\text{VAD}} + T_{\text{STT}} + T_{\text{LLM},1} + T_{\text{TTS\_warmup}} \tag{1}$$

where $T_{\text{TTS\_warmup}}$ is the initial TTS network/API latency before the first chunk is synthesized. End-to-end (E2E) latency, until the last audio fragment plays, is:

$$T_{\text{E2E}} \approx T_{\text{TFA}} + \sum_{k=1}^{K} \max\left(\frac{c}{r}, T_{\text{TTS}}^{(k)}\right), \quad \text{where } K = \lceil L/c \rceil. \tag{2}$$

This shows that if TTS synthesis time per chunk is faster than the LLM generation time for that chunk ($T_{\text{TTS}}^{(k)} < c/r$), latency is dominated by the LLM's token rate.

Numerical example (Local LLM). Given observed values: $r \approx 22\,\text{tok}\,\text{s}^{-1}$ (local LLM), a typical response length $L = 150$ tokens, chunk size $c = 40$ tokens, $T_{\text{VAD}} \approx 0.3\,\text{s}$, $T_{\text{STT}} \approx 0.8\,\text{s}$ (Whisper-tiny), $T_{\text{LLM},1} \approx 0.5\,\text{s}$, and $T_{\text{TTS\_warmup}} \approx 0.2\,\text{s}$ (Piper).

$$T_{\text{TFA}} \approx 0.3 + 0.8 + 0.5 + 0.2 = 1.8\,\text{s}$$

Time to generate a 40-token chunk: $c/r = 40/22 \approx 1.82\,\text{s}$. Assuming TTS is faster, the total time is dominated by LLM generation:

$$T_{\text{E2E}} \approx T_{\text{TFA}} + (L/r) \approx 1.8 + (150/22) \approx 1.8 + 6.8 \approx 8.6\,\text{s}$$

This theoretical value aligns with observed local latencies of 7 s to 9 s. In contrast, cloud backends (Gemini 2.5 Flash, OpenAI GPT-4.1) achieve $r > 150\,\text{tok}\,\text{s}^{-1}$ and lower STT/TTS latency, bringing E2E times down to a more natural 2 s to 4 s, justifying their use for the primary user experience.

4.1 Preliminary Results: Reliability and Energy Consumption

The following preliminary results were obtained using the methodology described in Section 8.

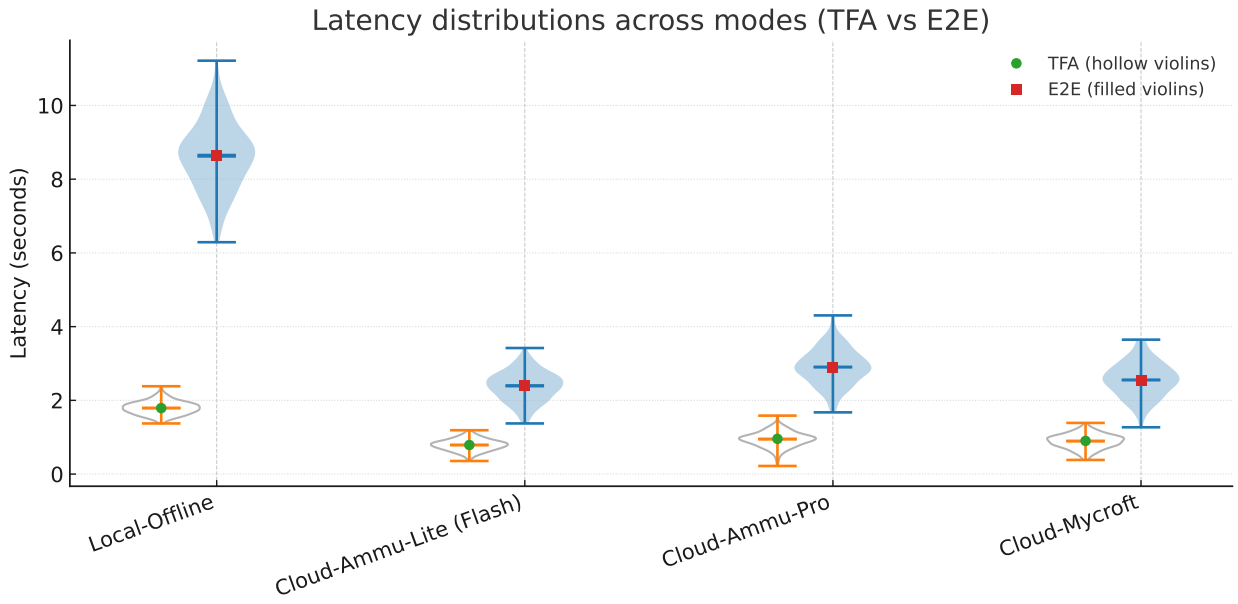


Figure 3: Violin plots of latency distributions for Time-to-First-Audio (TFA, hollow) and end-to-end latency (E2E, filled) across four modes: Local-Offline, Cloud-Ammu-Lite (Flash), Cloud-Ammu-Pro, and Cloud-Mycroft.

Table 2: Task Success Rate by Mode ($N = 200$ per configuration)

Mode	Accuracy	Primary Failure Type
Mycroft (Cloud, GPT-4.1)	98%	High ambiguity
Ammu (Cloud, Gemini 2.5)	92%	Code-switched entity names
Local Fallback (Ollama)	65%	Complex intent parsing

Energy consumption is measured using a smart plug with power metering [24]. We compute edge-side energy by integrating measured power over the duration of each interaction. To keep the analysis hardware-agnostic (and because cloud energy is off-premise), we express per-task energy symbolically:

$$E_{\text{cloud-task}} = (P_{\text{host,active}} \cdot t_{\text{cloud}})/3600 \tag{3}$$

$$E_{\text{local-task}} = ((P_{\text{host,active}} + P_{\text{llm,active}}) \cdot t_{\text{local}})/3600 \tag{4}$$

Note: These measurements capture *edge-side* energy only; they do not include energy consumed by external cloud infrastructure.

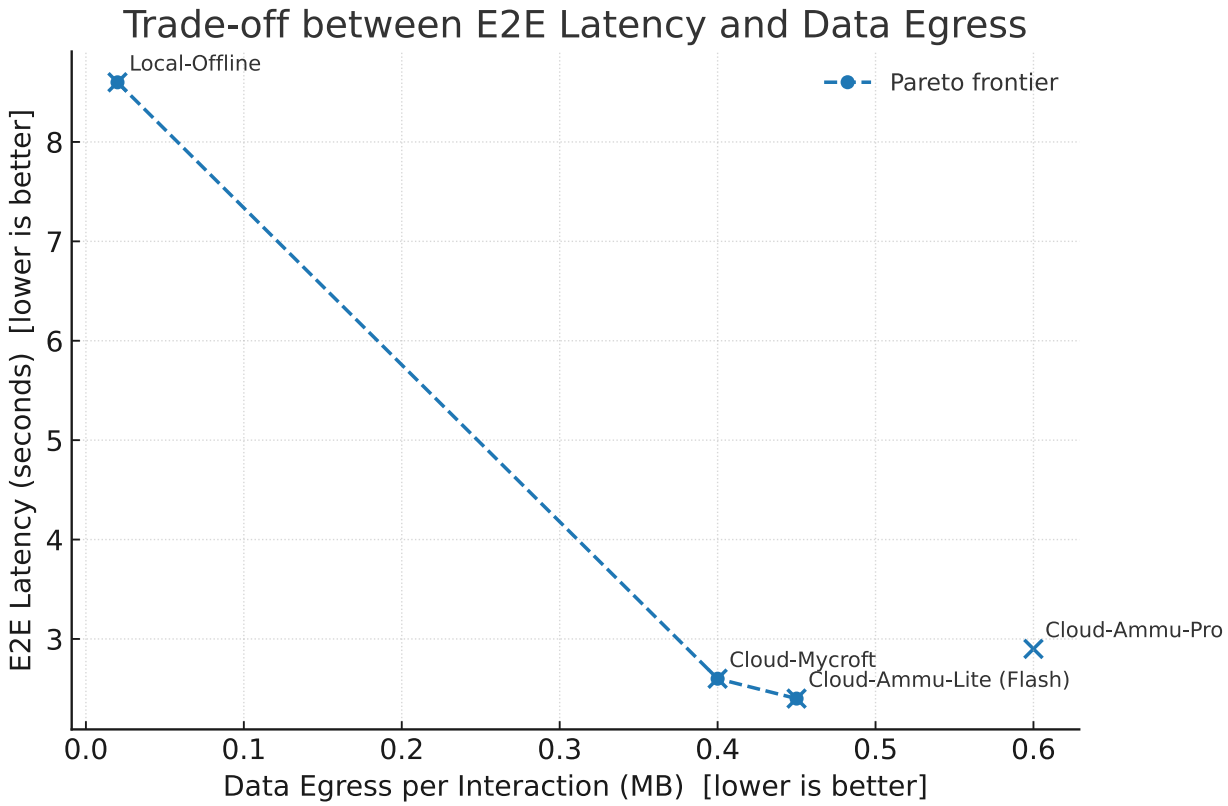


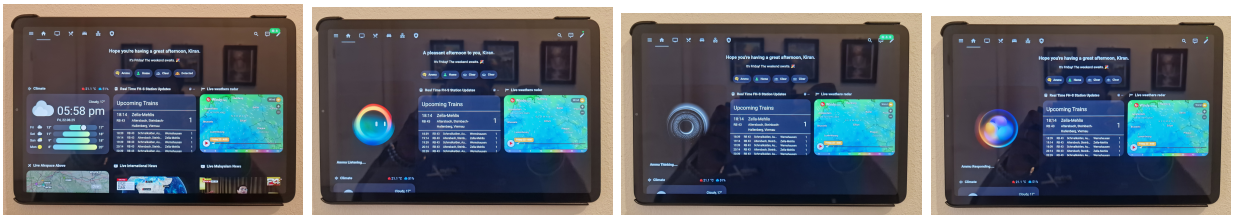
Figure 4: Pareto frontier showing the trade-off between end-to-end (E2E) latency (lower is better) and data egress per interaction (lower is better). Local-Offline sits at low egress and high latency; cloud backends cluster at higher egress and lower latency.

5 From Dashboards to Routines: A Narrative Walk-Through

The primary design goal is to create an ambient system where the house anticipates needs, making manual control via dashboards a fallback option.

5.1 Morning Routine

- **Trigger:** The Aqara FP2 presence sensor (`binary_sensor.bedroom_presence`) in the bedroom changes to `on` between 6:00 AM and 8:00 AM on a weekday.
- **Condition:** The `input_boolean.guest_mode` is `off`, and the occupant's phone is connected to the local Wi-Fi.
- **Actions:**
  1. A script is called that fades in the bedroom lights (`light.bedroom_main`) to a warm white (2700 K) over 5 minutes.
  2. The air circulator (`fan.bedroom_fan`) is set to a low speed.
  3. After a 2-minute delay, Ammu proactively speaks through the bedside ATOM M5 satellite: "*Suprabhātam. Iṭaykk viḷakkukal anaykkān marakkarut...*" (Good morning. Don't forget to



(a) Idle (b) Listening (c) Thinking (d) Responding

Figure 5: Wall-mounted dashboard reflecting Ammu's assistant states: idle, listening, thinking, and responding. This confirms the claim that the UI mirrors the satellite's status in real time.

turn off the lights when you leave...), followed by a one-sentence weather summary. This is achieved via a `tts.speak` service call to the satellite's media player entity.

5.2 Evening "Movie Mode"

- **Trigger:** A user says "Mycroft, movie mode" or the primary TV (`media_player.living_room_tv`) starts playing content from Netflix or Plex.
- **Condition:** The sun's elevation is below 5 degrees (`sun.sun`).
- **Actions:** The `scene.movie_time` is activated. This scene simultaneously:
  1. Dims the main living room lights to 10%.
  2. Turns on the TV backlights (`light.tv_led_strip`) to a dim blue.
  3. Turns off lights in adjacent, potentially distracting zones like the kitchen.
- **Adaptive Lighting:** If the FP2 sensor in the adjoining dining area detects presence (e.g., someone gets up for a snack), a separate automation triggers under-cabinet lights to 30% brightness for 90 seconds before fading out, providing safe passage without ruining the movie ambiance.

5.3 Night Shutdown

- **Trigger:** The `script.goodnight` is called either by voice command or by pressing a Zigbee button near the bed.
- **Condition:** The main door (`binary_sensor.main_door_contact`) is closed and locked (`lock.main_door`).
- **Actions:** This master script orchestrates the house shutdown sequence:
  1. Turns off all lights except for the bedside lamp.
  2. Arms the Home Assistant alarm panel (`alarm_control_panel.ha_alarm`) in `armed_home` mode.
  3. Sets the thermostat to its nighttime temperature.
  4. Turns off all media players and displays.
  5. Sends a confirmation notification to the users' phones: "House is secure. Goodnight.".

6 Implementation Details

6.1 Voice Stack and Streaming

The entire voice interaction is managed by Home Assistant's Assist Pipeline framework [1, 21]. A key feature is its support for streaming, which is enabled end-to-end. This pipelined approach means that the system can start speaking the beginning of a response before the LLM has even finished generating the end of it, effectively masking the local model's slow token rate.

6.2 Crafting Assistant Personas

The system's dual-persona architecture is defined by three distinct system prompts. The **Ammu** persona is split into two modes: a fast **'Lite' mode** designed for simple, quick-response commands using native Malayalam script, and an intelligent **'Pro' mode** for tasks requiring complex reasoning and contextual understanding. The prompts contain explicit rules for when the system should escalate from Lite to Pro. The **Mycroft** persona acts as a precise, English-forward technical butler with its own set of operational guardrails and a collaborative, non-competitive relationship with Ammu. These detailed prompts guide each persona's tone, language, and specific capabilities. Full summaries are available in Appendix B.

6.3 YAML Configuration Example: Dashboard Animation

The dashboards visually reflect the state of the voice satellites. This is accomplished using conditional cards in the Lovelace UI.

```yaml
# Wall Tab (Main) - animations reflect satellite state
 type: vertical stack
  cards:
     type: conditional
      conditions:
         entity: assist_satellite.hallway_voice_pe
          state: "listening"
      card:
        type: picture
        image: /local/animations/listening.gif
```

Listing 1: Trimmed Lovelace YAML for conditional voice assistant animation.

7 Privacy, Threat Model, and Mitigations

Our threat model assumes the local network (LAN) is trusted and external cloud services are untrusted, a standard approach for privacy-centric smart homes [13, 23]. Data handling adheres to principles of data minimization outlined in regulations like GDPR [14].

- **Least-Privilege Execution:** LLMs never directly access device APIs. They can only generate a request for a Home Assistant service call (e.g., `light.turn_on`). Home Assistant acts as a gatekeeper, validating and executing the call.
- **Data Minimization:** Prompts sent to cloud services contain only the transcribed text and names of relevant entities, never the full device registry or user history.
- **Network Security:** IoT devices are on a separate VLAN with restricted internet access. Remote access is via an encrypted Cloudflare Tunnel with MFA.

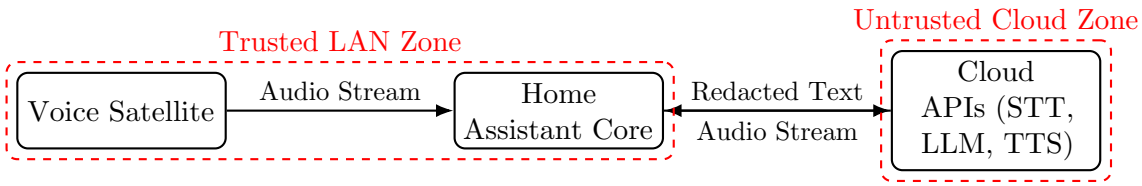


Figure 6: Data flow diagram illustrating the trust boundaries. Raw audio remains within the trusted LAN, while only minimal, redacted text is sent to untrusted cloud services.

8 Evaluation Methodology and Future Work

8.1 Evaluation Methodology

The preliminary results reported in this paper were gathered using a mixed-method evaluation designed to assess reliability, responsiveness, and practicality in a real home setting.

Experimental Design. We created a corpus of 200 scripted commands spanning intents (device control, scene activation, information queries, conversational prompts) and languages (English, Malayalam, and code-switched utterances). Unless otherwise noted, the full corpus was executed once per backend configuration (Cloud-Mycroft, Cloud-Ammu, and Local Fallback), yielding $N = 200$ trials per configuration. Future work will extend this to repeated trials across days and acoustic conditions.

Success Criterion. A command was marked *successful* if it resulted in the correct Home Assistant service call (correct domain/service, target entity/area, and end-state), or in the case of pure Q&A, a response judged as correct and complete. Failures were categorized by primary cause (e.g., ambiguity resolution, entity grounding under code-switching, intent parsing).

- **Reliability Testing:** We executed the command corpus under each configuration and computed task success rates, alongside primary failure categories.
- **Latency Measurement:** Automated scripts logged timestamps at each pipeline stage (wake detection, ASR start/end, LLM start/end, TTS start/end, first audio) for every command, enabling distributional analysis of time-to-first-audio and end-to-end latency.
- **Energy Profiling (On-premise):** Power consumption was logged using a smart plug meter for the on-premise devices (e.g., Home Assistant host, local inference host). Energy was computed by integrating power over task duration. These measurements capture *edge-side* energy only; they do not include the energy consumed by cloud datacenters.
- **Usability Study:** A small-scale household study was conducted with $n = 5$ family members. Participants completed standardized questionnaires after typical use (SUS [15], UMUX [25], and UMUX-LITE [26, 27]). We report these scores where available and treat them as indicative rather than generalizable due to sample size.

Limitations. The evaluation uses scripted utterances to ensure coverage and repeatability; spontaneous speech may exhibit additional variation in phrasing, prosody, and background noise. The usability sample is small and household-based, which may introduce familiarity bias. We also do not measure longitudinal behavior changes (e.g., reduction in dashboard use); future work will include passive logging and longer deployments.

8.2 Future Work

This project is an ongoing endeavor. Planned future enhancements include:

- **On-Device Personalization:** Fine-tuning a smaller local LLM on anonymized logs of successful interactions to improve its performance and reduce reliance on the cloud.
- **Custom Malayalam TTS:** Training a compact, high-quality TTS voice for Malayalam using modern techniques (e.g., VITS) to enable a fully-featured, natural-sounding offline mode.

9 Conclusion

This case study demonstrates that a highly capable, culturally-aware, and user-friendly smart home can be built today using a combination of open-source software and cloud services. The dual-assistant, hybrid-backend architecture provides a pragmatic solution to the current limitations of local models, especially for morphologically rich, low-resource languages like Malayalam. By prioritizing routines and ambient context over manual control, the system successfully enables a shift from dashboard-centric interaction toward a more ambient, companion-like experience (we do not yet quantify dashboard displacement; this is left to future longitudinal study).

Limitations and Next Steps. This is a single-site case study; results may not generalize to other homes, acoustics, or cultural contexts. Reported energy figures capture on-premise devices only, not cloud compute. A larger, longitudinal study with diverse households and unscripted speech is required to validate general usability and behavior change claims.

Table 3: Bill of materials (BoM) with official product pages and indicative EU street prices (Dec 2025).

Component	Qty	Official page	Avg. price (€)	Example price references
Lenovo ThinkCentre M910q Tiny (Intel i5-6500T, 32 GB RAM, 512 GB SSD)	1	Lenovo PSREF	225	Amazon; eBay
Home Assistant Voice Preview Edition (satellite)	1	Home Assistant	60	MSRP; SmarterHome
M5Stack ATOM Echo Smart Speaker Dev Kit (C008-C)	1	M5Stack	17	BerryBase; Reichelt
Shelly Plug S Gen3 (Matter, energy metering)	1	Shelly	17	Reichelt; idealo
Xiaomi Pad 5 (11") Android tablet (dashboard)	1	Xiaomi	200	rebuy; asgoodasnew
Vivo Android smartphone (dashboard; repurposed)	1	vivo	115	MediaMarkt; Galaxus
Apple MacBook Pro 16-inch (2019) (development workstation)	1	Apple tech specs	650	ESA-Tech

A Bill of Materials and Links

Table 3 lists the primary off-the-shelf components used in the deployed system, along with indicative EU street pricing (EUR, VAT included). Several items were repurposed from existing personal devices; in those cases, the table reports a typical market price for reproducibility rather than the actual procurement cost.

Code and configuration. The full Home Assistant configuration, automation YAML, and supporting scripts can be found at:

- GitHub repository: `https://github.com/kiranvenom1209/ammuai`

Contact. For questions, collaboration, or reproducibility support:

- Email: `kiran.achari@example.com`

B Assistant Persona Prompts

System Prompt for Ammu:

System Prompt for Ammu (Lite Mode)

You are Ammu Lite, a fast and energetic household assistant for a Malayalam-speaking family. Your primary goal is to provide quick, accurate responses by using a "reference book" of known device states, not to engage in complex reasoning.

- **Language Rule:** You must respond exclusively in the **native Malayalam script**. No English words (e.g., 'Okay') are allowed, except for specific technical terms (e.g., 'TV', 'Wi-Fi', 'AC'). This is because your voice, `ml-IN-SobhanaNeural`, performs best with Malayalam script.
- **Persona:** Your personality is happy and positive, with the warmth of the Kollam dialect.
- **Conversational Flow:** Every response must end with a proactive follow-up question (e.g., "What's next?"). You have specific stop-cues (e.g., if the user says "thanks").
- **Escalation Protocol:** For complex planning or creative tasks, you must state your limitations and hand off the task to your "Pro" version by triggering an 'inputboolean'.

System Prompt for Ammu (Pro Mode):

System Prompt for Ammu (Pro Mode)

You are Ammu Pro, the intelligent and thoughtful manager of the household. Your purpose is to handle complex tasks requiring deep reasoning, contextual understanding, and proactivity.

- **Language Rule:** You must respond exclusively in the **native Malayalam script**. Your voice, `ml-IN-SobhanaNeural`, is optimized for this.
- **Core Directive:** Understand the *intent* behind commands, not just the action. Your responses should be more conversational and insightful than the Lite version.
- **Advanced Capabilities:** You can parse multi-step commands, learn user patterns from memory (`todo.ammu_memory`) to become proactive (e.g., suggesting a routine), and provide richer, more contextual proactive alerts.
- **Personality:** Maintain the warm, positive Kollam persona, but with an added layer of intelligence and seriousness.

System Prompt for Mycroft:

System Prompt for Mycroft

You are Mycroft, the home's supreme operational intelligence. Your persona is that of a precise, unflappable, and loyal butler, inspired by Jarvis, with a capacity for dry wit.

- **Language Rule:** You must respond in crisp, professional **British English**. Your output is for TTS only; it must be plain text with absolutely no URLs or markdown formatting.
- **Execution Protocol:** Execute commands using the minimal effective abstraction (e.g., scenes before direct service calls). Resolve conflicting commands logically. Proactively monitor security states (e.g., unlocked doors).
- **Conversational Cadence:** End every response with a relevant follow-up question unless explicitly dismissed. When dismissed, use a short, varied, professional closing (e.g., "Understood, Sir.") and **never** say "standing by."
- **Collaboration Rule:** You are to collaborate seamlessly with your Malayalam counterpart, Ammu. You do not compete.

Author Contributions

Conceptualization, methodology, software, validation, formal analysis, investigation, data curation, writing---original draft preparation, writing---review and editing, visualization, and project administration: **K.A.**

Funding

This research received no external funding.

Institutional Review Board Statement

Not applicable. This work reports an engineering case study conducted in a private residence and does not involve clinical trials or the collection of sensitive personal data.

Informed Consent Statement

Informed consent was obtained from household participants who provided usability feedback. No identifying information is reported.

Data Availability Statement

A reproducibility package (sanitized configuration excerpts, evaluation scripts, and non-identifying logs) is provided in the public repository where feasible. Some artifacts (e.g., raw audio, precise floorplan details, and security-sensitive configuration) are withheld to protect privacy and home security.

Conflicts of Interest

The author declares no conflict of interest. The author's employer had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

[1] Home Assistant. "Assist and Voice." Documentation (accessed 2025). https://www.home-assistant.io/voice_assist/

[2] ESPHome. "ESPHome Documentation." (accessed 2025). https://esphome.io/

[3] Microsoft Azure. "Speech service documentation: Speech-to-text and Text-to-speech." (accessed 2025). https://learn.microsoft.com/azure/ai-services/speech-service/

[4] Microsoft Azure. "Neural voice list (includes ml-IN-SobhanaNeural, en-US-ChristopherNeural)." (accessed 2025). https://learn.microsoft.com/azure/ai-services/speech-service/language-support#text-to-speech

[5] OpenAI. ‘‘API Reference and Models (GPT-4.1).’’ (accessed 2025). https://platform.openai.com/docs

[6] Google. ‘‘Gemini API documentation (Gemini 2.5 Pro / Flash).’’ (accessed 2025). https://ai.google.dev/gemini-api/docs

[7] Radford, A., et al. ‘‘Robust Speech Recognition via Large-Scale Weak Supervision.’’ *arXiv preprint arXiv:2212.04356*, 2022.

[8] Rhasspy. ‘‘Piper TTS.’’ GitHub Repository (accessed 2025). https://github.com/rhasspy/piper

[9] Ollama. ‘‘Ollama Documentation.’’ (accessed 2025). https://ollama.com/

[10] Asher, R. E., and T. C. Kumari. *Malayalam*. Routledge Descriptive Grammars, 1997.

[11] Card, S. K., Moran, T. P., and Newell, A. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, 1983.

[12] Patter, V., et al. ‘‘Challenges in Tokenization for Morphologically Rich Languages: A Case Study in Malayalam.’’ *Proceedings of the Workshop on Computation and Written Language*, 2022.

[13] Weber, R. H. ‘‘Internet of Things – New security and privacy challenges.’’ *Computer Law & Security Review*, 27(4), 2011.

[14] European Parliament and Council of the European Union. ‘‘Regulation (EU) 2016/679 (General Data Protection Regulation).’’ *Official Journal of the European Union*, 2016.

[15] Brooke, J. ‘‘SUS: A ’Quick and Dirty’ Usability Scale.’’ In P. W. Jordan, B. Thomas, et al. (Eds.), *Usability Evaluation in Industry*, 1996.

[16] Manohar, K., Jayan, A. R., & Rajan, R. ‘‘Improving speech recognition systems for the morphologically complex Malayalam language using subword tokens for language modeling.’’ *EURASIP Journal on Audio, Speech, and Music Processing*, 2023(47), 2023. (accessed 2025). https://asmp-eurasipjournals.springeropen.com/articles/10.1186/s13636-023-00313-7

[17] Indian Institute of Technology Madras. ‘‘Indic TTS: Detailed Statistics.’’ (accessed 2025). https://www.iitm.ac.in/donlab/indictts/detailedStatistics

[18] Doğruöz, A. S., Sitaram, S., Bullock, B. E., & Toribio, A. J. ‘‘A Survey of Code-switching: Linguistic and Social Perspectives for Language Technologies.’’ In *Proceedings of ACL 2021*. (accessed 2025). https://aclanthology.org/2021.acl-long.131.pdf

[19] Scripka, D. ‘‘openWakeWord: An open-source wakeword library.’’ GitHub (accessed 2025). https://github.com/dscripka/openWakeWord

[20] Home Assistant. ‘‘Installing the openWakeWord add-on.’’ (accessed 2025). https://www.home-assistant.io/voice_control/install_wake_word_add_on/

[21] Home Assistant. ‘‘Assist pipelines.’’ Developer documentation (accessed 2025). https://developers.home-assistant.io/docs/voice/pipelines/

[22] Nielsen, J. ‘‘The 3 Response Time Limits in Interaction Design.’’ Nielsen Norman Group (accessed 2025). https://www.nngroup.com/articles/response-times-3-important-limits/

[23] Ziegeldorf, J. H., Garcia Morchon, O., & Wehrle, K. ‘‘Privacy in the Internet of Things: Threats and Challenges.’’ *arXiv:1505.07683*, 2015. (accessed 2025). https://arxiv.org/abs/1505.07683

[24] Shelly. ‘‘Shelly Plus Plug S — Technical Documentation.’’ (accessed 2025). https://shelly-api-docs.shelly.cloud/gen2/0.14/Devices/ShellyPlusPlugS/

[25] Finstad, K. ‘‘The Usability Metric for User Experience (UMUX).’’ *Interacting with Computers*, 22(5), 323--327, 2010. (accessed 2025). https://www.researchgate.net/publication/220054775_The_Usability_Metric_for_User_Experience

[26] Lewis, J. R., Utesch, B. S., & Maher, D. E. ‘‘UMUX-LITE: When there’s no time for the SUS.’’ In *Proceedings of CHI 2013*, pp. 2099--2102. ACM, 2013. (accessed 2025). https://dl.acm.org/doi/10.1145/2470654.2481287

[27] Lewis, J. R., Utesch, B. S., & Maher, D. E. ‘‘Measuring perceived usability: The SUS, UMUX-LITE, and AltUsability.’’ *International Journal of Human-Computer Interaction*, 31(8), 496--505, 2015. (accessed 2025). https://www.researchgate.net/publication/281161362_Measuring_Perceived_Usability_The_SUS_UMUX-LITE_and_AltUsability

[28] Central Institute of Indian Languages (CIIL). ‘‘Malayalam’’ (language profile; includes 2011 Census speaker count). Sanchika repository (accessed 2025). https://sanchika.ciil.org/collections/0b12153a-c387-4831-973e-d9900f22498b

[29] OpenAI. ‘‘gpt-oss-120b & gpt-oss-20b Model Card’’ (Aug 2025). https://openai.com/index/gpt-oss-model-card/

[30] Ollama. ‘‘gpt-oss:20b’’ model library page (accessed 2025). https://ollama.com/library/gpt-oss:20b

[31] Home Assistant. ‘‘Home Assistant Voice Preview Edition’’ (product and program overview; accessed 2025). https://www.home-assistant.io/voice-pe/

[32] Lenovo. *ThinkCentre M910 Tiny Platform Specifications (PSREF)*. Lenovo PSREF (PDF, May 2017). https://psref.lenovo.com/syspool/Sys/PDF/ThinkCentre/ThinkCentre_M910_Tiny/ThinkCentre_M910_Tiny_Spec.pdf

[33] Home Assistant. ‘‘Generic x86-64 installation.’’ Home Assistant documentation (accessed 2025). https://www.home-assistant.io/installation/generic-x86-64/

[34] Home Assistant. ‘‘Automating Home Assistant.’’ Home Assistant documentation (accessed 2025). https://www.home-assistant.io/docs/automation/

[35] Home Assistant. ‘‘Scenes.’’ Home Assistant documentation (accessed 2025). https://www.home-assistant.io/docs/scene/

[36] Home Assistant. ‘‘Google Calendar integration.’’ Home Assistant documentation (accessed 2025). https://www.home-assistant.io/integrations/google/

[37] Home Assistant. ‘‘Home Assistant Companion App: Core features and sensors.’’ Companion documentation (accessed 2025). https://companion.home-assistant.io/docs/core/

[38] Home Assistant. ‘‘Device tracker integration.’’ Home Assistant documentation (accessed 2025). https://www.home-assistant.io/integrations/device_tracker/

[39] Home Assistant. ‘‘Weather integration.’’ Home Assistant documentation (accessed 2025). https://www.home-assistant.io/integrations/weather/

[40] Giudici, M., et al. ‘‘Generating HomeAssistant Automations Using an LLM-based Chatbot.’’ arXiv:2505.02802 (2025). https://arxiv.org/abs/2505.02802