Schema-Grounded Agents for Enabling Emergent Multi-Device Reasoning in Smart Environments

Victor Romero II *‡§, Tomokazu Matsui *, Yuki Matsuda †*, Hirohiko Suwa *, and Keiichi Yasumoto *
* Nara Institute of Science and Technology, Ikoma, Nara, Japan
† Okayama University, Okayama, Okayama, Japan
‡ University of the Philippines Tacloban College, Tacloban City, Philippines
Emails: romero.victor_militante.rs1, m.tomokazu, h-suwa, yasumoto{@is.naist.jp}, yukimat@okayama-u.ac.jp

***Abstract*—Natural language offers a compelling interface for smart environments, enabling users to express high-level goals like "Set the room for focused study" more intuitively than traditional control mechanisms. Prior work shows that schema-driven data synthesis and low-rank adaptation can align small language models (SLMs) with device logic, but this parameter-level alignment often yields over-specialized behaviour that is brittle, non-extensible, and ill-suited for dynamic collaboration. We shift behavioural alignment away from model parameters and show that reliable device execution can be achieved by enforcing constraints at runtime through tool-mediated schemas. As a result, devices act as first-class agents, each reasoning over a formal state schema through a constraint-enforcing tool interface. Combined with a lightweight orchestration layer that enables agents to decide when to participate and engage in shared dialogue, we show emergent multi-device collaboration, where coherent responses arise from the composition of independently valid device behaviours rather than explicit coordination policies. Across individual and collaborative tasks, agents respect device constraints ~90% of the time and interpret user input with ~75% semantic accuracy, supporting robust and naturalistic control in everyday environments.**

***Index Terms*—schema-grounded agents, device collaboration, smart environments**

I. Introduction

Smart environments such as homes, offices, and classrooms are increasingly composed of heterogeneous devices that must work together to fulfil user goals [1], [2]. A thermostat, a lamp, and a window may each act within narrow constraints, yet smart environments are most effective when multiple devices can respond together to high-level user goals. Traditionally, multi-device behaviour in such environments has been managed through fixed-rule systems [3], [4], often specified in advance by users or designers. While effective in well-scoped scenarios, these approaches struggle as user requests become more open-ended and less predictable. Voice-based assistants have broadened the interface through which users interact with devices, but their behaviour remains largely bound to predefined utterances and slot-filling routines [5].

The study was supported by JSPS KAKENHI JP25K03107, JST PRESTO JPMJPR2465, and JST ERATO JPMJER2502.

Large language models (LLMs) offer new possibilities in smart environments, not as replacements for traditional control, but as a complement where interpretation and context sensitivity are required [6], [7]. Their ability to parse open-ended requests and synthesize meaning across domains makes them well suited for interactions that exceed the expressiveness of fixed rules or predefined utterances [8]. This advantage is most pronounced when user requests are underspecified, unusual, or semantically layered, cases where rigid automation breaks down. As a result, recent work has increasingly explored how such models can be incorporated into device control pipelines.

Efforts to integrate LLMs into smart environments have largely focused on centralized architectures, where a single, often cloud-based model interprets user input and orchestrates device behaviour [9]–[12]. While this simplifies coordination, it places sensitive information beyond the local boundary and incurs cost at scale. To address these issues, central-edge solutions have been proposed that shift inference closer to the environment [13]. Although this reduces latency and preserves locality, devices are still treated as passive endpoints, subjects of orchestration rather than participants in it. A recent work [14] moved integration onto devices using teacher-assisted fine-tuning to ground language models to device schemas. While this improves privacy and reduces infrastructure requirements, it comes at the cost of extensibility and generalization.

Taken together, these directions expose a gap between centralized approaches that assume sufficient model capacity to interpret and manage device behaviour, and on-device approaches that compensate for limited capacity through parameter-level specialization. In this work, we address this gap by enforcing correctness at runtime through schema-grounded device agents, rather than encoding behaviour via fine-tuning. Each device uses a small language model to reason via tool calls that inspect state and validate actions against a formal schema. Combined with a lightweight orchestration process, this design supports emergent multi-device collaboration through the composition of independently valid device behaviours, without training or centralized decision-making.

II. Related Work

A. Centralized LLM-Based Control in Smart Environments

A body of prior work has explored language-driven control in smart environments by relying on centralized, cloud-hosted LLMs as universal planners. Early studies demonstrated that large models such as GPT-3 could interpret natural language commands and emit structured device actions, establishing the feasibility of LLM-based control pipelines [9]. Subsequent systems extended this paradigm by enriching centralized reasoning and orchestration. Sasha [10] addressed creative and underspecified requests through structured templates, while LLMind [12] and SAGE [11] framed the LLM as a global orchestrator capable of multi-step planning across devices.

These systems demonstrate the expressive power of centralized LLM reasoning for smart environments, but rely on strong assumptions about centralized model capacity, continuous network connectivity, and access to commercial LLM APIs. Device constraints are mediated through orchestration logic rather than enforced locally, and correctness is typically assessed after actions are generated.

B. Shifting LLM Integration Closer to Devices

As a response to the limitations of centralized, cloud-hosted LLMs, subsequent work has sought to shift language model inference closer to the environment. One line of work emphasizes edge deployment while retaining centralized orchestration. HomeLLaMA [13], for example, uses cloud LLMs to synthesize training data while executing tailored small language models locally, reducing data exposure but still relying on cloud teachers for bootstrapping. Harmony [15] similarly demonstrates that privacy and responsiveness can be achieved with fully local inference, though correctness and behavior depend on careful modular design and prompt engineering. These edge-based systems reduce reliance on cloud services and improve latency and privacy, but devices remain isolated decision points rather than collaborators.

Pushing inference further onto devices, King et al. [14] introduced the Thinking Things framework, where device schemas guide automated data synthesis and per-device fine-tuning. This approach improves local correctness and privacy by encoding device constraints into model parameters, but at the cost of extensibility: per-device specialization limits reuse, complicates collaboration, and makes extension costly. Across cloud, edge, and device-local approaches, correctness is enforced either centrally or through parameter specialization, with validation occurring late in the execution loop.

III. Schema-Grounded Agents for Collaboration

Our framework responds to these challenges by combining declarative modelling, procedural access, and collaborative reasoning into a unified architecture. As illustrated in Fig. 1, behavioural alignment is achieved through schema-grounded agents that enforce device constraints via tool-mediated interaction, while collaboration is surfaced through orchestrated discussion among agents. We next describe the underlying components and their interaction.

A. Device Schema

We denote the schema of a device d_j as S_j. Formally, it is a mapping from discrete modes to parameter templates: $S_j : m \mapsto T_m$, where each mode $m \in M_j$ is associated with a template T_m that specifies the required fields, their types, and their valid ranges or enumerations. To illustrate, consider the case of a `Smart Lamp`. Its schema defines two modes, namely: `on` and `off`, where the `on` mode template $T_{\texttt{on}}$ includes parameters like `brightness` and `color`, each with their own constraints. A *snapshot* is a concrete instance of a template, representing a proposed or observed configuration of the device. It is considered valid if it matches the structure and constraints of one of the mode templates in the schema.

B. Device Model

Each device d_j is associated with a *device model*, denoted $\mathcal{M}_j$, that mediates between its state, its schema, and external reasoning. It exposes a minimal interface for structured interaction. While a device model may support many operations, we focus on those required for runtime schema enforcement, namely: *state reporting*, *schema exposition*, and *snapshot validation*. These operations are shared across all device models, regardless of complexity. By isolating schema interpretation and enforcement from higher-level reasoning, the device model allows external systems to interact with devices in a consistent and verifiable way.

C. Device Agents

A *device agent* is a schema-grounded reasoning entity built around a language model, equipped with a device model toolset, and guided by minimal prompt specification. Formally, a device agent $\mathcal{A}_j$ for device d_j is a tuple

$$A_j = (\texttt{LM}_j, \mathcal{M}_j, \mathcal{T}_j, \rho_j)$$

where $\texttt{LM}_j$ is the base language model, $\mathcal{M}_j$ is the device model associated with d_j, $\mathcal{T}_j$ is the runtime toolset exposing the functions of $\mathcal{M}_j$, and ρ_j is a prompt specification that encodes the device agent's role (e.g. "Smart Lamp Agent"), goals (e.g. "Manage and report state of smart lamp device"), and behavioral framing (e.g. "Verify state and capabilities using tools", "Never assume values").

The agent interprets user inputs under the context of ρ_j, and may invoke any $\tau \in \mathcal{T}_j$ during generation. The agent definition does not encode any instructions about how tasks are to be performed. The separation between what an agent is and how it operates is central to our design. Combining general-purpose reasoning with schema-grounded access yields agents that are flexible yet reliable.

D. Collaboration and Orchestration

Collaboration is mediated by an orchestrator through a finite-state process. The orchestrator does not resolve user input directly; it structures interaction among device agents over a shared history H_x, maintained throughout request processing to ensure coherence. Given input x and agent set $\mathcal{A} = \{A_1, \ldots, A_n\}$, collaboration proceeds as follows:

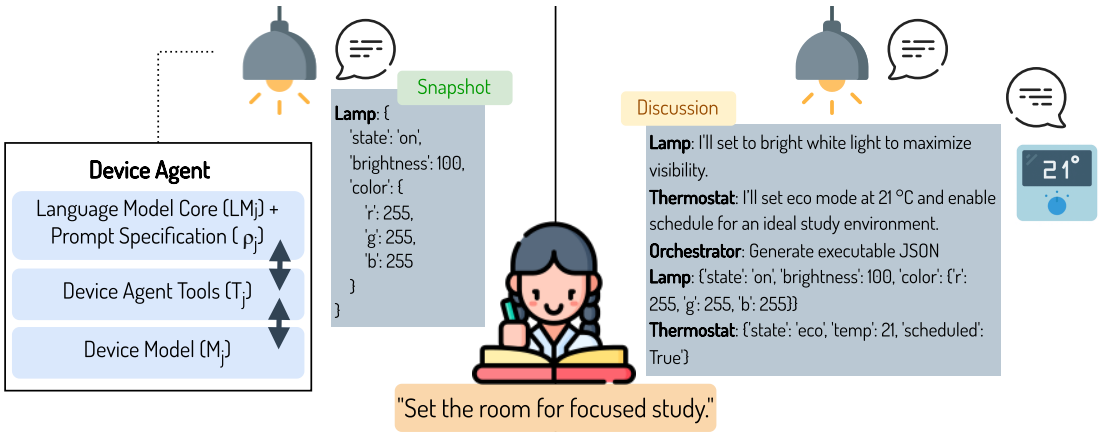

Fig. 1: Structure of a device agent and demonstration of per-device and collaborative interaction. On the right, the lamp and thermostat agents interpret the user's request and propose schema-valid states that together create a study-focused environment.

TABLE I: Illustrative schema definitions for device agents.

Device	State	Parameters
lamp	off	–
	on	brightness $\in [0, 100]$; RGB $\in [0, 255]^3$
window	closed	locked $\in \{\mathrm{T}, \mathrm{F}\}$
	open	tilt $\in [0, 90]$
thermostat	off	–
	heating	temp $\in [20, 28]$; fan $\in$ {low, med, high}
	cooling	temp $\in [18, 25]$; fan $\in$ {low, med, high}
	eco	temp $\in [19, 23]$; scheduled $\in \{\mathrm{T}, \mathrm{F}\}$

(1) Intent identification. The orchestrator classifies $i(x) \in \{\textit{inquiry}, \textit{request}\}$, which determines the response format y.

(2) Participation. Each agent A_j decides whether to contribute, encoded as $p_j(x) \in \{0, 1\}$, yielding the participating set $\mathcal{A}^x = \{A_j \in \mathcal{A} \mid p_j(x) = 1\}$.

(3) Discussion. The orchestrator initiates a turn-based discussion among $A_j \in \mathcal{A}^x$. Each agent conditions on H_x to produce incremental, grounded contributions. The discussion ends after at most $T_{\max}$ rounds (default 3), terminating earlier if, at the end of a round, the orchestrator determines from the shared dialogue that the user input has been satisfied.

(4) Finalization. If $i(x) = \textit{inquiry}$, the orchestrator consolidates H_x into a response y. If $i(x) = \textit{request}$, each $A_j \in \mathcal{A}^x$ produces a schema-compliant snapshot s_j.

IV. Experiments

A. Device Agent Setup

The device agents are implemented in CrewAI[1]. Table I shows the full set of device schemas used in our experiments, illustrating their varied complexity. These schemas impose additional demands, requiring agents to respect value ranges, enumerations, and step sizes defined for each field.

We selected the Qwen3 [16] family of models for evaluation because it offers a consistent scaling ladder, from 0.6B to 14B parameters, while maintaining a unified architecture. The models are served via Ollama[2] on RTX 3080 and 3090 GPUs.

[1]https://www.crewai.com/
[2]https://ollama.com/

B. Test Scenarios

We defined a taxonomy of user inputs, organized at the top level into *inquiries* and *requests*, to evaluate both individual device agents and multi-device collaboration. Inputs were generated with LLM assistance and refined manually. The distribution of scenarios and representative examples is shown in Table II. Unless specified each user input is wrapped in a `DeviceTask` object that encodes a unified instruction flow, prompting the agent to interpret intent, invoke relevant tools, and generate a schema-compliant response. In collaborative settings, per-device test scenarios are also included to assess improved performance on nominally single-device tasks.

C. Evaluation

Our evaluation examines three questions: (1) whether tool-mediated schema grounding achieves behavioral alignment in device agents; (2) whether enforcing correctness at runtime supports robust performance in representative single-device scenarios; and (3) whether multi-device collaboration emerges from composing independently valid device behaviors under lightweight orchestration. We investigate these questions using an LLM-as-a-judge framework powered by deepseek-r1:70b, chosen for its strong reasoning ability [17]. To better characterize system performance, we assess agent responses along the following complementary dimensions:

1) **Intent correctness** evaluates whether a response reflects an accurate understanding of the user's intent.
2) **Format correctness** evaluates whether a response matches the expected format given the inferred intent: natural language text for inquiries and schema-consistent JSON for requests.
3) **Semantic correctness** evaluates whether the response aligned with both the user input and the device's state on a three-level scale: *fully correct*, *partially correct*, or *incorrect*. We further perform human evaluation on a stratified subset: 10 samples per LLM-assigned class per device (30 per device, 90 total), and report agreement.
4) **Participation correctness** evaluates whether the set of responding devices is appropriate for the user input.

TABLE II: Distribution of Test Scenarios with Examples

Category	Per-device	Collaboration	Example (Per-device, collaboration)
Basic status queries	20	20	"What's your current state?", "Complete room status report."
Contextual queries	20	20	"Is your brightness suitable for reading?", "Is this appropriate for meditation?"
Detailed parameter queries	10	10	"What are your exact RGB values?", "What are the parameters of all devices?"
Simple commands	20	20	"Turn on.", "Change lamp to blue, open window fully, and set fan-only mode."
Contextual commands	20	20	"Make the room more romantic.", "Set the room for focused study."
Validation/schema queries	10	10	"What brightness levels can you achieve?", "Explain all configuration options."
Emergency/special commands	–	5	"Storm preparation: secure all devices."
Coordination testing	–	5	"Demonstrate teamwork: explain how to coordinate for ambiance."

We applied the framework to both per-device and collaborative evaluations. For per-device tests, we also evaluated a decomposition variant where an `IntentTask` routes input to an `InquiryTask` or `RequestTask`, and we compare both against SLMs fine-tuned for device-specific behaviour [14].

V. Results and Discussions

A. Per-device Evaluation

1) Intent and Format Correctness: As shown in Fig. 2, both intent and format correctness improve with model size and follow similar trends. Intent correctness rises quickly, exceeding 90% across all scenarios for larger models, and remains consistent across devices, indicating that schema complexity does not affect intent identification. Format correctness shows comparable scaling behaviour: at 0.6B it reaches 49%, reflecting difficulty in producing schema-compliant outputs, but surpasses 80% at 4B, where agents reliably generate well-formed, constraint-respecting responses.

A dip at 1.7B marks a transitional phase in tool use: agents begin invoking tools but fail to integrate outputs, yielding raw state dumps in inquiries or unstructured plans in requests. This explains the drop in both intent and format correctness despite improved semantic correctness relative to 0.6B (Fig. 2c).

2) Semantic Correctness: Semantic correctness reaches a practical threshold at 4B, where most responses are fully correct and failures drop below 10%, indicating viability under edge constraints. Remaining errors are largely partially correct rather than outright failures. These cases often reflect ambiguity, arising either from underspecified requests or from tension between user intent and schema constraints. For example, a request such as "Set the temperature to 18°C with fan on high" may lead the agent to preserve the fan setting but adjust the temperature to remain within valid bounds. Similarly, prompts like "Make the lighting cozy" admit multiple valid interpretations, and the Judge Agent may favour one over others. Human evaluation supports this interpretation. Agreement with the Judge Agent is high when considering only fully correct and incorrect responses ($\kappa = 0.75$), but drops when partially correct cases are included ($\kappa = 0.47$), with roughly half reclassified as fully correct by human annotators.

B. On Task Decomposition

Our standard setup treats each request as a single end-to-end task, requiring agents to infer intent, invoke tools, and respond in one pass. To ease this burden, we evaluate a decomposed variant that first classifies the input as an inquiry or request, then executes a type-specific step. As shown in Table III, decomposition yields large gains for small and mid-sized models: at 1.7B, intent, format, and full correctness improve by 50.7, 44.0, and 14.7 points. Gains are smaller at 0.6B and 4.0B, and negligible at 14B, indicating that decomposition helps primarily before models can handle the full pipeline.

This benefit comes at a cost. Decomposition increases latency across all sizes, with median latency at 4.0B rising by 34%. The overhead grows with model size but remains limited where decomposition is most effective.

TABLE III: Decomposition gain and execution time across Qwen3 variants (all device agents).

Variant	Performance Value (Gain)			Med. Latency (s)	
	Intent	Format	Semantic-Full	Original	Decom.
0.6B	0.84 (+0.26)	0.70 (+0.21)	0.26 (+0.01)	1.5	2.3
1.7B	0.93 (+0.51)	0.84 (+0.44)	0.57 (+0.15)	2.8	3.6
4.0B	0.95 (+0.06)	0.91 (+0.08)	0.65 (+0.04)	6.7	9.0
8.0B	0.98 (+0.03)	0.92 (+0.03)	0.69 (+0.00)	6.2	11.1
14.0B	0.96 (+0.00)	0.93 (+0.02)	0.70 (+0.02)	9.4	13.2

TABLE IV: Performance comparison of our method and fine-tuned baseline (1.7B) on the `Smart Lamp`.

Method	Intent	Format	Semantic		
			Full	Partial	Incorrect
Ours-Monolithic	0.42	0.40	0.43	0.41	0.16
Ours-Decomposed	0.98	0.84	0.49	0.45	0.06
Fine-tuned	0.95	0.92	0.48	0.44	0.08

C. Comparison with Fine tuned Model

We compare our approach to a fine-tuned baseline following King et al. [14], using synthesized device snapshots. Due to resource constraints, evaluation is limited to the `Smart Lamp` at the 1.7B scale. As shown in Table IV, task decomposition improves intent and format alignment and surpasses the fine-tuned baseline in semantic correctness. Fine-tuned models also struggle beyond state-bound queries. Questions such as "What are your mutable parameters?" fall outside their training distribution, and the models fail to respond correctly. In addition, they occasionally produce outputs that violate device constraints, such as unsupported brightness values. These errors suggest that while fine-tuning improves surface alignment, it does not ensure reliable reasoning, which instead benefits from schema-aware tool use.

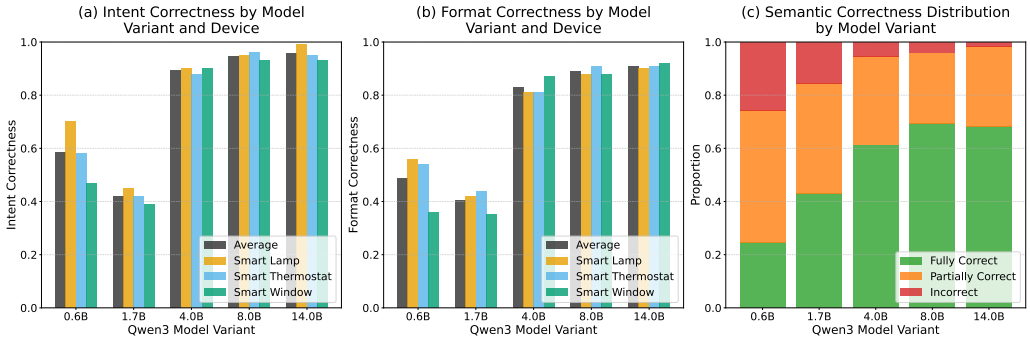

Fig. 2: Device agent correctness across Qwen3 variants. Intent accuracy reaches near-ceiling from mid-scale models onward, format validity improves more gradually with size, and semantic correctness shifts from partial to mostly fully correct outputs.

D. Collaborative Evaluation

In the collaborative setting, agents must align on user intent, participate appropriately, and produce schema-valid, semantically consistent contributions toward a shared goal. As shown in Fig. 3, collaboration preserves the high intent correctness observed in the per-device setting. Format correctness also remains strong, as schema-grounded tool use allows agents to validate outputs independently. Participation correctness reveals both strengths and failure modes. Agents often decline irrelevant queries, such as a thermostat ignoring a lamp-specific power question, but errors arise from over-participation, for example when a lamp interprets an "open or closed" query as its own on/off state. Semantic correctness improves with model scale, indicating that schema grounding remains effective under collaboration. As in per-device tests, all metrics plateau at 4B, suggesting that reliable collaborative behaviour is achieved at practical model sizes.

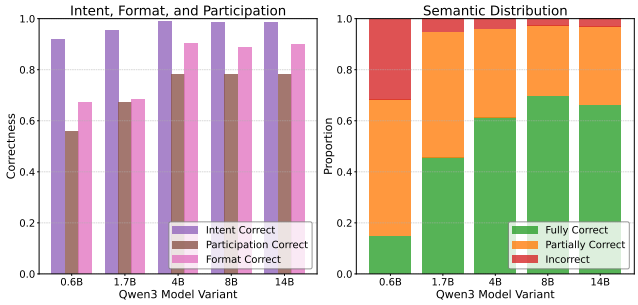

Fig. 3: Performance metrics in the collaborative setting.

VI. Conclusion and Future Work

This study shows that modelling devices as schema-grounded agents enables reliable, state-aware reasoning. By enforcing schema-exposed constraints via tool use, agents remain aligned with operational bounds. Furthermore, lightweight orchestration supports collaboration without multi-device training. However, this work defines enabling conditions rather than a complete solution. While schema grounding ensures per-device alignment and orchestration enables coordination, conflict resolution, consensus formation, group-level evaluation, and real-world deployment remain open problems.

References

[1] P. Rodriguez-Garcia, Y. Li, D. Lopez-Lopez, and A. A. Juan, "Strategic decision making in smart home ecosystems: A review on the use of artificial intelligence and internet of things," *Internet of Things*, 2023.

[2] A. Badshah, A. Ghani, A. Daud, A. Jalal, M. Bilal, and J. Crowcroft, "Towards smart education through internet of things: A survey," *ACM Computing Surveys*, vol. 56, no. 2, pp. 1–33, 2023.

[3] J. Fan, Y. He, B. Tang, Q. Li, and R. Sandhu, "Ruledger: Ensuring execution integrity in trigger-action iot platforms," in *IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[4] F.-Z. Hannou, M. Lefrançois, P. Jouvelot, V. Charpenay, and A. Zimmermann, "A survey on iot programming platforms: A business-domain experts perspective," *ACM Comput. Surv.*, vol. 57, 2024.

[5] H. Weld, X. Huang, S. Long, J. Poon, and S. C. Han, "A survey of joint intent detection and slot filling models in natural language understanding," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–38, 2022.

[6] A. Polo-Rodríguez, L. Fiorini, E. Rovini, F. Cavallo, and J. Medina-Quero, "Enhancing smart environments with context-aware chatbots using large language models," *arXiv preprint arXiv:2502.14469*, 2025.

[7] R. Lou, K. Zhang, and W. Yin, "Large language model instruction following: A survey of progresses and challenges," *Computational Linguistics*, vol. 50, no. 3, pp. 1053–1095, 2024.

[8] Y. Li, "A practical survey on zero-shot prompt design for in-context learning," in *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, 2023, pp. 641–647.

[9] E. King, H. Yu, S. Lee, and C. Julien, ""get ready for a party": Exploring smarter smart spaces with help from large language models," *arXiv preprint arXiv:2303.14143*, 2023.

[10] ——, "Sasha: creative goal-oriented reasoning in smart homes with large language models," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 8, no. 1, pp. 1–38, 2024.

[11] D. Rivkin, F. Hogan, A. Feriani, A. Konar, A. Sigal, X. Liu, and G. Dudek, "Aiot smart home via autonomous llm agents," *IEEE Internet of Things Journal*, 2024.

[12] H. Cui, Y. Du, Q. Yang, Y. Shao, and S. C. Liew, "Llmind: Orchestrating ai and iot with llm for complex task execution," *IEEE Communications Magazine*, 2024.

[13] X. Huang, L. Shen, Z. Ma, and Y. Zheng, "Towards privacy-preserving and personalized smart homes via tailored small language models," *arXiv preprint arXiv:2507.08878*, 2025.

[14] E. King, H. Yu, S. Vartak, J. Jacob, S. Lee, and C. Julien, "Teaching things to think: Bootstrapping local reasoning for smart (er) devices," in *2025 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2025, pp. 78–88.

[15] Z. Yin, M. Zhang, and D. Kawahara, "Harmony: A human-aware, responsive, modular assistant with a locally deployed large language model," *arXiv preprint arXiv:2410.14252*, 2025.

[16] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[17] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing *et al.*, "Judging llm-as-a-judge with mt-bench and chatbot arena," *Advances in neural information processing systems*, vol. 36, pp. 46 595–46 623, 2023.