*International Journal of Latest Research in Engineering and Management (IJLREM)* *ISSN: 2456-0766*
*www.ijlrem.org || Volume 09 - Issue 05 || September-October 2025 ǁ PP. 14-20*

# A Hybrid Agent Framework for Device Control Using Small Language Models on Edge Devices

Yu-Ching Lin, Chia-Chin Chen, Jyh-Horng Wu
*National Center for High-Performance Computing, National Applied Research Laboratories, Taiwan*

**Abstract:** The deployment of intelligent, natural language-driven device control on resource-constrained edge devices presents a significant challenge, forcing a trade-off between the high performance of cloud-based Large Language Models (LLMs) and the privacy and low latency of on-device Small Language Models (SLMs). This paper introduces EdgeControl-AI, a novel framework designed to resolve this performance-privacy dilemma. EdgeControl-AI utilizes a hybrid agentic architecture that operates entirely on-device. Its core is a specialized SLM, optimized via Parameter-Efficient Fine-Tuning (PEFT), which functions as a planning and reasoning engine. This engine is groundedin reality through two key mechanisms: an advanced Retrieval-Augmented Generation (RAG) system that sources context from both static knowledge and a dynamic, lightweight graph-based world model representing the real-time state of the environment. The framework translates natural language commands into structured, verifiable action plans, which are then executed through a secure function-calling module. Our primary contributions include the novel design of this integrated agentic framework and a comprehensive evaluation using simulated benchmarks. The results demonstrate that EdgeControl-AI significantly surpasses baseline on-device approaches in task accuracy, particularly for complex and error-prone instructions, while maintaining exceptional resource efficiency. This work paves the way for a new generation of private, responsive, and highly capable on-device AI assistants for applications in smart homes, IoT, and robotics.

**Keywords:** SLMs, Edge Devices, Device Control, RAG, Agent

## I. Introduction

The paradigm of human-computer interaction is undergoing a fundamental transformation, moving away from rigid, explicit command-line interfaces (CLIs) and graphical user interfaces (GUIs) towards more fluid, intuitive, and intelligent conversational systems. This evolution is driven by the advanced reasoning and natural language understanding capabilities of language models, which can serve as a bridge between high-level human instructions and concrete robotic or device actions. The ultimate vision is one of pervasive, ambient intelligence where a user can interact with a complex ecosystem of interconnected devices—spanning Internet of Things (IoT) appliances, smart home systems, and personal robots—using natural language commands. In such an environment, devices would not only respond to simple directives but also collaborate to interpret ambiguous requests and autonomously execute complex, multi-step tasks based on the user's underlying intent.

While large, cloud-hosted LLMs have demonstrated remarkable capabilities, their application to real-time device control is fraught with inherent limitations that hinder practical deployment. The primary challenges are threefold: 1.Latency: For time-sensitive control applications, such as adjusting robotic manipulators or responding to immediate safety alerts, the network round-trip time required to communicate with a cloud server introduces unacceptable delays. Direct on-device inference eliminates this bottleneck, enabling faster, more responsive interactions. 2. Privacy: Device control, particularly within personal environments like smart homes, involves the continuous generation of sensitive data, including user commands, sensor readings, and device states. Transmitting this data to third-party cloud servers creates significant privacy risks and erodes user trust, as it exposes personal routines and behaviors to potential breaches or misuse. On-device processing ensures that sensitive information remains within the user's local environment. 3. Reliability and Cost: Cloud-centric systems are fundamentally dependent on stable, high-bandwidth internet connectivity. This makes them unreliable in offline scenarios or areas with intermittent network access, which is a critical failure point for essential systems like home security or industrial automation. Furthermore, the reliance on proprietary cloud APIs can lead to substantial and ongoing operational costs, limiting scalability and accessibility.

Small Language Models (SLMs) have emerged as a compelling solution to the drawbacks of cloud-based AI. Designed to be lightweight and efficient, SLMs offer low inference latency, cost-effectiveness, and inherent privacy, making them ideally suited for deployment on edge devices. The rapid development and release of powerful open-source SLMs such as Phi, Gemma, and Qwen have demonstrated that meaningful on-device generative AI is not only possible but increasingly practical for real-world applications. However, this promise is tempered by a significant challenge: the performance gap. Due to their smaller parameter counts, SLMs inherently lack the vast knowledge and complex reasoning capabilities of their larger counterparts. This often results in diminished performance, particularly when faced with ambiguous instructions, complex multi-step tasks, or domains requiring specialized knowledge. The central research question is how can we augment and architect SLM-based systems to bridge this performance gap,

enabling them to execute complex device control tasks with the accuracy and reliability of larger models, but within the constraints of an edge device.

This research introduces EdgeControl-AI, a novel framework designed to overcome the limitations of standalone SLMs and enable high-fidelity device control on resource-constrained hardware. The framework is not merely a compressed model but a complete, grounded agentic system. We propose a modular agent framework that decouples the core cognitive functions of perception, planning, and action.The framework integrates a highly specialized SLM, fine-tuned for planning using Parameter-Efficient Fine-Tuning (PEFT), with an advanced, hybrid Retrieval-Augmented Generation (RAG) system. This RAG mechanism grounds the SLM's reasoning by retrieving real-time context from a dynamic, on-device world model. We detail the implementation of a lightweight, graph-based world model for maintaining a coherent representation of the environment's state. This is coupled with a robust planning and function-calling mechanism that translates the SLM's output into verifiable and executable device commands. Through the simulated experiments, we demonstrate that the EdgeControl-AI framework significantly outperforms baseline on-device approaches in both complex task success rates and resource efficiency, effectively addressing the performance-privacy dilemma.

The core of the problem is not simply about making models smaller, but about fundamentally re-architecting the AI system for the edge. Recent benchmarks show that even state-of-the-art LLMs like GPT-4o fail on complex smart home tasks involving invalid instructions or multiple devices, not because of a lack of general knowledge, but due to a failure of grounding: they hallucinate non-existent devices, ignore contextual state, and misinterpret user intent within the specific environment. Conversely, a narrowly fine-tuned SLM can outperform a large generalist model on a specific task like Android Intent invocation, highlighting the power of specialization. This suggests that the optimal solution is not a single, all-knowing model but rather an intelligent system—an AI agent—that combines a specialized reasoning core with robust mechanisms for perception and action. EdgeControl-AI is the realization of this system-level approach.

## II. Related Work

To situate the novelty and importance of EdgeControl-AI, this section provides a comprehensive review of the state of the art. Our framework sits at the confluence of four key research areas: leveraging language models as agents for control, establishing grounding mechanisms for embodied agents, advanced retrieval-augmented generation techniques, and efficient on-device model specialization. This section systematically reviews these domains to highlight the unique contributions of EdgeControl-AI.

### 2.1 Language Models as Agents for Device and Robotic Control

Using Language Modelsas the "brain" of an agent is an active area of research that aims to leverage their natural language understanding and reasoning capabilities to plan and execute complex tasks [1,2].Many early systems utilized large, cloud-based models like GPT-4 for high-level task decomposition and planning. These systems demonstrated the powerful potential of LLMs to generate complex action sequences in zero- or few-shot scenarios. However, as previously discussed, their reliance on network connectivity, high latency, and privacy concerns make them unsuitable for edge applications requiring real-time response and data sovereignty[3,4].

With the advent of efficient SLMs, researchers have begun exploring agentic systems that run entirely at the edge [5,6,7]. The Vega system, for instance, uses LLMs to enable natural language control of sensors and actuators connected to a Raspberry Pi, demonstrating the feasibility of on-device control. Other work has explored deploying NLP and LLM techniques at the edge for applications like controlling mobile robots. Compared to these systems, EdgeControl-AI offers a more robust and systematic solution for handling environmental dynamics and instruction complexity through its hybrid architecture, dynamic world model, and advanced RAG mechanism.

To address the limitations of single LLMs in complex tasks, such as hallucination and reasoning errors, the research community has proposed hybrid agentic frameworks. The COCORELI framework, for example, integrates multiple medium-sized LLM agents, each specialized for a sub-task, coordinated by a discourse module to reduce hallucination and handle incomplete instructions [8]. This idea of decomposing complex tasks among specialized agents resonates with the modular design philosophy of EdgeControl-AI, which separates functions like perception, planning, and action into distinct deterministic or model-based components. Furthermore, broader research in Multi-Agent Systems (MAS) explores collaboration and communication among multiple autonomous agents to solve problems in complex, distributed systems like smart grids. EdgeControl-AI, as a foundational single-agent controller, provides a solid basis for future expansion into such multi-agent collaborative scenarios.

### 2.2 Grounding Mechanisms for Embodied Agents

For an agent to act effectively in the physical world, its internal reasoning must be consistent with external reality. This requires the agent to have an internal state representation, or "belief state," that connects its planning to the real world. Implementations of world models vary. Some approaches use simple textual memory, where historical observations and action outcomes are stored in natural language for the agent to refer to in subsequent steps. While easy to implement, unstructured text makes precise querying and reasoning difficult and is susceptible to ambiguity in model interpretation[9, 10].

In contrast, structured world models offer more powerful representations. EdgeControl-AI's choice to employ a graph-based world model is a key design decision. In an IoT environment, devices, users, locations, and their relationshipsnaturally form a graph structure. Using a graph database to store this world model allows for the precise capture of these complex entity-relationship structures and supports efficient, complex queries about the state of the environment, such as "List all lights in the living room that are currently on" or "Find the preferred temperature setting for UserA in the evening". This structured knowledge is critical for overcoming grounding failures, as it provides the agent with a fact-based, relationally explicit context that is difficult to achieve with unstructured text.

### 2.3 Advanced Retrieval-Augmented Generation

RAG has become a standard technique for enhancing LLM context by retrieving relevant information from external knowledge bases, thereby improving the factual accuracy of its generations and reducing hallucinations [11, 12]. RAG technology itself is continuously evolving.Initial RAG systems relied primarily on dense retrieval, finding relevant documents through semantic similarity. However, this approach can sometimes miss exact keyword matches. Consequently, hybrid search emerged, combining the semantic understanding of dense retrieval with the keyword-matching precision of sparse retrieval, thereby improving both recall and accuracy.More recent cutting-edge research has further integrated RAG with structured knowledge, leading to Graph-RAG [13]. In Graph-RAG, the retrieval process operates directly on a knowledge graph, leveraging the graph's structure to find information with rich, contextual relationships. Advanced frameworks like GAHR-MSR explicitly combine graph retrieval with vector retrieval in a multi-stage retrieve-and-rerank pipeline to maximize the fidelity of the context provided to the LLM.

EdgeControl-AI's hybrid RAG design is built upon this research frontier [14]. It combines vector retrieval for semantic search with graph retrieval for factual state queries. This design is not incidental but has a deep synergistic effect: the graph-based world model is not just a passive state repository but also the infrastructure for the factual retrieval part of the hybrid RAG system. Vector retrieval provides the semantic knowledge for "how to do it", while graph retrieval provides the factual knowledge for "what to do it with" and "what is the current state". The combination of these two provides the SLM with a comprehensive and grounded context, enabling it to generate action plans that are both consistent with user intent and physical reality.

### 2.4 Efficient On-Device Model Specialization

One of the core challenges of deploying language models on edge devices is resource efficiency. Traditional approaches have focused on model compression, including quantization, pruning, and knowledge distillation. These methods aim to shrink the model's size but can lead to a decline in general performance. In recent years, Parameter-Efficient Fine-Tuning (PEFT) has offered a superior strategy, aiming not just to shrink the model but to specialize it for a specific task while minimizing the demand on computational resources [15,16]. In PEFT methods, the majority of the pre-trained model's weights remain frozen, and only a small fraction of additional parameters are trained.

QLoRA (Quantized Low-Rank Adaptation) is a PEFT technique particularly well-suited for edge deployment. It cleverly combines two strategies: first, it quantizes the base model's weights to 4-bits, drastically reducing the model's memory footprint; second, it employs Low-Rank Adaptation (LoRA), training and storing only a small number of low-rank adapter matrices. This approach not only significantly reduces the computational cost and storage requirements of fine-tuning but also effectively mitigates catastrophic forgetting. Catastrophic forgetting, where a model forgets previously learned knowledge while learning a new task, is a critical issue in agentic systems that need to learn and adapt continuously [17]. By adopting QLoRA, EdgeControl-AI can efficiently specialize a general-purpose SLM into an expert model for device control planning without sacrificing its general pre-trained knowledge.

## III. The Edge Control-Ai Framework

### 3.1 System Architecture

The EdgeControl-AI framework is architected as a modular, closed-loop agentic system designed to operate entirely on a resource-constrained edge device, such as a smart home hub or an integrated robotics controller. The architecture moves beyond a monolithic model design to a system of specialized, interacting components, which enhances robustness, verifiability, and efficiency. The core components are:

1. Perception and Belief State Module: This module acts as the agent's sensory interface, observing the environment and maintaining an internal, real-time "world model."
2. Planning and Reasoning Core: This is the cognitive "brain" of the agent, responsible for interpreting user intent, grounding it in the current context, and formulating a structured, executable plan.
3. Action and Verification Module: This module serves as the agent's effectors, translating the abstract plan into concrete actions by interacting with device APIs.

A user's natural language command initiates a continuous operational loop: the command is processed by the Planning Core, which is informed by the Belief State Module. The resulting plan is passed to the Action Module for execution. The outcome of the action is then fed back to the Perception Module, which updates the belief state. This

cycle enables the agent to handle complex, multi-step tasks, correct errors, and adapt to dynamic changes in its environment, reflecting the principles of classic agent architectures.

The primary function of this module is to provide the agent with an accurate and persistent understanding of its environment, a concept known as situational awareness or grounding. This is critical for overcoming the "hallucination" problem where models generate commands for non-existent or incorrectly configured devices.

We formalize the state management process to ensure the agent's world model remains synchronized with reality. Percepts are defined as discrete, raw data inputs from the environment. These can be sensor readings (e.g., {"device": "thermostat", "state": {"temperature": 25}}), user inputs, or feedback from the Action Module (e.g., {"action": "turn_on_light", "status": "success", "timestamp"}). Beliefs represent the agent's internal, structured, and coherent understanding of the world state. The module continuously runs a belief revision function, $Beliefs_{t+1} = f(Beliefs_t, Percepts_t)$, which updates the agent's world model based on new percepts. This persistent update cycle prevents the agent from operating on stale information, a critical flaw in systems that rely solely on static, prompt-based context. This mechanism is designed to handle the temporal variations and dynamic nature of real-world environments.

To store and manage the agent's beliefs, we propose a lightweight, on-device graph database. This is a pivotal design choice that provides a richer and more structured representation of the environment than simple text or key-value stores. In this graph-based world model, the nodes represent entities, such as devices, users, and abstract concepts.Properties on nodes store their current, real-time state.And edges represent the rich relationships between entities.This structured knowledge graph is superior to unstructured text for grounding an agent. It allows for precise, efficient, and complex queries about the environment's state, such as "List all lights in the living room that are currently on" or "Find the preferred temperature setting for UserA in the evening." This capability is essential for both the RAG system and the action validation process, as it provides factual, relational context that is difficult to capture otherwise.

### 3.2 The Perception and Belief State Module

The primary function of this module is to provide the agent with an accurate and persistent understanding of its environment, a concept known as "situational awareness" or "grounding." This is critical for overcoming the "hallucination" problem, where models generate commands for non-existent or incorrectly configured devices.

We formalize the state management process to ensure the agent's world model remains synchronized with reality. Perceptsare defined as discrete, raw data inputs from the environment. These can be sensor readings (e.g., {"device": "thermostat", "state": {"temperature": 25}}), user inputs, or feedback from the Action Module (e.g., {"action": "turn_on_light", "status": "success", "timestamp"}).Beliefs represent the agent's internal, structured, and coherent understanding of the world state. The module continuously runs a belief revision function, $Beliefs_{t+1} = f(Beliefs_t, Percepts_t)$, which updates the agent's world model based on new percepts. This persistent update cycle prevents the agent from operating on stale information, a critical flaw in systems that rely solely on static, prompt-based context.

To store and manage the agent's beliefs, we propose a lightweight, on-device graph database. This is a pivotal design choice that provides a richer and more structured representation of the environment than simple text or key-value stores. In this graph-based world model, nodes represent entities, such as devices, users, and abstract concepts, properties on nodes store their current, real-time state, and edges represent the rich relationships between entities.

This structured knowledge graph is superior to unstructured text for grounding an agent. It allows for precise, efficient, and complex queries about the environment's state, which is essential for both the RAG system and the action validation process, as it provides factual, relational context that is difficult to capture otherwise. However, maintaining a dynamic knowledge graph on an edge device also presents long-term scalability challenges, including handling growing data volumes, integrating heterogeneous data sources, and efficiently updating and querying the graph on resource-constrained hardware.

### 3.3 The Planning and Reasoning Core

This module is the heart of the framework, translating ambiguous user intent into a precise, executable plan. The reasoning engine is an SLM, such as a model in the 2-billion to 8-billion parameter range (e.g., Gemma, Phi, Llama 3 8B). To overcome the inherent performance limitations of a general-purpose SLM, we employ a highly targeted fine-tuning strategy using QLoRA (Quantized Low-Rank Adaptation). This method is chosen for its exceptional efficiency on edge devices:

- Quantization: The base model's weights are quantized to 4-bits, drastically reducing its memory footprint and making it feasible to run on devices with limited RAM.
- Low-Rank Adaptation (LoRA): Instead of updating the entire model, only a small set of low-rank adapter matrices are trained. This minimizes the computational cost of fine-tuning, reduces storage requirements (as only the small adapter needs to be stored), and mitigates the risk of catastrophic forgetting, where a model loses its general pre-trained knowledge.

The SLM is fine-tuned on a synthetically generated, domain-specific dataset. The objective of this fine-tuning is not to bake factual knowledge into the model, but to teach it the skill of task decomposition and structured plan generation for device control.To provide the SLM with the real-time knowledge it needs to plan effectively, we use an

advanced RAG mechanism. This offloads the burden of "memorizing" facts about the world into an external, quarriable knowledge store. Our framework employs a hybridRAG approach, inspired by recent research in network optimization, which combines two distinct retrieval strategies:

- Vector Retrieval: A vector database, populated with embeddings from device manuals, user guides, and historical user interactions, is used for semantic search. For a query like "make it feel like a movie theater," this can retrieve documents describing typical lighting and sound settings for watching a movie.
- Graph Retrieval: The on-device graph database (our world model) is queried to retrieve the current, factual state of the environment. This provides a precise list of available devices, their current states, their capabilities, and their relationships.

The context retrieved from both the vector store (semantic context) and the graph database (factual context) is prepended to the user's command and fed into the QLoRA-tuned SLM. This dual-pronged approach provides the SLM with both general knowledge and specific, grounded facts.

The primary output of the SLM is a structured plan. We employ a hierarchical planning approach where the SLM first decomposes a high-level, complex command into a logical sequence of smaller, concrete sub-goals. For each sub-goal, the SLM is fine-tuned to generate a formal, machine-readable plan representation. We propose using an intermediate, PDDL-like (Planning Domain Definition Language) structure. For example, the SLM would output a structured object like (action: set_temperature, device: main_thermostat, parameters: {value: 25}). This structured output is a critical design choice. Unlike free-form text, it is unambiguous, programmatically verifiable, and directly executable, which significantly mitigates the risk of model hallucination and ensures the reliability of the generated plan.

### 3.4 The Action and Verification Module

The Action Module is responsible for translating the agent's plan into tangible changes in the environment. It parses each step of the PDDL-like plan generated by the reasoning core and maps it to a specific function call within a predefined library of device control APIs.

Secure Function Calling: The operation of this module can be seen as a secure function-calling mechanism. The SLM itself does not execute any code; it simply generates a structured request (the plan), which is then executed by a secure, deterministic module. For instance, the plan step (action: turn_on, device: living_room_light) would be translated into a concrete Python call like device_api.set_state("living_room_light", {"power": "on"}).

"Plan-Then-Verify" Safety Gate: A crucial feature of this module is pre-execution validation. Before invoking any device API, the module performs a final check against the live belief state stored in the graph database. This "plan-then-verify" step serves as a final safety gate, preventing the agent from attempting to execute hallucinated or invalid actions, which was identified as a major failure mode in other systems. For example, if the plan calls for increasing the temperature of a thermostat, but that thermostat has gone offline due to a power outage before execution (a state updated in the world model), the validation step will fail, preventing the invalid API call and reporting the failure to the planning core to trigger a replan.

The outcome of each executed action (success, failure, new state) is then packaged as a new percept and sent back to the Perception Module, closing the feedback loop and allowing the agent to confirm success or replan if an action fails.

## IV. System Evaluation

### 4.1 Experimental Setupand Benchmarks

To rigorously evaluate the EdgeControl-AI framework, a simulated smart home environment was developed in Python. This environment is inspired by the virtual testbeds described in related research and includes a diverse set of virtual devices with programmable states and APIs, such as lights, thermostats, TVs, security systems, and speakers.The evaluation benchmark is a synthetically generated dataset structured after HomeBench, which is critically important as it includes not only valid commands but also the challenging invalid and multi-device instructions where standard models often fail. The test dataset was categorized into five distinct scenarios to assess performance across a range of complexities:

- Valid Single-Device (VS): Correct instructions for a single device.
- Invalid Single-Device (IS): Incorrect or impossible instructions for a single device.
- Valid Multi-Device (VM): Correct instructions involving multiple devices.
- Invalid Multi-Device (IM): Incorrect instructions involving multiple devices.
- Mixed Multi-Device (MM): Instructions with a mix of valid and invalid operations across multiple devices.

This evaluation focuses on the primary goal of improving control accuracy. Three configurations were compared:

- Baseline SLM: A generic, off-the-shelf Gemma-2B model without any fine-tuning.
- Fine-tuned SLM: The Gemma-2B model fully fine-tuned on a dataset of valid control commands.
- EdgeControl-AI: Our complete framework, featuring the QLoRA-tuned Gemma-2B model integrated with the hybrid RAG and graph-based world model.

The primary metrics, adopted from the HomeBench methodology, were Task Success Rate and F1 Score. Task Success Rate is a strict, holistic measure where a task is only considered successful if the final state of all relevant devices perfectly matches the intended goal state. The F1 score provides a more granular view of the precision and recall of the individual API calls generated by the agent.

The simulated results, presented in Table 1, confirm the hypothesis that the architectural innovations of EdgeControl-AI lead to substantially improved performance. While the Baseline SLM struggles across all but the simplest tasks, and the Fine-tuned SLM shows improvement but fails on complex and invalid instructions, EdgeControl-AI demonstrates robust performance, especially in the most challenging IM and MM scenarios. This high success rate on invalid tasks is a direct result of the framework's grounding and validation mechanisms, which allow it to correctly identify and reject impossible commands rather than attempting to execute them.

Table 1: Task Success Rate and F1 Score Comparison

| Model | Metric | VS | IS | VM | IM | MM |
|---|---|---|---|---|---|---|
| **Baseline SLM** | Success Rate (%) | 65.2 | 8.5 | 21.4 | 5.1 | 3.2 |
| | F1 Score | 0.71 | 0.12 | 0.35 | 0.08 | 0.05 |
| **Fine-tuned SLM** | Success Rate (%) | 94.3 | 25.6 | 55.8 | 15.3 | 18.9 |
| | F1 Score | 0.96 | 0.38 | 0.67 | 0.23 | 0.29 |
| **EdgeControl-AI** | Success Rate (%) | 98.1 | 92.4 | 91.5 | 88.7 | 85.3 |
| | F1 Score | 0.99 | 0.95 | 0.94 | 0.91 | 0.90 |

A framework for edge devices is only viable if it is resource efficient. This evaluation assesses the practical deployment costs of our approach by comparing the resource consumption of a standard fully fine-tuned model against the optimized EdgeControl-AI configuration. The metrics measured include Inference Latency, Peak Memory Usage, and total Model Size on disk. Theresults highlight the profound efficiency gains from using QLoRA. The EdgeControl-AI model has a dramatically smaller storage footprint, as it only requires the 4-bit quantized base model plus a very smallLoRA adapter, compared to the full 16-bit weights of the fine-tuned model. This also translates to significantly lower peak memory usage during inference. The latency is comparable, with the slight increase for EdgeControl-AI attributable to the RAG retrieval step, an acceptable trade-off for the massive gains in accuracy. These results validate that the accuracy improvements demonstrated in Table 1 do not come at an unacceptable performance cost, making the framework suitable for resource-constrained edge deployment.

### 4.2 Analysis of the Performance-Efficiency

The resultsdemonstrate that EdgeControl-AI achieves a superior position on the performance-efficiency Pareto frontier. It delivers accuracy on par with or exceeding what one might expect from a much larger model, but does so within a resource budget suitable for edge devices. This outcome is not merely the result of a better model, but of a better architecture. The synergistic combination of a QLoRA-specialized planner, a hybrid RAG system for grounding, a structured graph-based world model for state management, and a final validation layer in the action module creates a system that is more than the sum of its parts. This suggests that for complex, interactive tasks like device control, sophisticated system architecture and grounding are more critical levers for improving performance than simply increasing a model's parameter count.

The current EdgeControl-AI framework provides a robust but relatively static solution. A key area for future research is enabling the agent to learn, adapt, and generalize over time directly on the device. This presents several challenges and also touches upon broader issues of safety, reliability, and ethics. Ina real smart home, devices are dynamic: new smart appliances are added, and existing ones may receive firmware updates that change their APIs. An ideal agent should be able to automatically discover the capabilities (i.e., available APIs and states) of new devices and integrate them into their world model without requiring a full retraining cycle. This requires the agent to be robust to tool changes and capable of exploring and adapting to its environment without formal documentation. Adapting to New Devices: When a new smart device is added to the network, the agent should be able to automatically discover its capabilities and integrate it into the world model without requiring a full retraining cycle.The agent should be able to learn and refine its understanding of user habits and routines, through observation and interaction, moving towards a truly personalized experience.Ensuring that an agent fine-tuned on one set of devices and tasks can generalize effectively to new, unseen scenarios is a significant challenge. This points towards future research in on-device continual learning, few-shot adaptation, and meta-learning to improve the agent's flexibility and long-term utility.

## V. Conclusion

This research addressed the critical challenge of deploying high-fidelity, intelligent device control on resource-constrained edge hardware. We identified a fundamental performance-privacy dilemma, where the power of cloud-based LLMs comes at the cost of latency and privacy, while on-device SLMs suffer from a significant performance gap. To resolve this, we proposed EdgeControl-AI, a complete on-device agentic framework. The framework's strength lies

in its hybrid architecture, which synergistically integrates a specialized SLM planner, a grounded world model, an advanced RAG system, and a secure action execution loop. The SLM, made efficient via QLoRA, is not tasked with knowing everything, but is instead specialized for planning and reasoning. Its knowledge is augmented in real-time by a hybrid RAG system that pulls factual state from a lightweight graph database and semantic context from vector stores. The resulting structured plans are then validated and executed by a deterministic action module, creating a robust and verifiable control loop. Our simulated evaluations show that this architectural approach allows EdgeControl-AI to dramatically outperform baseline models in handling complex and invalid user instructions, while maintaining a resource footprint suitable for edge deployment. This work demonstrates that by moving from a model-centric to a system-centric design, it is possible to bridge the performance-privacy gap, paving the way for the next generation of truly smart, autonomous, and trustworthy AI agents in our most personal environments.

## Acknowledgements

This research was funded by National Center for High-Performance Computing, National Applied Research Laboratories and the Ministry of Science and Technology in Taiwan.

## References

[1] Huang, W., Abbeel, P., Pathak, D., and I. Mordatch. “Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents.” *Proceedings of the International Conference on Machine Learning*, pp. 9118-9147, 2022.

[2] Kaur, D., Uslu, S., Durresi, M., and A. Durresi. “LLM-Based Agents Utilized in a Trustworthy Artificial Conscience Model for Controlling AI in Medical Applications.” *Proceedings of the 38th International Conference on Advanced Information Networking and Applications*, pp. 198-209, 2024.

[3] X. Guo, D. Kevian, U. Syed, L. Qin, H. Zhang, G. Dullerud, P. Seiler, and B. Hu, “ControlAgent: Automating Control System Design via Novel Integration of LLM Agents and Domain Expertise,” *https://doi.org/10.48550/arXiv.2410.19811*, 2024.

[4] 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, vol. 12, no. 3, pp. 2458-2472, 2024.

[5] Jeon, Changhyun, et al. "SLM-Based Agentic AI with PCG: Optimized for Korean Tool Use." a*rXiv preprint arXiv:2509.19369*, 2025.

[6] Belcak, Peter, et al. "Small Language Models are the Future of Agentic AI." *arXiv preprint arXiv:2506.02153*,2025.

[7] Scherer, Moritz, et al. "Deeploy: Enabling energy-efficient deployment of small language models on heterogeneous microcontrollers." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43.11,pp. 4009-4020, 2024.

[8] Bhar, Swarnadeep, et al. "COCORELI: Cooperative, Compositional Reconstitution& Execution of Language Instructions." *arXiv preprint arXiv:2509.04470*,2025.

[9] Liu, Jia’an, et al. "Agentic AI for sustainable development: Leveraging large language model-enhanced agent-based modeling for complex policy strategies." *Emerging Media* 3.3: 401-413, 2025.

[10] Ferrag, M. A., Tihanyi, N., and M. Debbah. “From LLM Reasoning to Autonomous AI Agents: A Comprehensive Review.” *arXiv preprint arXiv:2504.19678*, 2025.

[11] Z. Zhang, Z. Shen, M. Yuan, F. Zhu, H. Ali and G. Xiong, “RAGTraffic: Utilizing Retrieval-Augmented Generation for Intelligent Traffic Signal Control,” *International Annual Conference on Complex Systems and Intelligent Science*, pp. 728-735, 2024.

[12] Dong, Yuxin, et al. "Advanced rag models with graph structures: Optimizing complex knowledge reasoning and text generation." *IEEE 5th International Symposium on Computer Engineering and Intelligent Communications (ISCEIC)*, 2024.

[13] Procko, Tyler Thomas, and Omar Ochoa. "Graph retrieval-augmented generation for large language models: A survey." *IEEE Conference on AI, Science, Engineering, and Technology (AIxSET)*, 2024.

[14] Shi, Luyao, et al. "Ask-eda: A design assistant empowered by llm, hybrid rag and abbreviation de-hallucination." *IEEE LLM Aided Design Workshop (LAD)*, 2024.

[15] X. Lin, W. Wang, Y. Li, S. Yang, F. Feng, Y. Wei, and T. S. Chua, “Data-Efficient Fine-Tuning for LLM-based Recommendation,” *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 365-374, 2024.

[16] Chen, Yuxuan, Yixin Han, and Xiao Li. "FASTNav: Fine-tuned Adaptive Small-language-models Trained for Multi-point Robot Navigation." *IEEE Robotics and Automation Letters*, 2024.

[17] Paul, S., Zhang, L., Shen, Y., and H. Jin. “Enabling Device Control Planning Capabilities of Small Language Model.” *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 12066-12070, 2024.