DETERMINISTIC EDGE CONTROL FOR AI AGENTS VIA MODEL CONTEXT PROTOCOL OVER MQTT

Simeon Monov, Emre Myumyun

Abstract. The integration of Artificial Intelligence (AI) agents with the Internet of Things (IoT) represents a paradigm shift from static, rule-based automation to dynamic physical orchestration [1]. However, bridging probabilistic Large Language Models (LLMs) with deterministic hardware endpoints introduces severe reliability challenges, most notably the risk of models generating structurally invalid parameters during execution. To address this, we propose a standardized integration architecture utilizing the Model Context Protocol (MCP) over the Message Queuing Telemetry Transport (MQTT) protocol [2]. By explicitly constraining the AI's action space with strict JSON schemas, the framework prevents hallucinated or malformed commands. Transmitting these structured JSON-RPC payloads over MQTT ensures minimized jitter, high temporal consistency, and the deterministic low latency required for safe, reliable real-time edge computing.

Key Words: *autonomous agents, Large Language Models, LLM, Model Context Protocol, MCP, MQTT, JSON-RPC, edge computing, IoT*

Introduction

The integration of Artificial Intelligence (AI) agents with the Internet of Things (IoT) represents a fundamental shift from static, rule-based automation toward dynamic, AI-driven physical orchestration. However, bridging probabilistic Large Language Models (LLMs) with strict, deterministic hardware introduces severe reliability challenges. When tasked with controlling physical devices, LLMs inherently risk generating structurally invalid parameters or hallucinating unsupported device methods. Historically, attempting to mitigate these hallucinations by pre-loading massive device APIs into the model's system prompt causes severe "prompt bloat", which wastes computational tokens and drastically degrades cognitive focus. Furthermore, relying on heavy, traditional network protocols to transmit these bloated payloads introduces unpredictable latency that is fundamentally unsuitable for real-time smart environments.

To enable autonomous AI agents to control physical environments safely, reliably, and in real-time, we propose a standardized integration architecture

utilizing the Model Context Protocol (MCP) over the Message Queuing Telemetry Transport (MQTT) protocol. This framework resolves the tension between probabilistic AI and deterministic hardware through a tripartite approach. First, it avoids prompt bloat by dynamically loading specific device documentation "just-in-time". Second, it eliminates hallucinated commands by explicitly constraining the AI's generation space using strictly enforced JSON schemas. Finally, transmitting these structured JSON-RPC payloads over an asynchronous MQTT transport layer bypasses heavier network protocols, guaranteeing the minimized jitter, high temporal consistency, and deterministic low latency required for safe edge computing.

Related Work

Recent literature extensively explores integrating Large Language Models (LLMs) into Internet of Things (IoT) environments to enable natural language interfaces for physical systems [3, 7]. To standardize these integrations, the Model Context Protocol (MCP) has emerged as a crucial mechanism for bridging the architectural gap between LLMs and connected networks [2]. However, mitigating the probabilistic nature of LLMs to prevent hallucinated or structurally invalid commands remains a critical challenge for deterministic hardware control. While various constraint-guided frameworks have been proposed to enforce structured artifact generation and mitigate hallucinations during LLM execution, our proposed architecture distinctly builds upon these foundations by coupling the MCP framework with the asynchronous Message Queuing Telemetry Transport (MQTT) protocol. By enforcing strict JSON-RPC schemas over an MQTT transport layer, this approach specifically ensures the minimized jitter, high temporal consistency, and deterministic low latency necessary for safe, real-time edge orchestration.

Table 1. Comparison of the proposed architecture with existing AI-IoT orchestration approaches

Criterion	**Proposed (this work): MCP over MQTT**	**IoT-MCP [2]**	**LLM + MCP for IoT data spaces [1]**	**Baseline: LLM with preloaded device APIs**
Standardization layer	MCP	MCP (edge servers)	MCP (fastMCP)	None (raw prompt)
Transport / messaging	MQTT (publish-subscribe)	Edge MCP servers (no MQTT)	HTTP / NGSI-LD	HTTP / REST per device
Context delivery	Just-in-time, per device	Edge-resident server	Tool descriptions	Full APIs in prompt (prompt bloat)
Output constraint	Strict JSON-RPC schema	MCP tool schema	MCP tool schema	Unconstrained

Communication model	Asynchronous, decoupled	Request-response	Request-response	Request-response
Real-time / jitter focus	Explicit (low jitter, determinism)	Latency-aware	Not a focus	Not a focus
Bi-directional state verification	Yes (state read-back)	Yes (sensing + actuation)	Read / query-oriented	Varies
Primary target	Physical actuation + telemetry	Embedded actuation + sensing	IoT data-space querying	General device control

Approach overview

Our proposed architecture functions through a strictly defined execution pipeline that securely connects an AI agent's probabilistic reasoning with physical hardware endpoints. To maintain high reliability and low latency, the system avoids pre-loading massive instruction sets. Instead, it follows a deterministic, step-by-step process entirely abstracted from the user:

1. **Initial Protocol Establishment:** Upon receiving a user command, the agent first retrieves the foundational communication rules required by the specific hardware ecosystem. This ensures the AI understands the baseline grammar needed to formulate valid messages.
2. **Device Resolution:** The system maps the user's natural language request into exact hardware metadata. This step identifies the precise target device on the network and locates its dedicated MQTT communication topic.
3. **Contextual Knowledge Injection:** Once the specific device is targeted, the agent dynamically requests only the technical documentation and functional boundaries relevant to that exact component. This "just-in-time" knowledge delivery prevents the AI from being overwhelmed with irrelevant data, maintaining high cognitive focus.
4. **Constrained Action Generation:** Using the newly acquired documentation, the AI formulates a structurally valid JSON-RPC payload [5]. By enforcing these strict schema boundaries, the system completely blocks hallucinated or unsupported actions.
5. **Asynchronous Execution:** The finalized payload is transmitted via MQTT, which operates as an asynchronous publish-subscribe protocol. Publishing over this lightweight protocol bypasses heavier traditional network methods, guaranteeing the minimized jitter and high temporal consistency required for real-time edge control [3, 8].
6. **State Verification and Feedback:** Because MQTT is an asynchronous protocol, the execution pipeline explicitly waits for the hardware

endpoint to process the action and publish its structured JSON response back to the system. The agent interprets this verified state and translates it into accurate, natural language feedback for the user, confirming either a successful physical change or delivering the requested device data.

System architecture components

The proposed architecture is delineated into three primary operational layers, ensuring a strict separation of concerns between cognitive processing and physical execution. At the cognitive layer, the Large Language Model serves as the central reasoning engine, responsible for interpreting natural language intent and formulating structured actions based on dynamically injected constraints. The intermediary integration layer consists of the Model Context Protocol (MCP) server, which abstracts device discovery, documentation retrieval, and payload construction into standardized tool calls. Finally, the physical execution layer comprises the local MQTT broker and the corresponding hardware endpoints, such as edge sensors and smart relays. This physical layer handles the asynchronous transmission of JSON-RPC payloads, guaranteeing the low-latency and decoupled communication necessary for edge computing environments.

Figure 1. High-level system architecture: the AI client (cognitive layer), the MCP server (integration layer), and the physical MQTT layer

Model Context Protocol (MCP) details

The Model Context Protocol (MCP) functions as the critical standardization mechanism within this architecture, explicitly resolving the tension between deterministic hardware requirements and probabilistic AI models [4]. Rather than relying on static, monolithic system prompts that degrade model focus, the MCP framework exposes device capabilities, technical schemas, and local state data as standardized, callable resources. This allows the AI agent to traverse a secure, predictable environment where necessary context is acquired dynamically. By formalizing the interface between the language model and the physical local network, the MCP ensures that the cognitive engine is rigorously bounded by verifiable hardware constraints, systematically eliminating the generation of arbitrary commands or unsupported methods [6].

Model performance evaluation

To quantitatively evaluate the efficiency of the proposed architecture, an empirical latency test was conducted across multiple variants of the OpenAI model family, specifically comparing the GPT-5 and GPT-5.4 generations. Each

model was subjected to an identical natural language prompt: ***"Turn on the living room light!"*** The primary metric recorded was the end-to-end execution latency, measured strictly from the moment the user command was issued until the physical action was successfully executed by the hardware endpoint.

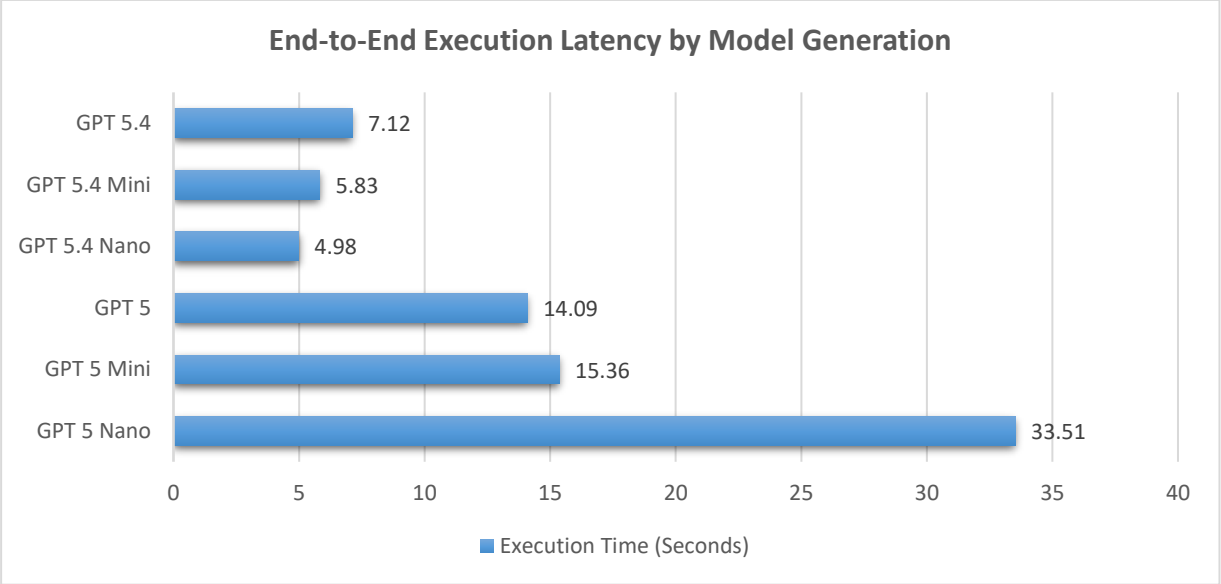

Figure 2. End-to-end execution latency by model generation (GPT-5 vs. GPT-5.4 variants)

The experimental results demonstrate a substantial generational improvement in tool-calling efficiency and overall reasoning speed. The GPT-5.4 family consistently exhibited significantly lower execution times compared to their GPT-5 counterparts. Furthermore, the tests revealed that within the newer generation, the more compact models optimized the multi-step execution pipeline most effectively, achieving the fastest physical orchestration. Conversely, the older models exhibited noticeable latency bottlenecks while navigating the identical schema-constrained steps, highlighting the importance of advanced tool-use capabilities for real-time edge computing. Table 2 reports the corresponding statistics (mean, variance, and 95% confidence interval) for each model over N = 30 runs.

Table 2. End-to-end execution latency statistics per model (N = 30 runs each)

Model	N	Mean (s)	Std. dev. (s)	Variance (s^2)	95% CI (s)
GPT-5 Nano	30	33.51	1.36	1.84	33.00–34.02
GPT-5 Mini	30	15.36	0.64	0.41	15.12–15.60
GPT-5	30	14.09	0.63	0.40	13.85–14.32
GPT-5.4 Nano	30	4.98	0.27	0.07	4.88–5.08
GPT-5.4 Mini	30	5.83	0.33	0.11	5.71–5.95
GPT-5.4	30	7.12	0.34	0.11	6.99–7.25

Test experiments

To further validate the bi-directional capabilities of the framework, a series of state verification experiments were conducted. The first experiment tested the AI agent's ability to accurately retrieve and report real-time telemetry data from the physical environment.

The first experiment evaluated the AI agent's ability to accurately retrieve real-time telemetry data. To test this, the user issued a natural language query requesting the current power consumption of a specific target ('living room light'). Utilizing the established deterministic execution pipeline, the agent successfully queried the hardware endpoint and accurately reported a real-time consumption of 44.4 W back to the user. This response was independently verified against the target device's local management interface, which confirmed the active power output was exactly 44.4 W at the time of the request. This experiment successfully demonstrates that the architecture's strict schema constraints effectively prevent AI hallucinations when interpreting and reporting dynamic, real-time sensor data.

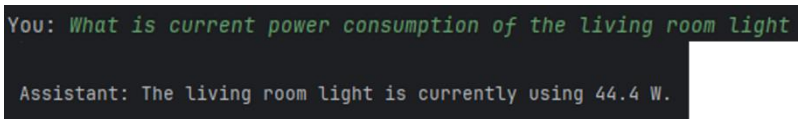

Figure 3. Agent response reporting the real-time power consumption (44.4 W) of the living room light

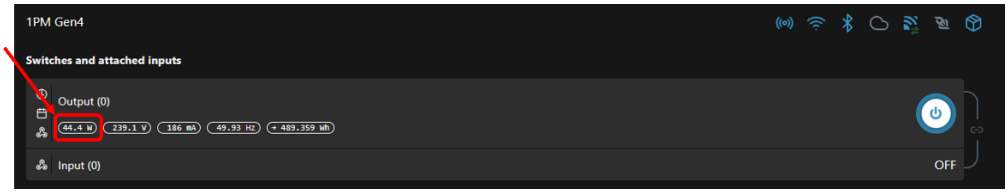

Figure 4. The device's local management interface confirming the 44.4 W power output (independent verification)

In the second experiment, the system's capability to retrieve and interpret historical energy metrics was evaluated. The user queried the system to determine how much energy the living room light had consumed over the last minute. Utilizing the schema-constrained execution flow, the AI agent successfully accessed the device's localized energy logs and accurately reported a consumption of about 598.6 mWh for the last complete minute.

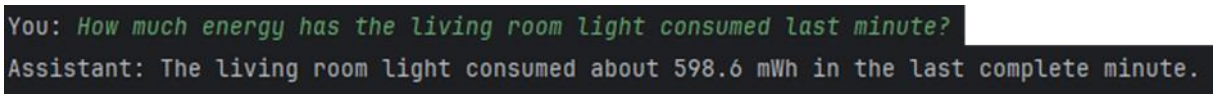

Figure 5. Agent response reporting the historical energy consumption (598.6 mWh) over the last complete minute

The third experiment assessed the agent's ability to perform system-level maintenance checks beyond standard physical actuation. The user prompted the

agent to check if a new firmware update was available for the target device. By dynamically loading the relevant system documentation via the MCP server, the agent queried the hardware's update status and confirmed that a new stable firmware version (1.7.5) was available. This highlights the architecture's extended utility for autonomous IoT device management and maintenance.

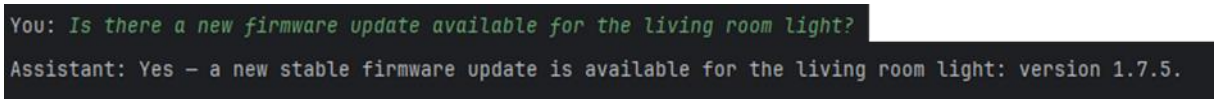

Figure 6. Agent response confirming the availability of a new firmware version (1.7.5)

The final experiment tested the AI's capacity to accurately read and report basic operational states. The user submitted a straightforward query asking whether the living room light was currently turned on. The agent processed the request, retrieved the current status from the designated MQTT topic, and provided a definitive natural language confirmation that the living room main light was currently on. This confirms the foundational reliability of the bi-directional communication loop in tracking and verifying real-time physical states without hallucinating false positive or negative statuses.

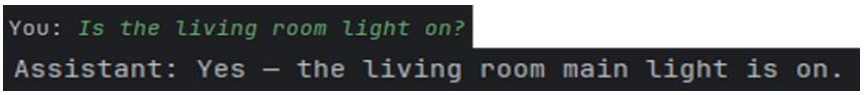

Figure 7. Agent response confirming the real-time on/off state of the living room light

These experiments demonstrate the architecture's robust performance and reliability in managing bi-directional communication. The AI agent successfully sends targeted requests, awaits responses, and accurately delivers device data to the user. Achieving this deterministic feedback loop over an inherently asynchronous MQTT transport layer minimizes jitter, guaranteeing that probabilistic LLMs can safely and reliably control physical environments in real-time.

Acknowledgments

This study is supported by the project SP25-FMI-008 "Research and implementation of modern language models and artificial neural networks for automated processing, forecasting, and structuring of data and texts in specific application domains" at the Plovdiv University "Paisii Hilendarski".

References

[1] A. Athanasopoulou, N. Fotiou, A. Chatzopoulos, Interacting with IoT Data Spaces Using LLMs and the Model Context Protocol, *Sensors*, 2026, vol. 26, pp. 1193, ISSN: 1424-8220, DOI: https://doi.org/10.3390/s26041193

[2] N. Yang, G. Lyu, M. Ma, Y. Lu, Y. Li, Z. Gao, H. Ye, J. Zhang, T. Chen, Y. Chen, IoT-MCP: Bridging LLMs and IoT Systems Through Model Context Protocol, *Proc. of the ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization (WiNTECH'25)*, 2025, DOI: https://doi.org/10.1145/3737895.3768303

[3] A. Kalita, Talk with the Things: Integrating LLMs into IoT Networks, *arXiv preprint*, 2025, DOI: https://doi.org/10.48550/arXiv.2507.17865

[4] Model Context Protocol Specification, Model Context Protocol Community, 2025, available at https://modelcontextprotocol.io, accessed: 27 June 2026

[5] S. Wang et al., ATLAS: A Layered Constraint-Guided Framework for Structured Artifact Generation in LLM-Assisted MDE, *arXiv preprint*, 2025, DOI: https://doi.org/10.48550/arXiv.2510.25890

[6] L. Grillo, LLMASP: A Framework for Mitigating Hallucinations and Supporting Symbolic Reasoning in Large Language Models, *Joint Proc. of the Workshops and Doctoral Consortium of the 41st International Conference on Logic Programming (ICLP-WS-DC'2025)*, CEUR Workshop Proceedings, 2025, vol. 4117, ISSN: 1613-0073, available: https://ceur-ws.org/Vol-4117/paper_dc_9.pdf

[7] L. Wang, C. Ma, X. Feng, et al., A survey on large language model based autonomous agents, *Frontiers of Computer Science*, 2024, vol. 18, pp. 186345, ISSN: 2095-2228, DOI: https://doi.org/10.1007/s11704-024-40231-1

[8] E. Shahri, P. Pedreiras, L. Almeida, Extending MQTT with Real-Time Communication Services Based on SDN, *Sensors*, 2022, vol. 22, pp. 3162, ISSN: 1424-8220, DOI: https://doi.org/10.3390/s22093162

Simeon Monov[1,*], Emre Myumyun[1]

[1] Paisii Hilendarski University of Plovdiv,

Faculty of Mathematics and Informatics,

236 Bulgaria Blvd., 4027 Plovdiv, Bulgaria

Corresponding author: smonov@uni-plovdiv.bg