International Journal of Engineering and Technology Innovation, vol. 16, no. 2, 2026, pp. 239-252

Design of Equipment Control System Based on Large Language Models with Intelligent Validator Agent

Yu-Ching Lin*, Chia-Chin Chen, Jyh-Horng Wu

National Center for High-Performance Computing, National Institutes of Applied Research, Taipei, Taiwan, ROC

Received 03 November 2025; received in revised form 11 February 2026; accepted 26 February 2026

DOI: https://doi.org/10.46604/ijeti.2026.15849

Abstract

This study presents a comprehensive framework leveraging large language models (LLMs) for robust and accurate equipment control. The system integrates Retrieval-Augmented Generation (RAG) to provide LLMs with real-time contextual information retrieved from external knowledge sources. Furthermore, a Mixture-of-Agents (MoA) architecture is employed to harness the collective intelligence of multiple LLMs, enhancing the quality and reliability of generated control suggestions. The proposed intelligent Validator Agent, equipped with natural language processing capabilities and a feedback-correction mechanism, serves as the core component. The Validator Agent translates the LLM's textual responses into actionable commands, validates them against real-time equipment status and predefined rules from a database, and corrects potential errors before execution. Performance evaluation demonstrates that although RAG and MoA improve accuracy, the Validator Agent's correction loop remains essential. This research highlights the potential of combining LLMs with RAG, MoA, and intelligent agents to create highly accurate and reliable natural language-based equipment control systems.

Keywords: large language model (LLM), Retrieval-Augmented Generation (RAG), Mixture-of-Agents (MoA), Validator Agent, equipment control

1. Introduction

The proliferation of large language models (LLMs) has revolutionized natural language processing (NLP), enabling intuitive human-machine interfaces across various domains. Despite LLMs exhibiting exceptional capabilities in understanding instructions, applying them to industrial equipment control presents unique challenges. Unlike dialogue systems, where approximate answers are tolerable, equipment control requires high precision, reliability, and strict adherence to safety protocols. Recently, LLM-based equipment control has emerged as an important research direction. This involves translating natural language instructions into specific operational commands for physical or virtual devices. To attain reliable and precise control, integrating LLMs with techniques such as fine-tuning [1-2], Retrieval-Augmented Generation (RAG) [3], Mixture-of-Agents (MoA) [4-5], and intelligent AI agents [6-7] becomes crucial. RAG grounds LLM responses in real-time external data, MoA leverages multiple models for improved reasoning, and AI agents provide a vital layer of validation and correction before command execution.

This paper delineates a framework that combines these techniques to architect a robust LLM-based equipment control system. The methodologies employed are explicated, an algorithm centered around the Validator Agent feedback loop for accuracy enhancement is formulated, the system architecture and implementation are presented, and performance is evaluated

* Corresponding author. E-mail address: 1203043@narlabs.org.tw

through exhaustive empirical testing. The findings demonstrate the viability of this approach and highlight the key role of the Validator Agent in improving control accuracy. The deployment of LLMs in engineering has transitioned from rudimentary text processing to complex decision-making in industrial automation and robotics. Key developments in LLM-based control are reviewed, and the proposed framework is explicitly differentiated from existing approaches by comparing some critical architectural constituents.

As articulated by Sajja and Addula [8], the integration of robotic automation and AI algorithms enables systems to learn from operational data, thereby augmenting production quality, reducing errors, and optimizing efficiency. Furthermore, the scope of industrial automation is expanding towards "smart manufacturing", where the synergy between AI perception and robotics becomes crucial. Addula and Tyagi [9] emphasized that advancing industrial robotics lies in integrating advanced AI capabilities to transform machines from executing simple, repetitive tasks to handling complex, context-aware manufacturing processes. Xia et al. [10] proposed deploying LLM agents to control industrial automation systems, enhancing operational flexibility by enabling machines to interpret and execute instructions that typically require specialized programming.

Similarly, Huang et al. [11] and Wang et al. [12] explored LLMs as zero-shot planners in robotics, converting natural language into action sequences. Fan et al. [13] further investigated the concept of embodied intelligence in manufacturing, demonstrating how LLMs can be leveraged to enhance autonomous industrial robotics. Additionally, Paul et al. [14] transferred these capabilities to small language models (SLMs) for smart home device control. While these studies demonstrate the potential for intuitive human-machine interaction, they typically rely on the probabilistic reasoning of the model per se. The architectural constraint in these "direct control" frameworks is the absence of a distinct, deterministic validation layer. The correctness of the operation mostly depends on the LLM's immediate output quality; if the model hallucinates or misinterprets a prompt, the incorrect command is passed directly to the execution layer.

To address the limitations of single-model approaches, research has transitioned towards agent-based systems that incorporate reasoning capabilities. Wang and Qin [15] proposed an intelligent regulation system based on LLM agents for industrial production, focusing on self-learning to optimize energy efficiency. Rivkin et al. [16] explored the deployment of autonomous LLM agents in AIoT environments for smart home applications. Expanding on agent capabilities, Sun et al. [17] assessed the challenges and potential of LLM-based multi-agent systems in complex decision-making processes. Furthermore, Liu et al. [18] introduced "Agents4PLC," which utilizes LLM agents to automate the closed-loop generation and verification of PLC code for industrial control systems. However, these agent-based frameworks primarily focus on optimization, design iteration, or long-horizon planning instead of immediate operational safety. Existing architectures often lack a stringent verification mechanism that validates individual commands against real-time physical states before execution.

As LLMs are increasingly deployed in safety-critical environments, recent studies have accentuated the necessity of deterministic verification layers to mitigate stochastic failures. Galitsky and Rybalov [19] identified "process-control hallucinations," where LLMs generate plausible but physically dangerous commands, as a primary risk in industrial settings. They advocated for neuro-symbolic architectures that combine the generative power of LLMs with rigid logic constraints. Similarly, Ogenyi et al. [20] postulated that in the era of autonomous Internet of Things (A-IoT), security systems must transition from static rules to dynamic anomaly detection, whereas they explicitly cautioned that autonomous decisions in critical infrastructure still require formal guardrails. These concepts demonstrate the importance of agent design.

Collectively, the aforementioned studies illuminate the potential of LLMs in various control contexts and, simultaneously, reveal persistent challenges pertaining to accuracy, reliability, safety, real-time performance, and handling ambiguity. Distinguishing itself from existing literature, the proposed work addresses the critical issue of accuracy and reliability by synthesizing RAG for context-awareness, MoA for enhanced reasoning, and a crucial Validator Agent feedback loop for validation and correction.

2. Methodologies and System Design

This study proposes a composite framework tailored for high-precision equipment control. The architecture of the proposed equipment control system functions as a local framework that seamlessly integrates NLP with deterministic industrial control protocols, as depicted in Fig. 1. The system architecture integrates four key technological components: domain-specific model fine-tuning, advanced RAG, hierarchical MoA, and the intelligent Validator Agent. The following sections detail the implementation strategies and design rationale for each component.

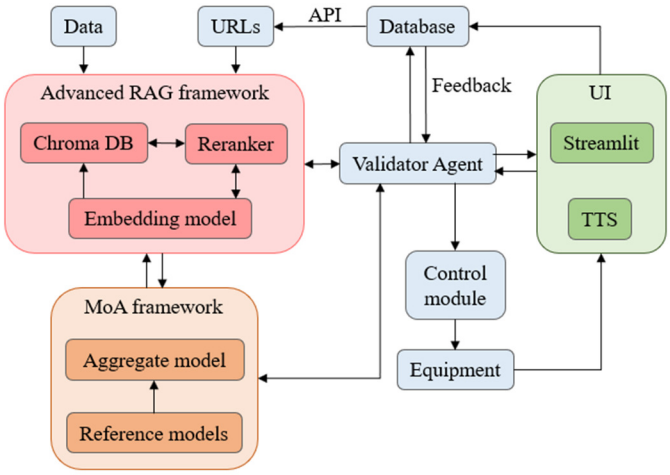

Fig. 1 The architecture of the equipment control system

2.1. *Domain-specific model fine-tuning*

Fine-tuning plays a crucial role in adapting the pre-trained LLM to domain-specific tasks. To ensure strict adherence to industrial control APIs, a targeted supervised fine-tuning (SFT) strategy was implemented using quantized low-rank adaptation (QLoRA). A proprietary dataset of 2,000 "instruction-input-output" triplets was established to train the model to map natural language commands to precise JSON-formatted control parameters. NF4 quantization was applied to the base models, and low-rank adapters with rank $\gamma = 64$ and scaling factor $\alpha = 16$ were injected. The SFT phase effectively reduced format errors in subsequent generation phases.

Fine-tuning enables the model to leverage its pre-existing knowledge while specializing in its capabilities for the specific requirements of equipment control, avoiding the prohibitive cost of training a large model from scratch. This approach ensures the generation of structurally valid command objects, allowing the subsequent Validator Agent to focus on logic validation rather than syntax parsing.

2.2. *Advanced Retrieval-Augmented Generation (RAG)*

The accuracy of the RAG system is highly dependent on the quality of the embedding model. Standard vector search (i.e., bi-encoder) calculates similarity based on vector distance, which can lead to retrieval errors in industrial contexts where negations matter. To mitigate these errors, a hybrid search based on dense and sparse vectors (BM25) was implemented, followed by cross-encoder re-ranking. The system ingests two distinct types of data into the Chroma DB vector store. One is static control protocols, encompassing predefined operational rules and safety constraints. The other is dynamic sensor streams, which consist of real-time sensor data obtained through Web APIs. To maintain low latency, sensor data is formatted as standardized JSON URLs, which are refreshed and re-embedded at fixed intervals.

Although standard vector search excels at capturing semantic intent, it is often vulnerable to lexical divergence, particularly when processing exact industrial equipment identifiers or specific error codes. By implementing the hybrid search strategy, the system leverages the precision of sparse keyword retrieval to capture exact terminologies, whereas dense vector

retrieval handles the semantic nuances of the user's intent. This architectural choice substantially reduces the system's susceptibility to prompt ambiguity. By providing the LLM with strictly re-ranked context that delineates the current sensor values and corresponding safety rules, the generation process becomes highly constrained and grounded. Previous studies have shown that the "retrieve-then-rerank" architecture enhances robustness in noisy environments compared to single-stage retrieval [21]. Within the proposed framework, this approach establishes a strictly constrained and grounded foundation for subsequent validation phases.

2.3. MoA framework

To mitigate the stochastic errors and reasoning limitations inherent in single-model generation, the MoA framework is employed in the system. This architecture is engineered to enhance control signal robustness by aggregating the collective intelligence of heterogeneous models, thereby establishing a collaborative reasoning layer before any command is proposed for validation. The system employs a two-tier "proposer-aggregator" structure, designed to balance computational efficiency with reasoning depth. The heterogeneous proposers layer comprises three distinct LLMs acting as proposers: Fine-tuned Llama-3.2 3B, Mistral 7B, and Phi-3.5 8B. Among them, Mistral 7B demonstrates superior performance in handling complex conditional logic. Conversely, the compact Llama-3.2 3B is applied for its low latency and ability to process standard, unambiguous commands rapidly.

This diversity ensures that the system circumvents reliance on the biases or specific failure modes of a single architecture. It employs the model's inherent self-attention mechanism to dynamically assign attention weights to each proposer's output based on their coherence and alignment with the RAG context. The outputs from the Proposer layer serve as reference inputs for the Aggregator Agent, powered by the Llama-3.1 70B model. As the most capable model in the pipeline, the Aggregator selects the optimal answer and synthesizes a final response by evaluating the consensus and resolving discrepancies among the proposers, allowing the system to statistically exclude hallucinations from smaller models.

A prominent merit of this hierarchical design is the reduction of system sensitivity to individual model hallucinations. In equipment control, a single model might erroneously interpret a device ID or fail to recall a specific context constraint. By incorporating the Aggregator layer, such outliers are statistically discounted. For instance, if the smaller Llama-3.2 3B model hallucinates a non-existent device ID, whereas both Mistral 7B and Phi-3.5 8B generate the correct ID, the Aggregator is designed to recognize this pattern and discard the erroneous proposal. While this mechanism incurs higher computational and time costs, it ensures that the final suggestion passed to the Validator Agent is of higher quality than what could be achieved by any single model. This architectural redundancy is essential for maintaining high availability and reliability in industrial environments.

2.4. Intelligent Validator Agent

In previous studies [16-17], AI agents have been integrated with LLMs to perform perception, decision-making, and task execution similar to humans in open environments. To address the stringent requirement for high accuracy in equipment control, this study introduces a dedicated supervision agent, designated as the Validator Agent. This agent mediates between the NLP capabilities of the LLM and the precise operational requirements of industrial devices.

The primary function of the Validator Agent is to validate, correct, and ensure the safe and accurate execution of commands. To achieve contextual awareness, the agent integrates with IoT platforms to continuously monitor the real-time state of the environment. Functioning as a rigid firewall, the Validator Agent not only executes commands but also rigorously validates them against physical constraints using a quantitative scoring algorithm. Upon receiving a parsed control command, it initiates a multi-faceted evaluation to calculate a composite validation score, employing distinct logic checks with weighting factors empirically tuned to prioritize operational safety.

Furthermore, interfacing industrial control APIs with LLMs introduces significant cybersecurity risks. Malicious inputs could exploit the model to generate harmful API calls. To mitigate these risks, the Validator Agent implements rigorous safeguards to prevent the system from becoming a vector for prompt injection attacks. Its control algorithm systematically validates and corrects the intentions generated by the LLMs before execution. The control algorithm proceeds as follows:

(1) Receive LLM response (R_{LLM}): The Validator Agent receives a textual response from the LLM, denoted as R_{LLM}. This response represents the LLM's interpretation of the user's command, potentially enhanced through RAG and MoA.

$$R_{LLM} = LLM\left(UserPrompt, Context_{RAG}, Config_{MoA}\right) \tag{1}$$

(2) Semantic parsing and command translation (Cmd_{parsed}): The Validator Agent parses R_{LLM} to extract specific control parameters: target devices (D_t), intended actions (A_i), and associated values (V_p) (e.g., temperature settings, on/off state). It then translates these into a structured command format, Cmd_{parsed}, suitable for further processing and device interaction (e.g., JSON). By enforcing stringent semantic parsing, the system prevents the LLM from directly accessing the control API, and its output is validated against a strict JSON schema. If a malicious command attempts to inject code outside the predefined action whitelist, the parsing step will fail to map such inputs to valid A_i, and the Validator Agent will immediately reject the command.

$$Cmd_{parsed} = Parse\left(R_{LLM}\right) \rightarrow \left\{D_t, A_i, V_p\right\} \tag{2}$$

(3) Contextual verification data acquisition (D_{cv}): For each D_t in Cmd_{parsed}, the Validator Agent queries external databases and sensor networks via API calls. These queries retrieve the real-time status of the target devices ($S_{rt}(D_t)$) (e.g., current temperature, humidity, operational state), the predefined operational rules and constraints ($Rules(D_t, A_i)$) (e.g., safe operating temperature ranges, interlocks with other devices, energy-saving policies), the historical operational data ($H_{data}(D_t)$) for anomaly detection, and the user preferences ($Prefs(User, D_t)$) if applicable.

$$D_{cv} = \left\{S_{rt}\left(D_t\right), Rules\left(D_t, A_i\right), H_{data}\left(D_t\right), Prefs\left(User, D_t\right)\right\} \tag{3}$$

(4) Validation and confidence assessment (V_{agent}): The Validator Agent applies a multi-stage validation logic to Cmd_{parsed} using D_{cv}. This step calculates a validation score, V_{score}, for the parsed command. The validation procedure is as follows:

(a) Feasibility check (F_{feas}): Determine whether A_i is feasible given $S_{rt}(D_t)$ (e.g., a device marked as "under maintenance" cannot be turned on). $F_{feas}(Cmd_{parsed}, S_{rt}) \in \{0, 1\}$.

(b) Rule compliance check (F_{rule}): Evaluate whether Cmd_{parsed} violates any $Rules(D_t, A_i)$ (e.g., setting temperature outside the allowed range). $F_{rule}(Cmd_{parsed}, Rules) \in \{0, 1\}$.

(c) Anomaly detection ($F_{anomaly}$): Assess whether the command is anomalous compared to $H_{data}(D_t)$ or typical operational patterns. $F_{anomaly}(Cmd_{parsed}, H_{data}) \in [0, 1]$, where 1 indicates a high anomaly.

(d) Conflict resolution ($F_{conflict}$): Check for conflicts with other scheduled tasks or the states of interdependent devices. $F_{conflict}(Cmd_{parsed}, SystemState) \in \{0, 1\}$.

The overall validation score, V_{score}, is defined as a weighted function of these checks and the initial LLM confidence, $C(R_{LLM})$:

$$V_{score} = w_1 C\left(R_{LLM}\right) + w_2 F_{feas} + w_3 F_{rule} - w_4 F_{anomaly} + w_5\left(1 - F_{conflict}\right) \tag{4}$$

where w_i represents the weights corresponding to the importance of each validation check. The weights were empirically tuned to prioritize safety and rule compliance, and the specific values assigned in this study were: the confidence weight $w_1 = 0.3$, the feasibility weight $w_2 = 0.2$, the rule compliance weight $w_3 = 0.3$, the anomaly detection weight $w_4 = 0.1$, and the conflict resolution weight $w_5 = 0.1$. A command is considered valid if $V_{\text{score}} \geq \theta_{valid}$, where θ_{valid} is a predefined threshold. For the experiments, θ_{valid} was set to 0.85 to ensure a high margin of safety before execution.

(1) Correction and refinement loop (CR_{agent}): If $V_{score} < \theta_{valid}$, or if specific checks fail (e.g., $F_{rule} = 0$), the Validator Agent triggers the correction and refinement process:

(a) Deterministic (rule-based correction): If an explicit rule violation is detected and a deterministic correction is available (e.g., the LLM suggests "cooler OFF" but temp is > 40 ℃ and the rule is "cooler ON if temp > 35 ℃"), the Validator Agent corrects it to "cooler ON". Let $Correct_{rule}(Cmd_{parsed}, D_{cv})$ denote the rule-based correction function.

(b) Iterative refinement (clarification query to LLM): If ambiguity is elevated or a simple rule-based correction is insufficient, the Validator Agent formulates a new prompt to the LLM, incorporating the discrepancy detected from D_{cv} or the reason for low confidence.

$$Prompt_{clarify} = FormulateClarification\left(UserPrompt, R_{LLM}, DiscrepancyInfo\right) \tag{5}$$

$$R_{LLM}{}' = LLM\left(Prompt_{clarify}, RAG_{context}\right) \tag{6}$$

The process subsequently loops into step 2 with R_{LLM}'. This loop continues for a maximum of N_{iter} iterations or until $V_{\text{score}} \geq \theta_{valid}$.

(c) User confirmation (human-in-the-loop): In scenarios entailing severe structural discrepancies, or if iterative refinement fails, the Validator Agent escalates to the user for confirmation or manual override. The terminal output of this loop is a refined command, $Cmd_{refined}$, or a decision to abort if validation fails persistently.

(1) Validated command dispatch ($E_{dispatch}$): Provided Cmd_{parsed} is validated (i.e., $V_{\text{score}} \geq \theta_{valid}$), or $Cmd_{refined}$ is produced and validated, the Validator Agent dispatches the final command (Cmd_{final}) to the designated device control module: $E_{dispatch}(Cmd_{final} \rightarrow D_t)$.

(2) Execution feedback ($M_{feedback}$): The Validator Agent monitors the device's response to Cmd_{final} to verify successful execution. This feedback is logged to dynamically update the knowledge base ($Rules, H_{data}$), while providing learning information for future Validator Agent decisions.

The overall objective is to maximize the probability of correct execution, $P(CorrectAction \mid UserPrompt)$, by optimizing the Validator Agent's correction policy π_{agent}. The overall objective function can be expressed as follows:

$$Maximize\left(\pi_{agent}\right)\sum_t P\left(CorrectAction_t \middle| UserPrompt_t, D_{cvt}, \pi_{agent}\right) \tag{7}$$

To demonstrate the efficacy of the Validator Agent's correction mechanism, a representative scenario involving a conflict between an ambiguous user command and strict safety constraints is presented. In this instance, a user requested to set a laboratory dehumidifier to the maximum available desiccation while a safety rule mandated a minimum humidity of 30% to prevent static discharge. The MoA layer initially processed the user's intent literally, generating a command to set the target humidity to 0%. However, during the contextual verification phase, the Validator Agent detected a critical violation of the safety rule (0% < 30%), resulting in an insufficient validation score ($V_{score} = 0.6$). Consequently, the agent triggered its deterministic correction loop (CR_{agent}), automatically overriding the unsafe value with the database-defined lower bound of 30%. The refined command, currently achieving a passing score ($V_{score} = 0.9$), was dispatched to the equipment, and the system transparently informed the user that the setting was adjusted to the safety limit. This process illustrates how the agent effectively acts as a deterministic firewall against plausible but hazardous LLM hallucinations.

To ensure the infrastructure's robustness matches industrial specifications, the Validator Agent is designed to conform to the foundational functions of the NIST Cybersecurity Framework (CSF) 2.0 and NIST SP 800-82 (Guide to Industrial Control Systems Security). Specifically, the Agent implements the Protect function by enforcing strict JSON schema parsing and whitelist-based command translation, which satisfies the SI-10 (Information Input Validation) control to establish a boundary

against malformed or malicious injections. Furthermore, the system addresses the Detect function through its anomaly detection logic ($F_{anomaly}$), aligning with DE.AE (Anomalies and Events) by continuously comparing proposed commands against historical operational baselines (H_{data}) to identify potential compromises. Finally, the Respond function is realized via the correction and refinement loop (CR_{agent}), serving as an automated RS.MI (Risk Mitigation) strategy that actively neutralizes hazardous commands in real-time, thereby ensuring system stability even under erroneous LLM behavior.

While the rule-based core safety mechanism is retained, the Validator Agent is justified by its agentic workflow. This Validator Agent control algorithm, with its emphasis on quantitative validation and iterative refinement, positions the agent as a critical intelligence layer that goes beyond translation. It actively ensures that LLM-generated commands are accurate, safe, and contextually appropriate before execution, thereby improving the overall reliability of the equipment control system.

3. System Evaluation

To rigorously validate the proposed framework, a comprehensive experimental scenario that simulates the operational criticality of modern AIoT industrial systems was designed. This section details the experimental configuration, the rationale behind the command datasets, and the performance metrics used to quantify system reliability.

3.1. Experimental setup

The experimental environment adopts the control system of intelligent computer facilities as an example. The system comprises five distinct servers (Hosts A through E). Each host integrates a multi-variable sensor array to monitor temperature, humidity, and illumination, and is equipped with controllable actuators, including active cooling systems, dehumidifiers, and lighting grids, to replicate the combinatorial complexity of real supervisory control and data acquisition (SCADA) deployments. Experiments were conducted on a workstation with an Intel Xeon Gold CPU, 180GB RAM, and two NVIDIA Tesla V100 GPUs, leveraging Ollama and LangChain for model orchestration.

The system performance was evaluated across four progressive configurations to isolate the contribution of each component: Phase 1 (Baseline): Employing only LLM with RAG to establish a performance baseline. Phase 2 (MoA Enhanced): Incorporating the MoA architecture with Phase 1 to determine whether collective reasoning improves accuracy. Phase 3 (Validator Agent Enhanced): Coupling the Validator Agent directly with the Phase 1 baseline. This phase isolates the specific contribution of the Validator Agent's correction loop without the computational overhead of MoA. Phase 4 (Full System): The complete architecture combining RAG, MoA, and the Validator Agent to evaluate the maximum achievable performance. Relying exclusively on the pre-trained knowledge of the LLM model without the configuration of the RAG context results in a very low success rate; thus, it is omitted from the experimental comparisons. This underperformance is expected, as the model lacks access to specific device IDs (e.g., host A) and real-time states, leading to frequent hallucinations of non-existent devices.

To evaluate the performance of the LLM-based equipment control system, with an explicit focus on the Validator Agent's contribution to accuracy and robustness within the AIoT factory host control scenario, a multi-faceted evaluation methodology was employed. This involved assessing task completion rate (TCR), LLM suggestion accuracy (LSA), Validator Agent correction effectiveness (VACE), final execution accuracy (FEA), and response time (RT). It also considered some metrics, such as command processing throughput (CPT), latency variation under load (LVL), false positive rate (FPR), and false negative rate (FNR). These key metrics are described as follows:

(1) TCR: The proportion of user commands that successfully executed all intended actions on the target host devices. $N_{success\_executed\_tasks}$ denotes the number of tasks that were successfully executed, and $N_{total\_executed\_tasks}$ represents the total number of executed tasks.

$$\text{TCR} = \frac{N_{success\_executed\_tasks}}{N_{total\_executed\_tasks}} \times 100\% \tag{8}$$

(2) LSA: The percentage of textual suggestions generated by the LLM that correctly identify the target devices, intended actions, and associated parameters derived from the RAG context. $N_{correct\_llm\_suggestions}$ denotes the number of correct suggestions generated by the LLM, and $N_{total\_llm\_suggestions}$ represents the total number of suggestions generated by the LLM.

$$\text{LSA} = \frac{N_{correct\_llm\_suggestions}}{N_{total\_llm\_suggestions}} \times 100\% \tag{9}$$

(3) VACE: The percentage of incorrect LLM suggestions identified by the Validator Agent that can be successfully corrected. $N_{llm\_errors\_resolved}$ denotes the number of errors successfully resolved by the Validator Agent, and $N_{total\_llm\_errors\_detected}$ represents the total number of LLM errors detected by the Validator Agent.

$$\text{VACE} = \frac{N_{llm\_errors\_resolved}}{N_{total\_llm\_errors\_detected}} \times 100\% \tag{10}$$

(4) FEA: The percentage of individual device commands dispatched by the Validator Agent to the factory host equipment that are correct and result in the desired physical action. $N_{correct\_executed\_commands}$ denotes the number of correctly executed individual device commands, and $N_{total\_dispatched\_commands}$ represents the total dispatched individual device commands.

$$\text{FEA} = \frac{N_{correct\_executed\_commands}}{N_{total\_dispatched\_commands}} \times 100\% \tag{11}$$

(5) RT: The average time taken from the user issuing a command for factory host control to the Validator Agent confirming final actions or providing a definitive response. Measured in milliseconds.

(6) CPT: Defined as the number of complex user commands that the system can successfully process end-to-end per minute.

(7) LVL: Defined as the standard deviation of $RT_{complex}$ as the number of simultaneously simulated controlled hosts increases.

(8) FPR: Defined as valid, safe commands erroneously rejected by the Validator Agent.

(9) FNR: Defined as hazardous commands incorrectly authorized for execution.

To assess system robustness, the test employs a dataset containing 100 distinct control commands, ranging from explicit instructions to vague user intents. Some commands were manually formulated, while others were generated by LLMs. The dataset was tested three times, and the average was taken to obtain more reliable results. The dataset is structured into five progressive test suites (TS):

(1) TS 1: Fundamental single device control with explicit commands, focusing on basic command understanding.
(2) TS 2: Conditional single device control with clear commands, focusing on conditional logic processing.
(3) TS 3: Multiple device control with clear commands, focusing on handling multiple actions and targets.
(4) TS 4: Ambiguous commands, focusing on the system's disambiguation and rule enforcement.
(5) TS 5: Commands with fluctuating or discrepant data, focusing on error correction and secure execution.

3.2. Experimental results

The evaluation first examines the impact of the proposed architecture across varying LLM sizes. Table 1 presents the FEA for three distinct models. The baseline Phase 1 demonstrates a strong correlation between model size and accuracy, with the Llama-3.1 70B model achieving 85.7%, while the Phi-3 14B model lags at 65.3%. Phase 2 reduces this disparity by

introducing the MoA architecture, significantly enhancing the reasoning ability of smaller models. Phase 3 evaluates the impact of the Validator Agent without the overhead of MoA, and with the intervention of the Validator Agent, FEA accuracy surpassed 92%. Phase 4 shows that by applying MoA and the Validator Agent simultaneously, the accuracy of all models improves to over 96%. This confirms that for industrial safety, deterministic validation is more fundamental than improvements in probabilistic reasoning.

Table 1 Comparative FEA across different models

LLM model	Phase 1	Phase 2	Phase 3	Phase 4
Llama-3.1 70B	85.7%	89.2%	96.8%	99.6%
Mixtral 8x7B	78.9%	83.1%	94.5%	97.9%
Phi-3 14B	65.3%	79.6%	92.1%	96.5%

Table 2 provides an evaluation of the Llama-3.1 70B model. Unlike Table 1, which focuses on general accuracy, this table further delineates the accuracy profile. A key observation in Phase 3 is the discrepancy between LSA (75.4%) and FEA (96.8%). Since Phase 3 lacks the MoA layer, the raw LLM suggestions remain prone to errors. However, the elevated VACE of 96.5% compensates for this, effectively rectifying defective commands before execution. Comparing Phase 3 and Phase 4, the Full System exhibits a slightly lower FPR. This can be attributed to the MoA layer in Phase 4 providing higher-quality initial drafts (LSA 86.5%), thereby reducing the burden on the Validator Agent and minimizing cases where the Agent might over-correct a poorly phrased but valid command. Nevertheless, Phase 3's FNR of 0.5% demonstrates that it is already a highly secure solution for standard deployments.

The Full System attained a consolidated FEA of 99.6%, driven by VACE. These metrics indicate that nearly all potentially hazardous hallucinations generated by the LLM were successfully intercepted and corrected. The FPR is primarily due to the verification agent's strict adherence to security rules, which occasionally results in overcorrection and the rejection of some valid but anomalous indications. Specifically, the system satisfies the "fail-safe" design requirement, ensuring that the risk of executing a dangerous command is reduced to a negligible level.

Table 2 The key metrics performance of Llama-3.1 70B

Metric	Phase 1	Phase 2	Phase 3	Phase 4
TCR	92.5%	98.9%	99.1%	99.8%
LSA	75.1%	82.3%	75.4%	86.5%
VACE	N/A	N/A	96.5%	99.1%
FEA	85.7%	89.2%	96.8%	99.6%
FPR	N/A	N/A	2.1%	1.8%
FNR	9.7%	4.3%	0.5%	0.2%

This enhancement in accuracy and security entails a calculated operational trade-off. As depicted in Table 3, the average RT for the Llama-3.1 70B model increased by 2.89 times, from 1,375 milliseconds in Phase 1 to 3,615 milliseconds in Phase 4. Correspondingly, the computational overhead also increased fourfold, requiring an average of 1,280 tokens per transaction. After removing the MoA layer in Phase 3, the average RT decreased from 3,615 milliseconds to 1,650 milliseconds, representing a latency reduction of 54%. Despite this speed increase, the FEA remains high at 96.8%, as the Validator Agent successfully intercepts most safety violations defined by rigid rules.

By comparing the average RTs across the four discrete experimental phases, the marginal latency contribution of each architectural component can be effectively deduced. The results indicate that the Validator Agent is computationally lightweight, adding approximately 0.3 seconds to the process, whereas the primary latency arises from the MoA layer, approximately 1.9 seconds. Consequently, for industrial tasks requiring faster RTs, employing the Phase 3 configuration is a strategic choice, as it preserves the critical safety benefits of the Validator Agent while maintaining a swift RT, without incurring the heavy computational load of the MoA layer.

However, given that this compromise yields an absolute increase in execution accuracy and ensures operational safety, the additional computational cost is justified for industrial applications where the cost of failure is high. While this increase makes the system unsuitable for "hard real-time" applications that require millisecond-level response, it remains within acceptable limits for "soft real-time" supervisory control, such as SCADA or heating, ventilation, and air conditioning (HVAC) management, where the physical inertia of the equipment renders multi-second delays negligible.

Table 3 Average system RT comparison (milliseconds)

LLM model	Phase 1	Phase 2	Phase 3	Phase 4
Llama-3.1 70B	1375 ms	3312 ms	1650 ms	3615 ms
Mixtral 8x7B	1025 ms	2667 ms	1280 ms	3138 ms
Phi-3 14B	812 ms	2463 ms	1050 ms	2575 ms

The system's FEA was further tested under varying TSs using the Llama-3.1 70B model, as shown in Table 4. While all phases performed well on clear commands (TS1–TS3), the baseline Phase 1 faltered significantly under Ambiguous Commands (TS4) and Data Fluctuations (TS5). The Validator Agent effectively mitigated these deficits, recovering accuracy in TS4 to 91.6% and TS5 to 86.1%. In the Data Fluctuation (TS5) scenario, the RAG context conflicts with real-time sensors, and the accuracy of Phase 3 (83.9%) is significantly improved compared to Phase 2 (55.3%). This confirms that the agent's real-time lookup capability is the primary driver for handling dynamic data errors.

Nevertheless, the Ambiguous Commands (TS4) highlight the necessity of the MoA layer. Phase 3 (84.5%) struggles more than Phase 4 (91.6%) when interpreting vague intent. Without the collective intelligence of MoA to disambiguate the user's prompt before it reaches the validator, the single LLM in Phase 3 is more likely to generate a command that is semantically valid but contextually wrong, which the rule-based Validator Agent may simply reject rather than correct. Thus, Phase 4 remains the superior choice for handling highly ambiguous linguistic input. Although it can achieve acceptable accuracy with the correction, it is still insufficient for deployment in intelligent factory equipment operations requiring high accuracy.

The comparison between Phase 3 and Phase 4 can serve as a reference for the "accuracy-latency" trade-off necessitated for industrial applications. Hence, a flexible deployment strategy can be adopted: implement the Phase 3 architecture for standard operations to maximize throughput, and selectively activate the MoA layer only for high-risk or highly ambiguous tasks.

Table 4 FEA under different test suites using Llama-3.1 70B

Test suite	Phase 1	Phase 2	Phase 3	Phase 4
TS 1	97.9%	98.6%	99.6%	99.9%
TS 2	92.1%	94.3%	98.7%	99.8%
TS 3	86.7%	89.2%	97.2%	98.4%
TS 4	64.8%	73.2%	84.5%	91.6%
TS 5	47.9%	55.3%	83.9%	86.1%

Critical operational analysis reveals residual failure modes in complex scenarios. These failures primarily stemmed from three sources: persistent ambiguity, over-correction, and contextual conflicts. In cases of persistent ambiguity, the system triggered a "human-in-the-loop" protocol. While recorded as a task completion failure, this signifies "safety success," as the system prioritized inaction over guessing. Conversely, over-correction highlighted a flexibility gap where rigid safety rules blocked necessary emergency interventions. Furthermore, hardware-related issues, such as sensor drift or intermittent network connectivity, occasionally caused discrepancies between RAG-retrieved context and real-time validation checks, resulting in command rejection.

To assess scalability, CPT and LVL were analyzed, as shown in Fig. 2 and Fig. 3. Phase 1 exhibits the highest linear scalability, while Phase 3 closely follows the baseline, demonstrating robust throughput. This confirms that the Validator Agent's lightweight logic imposes minimal penalty on system capacity. In contrast, Phase 2 and Phase 4 exhibit early

saturation, with throughput dropping by nearly 50% compared to Phase 3 due to the heavy computational overhead of the multi-model aggregation. Phase 4 exhibits the highest LVL, approaching 30 seconds of standard deviation under extreme load. This instability stems from the correction loop logic of the Validator Agent combined with the token generation variance of the MoA layer.

Notably, Phase 3 shows moderate variation; while it is more stable than Phases 2 and 4, the Validator Agent's dynamic decision inherently introduces some latency fluctuations compared to Phase 1. This increased variability stems from the Validator Agent, whose processing time fluctuates significantly depending on whether a command passes immediately or triggers a time-consuming correction loop. As the number of concurrent simulated hosts exceeds 300, this variance risks exceeding standard SCADA timeout thresholds, suggesting a need for decentralized deployment in large-scale facilities.

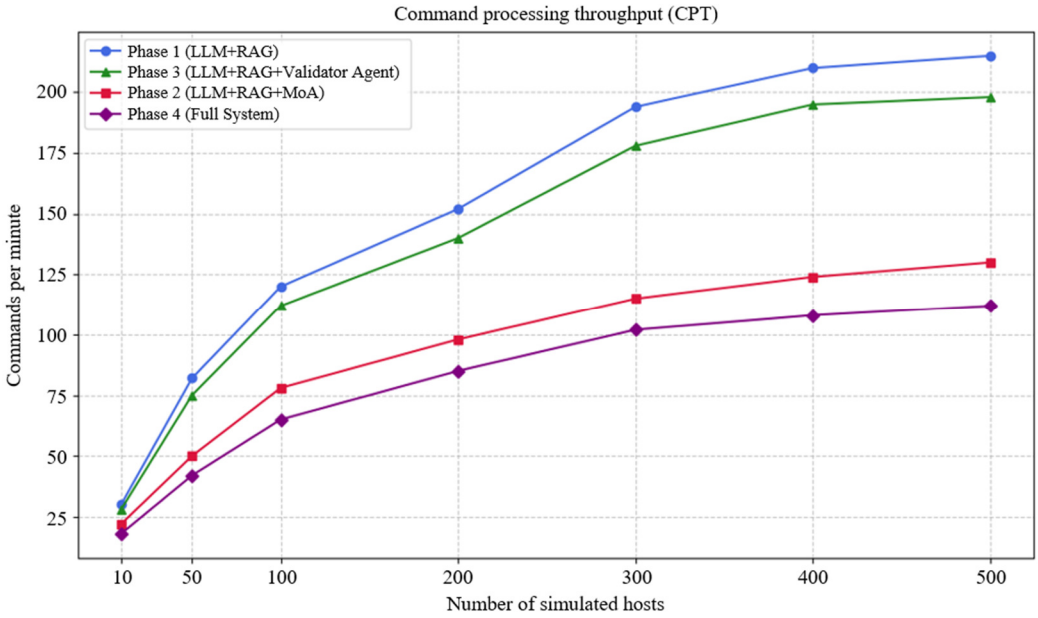

Fig. 2 Comparison of CPT

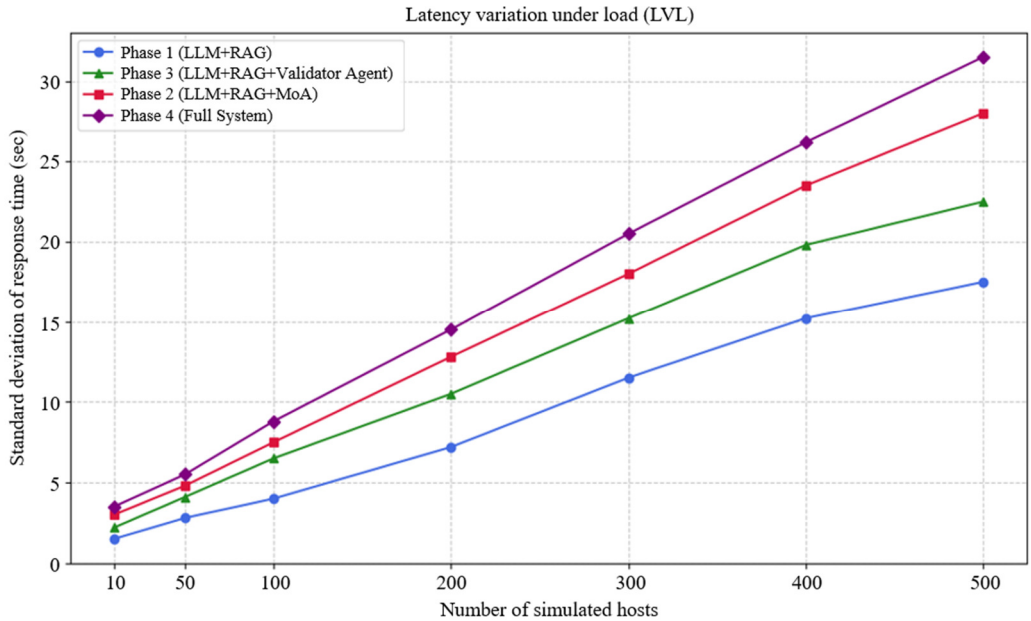

Fig. 3 Comparison of LVL

The overall system metrics comparison is shown in Fig. 4. The experimental results provide important references for the performance of the proposed LLM-based equipment control system and the critical role of each architectural component. The comprehensive evaluation indicates that while LLMs combined with RAG and MoA provide a powerful and flexible

foundation for natural language equipment control, an intelligent Validator Agent is required to achieve the high levels of accuracy, reliability, and safety necessary for practical deployment in AIoT factory environments. Although this system sacrifices raw execution speed and computing resources, it ensures the logical safety required for automated equipment control.

The configuration used in this study mandates a "safety-first" policy. However, in different industrial contexts, these weights should be recalibrated to match specific operational goals. For instance, for applications that focus on early fault warning rather than direct control, the weight for anomaly detection (w_4) should be increased; and in scenarios with fewer predefined constraints, the system must rely more on the LLM's reasoning confidence (w_1) and conflict resolution (w_5). Therefore, it is proposed to include a "weight calibration phase" when deploying this framework, in which the w_i values are adjusted using a validation dataset. This enables facility managers to optimize the trade-off between the FPR and the FNR according to their specific risk tolerance.

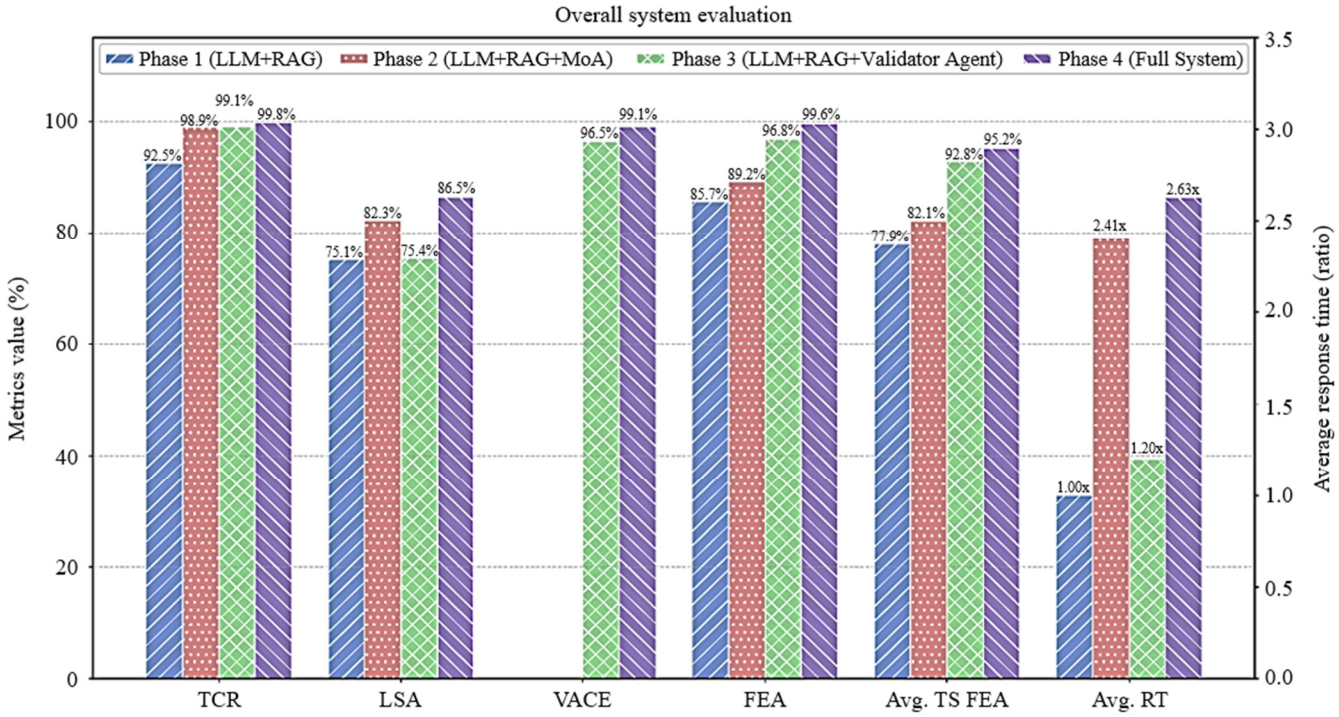

Fig. 4 The overall system metrics comparison

4. Conclusions

This study successfully developed a comprehensive framework for industrial equipment control by integrating RAG, MoA, and a Validator Agent, offering an effective and deterministic safety solution for LLM applications. The primary conclusions and findings can be summarized as follows:

(1) The integration of the RAG and MoA architectures achieves a superior balance between contextual awareness and reasoning robustness. By leveraging the collective intelligence of heterogeneous models, the system effectively resolved ambiguous user intents and mitigated single-model stochastic errors.

(2) The Validator Agent was the most critical component for operational safety. By actively validating and correcting commands based on predefined rules, the agent effectively neutralizes plausible but hazardous commands before execution.

(3) The Full System is currently optimally suited for "soft real-time" supervisory control, whereas a streamlined configuration is recommended for more time-sensitive operations to balance latency and safety.

This research presents a robust and viable solution for AI-driven smart manufacturing. However, future studies need to focus on optimizing real-time performance through model quantization, mitigating over-correction during sensor data fluctuations, and testing the framework with multimodal inputs to facilitate the widespread commercial deployment of this system in complex factory environments.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] N. Moenks, P. Penava, and R. Buettner, "A Systematic Literature Review of Large Language Model Applications in Industry," IEEE Access, vol. 13, pp. 160010-160033, 2025.

[2] X. Lin, W. Wang, Y. Li, S. Yang, F. Feng, Y. Wei, et al., "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.

[3] 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.

[4] A. Onan, A. H. Nasution, and T. Celikten, "Toward Reliable Annotation in Low-Resource NLP: A Mixture of Agents Framework and Multi-LLM Benchmarking," IEEE Access, vol. 13, pp. 211620-211644, 2025.

[5] S. S. Bavirthi, D. P. Sreya, and T. Poojitha, "Comparative Analysis of Mixture-of-Agents Models for Natural Language Inference with ANLI Data," Natural Language Processing Journal, vol. 11, article no. 100140, 2025.

[6] D. Kaur, S. Uslu, M. Durresi, 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, vol. 3, pp. 198-209, 2024.

[7] D. B. Acharya, K. Kuppan, and B. Divya, "Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey," IEEE Access, vol. 13, pp. 18912-18936, 2025.

[8] G. S. Sajja and S. R. Addula, "Automation Using Robots, Machine Learning, and Artificial Intelligence to Enhance Production and Quality," Second International Conference Computational and Characterization Techniques in Engineering & Sciences, pp. 1-4, 2024.

[9] S. R. Addula and A. K. Tyagi, "Future of Computer Vision and Industrial Robotics in Smart Manufacturing," Artificial Intelligence-Enabled Digital Twin for Smart Manufacturing, Hoboken, NJ: John Wiley & Sons, Inc., pp. 505-539, 2024.

[10] Y. Xia, N. Jazdi, J. Zhang, C. Shah, and M. Weyrich, "Control Industrial Automation System with Large Language Model Agents," IEEE 30th International Conference on Emerging Technologies and Factory Automation, pp. 1-8, 2025.

[11] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents," Proceedings of the 39th International Conference on Machine Learning, vol. 162, pp. 9118-9147, 2022.

[12] J. Wang, E. Shi, H. Hu, C. Ma, Y. Liu, X. Wang, et al., "Large Language Models for Robotics: Opportunities, Challenges, and Perspectives," Journal of Automation and Intelligence, vol. 4, no. 1, pp. 52-64, 2025.

[13] H. Fan, X. Liu, J. Y. H. Fuh, W. F. Lu, and B. Li, "Embodied Intelligence in Manufacturing: Leveraging Large Language Models for Autonomous Industrial Robotics," Journal of Intelligent Manufacturing, vol. 36, no. 2, pp. 1141-1157, 2025.

[14] S. Paul, L. Zhang, Y. Shen, 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.

[15] Z. Wang and H. Qin, "Intelligent Industrial Production Process Automatic Regulation System Based on LLM Agents," 5th International Conference on Artificial Intelligence and Electromechanical Automation, pp. 133-137, 2024.

[16] D. Rivkin, F. Hogan, A. Feriani, A. Konar, A. Sigal, X. Liu, et al., "AIoT Smart Home via Autonomous LLM Agents," IEEE Internet of Things Journal, vol. 12, no. 3, pp. 2458-2472, 2025.

[17] C. Sun, S. Huang, and D. Pompili, "LLM-Based Multi-Agent Decision-Making: Challenges and Future Directions," IEEE Robotics and Automation Letters, vol. 10, no. 6, pp. 5681-5688, 2025.

[18] Z. Liu, R. Zeng, D. Wang, G. Peng, X. Liu, Q. Liu, et al., "Agents4PLC: Automating Closed-Loop PLC Code Generation and Verification in Industrial Control Systems Using LLM-Based Agents," IEEE Transactions on Software Engineering, pp. 1-16, 2026.

[19] B. Galitsky and A. Rybalov, "Neuro-Symbolic Verification for Preventing LLM Hallucinations in Process Control," Processes, vol. 14, no. 2, article no. 322, 2026.

[20] F. C. Ogenyi, C. N. Ugwu, and O. P. C. Ugwu, "Securing the Future: AI-Driven Cybersecurity in the Age of Autonomous IoT," Frontiers in the Internet of Things, vol. 4, article no. 1658273, 2025.

[21] S. Xu, Z. Yan, C. Dai, and F. Wu, "MEGA-RAG: A Retrieval-Augmented Generation Framework with Multi-Evidence Guided Answer Refinement for Mitigating Hallucinations of LLMs in Public Health," Frontiers in Public Health, vol. 13, article no. 1635381, 2025.