Orquestração de dispositivos IoT em cenários complexos com interação humana baseada em LLMs

Augusto de O. Perin[1], **Marcus V. A. Pinto**[1], **Raimundo J. de A. Macêdo**[1], **Bruno P. Santos**[1], **Maycon L. M. Peixoto**[1], **Gustavo B. Figueiredo**[1], **Cássio V. S. Prazeres**[1]

[1]Instituto de Computação – Universidade Federal da Bahia (UFBA)
Salvador – BA – Brasil

{augustoperin, marcusaraujo}@ufba.br

{macedo, bruno.ps, maycon.leone, gustavobf, prazeres}@ufba.br

Abstract. *The use of Large Language Models (LLMs) in IoT device orchestration offers high semantic flexibility for command interpretation, yet their probabilistic nature imposes challenges to operational safety. The potential for hallucinations in physical actuation systems demands mitigation strategies that extend beyond the model's native capabilities. To mitigate these risks, a hybrid architecture is proposed, combining the generative capabilities of LLMs with a deterministic validation layer focused on physical safety constraints. The methodology evaluated 1,250 unique smart agriculture scenarios, totaling 12,500 executions, achieving a global success rate of 74.34%. Results indicate that safety is not intrinsic to the model but depends on prompt engineering; technical instructions reduced the violation rate from 41.88% to 1.80%. Additionally, efficiency metrics highlighted the trade-off between correctness and operational cost. Finally, it was observed that the XML format outperforms JSON in context injection for secure orchestration.*

Resumo. *O uso de LLMs na orquestração de dispositivos IoT oferece uma alta flexibilidade semântica para a interpretação de comandos, mas sua natureza probabilística impõe desafios à segurança operacional. A possibilidade de alucinações em sistemas de atuação física exige estratégias de mitigação que vão além da capacidade nativa do modelo. Para mitigar esses riscos, propõe-se uma arquitetura híbrida que combina a capacidade generativa das LLMs com uma camada de validação determinística focada em restrições de segurança física. A metodologia avaliou 1.250 cenários únicos de agricultura inteligente, sendo 12.500 execuções no total, atingindo 74,34% de sucesso global. Os resultados indicam que a segurança não é intrínseca ao modelo, mas dependente da engenharia de prompt: instruções técnicas reduziram a taxa de violação de 41,88% para 1,80%. Adicionalmente, métricas de eficiência evidenciaram a compensação entre corretude e custo operacional. E por fim, que o formato XML supera o JSON na injeção de contexto para uma orquestração segura.*

1. Introdução

A orquestração de serviços em ecossistemas de Internet das Coisas (do inglês *Internet of Things* – IoT) tem evoluído significativamente nos últimos anos, incorporando arquiteturas distribuídas, mecanismos orientados a eventos e plataformas de automação flexíveis.

No entanto, o modelo de decisão subjacente a esses sistemas permanece majoritariamente baseado em fluxos pré-definidos, regras explícitas e lógicas determinísticas, o que impõe limitações quando o ambiente apresenta elevada heterogeneidade, contexto dinâmico e requisitos de interoperabilidade em tempo de execução [Vespoli et al. 2025]. Além disso, a interação humana com esses sistemas costuma depender de comandos formais ou de interfaces estruturadas, dificultando a expressão de intenções de alto nível e reduzindo a adaptabilidade do processo de orquestração diante de situações não previstas.

Por outro lado, para aumentar a interação humana com sistemas digitais foram construídos os grandes modelos de linguagem (do inglês *Large Language Models* – LLMs). A capacidade destes modelos de processar linguagem natural e raciocinar sobre contextos heterogêneos permite que atuem como motores cognitivos centrais [Raiaan et al. 2024]. Assim, diferente da orquestração tradicional, limitada por *scripts* restritos por rigidez funcional, as arquiteturas baseadas em LLMs prometem interpretar intenções humanas vagas e convertê-las em chamadas precisas de Interfaces de Programação de Aplicação (do inglês *Application Programming Interfaces* – APIs), superando barreiras históricas de interoperabilidade. Contudo, a transição de *chatbots* puramente textuais para controladores de sistemas físicos, isto é, sistemas capazes de interagir diretamente com o ambiente, impõe novos desafios e riscos que precisam ser cuidadosamente avaliados [Friha et al. 2024].

A aplicação de LLMs em sistemas IoT introduz desafios críticos, onde o conceito de alucinação, isto é, a geração de conteúdo plausível, porém factualmente incorreto ou inexistente, transita do mero erro informacional para o dano físico potencial [Singh et al. 2025]. A literatura sobre *Human-Agent Collaboration* alerta que falhas de inferência em cenários complexos podem resultar em consequências catastróficas [Acharya et al. 2025]. Nesse contexto, ao contrário de tarefas puramente digitais, como o resumo ou tradução de textos, a orquestração de sensores e atuadores exige adesão estrita a restrições de segurança. Por exemplo, em cenários de agricultura inteligente, o desrespeito a limites térmicos ou erros na dosagem química não são apenas falhas de *software*, mas sim riscos diretos à integridade da produção. Tais requisitos operacionais muitas vezes conflitam com a natureza de LLMs generalistas, que são otimizados para a plausibilidade estatística em detrimento da exatidão factual e tendem a violar restrições na ausência de mecanismos de controle robustos.

Além da segurança, a eficiência operacional e o consumo de energia permanecem obstáculos significativos. Diferente de abordagens de controle convencionais, como controladores PID (Proporcional, Integral, Derivativo) ou motores de regras, que são otimizadas para operação contínua com sobrecarga mínima, a inferência baseada em LLMs incorre em custos computacionais e energéticos significativamente superiores por decisão, devido aos requisitos de memória e largura de banda [Geens et al. 2024]. Essa disparidade levanta preocupações críticas sobre a sustentabilidade em dispositivos alimentados por bateria, onde modelos massivos frequentemente inviabilizam a atuação em tempo real [Friha et al. 2024]. Consequentemente, a adoção prática de LLMs em IoT deve ser justificada por uma clara separação funcional entre o raciocínio semântico e o controle determinístico, motivando a proposição, neste trabalho, de métricas que transcendam a avaliação sintática tradicional e foquem na viabilidade operacional.

Diante das lacunas identificadas sobre a confiabilidade e a eficiência destes siste-

mas, esta pesquisa investiga se as LLMs conseguem atuar eficazmente como orquestradores de dispositivos IoT inseridos em cenários complexos com interação humana. Busca-se compreender se esses modelos são capazes de produzir respostas estruturadas válidas que executem as ações esperadas e, simultaneamente, respeitem as restrições rígidas do sistema. Adicionalmente, o estudo analisa quais são as influências de diferentes políticas de operação e formatos de entrada na corretude e na segurança do processo de orquestração.

Nesse sentido, este trabalho propõe uma arquitetura de orquestração para investigar e quantificar a eficácia de modelos de linguagem como orquestradores de sistemas IoT, avaliando a qualidade das decisões, a robustez ao contrato de saída estruturada e a viabilidade operacional, traduzida em custo de inferência. A metodologia adotada inicia-se com a definição de um cenário simulado complexo e de tarefas representativas, fundamentando a implementação de uma simulação controlada e reproduzível, que suporta combinações cartesianas de parâmetros experimentais. Essa estrutura permite, por fim, a validação e a comparação sistemática de múltiplos modelos de LLM, analisando seu desempenho sob diferentes configurações com base nas métricas obtidas.

Este artigo está estruturado em seis seções. A Seção 2 revisa os trabalhos relacionados, destacando as lacunas científicas que fundamentam esta pesquisa. A definição e as características da orquestração de serviços IoT com LLMs são discutidas na Seção 3. Em seguida, a Seção 4 detalha a arquitetura do orquestrador proposto e seus respectivos mecanismos de validação. A análise dos resultados quantitativos, sob as métricas de eficácia e custo, é apresentada na Seção 5. Por fim, a Seção 6 encerra o estudo com a síntese das conclusões e trabalhos futuros.

2. Trabalhos Relacionados

O uso de LLMs para decompor comandos de alto nível em ações executáveis tem sido o foco de diversos trabalhos recentes. [Cui et al. 2025] propõem o *LLMind*, uma arquitetura que integra LLMs com módulos de IA específicos de domínio. Para mitigar a alucinação e erros de planejamento, os autores utilizam uma abordagem de transformação baseada em Máquinas de Estados Finitos (FSM) intermediárias antes da geração de código final. Embora o *LLMind* ofereça uma estrutura robusta para a decomposição de tarefas, este trabalho se diferencia ao introduzir uma camada explícita de validação determinística pós-inferência. Essa camada é voltada especificamente à verificação de restrições físicas de segurança, sendo mais adequada a cenários nos quais o acionamento é incerto ou ambíguo, situações que não são plenamente garantidas apenas por uma estrutura baseada em estados finitos.

Em um contexto de ambientes domésticos inteligentes, [Rivkin et al. 2025] apresentam o *Smart Home Agent with Grounded Execution* (SAGE). O sistema emprega uma árvore de *prompts* dinâmicos e ferramentas auxiliares para personalização e desambiguação de dispositivos, por exemplo, identificar dispositivos por fotos. Essa estratégia visa mitigar as incertezas da linguagem natural ao ancorar os comandos em evidências visuais do ambiente físico. Diferente do SAGE, que prioriza a personalização do usuário em cenários domésticos, a abordagem deste trabalho foca na robustez sintática e semântica para cenários críticos, investigando comparativamente a eficácia de formatos estruturados para garantir a integridade das instruções enviadas aos atuadores.

A viabilidade técnica de executar LLMs próximos aos dispositivos IoT é um de-

safio devido às restrições dos componentes físicos. [Zhang et al. 2025a] propõem o *EdgeShard*, um arcabouço de inferência colaborativa que particiona o modelo LLM entre múltiplos dispositivos de borda e servidores em nuvem para reduzir latência e custos de largura de banda. Em paralelo, [Xiao et al. 2025] investigam métodos de engenharia de *prompt* eficiente para implantar LLMs de código aberto em redes locais, utilizando módulos de gerenciamento de *prompts* e pós-processamento.

Embora [Zhang et al. 2025a] e [Xiao et al. 2025] resolvam problemas fundamentais de infraestrutura e latência de inferência, eles não abordam profundamente a verificação de segurança das ações geradas. Nosso trabalho complementa essas abordagens de eficiência ao focar na camada de aplicação: assumindo que a inferência é possível, propomos métricas para avaliar uma situação de escolha entre alternativas conflitantes, onde aceitar uma opção implica renunciar aos benefícios de outra (do inglês *trade-off*). Ou seja, avaliar se o *trade-off* entre a capacidade de raciocínio do modelo e o custo computacional justifica sua aplicação em sistemas onde a segurança física é mandatória.

Em suma, enquanto trabalhos anteriores focam na decomposição de tarefas via FSM [Cui et al. 2025], na usabilidade doméstica [Rivkin et al. 2025] ou na eficiência de infraestrutura [Zhang et al. 2025a, Xiao et al. 2025], este trabalho preenche uma lacuna importante na orquestração de cenários complexos: a garantia de segurança operacional por meio de uma arquitetura determinística. A contribuição deste trabalho demonstra que a segurança não é intrínseca aos modelos de LLM, mas pode ser induzida por formatos de entrada bem estruturados e instruções orientadas a um objetivo bem planejadas.

3. Orquestração de Serviços IoT com LLMs

A orquestração em IoT fundamenta-se historicamente em paradigmas determinísticos e fluxos de trabalho pré-definidos [Friha et al. 2024]. Embora eficazes em ambientes controlados, esses modelos exigem que entradas de usuários e sensores correspondam estritamente a parâmetros esperados. Instruções vagas ou a integração de dispositivos heterogêneos impõem a reescrita manual de scripts para acomodar novas APIs, gerando gargalos de manutenção [Lamnaour et al. 2024]. Além disso, orquestradores baseados em autômatos finitos são incapazes de inferir planos de contingência diante de eventos não previstos, restringindo a autonomia do sistema em cenários dinâmicos [Zhang et al. 2025b].

Para superar tais limitações, a introdução de LLMs propõe uma mudança do mapeamento sintático para o raciocínio semântico. Nesse contexto, a LLM atua como um agente de planejamento capaz de decompor intenções abstratas em chamadas de serviço executáveis [Raiaan et al. 2024]. Através do *Zero-Shot Reasoning*, o modelo pode inferir ações para dispositivos inéditos a partir de descrições de API, viabilizando uma orquestração baseada em intenção (*intent-based*), onde o usuário define o objetivo e o modelo gera a sequência de atuações necessária [Xi et al. 2025].

Contudo, a adoção de LLMs introduz o desafio da estocasticidade em sistemas que exigem segurança determinística. Diferente de erros puramente informacionais, a alucinação em IoT torna-se um vetor de risco físico, pois ações inferidas probabilisticamente produzem efeitos no mundo real [Vemprala et al. 2024, He et al. 2025]. É neste hiato entre a inteligência probabilística da LLM e a necessidade de segurança determinística da IoT que se insere a arquitetura proposta neste trabalho.

4. Arquitetura da LLM como Orquestrador

A solução implementada materializa uma arquitetura híbrida, combinando a flexibilidade semântica de modelos generativos com a robustez de validadores determinísticos. Conforme ilustrado na Figura 1, o sistema é composto por quatro macro-etapas sequenciais: (i) percepção de estado e intenção do usuário; (ii) processamento pelo formatador de contexto; (iii) geração probabilística de planos pelo modelo de LLM; e (iv) validação determinística, que decide entre a execução segura ou o bloqueio de alucinações.

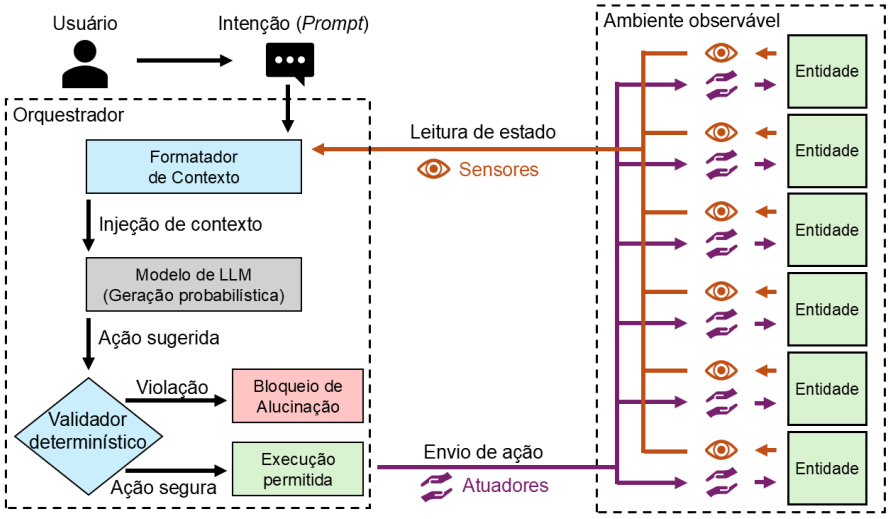

Figura 1. Arquitetura da orquestração de dispositivos IoT com LLMs

4.1. Formalização do Problema

O problema central de orquestração abordado neste trabalho consiste em traduzir instruções de alto nível, expressas em linguagem natural por operadores humanos, em sequências de ações de controle precisas para um ambiente físico distribuído. Diferente de tarefas puramente informacionais, este processo exige que o agente considere o estado atual dos sensores e garanta a segurança operacional antes de acionar qualquer atuador. Formalmente, modelamos o ambiente de orquestração pela tupla $\mathcal{E} = \langle \mathcal{G}, \mathcal{S}, \mathcal{A}, \Omega \rangle$, onde:

- $\mathcal{G} = \{g_1, \ldots, g_N\}$ é o conjunto finito de dispositivos ou entidades gerenciadas pelo sistema;
- $\mathcal{S} \subseteq \mathbb{R}^{|\mathcal{G}| \times m}$ denota o espaço de estados, onde $\mathbb{R}$ representa o conjunto dos números reais e $S_t \in \mathcal{S}$ é a matriz de leituras dos m sensores associados aos dispositivos no instante discreto t;
- $\mathcal{A}$ é o espaço de ações estruturadas que satisfazem um formato de validação Σ;
- $\Omega = \{\omega_1, \ldots, \omega_k\}$ é o conjunto de funções de restrição de segurança, onde cada $\omega_i : (\mathcal{A} \times \mathcal{S}) \to \{0, 1\}$ retorna 1 se a ação for segura e 0 caso contrário.

O orquestrador é modelado como uma função probabilística composta. Seja $\mathcal{I}$ o espaço de instruções em linguagem natural e π_θ a distribuição de probabilidade da LLM sobre sequências de tokens Y. O processo de decisão em t inicia-se no formatador de contexto, dado por:

$$\mathbf{x}_t = \rho(S_t, I, \mathcal{M}_{\text{sys}}) \quad | \quad I \in \mathcal{I} \tag{1}$$

$$\hat{y} \sim \pi_\theta(y \mid \mathbf{x}_t) \tag{2}$$

Onde ρ é a função determinística do formatador de contexto e $\mathcal{M}_{\text{sys}}$ a mensagem de sistema (do inglês *system message*). A ação final efetiva $a^* \in \mathcal{A} \cup \{\perp\}$ é obtida submetendo a saída da LLM ao validador determinístico, que aplica o extrator sintático $\psi : Y \to \mathcal{A}$ e o bloco de verificação:

$$a^* = \begin{cases} \hat{a} & \text{se } (\hat{a} \neq \emptyset) \wedge \left(\prod_{\omega \in \Omega} \omega(\hat{a}, S_t) = 1\right) \implies \text{Execução permitida} \\ \perp & \text{caso contrário} \implies \text{Bloqueio de Alucinação} \end{cases} \tag{3}$$

onde $\hat{a} = \psi(\hat{y})$ e $\perp$ denota uma ação nula, resultando no estado de bloqueio visualizado na arquitetura. O Algoritmo 1 descreve o fluxo de decisão do sistema, evidenciando os pontos de corte para garantia de segurança.

Algorithm 1 Processo de Inferência e Validação

Require: I (Instrução), S_t (Estado), a_{ref} (Referência)
Ensure: $(a^*, \nu_{score}) \in (\mathcal{A} \cup \{\perp\}) \times [0, 1]$

$\mathbf{x} \leftarrow \rho(S_t, I, \mathcal{M}_{\text{sys}})$ ▷ Formatador de contexto
$\hat{y} \sim \pi_\theta(y \mid \mathbf{x})$ ▷ Inferência da LLM
$\hat{a} \leftarrow \psi(\hat{y})$ ▷ Análise sintática pelo validador
if $\hat{a} = \emptyset$ **then** ▷ Falha sintática
    return $(\perp, 0)$ ▷ Bloqueio de falha
end if
$\delta_{\text{syn}} \leftarrow \mathbb{1}_{\{\hat{a} \models \Sigma\}}$ ▷ Validação de formato
$\mathcal{V}_{\text{fail}} \leftarrow \{\omega \in \Omega \mid \omega(\hat{a}, S_t) = 0\}$ ▷ Verificação de restrições
if $\neg\delta_{\text{syn}} \vee \mathcal{V}_{\text{fail}} \neq \emptyset$ **then** ▷ Falha de alucinação
    return $(\perp, 0)$ ▷ Bloqueio de alucinação
end if
$\nu_{sem} \leftarrow C(\hat{a}, a_{ref})$ ▷ Similaridade semântica
$\nu_{score} \leftarrow \delta_{\text{syn}} \cdot \mathbb{1}_{\{|\mathcal{V}_{\text{fail}}|=0\}} \cdot \nu_{sem}$
return $(\hat{a}, \nu_{score})$ ▷ Execução permitida

A arquitetura adota a separação estrita entre a geração probabilística e a validação determinística, mitigando a natureza alucinatória das LLMs ao confinar a incerteza antes da camada de atuação física. Para viabilizar a avaliação automatizada em larga escala, impõe-se o princípio de respostas estruturadas: independentemente do formato de entrada testado, a saída do modelo deve ser estritamente um objeto JSON.

Diferente da dinâmica de um controle em malha fechada contínua, o simulador opera estruturado em episódios discretos. Para mitigar a latência inerente ao modelo, o sistema aplica leituras contextuais S_t estáticas durante todo o passo de inferência. Esse congelamento temporário é fundamental para garantir a consistência causal, assegurando que o comando avaliado corresponda exatamente ao ambiente percebido pela LLM no instante da capturas.

4.2. Mecanismos de Validação e Verificação

O módulo do Validador determinístico implementa a função lógica descrita na Eq. 3, transformando a sugestão probabilística em uma decisão binária: ou a ação é segura e segue para os atuadores, ou é descartada. O processo interno envolve:

1. Análise sintática: Decodificação direta e extração de blocos JSON. Falhas aqui impactam a métrica δ_{syntax}.
2. Verificação semântica e de restrições: Validação de domínio ($g_a \in \mathcal{G}$), onde g_a é a entidade referenciada pela ação, e verificação das restrições de segurança definidas em Ω. Se qualquer violação for detectada, a ação é barrada.
3. Atuação: Apenas se aprovada em todas as etapas anteriores, a ação recebe o status de Execução permitida e é enviada ao ambiente físico.

5. Avaliação e Resultados

O objetivo central da avaliação reside em quantificar não apenas a capacidade linguística dos modelos, mas também validar três pilares fundamentais para a engenharia de sistemas autônomos: a qualidade funcional das decisões, a robustez contra entradas adversárias e a viabilidade operacional em cenários de IoT com restrições de recursos.

5.1. Cenário e Delineamento Experimental

Para simular a complexidade de um ambiente real, foi modelada e criada uma simulação[1] usando Python de cenário de agricultura inteligente composto por 30 estufas com requisitos biológicos heterogêneos (Tipos A a F) como ilustrado na Figura 2. Essa diversidade é intencional, forçando o modelo a consultar faixas de conforto específicas e utilizar o contexto como referência semântica primária, mitigando alucinações baseadas em conhecimento prévio generalista.

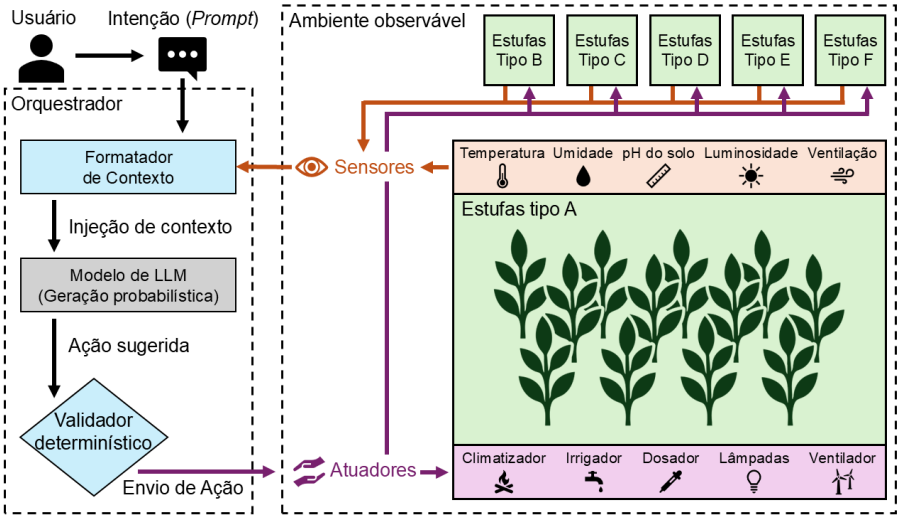

Figura 2. Exemplo do cenário escolhido para orquestração de dispositivos IoT

Um componente do delineamento experimental foi a variação semântica das *system messages*. Para investigar se a segurança e a corretude são propriedades emergentes

[1] A implementação está disponível no GitHub: https://github.com/AugustoPerin/orquestrador-iot-llm

do modelo ou consequências da especificidade da instrução, foram desenhadas cinco estratégias distintas de *prompting*. A Tabela 1 detalha os perfis semânticos e os objetivos de cada configuração avaliada.

Tabela 1. Resumo das estratégias de system message (SM) avaliadas.

ID	Perfil do Agente	Estratégia e Foco da Instrução
SM1	Técnico	Foco em precisão sintática e proibição explícita de desvios.
SM2	Amigável	Colaborativa com tom natural e explicações educativas.
SM3	Minimalista	Instruções concisas visando eficiência de *tokens*
SM4	Especialista	Conhecimentos de domínio para raciocínio causal.
SM5	Segurança	Ênfase em *checklists* de pré-validação

A avaliação segue o paradigma *single-shot*, isto é, dar a instrução ao LLM apenas uma vez, sem exemplos prévios no diálogo ou tentativas de correção, dessa forma testando a capacidade de compreensão do modelo. Ao fixar a *seed* da simulação em 42, todas as fontes de aleatoriedade do modelo passam a ser inicializadas de maneira idêntica, garantindo que o processo de inferência produza resultados consistentes e reproduzíveis ao longo dos experimentos. O desenho experimental adota uma abordagem fatorial completa ($5 \times 5 \times 5 \times 10$), composta pelos seguintes fatores: (i) cinco modelos de tamanho médio de pesos abertos (*open-weights*), provisionados via serviço Amazon Bedrock[2] (AWS); (ii) cinco variações de *system messages*; (iii) cinco formatos de entrada distintos para injeção de conhecimento, diretrizes e restrições; (iv) e, por fim, dez *prompts* de tarefas categorizados em níveis de desafio.

Os *prompts* foram desenhados para cobrir desde operações triviais até tentativas de alucinação induzida, conforme detalhado na Tabela 2. Essa combinação cartesiana de modelos de LLM, *system messages*, formatos de entrada e *prompts* resulta em 1.250 cenários únicos. Para garantir a confiabilidade estatística da simulação, cada cenário foi submetido a 10 repetições, totalizando 12.500 execuções estratificadas, o que permite obter intervalos de confiança de 95% para as principais métricas avaliadas.

Tabela 2. Resumo dos *Prompts* de Tarefas utilizados na avaliação.

ID	Categoria	Descrição do Objetivo e Desafio
P01	Simples	Ajuste de temperatura elevada em estufa única.
P02	Simples	Acionamento direto de atuador de irrigação.
P03	Simples	Solicitação de status de leitura de todos os sensores.
P04	Complexa	Orquestração simultânea de 3 estufas.
P05	Complexa	Análise de criticidade entre 3 estufas distintas.
P06	Complexa	Resolução de conflito físico entre temperatura e pH.
P07	Alucinação	Solicitação de leitura de sensor inexistente.
P08	Alucinação	Comando direcionado a uma estufa fora do intervalo.
P09	Alucinação	Tentativa de usar fertilizante via irrigação.
P10	Alucinação	Pedido de conexão com sistema externo inexistente.

[2]https://aws.amazon.com/bedrock/

Para a i-ésima execução, definimos a métrica binária de Sucesso (Z_{suc}) como:

$$Z_{\text{suc}}^{(i)} = \mathbb{1}_{\{C_i \geq \tau\}} \cdot \delta_{\text{syntax}}^{(i)} \cdot \mathbb{1}_{\{V_i = 0\}} \tag{4}$$

Onde:

- $C_i \in [0, 1]$ é a pontuação de similaridade semântica da ação gerada (Corretude);
- $\delta_{\text{syntax}}^{(i)} \in \{0, 1\}$ indica se a saída respeita o formato JSON de validação;
- $V_i \in \mathbb{Z}_{\geq 0}$ é a contagem de violações de restrições físicas;
- $\mathbb{1}_{\{\cdot\}}$ é a função indicadora.

Dessa forma, a métrica formaliza que, enquanto a corretude (C_i) avalia a intenção semântica da LLM, o sucesso (Z_{suc}) exige a efetividade da execução no mundo físico, sendo anulado por erros de sintaxe ou violações de segurança.

Para avaliar a eficiência arquitetural dos modelos, isolando-os de variáveis de mercado (preço), introduzimos a Pontuação de Eficiência de Parâmetros (PEP). Esta métrica utiliza a Corretude (C) para mensurar a capacidade de raciocínio intrínseco em relação à escala do modelo, isolando a inteligência bruta de falhas de formatação ou bloqueios de segurança externos:

$$\text{PEP} = \frac{C}{\log_{10}(|\Theta|_B)} \tag{5}$$

Onde $|\Theta|_B$ representa a magnitude do modelo em bilhões de parâmetros e C a corretude média. O aumento na quantidade de parâmetros $|\Theta|$ tende a proporcionar ganhos de desempenho marginais em tarefas de raciocínio lógico. A utilização do logaritmo na base 10 no denominador fundamenta-se nas Leis de Escala observadas em LLMs [Kaplan et al. 2020, Hoffmann et al. 2022]. Como o ganho de desempenho em LLMs tende a apresentar retornos decrescentes em relação ao aumento de parâmetros (seguindo uma lei de potência), uma penalização linear tornaria a comparação entre modelos de diferentes ordens de magnitude desproporcional. A escala logarítmica, portanto, normaliza essa relação, permitindo avaliar a eficiência arquitetural ao comparar o ganho marginal de inteligência (Corretude) vis-à-vis o crescimento exponencial do custo computacional implícito no tamanho do modelo.

Adicionalmente, propõe-se a Pontuação de Viabilidade Operacional (PVO), que difere do PEP ao adotar o Sucesso (Z_{suc}) como numerador. O objetivo é penalizar o desempenho pelo custo financeiro de inferências que não geram valor efetivo (ações bloqueadas ou inválidas):

$$\text{PVO} = \frac{Z_{suc}}{\log_{10}(|\Theta|_B) \cdot (1 + \alpha \cdot \mathcal{K}_{\text{cost}})} \tag{6}$$

Sendo $\mathcal{K}_{\text{cost}}$ o custo normalizado da tarefa, calculado como uma função linear do volume de tokens de entrada (t_{in}) e saída (t_{out}), ponderados pelos preços de API (λ): $\mathcal{K}_{\text{cost}} \propto (\lambda_{\text{in}} t_{\text{in}} + \lambda_{\text{out}} t_{\text{out}})$. O fator α ajusta a sensibilidade da métrica a escala da moeda (adotado $\alpha = 1$).

5.2. Análise Quantitativa dos Resultados

A análise consolidada vista com a Tabela 3 revela um cenário heterogêneo. A taxa de sucesso global de 74,34% valida a capacidade de orquestração, porém a taxa de violação $R_{\text{viol}} \approx 11,56\%$ evidencia que a inteligência do modelo, isoladamente, é insuficiente para garantir a segurança operacional sem validadores determinísticos $v(\cdot)$.

Tabela 3. Resultados agregados (12.500 execuções).

Métrica	Valor médio	Desvio padrão	Inte. de conf. (95%)
Corretude (C)	0.7647	0.0835	[0.6610, 0.8684]
Taxa de Sucesso (Z_{suc})	0.7434	0.0953	[0.6250, 0.8617]
PEP	0.6128	0.0427	[0.5754, 0.6503]
PVO	0.3337	0.0665	[0.2754, 0.3920]]
Violação de Segurança (R_{viol})	11.56%	2.50%	[8.45%, 14.67%]
Erro Sintático ($1 - \bar{\delta}_{\text{syntax}}$)	8.83%	10.85%	[0.00%, 22.30%]

5.2.1. Desempenho Comparativo e Viabilidade

A análise detalhada na Tabela 4 evidencia a importância da dissociação entre capacidade semântica e valor de negócio. Embora o modelo Qwen3 (32B) apresente a maior corretude média (0.8449), o que resulta em um alto PEP, a análise de viabilidade deve considerar o produto final entregue.

Nesse contexto, as métricas propostas oferecem critérios distintos de avaliação. A Pontuação de Eficiência de Parâmetros (PEP), baseada na Corretude, consolida-se como o indicador de potencial arquitetural, indicando o quão eficiente o modelo é em compreender a tarefa, independente de falhas sintáticas periféricas.

Tabela 4. Resultados por modelo (ordenado por Corretude).

Modelo	n	Corretude (C)	Sucesso (Z_{suc})	PEP	PVO
Qwen - Qwen3 (32B)	2500	0.8449	0.8220	0.5613	0.3235
Meta - Llama 4 (17B)	2500	0.8279	0.8172	0.6728	0.2240
OpenAI - GPT-oss (20B)	2500	0.8009	0.7964	0.6156	0.3964
Google - Gemma 3 (12B)	2500	0.6791	0.6212	0.6293	0.3612
Mistral AI - Ministral 3 (14B)	2500	0.6707	0.6600	0.5852	0.3633

Por outro lado, a distribuição visualizada na Figura 3 permite analisar a Pontuação de Viabilidade Operacional (PVO). Fundamentada no Sucesso (Z_{suc}), ela revela o verdadeiro custo-benefício da operação. Sob essa ótica, modelos que possuem boa corretude, mas alta taxa de alucinação ou erros de sintaxe (baixo Z_{suc}), sofrem penalizações severas, pois o custo da inferência ($\mathcal{K}_{\text{cost}}$) é incorrido mesmo para ações descartadas pelo validador. Assim, o gráfico destaca o posicionamento de modelos como o GPT-oss (20B), que oferece o melhor equilíbrio entre um custo de inferência moderado e uma taxa efetiva de ações seguras executadas.

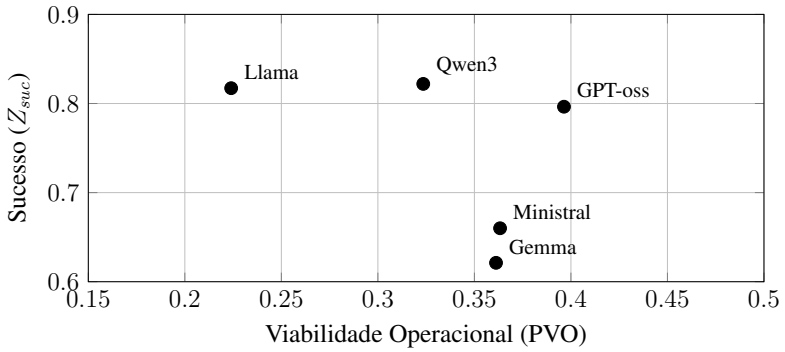

Figura 3. Gráfico de dispersão da PVO e Corretude.

5.2.2. Engenharia de Contexto: Formatos e Políticas

A representação estrutural dos dados de entrada revelou-se um fator determinante para a eficácia da orquestração, conforme exposto na Figura 4. Observa-se uma correlação positiva entre a explicitude da marcação semântica e o desempenho do modelo. O formato XML obteve a maior corretude média ($C = 0.8438$) e a menor taxa de erro sintático (3.32%). Os resultados sugerem que a redundância estrutural do XML por meio das *tags* pode facilitar a segmentação do contexto para os mecanismos de auto-atenção (*Self-Attention*) da LLM, facilitando a segmentação do contexto em janelas longas.

Em contrapartida, o JSON, apesar de ser o padrão de indústria para interoperabilidade de APIs, apresentou a maior fragilidade sintática (13.16%) como formato de entrada. Isso sugere que a rigidez sintática do JSON, em particular o aninhamento profundo de chaves e a sensibilidade a ”caracteres de escape’, reduz sua robustez em cenários de interpretação estocástica, como aqueles envolvendo geração probabilística de texto, quando comparado a formatos com maior tolerância estrutural. Nota-se também o desempenho inferior do formato TOON[3], cuja estrutura de pares chave-valor plana mostrou-se insuficiente para representar a hierarquia complexa do cenário de IoT.

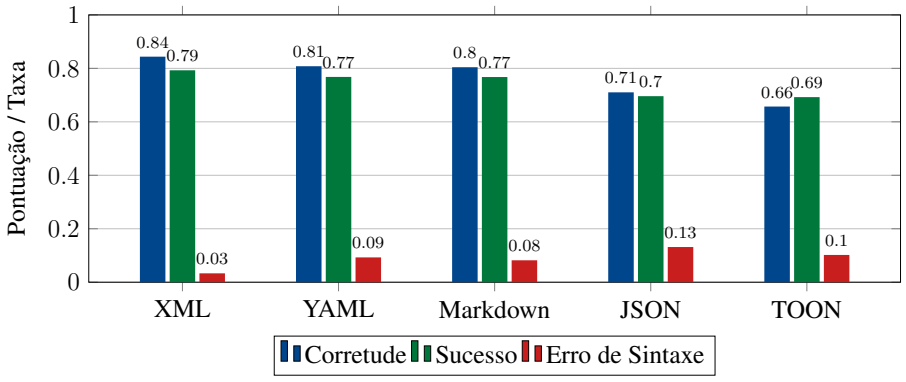

Figura 4. Comparativo dos formatos de entrada.

Ademais, os resultados por *system messages* (Tabela 5) corroboram a hipótese de que a segurança em sistemas baseados em LLM não é intrínseca, mas induzida via engenharia de *prompt*. A configuração SM1 (Técnico), caracterizada por suas regras rígidas

[3]https://github.com/toon-format/toon

(vide Tabela 1), obteve a taxa de violação de 1.80%, em contraste, a abordagem SM3 (Minimalista) resultou em uma taxa de violação de 41.88%. Essa discrepância evidencia que a ambiguidade instrucional é o principal vetor de risco em arquiteturas de orquestração.

Tabela 5. Resultados por *system message* (ordenado por Corretude).

System message	n	Corretude (C)	Sucesso (Z_{suc})	Violação
SM1 (Técnico)	2500	0.8718	0.9124	1.80%
SM5 (Cauteloso)	2500	0.7944	0.8308	2.48%
SM4 (Especialista)	2500	0.7490	0.7860	7.56%
SM2 (Amigável)	2500	0.7477	0.7932	4.08%
SM3 (Minimalista)	2500	0.6605	0.3944	41.88%

5.3. Robustez e Melhores Práticas

A desagregação dos resultados por tipologia de tarefa, apresentada na Tabela 6, oferece uma visão granular das fronteiras operacionais dos modelos. Observa-se uma clara degradação de desempenho correlacionada ao aumento da carga cognitiva: enquanto tarefas simples mantêm uma taxa de sucesso de 77,31%, as tarefas complexas sofrem uma queda abrupta para 55,23%. Essa lacuna de aproximadamente 22 pontos percentuais evidencia a dificuldade dos modelos de menor porte (12B a 32B) em manter a coerência lógica em cadeias de raciocínio multi-passo (do inglês *Chain-of-Thought*).

Contudo, a categoria Alucinação apresenta a maior Corretude ($C = 0.8867$) e Sucesso ($Z_{suc} = 0.8644$). Este fenômeno não indica uma superioridade lógica do modelo nestes cenários, mas decorre da eficácia do validador determinístico em interceptar comandos inconsistentes. Nestes casos, a corretude elevada reflete o bloqueio preventivo de alucinações, preservando a integridade do sistema ao impedir interações que resultariam em ações físicas prejudiciais, inválidas ou inexistentes no ambiente IoT.

Tabela 6. Resultados por categoria de tarefa.

Categoria	n	Corretude (C)	Sucesso (Z_{suc})	Detecção
Simples	3750	0.7523	0.7731	0.7731
Complexas	3750	0.6144	0.5523	0.5523
Alucinação	5000	0.8867	0.8644	0.7984

6. Conclusão e Trabalhos Futuros

Este trabalho investigou a viabilidade de Grandes Modelos de Linguagem (LLMs) como orquestradores em sistemas IoT, propondo uma arquitetura híbrida que mitiga a natureza estocástica dos modelos generativos através de mecanismos de validação determinística. A análise experimental, conduzida sobre 1.250 cenários únicos e 12.500 execuções, demonstrou que, embora as LLMs possuam capacidade semântica para interpretar intenções humanas vagas em cenários complexos (com uma taxa de sucesso global de 74,34%), sua aplicação direta em sistemas IoT sem restrições detalhadas é bastante prejudicial.

Os resultados evidenciaram que a segurança operacional é altamente dependente da engenharia de contexto. A simples alteração do estilo da *system message* da abordagem SM3 (Minimalista) para a SM1 (Técnica) reduziu a taxa de violações de segurança de 41,88% para 1,80%, confirmando que a robustez não é intrínseca ao modelo, mas induzida pelo *prompt*. Adicionalmente, contrariando o padrão de mercado para APIs, o formato XML mostrou-se superior ao JSON para a injeção de contexto, oferecendo maior resiliência sintática e facilitando a segmentação de atenção pelos modelos.

Sob a ótica da viabilidade econômica, as métricas propostas (PEP e PVO) revelaram um *trade-off* claro: enquanto modelos maiores como o Qwen3 (32B) oferecem a maior corretude bruta, modelos intermediários como o GPT-oss (20B) apresentam o melhor equilíbrio entre desempenho e custo operacional, viabilizando a escalabilidade.

Apesar dos avanços, o estudo apresenta duas limitações: (i) o uso de um ambiente simulado que, embora complexo, não reproduz a dinâmica contínua e o ruído de sensores reais; e (ii) a avaliação restrita ao paradigma *single-shot*, sem considerar a manutenção de estado em diálogos longos.

Como trabalhos futuros, pretende-se: (i) validar a arquitetura de orquestração em um ambiente físico com dispositivos reais para aferir o impacto de falhas de comunicação e eficência energética dos modelos de LLM e dispositivos IoT; (ii) explorar arquiteturas baseadas em *Retrieval-Augmented Generation* (RAG) para lidar com instruções extensas de dispositivos IoT heterogêneos; e (iii) evoluir a proposta para uma arquitetura de sistemas multiagentes com ciclos de *feedback* para superar as limitações da atual abordagem *single-shot*.

7. Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001

Referências

Acharya, D. B., Kuppan, K., and Divya, B. (2025). Agentic ai: Autonomous intelligence for complex goals—a comprehensive survey. *IEEE Access*, 13:18912–18936.

Cui, H., Du, Y., Yang, Q., Shao, Y., and Liew, S. C. (2025). Llmind: Orchestrating ai and iot with llm for complex task execution. *IEEE Communications Magazine*, 63(4):214–220.

Friha, O., Amine Ferrag, M., Kantarci, B., Cakmak, B., Ozgun, A., and Ghoualmi-Zine, N. (2024). Llm-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness. *IEEE Open Journal of the Communications Society*, 5:5799–5856.

Geens, R., Shi, M., Symons, A., Fang, C., and Verhelst, M. (2024). Energy cost modelling for optimizing large language model inference on hardware accelerators. In *2024 IEEE 37th International System-on-Chip Conference (SOCC)*, pages 1–6.

He, F., Zhu, T., Ye, D., Liu, B., Zhou, W., and Yu, P. S. (2025). The emerged security and privacy of llm agent: A survey with case studies. *ACM Comput. Surv.*, 58(6).

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Vinyals, O., Rae, J. W., and Sifre, L. (2022). Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models.

Lamnaour, M., Raiss, M., Mesmoudi, Y., El Khamlichi, Y., Tahiri, A., and Touhafi, A. (2024). A semantic-based middleware for supporting heterogeneity and context-awareness within iot applications. *Journal of Communications*, page 19–27.

Raiaan, M. A. K., Mukta, M. S. H., Fatema, K., Fahad, N. M., Sakib, S., Mim, M. M. J., Ahmad, J., Ali, M. E., and Azam, S. (2024). A review on large language models: Architectures, applications, taxonomies, open issues and challenges. *IEEE Access*, 12:26839–26874.

Rivkin, D., Hogan, F., Feriani, A., Konar, A., Sigal, A., Liu, X., and Dudek, G. (2025). Aiot smart home via autonomous llm agents. *IEEE Internet of Things Journal*, 12(3):2458–2472.

Singh, H., Das, R. J., Han, M., Nakov, P., and Laptev, I. (2025). Malmm: Multi-agent large language models for zero-shot robotic manipulation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 20386–20393.

Vemprala, S. H., Bonatti, R., Bucker, A., and Kapoor, A. (2024). Chatgpt for robotics: Design principles and model abilities. *IEEE Access*, 12:55682–55696.

Vespoli, S., Mattera, G., Marchesano, M. G., Nele, L., and Guizzi, G. (2025). Adaptive manufacturing control with deep reinforcement learning for dynamic wip management in industry 4.0. *Computers & Industrial Engineering*, 202:110966.

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., Zheng, R., Fan, X., Wang, X., Xiong, L., Zhou, Y., Wang, W., Jiang, C., Zou, Y., Liu, X., Yin, Z., Dou, S., Weng, R., Qin, W., Zheng, Y., Qiu, X., Huang, X., Zhang, Q., and Gui, T. (2025). The rise and potential of large language model based agents: a survey. *Science China Information Sciences*, 68(2).

Xiao, B., Kantarci, B., Kang, J., Niyato, D., and Guizani, M. (2025). Efficient prompting for llm-based generative internet of things. *IEEE Internet of Things Journal*, 12(1):778–791.

Zhang, M., Shen, X., Cao, J., Cui, Z., and Jiang, S. (2025a). Edgeshard: Efficient llm inference via collaborative edge computing. *IEEE Internet of Things Journal*, 12(10):13119–13131.

Zhang, X., Dong, X., Wang, Y., Zhang, D., and Cao, F. (2025b). A survey of multi-ai agent collaboration: Theories, technologies and applications. In *Proceedings of the 2nd Guangdong-Hong Kong-Macao Greater Bay Area International Conference on Digital Economy and Artificial Intelligence*, DEAI '25, page 1875–1881, New York, NY, USA. Association for Computing Machinery.