Controle Interoperável no Contínuo Computacional de IoT com Agentes de IA

Dener Silva[1], Alexandre Heideker[1], Ednaldo Ferreira [3], Carlos Kamienski[2], Reinaldo Bianchi[1]

[1]Centro Universitário FEI, São Bernardo do Campo, SP, Brazil

[2]Universidade Federal do ABC (UFABC), Santo André, SP, Brazil

[3]Empresa Brasileira de Pesquisa Agropecuária (Embrapa), São Carlos, SP, Brazil

{dener.silva, alexandre.heideker,
carlos.kamienski}@ufabc.edu.br
ednaldo.ferreira@embrapa.br
rbianchi@fei.edu.br

Abstract. *The growing adoption of smart agriculture as a response to increasing demands for productivity, product quality, and efficient use of natural resources requires increasingly advanced and resilient cyber-physical infrastructures. In this context, one of the main infrastructure and computing challenges is to ensure interoperability, low latency, and reliability in systems distributed across the IoT–edge–cloud continuum. The literature has addressed these challenges through cloud architectures, edge computing, digital twins, and IoT technologies, which remain largely fragmented and dependent on ad hoc integrations. This work proposes an interoperable multi-agent edge architecture based on Large Language Models (LLMs) and the Model Context Protocol (MCP) to integrate decision-making, monitoring, and actuation in a coordinated manner. Experimental results show that, compared with a structured controller, the LLM-based approach improves diagnostic flexibility in context-dependent scenarios, at the cost of higher latency and resource consumption.*

Resumo. *A crescente adoção da agricultura inteligente, como resposta às demandas por maior produtividade, qualidade do produto e uso eficiente de recursos naturais, exige infraestruturas ciberfísicas cada vez mais avançadas e resilientes. Nesse contexto, um dos principais desafios de infraestrutura e computação é garantir interoperabilidade, baixa latência e confiabilidade em sistemas distribuídos ao longo do contínuo IoT–borda–nuvem. A literatura tem abordado esses desafios por meio de arquiteturas em nuvem, computação de borda, gêmeos digitais e tecnologias de IoT, ainda marcadas pela fragmentação e integrações ad hoc. Este trabalho propõe uma arquitetura de borda com multiagentes interoperáveis, baseada em Modelos de Linguagem de Grande Escala (LLMs) e no Protocolo de Contexto de Modelo (MCP), para integrar decisão, monitoramento e atuação de forma coordenada. Os resultados experimentais mostram que, em comparação com um controlador estruturado, a abordagem com LLM amplia a flexibilidade diagnóstica em cenários dependentes de contexto, ao custo de maior latência e maior consumo de recursos.*

1. Introdução

As mudanças climáticas e a demanda por produtividade na agricultura impulsionam o investimento em tecnologia. Nesse cenário, a agricultura inteligente depende de infraestruturas ciberfísicas que integrem sensoriamento, conectividade e atuação heterogêneos em grandes áreas. Essa automação de alta precisão traz desafios de interoperabilidade, latência e confiabilidade, sobretudo quando decisões precisam ocorrer no próprio espaço de produção, como no controle de irrigação, na análise dos dados e no armazenamento em nuvem [Kong et al. 2022, Al-Dulaimy et al. 2024].

Um desafio central é prover infraestrutura resiliente diante da distribuição da computação ao longo do contínuo IoT–borda–nuvem. Arquiteturas centradas na nuvem simplificam o gerenciamento, mas podem introduzir latências inaceitáveis em controle em tempo real. A execução na borda melhora a responsividade, porém aumenta a complexidade arquitetural, especialmente quando múltiplas plataformas, protocolos e modelos de dados coexistem [Kong et al. 2022, Al-Dulaimy et al. 2024]. Essa complexidade é agravada pela fragmentação do ecossistema de IoT e pela diversidade de *stacks* de código aberto, resultando em integrações *ad hoc* e baixa portabilidade [Domínguez-Bolaño et al. 2022]. Nesse contexto, o projeto SMART [UFABC 2026] enfrenta esses desafios na agricultura sustentável, usando IoT e Inteligência Artificial (IA) para monitorar lavouras e otimizar irrigação e fertilização.

Trabalhos recentes indicam que aplicações robustas de IoT requerem suporte arquitetônico explícito para desempenho, implantação e observabilidade em todas as camadas [Zyrianoff et al. 2020]. Na agricultura inteligente, somam-se restrições de conectividade intermitente, dispositivos com recursos limitados e a coordenação de múltiplos *loops* de controle e monitoramento [Heideker et al. 2020]. Abordagens como gêmeos digitais (*Digital Twins*) conectam as visões física e cibernética da fazenda, mas sua implantação em infraestruturas Nuvem–Fog–Edge ainda enfrenta desafios de orquestração e integração [Kalyani et al. 2023].

Modelos de Linguagem de Grande Escala (*Large Language Models* – LLMs) são modelos neurais treinados para prever e/ou gerar sequências, aprendendo padrões sintáticos e semânticos a partir de grandes volumes de texto. A maioria adota a arquitetura Transformer [Vaswani et al. 2017] e produz representações que podem ser adaptadas a tarefas como classificação, extração de informação e geração controlada, tipicamente via pré-treinamento em larga escala e posterior especialização.

A adoção de ferramentas baseadas em agentes com LLMs renovou o interesse por padrões para descoberta de capacidades e invocação segura de ferramentas externas. O *Model Context Protocol* (MCP) tem sido discutido como mecanismo estruturado, orientado a sessões, para melhorar a interoperabilidade entre ferramentas e reduzir integrações personalizadas [LF Projects 2026]. Ainda assim, sua aplicação em circuitos reais de controle na agricultura inteligente exige modelagem cuidadosa, desenho do fluxo de dados e avaliação da latência de ponta a ponta sob cargas representativas.

Nesse contexto, este artigo investiga como agentes de IA integrados via MCP podem apoiar a coordenação operacional de aplicações agrícolas distribuídas ao longo do contínuo IoT–borda–nuvem. O objetivo não é defender a substituição irrestrita de controladores estruturados por LLMs, mas analisar em que medida uma arquitetura in-

teroperável baseada em agentes amplia a capacidade de diagnóstico e a integração entre serviços, bem como o custo computacional e temporal dessa escolha. Para isso, o trabalho aborda a coordenação interoperável de decisões agrícolas nesse contínuo, propõe uma arquitetura de borda com acesso controlado a ferramentas e tomada de decisão assistida por LLM, e a avalia em um *testbed* derivado do projeto SMART, comparando um controlador estruturado e uma abordagem com LLM em termos de latência, consumo de recursos e cobertura diagnóstica, além de discutir os limites práticos impostos pelo custo de inferência e pela concorrência em ambiente com recursos restritos.

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta motivação e trabalhos relacionados; a Seção 3 define o modelo do sistema e a formulação do problema; a Seção 4 descreve a metodologia experimental; e a Seção 5 discute os resultados.

2. Motivação e Trabalhos Relacionados

A agricultura inteligente tem avançado rapidamente ao combinar sensoriamento, conectividade e automação para apoiar decisões operacionais (por exemplo, irrigação, fertirrigação, controle de pragas e logística) sob forte variabilidade espacial e temporal. Contudo, a transformação digital em ambientes rurais segue limitada por desafios estruturais: heterogeneidade de dispositivos e protocolos, conectividade intermitente, restrições energéticas e de custos, necessidade de respostas em tempo real e preocupações com privacidade e soberania de dados. Esses fatores tornam insuficientes abordagens centradas apenas na nuvem e impulsionam arquiteturas distribuídas no contínuo IoT—borda—névoa—nuvem, nas quais partes do processamento e da tomada de decisão ocorrem próximas às fontes de dados, reduzindo a latência e a dependência de enlaces de longa distância [Al-Dulaimy et al. 2024, He et al. 2024].

Nesse cenário, a irrigação se destaca como um caso crítico, pois depende de respostas rápidas diante de variações de microclima, umidade do solo e demanda hídrica. Trabalhos recentes reforçam o papel da borda para decisões mais responsivas [Kasera and Acharjee 2024], enquanto estudos sobre interoperabilidade em plataformas agroalimentares destacam a necessidade de alinhar modelos de dados, serviços e ferramentas para reduzir acoplamento e aumentar reuso [Urdu et al. 2024, Kapitan et al. 2025]. Em paralelo, mecanismos de implantação no contínuo computacional [Oliveira et al. 2024], gêmeos digitais [Escribà-Gelonch et al. 2024] e protocolos de *tool calling*, como o MCP [LF Projects 2026], vêm sendo discutidos como meios para integrar serviços heterogêneos com maior controle e rastreabilidade. Nesse contexto, o diferencial deste trabalho está em combinar interoperabilidade operacional, integração agente-ferramenta via MCP e avaliação fim a fim do ciclo detectar–decidir–atuar em um cenário realista de agricultura inteligente [Kimovski et al. 2023, Urdu et al. 2024].

A tendência de consolidar inteligência no contínuo computacional também se observa na engenharia de implantação: em vez de mover todo o processamento para a nuvem, cresce a necessidade de mecanismos sistemáticos para decidir ”*onde* e o *como* ”os serviços IoT devem ser alocados, atualizados e escalados, considerando recursos disponíveis, metas de desempenho e condições dinâmicas do ambiente. Soluções recentes atacam esse problema ao propor modelos e ferramentas para implantação de aplicações IoT no contínuo, buscando reduzir a complexidade operacional e melhorar re-

petibilidade experimental [Oliveira et al. 2024]. Em paralelo, gêmeos digitais têm sido discutidos como um meio de orquestrar dados e atuar como "camada de integração" entre ativos físicos, modelos analíticos e processos de controle. Evidências recentes apontam oportunidades e desafios específicos para a agricultura, sobretudo no acoplamento entre a orquestração, a governança e a integração de múltiplas fontes de dados [Escribà-Gelonch et al. 2024].

Nos últimos anos, agentes baseados em LLMs têm sido explorados para automatizar tarefas de integração, operação e tomada de decisão em sistemas distribuídos. Entretanto, para que agentes sejam utilizáveis em ambientes reais (como borda agrícola), é necessário que sejam capazes de invocar ferramentas externas de forma controlada, interoperar com serviços heterogêneos e manter rastreabilidade das ações e dos dados utilizados. Nessa direção, protocolos e padrões de "*tool calling*" vêm emergindo para estruturar a interação entre modelos e ferramentas. Em particular, o MCP tem sido discutido como um mecanismo para padronizar a forma como ferramentas são expostas a modelos e agentes, promovendo a interoperabilidade entre componentes e reduzindo integrações *ad hoc* [LF Projects 2026]. Ao mesmo tempo, a literatura recente tem alertado para superfícies de ataque e riscos em ecossistemas de agentes, incluindo a exploração de ferramentas, injeções e falhas de encapsulamento, o que exige princípios de minimização de privilégios, isolamento e auditoria [Ferrag et al. 2025].

Diante desse panorama (contínuo computacional, automação responsiva, interoperabilidade e agentes com ferramentas), abre-se espaço para arquiteturas de borda que organizem componentes como agentes especializados, com integração padronizada e governança clara, preservando desempenho e segurança. A motivação central é reduzir os custos de integração e operação em agricultura inteligente, ao mesmo tempo em que se viabiliza a tomada de decisão mais rápida e resiliente, mantendo a portabilidade entre plataformas e a rastreabilidade das ações executadas pelos agentes. Nesse contexto, o diferencial deste trabalho está em combinar interoperabilidade operacional, integração agente-ferramenta via MCP e avaliação fim a fim do ciclo detectar–decidir–atuar em um cenário realista de agricultura inteligente [Kimovski et al. 2023, Urdu et al. 2024].

3. Modelagem do Problema

3.1. Caso de uso

A Figura 1 apresenta o cenário real utilizado no projeto SMART para integrar sensores e estações meteorológicas heterogêneas a uma plataforma IoT na nuvem, combinando uma camada de névoa (*fog/edge*), um barramento MQTT (*Message Queuing Telemetry Transport*) e um *pipeline* de normalização e orquestração com suporte a LLM. O objetivo é desacoplar os protocolos do campo dos serviços de nuvem, permitindo evoluir dispositivos e formatos sem reestruturar a plataforma.

No campo, dispositivos operam com diferentes tecnologias de comunicação e de representação de dados (por exemplo, LoRaWAN, serial e formatos derivados de SDI-12). A névoa atua como ponto de consolidação: dados que exigem coleta local (como estações conectadas via comunicação serial) são processados por um *Weather Station Processor* (WS Processor), que interpreta o formato da estação e encaminha as medições. Em paralelo, o componente de encaminhamento (PCK FW) organiza o tráfego de saída. No caso de LoRaWAN, os *uplinks* são recebidos pelo ChirpStack na nuvem e a saída

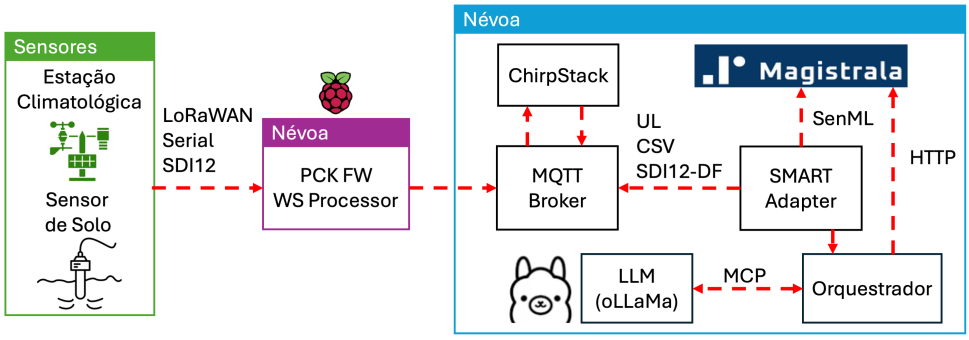

Figura 1. Cenário do projeto SMART: integração de dispositivos heterogêneos via névoa e MQTT, normalização em SenML e ingestão no Magistrala, com orquestração baseada em LLM e MCP.

decodificada é publicada no *broker* MQTT. Para os demais protocolos, as mensagens são encaminhadas diretamente ao mesmo *broker*. Assim, o MQTT funciona como ponto de convergência para telemetria heterogênea, simplificando a integração posterior.

O SMART Adapter, desenvolvido no projeto SMART, consome mensagens MQTT em diferentes formatos (por exemplo, UL decodificado, CSV e SDI12-DF) e utiliza um mecanismo baseado em *parsers* para identificar e traduzir cada *payload*. Esse desenho facilita a extensibilidade: para incorporar um novo protocolo ou dispositivo, tipicamente basta adicionar um *parser* com o mapeamento de campos. O SMART Adapter converte as medições em SenML[Jennings et al. 2018] e injeta os dados no Magistrala via HTTP, garantindo a consistência de nomes, unidades e *timestamps*. Além da conversão, o SMART Adapter concentra funções operacionais relevantes no ambiente real, como gerenciamento de credenciais e chaves, renovação de acesso, criptografia, suporte ao ciclo de vida de dispositivos, geração de alertas por *thresholds* e uma fila de mensagens mortas (*Dead Letter Queue* - DLQ) local para persistir mensagens quando a plataforma está indisponível, reenviando-as quando o serviço retorna. O SMART Adapter também pode gerar logs estruturados, adequados ao consumo por LLM, reduzindo o custo de preparação do contexto.

O Orquestrador é responsável por transformar eventos e contexto em decisões. Ele pode operar de forma proativa, consultando continuamente os dados/logs para identificar falhas e inconsistências, ou de forma reativa, acionado por alertas emitidos pelo SMART Adapter quando valores saem do padrão. Em ambos os casos, o orquestrador elabora um *prompt* com o contexto mínimo e consulta o LLM (executado via Ollama [1]). O MCP é utilizado como interface de capacidades entre o LLM e o orquestrador: o orquestrador expõe funções controladas (por exemplo, consultar medições recentes e registrar diagnósticos) e o LLM pode solicitar essas capacidades durante a análise. A comunicação com o Magistrala é realizada pelo próprio orquestrador via HTTP (isto é, o LLM não interage diretamente com o Magistrala). Essa separação mantém o LLM restrito a uma camada de ações bem definida e preserva a integração com a plataforma por meio de interfaces tradicionais.

[1] https://github.com/ollama/ollama

4. Metodologia e Experimentos

As Figuras 2(a) e (b) ilustram o ambiente (*test bed*) usado para reproduzir o cenário real do projeto SMART: uma plantação é dividida em 8 setores, e cada setor possui (i) uma sonda de solo com dois sensores (umidade e temperatura), (ii) um sensor de vazão e (iii) uma válvula para acionamento de irrigação; além disso, uma estação meteorológica fornece variáveis de referência, como chuva no momento e previsão para as próximas 6 horas (obtida a partir de serviços de previsão). O objetivo do *test bed* é avaliar o tempo de reação e o consumo de recursos computacionais em um *looping* de *detecção* → *decisão* → *atuação*.

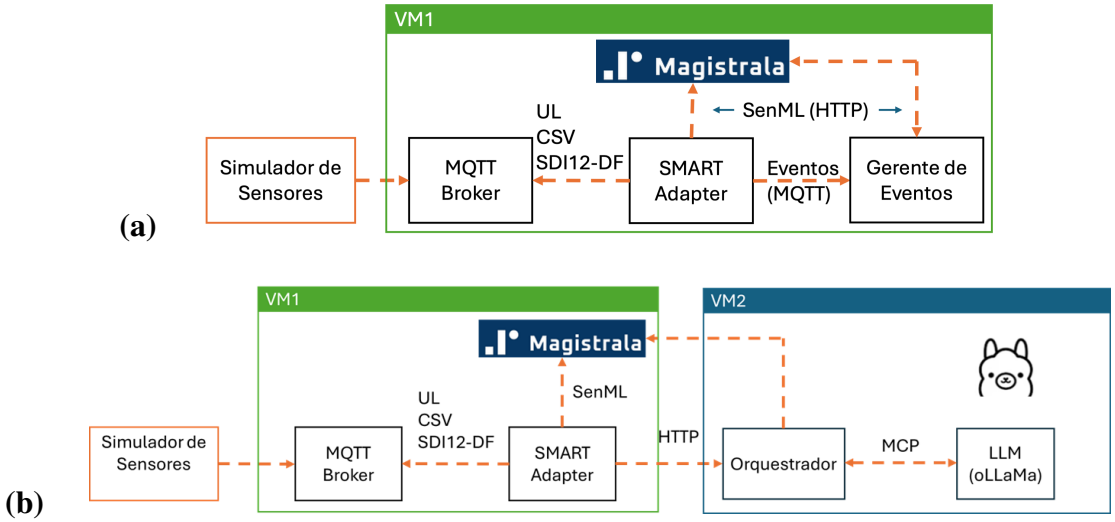

Figura 2. Configuração do *test bed*: (a) decisões determinísticas por *thresholds* (Gerente de Eventos); (b) decisões via Ollama/MCP para irrigação.

O Simulador de Sensores emula os dispositivos de campo e publica continuamente o estado de cada setor (oito conjuntos de leituras), seguindo a arquitetura do projeto: as mensagens convergem no *MQTT Broker*, são consumidas pelo *SMART Adapter* e, então, persistidas no *Magistrala* (via SenML sobre HTTP). Em paralelo, o simulador realiza *polling* periódico para verificar se há um comando de irrigação ativo em cada setor (por exemplo, uma entidade/campo `cmd.*`); nos experimentos, esse *polling* foi configurado a cada 10 segundos, de modo que o acionamento do atuador (liga/desliga irrigação) ocorre na próxima leitura do comando.

Os limiares de referência para classificar a umidade do solo definem os gatilhos do cenário de irrigação. Os parâmetros adotados nos experimentos seguem os valores obtidos em campo pelo projeto SMART: (0.01–0.20 → *Irrigation Alert*, 0.21–0.80 → *Normal*, e 0.81–1.00 → *Warning*). Leituras inválidas (por exemplo, 0.00) são tratadas como condição de manutenção. Além disso, o *test bed* utiliza a vazão para capturar inconsistências operacionais, como a presença de fluxo de água quando a umidade já está normalizada e não há comando ativo de irrigação, o que pode sugerir vazamento ou falha no desligamento do sistema. Esse cenário foi incluído por exigir a correlação entre múltiplos sinais do estado operacional, diferentemente do acionamento inicial da irrigação, que pode ser tratado por regras mais diretas baseadas em limiares.

Em todos os cenários foram coletadas métricas de atraso (em milissegundos) que

caracterizam o fluxo fim a fim, como o intervalo entre a publicação do dado pelo simulador e a detecção da condição que gera o evento (Injeção → Detecção), o tempo até o comando resultar no acionamento efetivo do irrigador (Detecção → Início) e o intervalo entre reconhecer que a umidade voltou ao aceitável e a interrupção da irrigação (Normalização → Desligamento). Além dos tempos, foram coletadas as métricas de *hardware* dos principais elementos da plataforma, como CPU e uso de memória RAM, para quantificar o custo computacional.

4.1. Avaliação do Tempo de Resposta

O primeiro conjunto de testes compara um controlador estruturado (baseline) com um controlador apoiado por LLM, utilizando Ollama local em modo CPU-only (Llama 3.2 padrão), a fim de avaliar o trade-off entre latência de resposta e flexibilidade de decisão. No *baseline*, todos os componentes executam na VM1: o SMART Adapter normaliza e injeta mensagens no Magistrala e, ao detectar violação de limiar (e.g., umidade abaixo do patamar), envia um alerta a um Gerente de Eventos que aplica regras simples combinando umidade, contexto meteorológico (chuva e previsão) e verificações básicas de segurança. Em geral, a irrigação é acionada quando a umidade está baixa e não há chuva/previsão relevante, e interrompida quando a lâmina de água calculada é aplicada e a umidade retorna à faixa normal.

No cenário com LLM, o *pipeline* de dados permanece na VM1 (MQTT *Broker*, SMART Adapter e Magistrala), enquanto o Orquestrador e o Ollama executam na VM2. O orquestrador recebe os alertas, consulta o Magistrala para obter contexto (últimas medições e comando atual), prepara um *prompt* estruturado e traduz a saída do modelo em uma ação, publicando `cmd=ON/OFF` no Magistrala. O MCP atua como interface de capacidades: o orquestrador expõe funções controladas de leitura e atuação, executa e registra as chamadas solicitadas pelo modelo, restringindo o escopo de ação e facilitando auditoria e rastreabilidade.

Em ambas as configurações, a atuação depende de *polling* a cada 10s, definindo o intervalo máximo entre decisão e acionamento. As VMs (VM1 e VM2) foram configuradas de forma idêntica (4 vCPUs a 3,2,GHz e 4,GB de RAM), permitindo comparação direta e isolando o impacto do custo de inferência e do posicionamento dos componentes.

4.2. Avaliação da Acurácia da Decisão

O segundo conjunto de testes compara o tempo de decisão e a acurácia entre as duas configurações do *testbed*, avaliando se a abordagem baseada em LLM amplia a cobertura diagnóstica em cenários que exigem correlação contextual, em comparação com o controlador estruturado. Em cada execução, o controlador deve decidir entre acionar a irrigação, mantê-la desligada e/ou emitir um diagnóstico. A acurácia é definida pela correspondência entre a decisão obtida e a esperada para o cenário.

Três cenários foram avaliados: (1) falha no sensor 1, em que apenas um setor apresenta leituras inválidas e a irrigação não deve ser iniciada; (2) chuva prevista, em que todos os setores estão com baixa umidade, mas a previsão de precipitação deve levar à decisão de não irrigar; e (3) vazamento, em que a umidade está normal, porém há fluxo contínuo sem comando ativo, o que deve ser identificado como anomalia e também não deve resultar em irrigação.

Assim, este conjunto de testes enfatiza decisões em condições adversas e avalia simultaneamente (i) a latência de decisão e (ii) a acurácia das respostas, permitindo comparar o comportamento do *baseline* estruturado e do LLM em cenários críticos e bem definidos. Os cenários simulados não visam reproduzir a complexidade agronômica completa da irrigação (p. ex., variáveis de solo, hidráulica, cultura e agrometeorologia em alta resolução), mas sim fornecer um *testbed* controlado para validar preliminarmente a proposta e isolar efeitos computacionais — especialmente o impacto da tomada de decisão assistida por LLM sobre desempenho, robustez e consistência operacional em sistemas IoT.

4.3. Avaliação do Estresse do LLM com Recursos Limitados

O terceiro conjunto de testes avalia apenas o cenário configurado com o motor de inferência utilizando o LLM em um ambiente com recursos limitados, buscando quantificar o impacto da concorrência sobre a latência, o consumo de recursos e a estabilidade das decisões. O objetivo não é comparar com o *baseline* utilizado no primeiro conjunto de experimentos, mas analisar como a concorrência no motor de inferência afeta o atraso da decisão, o consumo de CPU/RAM e a acurácia em situações críticas, refletindo um contexto realista em que o nó de inferência (VM2) pode atender a múltiplas requisições ou compartilhar recursos com outras tarefas.

Define-se concorrência como o número de requisições ao LLM ativas simultaneamente, em processamento ou aguardando resposta. O estresse foi induzido diretamente sobre o Ollama por um gerador de carga que emite chamadas HTTP concorrentes ao endpoint do modelo. O gerador opera com três parâmetros: CONCURRENCY (número de requisições paralelas), ALERT KB (tamanho aproximado do *payload* incluído no *prompt*) e DURATION S (duração do estresse). Em cada ciclo, ele dispara CONCURRENCY requisições em paralelo, com o mesmo *prompt* estruturado, e aguarda a conclusão do lote antes de iniciar o próximo, mantendo o sistema sob carga constante.

Três níveis de carga foram definidos, mantendo constantes o tamanho de *prompt* e a duração, variando apenas `CONCURRENCY`: baixa (`1`), média (`4`) e alta (`12`). Durante cada ciclo de teste, uma falha aleatória foi injetada para verificar se o LLM mantém decisões corretas. Em cada execução, foram medidos o atraso fim a fim até a decisão, a CPU e a RAM do ambiente de inferência, e a acurácia (decisão tomada versus decisão esperada), permitindo caracterizar a degradação sob estresse e discutir limites práticos de uso de LLM em *loopings* de controle na borda, especialmente em configurações *CPU-only* e sob concorrência.

4.4. *Prompt* do LLM, Funções expostas e Integração com a plataforma

O *prompt* foi projetado para ser sucinto, estruturado e rastreável. Ele inclui um resumo das leituras relevantes (umidade e, quando aplicável, vazão), variáveis meteorológicas (chuva e previsão de curto prazo), o estado do comando atual (irrigação ligada/desligada) e limites do cenário (faixas de umidade e regras de segurança, como suspeita de vazamento quando há vazão com umidade normal). O *prompt* também restringe explicitamente o espaço de ações permitido, limitando a resposta do modelo a decisões como "ligar irrigação", "desligar irrigação" ou "emitir alerta diagnóstico".

A integração via MCP operacionaliza a decisão sem conceder acesso irrestrito do modelo à plataforma. O Orquestrador expõe ferramentas para consultar medições recen-

tes (API do Magistrala), obter o estado do comando e publicar atualizações (*endpoints* HTTP em SenML). O LLM produz a decisão e, quando necessário, solicita o uso dessas ferramentas; o orquestrador executa as chamadas correspondentes, valida os retornos e registra logs e *timestamps* para análise. Dessa forma, mesmo quando o LLM influencia a decisão, a ação final é executada por um componente estruturado (o Orquestrador), preservando o controle e a rastreabilidade do processo.

5. Resultados e Discussões

5.1. Modelo estruturado *vs.* LLM

Os resultados do primeiro conjunto de testes, ilustrados na Figura 3(a), mostram dois comportamentos distintos e coerentes com a arquitetura e com o mecanismo de atuação adotado. No *baseline*, as três métricas fim a fim permanecem próximas de 5s: Injeção→Detecção (5,79 s), Detecção→Início (4,96 s) e Normalização→Desligar (4,98 s). Essa estabilidade era esperada porque a atuação depende do *polling* periódico dos sensores (10 s), de modo que os tempos observados tendem a se concentrar em torno de uma fração desse ciclo; na prática, a decisão determinística é rápida e o gargalo dominante passa a ser o intervalo de leitura/atuação do atuador/sensor, e não o processamento do gerenciador de eventos.

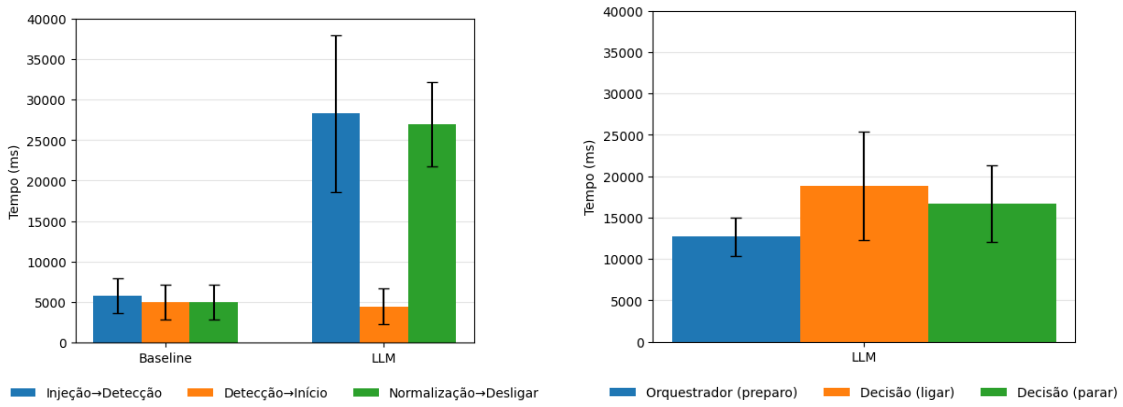

Figura 3. Conjunto de Teste 1: (a) Atraso observado entre o ciclo de testes. (b) Detalhamento da latência no motor de inferência LLM.

Nos cenários com LLM, observa-se um custo adicional substancial nas etapas em que a decisão depende do modelo. A latência Injeção→Detecção cresce para 28,27 s, e a Figura 3(b) mostra tempos de decisão de 18,86 s para *ligar* e 16,67 s para *parar*, indicando que a inferência passa a dominar o ciclo. Ainda assim, Detecção→Início permanece em torno de ∼4,5–5 s, sugerindo que, após a decisão, o acionamento do atuador continua limitado pelo *polling* e se comporta de forma semelhante ao *baseline*. A acurácia diagnóstica foi de 100% neste conjunto (30 repetições), mostrando que, embora mais custosa em tempo, a abordagem baseada em LLM produziu ações consistentes e maior flexibilidade de decisão.

Em relação ao consumo de recursos, no *baseline*, apresentado na Figura 4o consumo concentra-se no *stack* do Magistrala: a CPU média sobe de 37,6% (sem carga)

para 73,5% sob 10 msg/s, enquanto a memória permanece relativamente estável (aprox. 1,7–1,8 GiB). O SMART Adapter apresenta custo baixo em ambos os casos (cerca de 3% de CPU e dezenas de MiB de RAM), indicando que *parsing*/conversão não é o principal gargalo. No cenário com LLM, a carga adicional desloca-se para a VM do modelo: o Ollama domina o uso de recursos (aprox. 328–337% de CPU e 2,47–2,51 GiB de RAM), ao passo que orquestrador permanece leve (sub-1% de CPU e <50 MiB). Assim, o aumento de tráfego impacta principalmente a CPU do Magistrala, enquanto o custo extra da abordagem baseada em LLM está associado quase integralmente à inferência.

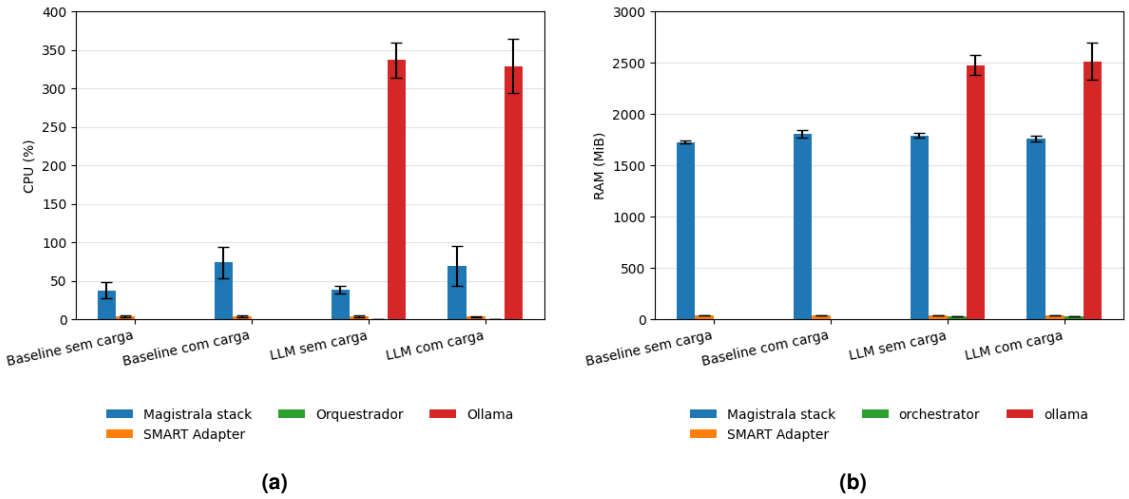

Figura 4. Uso de recursos no Conjunto de Testes 1: (a) Uso de CPU por componente; (b) Memória RAM por componente.

5.2. Decisão sob condições de falha e acurácia

Os cenários de falha (Figura 5) evidenciam diferenças entre regras estruturadas e inferência por LLM quando o diagnóstico exige contexto. No *baseline*, a lógica fixa atingiu 100% de acerto nos casos mais diretos (falha no sensor 1 e chuva prevista), mas não detectou o vazamento, indicando limitação diante de combinações não capturadas por limiares isolados (e.g., vazão anômala com umidade em faixa normal). O LLM manteve alta acurácia (100% em falha no sensor 1 e chuva prevista e 97% em vazamento, com 30 repetições), sugerindo maior capacidade de relacionar variáveis e identificar anomalias a partir do estado global. Em contrapartida, o custo temporal é maior: as decisões do LLM ocorrem em dezenas de segundos, refletindo a construção de contexto e a inferência. Observa-se, portanto, um *trade-off*: regras estruturadas oferecem menor latência e maior previsibilidade, enquanto o LLM amplia a cobertura diagnóstica em condições mais complexas, ao custo de maior tempo de decisão.

O perfil de recursos, ilustrado na Figura 6 permanece consistente com o Conjunto 1: no *baseline*, o consumo concentra-se no *stack* do Magistrala (cerca de 36–39% de CPU e ∼1,7–1,8 GiB de RAM), enquanto o SMART Adapter mantém custo reduzido (aprox. 3% de CPU e ∼33–36 MiB). Já no cenário com LLM, o principal acréscimo é a inferência: o Ollama domina o consumo (aprox. 305–338% de CPU e ∼2,5–2,6 GiB), ao passo que o orquestrador permanece leve (sub-1% de CPU e ∼28 MiB). O caso de vazamento no *baseline* é marcado como “ND” (não detectado), não contribuindo com

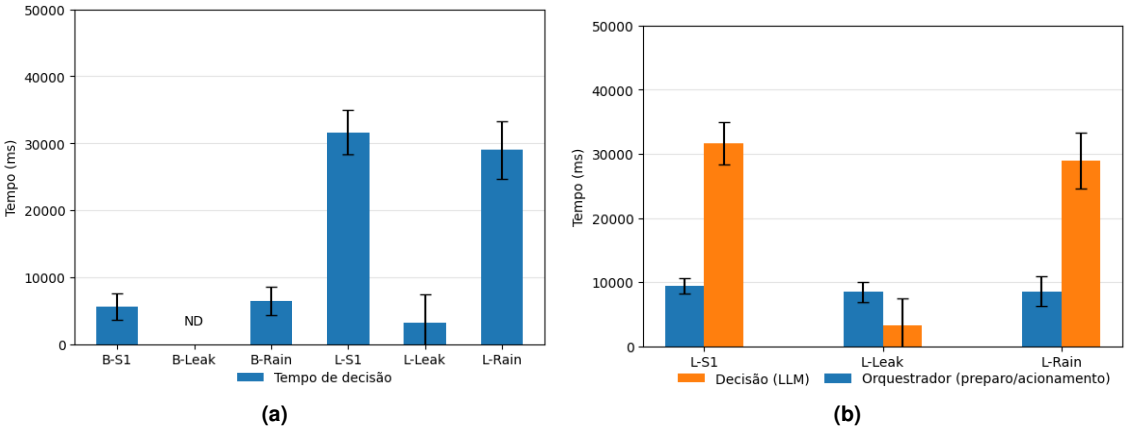

Figura 5. Conjunto 2 (falhas e acurácia). Eixo X: B-* (baseline) e L-* (LLM); S1 (falha sensor 1), Leak (vazamento) e Rain (chuva prevista). (a) Tempo fim-a-fim da decisão; ND = não detectado. (b) Decomposição no LLM: orquestrador e decisão.

carga adicional de decisão, reforçando que o ganho do LLM neste conjunto está associado à capacidade diagnóstica em situações de maior complexidade, e não a um aumento significativo de custo nos componentes de integração (fim a fim).

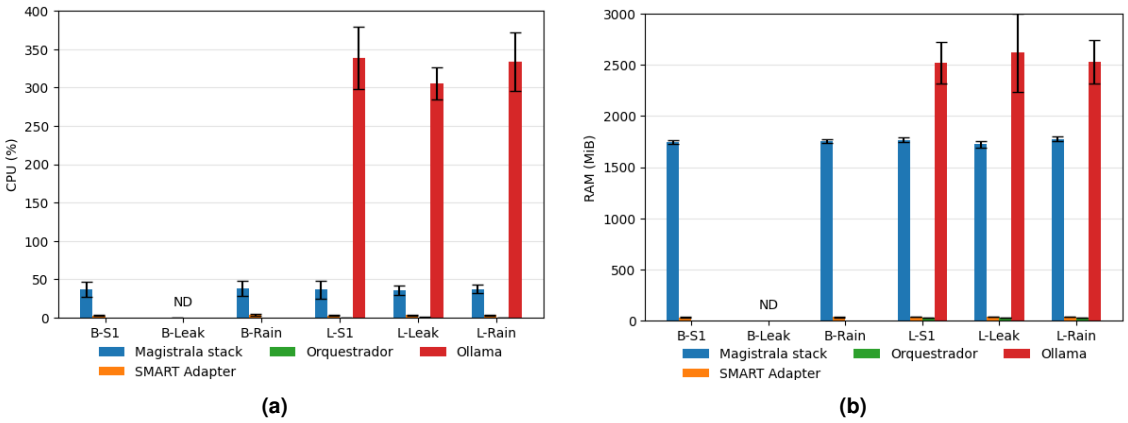

Figura 6. Uso de recursos no Conjunto 2. Siglas: B-* (baseline estruturado) e L-* (LLM); S1 (falha no sensor 1), Leak (vazamento) e Rain (chuva prevista): (a) CPU por componente; (b) Uso de memória RAM por componente.

5.3. Estresse do LLM sob concorrência

A análise dos resultados do terceiro conjunto de testes, apresentados na Figura 7, indica aumento progressivo do tempo de resposta conforme cresce o nível de concorrência aplicado ao modelo: 81,3 s (L1), 101,7 s (L4) e 126,0 s (L12), mantendo 100% de acurácia nos três níveis. Esse comportamento é consistente com um sistema em fila, no qual múltiplas requisições competem pelo mesmo recurso de inferência (CPU), elevando o tempo de espera até o atendimento. O intervalo de confiança mais alto em L12 é particularmente indicativo da dinâmica da fila: quando o evento observado chega e encontra a fila curta ou vazia, a requisição é atendida rapidamente; por outro lado, quando chega durante um período em que já há várias inferências pendentes, o tempo de espera cresce

substancialmente. Assim, a alternância entre instantes de fila "vazia" e "cheia" aumenta a dispersão das amostras e se traduz em maior intervalo de confiança, evidenciando que a concorrência afeta principalmente a latência, enquanto a qualidade da decisão permaneceu estável neste conjunto.

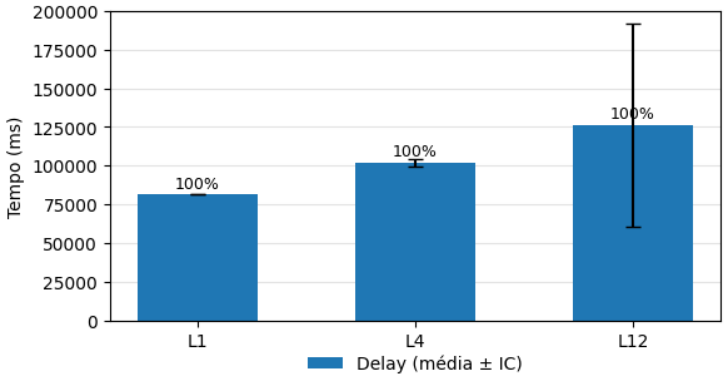

Figura 7. Impacto da concorrência no LLM. Siglas do eixo X indicam o nível de carga gerado por concorrência: L1 (baixa, 1 requisição concorrente), L4 (média, 4) e L12 (alta, 12).

Os resultados, ilustrados na Figura 8 indicam que o aumento de concorrência afeta principalmente o componente de inferência: o Ollama mantém consumo elevado de CPU (aprox. 328–373%) e memória estável em torno de 2,5 GiB, com tendência de maior CPU conforme a fila cresce. Em contraste, o *stack* do Magistrala e o SMART Adapter apresentam variações pequenas entre L1, L4 e L12 (cerca de 37–39% de CPU e ∼1,8 GiB para o Magistrala; ∼3% de CPU e dezenas de MiB para o Adapter), e o orquestrador permanece leve (sub-1% de CPU e ∼28 MiB). Assim, mesmo sob estresse por concorrência, o custo adicional concentra-se no modelo, enquanto os demais componentes preservam um perfil de recursos semelhante, sugerindo que o gargalo do cenário está na inferência.

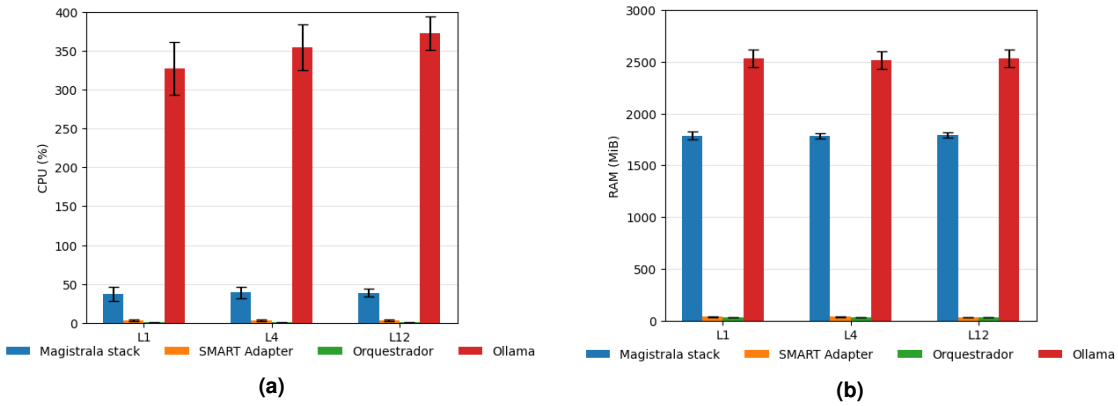

Figura 8. Uso de recursos no Conjunto 3. Eixo X indica o nível de carga: L1 (baixa, 1), L4 (média, 4) e L12 (alta, 12). (a) Uso de CPU por componente e (b) Uso de RAM por componente sob diferentes níveis de concorrência no LLM.

6. Conclusão e Trabalhos Futuros

Este trabalho teve como objetivo avaliar e comparar abordagens determinísticas e baseadas em LLM para a tomada de decisão em sistemas de agricultura inteligente distribuídos ao longo do contínuo IoT–borda–nuvem. Os resultados experimentais mostraram que, embora o modelo estruturado tenha apresentado melhor desempenho em termos de tempo de resposta, ele falhou em detectar corretamente alguns cenários relevantes, especialmente aqueles que exigiam a correlação de múltiplas variáveis ou a interpretação de condições de contorno mais complexas.

Em contraste, a abordagem baseada em LLM apresentou maior capacidade de diagnóstico nos cenários avaliados, especialmente quando a decisão exigia correlação contextual entre múltiplos sinais. Isso não implica substituir controladores estruturados em todos os casos, mas sugere uma arquitetura híbrida: mecanismos determinísticos seguem mais adequados para decisões rápidas e previsíveis, enquanto agentes apoiados por LLM podem atuar como camada complementar de diagnóstico e coordenação interoperável. A validação, entretanto, ainda é preliminar, limitada a um *testbed* controlado e a recursos restritos, indicando que o principal gargalo da arquitetura, no desenho atual, está no nó de inferência. Avaliações em campo e com maior diversidade de variáveis agronômicas permanecem como continuidade do trabalho.

Como trabalhos futuros, propõe-se integrar dados fisiológicos das plantas, imagens de satélite e previsões meteorológicas em tempo real ao processo de decisão, tornando a arquitetura mais próxima das condições reais e ampliando a capacidade dos agentes LLM de tomar decisões robustas e adaptativas no campo.

Agradecimentos

O presente trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) – processo nº 2024/15527-9.

Referências

Al-Dulaimy, A., Jansen, M., Johansson, B., Trivedi, A., Iosup, A., Ashjaei, M., Galletta, A., Kimovski, D., Prodan, R., Tserpes, K., Kousiouris, G., Giannakos, C., Brandic, I., Ali, N., Bondi, A. B., and Papadopoulos, A. V. (2024). The computing continuum: From iot to the cloud. *Internet of Things*, 27:101272.

Domínguez-Bolaño, T., Campos, O., Barral, V., Escudero, C. J., and García-Naya, J. A. (2022). An overview of iot architectures, technologies, and existing open-source projects. *Internet of Things*, 20:100626.

Escribà-Gelonch, M., Liang, S., van Schalkwyk, P., Fisk, I., Long, N. V. D., and Hessel, V. (2024). Digital twins in agriculture: orchestration and applications. *Journal of agricultural and food chemistry*, 72(19):10737–10752.

Ferrag, M. A., Tihanyi, N., Hamouda, D., Maglaras, L., Lakas, A., and Debbah, M. (2025). From prompt injections to protocol exploits: Threats in llm-powered ai agents workflows. *ICT Express*.

He, Q., Zhao, H., Feng, Y., Wang, Z., Ning, Z., and Luo, T. (2024). Edge computing-oriented smart agricultural supply chain mechanism with auction and fuzzy neural networks. *Journal of Cloud Computing*, 13(1):66.

Heideker, A., Ottolini, D., Zyrianoff, I., Neto, A. T., Salmon Cinotti, T., and Kamienski, C. (2020). Iot-based measurement for smart agriculture. In *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, pages 68–72.

Jennings, C. F., Shelby, Z., Arkko, J., Keränen, A., and Bormann, C. (2018). Sensor Measurement Lists (SenML). RFC 8428.

Kalyani, Y., Velasco Bermeo, N., and Collier, R. (2023). Digital twin deployment for smart agriculture in cloud-fog-edge infrastructure. *International Journal of Parallel, Emergent and Distributed Systems*.

Kapitan, D., Heddema, F., Dekker, A., Sieswerda, M., Verhoeff, B.-J., and Berg, M. (2025). Data interoperability in context: The importance of open-source implementations when choosing open standards. *J Med Internet Res*, 27:e66616.

Kasera, R. K. and Acharjee, T. (2024). A comprehensive iot edge based smart irrigation system for tomato cultivation. *Internet of Things*, 28:101356.

Kimovski, D., Saurabh, N., Jansen, M., Aral, A., Al-Dulaimy, A., Bondi, A. B., Galletta, A., Papadopoulos, A. V., Iosup, A., and Prodan, R. (2023). Beyond von neumann in the computing continuum: Architectures, applications, and future directions. *IEEE Internet Computing*, 28(3):6–16.

Kong, L., Tan, J., Huang, J., Chen, G., Wang, S., Jin, X., Zeng, P., Khan, M., and Das, S. K. (2022). Edge-computing-driven internet of things: A survey. *ACM Computing Surveys*, 55(8).

LF Projects, L. (2026). Model context protocol. `https://modelcontextprotocol.io/docs/getting-started/intro` Acesso em 31 jan. 2026.

Oliveira, F. B., Di Felice, M., and Kamienski, C. (2024). Iotdeploy: Deployment of iot smart applications over the computing continuum. *Internet of Things*, 28:101348.

UFABC (2026). Smart: Sustainable management of agriculture with the intelligent computing continuum. `https://smart.pesquisa.ufabc.edu.br/` Acesso em 31 jan. 2026.

Urdu, D., Berre, A. J., Sundmaeker, H., Rilling, S., Roussaki, I., Marguglio, A., Doolin, K., Zaborowski, P., Atkinson, R., Palma, R., et al. (2024). Aligning interoperability architectures for digital agri-food platforms. *Computers and Electronics in Agriculture*, 224:109194.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Zyrianoff, I., Heideker, A., Silva, D., Kleinschmidt, J. H., Soininen, J.-P., Salmon Cinotti, T., and Kamienski, C. (2020). Architecting and deploying iot smart applications: A performance–oriented approach. *Sensors*, 20(1):84.