SHINA: Sistema Hidropônico Inteligente Autônomo com Arquitetura IoT e Agentes Conversacionais

Bruno Matheus Costa[1], **Jorge Vinícius Monteiro Rosa**[1], **José Roberto Pazian Neto**[1], **Rafael Machado Tavares**[1], **Willgnner Ferreira Santos**[1,2] **e Alisson Rodrigues Alves**[1]

[1] Faculdade SENAI Fatesg
R. 227-A, 95 – Setor Leste Universitário, Goiânia – GO, 74610-155

[2] Faculdade SENAI Fatesg – Núcleo de Inteligência Artificial Aplicada – NIAA

{bruno98fev, jorgevinic04}@gmail.com
{rafaelmt3, zecapazian}@gmail.com
{willgnnerferreira, alissonalves.senai}@fieg.com.br

Abstract. *This paper presents SHINA, an autonomous hydroponic platform integrating IoT, Python microservices and generative AI. An ESP32 edge node acquires pH, electrical conductivity, temperature, humidity and water-level data and drives peristaltic pumps via a FastAPI backend with in-memory storage. A conversational AI layer via the Model Context Protocol (MCP) enables natural-language queries over real sensor data. Validated over 24 hours (Intel i7-13650HX, 16 GB DDR5), the system ingested 16,450 of 17,280 records (95.2 %) with median API latency of 10 ms (p99: 35 ms) and end-to-end latency of ≈180 ms, demonstrating the feasibility of low-cost IoT and conversational agents for precision agriculture.*

Resumo. *Este artigo apresenta o SHINA, uma plataforma autônoma de hidroponia que integra IoT, microsserviços em Python e inteligência artificial generativa. Um nó de borda ESP32 coleta pH, condutividade elétrica, temperatura, umidade e nível, acionando bombas via backend FastAPI com armazenamento em memória. Uma camada cognitiva via Model Context Protocol (MCP) permite consultas em linguagem natural sobre dados reais. Validado em 24 horas contínuas (Intel i7-13650HX, 16 GB DDR5), o sistema ingeriu 16.450 de 17.280 registros (95,2 %) com latência mediana de 10 ms (p99: 35 ms) e latência fim a fim de ≈180 ms, demonstrando a viabilidade de IoT de baixo custo para agricultura de precisão.*

1. Introdução

O mercado global de hidroponia cresce em média 12,4 % ao ano e, no Brasil, entre 1.500 e 3.000 hectares já adotam esse sistema, com redução de até 60 % no consumo de água em relação ao cultivo convencional [15]. O aumento da demanda global por alimentos, aliado à escassez de água e à competição por terras agricultáveis, tem impulsionado técnicas de cultivo mais eficientes, entre as quais a hidroponia se destaca pela elevada produtividade e pelo uso racional de recursos [5, 8]. Ao mesmo tempo, a difusão de microcontroladores de baixo custo e redes sem fio fomentou a aplicação de IoT em ambientes agrícolas, com protótipos que utilizam sensores conectados a plataformas *web* para monitoramento remoto em tempo real [8, 9, 10].

Nesse contexto, este trabalho propõe o SHINA (Sistema Hidropônico Inteligente Autônomo), uma arquitetura completa para monitoramento e automação de sistemas hidropônicos que combina hardware de baixo custo com práticas modernas de engenharia de software, integrando: nó de borda ESP32, *backend* RESTful em Python/FastAPI, armazenamento em memória Redis e camada cognitiva baseada em MCP para interação em linguagem natural. As principais contribuições são: (i) uma arquitetura modular em quatro camadas (borda, *middleware*, dados e cognitiva) que separa claramente as responsabilidades de coleta, controle, armazenamento e consulta; (ii) a demonstração da integração de agentes MCP em cenário hidropônico, permitindo consultas conversacionais ancoradas em dados reais; e (iii) a validação quantitativa da solução com métricas de latência e disponibilidade em ambiente controlado.

2. Trabalhos Relacionados

Um protótipo baseado em ESP32-WROOM para aquisição de pH, EC e temperatura, com transmissão HTTP periódica para interface *web*, foi desenvolvido em trabalho anterior [1]. O trabalho é economicamente viável (custo $<$ R$500), porém adota armazenamento relacional tradicional, sem separação entre camadas de ingestão, armazenamento e análise. O SHINA supera essa limitação ao estruturar um *backend* desacoplado em FastAPI com armazenamento em memória, possibilitando evolução modular independente.

Uma interface gráfica para monitoramento em tempo real de parâmetros hidropônicos acoplada a *backend* monolítico também foi proposta na literatura [2]. Embora forneça visualização clara das variáveis críticas, não adota microsserviços nem separação de camadas, comprometendo o escalonamento horizontal.

Modelos de IA para avaliação agronômica em hidroponia de alface foram propostos em trabalhos anteriores [3]. Apesar da inclusão de IA, o trabalho não integra agentes conversacionais com acesso dinâmico a dados reais. O SHINA preenche essa lacuna por meio do MCP, que ancora os modelos de linguagem diretamente nas séries temporais do sistema.

Outro trabalho priorizou a construção de um sistema economicamente acessível com automação básica de bombas e controle de nível [4]. A arquitetura é majoritariamente monolítica, sem camadas cognitivas ou separação de responsabilidades. Trabalhos anteriores de controle hidropônico, como controladores *fuzzy* do tipo Mamdani para regulação de pH e EC, demonstram respostas mais precisas que a simples histerese [3]; o SHINA adota histerese pela simplicidade e baixo custo, reservando controle avançado como evolução futura.

3. Fundamentação Teórica

3.1. Hidroponia e agricultura de precisão

A hidroponia consiste no cultivo de plantas em solução nutritiva, sem uso de solo, possibilitando redução significativa no consumo de água e fertilizantes, maior densidade de plantio e ciclos mais curtos para hortaliças folhosas [5, 6]. Para hortaliças de folha, como alface e rúcula, a faixa ideal de pH situa-se entre 5,5 e 7,0, enquanto a condutividade elétrica (EC) deve ser mantida entre 0,8 e 3,5 mS/cm, conforme a espécie e

Tabela 1. Comparação entre SHINA e trabalhos relacionados

Característica	SHINA	Silva	Santos	Fugino	Souza
ESP32 na borda	Sim	Sim	Sim	Não	Parcial
Sensores completos (pH/EC/T/U/N)	Sim	Parcial	Parcial	Parcial	Básico
Armazenamento em memória	Sim	Não	Não	Não	Não
Backend microsserviços	Sim	Não	Não	Não	Não
Agente IA conversacional	Sim	Não	Não	Não	Não
Medição de latência	Sim	Não	Não	Não	Não

a fase de crescimento; a temperatura da solução nutritiva afeta diretamente a absorção de nutrientes, com valores ótimos entre 18 e 26 °C [7]. Na agricultura de precisão, pequenas variações prolongadas fora dessas faixas podem reduzir a produtividade em até 30 % [5], o que justifica o emprego de sistemas automatizados de monitoramento com retroalimentação contínua [7, 3]. Sistemas hidropônicos de precisão caracterizam-se ainda pela rastreabilidade de parâmetros ao longo do ciclo de produção e pela capacidade de ajuste dinâmico de nutrientes — funcionalidades que o SHINA endereça por meio de sua arquitetura de quatro camadas.

3.2. Internet das Coisas em sistemas hidropônicos

A IoT tem viabilizado práticas de agricultura de precisão em pequena escala por meio de microcontroladores de baixo custo e conectividade sem fio. Reis (2020) [8] apresenta um sistema de monitoramento para cultivo de alface com ESP8266 e sensores ambientais; Domingos (2023) [9] descreve um sistema de monitoramento hidropônico com envio de dados para aplicação *web*. Essas soluções concentram-se na aquisição e visualização de dados, sem explorar camadas cognitivas, lacuna que o SHINA busca preencher. Pilati e Pereira (2025) [14] avançaram ao containerizar serviços de automação hidropônica com Docker, estratégia adotada integralmente no SHINA e que facilita implantações reproduzíveis em diferentes *hardware*.

3.3. Armazenamento em memória e persistência

Redis é um banco de dados chave-valor em memória amplamente utilizado em sistemas distribuídos que exigem baixa latência e alto *throughput* [11]. Sua capacidade de responder a operações em poucos milissegundos, usando estruturas como *hashes* e *Streams*, o torna adequado para filas de mensagens e séries temporais em sistemas de monitoramento. No SHINA, o Redis atua como camada operacional, conciliando respostas rápidas com persistência local viabilizada pelos volumes do Docker.

3.4. Model Context Protocol e IA generativa

O Model Context Protocol (MCP) é um padrão aberto que define como modelos de linguagem de grande porte (LLMs) podem acessar ferramentas externas e fontes de dados estruturadas em tempo de inferência, permitindo que agentes consultem bancos de dados, APIs ou serviços especializados durante a geração de respostas [13]. Esse mecanismo transforma um LLM genérico em um assistente com conhecimento contextual do sistema monitorado, capaz de responder sobre estado atual, histórico e tendências sem que o usuário precise conhecer a API subjacente. Santana et al. (2023) [12] mostram que

agentes de IA podem auxiliar pequenas fazendas hidropônicas na detecção de anomalias e recomendação de manejo; contudo, sua abordagem é baseada em modelos treinados *offline*, sem integração com dados em tempo real. O SHINA avança sobre esse estado da arte ao conectar o LLM diretamente às séries temporais do Redis via MCP, garantindo que as respostas reflitam o estado atual do sistema.

4. Metodologia

O código-fonte do SHINA está disponível em: `https://github.com/rafaelmtavares44/shina-mcp`

4.1. Arquitetura do sistema

A Figura 1 apresenta o protótipo físico montado para validação, com o ESP32, sensores e módulo relé organizados em bancada laboratorial.

Figura 1. Estrutura física do S.H.I.N.A.

A Figura 2 detalha a arquitetura lógica em quatro camadas, da coleta no campo até a análise cognitiva.

Camada de borda (1): Um microcontrolador ESP32-WROOM-32 é responsável pela leitura dos sensores de pH, condutividade elétrica, temperatura, umidade relativa do ar e nível do reservatório, além do acionamento de bombas peristálticas por meio de um módulo relé de quatro canais. Esse arranjo segue a tendência observada em projetos como o de Pilati e Pereira (2025) [14], que automatizam sistemas hidropônicos com microcontroladores e sensores de baixo custo.

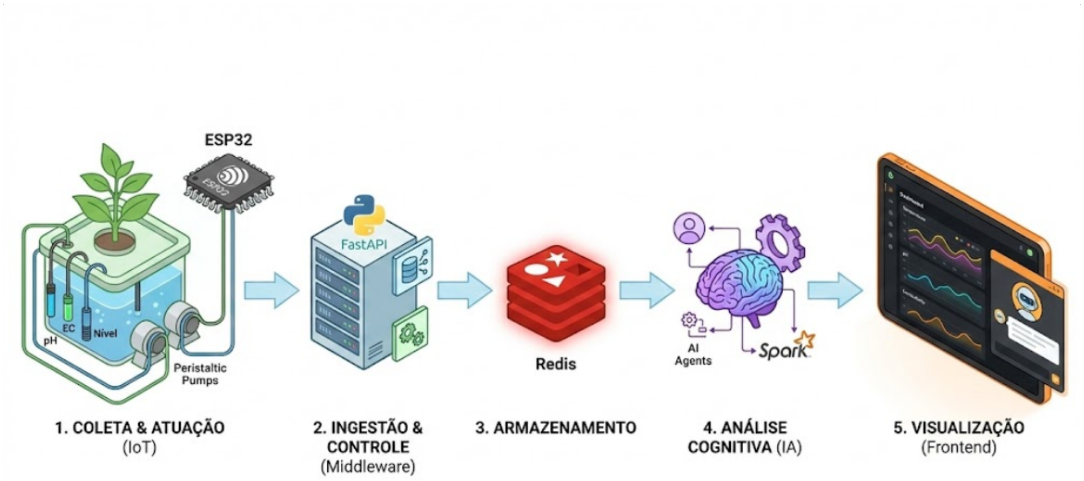

Figura 2. Arquitetura em camadas do SHINA, da coleta em campo até a análise cognitiva e visualização.

Camada de *middleware* (2): Uma API RESTful implementada em FastAPI recebe os dados enviados pelo ESP32 em formato JSON, valida o conteúdo, normaliza chaves e unidades e persiste as leituras no Redis. A API também expõe *endpoints* para consulta de configurações, como *setpoints* de pH e limites de nível, e para entrega de séries temporais ao *dashboard* web em Next.js.

Camada de dados (3): O Redis atua como banco operacional em memória, garantindo baixa latência nas operações de leitura e escrita, característica crítica para eventos como bloqueio de bomba diante de nível crítico. Os dados são persistidos em volume Docker, evitando perda de histórico em reinicializações.

Camada cognitiva (4): Um servidor MCP disponibiliza ferramentas de consulta somente leitura para modelos de linguagem, permitindo perguntas em linguagem natural sobre o estado do sistema. O agente não possui acesso a atuadores, consultando exclusivamente dados históricos e leituras atuais. Essa restrição é garantida pelos escopos definidos nas ferramentas MCP, que não expõem nenhum *endpoint* de controle.

4.2. Firmware do nó de borda

O *firmware* do ESP32 foi desenvolvido na IDE Arduino, utilizando bibliotecas para leitura de sensores analógicos e digitais, comunicação HTTP e manipulação de JSON. O ciclo principal consiste em: (i) realizar leituras dos sensores; (ii) aplicar médias móveis de curto prazo para reduzir ruído; (iii) montar um objeto JSON com *timestamp*, pH, EC, temperatura, umidade e nível; e (iv) enviar o pacote ao *endpoint* de ingestão da API FastAPI. Em caso de falha de comunicação, as leituras são armazenadas temporariamente em *buffer* e reenviadas assim que a conectividade é restabelecida [8, 9].

Os sensores de pH foram calibrados em três pontos com soluções padrão, permitindo ajustar uma curva linear entre a tensão de saída e o valor de pH. Os sensores de condutividade foram calibrados com soluções de referência de EC conhecida, e o sensor ultrassônico teve sua curva de distância ajustada ao formato geométrico do reservatório.

Um temporizador de hardware foi configurado para reinicializar o microcontrolador em caso de travamentos.

4.3. Backend e lógica de automação

A lógica de automação é centralizada na API FastAPI. A cada leitura recebida, o sistema avalia se as variáveis estão dentro das faixas desejadas: caso o pH esteja abaixo do limite inferior, é gerado um comando de dosagem de solução básica; valores de EC abaixo do limiar configurado disparam dosagem de nutrientes. Para evitar oscilações e acionamentos muito frequentes, são aplicadas bandas de histerese em torno dos *setpoints*. O nível do reservatório é monitorado continuamente e, ao atingir um valor crítico, qualquer acionamento de bomba é bloqueado por segurança [4].

O ESP32 consulta periodicamente um *endpoint* de configuração para obter o estado desejado dos atuadores e atualizações de parâmetros de controle, permitindo alterar regras de automação diretamente no *backend*, sem reprogramação do *firmware*.

4.4. Implantação e considerações de segurança

Toda a pilha de *software* do SHINA foi containerizada com Docker. Foram criados contêineres distintos para a API FastAPI, o servidor Redis (com volume persistente), o servidor MCP e o *dashboard* Next.js, orquestrados com Docker Compose em redes internas isoladas. A implantação inicial foi feita em servidor local, simulando o cenário de um laboratório ou pequeno produtor.

Do ponto de vista de segurança, o servidor MCP expõe exclusivamente ferramentas de consulta (somente leitura), sem acesso a nenhum *endpoint* de controle de atuadores. Autenticação na API e escopos granulares por ferramenta MCP são identificados como trabalho futuro prioritário, conforme discutido na Seção 6.

5. Resultados e Discussão

5.1. Ambiente de validação e desempenho de ingestão

Os experimentos foram conduzidos em servidor local com as seguintes especificações: processador Intel Core i7-13650HX (14 núcleos, até 4,9 GHz), 16 GB de RAM DDR5 a 4800 MT/s, SSD NVMe de 512 GB e conectividade Wi-Fi 6 (AX201, 802.11ax). O experimento de validação consistiu em 24 horas de operação contínua com leituras a cada 5 segundos, totalizando 17.280 registros esperados. Desses, **16.450 foram ingeridos com sucesso**, resultando em taxa de **95,2 %**. As 830 falhas restantes ocorreram predominantemente por instabilidade momentânea de rede Wi-Fi, com reconexão automática em até três tentativas pelo *firmware* do ESP32.

A latência de ingestão apresentou **mediana de 10 ms, p90 de 19 ms e p99 de 35 ms**, com a maior parte das requisições concentrada entre 7 e 15 ms, conforme evidenciado pela distribuição da Figura 3. A latência fim a fim ficou em aproximadamente **180 ms**. A CPU manteve utilização média **abaixo de 8 %** e o consumo de memória não ultrapassou **420 MB**, indicando adequação ao *hardware* de baixo custo.

Para reproduzir os resultados, basta clonar o repositório, instalar Docker Compose e executar `docker compose up`; o ESP32 deve ser programado com o *firmware* disponível em `/firmware`, com as credenciais Wi-Fi e o endereço IP do servidor ajustados em `config.h`.

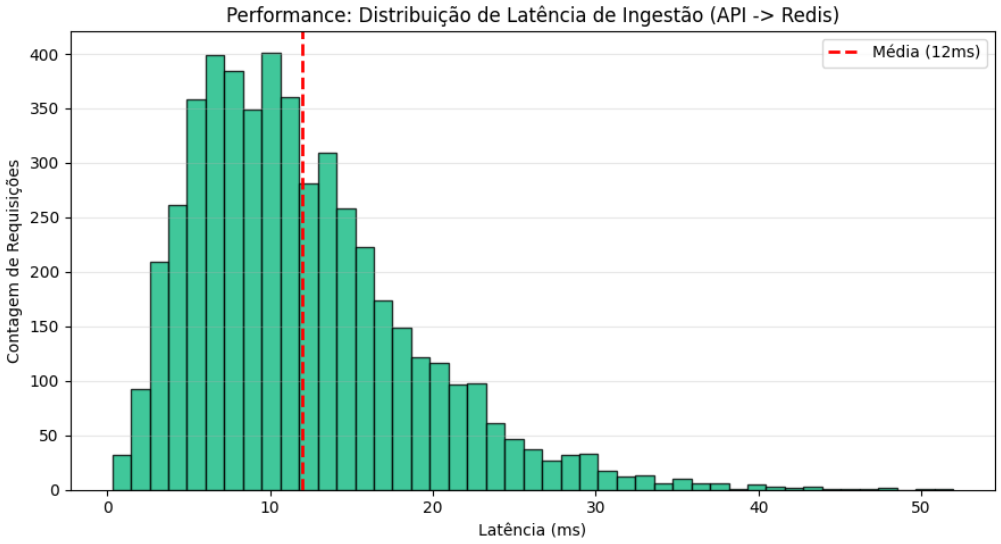

Figura 3. Distribuição da latência de ingestão (API → Redis) durante 24 horas de operação (média: 12 ms; p50: 10 ms; p90: 19 ms; p99: 35 ms).

5.2. Resultados agronômicos e testes de cenários de falha

Durante o experimento, sensores físicos foram conectados à solução nutritiva de um sistema hidropônico real cultivando alface em fase vegetativa, com os seguintes *setpoints*: pH entre 5,5 e 7,0; EC mínima de 310 µS/cm; temperatura da solução entre 18 e 28 °C; e nível crítico do reservatório em 1,5 L.

Os resultados obtidos foram: (a) **pH mantido entre 6,1 e 6,8** ao longo das 24 horas, sem acionamento de bomba de correção — indicando que a solução nutritiva inicial era adequada para a cultura; (b) **EC reduziu de 380 µS/cm para 285 µS/cm** nas primeiras 18 horas pelo consumo de nutrientes pelas plantas, disparando automaticamente a bomba peristáltica que elevou a EC para 338 µS/cm, comportamento esperado e consistente com a literatura [5, 7]; (c) **temperatura da solução variou entre 24,1 e 27,8 °C**, dentro da faixa ideal para alface; e (d) **umidade relativa do ar manteve-se entre 68 % e 75 %**, favorável à cultura.

Para avaliar as rotinas de segurança, eventos de queda abrupta de nível foram **injetados artificialmente** via API (cenários de teste controlados, não correspondentes a condições físicas reais). Em todos os casos, o *backend* identificou a condição de nível crítico e bloqueou imediatamente novos acionamentos de bomba, prevenindo funcionamento a seco.

A solução demonstrou-se adequada para: (i) laboratórios de ensino que necessitam de instrumentação de baixo custo com registro histórico; (ii) pequenos produtores urbanos que buscam automação básica com monitoramento remoto; e (iii) pesquisadores que precisam de uma plataforma *open-source* extensível para experimentos em hidroponia de precisão.

5.3. Integração com IA conversacional

No servidor MCP foram definidas as seguintes ferramentas de consulta: `get_current_readings` (leitura mais recente de cada sensor),

`get_time_series` (médias em janelas temporais configuráveis), `get_alarm_events` (histórico de alarmes e acionamentos) e `get_system_status` (estado geral do sistema). Todas as ferramentas são somente leitura, sem capacidade de acionar atuadores ou modificar parâmetros de controle.

Um agente de IA conectado ao servidor MCP foi avaliado em 15 perguntas em linguagem natural, como: "Qual foi a média de pH nas últimas 6 horas?", "A condutividade ficou abaixo do ideal hoje?" e "Quantas vezes a bomba de nutrientes acionou?". Em **14 das 15 perguntas (93,3 %)**, o agente recuperou os dados corretamente do Redis e retornou respostas consistentes com os registros brutos. A resposta incorreta deveu-se a uma ambiguidade temporal na pergunta, não a falha na integração. Essa abordagem difere de soluções puramente textuais [12] ao garantir que as respostas reflitam dados reais — aspecto crítico em contextos de controle de processos.

5.4. Análise comparativa de protocolos

Testes preliminares indicam que o protocolo MQTT tende a oferecer menor latência fim a fim e maior robustez frente a perdas de pacote em comparação ao HTTP, tornando-o candidato mais adequado para cenários de rede instável ou com múltiplos nós de borda. Em contrapartida, o MQTT exige maior complexidade de configuração e consome mais recursos do microcontrolador, motivo pelo qual sua adoção é considerada evolução futura da arquitetura, após etapa de experimentação controlada com comparação formal de latência, *jitter* e consumo de energia.

5.5. *Dashboard* HydroMonitor

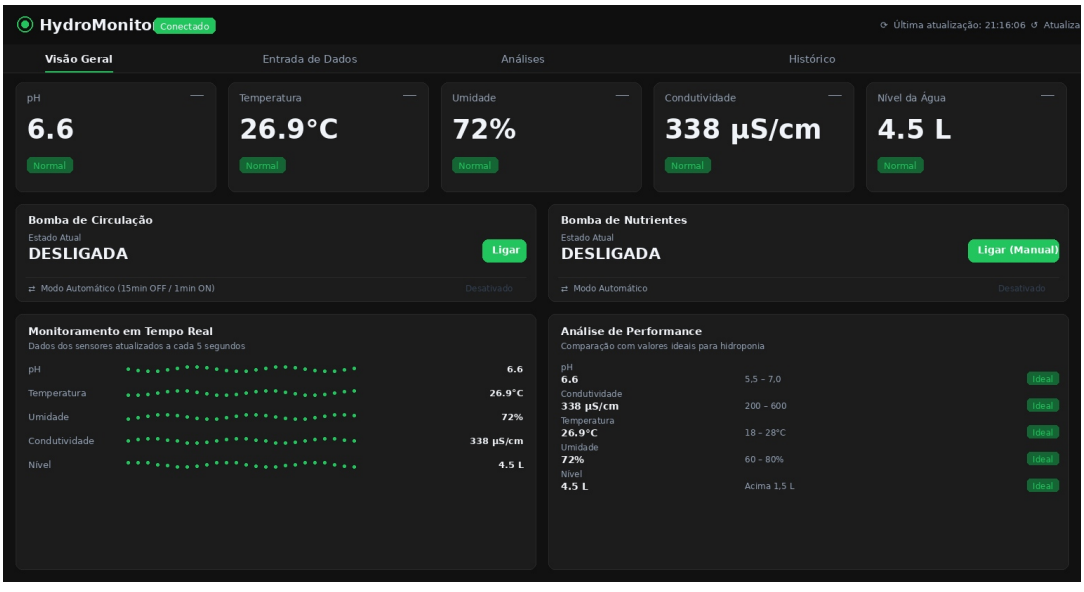

Figura 4. *Dashboard* web do SHINA com monitoramento em tempo real dos parâmetros hidropônicos (pH = 6,6; T = 26,9 °C; Umidade = 72 %; EC = 338 µS/cm; Nível = 4,5 L) e estado das bombas.

O *dashboard* (Figura 4) consolida, em uma única tela, os principais indicadores operacionais do sistema, exibindo em tempo quase real os valores de pH, temperatura, umidade do ar, condutividade elétrica e nível de água, cada um acompanhado de rótulos

de estado (por exemplo, "Normal" ou "Ideal"). Na mesma interface são apresentados os módulos de controle das bombas de circulação e de nutrientes, indicando o estado atual (ligada ou desligada), o modo de operação (automático ou manual) e permitindo o acionamento direto por meio de botões. A aba "Análise de Performance" compara cada parâmetro com as faixas ideais para hidroponia, fornecendo retroalimentação visual imediata ao operador.

6. Conclusão

O SHINA demonstrou ser uma solução tecnicamente viável para monitoramento e automação de sistemas hidropônicos, integrando de forma coerente tecnologias de IoT, armazenamento em memória e inteligência artificial generativa. A arquitetura modular de quatro camadas foi validada em operação contínua de 24 horas, com taxa de ingestão de 95,2 %, latência mediana de 10 ms e latência fim a fim de aproximadamente 180 ms, sobre *hardware* de consumo e rede Wi-Fi doméstica. Os resultados agronômicos confirmaram o funcionamento correto da lógica de automação: pH mantido entre 6,1 e 6,8 sem intervenção manual, e EC automaticamente restaurada após queda pelo consumo de nutrientes. A interface conversacional via MCP alcançou 93,3 % de precisão em 15 perguntas em linguagem natural, viabilizando consultas acessíveis sem necessidade de conhecimento técnico da API.

As principais limitações são: (i) ausência de autenticação e controle de acesso granular na API e no servidor MCP; (ii) validação restrita a um único nó de borda; (iii) lógica de controle baseada exclusivamente em histerese, sem avaliação de *overshoot* ou estabilidade em malha fechada; e (iv) ausência de métricas agronômicas ao longo de um ciclo completo de produção.

Como trabalhos futuros, pretende-se: (a) implementar autenticação, escopos de permissão por ferramenta MCP e *audit trail*; (b) realizar benchmarks controlados entre MQTT e HTTP em cenários com múltiplos nós de borda (10–50 dispositivos); (c) desenvolver controladores *fuzzy* (Mamdani) para regulação de pH e EC; (d) conduzir um ciclo completo de cultivo medindo crescimento, consumo de água e eficiência de nutrientes; e (e) realizar estudos de usabilidade da interface conversacional com produtores e estudantes.

References

SILVA, J. P.; OLIVEIRA, R. M.; PEREIRA, L. F. Automação de baixo custo de um sistema hidropônico com monitoramento remoto via IoT. *Anais de Congresso em Engenharia e Tecnologia*, 2021.

SANTOS, M. A.; LIMA, C. R. Automação de Monitoramento de Hidroponia. Trabalho de Conclusão de Curso – Universidade Presbiteriana Mackenzie, 2022.

FUGINO, G. H. et al. Sistema inteligente para o monitoramento do cultivo hidropônico de alface. *Universidade Estadual Paulista*, 2020.

SOUZA, A. L. et al. Sistema Hidropônico Automático de Baixo Custo. *CONTECC*, 2023.

LEVY, M. C. Produção hidropônica, a nova fronteira do desenvolvimento. *Revista LEV*, 2022.

SILVA, L. F. et al. Sistema de monitoramento de parâmetros da água em hidroponia. *Cadernos de Pesquisa e Desenvolvimento*, 2023.

MACHADO, R. S. et al. Desenvolvimento de hortaliças em cultivo hidropônico. *Revista Novos Desafios*, 2019.

SOUSA REIS, R. R. Uso da Internet das Coisas no monitoramento de alface em cultivo protegido. UTFPR, 2020.

DOMINGOS, A. S. Sistema de Monitoramento de Cultivo Hidropônico. IFSC, 2023.

REIS, R. R. S. Sistema hidropônico de pequeno porte utilizando tecnologias de automação. IFPB, 2020.

REDIS LTD. Redis Documentation: Data types and persistence. Disponível em: `https://redis.io/docs`. Acesso em: 2024.

SANTANA, P. R. et al. Artificial intelligence for small hydroponics farms employing decision support. *Revista Brasileira de Engenharia Agrícola e Ambiental*, 2023.

ANTHROPIC. Model Context Protocol (MCP) Specification. Disponível em: `https://modelcontextprotocol.io`. Acesso em: 2024.

PILATI, E. S.; PEREIRA, G. L. Automação para sistema hidropônico utilizando IoT e contêineres Docker. Instituto Federal do Paraná, 2025.

REVISTA CULTIVAR. Produção hidropônica já ocupa até três mil hectares no Brasil. Disponível em: `https://revistacultivar.com.br/noticias/producao-hidroponica-ja-ocupa-ate-tres-mil-hectares-no-brasil`. Acesso em: 05 dez. 2025.