UNIVERSIDADE FEDERAL DE ITAJUBÁ

INTELLIGENT CLIMATE CONTROL
MACHINE LEARNING ENGINEERING

ALEX ALVAREZ DUQUE - 2024005730
ARTUR GOMES SIMÃO - 2024006513
PEDRO LUCAS PEREIRA FERREIRA - 2024005982
IESTI-UNIFEI

Itajubá, 1 de dezembro de 2025

SUMÁRIO

1. RESUMO

O projeto Intelligent Climate Control propõe o desenvolvimento e a implementação de um sistema de Inteligência Artificial de Borda (Edge AI) no Raspberry Pi 5, com o objetivo de monitorar continuamente as condições ambientais e tomar decisões autônomas de controle físico. A solução foi projetada para rodar integralmente on-device, garantindo baixa latência e total independência da nuvem. O sistema utiliza um Small Language Model (SLM) para analisar dados de temperatura e umidade (DHT22), e pressão barométrica (BMP280) e controlar atuadores simulados (LEDs para ventilação/aquecimento).

O núcleo do sistema é o modelo llama3.2:3b (com 3 bilhões de parâmetros), executado via Ollama. A principal otimização implementada é a técnica de Function Calling, que permite ao SLM raciocinar sobre o ambiente e chamar funções de hardware (read_environment_data, set_led_state) em um ciclo estruturado. O modelo é quantizado para 4-bit, o que otimiza o consumo de memória e alcança uma taxa de avaliação de 5-10 tokens/s, com latência de inferência em chamada única de aproximadamente 15-40 segundos, gerando uma resposta de 30-90 segundos com as duas chamadas.

A aplicação suporta entrada dupla, permitindo a interação via terminal (linguagem natural) e via um botão físico (gatilho para controle automático baseado em regras definidas no config.yaml). O projeto demonstrou que o uso do Function Calling torna a lógica de controle confiável e resiliente a falhas de comunicação. O sistema é modular e utiliza um script install.sh para automatizar a configuração do ambiente, incluindo a instalação do Ollama e o download do modelo quantizado, validando a promessa da Edge AI para aplicações IoT em tempo real. O repositório completo do projeto pode ser encontrado no GitHub: https://github.com/12FlyBreads/climate-control-slm.git.

.

2. INTRODUÇÃO E MOTIVAÇÃO

Este projeto tem como objetivo desenvolver um Sistema Inteligente de Ventilação capaz de monitorar continuamente a temperatura e a umidade de ambientes internos e, a partir desses dados, tomar decisões automáticas para melhorar o conforto térmico. Ambientes fechados, como salas, escritórios e residências, frequentemente apresentam condições inadequadas devido ao acúmulo de calor e umidade, o que pode gerar desconforto, reduzir produtividade e afetar a saúde dos ocupantes.

Embora existam soluções tradicionais de controle ambiental, muitas operam com regras fixas ou exigem intervenção manual constante, o que limita sua eficiência. Nos últimos anos, o avanço das tecnologias de computação em borda (Edge Computing) e dos Small Language Models (SLMs) abriu novas possibilidades para sistemas inteligentes instalados diretamente em dispositivos compactos. O Raspberry Pi 5, aliado a modelos leves como Llama 3.2 e Phi-3, permite o processamento local de linguagem natural e tomada de decisão baseada em dados, sem depender da nuvem. Isso proporciona maior privacidade, menor latência e maior confiabilidade mesmo em ambientes sem conexão à internet.

Nesse contexto, surge a oportunidade de desenvolver um sistema inteligente capaz de não apenas monitorar condições ambientais, mas também interpretar esses dados e agir de forma autônoma, simulando a tomada de decisão humana - porém com constância e precisão superiores. Assim, este projeto propõe a criação de um Sistema Inteligente de Ventilação, que utiliza sensores físicos e um SLM executado localmente para analisar temperatura e umidade, fornecer diagnósticos em linguagem natural e controlar automaticamente a ventilação (simulada por LEDs) conforme as condições do ambiente.

Essa abordagem integra conceitos modernos de IoT, automação e inteligência artificial generativa aplicada à borda, oferecendo uma solução acessível, de baixo custo e com potencial de aplicação em diversos tipos de ambientes. O sistema demonstra como a combinação entre dados de sensores e modelos de linguagem pode elevar a automação ambiental a um novo patamar, permitindo decisões fundamentadas e explicáveis em tempo real.

3. REVISÃO BIBLIOGRÁFICA

3.1. CONCEITOS BÁSICOS

Os avanços recentes em Internet das Coisas (IoT), computação em borda (Edge Computing) e modelos de linguagem de pequeno porte (Small Language Models – SLMs) possibilitaram o desenvolvimento de sistemas inteligentes capazes de processar dados localmente e tomar decisões de forma autônoma. Estes sistemas combinam sensores físicos, atuadores e algoritmos de inteligência artificial para interpretar o ambiente e executar ações em tempo real.

No contexto de IoT, sensores são dispositivos responsáveis por coletar informações do ambiente físico, como temperatura, umidade, luminosidade e movimento. No caso deste projeto, o sensor DHT22 desempenha papel fundamental ao medir continuamente temperatura e umidade, fornecendo os dados necessários para análise do conforto térmico. Atuadores, como LEDs, representam fisicamente as ações tomadas, simulando o acionamento da ventilação com base nas condições detectadas.

A Computação em Borda (Edge Computing) consiste no processamento de dados diretamente no dispositivo ou próximo da fonte de coleta, reduzindo a dependência da nuvem. Essa abordagem diminui a latência, aumenta a confiabilidade em ambientes sem internet e preserva a privacidade dos usuários. Dispositivos como o Raspberry Pi 5 são amplamente utilizados devido ao seu baixo consumo energético, custo acessível e capacidade de executar modelos de IA localmente.

Small Language Models (SLMs) são versões reduzidas de modelos de linguagem natural, capazes de interpretar informações e gerar textos de forma eficiente mesmo em hardware limitado. Diferentemente dos grandes modelos baseados em nuvem, SLMs como Llama 3.2 e Phi-3 podem ser executados localmente, permitindo que análises, explicações e recomendações sejam feitas diretamente no dispositivo, sem depender de servidores externos. Isso torna o sistema mais transparente, interpretável e capaz de justificar suas decisões.

A integração entre sensores ambientais, Edge Computing e SLMs permite criar sistemas capazes de imitar a tomada de decisão humana com maior consistência e precisão. No contexto do controle térmico, essa combinação possibilita monitoramento contínuo do ambiente, interpretação dos dados recebidos e ações automáticas voltadas para o conforto térmico.

3.2. TRABALHOS RELACIONADOS E APLICAÇÕES PRÁTICAS

Os sistemas de monitoramento ambiental e controle automático de ventilação são amplamente estudados em áreas como IoT, automação residencial e climatização inteligente. Muitos trabalhos utilizam microcontroladores como Arduino e ESP32 para acionar ventiladores ou sistemas de refrigeração com base em regras fixas. Embora funcionais, esses sistemas não interpretam o contexto e não oferecem explicações sobre suas decisões, limitando sua flexibilidade e capacidade de adaptação.

Com o avanço da Edge AI, surgiram pesquisas que exploram a execução local de modelos de linguagem e IA em dispositivos compactos. Estudos recentes demonstram que modelos como Llama 3.x, Gemma e Phi-3 podem ser executados diretamente em hardware embarcado, incluindo Raspberry Pi 4 e 5. Esses modelos permitem que o sistema gere interpretações e recomendações em linguagem natural, tornando-o mais acessível, transparente e útil ao usuário.

Projetos acadêmicos envolvendo ambientes inteligentes destacam a importância de integrar sensores de temperatura, umidade e qualidade do ar para otimizar ventilação, reduzir consumo

energético e melhorar o conforto. Tais sistemas são aplicados em casas inteligentes, escritórios, salas de servidores e ambientes industriais, onde o controle térmico é essencial para segurança e eficiência.

No campo das aplicações práticas, sistemas automatizados já são utilizados para acionar exaustores, ventiladores e sistemas de refrigeração com base em condições ambientais. Entretanto, poucos empregam modelos de linguagem capazes de explicar suas ações, o que representa uma inovação significativa. A introdução de SLMs permite que o sistema justifique decisões como “A temperatura está elevada e a umidade moderada; por isso, a ventilação foi ativada”, aumentando a confiabilidade e a compreensão do usuário.

Assim, os trabalhos analisados mostram que a combinação entre IoT, Edge Computing e modelos de linguagem representa um avanço significativo nos sistemas ambientais inteligentes, proporcionando maior autonomia, eficiência e clareza nas decisões. O projeto desenvolvido neste relatório segue essa tendência, oferecendo uma solução acessível, educacional e totalmente offline para automação ambiental.

4. DESIGN DO SISTEMA

4.1. ESCOPO DO PROJETO

A arquitetura do projeto de Sistema de Controle Inteligente do Clima (Intelligent Climate Control) foi projetada para garantir processamento local no Edge , modularidade e alta capacidade de raciocínio lógico através da técnica de Function Calling. O Raspberry Pi 5 é a plataforma central para todo o processamento.

A ideia principal do projeto é transformar o Raspberry Pi 5 em um Agente de Controle Inteligente totalmente autônomo, capaz de converter linguagem natural em ações de controle físico , sem qualquer dependência de serviços em nuvem.

Com base nisso, o aplicação foi desenvolvida com dois principais modos de operação: o primeiro sendo pelo usuário pelo próprio terminal, fazendo inferências sobre as condições climáticas do ambiente (como temperatura, umidade e pressão atmosférica) e podendo receber uma resposta através dos LEDs atuadores; o segundo modo consiste no pressionamento do botão do circuito, fazendo com que o SLM recalcule as condições climáticas e a partir de um prompt pré definido já calcule qual deve ser o melhor atuador a ser ativado.

4.2. ARQUITETURA E HARDWARE

A arquitetura é dividida em quatro camadas principais: Hardware, Interface de Hardware, Núcleo do Agente de IA e Interface do Usuário (UI).

O fluxo de dados é o seguinte:

1. **Entrada do Usuário/Gatilho:** O sistema aceita entrada via Terminal (chat) ou Botão físico.
2. **Início da Inferência:** O módulo main.py envia a intenção do usuário (ou o prompt de controle automático do botão) ao módulo inference.py.
3. **Ciclo de Function Calling (Decisão e Ação):**
   - O **SLM (via Ollama)** recebe o prompt e decide se precisa de dados de sensores.
   - Se for necessário ler sensores, o SLM gera uma chamada de função (read_environment_data).
   - O inference.py executa a função no hardware.py e injeta os dados de volta no SLM.
   - Com os dados, o SLM raciocina e gera uma chamada de função de controle (set_led_state) para ligar o atuador apropriado (LED).
4. **Saída:** O SLM retorna a resposta final de texto para o main.py, que a exibe na interface interativa do terminal.

A seguir é mostrada a tabela com os componentes utilizadas no circuito do projeto (todas localizadas no módulo hardware.py):

Componente	Tipo	Função	Conexão
DHT22	Sensor	Temperatura/umidade	GPIO 16
BMP280	Sensor	Pressão barométrica	I2C
LED amarelo	Atuador	Resfriamento	GPIO 19

LED vermelho	Atuador	Aquecedor	GPIO 13
LED verde	Atuador	Condições ideais	GPIO 26
Botão	Entrada Física	Gatilho de controle	GPIO 20

Tabela 1: Componentes de hardware

A arquitetura de software segue o design modular obrigatório, garantindo a separação de responsabilidades (Separation of Concerns).

A seguir é explicado brevemente o fluxo de Function Calling:

1. O main.py envia o user_input (ou button_prompt) ao slm_inference_with_tools.
2. A primeira chamada Ollama retorna a solicitação para a ferramenta read_environment_data.
3. O inference.py executa hardware.read_environment_data() e recebe o JSON dos dados.
4. O inference.py envia esses dados de volta ao SLM, forçando a segunda chamada Ollama.
5. A segunda chamada resulta na decisão de controle e, se aplicável, retorna uma chamada de função set_led_state (ex: set_led_state(color="yellow", state="on")).

4.3. MODELO E JUSTIFICATIVA

O modelo escolhido (llama3.2:3b) atende estritamente aos requisitos técnicos de rodar no Edge e ser pequeno e quantizado.

O modelo possui 3 bilhões de parâmetros além de seu tamanho ser ideal para o Raspberry Pi 5, garantindo menor consumo de RAM. Além disso, o modelo é utilizado na versão quantizada para 4-bit. A quantização reduz significativamente o uso de memória e a latência de inferência, crucial para o desempenho no Edge.

A funcionalidade de Function Calling é nativamente suportada pela API do Ollama, simplificando a implementação e melhorando a confiabilidade das ações de controle do SLM.

5. IMPLEMENTAÇÃO

5.1. INSTALAÇÃO

A instalação do sistema é realizada de forma automatizada por meio de um script Bash (install.sh), que cria o ambiente virtual, instala as dependências necessárias e prepara o modelo para execução. O passo a passo de instalação está disponível no repositório do GitHub do projeto: https://github.com/12FlyBreads/climate-control-slm.git.

Além disso, são necessários pré-requisitos como Python 3.9+, um computador de bordo com LINUX (Raspberry 5) e os componentes de hardware descritos.

Caso a instalação por meio do script dê algum erro, é recomendado fazer a criação do ambiente virtual e instalação das bibliotecas de forma manual. Os comandos são o seguinte:

- python3 -m venv CLIMATECTRL –system-packages : criar ambiente virtual
- source CLIMATECTRL/bin/activate : ativar ambiente virtual
- pip install <nome-do-pacote> : para instalar cada biblioteca utilizada

Além das instalações das bibliotecas Python, é necessário instalar o Ollama e os principais aplicativos localizados no install.sh.

5.2. ESTRUTURA GERAL DO SISTEMA

A arquitetura de software é totalmente baseada em Python e segue um rigoroso design modular para facilitar o desenvolvimento no Raspberry Pi 5. A comunicação entre a camada de lógica e a camada de hardware é feita exclusivamente através da técnica de Function Calling:

Inicialmente, falando sobre os os módulos util.py e config.yaml, eles gerenciam todos os parâmetros externos ao código, atendendo ao requisito de arquivo de configuração.

- config.yaml: É o repositório central de parâmetros. Inclui o nome do modelo SLM (llama3.2:3b), o mapeamento dos pinos GPIO e o template de regras para o controle automático (button_control_prompt).

- utils.py: Contém a função load_config, que usa a biblioteca PyYAML para carregar o config.yaml e parsear os dados para o Python. Esta função também inclui manipulação de erros (try-except) para FileNotFoundError e yaml.YAMLError.

O módulo hardware.py é a camada de abstração de hardware, isolando o restante do sistema dos detalhes de pinagem e protocolos de comunicação (I2C, GPIO).

- Inicialização: Inicializa todos os sensores (DHT22Sensor, bmp280Sensor) e atuadores (leds, button) usando os pinos definidos no config.yaml.

- Funções de Sensor (read_environment_data): Tenta ler todos os sensores em uma única chamada e retorna um dicionário estruturado. Inclui uma robusta manipulação de erros (try-except RuntimeError) e fallback para retornar um dicionário com {"error": ...} caso a leitura de um sensor falhe (comum com DHT22).

- Funções de Atuador (set_led_state): Aceita argumentos color e state (on/off) para manipular os LEDs, servindo como a ação que o SLM chama.

O módulo de inferência ao SLM é o inference.py, onde a lógica da IA reside e a otimização de Function Calling é implementada.

- SYSTEM_MESSAGE: Define o comportamento do modelo como um "IoT Climate Control Assistant" e o instrui a usar as ferramentas.
- get_ollama_tools: Mapeia as funções do hardware.py para o JSON Schema exigido pelo Ollama. Isso é a essência do Function Calling.
- slm_inference_with_tools: Implementa o ciclo multi-turno de inferência. Se o SLM retornar uma tool_call, o código Python executa a função correspondente (do dicionário AVAILABLE_TOOLS) e injeta o resultado da função no histórico da conversa (role: 'tool'), forçando o SLM a raciocinar sobre os dados do mundo real na próxima chamada.

Finalmente, o módulo principal main.py gerencia a experiência do usuário e o loop da aplicação.

- Carregamento de Configurações: Carrega todas as configurações do utils.py para usar variáveis globais (MODEL, BUTTON_CHECK_INTERVAL, PROMPT_TEMPLATE).
- preload_model: Garante que o modelo SLM seja carregado na RAM antes da primeira interação para mitigar a alta latência inicial do Raspberry Pi 5.
- interactive_mode: O loop principal implementa a funcionalidade de entrada dupla:
  - Lida com a entrada de Terminal (chat interativo).
  - Verifica o estado do Botão em intervalos de 0.5s (BUTTON_CHECK_INTERVAL). Se o botão for pressionado, injeta o button_prompt de controle automático no SLM.
- Métricas: A função display_status_and_control_leds exibe o status atual do hardware e as métricas de desempenho do SLM (Duração Total, Eval Rate em tokens/s).

5.3. SLM

A integração da inteligência artificial é realizada utilizando o Ollama como runtime de Edge AI. Esta abordagem garante o Processamento Local, pois o modelo é executado diretamente no Raspberry Pi 5, assegurando que toda a inferência seja on-device e sem dependências de serviços em nuvem. Para a Comunicação, o módulo inference.py utiliza o cliente Python do Ollama para gerenciar a sessão de chat e o histórico de mensagens, permitindo que o SLM mantenha o contexto completo da conversa. O Modelo base (llama3.2:3b) foi selecionado por seu bom desempenho e por cumprir o requisito de possuir 3B parâmetros.

Passando para as otimizações, o sistema utiliza a Function Calling como técnica de otimização principal. Esta abordagem é crucial para a interação estruturada com o hardware. Primeiramente, a função get_ollama_tools() no inference.py registra as funções de hardware (read_environment_data, set_led_state, get_led_status) no formato JSON Schema, que é o vocabulário que o SLM entende. A lógica Ciclo Multi-Turn é implementada em

slm_inference_with_tools: o SLM retorna um pedido de função (Tool Call), o Python executa a função no hardware.py, e o resultado (dado do sensor ou confirmação de ação) é enviado de volta ao SLM como uma mensagem de role tool. Finalmente, o SLM gera a resposta final ou uma segunda Tool Call (se for uma ação condicional).

Outro ponto importante de otimização é a quantização do modelo generativo, que reduz drasticamente a carga de memória e melhora a taxa de tokens/s.

O projeto também emprega uma estratégia de prompt dupla para garantir a confiabilidade e o raciocínio de controle do SLM. A Mensagem de Sistema (SYSTEM_MESSAGE) define a identidade e as regras de operação da IA, instruindo-a explicitamente a usar as ferramentas disponíveis. Complementarmente, o Prompt de Controle Automático é injetado no sistema (carregado como template do config.yaml) quando o botão é pressionado. O texto desse prompt contém as regras condicionais em linguagem natural, permitindo que o SLM aplique o raciocínio lógico aos dados lidos na primeira chamada de função, transformando o SLM em um motor de regras de controle altamente flexível.

6. AVALIAÇÃO DE DESEMPENHO

6.1. MÉTRICAS UTILIZADAS

A fase de testes e avaliação foi projetada para validar tanto a funcionalidade básica do sistema IoT quanto a confiabilidade e o desempenho da camada de Inteligência Artificial no Edge. Os testes foram executados inteiramente no Raspberry Pi 5, utilizando o modelo llama3.2:3b quantizado em 4-bit.

As métricas de desempenho foram coletadas pelo main.py para avaliar a eficiência do processamento local no Raspberry Pi 5.

A latência média para uma única chamada foi de aproximadamente 15-40 segundos, podendo às vezes demorar um pouco mais nos cenários de controle condicional que exigem duas chamadas (chegando a uma média de 30-90 segundos). Este desempenho é bem consistente e condiz com as expectativas esperadas.

Já a taxa de avaliação (Eval Rate), também se mostrou consistente, mas um pouco abaixo do esperado, com uma média de 5-10 tokens/s. Esse resultado se deve a quantização de 4 bits do modelo, que otimiza a taxa de transferência no hardware limitado.

Em geral os resultados obtidos pelo modelo foram bem consistentes, principalmente em inferências mais simples sobre as condições ideais.

6.2. CASOS DE FALHA E LIMITAÇÕES

O sistema foi testado para lidar com falhas críticas, imprimindo erros de hardware no terminal quando ocorridos (falha de sensores, comandos ambíguos, etc.). É importante salientar também que o programa faz impressões frequentes sobre qual estado o SLM está lidando.

6.3. ANÁLISE DE CONFIABILIDADE

A confiabilidade do sistema é sustentada por três pilares:

1. **Confiabilidade da Lógica de Controle**: A lógica condicional (IF/THEN) para o Botão de Ação Rápida é definida por Prompt Engineering em uma string clara no config.yaml. Isso torna o SLM a única autoridade de decisão, eliminando o risco de erros de lógica de controle embutidos no código Python.
2. **Resiliência a Falhas de Comunicação**: A abordagem de Function Calling é intrinsecamente resiliente. Se a função de hardware falhar, o resultado ({"error": ...}) é enviado de volta ao SLM, que é treinado para interpretar o erro e notificar o usuário.
3. **Estabilidade de Hardware**: O uso da biblioteca gpiozero e a separação de código no hardware.py aumentam a estabilidade do acesso ao GPIO, reduzindo a chance de erros de permissão ou timing. O script install.sh garante que as permissões de gpio e i2c sejam definidas corretamente.

6.4. DISCUSSÃO DOS RESULTADOS

O desempenho geral do modelo foi considerado satisfatório. A versão final quantizada apresenta um ótimo custo-benefício entre velocidade de inferência e taxa de avaliação. Sobre isso, algumas observações podem ser realizadas:

- O tempo médio de uma inferência (15-40 segundos) e total (30-90 segundos), pode ser considerado ótimo para o nível de complexidade de operação e limitações do dispositivo de bordo.
- A taxa de avaliação (5-10 tokens/s) está um pouco abaixo do esperado, mas ainda assim é um resultado bem consistente.

Em geral, as respostas e tomadas de decisões geradas pela inteligência artificial saíram como o esperado, principalmente com a utilização de comandos mais simples (como perguntar a temperatura atual, pedir para ligar algum atuador, etc.). Na utilização de comandos condicionais mais complexos (como o caso do button_prompt), o SLM se mostrou bem conciso com as respostas, portanto em alguns casos ainda houveram algumas inferências um tanto quanto confusas.

Um dos principais desafios do projeto foi exatamente este problema de respostas confusas geradas pelo SLM. A escolha de prompt ideal para realizar a inferência automática de qual atuador utilizado foi configurada várias vezes até que o melhor resultado médio fosse gerado.

Podemos assim concluir que mesmo com limitações de hardware, o modelo SLM desempenhou muito bem, assim como o esperado. As métricas se mostraram excelentes com os testes, e as respostas geradas foram bem precisas na maioria dos casos. É interessante se comentar que a escolha do prompt talvez seja uma das partes mais fundamentais para o funcionamento correto do sistema com base nas inferências.

7. CONCLUSÃO

7.1. CONCLUSÃO GERAL

O projeto Intelligent Climate Control demonstrou com sucesso a viabilidade de implantar um sistema complexo de Inteligência Artificial de Borda no Raspberry Pi 5. O objetivo de criar uma solução totalmente on-device foi atingido, utilizando o SLM quantizado para 4 bits para realizar raciocínio lógico e controle físico.

A implementação do Function Calling foi o pilar técnico que possibilitou a comunicação estruturada e confiável entre a camada de linguagem natural e o hardware, transformando o SLM em um agente de controle eficiente, capaz de responder a comandos de texto e a gatilhos físicos de forma autônoma. O projeto valida a promessa da Edge AI para aplicações IoT em tempo real.

7.2. MELHORIAS FUTURAS

A implementação atual atende aos requisitos principais, mas pode ser expandida em termos de funcionalidade e hardware:

- **Implementação de Controle Preditivo:** O projeto pode evoluir para incorporar o controle preditivo. Isso exigiria primeiro a implementação da coleta e registro de dados de sensor em CSV/JSON (uma etapa de Data Management) e, em seguida, o treinamento de um modelo simples ou o uso do próprio SLM para analisar a taxa de mudança de temperatura e atuar proativamente.
- **Expansão de Atuadores:** Substituir os LEDs por atuadores reais (relés acionando um ventilador ou aquecedor) para mover o projeto da simulação para uma aplicação prática.
- **Interface Web:** Adicionar uma interface Web (usando Flask ou FastAPI) para visualização em tempo real dos dados e métricas do SLM, complementando a interface de terminal atual.

Além disso, o sistema de Controle de Clima é altamente escalável, principalmente devido à sua arquitetura modular. A abordagem de Processamento Local (Edge) permite a fácil replicação. Cada Raspberry Pi 5 pode atuar como um gateway de IA independente, monitorando uma zona climática distinta.

Para gerenciar várias zonas, uma abordagem alternativa seria introduzir um protocolo de mensagens leve como MQTT, permitindo que os RPis se comuniquem e coordenem as ações climáticas em diferentes áreas.

8. REFERÊNCIAS

ADAFRUIT. **Adafruit CircuitPython DHT Sensor Library.** [S.l.]: Adafruit, [2025]. Disponível em: https://github.com/adafruit/Adafruit_CircuitPython_DHT. Acesso em: 6 dez. 2025.

GPIO ZERO DEVELOPERS. **GPIO Zero Documentation.** [S.l.]: GPIO Zero, [2025]. Disponível em: https://gpiozero.readthedocs.io/. Acesso em: 6 dez. 2025.

MANDUZIO, G. A. et al. Improving Small-Scale Large Language Models Function Calling for Reasoning Tasks. **arXiv**, 2024. Disponível em: https://arxiv.org/abs/2410.18890. Acesso em: 6 dez. 2025.

MORAIS, I. S.; GONÇALVES, P. F.; LEDUR, C. L. et al. **Introdução a Big Data e Internet das Coisas (IoT).** Porto Alegre: SAGAH, 2018. E-book.

OLLAMA. **Ollama Documentation (API and Installation Guide).** [S.l.]: Ollama, [2025]. Disponível em: https://docs.ollama.com/. Acesso em: 6 dez. 2025.

RASPBERRY PI FOUNDATION. **Raspberry Pi 5 Documentation.** Cambridge: Raspberry Pi Foundation, [2024]. Disponível em: https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-5/. Acesso em: 6 dez. 2025.

WANG, M. et al. Function Calling in Large Language Models: Industrial Practices, Challenges, and Future Directions. **OpenReview**, 2025. Disponível em: https://openreview.net/pdf/d01d50e27f7636724789f2aad6f4ac378749a0e1.pdf. Acesso em: 6 dez. 2025.