



UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Ervin Alain Bolivar Huayhua

**Integração de LLMs na Otimização e
Planejamento de Sistemas Multiagentes IoT e
Interação Conversacional**

Campinas

2025

Ervin Alain Bolivar Huayhua

Integração de LLMs na Otimização e Planejamento de Sistemas Multiagentes IoRT e Interação Conversacional

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Automação.

Orientador: Prof. Dr. Eric Rohmer

Este exemplar corresponde à versão final da dissertação defendida pelo aluno **Ervin Alain Bolivar Huayhua**, e orientada pelo Prof. Dr. Eric Rohmer

Campinas

2025

Ficha catalográfica
Universidade Estadual de Campinas (UNICAMP)
Biblioteca da Área de Engenharia e Arquitetura
Vanessa Evelyn Costa - CRB 8/8295

B638i Bolivar Huayhua, Ervin Alain, 1993-
Integração de LLMs na otimização e planejamento de sistemas
multiagentes IoRT e interação conversacional / Ervin Alain Bolivar
Huayhua. – Campinas, SP : [s.n.], 2025.

Orientador: Eric Rohmer.
Dissertação (mestrado) – Universidade Estadual de Campinas
(UNICAMP), Faculdade de Engenharia Elétrica e de Computação.

1. Grandes modelos de linguagem. 2. Sistemas multiagentes. 3.
Aprendizado profundo. 4. Aprendizado de máquina. 5. Otimização. 6.
Robótica. I. Rohmer, Eric, 1974-. II. Universidade Estadual de Campinas
(UNICAMP). Faculdade de Engenharia Elétrica e de Computação. III.
Título.

Informações complementares

Título em outro idioma: Integration of LLMs in the optimization and planning of
multi-agent IoRT systems and conversational interaction

Palavras-chave em inglês:

Large language models
Multi-agent systems
Deep learning
Machine learning
Optimization
Robotics

Área de concentração: Automação

Titulação: Mestre em Engenharia Elétrica

Banca examinadora:

Eric Rohmer [Orientador]
Ricardo Ribeiro Gudwin
Exuperry Barros Costa

Data de defesa: 22-04-2025

Programa de Pós-Graduação: Engenharia Elétrica

Objetivos de Desenvolvimento Sustentável (ODS)

ODS: 3. Saúde e bem-estar

ODS: 11. Cidades e comunidades sustentáveis

Identificação e informações acadêmicas do(a) aluno(a)

- ORCID do autor: <https://orcid.org/0009-0002-9626-3413>
- Currículo Lattes do autor: <http://lattes.cnpq.br/1692285115716301>

Prof. Dr. Eric Rohmer (Presidente)

Prof. Dr. Exuperry Barros Costa

Prof. Dr. Ricardo Ribeiro Gudwin

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de PósGraduação da Faculdade de Engenharia Elétrica e de Computação.

Aos meus avós, que partiram antes de ver esta conquista, mas cujo exemplo de trabalho silencioso e dignidade humilde se transformou na força que impulsionou minha perseverança.

Aos meus pais e irmãos, com uma gratidão especial ao meu irmão mais velho, por me ensinarem — com atitudes e não apenas com palavras — que, apesar das dificuldades, a educação é o único patrimônio que ninguém pode nos tirar. Por acreditarem em mim mesmo quando nem eu conseguia enxergar além. Dedico este trabalho, de coração cheio, àqueles que fizeram de mim uma pessoa curiosa, teimosa e, às vezes, até obsessiva por aprender — porque essa paixão, no fim das contas, sempre foi a minha forma de tentar compreender o mundo e encontrar meu lugar nele.

Ao Kevin, por me lembrar que ser estrangeiro não significa estar sozinho, mas sim poder confiar em um grande amigo. À Maria e ao César, por estarem presentes nos silêncios e nos cafés, mesmo quando eu já duvidava de mim: eles me mostraram lealdade e amor incondicional. À Nahikari, por tentar compreender esse amor estranho que tenho por códigos, robôs e perguntas sem resposta; que me questiona, mas também me acompanha com uma amizade sincera.

E a você, Rafaela, que chegou quando eu mais precisava de apoio e esperança. Obrigado por me dar força moral e espiritual enquanto eu lutava para compilar cada linha de código. Por motivar meu coração com palavras de amor e me aproximar de Deus da forma mais terna. Rafaela, a voz da minha alma.

Por fim, dedico este trabalho àqueles que não aparecem nos currículos nem nos artigos acadêmicos: aqueles que me sustentaram naquilo que ninguém vê — o ânimo, o chão e o afeto. Aos meus colegas do laboratório LCA e amigos da vida, por construírem comigo não apenas projetos, mas também memórias valiosas: Maury, Alvaro, Tarcio, Vinicius, Victor, Pedro, Fabricio, Ariel, Felipe, Alan, Percy e Breno. Cada conversa, cada linha de código compartilhada, cada pequena conquista celebrada como se fosse de todos — fizeram com que este caminho fosse mais humano e mais leve. Obrigado por fazerem do laboratório um lugar onde a ciência também é feita com amizade.

Agradecimentos

A Deus, fonte de toda sabedoria e paz, agradeço por ter me sustentado nos momentos de silêncio, nas noites solitárias diante da tela, e nas horas em que duvidei de mim. Se hoje escrevo estas palavras, é porque fui constantemente amparado por uma força maior que me lembrou que perseverar é também um ato de fé. Ao olhar para trás, percebo que este mestrado foi mais do que um percurso acadêmico: foi uma jornada de transformação pessoal. Aprendi a pensar com mais clareza, a lidar com a frustração como parte do processo criativo e, acima de tudo, a confiar no poder da colaboração humana e tecnológica.

Expresso minha mais profunda gratidão ao meu orientador, Prof. Dr. Eric Rohmer, por sua orientação firme e generosa ao longo deste percurso. Sua confiança, disponibilidade e atenção aos detalhes foram fundamentais para o desenvolvimento deste trabalho. Com ele, aprendi que a elegância de uma solução está na simplicidade bem fundamentada. Suas palavras — “não existem dados errôneos ou ruins, apenas resultados que nos aproximam, passo a passo, dos dados esperados” — marcaram profundamente minha forma de compreender a pesquisa.

Sou também muito grato aos membros da banca, Professores Ricardo Gudwin, Exuperry Barros e Jayr Pereira, por seus comentários críticos e generosos, que me fizeram perceber novos caminhos de investigação. Suas perguntas ampliaram minha visão sobre o que significa “otimização” em sistemas complexos.

Agradeço à Ericsson pelo apoio financeiro por meio da bolsa que sustentou esta pesquisa por 24 meses. Essa oportunidade me permitiu mergulhar com dedicação total no desenvolvimento deste trabalho, além de me aproximar de desafios reais da indústria.

Aos amigos e colegas do laboratório LCA, obrigado por tornarem o cotidiano mais leve, criativo e, por vezes, até divertido. Cada conversa no café, cada sessão de testes ou debugging coletivo contribuiu para a construção desta dissertação. Agradeço especialmente a Cesar e Maria, por serem não apenas companheiros de jornada, mas cocriadores de ideias e soluções.

Este trabalho também é fruto do suporte da Faculdade de Engenharia Elétrica e de Computação (FEEC/UNICAMP), que me ofereceu estrutura, liberdade e um ambiente de excelência para crescer.

“A inteligência artificial não substitui o pensamento humano, mas amplia nossa capacidade de imaginar e construir o futuro.”

(Fei-Fei Li)

Resumo

A interação entre humanos e sistemas robóticos em Ambientes Inteligentes *Smart Environments* (SEs) tem avançado consideravelmente nos últimos anos. Mas ainda existem desafios importantes na coordenação de agentes heterogêneos — como robôs móveis, manipuladores e dispositivos IoT — principalmente quando a comunicação é feita em linguagem natural. E apesar das abordagens tradicionais baseadas em planejamento simbólico, como o *Planning Domain Definition Language* (PDDL), essas apresentam limitações em ambientes dinâmicos e tarefas complexas de *Long-Horizon* (LH).

E é nesse contexto que este trabalho propõe um *framework* inovador, baseado na integração de *Large Language Models* (LLMs) de última geração — incluindo Gemini-1.5, GPT-3.5 turbo e GPT-4o — em um pipeline modular para a otimização e planejamento de tarefas em sistemas multiagentes *Internet of Robotic Things* (IoRT). O pipeline inclui raciocínio distribuído, armazenamento contextual e replanejamento adaptativo com a técnica *Reasoning and Acting* (ReAct), permitindo que agentes físicos colaborem de forma robusta e responsiva com agentes humanos.

Portanto, os experimentos demonstraram ganhos significativos na robustez, adaptabilidade e precisão dos planos gerados por LLMs em comparação com abordagens puramente simbólicas. Entre os modelos avaliados, o GPT-4o se destacou especialmente em tarefas complexas de LH, apresentando uma taxa de sucesso até 20% superior em relação a modelos como o LLaMa + ReAct. Esse desempenho mais elevado está diretamente relacionado à sua maior capacidade de interpretar intenções em linguagem natural e gerenciar de forma eficiente o *feedback* em tempo real proveniente dos agentes físicos e cognitivos durante a execução das atividades.

Palavras-chaves: LLM; Sistemas Multiagentes; *Long-Horizon*, Planejamento de Tarefas, *Internet of Robotic Things*.

Abstract

Interaction between humans and robotic systems in *Smart Environments (SEs)* has advanced considerably in recent years. But significant challenges still remain in coordinating heterogeneous agents—such as mobile robots, manipulators, and *IoT* devices—especially when communication is conducted through natural language. And while traditional approaches based on symbolic planning, such as *Planning Domain Definition Language (PDDL)*, provide a structured foundation, they present limitations in dynamic environments and complex *Long-Horizon (LH)* tasks.

It is in this context that this work proposes an innovative framework based on the integration of next-generation *Large Language Models (LLMs)* — including Gemini-1.5, GPT-3.5 turbo, and GPT-4o — into a modular pipeline for task optimization and planning in *Internet of Robotic Things (IoRT)* multi-agent systems. The pipeline includes distributed reasoning, contextual storage, and adaptive replanning using the *Reasoning and Acting (ReAct)* technique, enabling physical agents to collaborate robustly and responsively with human agents.

Therefore, the experiments demonstrated significant improvements in the robustness, adaptability, and accuracy of the plans generated by *LLMs* when compared to purely symbolic approaches. Among the evaluated models, GPT-4o stood out particularly in complex *LH* tasks, achieving a success rate up to 20% higher compared to models such as LLaMa + *ReAct*. This superior performance is directly attributed to its enhanced ability to interpret intentions expressed in natural language and to efficiently manage real-time feedback from both physical and cognitive agents during task execution.

Keywords: *Large Language Model (LLM)*; Multi-Agent Systems; Long Horizon; Task Planning; *Internet of Robotic Things (IoRT)*.

Lista de Siglas

API *Application Programming Interface.* 45, 46, 56

ASR *Average Success Rate.* 36

ASRS *ASR with Syntax Errors.* 36, 54

CE *Constrains Error.* 35, 37

CoT *Chain-of-Thought.* 24–27, 32, 33, 35, 60

GCR *Goal Condition Recall.* 36

HRI *Human Robot Interaction.* 20, 22–24, 29, 56

IA *Artificial Inteligence.* 18

IoRT *Internet of Robotic Things.* 7, 8, 19, 22, 50, 67

IoT *Internet of Thing.* 7, 8, 17, 18, 22, 26, 28, 30, 38, 44, 46, 67

LH *Long-Horizon.* 7, 8, 31, 33, 39, 64, 65, 67

LLM *Large Language Model.* 7, 8, 17–20, 22–36, 38–41, 43–46, 48, 50, 52–56, 59, 60, 65–67, 69, 70

MLDT *Multi-Level Decomposition Task planning.* 27, 30, 33, 34

MRED *Multi-Robot collaboration Error Diagnostic.* 35, 36

MTP *Meta-Task Planning.* 27, 30, 34

Only_SyE *Apenas Erro de Sintaxe.* 35, 37

PDDL *Planning Domain Definition Language.* 7, 8, 27, 30, 31, 34, 50

pHRC *Physical Human-Robot Collaboration.* 24

PIE *Planning Error.* 35, 37

PoE *Position Error.* 35, 37

RAG *Retrieval-Augmented Generation.* 21, 69

- ReAct** *Reasoning and Acting.* 7, 8, 11, 25–27, 32–34, 40, 48, 50, 56, 57, 60, 63, 65, 67
- REST** *Representational State Transfer.* 40, 56
- Sam** *Robot(Smart Academic Mentor and Guide).* 38, 43, 51, 53, 54
- SAR** *Socially Assistive Robotics.* 22, 24
- SE** *Smart Environment.* 7, 8, 17, 18, 22, 24, 28, 38, 48, 56, 60
- SkE** *Skill Error.* 35, 37
- SMART-llm** *Smart Multi-Agent Robot Task Planning.* 31, 34
- SR** *Success Rate.* 36
- STT** *Speech-to-Text.* 22, 45
- SyE** *Erro de Sintaxe.* 35, 37
- TCR** *Task Completion Rate.* 36
- TTS** *Text-to-Speech.* 22
- UE** *Erro Não Registrado.* 35, 37
- VLM** *Visual Language Model.* 70

Lista de tabelas

Tabela 2.1 – Sequência de ações mostram como ReAct pode localizar corretamente a faca, limpá-la e colocá-la em uma bancada	32
Tabela 2.2 – Tipos de Erros do Sistema MRED (WAN <i>et al.</i> , 2025)	37
Tabela 3.1 – Tipos de Solicitações e Complexidade no Sistema Multiagente	39
Tabela 3.2 – Matriz de Mapeamento Emocional	52
Tabela 4.1 – Resumo das métricas por categoria de tarefa	62
Tabela 4.2 – Categorias e passos ideais para diferentes tipos de tarefas	62
Tabela 4.3 – Resumo das taxas de sucesso por categoria de tarefa para os três modelos avaliados.	64
Tabela 4.4 – Comparação de desempenho entre modelos de linguagem	65

Lista de Siglas

API *Application Programming Interface.* 45, 46, 56

ASR *Average Success Rate.* 36

ASRS *ASR with Syntax Errors.* 36, 54

CE *Constrains Error.* 35, 37

CoT *Chain-of-Thought.* 24–27, 32, 33, 35, 60

GCR *Goal Condition Recall.* 36

HRI *Human Robot Interaction.* 20, 22–24, 29, 56

IA *Artificial Inteligence.* 18

IoRT *Internet of Robotic Things.* 7, 8, 19, 22, 50, 67

IoT *Internet of Thing.* 7, 8, 17, 18, 22, 26, 28, 30, 38, 44, 46, 67

LH *Long-Horizon.* 7, 8, 31, 33, 39, 64, 65, 67

LLM *Large Language Model.* 7, 8, 17–20, 22–36, 38–41, 43–46, 48, 50, 52–56, 59, 60, 65–67, 69, 70

MLDT *Multi-Level Decomposition Task planning.* 27, 30, 33, 34

MRED *Multi-Robot collaboration Error Diagnostic.* 35, 36

MTP *Meta-Task Planning.* 27, 30, 34

Only_SyE *Apenas Erro de Sintaxe.* 35, 37

PDDL *Planning Domain Definition Language.* 7, 8, 27, 30, 31, 34, 50

pHRC *Physical Human-Robot Collaboration.* 24

PIE *Planning Error.* 35, 37

PoE *Position Error.* 35, 37

RAG *Retrieval-Augmented Generation.* 21, 69

- ReAct** *Reasoning and Acting.* 7, 8, 11, 25–27, 32–34, 40, 48, 50, 56, 57, 60, 63, 65, 67
- REST** *Representational State Transfer.* 40, 56
- Sam** *Robot(Smart Academic Mentor and Guide).* 38, 43, 51, 53, 54
- SAR** *Socially Assistive Robotics.* 22, 24
- SE** *Smart Environment.* 7, 8, 17, 18, 22, 24, 28, 38, 48, 56, 60
- SkE** *Skill Error.* 35, 37
- SMART-llm** *Smart Multi-Agent Robot Task Planning.* 31, 34
- SR** *Success Rate.* 36
- STT** *Speech-to-Text.* 22, 45
- SyE** *Erro de Sintaxe.* 35, 37
- TCR** *Task Completion Rate.* 36
- TTS** *Text-to-Speech.* 22
- UE** *Erro Não Registrado.* 35, 37
- VLM** *Visual Language Model.* 70

Sumário

1	Introdução	17
1.1	Apresentação do tema e contextualização do problema	17
1.2	Objetivos	18
1.2.1	Objetivo Geral	18
1.2.2	Objetivos Específicos	19
1.3	Organização do Trabalho	19
1.3.1	Estrutura da Dissertação	20
2	Estado da Arte	22
2.1	Interação Humano-Robô com Sistemas Multiagentes	22
2.2	<i>Large Language Models</i>	24
2.2.1	Estratégias de Raciocínio: Zero-shot, Few-shot e Chain-of-Thought	26
2.2.2	ReAct: Integração de Raciocínio e Ação em LLMs	26
2.2.3	MLDT: Árvores de Decisão Multinível para Planejamento Hierárquico	27
2.2.4	MTP: Planejamento de Tarefas em Múltiplos Passos com LLMs	27
2.2.5	PDDL: Planejamento Formal e sua Superação via LLMs	27
2.3	Aplicação de LLMs em Sistemas Robóticos	27
2.4	Planejamento e decomposição de Tarefas em Sistemas Multiagentes	30
2.5	Metodos de Avaliação	34
3	Metodologia	38
3.1	Cenários Testados e Tarefas Avaliadas	38
3.2	Implementação dos Agentes e Modelos	41
3.3	Pipeline de Integração com LLM	44
3.3.1	<i>Context Generator</i>	50
3.3.2	<i>Task Planner</i>	50
3.4	Interação Não Verbal com o Robô SAM	51
3.5	Critérios de Avaliação	54
4	Resultados e Discussão	56
4.1	Experimentos	56
4.2	Erros	57
4.3	Taxa de Sucesso	61
4.4	Análise e Discussões	65
5	Conclusões	67
6	Trabalhos Futuros	69

1 Introdução

1.1 Apresentação do tema e contextualização do problema

A linguagem e a análise do conteúdo de diálogos desempenham um papel central na compreensão das intenções dos usuários que buscam assistência. Assim, mesmo uma conversa simples ou a interpretação de uma solicitação pode ser endereçada por uma frota de robôs sociais ou dispositivos *Internet of Thing* (IoT), que cooperam para oferecer respostas eficientes. Pesquisas como a de (BROXVALL *et al.*, 2006), que introduziram o conceito de *Ecology of Physically Embedded Intelligent Systems*, demonstram a evolução da robótica na cooperação entre agentes físicos para a execução de atividades diárias. Além disso, essas pesquisas ampliam o escopo da investigação, incorporando a necessidade de inteligência e cognição na tomada de decisões, com aplicações potenciais em Ambientes Industriais, Ambientes Domésticos e outros cenários. Com as inúmeras funcionalidades disponibilizadas pelos *Large Language Models* (LLMs), surge uma nova perspectiva para integrar a robótica e sistemas avançados de inteligência artificial.

Os *Smart Environments* (SEs) são ambientes onde frotas de robôs são integradas para fornecer assistência a usuários. Conforme apresentado no artigo (CORADESCHI; SAFFIOTTI, 2006), a visão futura da robótica prevê uma integração cada vez maior entre robôs, humanos e ambientes inteligentes, operando de maneira colaborativa e simbiótica. Nesse contexto específico, existe uma classificação que distingue os agentes com base em seus papéis e habilidades: (a) Agentes Cognitivos, responsáveis pelo raciocínio e planejamento de tarefas; (b) Agentes Físicos, que executam ações no ambiente, como robôs móveis, manipuladores, dispositivos IoT e humanoides; e (c) Agentes Humanos, que interagem diretamente com o sistema e podem emitir comandos ou solicitar assistência. Essa classificação permite estruturar um cenário de rotina diária para este projeto e definir atividades comuns para o atendimento aos usuários.

No contexto do projeto SE, desenvolvido no âmbito do Smartness (UNICAMP FAPESP, 2022), buscamos explorar soluções inovadoras com o apoio da FAPESP, UNICAMP e Ericsson. O objetivo é desenvolver aplicações que incorporem robótica em ambientes industriais e de saúde. As pesquisas sobre a cooperação entre robôs têm demonstrado resultados satisfatórios para a execução de ações específicas de interação com o ambiente. Essas atividades são tratadas como tarefas de alto nível, em vez de meras ações de baixo nível relacionadas ao controle dos atuadores dos robôs.

Considerando trabalhos como (GIL *et al.*, 2023) e outros que serão incluídos

na revisão bibliográfica, verificamos que a decomposição de missões para múltiplos robôs com habilidades heterogêneas abrange aspectos como transporte de objetos, deslocamento entre cômodos de uma casa, interação com dispositivos e conversação com usuários. No entanto, mesmo com essas abordagens, percebemos que ampliar as capacidades para múltiplos agentes físicos (robôs e dispositivos IoT) em SEs ainda apresenta desafios, principalmente na decomposição hierárquica de um objetivo geral, ou seja, na interpretação da solicitação do usuário por um sistema multiagente.

Embora existam abordagens para planejamento de tarefas e decomposição de missões para múltiplos robôs com diferentes habilidades, o gerenciamento de agentes físicos heterogêneos (robôs móveis, manipuladores e dispositivos IoT) ainda não foi suficientemente explorado. Isso inclui a coordenação de capacidades distintas, como manipulação, mobilidade e assistência a dispositivos IoT, sendo planejadas e gerenciadas por LLMs com base em uma decomposição orientada por capacidades. Dado que os agentes físicos (robôs e dispositivos IoT) possuem características heterogêneas, a interação com o ambiente se dá por meio de diferentes habilidades, como locomoção (*move_to()*), diálogo (*say()*) e interação com objetos (*do()*).

Esse cenário exige novas abordagens para o processamento de dados, a fim de permitir um planejamento eficiente de tarefas desenvolvidas pelos Agentes Cognitivos, focados na tomada de decisões para o planejamento de ações. Modelos LLMs aplicados em redes gerativas têm mostrado avanços na solução parcial de problemas relacionados à análise de dados e tomada de decisões, utilizando a taxonomia da solicitação (WAKE *et al.*, 2023). Dessa forma, um pedido pode ser decomposto hierarquicamente em Objetivo → Tarefa → Subtarefa → Ações → Subações, organizando as ações em sequências distribuídas entre os agentes físicos. Esse modelo amplia a análise contextual do ambiente, incluindo informações sobre os agentes, e melhora a eficiência do planejamento de tarefas.

1.2 Objetivos

1.2.1 Objetivo Geral

O presente trabalho propõe a implementação de um *pipeline* capaz de compreender as solicitações dos usuários e planejar tarefas em um cenário de SE, combinando capacidades de *Artificial Intelligence* com LLM e tarefas de alto nível em Robótica. O sistema considera habilidades heterogêneas de Agentes Físicos, incluindo robôs e dispositivos IoT, e utiliza Agentes Cognitivos para o planejamento de ações.

1.2.2 Objetivos Específicos

1. Implementar um ambiente virtual e/ou real para simular interações entre agentes físicos (robôs, IoT) e agentes cognitivos, permitindo validar o comportamento do sistema em cenários assistivos.
2. Desenvolver um sistema de armazenamento contextual para registrar, consultar e atualizar em tempo real informações sobre o ambiente, agentes e objetos, possibilitando que os LLMs utilizem esse contexto durante o planejamento e a execução das tarefas.
3. Desenvolver um módulo de planejamento que utiliza LLMs como agentes cognitivos para decompor solicitações em tarefas e subtarefas, coordenando a execução distribuída entre agentes físicos.
4. Projetar estratégias automáticas de diagnóstico e resolução de erros no ciclo de planejamento-execução, utilizando *feedback* dos agentes físicos e validações sintáticas e semânticas das ações propostas.
5. Implementar mecanismos para analisar as intenções do usuário com base em interações verbais ou textuais frequentes, permitindo aos LLMs interpretar comandos, preferências e objetivos de forma contextualizada.
6. Criar um módulo para monitorar e verificar automaticamente a execução das tarefas atribuídas aos agentes físicos, avaliando se os planos foram concluídos com sucesso ou requerem replanejamento.
7. Desenvolver um sistema de avaliação e classificação das ações executadas, com base em métricas como taxa de sucesso, completude dos planos e robustez da execução em diferentes tipos de tarefas.

1.3 Organização do Trabalho

Este trabalho propõe uma arquitetura cognitiva que integra modelos de linguagem de última geração LLMs com agentes físicos e virtuais em um ambiente IoRT, com foco na adaptabilidade, replanejamento dinâmico e interação conversacional multimodal.

A principal contribuição deste estudo reside na combinação inédita de três elementos:

- Uso do paradigma ReAct (razão + ação) com LLMs para permitir replanejamento iterativo em tarefas complexas e ambientes parcialmente observáveis;

- Coordenação entre agentes cognitivos e físicos, com controle real sobre dispositivos IoT simulados e integração de eventos de sensor;
- Implementação de um módulo de geração e verificação de contexto que adapta a resposta do sistema às condições e falhas detectadas em tempo real.

Embora existam trabalhos que exploram o uso de **LLMs** em ambientes simulados, a proposta aqui apresentada integra de forma coesa o pipeline de percepção, raciocínio e ação, com suporte a *feedback* e ciclos de correção, em uma única arquitetura funcional e reprodutível.

1.3.1 Estrutura da Dissertação

Capítulo 1 Apresenta uma visão geral da pesquisa, contextualizando o problema a ser investigado no âmbito dos Ambientes Inteligentes e da Robótica Assistiva com **LLMs**. Além disso, são definidos os objetivos gerais e específicos do trabalho, ressaltando a importância da integração entre agentes físicos e cognitivos em sistemas multiagentes.

Capítulo 2 Reúne os fundamentos teóricos e os principais trabalhos relacionados ao tema, incluindo conceitos sobre *Human Robot Interaction (HRI)*, aplicações de **LLMs** na robótica, estratégias de planejamento e decomposição de tarefas, bem como métodos de avaliação. Este capítulo estabelece a base conceitual e tecnológica que sustenta a proposta.

Capítulo 3 Descreve de forma detalhada o método adotado para a implementação do sistema proposto, abrangendo a modelagem dos agentes, a construção do pipeline cognitivo com **LLMs**, os cenários de simulação utilizados e os critérios de avaliação. Também são explicados os componentes técnicos envolvidos na orquestração entre agentes físicos, cognitivos e humanos.

Capítulo 4 Apresenta os experimentos realizados e os resultados obtidos, incluindo métricas quantitativas (como taxa de sucesso e número de passos) e análise qualitativa dos erros mais comuns. São discutidas comparações entre diferentes modelos **LLMs** (como GPT-3.5 e GPT-4o) e analisada a eficácia do pipeline proposto em tarefas de complexidade crescente.

Capítulo 5 Sistematiza as principais conclusões do estudo, relacionando os resultados obtidos com os objetivos estabelecidos no Capítulo 1. Também são destacados os avanços alcançados na integração entre **LLMs** e sistemas robóticos multiagente, bem como as limitações encontradas durante a implementação.

Capítulo 6 Propõe direções para pesquisas futuras, considerando oportunidades de aprimoramento do sistema proposto. Incluem-se sugestões como a incorporação

de sistemas proativos de assistência, fusão multimodal para interpretação emocional, uso de *Retrieval-Augmented Generation (RAG)* para consultas semânticas, e integração com interfaces em realidade aumentada para enriquecer a interação homem-robô.

2 Estado da Arte

Este capítulo apresenta os conceitos fundamentais para a compreensão dos *Smart Environments* (SEs), os tipos de robôs e dispositivos IoT aplicados nesta pesquisa, bem como a *Human Robot Interaction* (HRI) e o papel dos LLMs na otimização e no planejamento de sistemas multiagentes em ambientes inteligentes baseados em SEs, particularmente no contexto de IoRT e sua interação com os usuários.

2.1 Interação Humano-Robô com Sistemas Multiagentes

A HRI é um campo multidisciplinar essencial que investiga formas eficazes, seguras e naturais de interação entre humanos e sistemas robóticos, especialmente em contextos onde conforto, empatia e eficiência são prioritários (TAPUS *et al.*, 2007). Dentro desse escopo, destaca-se a *Socially Assistive Robotics* (SAR), voltada à prestação de assistência social e emocional, com foco no bem-estar do usuário, apoio à reabilitação e suporte cognitivo personalizado. A atuação da SAR baseia-se em interações sociais não necessariamente físicas, valorizando a comunicação empática e a resposta adaptativa dos robôs às necessidades humanas, sobretudo em ambientes domésticos (MCCOLL *et al.*, 2016; LUPERTO *et al.*, 2023).

No contexto da Robótica Assistiva, a capacidade dos robôs de reconhecer e interpretar estados afetivos humanos é um componente crítico. Para isso, são utilizados métodos autônomos de detecção afetiva que abrangem reconhecimento facial, análise postural, interpretação de sinais vocais e monitoramento fisiológico. Além das emoções básicas universais — como alegria, tristeza, raiva e surpresa — sistemas multimodais buscam identificar emoções mais complexas ao combinar diversos canais de comunicação, promovendo uma interação rica, sensível e altamente adaptativa (MCCOLL *et al.*, 2016). A comunicação verbal ocupa papel central nessa interação, sustentada por tecnologias de ponta como Google *Speech-to-Text* (STT), *Text-to-Speech* (TTS) e serviços avançados como ElevenLabs.¹ Nos últimos anos, serviços como o ElevenLabs tornaram-se referência no domínio de TTS neural, destacando-se por sua capacidade de gerar vozes sintéticas expressivas com baixa latência e qualidade comparável à voz humana. Essa evolução tem impacto direto na aceitação social de robôs assistivos, pois melhora a empatia percebida na interação verbal. Tais soluções permitem diálogos naturais entre humanos e robôs,

¹ O ElevenLabs é uma plataforma de síntese de voz neural avançada que utiliza modelos de deep learning para gerar fala hiper-realista com variações prosódicas e expressivas. Diferente de sistemas tradicionais de TTS, permite adaptar o tom, a velocidade e a emoção da fala com alta fidelidade, suportando múltiplos idiomas e perfis vocais customizáveis.

além de identificar nuances emocionais na voz — como ansiedade ou frustração — cruciais para gerar respostas adequadas e empáticas. Em situações de **HRI**, considera-se que os robôs com capacidades de interação para assistência verbal e percepção do ambiente têm aplicações como monitoramento e interações de mobilidade que desempenham um papel relevante em aplicações de reabilitação, como mostra a Figura 2.1 (a) (FEIL-SEIFER; MATARIC, 2005; LUPERTO *et al.*, 2023).

De forma complementar, a comunicação não verbal desempenha função estratégica na construção de uma relação empática e eficaz. Gestos, expressões faciais e sinais fisiológicos são integrados por meio de abordagens multimodais robustas, como o modelo Hume AI, que possibilita aos robôs detectar e reagir a estados emocionais com base em dados vocais e visuais combinados. É assim que, com técnicas de *Machine Learning*, busca-se associar 21 expressões faciais a significados específicos para enriquecer o contexto da comunicação. Essa competência fortalece a confiança e a aceitação dos sistemas em ambientes assistivos e sociais mostrados na Figura 2.1 (b) (KELTNER *et al.*, 2019; BAIRD *et al.*, 2022).



(a)



(b)

Figura 2.1 – Exemplos de reconhecimento multimodal na interação humano-robô (**HRI**):
a) Robô assistivo oferecendo apoio físico a um usuário durante a reabilitação (TAPUS *et al.*, 2007), b) Hume AI, sistema de inteligência artificial reconhecendo emoções humanas por meio da análise facial, segundo o modelo da (Hume AI, 2025)

O reconhecimento multimodal de emoções contribui significativamente para tornar a **HRI** mais empática, adaptativa e intuitiva. Segundo (TAPUS *et al.*, 2007), dotar os robôs da capacidade de interpretar emoções os torna socialmente mais inteligentes, o que facilita interações mais naturais e sensíveis aos estados emocionais dos usuários.

Outro canal relevante de interação é a comunicação textual, que se mostra vantajosa em cenários com ruído ambiental elevado ou preferências por interações silenciosas. A integração de **LLMs** com plataformas como Telegram e WhatsApp permite que robôs compreendam e respondam a comandos textuais de forma eficiente. Um exemplo

emblemático é o Henn-na Hotel, no Japão, onde robôs assistivos interagem com hóspedes de forma autônoma e personalizada, graças à fusão entre sistemas robóticos e suporte linguístico digital (REIS *et al.*, 2020).

A colaboração física entre humanos e robôs, conhecida como *Physical Human-Robot Collaboration* (pHRC), é outro componente essencial. Envolve o uso de robôs colaborativos equipados com sensores e atuadores de última geração, possibilitando operações seguras em ambientes compartilhados. Técnicas de aprendizado de máquina e controle adaptativo garantem que essas interações, que vão de tarefas domésticas a aplicações industriais, ocorram de maneira eficiente e precisa (OGENYI *et al.*, 2019).

Em conclusão a Robótica Assistiva e a Interação Humano-Robô constituem um campo em franca expansão, integrando tecnologias avançadas para promover uma interação empática, segura e adaptativa. A evolução contínua em reconhecimento emocional multimodal, comunicação verbal, não verbal e textual, bem como a colaboração física segura e eficiente, representa avanços significativos para sistemas robóticos mais integrados e aceitos na vida cotidiana das pessoas, especialmente em contextos sociais e assistivos, onde a empatia e precisão das interações são essenciais (MCCOLL *et al.*, 2016) (KELTNER *et al.*, 2019) (LUPERTO *et al.*, 2023).

2.2 *Large Language Models*

Os *Large Language Models* são arquiteturas avançadas baseadas em redes neurais profundas, especialmente nos modelos do tipo *transformer*, capazes de processar e gerar linguagem com alto grau de coerência, contextualização e adaptabilidade. Essas capacidades vêm revolucionando diversas aplicações, incluindo compreensão de linguagem natural, geração de conteúdo, tradução automática e, de forma particularmente relevante para este trabalho, planejamento de ações, tomada de decisão e controle de agentes autônomos em sistemas robóticos e multiagentes (DRIESS *et al.*, 2023; VEMPRALA *et al.*, 2024).

No contexto de sistemas robóticos, os *LLMs* destacam-se por sua habilidade de traduzir comandos em linguagem natural em sequências estruturadas de ações, permitindo que robôs e dispositivos inteligentes executem tarefas com maior autonomia e flexibilidade. Essa capacidade reduz significativamente as barreiras de *HRI*, fornecendo interfaces mais acessíveis e intuitivas para usuários finais — aspecto essencial em aplicações de *SAR* e *SEs* (WAN *et al.*, 2025; HU *et al.*, 2024).

Diversas pesquisas recentes exploraram técnicas específicas que potencializam as capacidades dos *LLMs*, como a *Chain-of-Thought* (CoT), que facilita o raciocínio com-

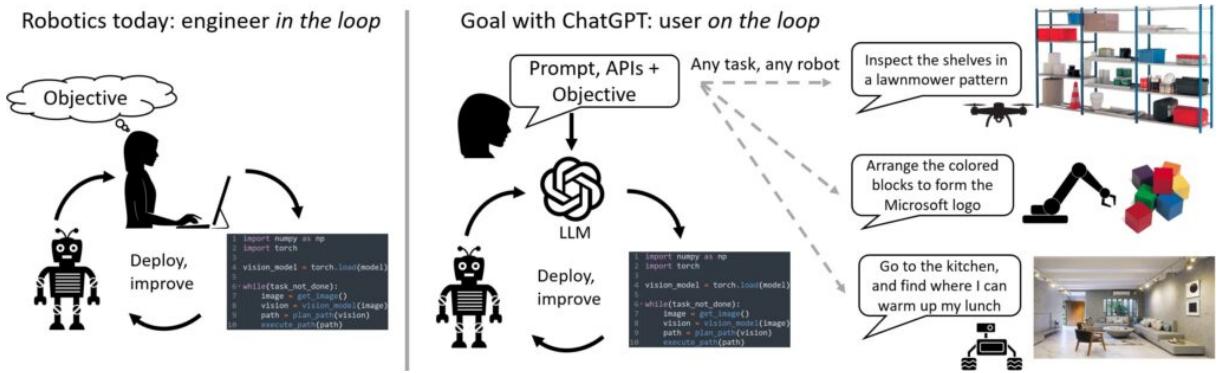


Figura 2.2 – Robótica com ChatGPT: Controle por Linguagem Natural ([VEMPRALA et al., 2024](#)).

plexo ao dividir problemas difíceis em etapas intermediárias explícitas. Essa abordagem aumenta significativamente a capacidade dos modelos para executar tarefas que requerem múltiplas etapas lógicas, essenciais para o planejamento robótico e multiagente ([YAO et al., 2023](#)), ([ZHANG et al., 2024a](#)).

Avanços recentes têm aprofundado o uso de técnicas complementares aos **LLMs**, como o **CoT**, que busca explicitar o raciocínio lógico por meio da decomposição de problemas complexos em etapas intermediárias. Essa estratégia melhora substancialmente o desempenho dos modelos em tarefas que exigem raciocínio multietapas, como a proposta por ([VEMPRALA et al., 2024](#)) que associa como os pipelines robóticos requeriam um *engineer in the loop*. No entanto, o objetivo com ChatGPT é ter um usuário *potentially non-technical* que conta com um **LLM** pensando em alto nível sobre os comandos de linguagem de controle robótico, como mostra a Figura 2.2 ([YAO et al., 2023](#); [ZHANG et al., 2024a](#)).

O enfoque **CoT** tem como objetivo fortalecer a capacidade de raciocínio dos **LLMs** mediante a geração explícita de passos intermediários antes de alcançar uma ação ou conclusão ([WEI et al., 2022](#)). No domínio da robótica, essa técnica permite que o modelo desdobre tarefas complexas em subprocessos lógicos que orientam o agente passo a passo. Ferramentas como LangChain viabilizam essa estratégia por meio da construção de cadeias de *prompts* encadeadas, onde a saída de cada etapa (raciocínio) serve como entrada para a próxima (ação). Essa abordagem permite a integração eficiente de **CoT** com **LLMs** como GPT-3.5, GPT-4 ou Llama em fluxos de planejamento modular para robôs autônomos ([HUANG et al., 2024](#)).

Complementarmente, a técnica **ReAct** estende o paradigma **CoT** ao integrar raciocínio lógico com execução iterativa de ações no ambiente ([WEI et al., 2022](#)). Diferentemente do **CoT**, que atua apenas na fase de planejamento, o **ReAct** propõe um ciclo contínuo de observação, raciocínio e ação. Nesse padrão, o **LLM** analisa o estado atual do ambiente, deduz os próximos passos, executa uma ação e atualiza seu modelo com base no

novo estado resultante. Esse ciclo dinâmico é especialmente útil para lidar com ambientes incertos ou em constante mudança (YAO *et al.*, 2023).

Tanto **CoT** quanto **ReAct** podem ser implementados com *frameworks* como LangChain, que oferece módulos para definir "ferramentas" (ações físicas disponíveis ao robô), "memória" (histórico do ambiente) e "controladores" (LLMs que orquestram o comportamento do agente). Alternativas como AutoGen também possibilitam a criação de arquiteturas multiagentes conversacionais que adotam esse padrão iterativo, permitindo maior adaptabilidade e coordenação entre agentes heterogêneos (AMATRIAIN, 2024; WU *et al.*, 2023b).

2.2.1 Estratégias de Raciocínio: Zero-shot, Few-shot e Chain-of-Thought

LLMs podem operar com diferentes estratégias de prompting. No contexto desta proposta, destacam-se três paradigmas: *zero-shot*, *few-shot* e *Chain-of-Thought CoT*.

- O zero-shot prompting permite que o modelo execute uma tarefa sem exemplos prévios, útil em interações rápidas e diretas com o usuário.
- O few-shot prompting apresenta ao modelo poucos exemplos, melhorando o desempenho em tarefas específicas e contextualizadas.
- O **CoT** prompting encoraja o modelo a gerar uma sequência de passos intermediários de raciocínio, o que é valioso para tarefas que exigem decomposição lógica, como o planejamento de ações em ambientes complexos (ZHANG *et al.*, 2024b)

Na presente dissertação, essas abordagens fornecem a base para o agente cognitivo interpretar solicitações do usuário e dividi-las em subtarefas compatíveis com agentes físicos e sensores, aproximando o modelo da cognição artificial.

2.2.2 ReAct: Integração de Raciocínio e Ação em LLMs

O *framework* **ReAct** combina o raciocínio deliberativo do LLM com a execução de ações no ambiente externo. Essa abordagem alterna entre "pensamento" (ex. gerar hipóteses, planos, análises) e "ações" (ex. buscar informações, interagir com APIs, acionar sensores) (ZHANG *et al.*, 2024b).

A arquitetura proposta nesta dissertação inspira-se diretamente no **ReAct**, ao permitir que o módulo cognitivo **LLM** planeje ações, consulte estados internos (via memória ou banco contextual) e atue em coordenação com agentes físicos e sensores **IoT**.

2.2.3 MLDT: Árvores de Decisão Multinível para Planejamento Hierárquico

A técnica *Multi-Level Decomposition Task planning* (MLDT) permite representar planos hierárquicos que consideram múltiplos níveis de decisão. Essa estrutura oferece vantagens em cenários onde múltiplos agentes com capacidades diferentes cooperam para atingir metas complexas (WU *et al.*, 2024).

Embora o presente trabalho não implemente MLDT diretamente, ele compartilha a motivação de planejamento em múltiplos níveis — desde a interpretação semântica da intenção até a execução distribuída entre os agentes. A estrutura modular de decisão pode ser útil em trabalhos futuros, particularmente para ampliar a escalabilidade do sistema.

2.2.4 MTP: Planejamento de Tarefas em Múltiplos Passos com LLMs

O *Meta-Task Planning* (MTP) explora a habilidade dos LLMs em decompor uma meta complexa em etapas executáveis. Embora ainda em desenvolvimento como técnica autônoma, tem conexão direta com abordagens como CoT e ReAct, onde a sequência de subtarefas é inferida e conduzida de forma autônoma.

No contexto desta dissertação, o módulo de planejamento cognitivo utiliza princípios semelhantes ao MTP para gerar planos que são posteriormente distribuídos e executados pelos agentes físicos. Ainda que o nome “MTP” não seja referenciado diretamente, a prática está incorporada à lógica de decomposição presente no pipeline.

2.2.5 PDDL: Planejamento Formal e sua Superação via LLMs

O *PDDL* é uma linguagem formal amplamente usada para descrever domínios e problemas de planejamento. Apesar de ser poderoso, requer modelagem rígida e conhecimento formal especializado (ZHANG *et al.*, 2024b).

O uso de LLMs neste trabalho representa uma alternativa flexível ao PDDL, permitindo descrever metas e condições iniciais em linguagem natural, com posterior conversão implícita em planos. Ainda assim, a comparação com o PDDL ajuda a evidenciar as vantagens em termos de acessibilidade, adaptabilidade e tempo de modelagem.

2.3 Aplicação de LLMs em Sistemas Robóticos

Sistemas robóticos são fundamentais na implementação de soluções inteligentes e eficientes em diferentes contextos — desde aplicações industriais até ambientes domésticos e assistivos. Essas plataformas envolvem distintas categorias de agentes físicos, como

robôs móveis, manipuladores, robôs colaborativos e dispositivos inteligentes baseados em IoT. Quando corretamente integradas, essas categorias ampliam significativamente as capacidades operacionais, perceptivas e interativas dos sistemas. Esta seção apresenta os principais tipos de agentes físicos explorados neste projeto, demonstrando como os LLMs contribuem para sua coordenação e otimização em tarefas reais (PARADA, 2025; WAKE *et al.*, 2023; VEMPRALA *et al.*, 2024; WAN *et al.*, 2025).

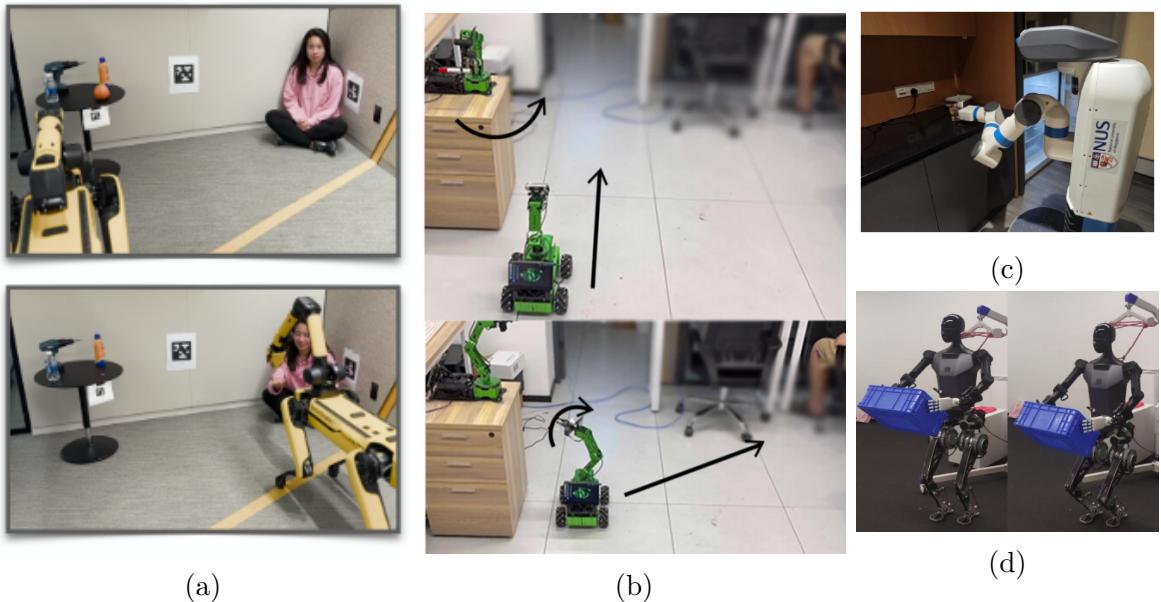


Figura 2.3 – Robôs heterogêneos empregados em sistemas multiagente: (a) robô móvel, (b) robô terrestre, (c) robô manipulador e (d) robô humanoide.

Large Language Models em Navegação

A navegação autônoma é uma competência essencial para robôs que operam em ambientes dinâmicos, como os SEs. Nesse contexto, os LLMs têm demonstrado eficácia ao gerar rotas e sequências de movimento que respeitam restrições espaciais, operacionais e colaborativas. Um exemplo marcante é o uso do *robot drone*, um robô aéreo especializado em deslocamento em ambientes domésticos ou logísticos. Esse agente é capaz de executar tarefas como “*go_to_location(fridge_location)*”, integrando comandos de linguagem natural à geração de planos de alto nível, coordenando suas ações com robôs terrestres em um mesmo plano de execução multiagente (HU *et al.*, 2024; WAN *et al.*, 2025). Outro caso relevante é o *robot dog* como mostra a Figura 2.3 (a), um robô terrestre limitado ao solo, que destaca a importância dos LLMs em considerar restrições físicas do ambiente ao planejar trajetórias viáveis (PENG *et al.*, 2024; HU *et al.*, 2024).

Já o *robot car/arm* que mostra a Figura 2.3 (b), que combina mobilidade com capacidade de manipulação, representa um desafio adicional: os LLMs precisam orques-

trar movimentos de navegação com ações coordenadas de interação física, respeitando restrições de espaço e sincronização entre capacidades (HU *et al.*, 2024).

Large Language Models em manipulação

A manipulação de objetos é uma das habilidades mais complexas e centrais em robótica assistiva, industrial e de serviços. Os LLMs têm sido empregados com sucesso para gerar sequências de ações em tarefas como pegar, posicionar, montar ou entregar objetos. O exemplo do *robot arm*, um manipulador doméstico, ilustra bem como os LLMs são capazes de decompor instruções de alto nível, como preparar uma omelete, em subtarefas operacionais: “*pick (egg)*”, “*crack (egg)*”, “*place (bread)*” (WAN *et al.*, 2025; DRIESS *et al.*, 2023; VEMPRALA *et al.*, 2024). Essa decomposição permite criar fluxos de trabalho coordenados com outros agentes em ambientes compartilhados.

Em cenários colaborativos, observa-se a interação entre um manipulador e um drone, onde o primeiro entrega fisicamente um item para o segundo transportar. Isso evi-dencia como os LLMs viabilizam sequências integradas de manipulação interagente, indo além da simples execução individual para formar verdadeiras cadeias robóticas heterogêneas (WAN *et al.*, 2025).

Large Language Models em humanoides

Robôs humanoides são particularmente adequados para integração com LLMs devido ao seu design centrado na interação social e multifuncionalidade. Esses robôs têm se destacado em tarefas que combinam percepção multimodal, comunicação empática e execução de ações em ambientes humanos. O robô Pepper, amplamente utilizado em pesquisas de HRI, exemplifica essa integração: equipado com sensores faciais e vocais, ele é capaz de interpretar expressões e comandos de voz, planejar tarefas de manipulação leve e mediar interações sociais com usuários (MYLAB, 2019; PARADA, 2025; SUN *et al.*, 2025).

Modelos como o Trinity vão além, sendo capazes de adaptar, em tempo real, gestos, diálogos e posicionamento espacial com base no estado emocional e nas intenções do usuário. Isso é particularmente relevante em contextos assistivos, nos quais o robô precisa demonstrar empatia, flexibilidade e sensibilidade à dinâmica social (SUN *et al.*, 2025). Adicionalmente, a inclusão de agentes como o *humanoid robot* — que representa o próprio ser humano no sistema multiagente (em interação com robôs móveis, manipuladores e outros humanoides) — reforça a visão de que os sistemas multiagentes com LLMs como agentes cognitivos possibilitam a incorporação do usuário como um agente ativo na interação (HU *et al.*, 2024).

2.4 Planejamento e decomposição de Tarefas em Sistemas Multi-agentes

O avanço dos *Large Language Models*, como GPT-4, Gemini, entre outros, trouxe novas perspectivas para a robótica assistiva. Esses modelos têm a capacidade de interpretar comandos em linguagem natural, raciocinar sobre tarefas complexas e decompor objetivos em subtarefas executáveis por robôs físicos. Em ambientes colaborativos e assistivos, essa habilidade de traduzir linguagem para ações práticas é essencial, especialmente quando há múltiplos agentes heterogêneos, como robôs móveis, manipuladores e dispositivos IoT. A incorporação de *LLMs* confere maior flexibilidade e adaptabilidade, tornando a interação humano-robô mais natural e robusta, além de facilitar a execução de tarefas em ambientes dinâmicos e imprevisíveis (WAN *et al.*, 2025).

Métodos de Planejamento de Tarefas

Diversas metodologias têm sido propostas para estruturar o planejamento de tarefas com *LLMs*, cada uma com abordagens distintas voltadas a diferentes cenários. O método **MLDT** de (WU *et al.*, 2024) destaca-se por sua decomposição hierárquica em três níveis: instruções de alto nível são desdobradas em subobjetivos, subtarefas e, por fim, ações primitivas que podem ser executadas por robôs. Essa abordagem permite que o **LLM** contextualize comandos com níveis crescentes de detalhe, favorecendo a escalabilidade e eficiência em tarefas de longo horizonte. Já a metodologia **MTP** adota um sistema distribuído com agentes do tipo “*manager*” e “*executors*”, que dividem tarefas complexas em metatarefas, coordenadas de forma colaborativa entre múltiplos agentes cognitivos. Isso possibilita paralelismo e reduz o tempo total de execução (ZHANG *et al.*, 2024a).

Por outro lado, os métodos simbólicos tradicionais, como os baseados em *Planning Domain Definition Language* (**PDDL**) que se mostra na Figura 2.4, enfrentam limitações quando integrados diretamente a *LLMs* em ambientes dinâmicos. **PDDL** é uma linguagem de codificação padronizada utilizada para representar problemas de planejamento. Neste paradigma, os problemas são definidos mediante arquivos principais de domínio e o arquivo de problema. No arquivo de domínio define-se o ambiente no qual o planejamento será realizado, descrevendo as possíveis condições ou fatos do mundo e também se definem as ações. O arquivo de problema define uma instância específica do domínio que se deseja resolver. Então, o planejador **PDDL** toma como entrada o arquivo de domínio Figura 2.4 (a) e o arquivo de problema Figura 2.4 (c) e busca uma sequência de ações (um plano) que conduz a um estado em que todas as condições do objetivo são verdadeiras. O planejador frequentemente busca o plano mais curto (SINGH *et al.*, 2024).

Como apontam (ZHANG *et al.*, 2024b) e (ZHOU *et al.*, 2024), a dificuldade em modelar a complexidade e variabilidade do mundo real — especialmente em tarefas de LH com interdependências entre agentes — compromete sua adaptabilidade. Esses métodos também apresentam rigidez, tornando difícil sua reconfiguração rápida diante de novos cenários, além de demandarem forte conhecimento de domínio para sua formalização.

<pre>(define (domain blocksworld-4ops) (:requirements :strips) (:predicates (clear ?x) (on-table ?x) (arm-empty) (holding ?x) (on ?x ?y))</pre>	<pre>(:action pickup :parameters (?ob) :precondition (and (clear ?ob) (on-table ?ob) (arm-empty)) :effect (and (holding ?ob) (not (clear ?ob)) (not (on-table ?ob)) (not (arm-empty))))</pre>
(a) PDDL domain description	(b) PDDL action operator
<pre>(define (problem BW-rand-3) (:domain blocksworld-4ops) (:objects b1 b2 b3) (:init (arm-empty) (on-table b1) (on b2 b3) (on b3 b1) (clear b2)) (:goal (and (on b3 b2) (on b1 b3))))</pre>	<pre>(unstack b1 b3) (putdown b1) (unstack b3 b2) (stack b3 b1) (pickup b2) (stack b2 b3) ; cost = 6 (unit cost)</pre>
(c) PDDL problem	(d) PDDL plan

Figura 2.4 – PDDL visão geral (SINGH *et al.*, 2024). O planejamento PDDL e elementos, predicados de estado (a), operadores de ação (b), condição de meta (c) e plano(d)

Apesar dessas limitações, a metodologia TWOSTEP propõe uma integração híbrida, combinando inferência linguística dos LLMs com planificadores simbólicos baseados em PDDL. Nesse modelo, os LLMs geram subobjetivos em linguagem natural, que são posteriormente formalizados para resolução simbólica. Essa estratégia tem se mostrado eficaz em sistemas multiagente, otimizando a alocação de tarefas e melhorando a eficiência da equipe robótica (SINGH *et al.*, 2024).

Outra proposta relevante é o *Smart Multi-Agent Robot Task Planning* (SMART-lm), que incorpora *prompts* programáticos em Python para guiar os LLMs na decomposição de tarefas, formação de coalizões e alocação entre robôs heterogêneos. Essa integração entre programação e linguagem natural revela-se crucial para alinhar os planos gerados pelos modelos com a execução prática dos agentes físicos (KANNAN *et al.*, 2024).

Esse processo de raciocínio explícito facilita a identificação das ações e dependências necessárias para alcançar o objetivo global. Por exemplo, em tarefas como cozinhar, o modelo pode deduzir que é necessário reunir os ingredientes antes de iniciar

o preparo. Essa compreensão de dependências é crucial para a coordenação eficiente entre agentes, especialmente quando múltiplos robôs colaboram para uma mesma missão (ZHANG *et al.*, 2024a).

Além disso, a estrutura lógica proporcionada pelo CoT permite que a alocação de subtarefas seja feita com base nas capacidades específicas de cada agente. Ao entender quais habilidades são requeridas em cada etapa, o sistema pode distribuir as tarefas de forma otimizada, maximizando a eficiência da equipe e a chance de sucesso da execução (KANNAN *et al.*, 2024).

Essa técnica também serve como base para abordagens híbridas como o ReAct, que intercala passos de raciocínio com ações físicas. Dessa forma, faz-se uma representação da Tabela 2.1 como base para que o agente ReAct comece seu processo de resolução de problemas sobre o ambiente ALFWorld. Nesta sequência, distingue-se a informação enviada ao LLM: a informação inicial sobre os objetos disponíveis e sua localização, o que permite ao agente começar a raciocinar, planejar e atuar para alcançar o objetivo. Dessa forma, realiza uma observação inicial como entrada para o primeiro passo de "Thought" e começa a gerar pensamentos sobre como proceder. Os pensamentos iniciais implicam decompor o objetivo em subobjetivos e planejar a primeira ação. A presença e os objetos influenciam no raciocínio de senso comum do agente ReAct, logo a sequência de "Observação-pensamento-ação" se repete ao longo da trajetória ReAct.

Tabela 2.1 – Sequência de ações mostram como ReAct pode localizar corretamente a faca, limpá-la e colocá-la em uma bancada

Etapa	Ação
Planejamento Inicial	O agente decide que precisa encontrar uma faca, limpá-la e colocá-la sobre a bancada.
Busca pelo objeto (faca)	O agente verifica os seguintes locais em ordem: <ul style="list-style-type: none"> • go to cabinet 1 – apenas uma tigela • go to cabinet 2 – fechado • go to cabinet 3 – apenas uma garrafa de vidro • go to cabinet 4 – apenas uma caneca • go to cabinet 5, 6 – fechados • go to drawer 1-3 – todos fechados • go to countertop 1 – não contém faca

Etapa	Ação
	• go to countertop 2 – encontra knife 1
Coleta do objeto	O agente pega a knife 1 da countertop 2.
Limpeza da faca	O agente se dirige até a sinkbasin 1 e limpa a faca.
Colocação do objeto	O agente se dirige à countertop 1 e coloca a faca limpa sobre ela.

Apesar das vantagens do **CoT** e **ReAct**, ambos apresentam limitações. Por exemplo, a criação de exemplos aumentados com cadeias de pensamento pode exigir engenharia especializada. Em modelos pequenos, o **CoT** pode inclusive reduzir o desempenho, ao gerar cadeias de raciocínio fluentemente escritas, porém logicamente incorretas (**WEI et al., 2022**).

O mesmo se aplica ao **ReAct**: se a fase de raciocínio não estiver bem direcionada ou se as ações executadas não fornecerem informações úteis para o próximo ciclo, o desempenho do agente pode se tornar ineficiente ou errático (**YAO et al., 2023**).

Ainda assim, a integração de **LLMs** em robótica assistiva enfrenta desafios persistentes, como a necessidade de adaptação a ambientes dinâmicos e a otimização do consumo computacional em plataformas embarcadas. O desenvolvimento de arquiteturas modulares, uso de modelos mais leves e aplicação de ciclos de *feedback* são estratégias promissoras para superar essas limitações (**WU et al., 2023a**).

Decomposição de Tarefas

A decomposição de tarefas constitui uma estratégia fundamental para a eficácia do planejamento robótico baseado em **LLMs**, especialmente diante de tarefas complexas e de **Long-Horizon**. Essa técnica consiste em dividir uma tarefa principal em subtarefas menores e mais manejáveis, o que facilita tanto o planejamento quanto a execução por parte dos agentes robóticos. Diferentes abordagens têm explorado essa decomposição com perspectivas complementares, oferecendo soluções inovadoras para lidar com ambientes dinâmicos e heterogêneos (**WU et al., 2024; ZHANG et al., 2024a; VEMPRALA et al., 2024; KANNAN et al., 2024; YAO et al., 2023**).

A evolução da decomposição de tarefas em sistemas multiagentes com **LLMs** reflete uma transição de abordagens sequenciais clássicas para modelos mais dinâmicos e adaptativos. O **MLDT** exemplifica uma decomposição hierárquica rigorosa, estruturada em três níveis (objetivo, subtarefa e ação), oferecendo clareza e organização, mas com menor flexibilidade frente a mudanças em tempo real (**WU et al., 2024**).

Por sua vez, o **MTP** introduz uma visão descentralizada e cognitiva, na qual múltiplos agentes autônomos dividem e gerenciam tarefas colaborativamente. Essa abordagem favorece a escalabilidade e a execução paralela, porém demanda estratégias complexas de coordenação e gerenciamento entre os agentes envolvidos (**ZHANG et al.**, 2024a).

A metodologia TWOSTEP representa uma tentativa de equilibrar raciocínio natural com formalização simbólica. Ao permitir que os **LLMs** gerem metas em linguagem natural, que são então formalizadas em **PDDL** para execução precisa, cria-se um *pipeline* híbrido promissor. Contudo, essa estrutura é menos flexível em ambientes dinâmicos, pois depende de pré-modelagens que rapidamente podem se tornar obsoletas (**SINGH et al.**, 2024).

Em contraste, métodos como o **SMART-llm** e o **ReAct** colocam a adaptabilidade e reatividade como princípios centrais. O **SMART-llm** utiliza prompts em Python para decompor tarefas diretamente em código executável, promovendo uma interação fluida entre raciocínio e programação. Já o **ReAct** explora um ciclo contínuo de raciocínio e ação, no qual o modelo reavalia continuamente o ambiente antes de cada novo passo, aumentando a robustez frente a cenários imprevisíveis (**KANNAN et al.**, 2024; **YAO et al.**, 2023).

Finalmente, observa-se uma tendência promissora de convergência entre essas metodologias. A integração entre decomposição hierárquica (**MLDT**), coordenação distribuída (**MTP**), formalização simbólica (TWOSTEP) e adaptabilidade em tempo real (**ReAct** e **SMART-llm**) aponta para a construção de arquiteturas híbridas e resilientes. Trabalhos como (**SILVA; RALHA, 2023**) reforçam essa necessidade ao evidenciar que sistemas robóticos operando em ambientes variáveis — como ambientes domésticos ou linhas de produção — exigem mecanismos de replanejamento e adaptação contínuos.

2.5 Metodos de Avaliação

A avaliação de sistemas baseados em **LLMs** para planejamento robótico tem como principal objetivo validar a capacidade dos modelos em traduzir comandos em linguagem natural em planos de ação robustos, eficientes e executáveis por agentes físicos em ambientes reais ou simulados. Os estudos revisados concentram-se na eficácia da decomposição de tarefas, na adaptabilidade a cenários dinâmicos e na escalabilidade em contextos multiagente. Modelos como **MLDT**, **MTP** e **SMART-llm** são frequentemente avaliados quanto à sua robustez em domínios diversos e à sua integração com planejadores simbólicos, módulos de navegação e manipuladores robóticos, em tarefas como assistência, transporte e colaboração física.

Avaliação de Modelos de Linguagem

Embora muitas avaliações se concentrem em métricas quantitativas, o comportamento, o raciocínio e a interação dos modelos em sistemas multiagentes também exigem análise qualitativa. A técnica *Chain-of-Thought* permite observar e interpretar os passos intermediários de raciocínio dos modelos, fornecendo uma lente qualitativa sobre sua coerência lógica. Mesmo quando medidas quantitativamente por desempenho, as cadeias de raciocínio geradas — como em (WEI *et al.*, 2022) — são analisadas por plausibilidade e alinhamento com o objetivo da tarefa. Em contextos robóticos, como nos trabalhos de (WAKE *et al.*, 2023) e (VEMPRALA *et al.*, 2024), são realizadas inspeções visuais das sequências de ações e estimativas ambientais para verificar se os planos gerados são viáveis e realistas, indo além dos números para avaliar a sensatez dos comportamentos.

Segundo (WAN *et al.*, 2025) e (SILVA; RALHA, 2023), avaliações humanas são empregadas para identificar tipos de erro através de métricas como *Multi-Robot collaboration Error Diagnostic* (MRED), que são um conjunto de métricas de avaliação desenhadas especificamente para analisar o rendimento de LLMs em planejamento e coordenação. MRED aborda a lacuna de avaliar os comandos abstratos e a execução física neste tipo de tarefas, além de permitir uma análise estruturada do rendimento do planejador. Na identificação de erros, MRED diferencia entre os tipos de erros de sintaxe: *Erro de Sintaxe* (SyE) e *Apenas Erro de Sintaxe* (Only_SyE) mostrados na Tabela 2.2 das respostas geradas pelos LLMs. Mas também apresentam outro tipo de erros relacionados com habilidades não vistas no ambiente, erros espaciais no posicionamento e erros não registrados *Erro Não Registrado* (UE), *Position Error* (PoE), *Planning Error* (PlE), *Skill Error* (SkE) e *Constrains Error* (CE). Estes erros foram explicados na Tabela 2.2, mas isso exige julgamento qualitativo sobre se uma ação está correta ou representa uma falha específica. Essa abordagem também se aplica à análise do comportamento colaborativo entre agentes, como ilustrado em (ZHANG *et al.*, 2024a), onde diferentes papéis (*manager, supervisor, executors*) são avaliados quanto à sua interação, coordenação e resolução de conflitos.

A interpretabilidade das decisões também é um critério qualitativo central: compreender por que um agente escolheu determinada ação ou como um plano foi formado é crucial para diagnóstico, depuração e aumento da confiança nos sistemas multiagente (ZHANG *et al.*, 2024a).

Avaliação qualitativa e métricas

Diversas métricas quantitativas têm sido empregadas para avaliar o desempenho de LLMs em tarefas de planejamento:

- *Success Rate* e *Task Completion Rate* medem a taxa de sucesso global na execução de tarefas (KANNAN *et al.*, 2024).
- *Efficiency* e *Executability* quantificam o tempo e a taxa de ações executáveis previstas nos planos (ZHANG *et al.*, 2024b; KANNAN *et al.*, 2024).
- *Goal Condition Recall* e *Robot Utilization* avaliam a adequação do plano aos objetivos propostos e o aproveitamento dos agentes.

Em contextos multiagente, métricas especializadas como *Average Success Rate*, *ASR with Syntax Errors* e *Multi-Robot collaboration Error Diagnostic* são amplamente adotadas (WAN *et al.*, 2025):

- *Average Success Rate (ASR)* calcula a proporção de execuções totalmente corretas (sem erros), sobre o total de tentativas;
- *ASR with Syntax Errors (ASRS)* amplia essa métrica ao incluir também execuções com apenas erros de sintaxe json (Only-SyE), considerando que a lógica de planejamento permanece válida;

Ambas são medidas em percentuais e refletem a precisão dos planos gerados sob diferentes níveis de dificuldade.

- *Multi-Robot collaboration Error Diagnostic* oferece um diagnóstico estruturado dos erros em sistemas distribuídos, categorizando as falhas por tipo, frequência e impacto.

Métricas Complementares e Avaliações Avançadas

Outras métricas têm sido utilizadas em tarefas específicas:

- *Average Rewards* e *Expenses* avaliam o custo computacional e a eficiência energética de execução.
- *Room Segmentation Precision/Recall/Purity*, usadas em tarefas de mapeamento e navegação, medem a precisão da segmentação espacial por *LLMs* em ambientes simulados (HUANG *et al.*, 2024; HONERKAMP *et al.*, 2024).

Além disso, técnicas qualitativas como inspeção visual, *role-plays*, validação humana e *feedback* iterativo com linguagem natural foram amplamente utilizadas. Essas estratégias permitem observar se os planos gerados são adaptados às preferências dos usuários e como os *LLMs* respondem a correções contextuais. Estudos como (WAN *et al.*,

Tabela 2.2 – Tipos de Erros do Sistema MRED ([WAN et al., 2025](#))

Tipo de Erro	Descrição
<i>Erro de Sintaxe (SyE)</i>	Erros na sintaxe JSON do sistema quando também ocorrem outros tipos de erros.
<i>Apenas Erro de Sintaxe (Only_SyE)</i>	Casos onde o único erro presente é um erro de sintaxe JSON.
<i>Erro Não Registrado (UE)</i>	Uso de elementos não registrados dentro do sistema. Divide-se em subtipos: <ul style="list-style-type: none"> • UE_robot: Robô não registrado. • UE_skill: Habilidade não registrada. • UE_obj: Objeto não registrado. • UE_pos: Posição não registrada.
<i>Position Error (PoE)</i>	Confusão ou imprecisões relacionadas com o posicionamento de elementos.
<i>Planning Error (PlE)</i>	Implementação de planos pouco práticos ou inviáveis.
<i>Skill Error (SkE)</i>	Aplicação incorreta de funções de habilidade. Inclui subtipos: <ul style="list-style-type: none"> • SkE_pos: Uso incorreto de parâmetros de posição. • SkE_obj: Uso incorreto de parâmetros de objeto.
<i>Constrains Error (CE)</i>	Descumprimento das restrições operacionais, como exceder a capacidade de carga do robô.

2025) e (HONERKAMP et al., 2024) evidenciam a importância dessas abordagens para ajustar modelos e refinar seus comportamentos em sistemas de planejamento colaborativo.

3 Metodologia

Este capítulo descreve a metodologia adotada para o desenvolvimento e validação de um sistema de robôs de serviço baseado em *Large Language Models*. O trabalho caracteriza-se como uma pesquisa aplicada, de natureza experimental, realizada em ambientes simulados, com foco em aplicações assistivas e colaborativas em contextos multiagente.

A pesquisa é classificada como aplicada, pois objetiva a resolução de problemas concretos relacionados à interação humano-robô, planejamento de tarefas e colaboração entre agentes físicos heterogêneos em ambientes inteligentes. Ao integrar *LLMs* com sistemas robóticos, busca-se não apenas explorar conceitos teóricos, mas propor soluções tecnológicas viáveis para contextos reais, como assistência doméstica e automação colaborativa.

3.1 Cenários Testados e Tarefas Avaliadas

A validação do sistema proposto foi realizada em um conjunto de cenários simulados que representam ambientes domésticos inteligentes, como cozinha, sala de estar e quarto. Esses cenários foram modelados em ambiente simulado utilizando a plataforma CoppeliaSim e representação real do cenário, com o objetivo de replicar situações comuns de interação humano-robô em contextos assistivos (ROHMER *et al.*, 2013). Cada ambiente foi composto com agentes físicos e cognitivos, incluindo robôs móvel, manipulador, humanoides e dispositivos IoT, todos coordenados por um sistema central baseado em *Large Language Models*.

As tarefas testadas foram classificadas em diferentes categorias, conforme seu nível de complexidade e o tipo de interação requerida:

- **Comunicação e diálogo:** Interações que não exigem ações físicas, mas demandam acesso a informações contextuais do SE, respondidas por meio de comunicação verbal pelo robô *Robot(Smart Academic Mentor and Guide)* (Sam) ou textual por canais como Telegram.
- **Assistência e resolução de problemas:** Comandos ambíguos, repetidos ou incompletos, nos quais o *framework* solicita reformulação, esclarecimentos ou orientações adicionais do usuário para continuar o planejamento.

- **Controle de dispositivos IoT:** Ações que envolvem o envio de comandos para dispositivos inteligentes (ligar a Smart TV, incrementar o som, desligar a TV) por meio de integração com serviços como Google Assistant.
- **Interação entre agentes físicos:** Solicitações que requerem a colaboração coordenada entre agentes móveis e manipuladores, com execução sequencial e sincronizada de ações físicas.
- **Tarefas estendidas de múltiplas etapas LH:** Requisições mais complexas que exigem a combinação de manipulação física e controle de IoT, realizadas por diferentes agentes ao longo de múltiplas etapas planejadas.

Cada tarefa foi estruturada em uma sequência de ações esperadas, detalhada na [Tabela 3.1](#), permitindo verificar não apenas a completude e correção das ações realizadas, mas também a fluidez das interações entre os agentes e o usuário. O sistema demonstrou capacidade de replanejar dinamicamente, adaptando os planos às condições do ambiente, tais como obstáculos, falhas temporárias ou ausência de objetos essenciais.

Tabela 3.1 – Tipos de Solicitações e Complexidade no Sistema Multiagente

Tipo de Solicitação	Complexidade	Tipo
Tarefas não atribuídas	Baixa (1 passos)	T1
Comunicação e diálogo	Baixa (2 passos)	T2
Assistência e resolução de problemas	Baixa (2 - 3 passos)	T3
Controle de dispositivos IoT	Média (3 passos)	T4
Interação entre agentes físicos	Média (4 passos)	T5
Tarefas estendidas com múltiplas etapas	Alta (5 passos)	T6

Esses cenários demonstraram a capacidade robusta e flexível da abordagem proposta, validando que diferentes tipos de solicitações, tanto verbais quanto textuais, podem ser eficazmente transformadas em planos concretos e executáveis por um conjunto variado de agentes físicos, todos orquestrados pelos agentes cognitivos e [LLMs](#).

Como apresentado na [Figura 3.1](#), a interação entre o usuário e o sistema pode ocorrer de duas formas principais. Se o usuário estiver próximo a um robô com capacidade de captar voz, a comunicação é verbal, permitindo interações diretas e intuitivas. Caso contrário, a interação ocorre por meio textual, utilizando aplicativos como o Telegram. A partir desse ponto, a consulta (*query*) do usuário é recebida e concatenada com informações extraídas da base de dados (informações de contexto), que fornece conteúdos relevantes sobre os agentes, o mapa e os objetos presentes no ambiente. Essa base de dados considera identificadores específicos, conexões entre salas e características distintivas de objetos e estruturas, formando um conjunto contextual.

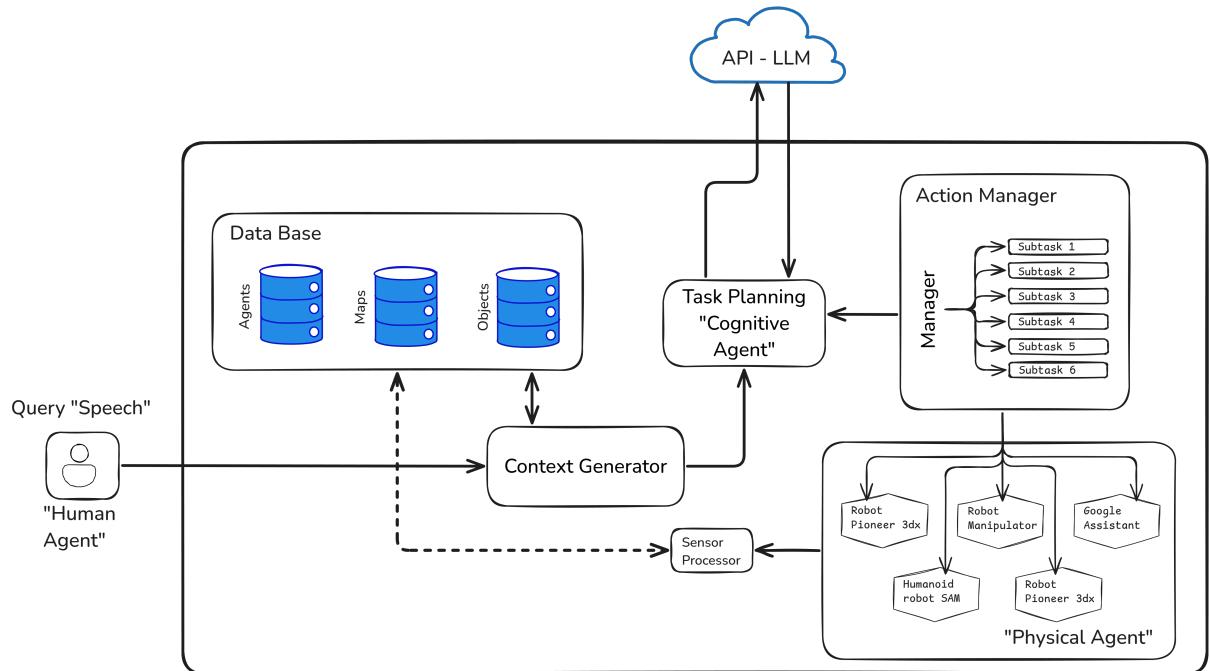


Figura 3.1 – Interação com o *Framework*, representação do Agente Humano interagindo com o sistema completo.

Essas informações são então armazenadas no módulo *Context Generator*, que as envia ao *Task Planner* — responsável por identificar o objetivo do usuário e decompor a tarefa em uma sequência de ações. Essa decomposição é realizada por um *LLM*, que opera como núcleo de raciocínio do Agente Cognitivo. O agente envia uma requisição ao modelo, que retorna um plano de ações. Caso haja erros decorrentes de alucinações ou inconsistências no raciocínio gerado, o sistema realiza uma nova consulta ao *LLM* solicitando correções e replanejamento, aplicando a técnica *ReAct* como padrão para o ciclo iterativo de planejamento.

Uma vez obtida uma sequência coerente de subtarefas que atendem ao objetivo, estas são armazenadas no *Action Manager*, que organiza e distribui as ações para os Agentes Físicos. Cada agente executa as ações designadas, e as alterações resultantes são registradas na base de dados, atualizando o estado do ambiente, incluindo agentes, mapas e objetos afetados. Durante todo esse processo, utiliza-se um servidor *Representational State Transfer (REST)* para estabelecer a comunicação entre o Agente Humano, o Agente Cognitivo e os Agentes Físicos. Esse modelo de comunicação permite padronizar a interação entre os diferentes módulos do sistema multiagente, conforme ilustrado na Figura 3.1.

3.2 Implementação dos Agentes e Modelos

A arquitetura proposta neste trabalho integra diferentes tipos de agentes físicos — robôs móveis, manipuladores, humanoides e dispositivos IoT — coordenados por um modelo de [LLM](#) e um agente cognitivo. Essa integração permite a conversão eficiente e precisa de comandos em linguagem natural em ações concretas, executáveis por agentes heterogêneos em ambientes simulados.

Principais Agentes Físicos

- **Robô Móvel — Pioneer 3DX:** Responsável pela locomoção entre pontos do ambiente e pelo transporte de pequenos objetos. Possui sensores de distância, sistema de tração diferencial e arquitetura modular que permite integração com LIDAR e câmeras. Suas capacidades e limitações estão detalhadas na [Figura 3.2](#).
- **Manipulador Robótico — Kinova Gen3:** Robô fixo, especializado em preensão e manipulação de objetos, com 6 graus de liberdade. Sua leveza e precisão o tornam ideal para tarefas colaborativas. As capacidades são descritas na [Figura 3.3](#).
- **Robô Humanoide — SAM (baseado no Pepper):** Atua como mediador social, realizando interação verbal, gestual e emocional com o usuário. Possui sensores multimodais e expressividade corporal. Ver [Figura 3.5](#).
- **Dispositivos IoT — Google Nest Mini:** Integrado ao sistema como agente de controle ambiental, permite operações como ligar/desligar TV ou ajustar volume. Suas especificações encontram-se na [Figura 3.4](#).

Pioneer 3-DX

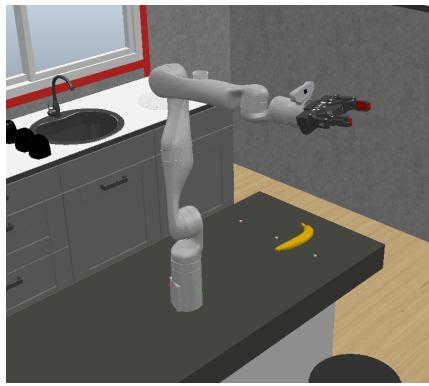


(a)

Atributo	Especificação
Material	Alumínio, pneus borracha
Peso	9 kg (carga: 17 kg)
Capacidade	Degrais: 2,5 cm; buracos: 5 cm; inclinação: 25%
Autonomia	8-10 horas
I/O	Digital/analógico, portas serial expansão

(b)

Figura 3.2 – Representação e especificações do robô Pioneer 3DX utilizado na implementação ([3DX, 2025](#))

Kinova Gen3

(a)

Atributo	Especificação
Mobilidade	Alcance máximo de 891 mm (6 DoF)
Manipulação	Especializado em preensão de objetos
Carga útil	2 kg (alcance total), 4 kg (alcance médio)
Controle	Software Kinova Kortex, controle a 1 kHz

(b)

Figura 3.3 – Representação e especificações do robô Kinova utilizado na implementação ([ROBOTICS, 2024](#))

Google Nest

(a)

Atributo	Especificação
Áudio	Alto-falante 40 mm, 3 microfones, Voice Match
Sensores	Toque capacitivo, detecção ultrassônica
Processador	ARM 64 bits, quad-core 1.4 GHz, hardware IA
Conectividade	Wi-Fi 802.11b/g/n/ac, Bluetooth 5.0, Google Cast
Peso	181 g

(b)

Figura 3.4 – Representação e especificações do Google Nest Mini utilizado na implementação ([GOOGLE, 2025](#))

SAM



(a)

Atributo	Especificação
Processador	ATOM E3845, Quad-core, 4GB RAM
Câmeras	2D e 3D para detecção e reconhecimento visual
Sensores	Lasers, IR, sonares e unidade inercial
Autonomia	12h (uso intensivo), 19h (modo espera)
Tela	Touch screen de 10.1 polegadas

(b)

Figura 3.5 – Representação e especificações do robô Pepper e simulação do Sam (MYLAB, 2019)

Integração no Ambiente Simulado

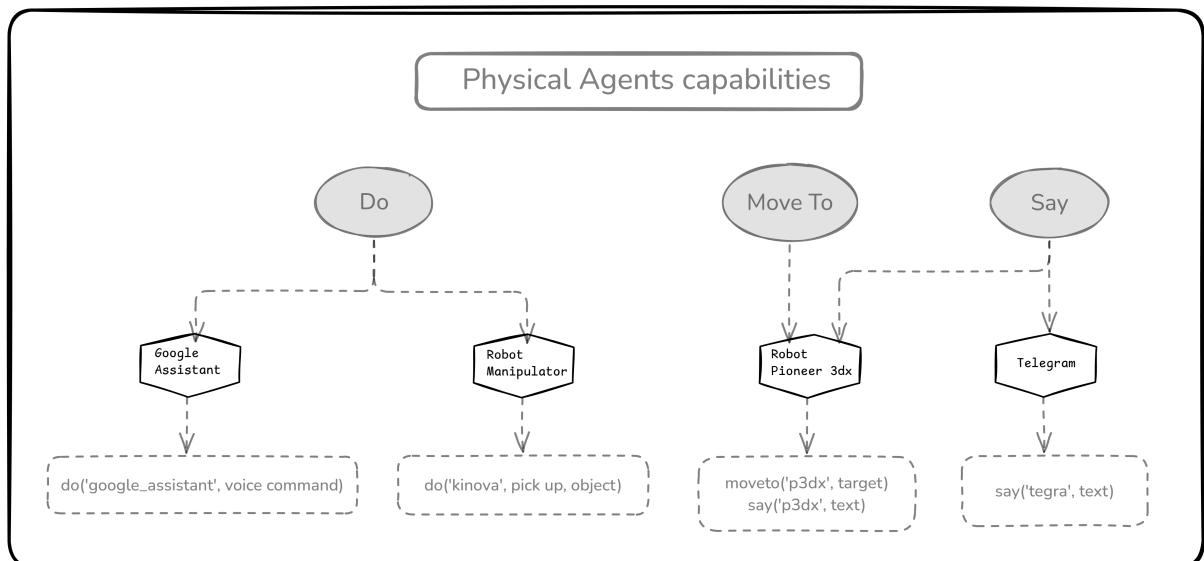


Figura 3.6 – Mapa de Ações e Limitações dos Agentes Físicos

A implementação dos agentes ocorreu em ambiente real e simulado através do software CoppeliaSim, com o controle operacional dos agentes realizado por interfaces desenvolvidas em Python. Utilizaram-se ferramentas como LangChain e *pipeline* próprio mostrado na Figura 3.7 para facilitar a comunicação eficiente com os LLMs. A compatibilidade operacional e restrições dos agentes foram cuidadosamente definidas em um mapa

de ações, consultada pelo **LLM** durante o processo de planejamento e decomposição das tarefas.

A [Figura 3.6](#) apresenta o mapa de ações e limitações dos Agentes Físicos, categorizando as capacidades operacionais de cada agente no sistema multiagente. O Robô manipulador é responsável exclusivamente por ações de preensão e manipulação, utilizando a estrutura *do()* com três parâmetros: o nome do manipulador, o tipo de ação e o objeto a ser manipulado. O robô móvel Pioneer 3-DX pode realizar duas ações distintas: movimentação no ambiente, por meio de *move_to()* com dois parâmetros: nome do robô e posição de destino, e comunicação verbal com *say()*, também com dois parâmetros: nome do robô e mensagem.

O Google Assistant é tratado como um agente físico de controle de voz no ambiente, realizando ações *do()* com dois parâmetros: nome do agente e comando de voz, possibilitando a ativação de dispositivos ou funções específicas no cenário. Já o Telegram, embora seja um canal de comunicação textual, é modelado como um agente capaz de emitir respostas, utilizando *say()* com dois parâmetros: nome do canal e mensagem de texto.

Essa organização permite que o planejador cognitivo atribua corretamente cada ação aos agentes apropriados, considerando não apenas suas capacidades técnicas, mas também a estrutura dos parâmetros exigidos por cada tipo de ação.

3.3 Pipeline de Integração com LLM

A integração entre **LLMs** e sistemas robóticos foi viabilizada por um *pipeline* modular, responsável por converter instruções em linguagem natural em planos de ação interpretáveis e executáveis por agentes físicos e dispositivos conectados. O *pipeline* ilustrado na [Figura 3.7](#) foi projetado com foco em flexibilidade, permitindo o controle de diferentes tipos de robôs e a integração com plataformas de comunicação e dispositivos IoT.

Esse *pipeline* é composto por uma sequência de módulos interdependentes, que conduzem o processo desde a entrada dos dados do usuário até a execução das ações no ambiente físico. Especificamente, ele organiza o fluxo em quatro etapas principais: (i) recepção e pré-processamento dos dados de entrada (voz, texto ou comandos via aplicativos); (ii) interpretação e raciocínio por meio de modelos **LLMs**, integrando agentes cognitivos como OpenAI e Gemini; (iii) planejamento das tarefas com base na análise contextual; e (iv) execução e comunicação das ações por meio dos agentes físicos.

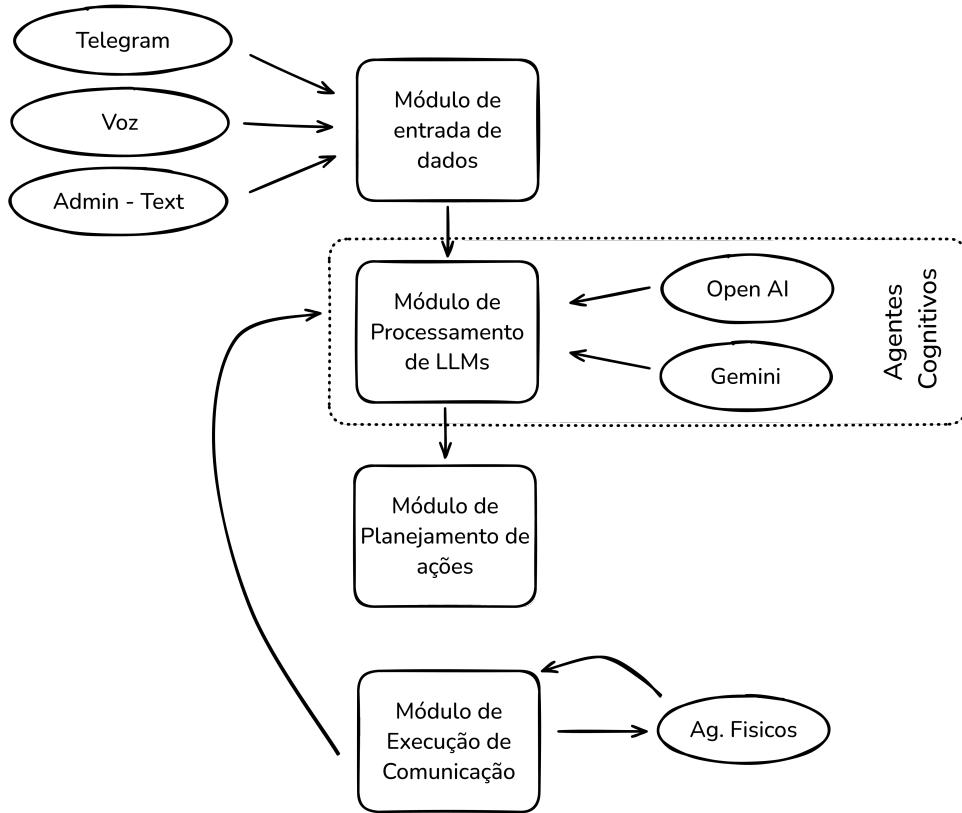


Figura 3.7 – Representação base da arquitetura modular do pipeline.

1. Recebimento da Instrução

- O sistema aceita entradas via:
- Voz: comandos orais são processados por sistemas de reconhecimento de fala Google *Speech-to-Text*;
 - Texto: por meio de interfaces como Telegram, onde o usuário envia mensagens para o sistema;
 - Interações internas: comandos disparados por eventos no ambiente simulado ou por outros agentes.

2. Interpretação semântica via LLM

O comando recebido é repassado ao **LLM** (GPT-4 e gemini-1.5-flash), utilizando o *framework*:

- Própria: baseada em uma comunicação contínua com *Application Programming Interfaces (APIs)* de ambos os modelos, para adaptar ao sistema robótico.
- LangChain: para estruturação dos *prompts*, controle de fluxo e encadeamento de raciocínio com ferramentas externas;

O modelo analisa o contexto, identifica intenções e gera um plano de alto nível, normalmente em linguagem semi-estruturada JSON, descrevendo subtarefas e parâmetros como: [moveto(robot1, positionX)], [do(robot2, pick up, object)],

```
[say(robot2, text)], [do(robot2, pick up, object)], [do(robot1, voice command)]
```

3. **Parsing e Validação do Plano** O plano gerado é processado por um módulo de interpretação que:

- Verifica a coerência sintática e lógica do plano em Sintaxe JSON;
- Realiza correlação de funções de interação com o ambiente de simulação;
- Dispara mensagens de erro e solicita reformulação do comando ao usuário em caso de inconsistências.

4. **Execução nos Agentes Físicos e IoT** Uma vez validado, o plano é executado parcialmente pelos agentes:

- Robôs móveis: controlados por comandos de navegação e rotas geradas no ambiente;
- Manipuladores: acionados para pegar, posicionar ou entregar objetos;
- Humanoides: executam ações sociais, verbais e gestuais (via motores e síntese de fala);
- Dispositivos IoT: integrados via APIs ou simulações que emulam dispositivos reais (como smart TVs, alto-falante inteligentes de Google Nest).

5. **Feedback e Atualização do Estado** Após a execução de cada etapa, o sistema:

- Registra logs de desempenho;
- Atualiza o estado do ambiente;
- Envia mensagens de retorno ao usuário, confirmando ou solicitando novas instruções;
- Realimenta o LLM com contexto atualizado.

Em comparação com a base da arquitetura modular mostrada na [Figura 3.7](#) apresentada anteriormente, a [Figura 3.8](#) oferece uma representação mais detalhada do fluxo de planejamento e execução de tarefas do agente cognitivo. Nesta figura, é possível observar de forma clara todas as etapas envolvidas desde a recepção da solicitação do usuário — seja por voz ou texto — até a finalização do processo.

O fluxo inicia com a recuperação de informações contextuais armazenadas no banco de dados, que são utilizadas pelo Agente Cognitivo para interpretar a solicitação e decompor o objetivo em subtarefas. O planejador de tarefas, integrado ao agente, define a estratégia de execução e atribui as ações aos agentes físicos apropriados. A verificação da

qualidade do planejamento é feita por meio de um validador implementado em Python, que inspeciona a estrutura sintática das ações geradas, conferindo a correção das funções e a consistência dos parâmetros com base nos dados armazenados na base de dados. Isso assegura que os comandos estejam semanticamente alinhados ao ambiente e evita a execução de ações inválidas ou sem efeito.

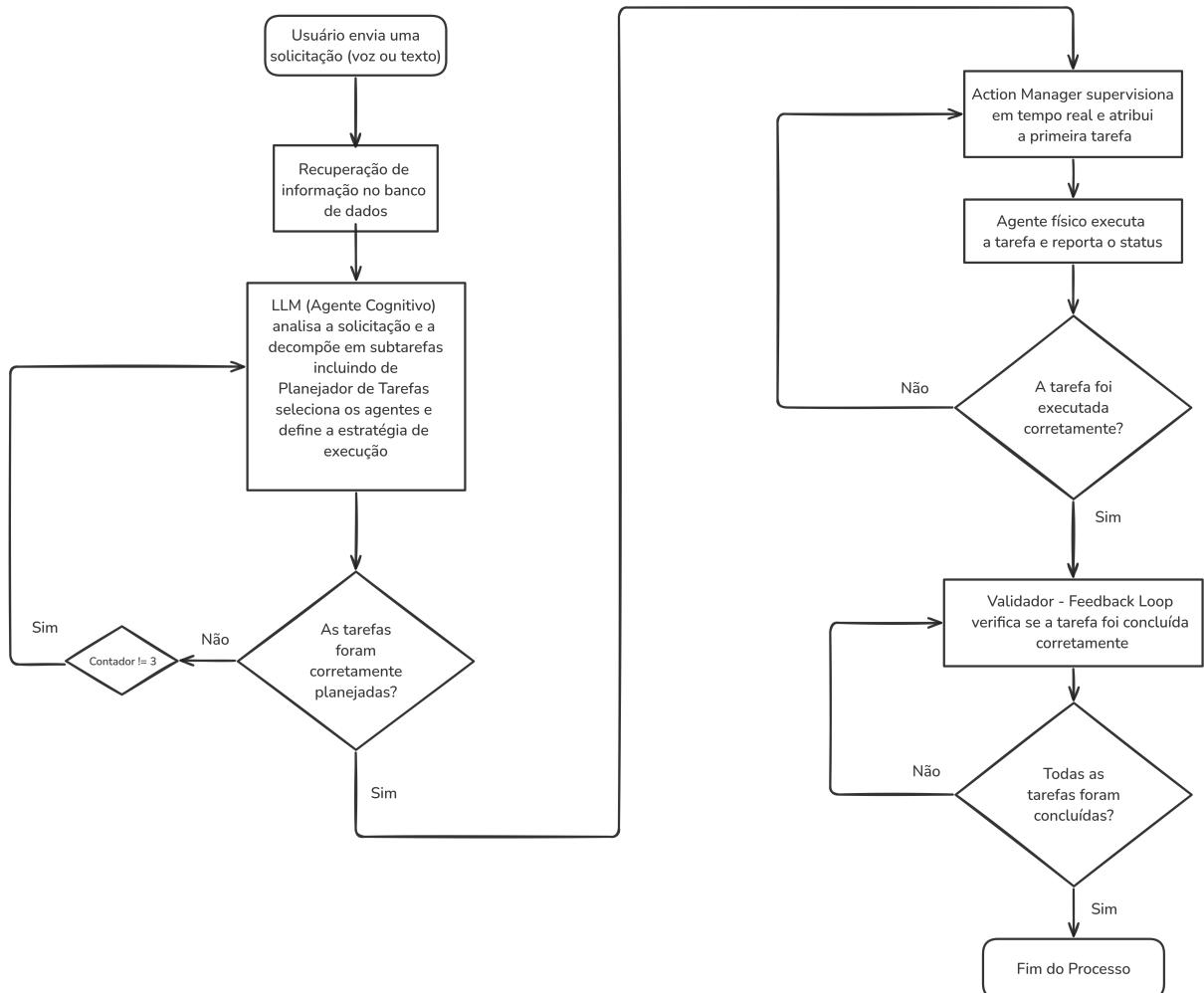


Figura 3.8 – Fluxo de Planejamento e Execução de Tarefas pelo Sistema Multiagente com Agente Cognitivo

Caso o planejamento não seja validado, o sistema ativa um ciclo de replanejamento, limitado por um contador que permite até três tentativas. Se após esse limite o erro persistir, o processo é interrompido e o sistema informa que não foi possível gerar um plano viável, solicitando uma nova reformulação da pergunta por parte do usuário humano. Uma vez validadas, as tarefas são supervisionadas em tempo real pelo *Action Manager*, que distribui as ações uma a uma para execução pelos agentes físicos. Após cada execução, um sistema de verificação avalia se a tarefa foi concluída com sucesso, possibilitando o reenvio ou replanejamento em caso de falha.

O Pseudocódigo 3.1 representa o núcleo do pipeline cognitivo do sistema ro-

bótico proposto, sendo responsável por coordenar todas as interações entre o usuário, os agentes cognitivos baseados em LLMs e os agentes físicos presentes no ambiente inteligente. Ao iniciar a execução do sistema, estabelece-se a conexão para a comunicação entre os módulos, e é inicializado um histórico de conversação que servirá como memória contextual para futuras interações. Caso seja a primeira execução, o sistema realiza uma coleta dos dados do ambiente — incluindo os agentes disponíveis, objetos e suas posições — e prepara um contexto inicial que é então adicionado ao histórico da conversa. Esse procedimento garante que o LLM tenha acesso a uma base contextual mínima e atualizada desde o início da operação.

O ciclo principal do sistema funciona de forma contínua, monitorando constantemente a entrada do usuário. Essa entrada pode ser captada por diferentes canais, com prioridade dada ao canal de voz (via microfone ou dispositivo como o Google Nest), seguido por mensagens enviadas via Telegram e, por fim, por comandos inseridos diretamente na consola de texto. Quando uma entrada é detectada, ela é processada pelo modelo de linguagem (como GPT-3.5 ou GPT-4o), o qual analisa tanto a nova solicitação quanto o histórico prévio e o contexto armazenado. A partir disso, o LLM gera uma resposta estruturada contendo um plano de ações, geralmente descrito em uma linguagem semi-estruturada como JSON, incluindo os nomes dos agentes, ações e objetos envolvidos.

Com a resposta recebida, o sistema extrai as ações propostas e inicia um ciclo interno de validação e execução. Primeiro, verifica-se a validade sintática e lógica de cada comando, assegurando que estejam de acordo com os parâmetros esperados pelo ambiente de simulação e com as capacidades operacionais dos agentes físicos. Se todas as ações forem consideradas válidas, elas são então executadas sequencialmente, e o sistema envia um resumo da execução ao usuário, encerrando esse ciclo. Caso contrário, se houver comandos inválidos, inicia-se automaticamente um processo de replanejamento, em que os erros são identificados e reportados de volta ao LLM, que tenta gerar um novo plano corrigido, aplicando a técnica ReAct — um ciclo iterativo de raciocínio e ação. Esse ciclo de replanejamento continua até que um plano viável seja obtido ou que o número máximo de tentativas seja atingido.

Ao final de cada ciclo bem-sucedido, o sistema registra os dados da interação no histórico de conversação, atualizando permanentemente a memória contextual para próximas requisições. O laço principal do sistema permanece ativo, em estado de espera, pronto para receber e processar novas instruções, assegurando que o sistema opere de forma contínua. Em síntese, o Pseudocódigo 3.1 encapsula uma arquitetura robusta e modular, que combina percepção de entrada multimodal, raciocínio semântico distribuído por LLMs e execução física por agentes especializados em SEs e dinâmicos.

Pseudocódigo 3.1 Pipeline Principal do Sistema Robótico

```

1: procedure SISTEMAROBOTICO(parametros)
2:   //Inicialização da conexão e histórico//
3:   Inicializar conexão de rede
4:   historia-conversacao ← [ ]
5:
6:   dados-ambiente ← ObterDadosAmbiente()
7:
8:   if PrimeiraExecucao() then
9:     contexto ← PrepararContexto(dados-ambiente)
10:    historia-conversacao.adicionar(contexto)
11:   end if
12:
13:   while Verdadeiro do
14:     //Obtenção da entrada com prioridade: Voz > Telegram > Texto// 
15:     entrada ← ObterEntrada()
16:
17:     if entrada ≠ Nulo then
18:       resposta ← ProcessarComLLM(entrada, historia-conversacao)
19:
20:     if resposta ≠ Nulo then
21:       acoes ← FiltrarAcoes(resposta)
22:
23:       //Ciclo de validação e execução de ações// 
24:       intercambio ← Verdadeiro
25:       while acoes.estado = "tarefa" e intercambio do
26:         acoes-validadas ← ValidarAcoes(acoes.comandos)
27:
28:         if TodasValidas(acoes-validadas) then
29:           ExecutarAcoes(acoes-validadas)
30:           EnviarResposta(acoes.resumo)
31:           intercambio ← Falso //Finaliza o ciclo de replane// 
32:         else
33:           erro ← IdentificarErros(acoes-validadas)
34:           resposta ← ReplanejamentoLLM(erro)
35:           acoes ← FiltrarAcoes(resposta)
36:         end if
37:       end while
38:
39:       SalvarHistorico(historia-conversacao)
40:     end if
41:   end if
42:
43:   Esperar()
44: end while
45: end procedure

```

3.3.1 *Context Generator*

O módulo denominado *Context Generator* é responsável por reunir e organizar as informações disponíveis do ambiente (sensores, atuadores, estado interno dos agentes, e histórico de interações) e transformá-las em um texto estruturado compreensível pelo modelo de linguagem.

Este módulo atua como uma ponte semântica entre os dados brutos do ecossistema **IoRT** e o raciocínio textual esperado pelos modelos gerativos. A saída gerada inclui variáveis contextuais, como: localização do usuário, estado dos dispositivos (por exemplo, lâmpada ligada ou desligada), tempo do dia, interações anteriores e ocorrências recentes de falha.

Funcionamento:

- Entrada: snapshot do estado do ambiente (sensores IoT, eventos e interações recentes);
- Processamento: mapeamento semântico das variáveis, preenchimento de templates contextuais e enriquecimento com expressões temporais;
- Saída: texto estruturado do tipo "O usuário está na sala, a lâmpada está ligada, e foi detectada uma falha anterior no plano.", enviado como prompt system ao **LLM**.

Essa estratégia permite ao LLM acessar o estado do mundo em linguagem natural, simulando um modelo mental dinâmico. O gerador foi implementado em Python com lógica declarativa e regras heurísticas baseadas nos sensores disponíveis.

3.3.2 *Task Planner*

O *Task Planner* é o componente responsável por interpretar a intenção do usuário e gerar, em colaboração com o **LLM**, uma sequência estruturada de ações a serem realizadas por agentes físicos ou virtuais no ambiente.

Sua implementação é inspirada na técnica **ReAct** (Reason + Act), permitindo ao modelo intercalar raciocínio textual com chamadas de ações concretas. A proposta adota uma representação textual intermediária no estilo de script, onde cada linha representa uma ação parametrizada com seus respectivos argumentos.

Diferente de abordagens baseadas em linguagens formais como o **PDDL**, que exigem a definição explícita de pré-condições e efeitos, o sistema proposto utiliza uma sintaxe orientada à execução, permitindo ao agente cognitivo gerar planos de forma mais natural e contextual. Um exemplo típico de saída gerada pelo Task Planner é:

```

moveto("p3dx", "kinova")
do("kinova", "pick up", "red gummy bear")
moveto("p3dx", "Eric")
say("p3dx", "Here is your red gummy bear.")

```

Funcionamento:

- Entrada: prompt contendo o objetivo do usuário e o contexto atual (fornecido pelo Context Generator);
- Processamento: o LLM gera um plano textual passo a passo, que é convertido para o formato de plano semiestruturado (JSON);
- Validação: o plano passa por uma verificação de coerência semântica e executabilidade (por exemplo, se os agentes físicos disponíveis podem realizar as ações propostas);
- Execução: os passos válidos são enviados sequencialmente aos agentes, com monitoramento de falhas e possibilidade de replanejamento.

O Task Planner combina raciocínio simbólico, aprendizado gerativo e regras operacionais fixas, resultando em um módulo híbrido e adaptável.

3.4 Interação Não Verbal com o Robô SAM

O **Sam** atua como agente central na mediação social entre o sistema e o usuário, oferecendo uma experiência de comunicação natural e empática. Essa interação é especialmente importante em contextos assistivos, onde o engajamento emocional e a sensibilidade social são essenciais. Para isso, foi desenvolvido um sistema que integra módulos avançados de percepção não verbal, reconhecimento emocional e resposta adaptativa.

A capacidade de **Sam** em perceber emoções é garantida pela integração com o sistema Hume AI, que analisa sinais não verbais como expressões faciais e tom de voz. Esse sistema é capaz de identificar, em tempo real, estados emocionais. As informações emocionais extraídas são incorporadas ao *pipeline* cognitivo, possibilitando ajustes contextuais e comportamentais mais refinados durante a interação.

A associação entre emoções percebidas e respostas geradas é baseada em uma matriz de mapeamento emocional, representada na [Tabela 3.2](#). Nessa matriz, cada emoção detectada corresponde a uma emoção de resposta esperada, permitindo ao robô executar

Tabela 3.2 – Matriz de Mapeamento Emocional

Emoção Detectada	Resposta Emocional	Categoría
Admiração	Amor / Feliz	Positiva
Ansiedade	Triste / Pensativo	Negativa
Alegria	Feliz	Positiva
Raiva	Furioso / Pensativo	Negativa
Calma	Amor / Feliz	Positiva
Confusão	Pensativo	Neutra
Nojo	Furioso / Triste	Negativa
Surpresa (positiva)	Feliz	Positiva
Surpresa (negativa)	Pensativo / Triste	Negativa
Tristeza	Triste	Negativa
Orgulho	Feliz / Amor	Positiva
Medo	Triste / Pensativo	Negativa

ações que ampliam a empatia e a adequação do comportamento frente ao estado emocional do usuário.

O *pipeline* de reconhecimento de expressões faciais constitui um sistema sequencial composto por múltiplas etapas interdependentes. O processo inicia-se com a aquisição da imagem, seguida pela detecção do rosto por meio de algoritmos fornecidos pela plataforma Hume AI. Após a identificação da *face*, são extraídos pontos-chave faciais (*landmarks*), que servem como base para a análise dos padrões geométricos do rosto mostrados na [Figura 3.9](#).

Esses padrões são então processados com o objetivo de identificar características específicas associadas a emoções universais, como alegria, tristeza, raiva, surpresa, medo, nojo e neutralidade emocional ([Hume AI, 2025](#)).

Quando o agente cognitivo baseado em [LLM](#) fornece o contexto da interação em andamento, a informação emocional detectada é encaminhada a um agente responsável por gerar a resposta apropriada em formato textual. Essa resposta é, então, enviada simultaneamente para o ambiente CoppeliaSim e para a plataforma Choregraphe ([POT et al., 2009](#)), mostrados na [Figura 3.10](#).

No Choregraphe, a emoção rotulada é verificada em sua biblioteca interna, sendo associada a um conjunto específico de gestos e expressões corporais pré-definidos. Dessa forma, o robô executa, de maneira sincronizada, tanto o movimento corporal quanto a reprodução do áudio correspondente, resultando em uma manifestação multimodal da emoção reconhecida, coerente com o estado emocional do usuário e com o contexto da interação.

O sistema de reconhecimento emocional segue as seguintes etapas:

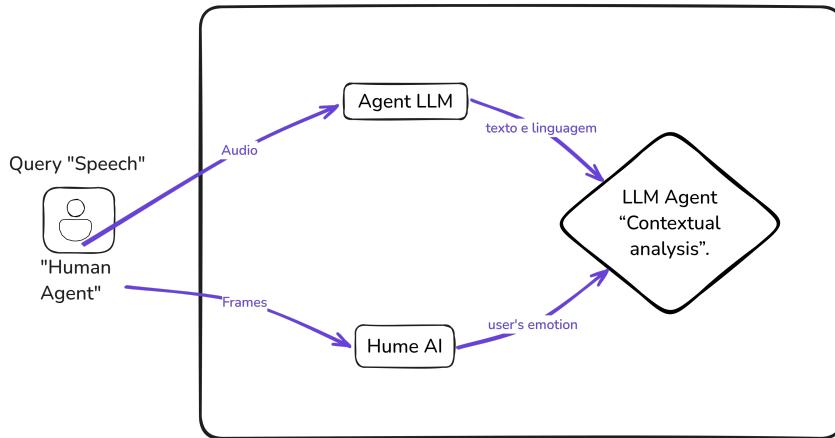


Figura 3.9 – *Pipeline* de Reconhecimento de Expressões Faciais: Fluxo de interação com *Agent LLM*

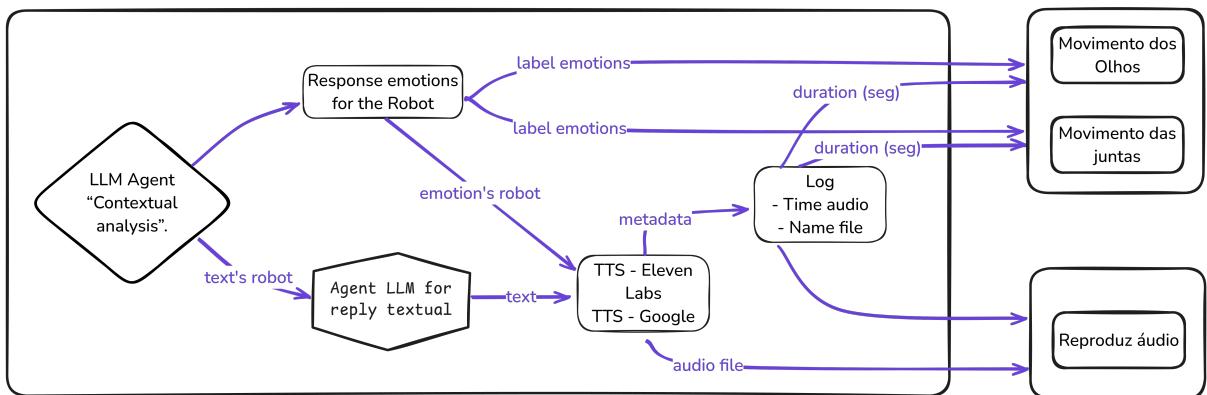


Figura 3.10 – *Pipeline* de Reconhecimento de Expressões Faciais: Fluxo de Processamento Emoções em Comunicação não verbal

- 1. Monitoramento contínuo:** O *Sam* observa e escuta continuamente o usuário através da câmera e microfone integrado;
- 2. Detecção emocional:** O sistema *Hume AI* processa as imagens e *Gemini* os áudios capturados, classificando em tempo real as emoções do usuário;
- 3. Análise contextual e emocional pelo *LLM*:** Com base nas emoções detectadas, o *LLM* realiza uma avaliação do contexto e ajusta o plano de interação, podendo modificar desde a formulação verbal até o uso de gestos e expressões não verbais;
- 4. Execução do comportamento adaptado:** O *Sam* executa as respostas ajustadas emocionalmente, realizando gestos, movimentos corporais e adaptações vocais específicas, conforme as necessidades percebidas no usuário.

Essa abordagem permite ao robô *Sam* reconhecer e reagir a sinais emocionais em tempo real, promovendo uma interação mais natural e afetiva, fortalecendo o vínculo entre humano e robô.

Ferramentas para Interação de Comunicação Não Verbal

Para operacionalizar essa interação, foram implementadas ferramentas e *frameworks* específicos que processam voz, imagem e emoções de maneira integrada:

- **Análise de Expressões Faciais:** O sistema Hume AI realiza a detecção de emoções em tempo real via conexão por *websockets*, gerando etiquetas emocionais a partir das expressões faciais detectadas.
- **Geração de Respostas com LLMs:**
 - Gemini-1.5 Flash: Utilizado para respostas curtas e diretas.
 - Gemini-1.5 Pro: Empregado em interações mais elaboradas e contextualizadas.
 - Execução paralela: Implementada via `asyncio`, permitindo lidar com múltiplas entradas simultaneamente.
- **Integração com Choregraphe e NAOqi:** O [Sam](#) é simulado no CoppeliaSim e controlado via API NAOqi ([NAOQI, 2025](#)), permitindo executar movimentos sincronizados com comandos verbais e emocionais.
- **Síntese de Voz (TTS):** A resposta de áudio é gerada com o sistema ElevenLabs, oferecendo suporte a múltiplos idiomas e entonações naturais, enriquecendo a expressividade do robô.

A combinação dessas ferramentas proporciona uma experiência de interação rica e natural, em que a comunicação não verbal contribui ativamente para a adaptação do comportamento do robô às emoções e intenções dos usuários em tempo real.

3.5 Critérios de Avaliação

A avaliação do sistema proposto baseou-se em um conjunto de critérios que medem a eficácia, a viabilidade e a robustez dos planos gerados por [LLMs](#) em cenários robóticos assistivos. Esses critérios foram aplicados tanto em testes unitários com agentes isolados, quanto em tarefas colaborativas multiagente, considerando os seguintes aspectos:

- **Executabilidade do Plano:** Verificação de se os planos podiam ser executados integralmente pelos agentes físicos, considerando erros sintáticos e semânticos, mensurado por métricas como [ASRS](#) mostradas na [seção 2.5](#).
- **Adaptabilidade e Resiliência:** Capacidade do sistema de lidar com falhas e replanejar parcialmente sem necessidade de intervenção humana mostrado [seção 2.4](#).

- **Coordenação Multiagente:** Grau de sincronização e cooperação entre agentes móveis, manipuladores e humanoides na realização de tarefas distribuídas mostrado na [seção 2.4](#).
- **Qualidade da Interação:** Clareza e naturalidade das interações textuais, verbais e não verbais, e o tempo de resposta entre o comando e a ação executada, especialmente com o robô SAM como projetos anteriores mostrado na [seção 2.3](#).

Métricas de Avaliação e Validação

O desempenho do sistema será avaliado com base em critérios quantitativos extraídos e adaptados de trabalhos anteriores, como os apresentados em ([WAN et al., 2025](#); [KANNAN et al., 2024](#)), que avaliam principalmente a eficiência na execução das tarefas. O critério fundamental é a precisão na interpretação das solicitações, ou seja, a capacidade dos [LLMs](#) em decompor corretamente as instruções recebidas e distribuí-las eficientemente entre os diferentes agentes físicos.

Também serão analisados os erros ocorridos durante a execução, incluindo falhas na sequência planejada de ações decorrentes de limitações físicas dos agentes (*hardware*) ou erros na interpretação contextual pelo modelo. Esses critérios combinados oferecem uma visão abrangente do desempenho geral do sistema, contribuindo para sua otimização contínua e assegurando sua eficácia em cenários reais e aplicações práticas.

4 Resultados e Discussão

Neste capítulo, apresentam-se os dados quantitativos e qualitativos obtidos durante os experimentos, abordando todas as questões expostas anteriormente nos Objetivos do [Capítulo 1](#) e desenvolvidas ao longo da Metodologia proposta no [Capítulo 3](#). Este processo permite detalhar e validar as proposições feitas nos capítulos anteriores.

Como a experimentação foi realizada tanto em um ambiente real quanto em um ambiente virtual, para o primeiro cenário (ambiente real), considerou-se um espaço composto por três cômodos, nos quais foram integrados quatro Agentes Físicos e dois Agentes Humanos. A estratégia de planejamento de tarefas foi conduzida por um Planejador que contém um Agente Cognitivo baseado na técnica [ReAct](#), a qual foi avaliada em comparação com diferentes modelos, incluindo LlaMa-3B, GPT-3.5 e GPT-4o.

Além disso, foi implementado um módulo de validação, responsável por confirmar o sucesso de cada tarefa executada ao término da ação de cada Agente Físico, utilizando um sistema baseado em servidor [REST](#) para comunicação entre os dispositivos do [SE](#). A proposta visa explorar novas estratégias de uso de planejadores, com o intuito de aumentar a precisão na decomposição de tarefas em sequências mais enxutas de passos. Intrinsecamente, buscou-se utilizar modelos externos via [API](#), aproveitando o baixo custo computacional ao empregar modelos pré-treinados, como os [LLM](#).

4.1 Experimentos

Nos experimentos, considerando a atuação de quatro Agentes Físicos, as ações foram distribuídas com base na [Tabela 3.1](#), que apresenta as categorias de solicitações de [HRI](#) realizadas neste Sistema Multiagente. Adicionalmente, em conformidade com as regras definidas no [Capítulo 3](#), as capacidades de cada Agente Físico foram descritas na [Figura 3.6](#), permitindo uma adequada representação das métricas de desempenho deste trabalho.

Cada solicitação foi avaliada com base em um número de passos considerados corretos mostrado na [Tabela 3.1](#), de acordo com critérios estabelecidos a partir da percepção humana. Contudo, os experimentos exploraram a diversidade e granularidade das tarefas, partindo de instruções bem definidas em um *prompt* inicial e do raciocínio contextual gerado pelo [LLM](#).

Os experimentos de planejamento, gerenciados por um Agente Cognitivo, foram realizados por meio da interação com as solicitações dos usuários, aplicando o funcio-

namento do [ReAct](#) para replanejamento dinâmico. Dessa forma, realizou-se uma comparação com o modelo LLaMA3-70B, tomando como referência os resultados apresentados por ([SILVA et al., 2024](#)), que propõem uma classificação de tarefas em *Short-term memory tasks*, *Medium-term memory tasks* e *Long-term memory tasks*. A partir desses experimentos, foi possível construir um *framework* que seleciona sequências de tarefas alinhadas às capacidades específicas dos agentes.

Para a avaliação dos resultados, foram utilizadas métricas como *success rate* e o número de passos completados (*Steps*). Embora outras variáveis relacionadas às capacidades dos Agentes Físicos possam influenciar os resultados, a métrica de *Steps* foi adotada para quantificar o número de subtarefas necessárias para atingir cada objetivo definido.

4.2 Erros

Antes de apresentar a etapa de planejamento de tarefas e os resultados bem-sucedidos, são descritos os tipos de erros mais comuns identificados durante as interações com os modelos "gpt-3.5-turbo-1106" e "gpt-4o". Nas [Figura 4.1](#) e [4.2](#), observam-se as falhas recorrentes registradas em ambos os modelos durante os experimentos. Foram classificadas seis categorias principais de erros e uma sétima categoria que engloba as "Execuções OK", ou seja, aquelas que atingiram o objetivo final da solicitação, mesmo sem completar todos os passos esperados.

Ao todo, foram realizadas 340 interações experimentais para ambos os modelos, distribuídas entre os seguintes tipos de erros:

- **Erro na Captura de Áudio:** Problemas na recepção do áudio, devido a ruído ambiental ou falhas na transcrição automática.
- **Perda de Memória Sequencial:** O modelo falha em manter corretamente as instruções anteriores na memória, comprometendo a sequência de ações necessárias.
- **Erro de Formato da Resposta:** A resposta gerada pelo modelo apresenta erros de estrutura no JSON, inviabilizando a integração com as etapas seguintes do framework e prejudicando a execução das tarefas.
- **Falha na Gestão de Usuários:** O sistema não detecta a troca de interlocutor e assume que as interações ainda pertencem ao mesmo usuário.
- **Falha no Replanejamento:** O modelo não consegue propor uma nova sequência de ações após uma falha inicial no plano.

- **Erro de Contexto e Raciocínio:** O modelo interpreta de forma inadequada a solicitação ou o contexto do ambiente.
- **Geração de Informação Incorreta:** Fenômeno conhecido como "alucinação", no qual o modelo gera dados irreais ou inconsistentes.
- **Resposta Lenta:** O tempo de resposta do modelo é superior ao aceitável para o fluxo de trabalho.
- **Execuções Bem-Sucedidas:** Refere-se ao percentual de tarefas que foram completadas corretamente.

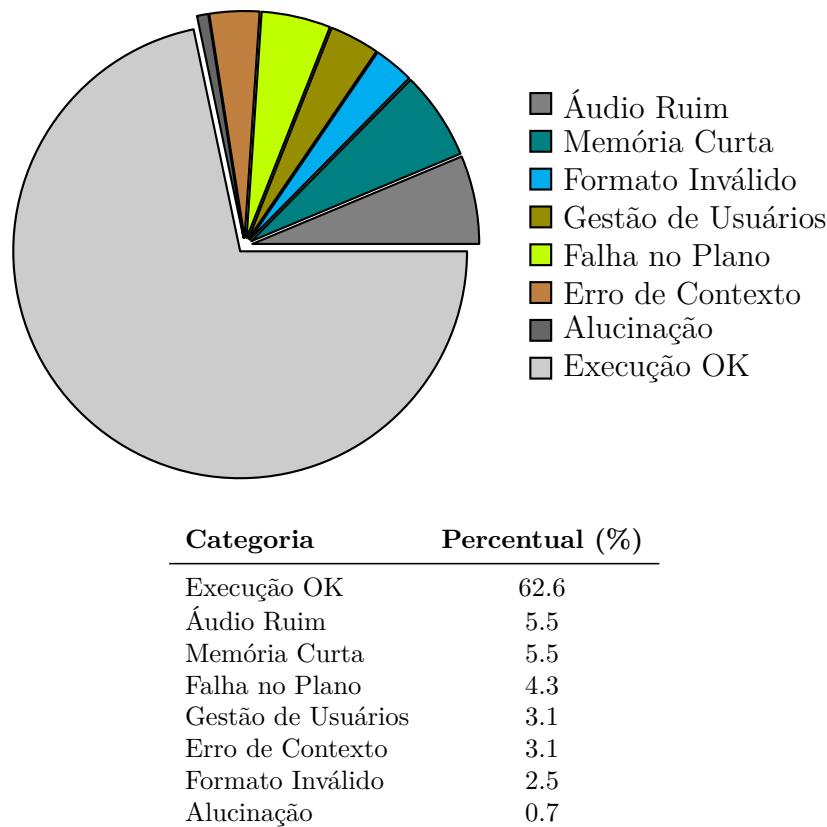


Figura 4.1 – Distribuição de Erros e Amostras Aceitáveis para GPT-3.5-turbo

Ao analisar ambas [Figura 4.1](#) e [4.2](#), observa-se que a ocorrência de erros é, em parte, estocástica, dado que as interações foram conduzidas com Agentes Humanos em um ambiente não controlado. Dentre as falhas mais representativas para ambos os modelos, destacam-se: problemas na captura de áudio, perdas de memória sequencial, falhas de replanejamento e tempos de resposta elevados.

Para tornar mais claro o funcionamento interno do sistema proposto, a [Figura 4.3](#) apresenta o fluxo de tratamento de falhas e replanejamento dinâmico adotado durante a execução das tarefas, especialmente em situações onde o plano inicial gerado

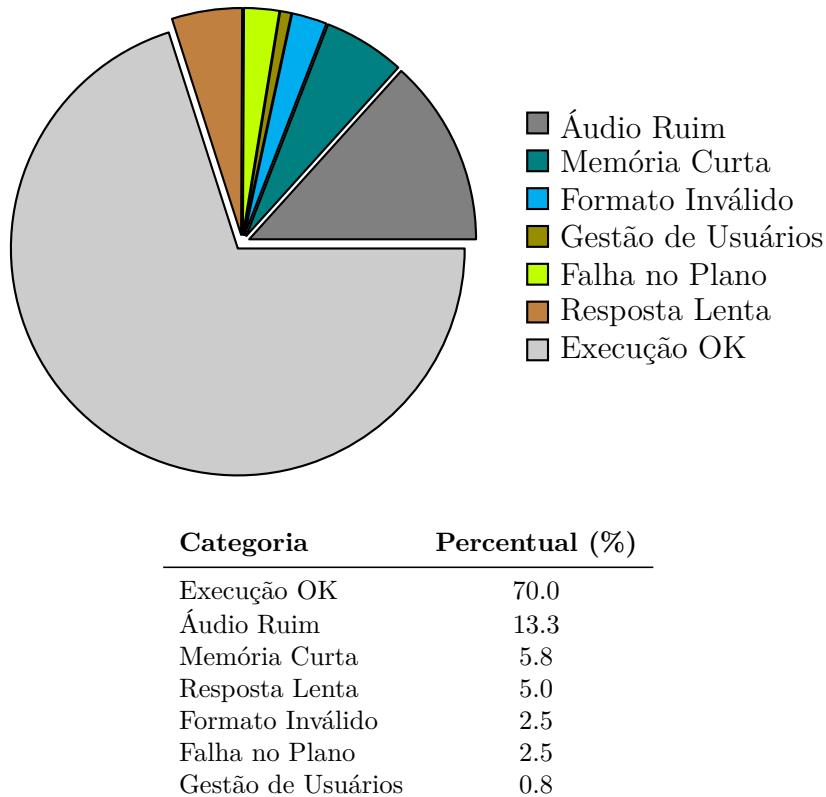


Figura 4.2 – Distribuição de Erros e Amostras Aceitáveis para GPT-4o

pelo [LLM](#) contém erros estruturais, semânticos ou de execução. Esse processo é particularmente importante quando se utiliza a técnica ReAct, que combina raciocínio e ação iterativos.

O fluxo apresentado na [Figura 4.3](#) reflete o funcionamento interno do pipeline cognitivo descrito no [Capítulo 3](#). Uma vez recebida a solicitação do usuário (por voz ou texto), o modelo [LLM](#) (GPT-3.5, GPT-4o ou LLaMA) gera um plano estruturado, geralmente em formato JSON. Esse plano é então verificado por um validador interno quanto à sintaxe, coerência semântica e compatibilidade com os agentes disponíveis.

Caso o plano esteja correto, a execução é iniciada. Em caso de erro, um novo ciclo de replanejamento é ativado automaticamente, com feedback ao [LLM](#). O sistema permite até três tentativas de correção (limitadas por desempenho). Se mesmo após esse processo o plano continuar inválido, o usuário é notificado e convidado a reformular a solicitação.

Esse processo foi essencial para o sucesso nas tarefas de longa duração (T6), onde os planos complexos exigiam maior número de correções e ajustes adaptativos em tempo real.

- **Problemas na Captura de Áudio:** De acordo com ambas as figuras de distribuição, tanto nos experimentos com "GPT-3.5" quanto com "GPT-4o", as dificuldades

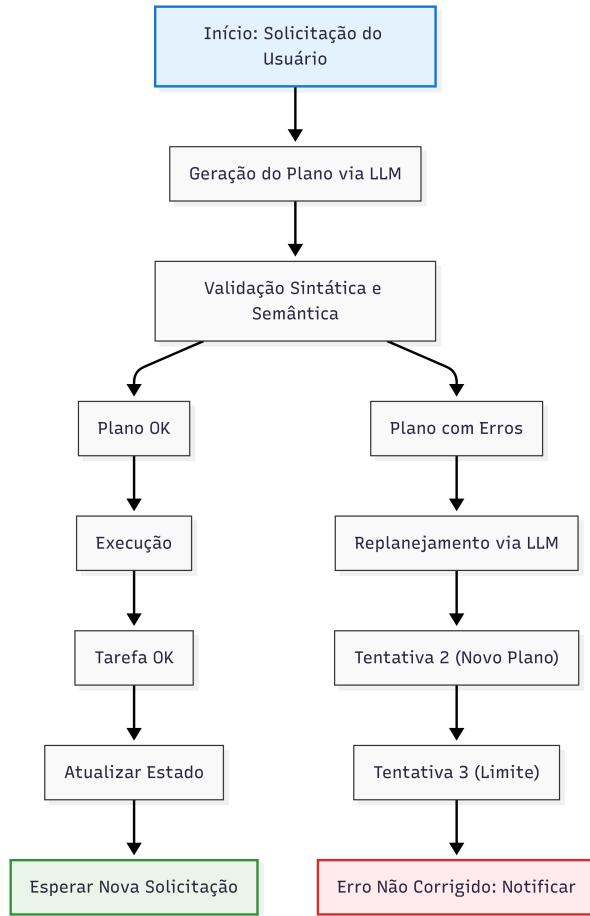


Figura 4.3 – Fluxo de tratamento de falhas com replanejamento baseado em *ReAct*

foram atribuídas à qualidade acústica do ambiente. Ruídos e sobreposição de falas afetaram a captação do áudio pelo framework de interação com o SE, principalmente em diálogos consecutivos.

- **Perda de Memória Sequencial:** Este erro ocorreu em cenários de conversas longas com múltiplas atualizações de contexto. O "GPT-3.5" apresentou mais falhas, provavelmente devido à sua *context window* de 16k tokens, enquanto o "GPT-4o", com uma janela de 128k tokens, mitigou parcialmente o problema. Este padrão segue o efeito identificado por (LIU *et al.*, 2024), conhecido como "*Lost in the middle*", em que os LLMs tendem a priorizar informações do início e do fim de contextos extensos.
- **Falhas no Replanejamento:** As falhas ocorrem quando o raciocínio inicial equivocado compromete o replanejamento subsequente. A técnica ReAct ainda demonstra certa vulnerabilidade ao propagar erros ao longo da cadeia de decisões (CoT), conforme discutido em (YAO *et al.*, 2023).

4.3 Taxa de Sucesso

Para definir a taxa de sucesso, considerou-se a quantidade de passos realizados dentro de um intervalo aceitável em relação ao número de passos corretos previstos. Dessa forma, um experimento é considerado bem-sucedido se o número de passos estiver dentro de uma margem de tolerância em relação à quantidade esperada para cada tarefa.

Como cada tipo de solicitação apresenta uma quantidade diferente de experimentos — sendo distribuídas ao todo 340 solicitações —, foram obtidas taxas de sucesso de 62,6% para o "GPT-3.5-turbo" e de 70% para o "GPT-4o" em tarefas completadas corretamente. Para generalizar o cálculo da taxa de sucesso e avaliar a variabilidade entre as categorias de tarefas, propôs-se a seguinte fórmula:

$$\text{Taxa de Sucesso} = \frac{\sum_{i=1}^N \text{Tarefas Bem-Sucedidas}_i}{\sum_{i=1}^N \text{Total de Tarefas}_i} \quad (4.1)$$

Onde:

- i : representa cada tipo de solicitação;
- N : é o número total de tipos de solicitações avaliadas;
- Tarefas Bem-Sucedidas: número de tarefas finalizadas com sucesso para a solicitação i ;
- Total de Tarefas: número total de tarefas executadas para a solicitação i .

Considera-se uma tarefa bem-sucedida quando o número de passos se encontra dentro de um intervalo baseado no desvio padrão (σ), conforme definido na equação 4.2, que considera um intervalo aceitável de passos:

$$\text{Taxa de Sucesso} = \frac{\sum_{i=1}^N \sum_{j=1}^{M_i} (|\text{Passos}_{ij} - \text{Passos Corretos}_i| \leq k \cdot \sigma_i)}{\sum_{i=1}^N \text{Total de Tarefas}_i} \quad (4.2)$$

- Passos_{ij} : quantidade de passos realizados na tarefa j do tipo de solicitação i ;
- Passos Corretos_i : número de passos idealmente esperado para a solicitação i ;
- σ_i : desvio padrão da quantidade de passos para a solicitação i ;
- k : fator de tolerância ($k=1$ para 68% de confiança, $k=2$ para 95% de confiança).

O intervalo aceitável é definido como:

$$PC \pm k \times \sigma \quad (4.3)$$

Considerou-se um valor de k variando entre 1 e 3, de acordo com a necessidade de flexibilidade para diferentes categorias de tarefas, tais como "Controle de Dispositivos IoT", "Interação entre Agentes Físicos" e "Tarefas Estendidas com Múltiplos Passos". Essa abordagem garante que o sistema avalie corretamente se o número de passos realizados está dentro de uma faixa aceitável de variação.

Na [Figura 4.4](#), é apresentada a quantidade média de passos para cada uma das seis categorias de solicitações, evidenciando as diferenças entre o desempenho do GPT-3.5 e do GPT-4o, permitindo uma comparação direta entre os modelos para cada tipo de tarefa.

Para manter a consistência nas análises, as próximas figuras também seguem a mesma categorização de tarefas, conforme os seis tipos de solicitações descritos na [Tabela 3.1](#).

Tabela 4.1 – Resumo das métricas por categoria de tarefa

Categoria	Passos Esperados	GPT-3.5-turbo		GPT-4o	
		Média	Desvio	Média	Desvio
Não Atribuído	1	1,55	0,5	1,9	0,3
Diálogo	2	1,83	0,37	1,67	0,47
Assistência	2	1,09	0,49	1,09	0,49
IoT	≤ 4	2,41	0,49	3,52	0,5
Interação	≥ 4	3,58	0,49	3,54	0,5
Sequência-LH	> 4	5,6	0,49	6,2	0,4

A [Figura 4.4](#) apresenta a comparação entre o GPT-3.5-turbo e o GPT-4o em relação ao número médio de passos necessários para a execução de tarefas, organizadas em diferentes categorias. Esses resultados estão em consonância com os dados da [Tabela 4.1](#), evidenciando que ambos os modelos se mostram eficientes e consistentes em tarefas de baixa e média complexidade, com desvios padrão reduzidos que indicam estabilidade na execução dos passos.

Tabela 4.2 – Categorias e passos ideais para diferentes tipos de tarefas

Categoria	Não Atr.	Diálogo	Assistência	IoT	Interação	Seq-LH
Passos Ideais	1	2	2	$2 < x \leq 4$	$2 < x \leq 4$	$4 < x$

Já na [Figura 4.5](#), associada à [Tabela 4.3](#), observa-se uma diferença marcante no desempenho para tarefas de *Long Tasks*. Enquanto ambos os modelos atingem taxas

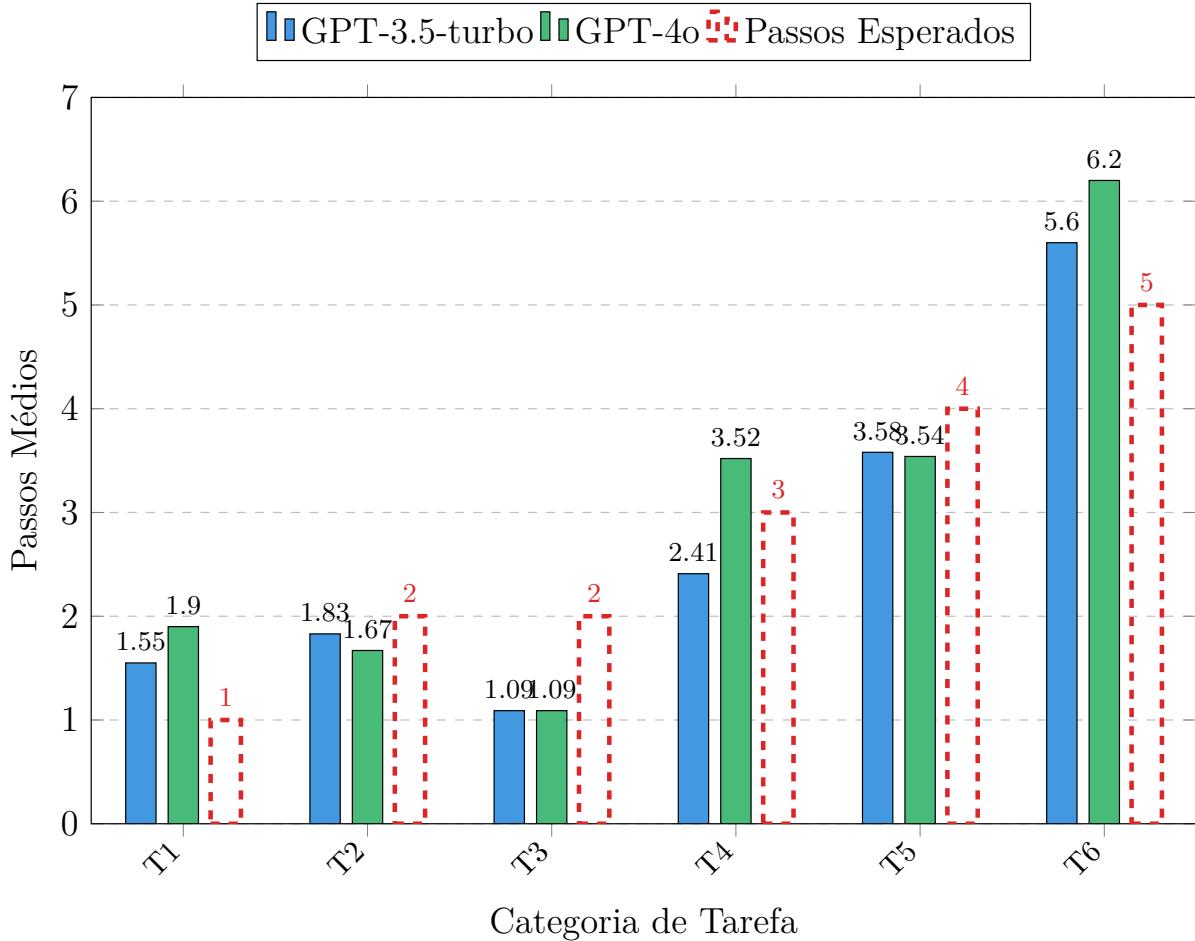


Figura 4.4 – Comparação entre os modelos GPT-3.5-turbo e GPT-4o em relação ao número médio de passos para completar tarefas em diferentes categorias. As linhas vermelhas tracejadas indicam os valores esperados para cada categoria.

- T1: Tarefas não atribuídas (1 passo esperado),
- T2: Comunicação e diálogo (2 passos esperados),
- T3: Assistência e resolução de problemas (2 passos esperados),
- T4: Controle de dispositivos IoT (3 passos esperados),
- T5: Interação entre agentes físicos (4 passos esperados),
- T6: Tarefas estendidas com múltiplas etapas (5 passos esperados).

satisfatórias para tarefas curtas *Short Tasks* e médias *Medium Tasks*, destaca-se que o pipeline integrado ao GPT-4o, utilizando a técnica [ReAct](#), apresenta um desempenho superior em tarefas com mais de 5 passos.

Em particular, a combinação de LLaMa + ReAct mostra bons resultados para tarefas simples, enquanto a arquitetura baseada em GPT-4o + ReAct supera significativamente em tarefas mais complexas e prolongadas. Essa vantagem está relacionada à capacidade do GPT-4o de lidar melhor com ciclos de realimentação de erros e ao refinamento dinâmico das subtarefas ao longo da execução.

Com base nas [Figura 4.5](#) e [Figura 4.4](#), que ilustram o desempenho dos modelos

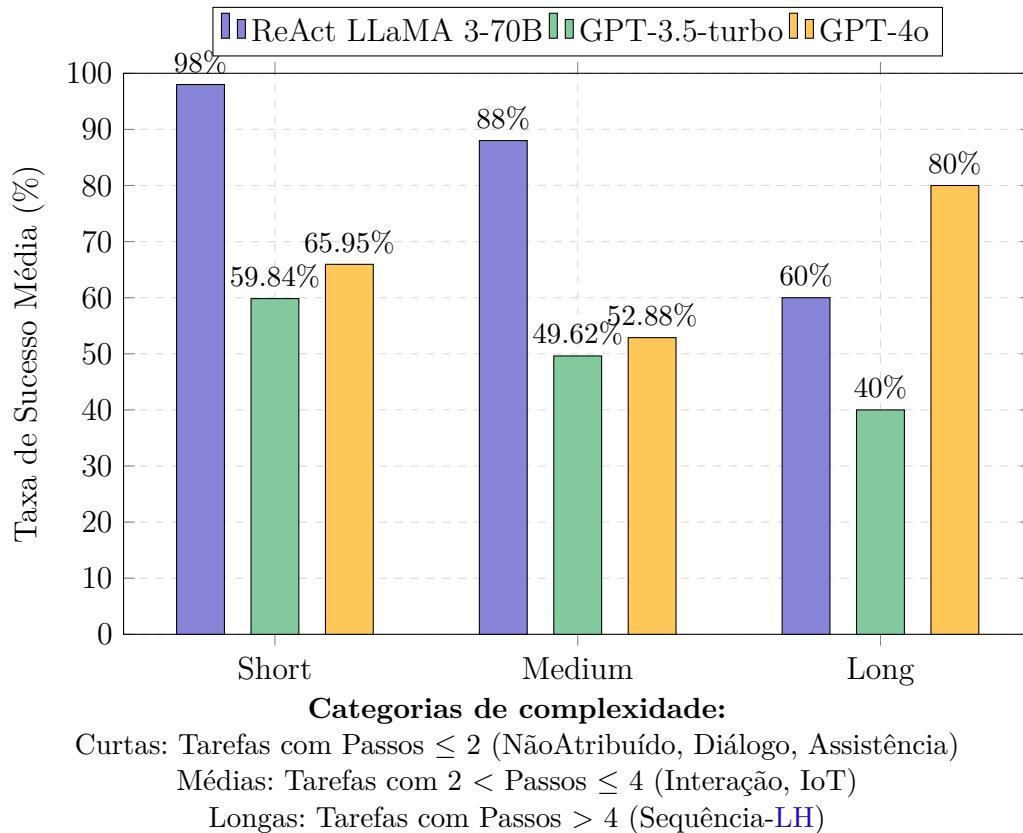


Figura 4.5 – Comparaçāo das taxas médias de sucesso por complexidade de tarefa para os três modelos avaliados. As barras de erro representam a variabilidade entre diferentes tipos de tarefas dentro de cada categoria.

Tabela 4.3 – Resumo das taxas de sucesso por categoria de tarefa para os três modelos avaliados.

Categoría de Tarea	ReAct LLaMA 3-70B	GPT-3.5-turbo	GPT-4o
NãoAtribuído	–	55%	90%
Diálogo	98%	83.33%	66.67%
Assistência	–	41.18%	41.18%
IoT	88%	41.18%	51.58%
Interação	88%	58.06%	54.17%
Sequência-LH	60%	40%	80%

em termos de número de passos médios e taxa de sucesso, a Tabela 4.4 a seguir resume as principais diferenças entre os modelos avaliados, destacando seus pontos fortes, limitações e adequação a diferentes tipos de tarefas.

Por fim, graças à melhor abstração contextual e ao raciocínio mais robusto oferecido pelo GPT-4o, o sistema alcançou um aumento de até 20% na taxa de sucesso para tarefas longas em comparação com soluções anteriores. Esse ganho demonstra a solidez do framework ao lidar com cenários que exigem uma coordenação complexa entre múltiplos agentes e a execução de sequências extensas de ações.

Tabela 4.4 – Comparaçāo de desempenho entre modelos de linguagem

Modelo	Taxa (%)	Pontos fortes	Limitações principais
GPT-3.5-turbo	62,6	Tempo de resposta rápido, baixo custo, bom em tarefas T1–T4	Perda de memória sequencial, dificuldade em replanejamento
GPT-4o	70,0	Alta capacidade de abstração, melhor raciocínio iterativo em T5–T6	Maior tempo de resposta, consumo computacional elevado
LLaMA 3B + ReAct	60,0	Implementação leve, bom desempenho em tarefas simples e definidas	Baixa generalização, dificuldades em tarefas sequenciais

4.4 Análise e Discussões

Os resultados obtidos demonstram que a arquitetura baseada em LLMs apresenta um desempenho consistente na etapa de planejamento e na execução de tarefas distribuídas em ambientes inteligentes. Em especial, a integração do *pipeline* multiagente com o modelo GPT-4o, utilizando a técnica ReAct, evidenciou uma vantagem significativa em cenários que exigem maior nível de abstração, especialmente em tarefas prolongadas.

Conforme ilustrado na Figura 4.4, tanto o GPT-3.5-turbo quanto o GPT-4o apresentaram resultados estáveis na execução de tarefas de baixa e média complexidade, com uma quantidade média de passos dentro das faixas esperadas para cada categoria e com desvios padrão controlados. No entanto, nas task *Long-Horizon* — caracterizadas pela necessidade de coordenar múltiplos agentes e gerenciar cadeias de ações com mais de cinco etapas — que o sistema impulsionado pelo GPT-4o se destaca.

A Figura 4.5 e a Tabela 4.3 evidenciam que a taxa de sucesso nas tarefas *Long-Horizon* melhora significativamente com o uso do GPT-4o, superando em até 20% as soluções baseadas em modelos como o LlaMa. Essa superioridade se deve, sobretudo, à maior capacidade do GPT-4o de gerenciar de forma eficiente o *feedback* proveniente dos agentes físicos e cognitivos durante a execução das atividades.

A utilização da técnica ReAct como mecanismo de *reasoning + acting* foi essencial para detectar, replanejar e corrigir ações durante a interação, especialmente em tarefas onde a incidência de falhas é mais elevada devido ao comprimento da sequência de ações. O GPT-4o demonstrou maior adaptabilidade frente a erros parciais, propondo soluções alternativas e conseguindo concluir as tarefas mesmo em ambientes dinâmicos ou sob restrições físicas.

Em contraste, o sistema baseado em LlaMa + ReAct apresentou desempenho satisfatório em tarefas curtas e bem definidas, porém revelou limitações na resolução de tarefas mais longas, atribuídas à menor capacidade de abstração e ao gerenciamento

limitado de contextos estendidos.

Durante a condução dos experimentos, também foi realizada uma avaliação exploratória utilizando o modelo Gemini 1.5 (Google DeepMind), com o intuito de verificar sua aplicabilidade ao pipeline proposto. No entanto, os resultados obtidos com esse modelo apresentaram inconsistências na geração de planos e instabilidade na condução de múltiplas iterações, o que impossibilitou sua inclusão nos resultados comparativos quantitativos junto aos demais modelos (GPT-3.5, GPT-4o e LLaMA).

Esta limitação não compromete a validade do trabalho, mas evidencia a necessidade de investigações futuras sobre a adaptação de **LLMs** não otimizados para ambientes iterativos e sensíveis ao contexto. Optou-se por não destacar os dados incompletos de forma gráfica, mas sim registrar esta tentativa como parte do escopo experimental exploratório. Assim, o foco da análise permanece nos modelos que apresentaram estabilidade e repetibilidade suficientes para garantir comparabilidade entre as tarefas propostas.

5 Conclusões

Nesta pesquisa, abordou-se o problema crítico da integração de LLMs em sistemas multiagentes para a IoRT, com foco específico na otimização da interação conversacional e na execução eficiente de tarefas complexas. A proposta concretizou-se por meio de um pipeline inovador, composto por módulos dedicados ao armazenamento de contexto, raciocínio distribuído, detecção de falhas e avaliação automatizada.

Os resultados obtidos demonstraram melhorias significativas na eficiência, precisão e adaptabilidade do sistema multiagente proposto, especialmente quando comparado com abordagens baseadas na técnica ReAct tradicional. Nesse método base, o modelo LLM executa raciocínio e ações sem acesso a uma memória contextual estruturada nem mecanismos dedicados de verificação e replanejamento.

A introdução de elementos como o *snapshot contextual*, a colaboração coordenada entre agentes físicos e cognitivos, e a verificação de consistência das ações resultaram em um aumento de aproximadamente 20% na taxa de sucesso das tarefas, em particular nas de *Long-Horizon*. Modelos avançados como Gemini-1.5, GPT-3.5 Turbo e GPT-4o destacaram-se pela maior eficácia na interpretação das intenções do usuário e na geração de planos coerentes e adaptativos em ambientes dinâmicos.

Ao avaliar o cumprimento dos objetivos estabelecidos inicialmente, constatou-se que a pesquisa alcançou de maneira efetiva todos os objetivos propostos. Foi criado um ambiente real/virtual controlado para simulações com robôs móveis, manipuladores e dispositivos IoT, permitindo validar o comportamento do sistema em diferentes cenários. Um sistema de armazenamento contextual foi desenvolvido para manter informações sobre o ambiente, agentes e objetos, servindo de base para o raciocínio situacional. O módulo de planejamento baseado em agentes cognitivos utilizou LLMs para decompor tarefas complexas em subtarefas interpretáveis e distribuíveis. Estratégias como o sistema MRED foram aplicadas para diagnosticar e resolver falhas durante o planejamento e a execução das tarefas. Por fim, um sistema de avaliação foi incorporado, considerando métricas como taxa de sucesso, *goal condition recall* e análise qualitativa com suporte da técnica ReAct.

Contudo, é importante reconhecer que esta pesquisa apresenta limitações específicas, especialmente no que diz respeito à complexidade dos cenários simulados em comparação com ambientes reais, possíveis erros de reconhecimento na captura de áudio e limitações inerentes aos modelos LLM relacionadas à retenção sequencial de informações.

A contribuição original deste trabalho está na demonstração de que é possível aplicar LLMs, como o GPT-4o, de forma confiável e robusta em sistemas multiagente

IoRT, com capacidade de replanejamento adaptativo, interpretação contextual e integração real com eventos sensoriais. A abordagem proposta extrapola o uso tradicional dos LLMs como meros geradores de texto, atribuindo-lhes papel central em um pipeline de controle distribuído, com impacto direto na execução de tarefas assistivas e interação humano-robô.

6 Trabalhos Futuros

Primeiramente, propõe-se o desenvolvimento de sistemas proativos de assistência personalizada, nos quais os LLMs sejam capazes de aprender continuamente com o histórico de interações dos usuários. Esses sistemas poderiam inferir padrões de comportamento, rotinas e preferências a partir de dados contextuais acumulados, permitindo antecipar comandos e fornecer sugestões adaptadas antes mesmo de qualquer solicitação explícita. Para isso, torna-se essencial a exploração de ontologias de ação e representações semânticas baseadas em tarefas recorrentes, que permitam ao sistema raciocinar sobre sequências típicas de ações. A compreensão da proatividade em diálogos humanos, como foi mostrado por (BALARAMAN *et al.*, 2020), fornece subsídios fundamentais para o desenvolvimento de LLMs mais colaborativos, capazes de detectar falhas, iniciar sugestões espontaneamente e atuar de forma antecipada em contextos interativos.

Outra vertente promissora está na fusão multimodal avançada, que visa integrar dados visuais, auditivos e outros sinais sensoriais para aprofundar a compreensão da intenção e do estado emocional do usuário. Essa abordagem permitiria que os LLMs operassem como sistemas cognitivos multimodais, capazes de interpretar comandos orientados à explicação, contexto situacional e intenção subjacente. A capacidade de analisar simultaneamente expressões faciais, entonações vocais e linguagem textual viabiliza uma interação mais natural e empática, especialmente em ambientes com múltiplos interlocutores. Estudos como o de (MOHAMED *et al.*, 2024) mostram unidades de ação facial e informações textuais para o reconhecimento de estados afetivos.

Adicionalmente, propõe-se a integração de planejadores baseados em RAG e arquiteturas de grafos semânticos, voltadas ao gerenciamento contextual da memória, à inferência em tempo real e à representação de estados e relações no ambiente inteligente. Tais arquiteturas permitem que os agentes cognitivos consultem dinamicamente uma base de conhecimento externa, raciocinem sobre situações passadas e atualizem representações do ambiente com maior fidelidade. Essa evolução pode resultar em pipelines mais escaláveis, eficientes e adaptáveis, capazes de gerar planos otimizados com base em dados atualizados e relações contextuais.

Além disso, torna-se fundamental explorar interfaces com Realidade Aumentada para suporte à interação homem-robô baseada em contexto visual. Interfaces de realidade aumentada podem projetar intenções, objetivos e instruções de agentes físicos no campo de visão do usuário, aumentando a transparência e a confiança na execução das tarefas. Um exemplo relevante é apresentado por (XIAO *et al.*, 2024), no qual o robô

assistente doméstico Robi Butler emprega **LLMs** e **Visual Language Models (VLMs)** para interpretar comandos multimodais e realizar ações em ambientes reais com alta eficácia. A Realidade Aumentada, nesse cenário, atua como um facilitador da comunicação entre humanos e robôs, promovendo uma experiência de colaboração mais fluida e intuitiva.

Baseados en la explicacion de Tasa de successo medio, conociendo las mayores capacidades de comprension tanto para tareas Long, tienen una mayor precision para dar esa descomposicion entonces se podría trocar en

Referências

- 3DX, P. *Pioneer 3DX (P3DX) RevA*. 2025. Acesso: 01-01-2025. Disponível em: <<https://www.generationrobots.com/media/Pioneer3DX-P3DX-RevA.pdf>>. Citado na página 41.
- AMATRIAIN, X. Prompt design and engineering: Introduction and advanced methods. *arXiv preprint arXiv:2401.14423*, 2024. Citado na página 26.
- BAIRD, A.; TZIRAKIS, P.; BROOKS, J. A.; GREGORY, C. B.; SCHULLER, B.; BATLINER, A.; KELTNER, D.; COWEN, A. The acii 2022 affective vocal bursts workshop & competition. In: IEEE. *2022 10th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*. [S.l.], 2022. p. 1–5. Citado na página 23.
- BALARAMAN, V.; MAGNINI, B.; KESSLER, F. B.; POVO, T. Investigating proactivity in task-oriented dialogues. *Computational Linguistics CLiC-it 2020*, p. 23, 2020. Citado na página 69.
- BROXVALL, M.; GRITTI, M.; SAFFIOTTI, A.; SEO, B.-S.; CHO, Y.-J. Peis ecology: Integrating robots into smart environments. In: IEEE. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. [S.l.], 2006. p. 212–218. Citado na página 17.
- CORADESCHI, S.; SAFFIOTTI, A. Symbiotic robotic systems: Humans, robots, and smart environments. *IEEE Intelligent Systems*, IEEE, v. 21, n. 3, p. 82–84, 2006. Citado na página 17.
- DRIESS, D.; XIA, F.; SAJJADI, M. S.; LYNCH, C.; CHOWDHERY, A.; WAHID, A.; TOMPSON, J.; VUONG, Q.; YU, T.; HUANG, W. *et al.* Palm-e: An embodied multimodal language model. 2023. Citado 2 vezes nas páginas 24 e 29.
- FEIL-SEIFER, D.; MATARIC, M. J. Defining socially assistive robotics. In: IEEE. *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005*. [S.l.], 2005. p. 465–468. Citado na página 23.
- GIL, E. B.; RODRIGUES, G. N.; PELLICCIONE, P.; CALINESCU, R. Mission specification and decomposition for multi-robot systems. *Robotics and Autonomous Systems*, Elsevier, v. 163, p. 104386, 2023. Citado na página 17.
- GOOGLE. *Especificaciones de Google Nest Mini*. 2025. Acesso: 01-01-2025. Disponível em: <<https://support.google.com/googlenest/answer/7072284?hl=es-419>>. Citado na página 42.
- HONERKAMP, D.; BÜCHNER, M.; DESPINOT, F.; WELSCHEHOLD, T.; VALADA, A. Language-grounded dynamic scene graphs for interactive object search with mobile manipulation. *IEEE Robotics and Automation Letters*, IEEE, 2024. Citado 2 vezes nas páginas 36 e 37.

- HU, M.; ZHAO, P.; XU, C.; SUN, Q.; LOU, J.; LIN, Q.; LUO, P.; RAJMOHAN, S. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation. *arXiv preprint arXiv:2408.00764*, 2024. Citado 3 vezes nas páginas 24, 28 e 29.
- HUANG, X.; LIU, W.; CHEN, X.; WANG, X.; WANG, H.; LIAN, D.; WANG, Y.; TANG, R.; CHEN, E. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024. Citado 2 vezes nas páginas 25 e 36.
- Hume AI. *Expression Measurement - Hume AI*. 2025. Acesso: 01-01-2025. Disponível em: <<https://www.hume.ai/expression-measurement>>. Citado 2 vezes nas páginas 23 e 52.
- KANNAN, S. S.; VENKATESH, V. L.; MIN, B.-C. Smart-llm: Smart multi-agent robot task planning using large language models. In: IEEE. *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2024. p. 12140–12147. Citado 6 vezes nas páginas 31, 32, 33, 34, 36 e 55.
- KELTNER, D.; SAUTER, D.; TRACY, J.; COWEN, A. Emotional expression: Advances in basic emotion theory. *Journal of nonverbal behavior*, Springer, v. 43, p. 133–160, 2019. Citado 2 vezes nas páginas 23 e 24.
- LIU, N. F.; LIN, K.; HEWITT, J.; PARANJAPE, A.; BEVILACQUA, M.; PETRONI, F.; LIANG, P. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, MIT Press One Broadway, 12th Floor, Cambridge, Massachusetts 02142, USA . . . , v. 12, p. 157–173, 2024. Citado na página 60.
- LUPERTO, M.; MONROY, J.; RENOUX, J.; LUNARDINI, F.; BASILICO, N.; BULGHERONI, M.; CANGELOSI, A.; CESARI, M.; CID, M.; IANES, A. *et al.* Integrating social assistive robots, iot, virtual communities and smart objects to assist at-home independently living elders: the movecare project. *International Journal of Social Robotics*, Springer, v. 15, n. 3, p. 517–545, 2023. Citado 3 vezes nas páginas 22, 23 e 24.
- MCCOLL, D.; HONG, A.; HATAKEYAMA, N.; NEJAT, G.; BENHABIB, B. A survey of autonomous human affect detection methods for social robots engaged in natural hri. *Journal of Intelligent & Robotic Systems*, Springer, v. 82, p. 101–133, 2016. Citado 2 vezes nas páginas 22 e 24.
- MOHAMED, Y.; LEMAIGNAN, S.; GUNEYSU, A.; JENSFELT, P.; SMITH, C. Fusion in context: A multimodal approach to affective state recognition. *arXiv preprint arXiv:2409.11906*, 2024. Citado na página 69.
- MYLAB. *Pepper Datasheet 1.8 V2*. 2019. Acesso: 01-01-2025. Disponível em: <https://mylab.ro/wp-content/uploads/2023/10/Pepper-Datasheet-1.8_V2_20190228.pdf>. Citado 2 vezes nas páginas 29 e 43.
- NAOQI. *NAOqi Framework*. 2025. Acesso: 01-01-2025. Disponível em: <<https://www.cs.cmu.edu/~cga/nao/doc/reference-documentation/dev/naoqi/index.html>>. Citado na página 54.
- OGENYI, U. E.; LIU, J.; YANG, C.; JU, Z.; LIU, H. Physical human–robot collaboration: Robotic systems, learning methods, collaborative strategies, sensors, and actuators.

- IEEE transactions on cybernetics*, IEEE, v. 51, n. 4, p. 1888–1901, 2019. Citado na página 24.
- PARADA, C. *Gemini Robotics: Bringing AI into the Physical World*. [S.l.], 2025. Disponível em: <https://storage.googleapis.com/deepmind-media/gemini-robotics/gemini_robotics_report.pdf>. Citado 2 vezes nas páginas 28 e 29.
- PENG, A.; SUCHOLUTSKY, I.; LI, B. Z.; SUMERS, T. R.; GRIFFITHS, T. L.; ANDREAS, J.; SHAH, J. A. Learning with language-guided state abstractions. *arXiv preprint arXiv:2402.18759*, 2024. Citado na página 28.
- POT, E.; MONCEAUX, J.; GELIN, R.; MAISONNIER, B. Chorographe: a graphical tool for humanoid robot programming. In: IEEE. *Ro-man 2009-the 18th ieee international symposium on robot and human interactive communication*. [S.l.], 2009. p. 46–51. Citado na página 52.
- REIS, J.; MELÃO, N.; SALVADORINHO, J.; SOARES, B.; ROSETE, A. Service robots in the hospitality industry: The case of henn-na hotel, japan. *Technology in Society*, Elsevier, v. 63, p. 101423, 2020. Citado na página 24.
- ROBOTICS, K. *Kinova Gen 3 Manual*. 2024. Acesso: 01-01-2025. Disponível em: <<https://ednex.me/wp-content/uploads/2024/01/Kinova-Gen-3-1.pdf>>. Citado na página 42.
- ROHMER, E.; SINGH, S. P.; FREESE, M. Vrep: A versatile and scalable robot simulation framework. In: IEEE. *2013 IEEE/RSJ international conference on intelligent robots and systems*. [S.l.], 2013. p. 1321–1326. Citado na página 38.
- SILVA, C. B. D.; RAMÍREZ, J. L. B.; MASTELARI, N.; LOTUFO, R.; PEREIRA, J.; ROHMER, E. Robotic action planning using large language models. In: IEEE. *2024 Latin American Robotics Symposium (LARS)*. [S.l.], 2024. p. 1–6. Citado na página 57.
- SILVA, C. J. T. da; RALHA, C. G. Multi-robot system architecture focusing on plan recovery for dynamic environments. In: IEEE. *2023 IEEE Symposium Series on Computational Intelligence (SSCI)*. [S.l.], 2023. p. 1668–1673. Citado 2 vezes nas páginas 34 e 35.
- SINGH, I.; TRAUM, D.; THOMASON, J. Twostep: Multi-agent task planning using classical planners and large language models. *arXiv preprint arXiv:2403.17246*, 2024. Citado 3 vezes nas páginas 30, 31 e 34.
- SUN, J.; ZHANG, Q.; HAN, G.; ZHAO, W.; YONG, Z.; HE, Y.; WANG, J.; CAO, J.; GUO, Y.; XU, R. Trinity: A modular humanoid robot ai system. *arXiv preprint arXiv:2503.08338*, 2025. Citado na página 29.
- TAPUS, A.; MATARIC, M. J.; SCASSELLATI, B. Socially assistive robotics [grand challenges of robotics]. *IEEE robotics & automation magazine*, IEEE, v. 14, n. 1, p. 35–42, 2007. Citado 2 vezes nas páginas 22 e 23.
- UNICAMP FAPESP, E. *UNICAMP, FAPESP e Ericsson inauguram Centro de Pesquisa em Engenharia Smartness 2030*. 2022. Acesso: 01-01-2025. Disponível em: <<https://www.inova.unicamp.br/2022/12/unicamp-fapesp-e-ericsson-inauguram-centro-de-pesquisa-em-engenharia-smartness-2030/>>. Citado na página 17.

- VEMPRALA, S. H.; BONATTI, R.; BUCKER, A.; KAPOOR, A. Chatgpt for robotics: Design principles and model abilities. *IEEE Access*, IEEE, 2024. Citado 6 vezes nas páginas 24, 25, 28, 29, 33 e 35.
- WAKE, N.; KANEHIRA, A.; SASABUCHI, K.; TAKAMATSU, J.; IKEUCHI, K. Chatgpt empowered long-step robot control in various environments: A case application. *IEEE Access*, IEEE, 2023. Citado 3 vezes nas páginas 18, 28 e 35.
- WAN, H.; ZHANG, Y.; WANG, J.; WU, D.; LI, M.; CHEN, X.; DENG, Y.; HUANG, Y.; SUN, Z.; ZHANG, L. *et al.* Toward universal embodied planning in scalable heterogeneous field robots collaboration and control. *Journal of Field Robotics*, Wiley Online Library, 2025. Citado 9 vezes nas páginas 11, 24, 28, 29, 30, 35, 36, 37 e 55.
- WEI, J.; WANG, X.; SCHUURMANS, D.; BOSMA, M.; XIA, F.; CHI, E.; LE, Q. V.; ZHOU, D. *et al.* Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, v. 35, 2022. Citado 3 vezes nas páginas 25, 33 e 35.
- WU, J.; ANTONOVA, R.; KAN, A.; LEPERT, M.; ZENG, A.; SONG, S.; BOHG, J.; RUSINKIEWICZ, S.; FUNKHOUSER, T. Tidybot: Personalized robot assistance with large language models. *Autonomous Robots*, Springer, v. 47, n. 8, p. 1087–1102, 2023. Citado na página 33.
- WU, Q.; BANSAL, G.; ZHANG, J.; WU, Y.; LI, B.; ZHU, E.; JIANG, L.; ZHANG, X.; ZHANG, S.; LIU, J. *et al.* Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023. Citado na página 26.
- WU, Y.; ZHANG, J.; HU, N.; TANG, L.; QI, G.; SHAO, J.; REN, J.; SONG, W. Mldt: Multi-level decomposition for complex long-horizon robotic task planning with open-source large language model. In: SPRINGER. *International Conference on Database Systems for Advanced Applications*. [S.l.], 2024. p. 251–267. Citado 3 vezes nas páginas 27, 30 e 33.
- XIAO, A.; JANAKA, N.; HU, T.; GUPTA, A.; LI, K.; YU, C.; HSU, D. Robi butler: Remote multimodal interactions with household robot assistant. *arXiv preprint arXiv:2409.20548*, 2024. Citado na página 69.
- YAO, S.; ZHAO, J.; YU, D.; DU, N.; SHAFRAN, I.; NARASIMHAN, K.; CAO, Y. React: Synergizing reasoning and acting in language models. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2023. Citado 5 vezes nas páginas 25, 26, 33, 34 e 60.
- ZHANG, C.; DEIK, D. G. X.; LI, D.; ZHANG, H.; LIU, Y. Meta-task planning for language agents. *arXiv preprint arXiv:2405.16510*, 2024. Citado 6 vezes nas páginas 25, 30, 32, 33, 34 e 35.
- ZHANG, X.; QIN, H.; WANG, F.; DONG, Y.; LI, J. Lamma-p: Generalizable multi-agent long-horizon task allocation and planning with lm-driven pdll planner. *arXiv preprint arXiv:2409.20560*, 2024. Citado 4 vezes nas páginas 26, 27, 31 e 36.
- ZHOU, Z.; SONG, J.; YAO, K.; SHU, Z.; MA, L. Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning. In: IEEE. *2024 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2024. p. 2081–2088. Citado na página 31.