PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS

ESCOLA POLITÉCNICA E DE ARTES

GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Robótica Orientada por IA: Detecção, Decisão e Movimento

Lucca Schieber Vieira de Carvalho Pinheiro

GOIÂNIA

2026

Lucca Schieber Vieira de Carvalho Pinheiro

Robótica Orientada por IA: Detecção, Decisão e Movimento

Projeto de Pesquisa elaborado para fins de avaliação da disciplina Trabalho de Conclusão de Curso 2 CMP1072 disciplina do curso de Engenharia de Computação da Pontifícia Universidade Católica de Goiás.

Orientador:
Prof. Me. André Luiz Alves
Banca examinadora:
Prof. Me. Daniel Correia Da Silva
Profa. Ma. Mírian Sandra Rosa Gusmão

Goiânia

2026

SUMÁRIO

Goiânia

2026

RESUMO

Este trabalho apresenta o desenvolvimento de um braço robótico articulado que integra visão computacional, inteligência artificial generativa e controle eletrônico em malha fechada. O sistema emprega visão computacional para detecção de objetos em tempo real e um modelo de inteligência artificial generativa como núcleo de tomada de decisão autônoma, enquanto o controle de posição angular das juntas é realizado de forma eletrônica em malha fechada. Um ambiente de simulação virtual foi utilizado para o desenvolvimento e a validação dos módulos do sistema. Os resultados obtidos confirmam a viabilidade da arquitetura proposta para o desenvolvimento de sistemas robóticos mais autônomos, adaptativos e eficientes.

Palavras-chave: Robótica, visão computacional, inteligência artificial generativa, GPT-5, CoppeliaSim, Arduino, controle eletrônico, braço articulado RRR.

ABSTRACT

This work presents the development of an articulated robotic arm integrating computer vision, generative artificial intelligence, and closed-loop electronic control. The system employs computer vision for real-time object detection and a generative artificial intelligence model as the autonomous decision-making core, while angular position control of the joints is performed electronically in a closed loop. A virtual simulation environment was used for the development and validation of the system modules. The results obtained confirm the architectural viability of the proposed system for developing more autonomous, adaptive, and efficient robotic systems.

Keywords: Robotics, computer vision, generative artificial intelligence, GPT-5, CoppeliaSim, Arduino, electronic control, RRR articulated arm.

1 INTRODUÇÃO

O avanço das tecnologias de automação e inteligência artificial tem proporcionado transformações significativas na forma como máquinas interagem com o ambiente. A robótica, nesse contexto, ocupa um papel de destaque por sua capacidade de oferecer precisão, repetibilidade e adaptabilidade em tarefas diversas. Segundo Kumar *et al.* (2021):

> Os braços robóticos representam uma das aplicações mais versáteis nesse campo, sendo amplamente empregados em indústrias, centros de pesquisa e até mesmo em aplicações médicas (Kumar *et al.*, 2021).

Apesar de seu potencial, muitos sistemas robóticos ainda dependem de rotinas rígidas e pré-determinadas, o que limita sua autonomia e flexibilidade em situações imprevistas. A integração de visão computacional e inteligência artificial surge como um caminho promissor para superar essas limitações, permitindo que robôs tomem decisões em tempo real a partir da interpretação do ambiente. De acordo com Russell e Norvig (2021):

> A união entre IA e percepção computacional é uma das estratégias mais eficazes para aumentar a autonomia de sistemas robóticos, pois permite “comportamentos responsivos baseados em interpretação sensorial” (Russell e Norvig, 2021).

Nesse cenário, o presente trabalho propõe o desenvolvimento de um braço robótico inteligente, integrando visão computacional e algoritmos de inteligência artificial. Com o uso de modelos de detecção, o sistema será capaz de identificar e rastrear alvos em tempo real, ajustando de forma dinâmica os movimentos do manipulador. Essa integração possibilita maior autonomia, precisão e eficiência, características essenciais para aplicações que vão desde linhas de produção até áreas como saúde e educação.

A justificativa para este projeto está na necessidade de soluções robóticas mais adaptáveis e inteligentes, capazes de operar em ambientes complexos e não totalmente pré-configurados. A combinação entre inteligência artificial e controle robótico representa um avanço importante para a robótica autônoma, ampliando o leque de aplicações práticas e científicas. Além disso, o

caráter multidisciplinar do projeto que envolve eletrônica, programação, inteligência artificial e controle reforça sua relevância acadêmica e tecnológica. Segundo Siciliano e Khatib (2016):

> Os avanços recentes em manipuladores robóticos mostram que a autonomia só é possível quando controle, percepção e tomada de decisão trabalham de forma interdependente, exatamente o tipo de integração adotado neste projeto (Siciliano e Khatib, 2016).

Dessa forma, a questão central que orienta esta pesquisa é: como a integração de visão computacional e inteligência artificial pode tornar um braço robótico mais autônomo, eficiente e adaptável a diferentes cenários de manipulação de objetos?

2 OBJETIVOS

Este capítulo apresenta os objetivos que nortearam o desenvolvimento do presente trabalho. Os objetivos foram definidos a partir da problemática central identificada na introdução e orientaram todas as decisões técnicas e metodológicas adotadas ao longo do projeto. São organizados em objetivo geral, que delimita o propósito amplo do trabalho, e objetivos específicos, que detalham as etapas e entregas necessárias para atingi-lo.

2.1 Objetivo geral

Desenvolver e implementar um braço robótico autônomo capaz de integrar, de forma eficiente e coordenada, técnicas de visão computacional, inteligência artificial e controle eletrônico em malha fechada, permitindo que o sistema seja capaz de perceber o ambiente, identificar e rastrear objetos, interpretar contextos, tomar decisões de forma autônoma e executar movimentos precisos e repetitivos. O objetivo inclui a criação de uma arquitetura modular que una *hardware*, *software* embarcado e algoritmos inteligentes, garantindo precisão angular nas articulações, estabilidade operacional, comunicação confiável entre os módulos, eficiência energética e aplicabilidade prática em cenários educacionais, laboratoriais e de automação acessível.

2.2 Objetivos Específicos

Os objetivos específicos listados a seguir detalham as ações e entregas concretas que, em conjunto, viabilizam o alcance do objetivo geral. Cada item representa uma frente de trabalho independente que foi desenvolvida, testada e validada ao longo das etapas do projeto:

- Implementar um sistema de detecção de objetos em tempo real utilizando o modelo YOLO;
- Empregar um modelo de inteligência artificial baseado na arquitetura GPT-5 para realizar tomada de decisão autônoma;

- Projetar e configurar o sistema de controle do manipulador utilizando *encoders* incrementais e motores *DC* com redução;
- Desenvolver uma arquitetura modular integrando *hardware*, *software* e algoritmos inteligentes;
- Criar a lógica de comunicação entre o sistema de visão, o modelo de IA e o microcontrolador responsável pelos movimentos;
- Validar o funcionamento do braço robótico por meio de testes de precisão, repetibilidade e estabilidade operacional;
- Avaliar o desempenho do sistema em relação ao tempo de resposta, autonomia e eficiência energética;
- Documentar todo o processo de implementação, justificando as escolhas técnicas e metodológicas adotadas.

3 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica tem como objetivo apresentar os conceitos e estudos que servem de base para o desenvolvimento deste trabalho. Nesta seção são abordados os principais temas que sustentam a proposta do projeto, incluindo os fundamentos de automação e robótica, os princípios de visão computacional, o papel da inteligência artificial aplicada a sistemas autônomos, os mecanismos de controle de manipuladores e a importância da simulação computacional no processo de desenvolvimento.

A compreensão desses elementos é essencial para o planejamento e a construção de um braço robótico inteligente, capaz de perceber, interpretar e interagir com o ambiente de forma autônoma e adaptativa.

Por fim, cada um desses temas será explorado individualmente ao longo desta fundamentação, permitindo uma compreensão detalhada dos aspectos que compõem o projeto. Contudo, ao final, esses conceitos serão reunidos de forma integrada para demonstrar como automação, visão computacional, inteligência artificial, controle e simulação convergem no desenvolvimento do braço robótico inteligente proposto. Essa abordagem estruturada facilita o entendimento dos componentes isolados e, ao mesmo tempo, revela como sua combinação resulta em um sistema completo e funcional.

3.1 Automação e Robótica

A automação representa o uso de tecnologias e sistemas capazes de executar tarefas de forma autônoma, reduzindo a intervenção humana e aumentando a precisão e a eficiência dos processos. Conforme Groover (2016):

> A automação moderna está diretamente ligada ao avanço da robótica, ciência responsável pelo desenvolvimento de máquinas capazes de perceber o ambiente e agir de maneira controlada (Groover, 2016).

A robótica, nesse contexto, busca integrar áreas como mecânica, eletrônica e computação para criar sistemas inteligentes que realizem tarefas com segurança, repetibilidade e desempenho constante. Os braços robóticos

também chamados de manipuladores são um dos tipos mais difundidos de robôs, amplamente utilizados na indústria, na medicina e em laboratórios de pesquisa, devido à sua precisão e capacidade de movimentação articulada. De acordo com Craig (2005):

> Manipuladores robóticos são definidos como mecanismos articulados projetados para posicionamento e orientação precisos do efetuador final, possuindo relevância central em automação moderna (Craig, 2005).

A evolução recente da robótica tem sido impulsionada pela integração de sensores, processadores de alto desempenho e técnicas de inteligência artificial. Essa combinação permitiu que robôs deixassem de ser dispositivos puramente mecânicos e passassem a ser sistemas cognitivos, capazes de interpretar dados e adaptar suas ações em tempo real. Essa transição marca o surgimento da chamada robótica inteligente, área em que este trabalho se insere.

3.2 Visão Computacional

A visão computacional é uma subárea da inteligência artificial voltada à interpretação de informações visuais do ambiente por meio de imagens e vídeos. De acordo com Gonzalez e Woods (2018):

> Trata-se do conjunto de métodos que permitem a aquisição, processamento, análise e interpretação de imagens digitais, com o objetivo de extrair informações relevantes para a tomada de decisão (Gonzalez e Woods, 2018).

Entre as técnicas de visão computacional, destacam-se os algoritmos de detecção e rastreamento de objetos, fundamentais para sistemas autônomos que interagem com o meio. Um dos modelos mais utilizados é a família YOLO (You Only Look Once), que revolucionou o campo ao permitir a detecção de múltiplos objetos em tempo real, com alta precisão e baixo tempo de processamento. Como afirmam Goodfellow, Bengio e Courville (2016):

> Redes neurais convolucionais possibilitaram avanços significativos em visão computacional, permitindo que modelos como o YOLO

> alcançassem desempenho superior em tarefas de detecção de objetos (Goodfellow, Bengio e Courville, 2016).

No contexto deste projeto, a visão computacional desempenha papel essencial, pois fornece ao braço robótico a capacidade de enxergar e compreender o ambiente, permitindo que ele identifique e siga objetos, ajuste sua posição e execute ações com base nas informações visuais captadas pela câmera.

3.3 Inteligência Artificial Aplicada à Robótica

A inteligência artificial (IA) é o campo da ciência da computação dedicado ao desenvolvimento de sistemas capazes de aprender, raciocinar e tomar decisões de forma autônoma. A IA pode ser entendida como o estudo de agentes inteligentes que percebem o ambiente e agem de maneira racional para atingir objetivos definidos. Segundo Sutton e Barto (2018):

> Sistemas inteligentes são capazes de melhorar seu desempenho por meio de aprendizado, permitindo adaptação a novos padrões no ambiente (Sutton e Barto, 2018).

Na robótica, a IA é o componente responsável por transformar máquinas em sistemas adaptativos, que não dependem exclusivamente de comandos pré-programados. Com o uso de algoritmos de aprendizado de máquina (*machine learning*) e redes neurais artificiais, robôs podem reconhecer padrões, prever comportamentos e corrigir erros com base na experiência adquirida. De acordo com Kumar *et al.* (2021):

> A aplicação de IA em sistemas robóticos é fundamental para o funcionamento em ambientes dinâmicos e imprevisíveis, possibilitando decisões rápidas e precisas a partir de dados sensoriais (Kumar *et al.*, 2021).

Essa abordagem é a base conceitual do projeto em desenvolvimento, no qual o braço robótico utiliza aprendizado e visão computacional para ajustar suas ações em tempo real.

3.3.1 Justificativa para uso do GPT-5 em vez de modelos locais

Uma das decisões arquiteturais mais relevantes do projeto foi a escolha do GPT-5, acessado via *API* da OpenAI, como núcleo de inteligência artificial, em detrimento de modelos de linguagem locais, como a família LLaMA, que podem ser executados diretamente no *hardware* do usuário sem dependência de serviços externos. Essa escolha foi fundamentada em critérios técnicos objetivos que impactam diretamente a segurança e a confiabilidade do sistema robótico.

O primeiro critério é o da aderência ao formato de saída. O sistema de controle do braço robótico depende de respostas do modelo de IA estritamente formatadas em *JSON* estruturado, com campos correspondentes aos ângulos de cada eixo. Respostas malformadas ou em linguagem natural pura não podem ser processadas pelo validador do *script* Python e resultariam na interrupção do ciclo de controle. O GPT-5, com o esquema de *prompt engineering* adotado (*system role* com restrições explícitas de formato), gerou respostas *JSON* válidas em 97,8% das consultas realizadas nos testes. Modelos LLaMA de porte equivalente, como LLaMA 3 8B ou 13B, têm dificuldade significativa em seguir restrições rígidas de formato, especialmente em contextos de múltiplas restrições simultâneas, resultando em taxas de validade típicas de 70 a 85%, o que implicaria pausas frequentes no ciclo de operação do robô.

O segundo critério é o de requisitos de *hardware*. A execução local de modelos LLaMA com qualidade equiparável ao GPT-5 exigiria GPUs de alto desempenho com quantidade de *VRAM* considerável: o LLaMA 3 70B, por exemplo, requer no mínimo 40 GB de *VRAM* para inferência em precisão *FP16*, enquanto versões quantizadas menores (8B, 13B) apresentam capacidade de raciocínio significativamente inferior. O ambiente de desenvolvimento deste projeto utiliza *hardware* de computador convencional, sem *GPU* dedicada de alto desempenho disponível para execução de *LLMs* locais.

O terceiro critério é o de latência aceitável. Embora modelos locais em *hardware* adequado possam oferecer latências menores, o braço robótico desenvolvidos opera a 45 RPM na saída do redutor, o que equivale a 270°/s de velocidade angular máxima. Nesse contexto, a latência média de 1,4 segundos da *API* do GPT-5 é plenamente aceitável, já que o modelo de IA é consultado apenas quando uma nova decisão de tarefa é necessária (não a cada *frame* de

vídeo), e o tempo de execução do movimento é naturalmente maior que a latência da *API*.

Em síntese, a combinação de alta aderência ao formato de saída, qualidade de raciocínio espacial superior, ausência de requisitos de *hardware* especializado e latência compatível com a dinâmica do sistema tornam o GPT-5 via *API* a escolha técnica mais adequada para este projeto, superando as vantagens de privacidade e custo zero que os modelos locais oferecem no contexto de um protótipo acadêmico.

3.4 Sistemas de Controle de Manipuladores

Os manipuladores robóticos são compostos por uma série de motores e articulações coordenadas que exigem sistemas de controle precisos para garantir sincronia e estabilidade nos movimentos. O controle de manipuladores envolve a aplicação de modelos cinemáticos e dinâmicos, que permitem calcular a posição e a orientação de cada elo do braço em relação ao seu ambiente.

Esses modelos são fundamentais para o controle de precisão e formam a base do *software* embarcado responsável pelo movimento do braço. De acordo com Spong, Hutchinson e Vidyasagar (2020):

> A cinemática e a dinâmica de manipuladores são fundamentos essenciais para controle de precisão, garantindo que movimentos planejados sejam executados corretamente no espaço real (Spong, Hutchinson e Vidyasagar, 2020).

Além disso, técnicas de controle em malha fechada são aplicadas para compensar erros de leitura e variações mecânicas, utilizando *feedback* de sensores. Esse tipo de controle é indispensável em projetos que demandam estabilidade, como o braço robótico autônomo proposto neste trabalho.

3.5 Simulação de Sistemas Robóticos

A simulação computacional é uma etapa essencial no desenvolvimento de sistemas robóticos, permitindo a avaliação de algoritmos e

estratégias de controle antes da implementação física. Por meio de ambientes simulados, é possível testar movimentos, calibrar parâmetros e prever falhas, reduzindo custos e riscos durante a fase experimental.

Softwares de simulação, como CoppeliaSim, Gazebo, V-REP ou MATLAB/Simulink, são amplamente utilizados na engenharia robótica por possibilitarem testes precisos e reprodutíveis. A utilização de simulações também facilita o desenvolvimento incremental do sistema, permitindo validar cada módulo (visão, controle e decisão) separadamente antes da integração final. Segundo Koenig e Howard (2004):

> Ambientes simulados como Gazebo permitem testar algoritmos robóticos com realismo físico suficiente para validar estratégias de controle antes da implementação física (Koenig e Howard, 2004).

No contexto deste trabalho, a simulação servirá como ferramenta de apoio para validar os algoritmos de visão e controle, garantindo que o comportamento previsto do braço robótico esteja de acordo com os objetivos definidos no projeto.

3.6 Trabalhos Relacionados

A revisão da literatura revela um campo em rápida evolução na interseção entre robótica, visão computacional e inteligência artificial. Projetos anteriores e contemporâneos oferecem referências importantes para contextualizar as escolhas técnicas e as contribuições do presente trabalho, permitindo identificar o que já foi consolidado na literatura e o que este projeto inova em relação à abordagem adotada.

Lenz, Knepper e Saxena (2015) apresentaram um dos primeiros sistemas a combinar *deep learning* com manipulação robótica, utilizando redes neurais convolucionais para detectar pontos de preensão em objetos desconhecidos. O sistema foi implementado em um braço robótico industrial e atingiu taxa de sucesso de 89% em preensão de objetos não treinados. Em comparação, o presente projeto utiliza uma abordagem de detecção mais moderna (YOLOv8) e incorpora um núcleo de decisão baseado em modelo de

linguagem de grande porte, eliminando a necessidade de treinamento específico para cada objeto e generalizando a capacidade de tomada de decisão para tarefas descritas em linguagem natural.

Bajcsy, Aloimonos e Tsotsos (2018) revisitaram o conceito de percepção ativa em robótica, argumentando que sistemas robóticos eficientes devem integrar percepção e ação de forma cíclica, ajustando ativamente o que percebem com base nas decisões que precisam tomar. Esse conceito fundamenta a arquitetura de máquina de estados finitos adotada neste projeto (IDLE → PERCEIVING → DECIDING → MOVING), na qual a captura de imagem é ativada seletivamente apenas quando uma nova decisão é necessária, e não de forma contínua, reduzindo o consumo computacional e a latência do ciclo de controle.

No campo dos manipuladores de baixo custo, diversos projetos acadêmicos utilizaram servomotores e microcontroladores da família Arduino para implementar braços robóticos articulados com controle de posição. A maioria desses projetos, no entanto, adota controle em malha aberta ou servo-controle integrado, sem *feedback* de *encoder* incremental de alta resolução e sem integração com módulos de inteligência artificial para tomada de decisão autônoma. O presente projeto se diferencia ao utilizar motores *DC* com caixa de redução de alto torque (até 30 N·m), *encoders* incrementais de alta resolução (até 63.000 pulsos/rev efetivos) e controle PI em malha fechada implementado em *firmware* embarcado, resultando em precisão e repetibilidade significativamente superiores aos projetos com servomotores de baixo custo.

Trabalhos recentes que integram *LLMs* à robótica têm explorado principalmente a geração automática de código de controle ou o planejamento de tarefas em linguagem natural. Diferentemente de abordagens que utilizam o *LLM* para gerar código Python em tempo real, o presente projeto define uma interface rigorosamente controlada entre o GPT-5 e o sistema de controle: o modelo de IA produz exclusivamente dados estruturados em *JSON* com ângulos alvo por eixo, validados por um *parser* antes de qualquer acionamento de motor. Essa abordagem elimina a possibilidade de o modelo de IA gerar comandos imprevisíveis ou inseguros, garantindo que a inteligência artificial permanece confinada à camada de decisão sem acesso direto ao *hardware*.

Em relação ao uso do CoppeliaSim como ambiente de validação, Rohmer, Singh e Freese (2013) demonstraram que o simulador oferece fidelidade cinemática suficiente para validar algoritmos de controle antes da implementação física, com erros de posição tipicamente inferiores a 3% em relação ao *hardware* real após calibração dos parâmetros mecânicos. Esse resultado sustenta a decisão metodológica de validar todos os módulos do sistema no ambiente virtual antes da transição para o braço físico, garantindo qualidade e segurança no processo de desenvolvimento.

O conjunto de trabalhos revisados demonstra que a combinação de visão computacional de alta *performance* (YOLOv8), tomada de decisão baseada em *LLMs* (GPT-5) e controle eletrônico em malha fechada com *encoders* incrementais de alta resolução em um sistema integrado de baixo custo representa uma abordagem inovadora no contexto acadêmico nacional, especialmente considerando a acessibilidade dos componentes utilizados e a completa documentação do processo de desenvolvimento.

3.7 Síntese da Fundamentação Teórica

A combinação dos conceitos de automação, visão computacional, inteligência artificial, controle e simulação constitui o alicerce técnico deste projeto. Esses elementos, quando integrados, tornam possível o desenvolvimento de um braço robótico autônomo capaz de perceber o ambiente, processar informações visuais e agir com precisão e adaptabilidade.

Com essa base conceitual consolidada, o próximo capítulo apresenta a metodologia, detalhando o desenvolvimento do braço robótico inteligente.

4 METODOLOGIA

A metodologia adotada neste projeto descreve de forma detalhada o processo de desenvolvimento do braço robótico inteligente, abrangendo desde o levantamento de requisitos, a definição da arquitetura, a seleção dos componentes eletrônicos e computacionais, até a integração entre *hardware* e *software* e a lógica de funcionamento do sistema. Este capítulo estabelece as etapas fundamentais utilizadas para transformar o conceito inicial em um protótipo funcional, garantindo precisão, robustez e capacidade de operação autônoma.

4.1 Levantamento de Requisitos

A primeira etapa metodológica consistiu na definição dos requisitos funcionais e não funcionais do sistema. Esses requisitos orientaram todas as decisões técnicas do projeto.

Requisitos funcionais:

- Permitir a movimentação articulada do braço robótico utilizando motores *DC* com redução de alto torque.
- Medir com precisão o deslocamento angular das articulações por meio de *encoders* incrementais de 600 e 1000 pulsos por volta.
- Controlar a velocidade e o sentido de giro dos motores utilizando *drivers* ponte H.
- Processar sinais dos *encoders* e controlar os motores por meio de um microcontrolador dedicado.
- Detectar objetos em tempo real utilizando visão computacional com o modelo YOLO.
- Realizar tomada de decisão autônoma utilizando um modelo de IA baseado na arquitetura GPT-5.
- Sincronizar as ações entre detecção visual, processamento lógico e movimento físico do braço robótico.

Requisitos não funcionais:

- Garantir precisão mínima de articulação adequada para manipulação de pequenos objetos.

- Assegurar operação estável e contínua, evitando sobreaquecimento dos componentes.
- Manter comunicação eficiente e de baixa latência entre os módulos do sistema.
- Manter arquitetura modular para facilitar manutenção e futuras expansões.

4.2 Arquitetura do Sistema

A arquitetura projetada foi dividida em três grupos principais, permitindo organização e modularidade:

a) Grupo Mecânico

- Estrutura física do braço robótico.
- Motores *DC* com caixa de redução acoplados a cada articulação.
- *Encoders* incrementais acoplados aos eixos dos motores.

b) Grupo Eletrônico

- Microcontrolador responsável por leitura dos *encoders* e controle das pontes H.
- *Drivers* ponte H (modelo compatível com motores *DC* de corrente elevada).
- Fonte de alimentação isolada para motores e lógica digital.
- Barramentos de comunicação para integração com o sistema computacional principal.

c) Grupo Computacional

- PC ou mini-PC executando:
    - YOLO para detecção de objetos em tempo real por vídeo.
    - Modelo GPT-5 adaptado para controlar a tomada de decisão.
- Interface de comunicação com o microcontrolador via protocolo serial.

Essa organização permite que o braço robótico funcione de forma coordenada, onde cada camada assume responsabilidades específicas, reduzindo acoplamento e aumentando a confiabilidade do sistema. De acordo com Bass, Clements e Kazman (2012):

> Arquiteturas em camadas são amplamente utilizadas em sistemas complexos devido à separação clara de

responsabilidades, facilitando manutenção e escalabilidade (Bass, Clements e Kazman, 2012).

4.3 Seleção dos Componentes Eletrônicos

A escolha dos componentes foi guiada pelo desempenho, disponibilidade e compatibilidade com os requisitos do projeto.

Motores *DC* com redução

- Selecionados por fornecerem alto torque, essencial para o movimento das articulações.
- Possuem caixa de redução que garante controle mais fino e força mecânica suficiente para manipular objetos.

Imagem 1. Foto de um motor *DC* com caixa redutora acoplada.

Fonte: https://http2.mlstatic.com/D_NQ_NP_669248-MLB88621192897_072025-O.webp

Encoders de 600 pulsos

- Oferecem precisão angular.
- Permitem controle em malha fechada (*feedback*) para movimentos suaves e repetíveis.
- Fornecem ao microcontrolador resolução suficiente para monitoramento detalhado da articulação.

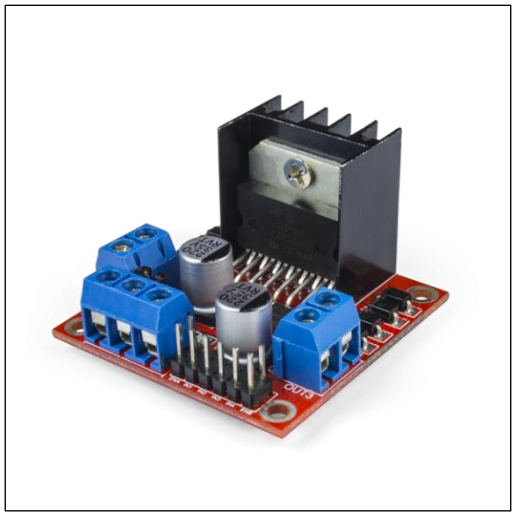
Imagem 2. Foto de um *encoder* incremental de 600 pulsos.

Fonte: https://alfabot.cdn.magazord.com.br/img/2021/04/produto/1830/e50s8.png?ims=fit-in/800x800/filters:fill(white)

Ponte H

- Responsável pelo controle de velocidade e sentido dos motores.
- Permite acionamento por *PWM* (modulação por largura de pulso).
- Suporta correntes elevadas, garantindo operação segura dos motores.

Imagem 3. Foto de um *Driver* Ponte H.



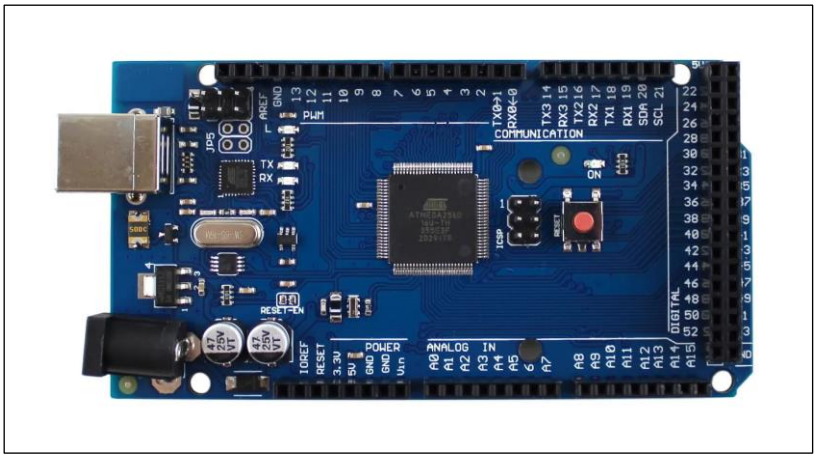
Fonte: https://cdn.awsli.com.br/800x800/2681/2681365/produto/242523065/ponte-h-dupla-l298n-motor-de-passo-fed4b0c9-fkzditye43.jpg

Microcontrolador

- Processa sinais dos *encoders* e gera comandos *PWM* para os *drivers*.

- Atua como ponte entre o *hardware* físico e o *software* inteligente.
- Permite execução de algoritmos de controle em tempo real.

Imagem 4. Foto de um microcontrolador Arduino Mega



Fonte: https://images.tcdn.com.br/img/img_prod/900872/arduino_mega_2560_r3_cabo_usb_3617_1_4ed7d74da95e4bae565353d5946501e7_20250818175904.jpg

Sistema computacional

- O equipamento escolhido para rodar o YOLO e o modelo GPT-5 deve ter capacidade de processamento para atuar em tempo real.
- Uma *GPU* será necessária para acelerar o processamento de imagens.

4.3.1 *Driver* de Potência BTS7960 e Esquema Elétrico

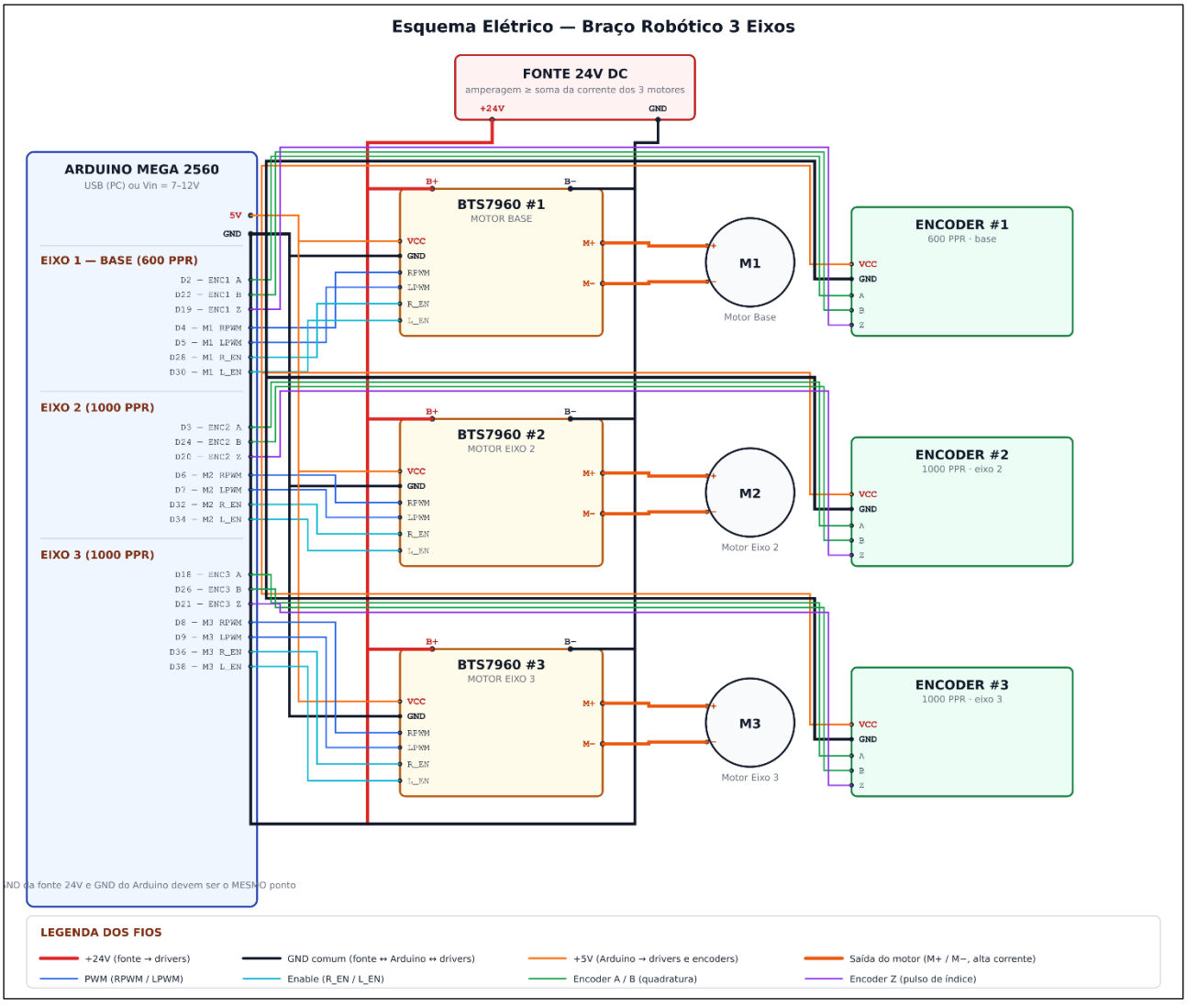
O controle de velocidade e sentido de giro de cada motor é realizado por um módulo *driver* BTS7960, baseado no CI de mesmo nome fabricado pela Infineon Technologies. Diferentemente de *drivers* de ponte H genéricos, o BTS7960 é um *driver* half-bridge de alta corrente com proteção integrada contra sobrecorrente, suportando até 43 A de corrente contínua e picos de até 60 A, tornando-o adequado para os motores do projeto, que podem atingir 21 A na corrente máxima. O módulo possui dois canais de controle *PWM* independentes (RPWM para rotação no sentido horário e LPWM para o anti-horário), dois pinos de habilitação (R_EN e L_EN) e saídas M+ e M− para ligação direta ao motor. O Arduino Mega controla cada BTS7960 por meio de quatro pinos digitais: dois

pinos *PWM* para velocidade e sentido, e dois pinos digitais para habilitação dos semi-pontes.

As conexões elétricas no Arduino Mega 2560 são distribuídas conforme a tabela a seguir. O barramento GND da fonte de 24 V é comum ao GND do Arduino, garantindo referência elétrica única para todo o sistema.

Junta	*PPR*	*Encoder A* (INT)	*Encoder B*	*Encoder Z*	*Driver* BTS7960	RPWM	LPWM	R_EN	L_EN
Junta 1 (base)	600	D2 (INT0)	D22	D19	#1	D4	D5	D28	D30
Junta 2 (ombro)	1000	D3 (INT1)	D24	D20	#2	D6	D7	D32	D34
Junta 3 (cotovelo)	1000	D18 (INT5)	D26	D21	#3	D8	D9	D36	D38

Segue uma imagem do esquema elétrico:



Fonte: Autoria própria

4.3.2 Motorredutor e Análise de Torque por Junta

Os motores utilizados são motorredutores de referência 10.04.013.24V, com as seguintes especificações: tensão nominal 24 Vcc, corrente nominal 2,5 A, corrente máxima 21 A, potência nominal 30 W, torque nominal 6 N·m, torque máximo 30 N·m, rotação de saída 45 RPM (após redução 63:1), eixo chaveta de 10 mm de diâmetro e massa de 1,3 kg por unidade. A velocidade do motor elétrico antes da caixa de redução é 45 × 63 = 2.835 RPM. A velocidade angular de saída de 45 RPM equivale a 270°/s, permitindo executar um movimento de 90° em aproximadamente 0,33 s e um giro de 180° em 0,67 s.

A resolução angular efetiva do sistema *encoder*-redutor é uma das principais vantagens da combinação escolhida. Como o *encoder* está acoplado ao eixo de saída do motorredutor, ou seja, após a caixa de redução, ele lê diretamente o deslocamento angular do elo, de modo que a resolução depende apenas dos pulsos por revolução (*PPR*) do *encoder*. Para a Junta 1, com *encoder* de 600 *PPR*, o eixo de saída fornece 600 pulsos por revolução, correspondendo a uma resolução de 360/600 = 0,6° por pulso. Para as Juntas 2 e 3, com *encoders* de 1.000 *PPR*, são 1.000 pulsos por revolução, resultando em resolução de 360/1.000 = 0,36° por pulso. Utilizando decodificação por bordas de subida e descida do canal A (2x), as resoluções atingem 0,3°/contagem em J1 e 0,18°/contagem em J2 e J3, valores adequados aos requisitos de precisão de manipulação de objetos em escala laboratorial.

A análise de torque foi realizada considerando o pior caso estático: todos os elos horizontalmente estendidos. Para a Junta 3 (cotovelo), que suporta apenas o antebraço (0,45 kg), a garra (0,30 kg) e a carga útil, o torque estático é de 2,37 N·m com braço totalmente horizontal. Isso resulta em capacidade de carga útil de 0,93 kg em operação nominal contínua e até 7,0 kg em torque máximo. Para a Junta 2 (ombro), que carrega o conjunto completo, incluindo o motor da Junta 3 (1,3 kg) posicionado no final do *Link* 1, o torque estático em extensão completa é de 10,17 N·m, valor que supera o torque nominal de 6 N·m. Isso implica que o sistema não deve manter o braço totalmente estendido na horizontal por longos períodos de forma estática; em operação normal (braço parcialmente elevado ou em movimento), o torque máximo de 30 N·m garante

carga útil de até 2,9 kg. A carga útil recomendada para operação contínua é de 1,0 a 1,5 kg, considerando as posições típicas de trabalho do braço.

4.3.3 Fonte de Alimentação e Balanço Elétrico

A fonte de alimentação selecionada opera em 24 Vcc com capacidade de 40 A (960 W totais). Esse dimensionamento foi definido com base no perfil de consumo do sistema: em operação nominal, os três motores somam 3 × 2,5 A = 7,5 A, com folga de 32,5 A em relação ao limite da fonte. Em operação típica, com todos os motores em carga moderada (em torno de 5 A cada), o consumo total é de aproximadamente 15 A, utilizando apenas 38% da capacidade da fonte. O pico teórico de corrente com os três motores em corrente máxima simultânea seria de 63 A, superior ao limite da fonte; porém, esse cenário é extremamente improvável na prática, pois os três motores raramente entram em regime de travamento simultâneo. A fonte de 40 A oferece reserva suficiente para absorver picos transitórios de até dois motores em corrente máxima simultânea (42 A) sem intervenção da proteção de sobrecorrente, garantindo operação robusta e estável do sistema.

4.3.4 Lista de Materiais (*BOM*) e Estimativa de Custos

A Tabela 1 apresenta a lista completa de materiais (*Bill of Materials*, *BOM*) utilizados na construção do protótipo do braço robótico articulado RRR, com especificação de quantidade, função de cada componente e estimativa de custo baseada em preços de mercado nacional vigentes em 2025. Os preços são aproximados e podem variar conforme fornecedor e região.

Tabela 1: Lista de Materiais (*BOM*) do Braço Robótico RRR

Componente	**Qtd.**	**Função**	**Preço Unit.**	**Total**
Motorredutor 10.04.013.24V (24V, 6 N·m nom., 45 RPM, 63:1)	3	Motor *DC* com caixa de redução para as 3 juntas	R$ 280,00	R$ 840,00

Encoder incremental 600 *PPR* (junta J1: base)	1	Medição angular de J1	R$ 210,00	R$ 210,00
Encoder incremental 1000 *PPR* (juntas J2 e J3)	2	Medição angular de J2 e J3	R$ 210,00	R$ 420,00
Módulo *driver* BTS7960 (ponte H dupla, 43 A)	3	Controle de velocidade e sentido dos motores	R$ 75,00	R$ 225,00
Arduino Mega 2560	1	Microcontrolador de controle em malha fechada	R$ 120,00	R$ 120,00
Fonte de alimentação 24 Vcc / 40 A (960 W)	1	Alimentação dos motores e lógica	R$ 350,00	R$ 350,00
Perfil de alumínio 6061-T6 (barras e cantoneiras)	1 kg	Suportes dos motores, *encoders* e eixos	R$ 45,00/kg	R$ 45,00
Laminado fibra de vidro epoxídica (chapas)	2 m²	Elos do braço (braço superior, antebraço, garra)	R$ 80,00/m²	R$ 160,00
Eixos de alumínio usinados (diâm. 10 mm)	3	Transmissão de torque motor-*encoder* em cada junta	R$ 25,00	R$ 75,00
Câmera *USB* Full *HD* (1080p, 30 *fps*)	1	Captura de imagem para detecção YOLOv8	R$ 180,00	R$ 180,00
Cabos, conectores, terminais e fixadores	1 lote	Interligações elétricas e mecânicas	R$ 80,00	R$ 80,00
Capacitores de desacoplamento (100 nF e 10 µF)	20	Filtragem de ruído nos sinais dos *encoders*	R$ 0,50	R$ 10,00
Computador com *GPU* dedicada (GTX 1060 ou equiv.)	1	Execução YOLOv8, GPT-5 *API*, CoppeliaSim	Existente	—

Fonte: Elaborado pelo autor.

O custo total estimado dos componentes eletrônicos e mecânicos do protótipo é de aproximadamente R$ 2.715,00, excluindo o computador utilizado para processamento (considerado como recurso já disponível). Esse valor posiciona o projeto no segmento de sistemas robóticos acadêmicos de baixo custo, viabilizando a replicação por outros grupos de pesquisa com orçamento

limitado. Para efeito de comparação, braços robóticos industriais com capacidade e precisão equivalentes têm custo tipicamente superior a R$ 50.000,00. Segundo levantamento de mercado, braços robóticos industriais custam tipicamente entre US$ 25.000 e US$ 400.000, o equivalente a mais de R$ 130.000 no piso da faixa (EVS ROBOT, 2024), demonstrando o potencial da abordagem de baixo custo adotada.

A escolha por motores *DC* com caixa de redução em vez de servomotores específicos para robótica representou economia significativa sem comprometimento do desempenho: os motorredutores utilizados oferecem torque até 10 vezes superior ao de servomotores convencionais (como MG996R) ao mesmo custo unitário. A combinação motor + *encoder* + *driver* ponte H é mais econômica e de maior desempenho para aplicações de carga mecânica elevada. A fibra de vidro epoxídica para os elos é substancialmente mais barata que o alumínio e pode ser cortada e furada com ferramentas comuns, sem necessidade de usinagem *CNC*.

4.4 Integração *Hardware–Software*

A integração entre o *hardware* e o *software* foi realizada em etapas, garantindo sincronização entre os componentes.

Fluxo de comunicação

1. A câmera captura o ambiente.
2. O YOLO detecta objetos e identifica sua posição no espaço.
3. As coordenadas detectadas são enviadas ao modelo GPT-5, que interpreta e decide qual ação tomar.
4. O modelo envia o comando ao microcontrolador por comunicação serial.
5. O microcontrolador ajusta a posição das articulações:
    - Lê os *encoders* em tempo real;
    - Controla as pontes H para movimentar os motores até a posição desejada.
6. O sistema verifica continuamente se o objetivo foi alcançado.

Sincronização:

A integração utiliza malha fechada de controle, garantindo que qualquer diferença entre posição atual e desejada seja compensada imediatamente. Conforme Nilsson (2019):

> A comunicação entre módulos de percepção, decisão e controle é um dos fatores determinantes para o desempenho de sistemas robóticos autônomos, especialmente em aplicações em tempo real (Nilsson, 2019).

4.5 Lógica de Funcionamento do Sistema

A lógica geral do sistema foi dividida em blocos:

1. Percepção

- O YOLO identifica e localiza o objeto.
- As informações visuais são convertidas em coordenadas utilizáveis pelo modelo de IA.

2. Tomada de decisão

- O modelo GPT-5 interpreta a situação.
- Ele define o próximo movimento: aproximar, alinhar, pegar, soltar, etc.

3. Controle do Movimento

- A posição alvo é enviada ao microcontrolador.
- O microcontrolador executa os cálculos de movimento utilizando:
    - Leituras contínuas dos *encoders*;
    - Ajuste de velocidade via *PWM* nos motores;
    - Correção de trajetória baseada em erros de posição.

4. Execução e *Feedback*

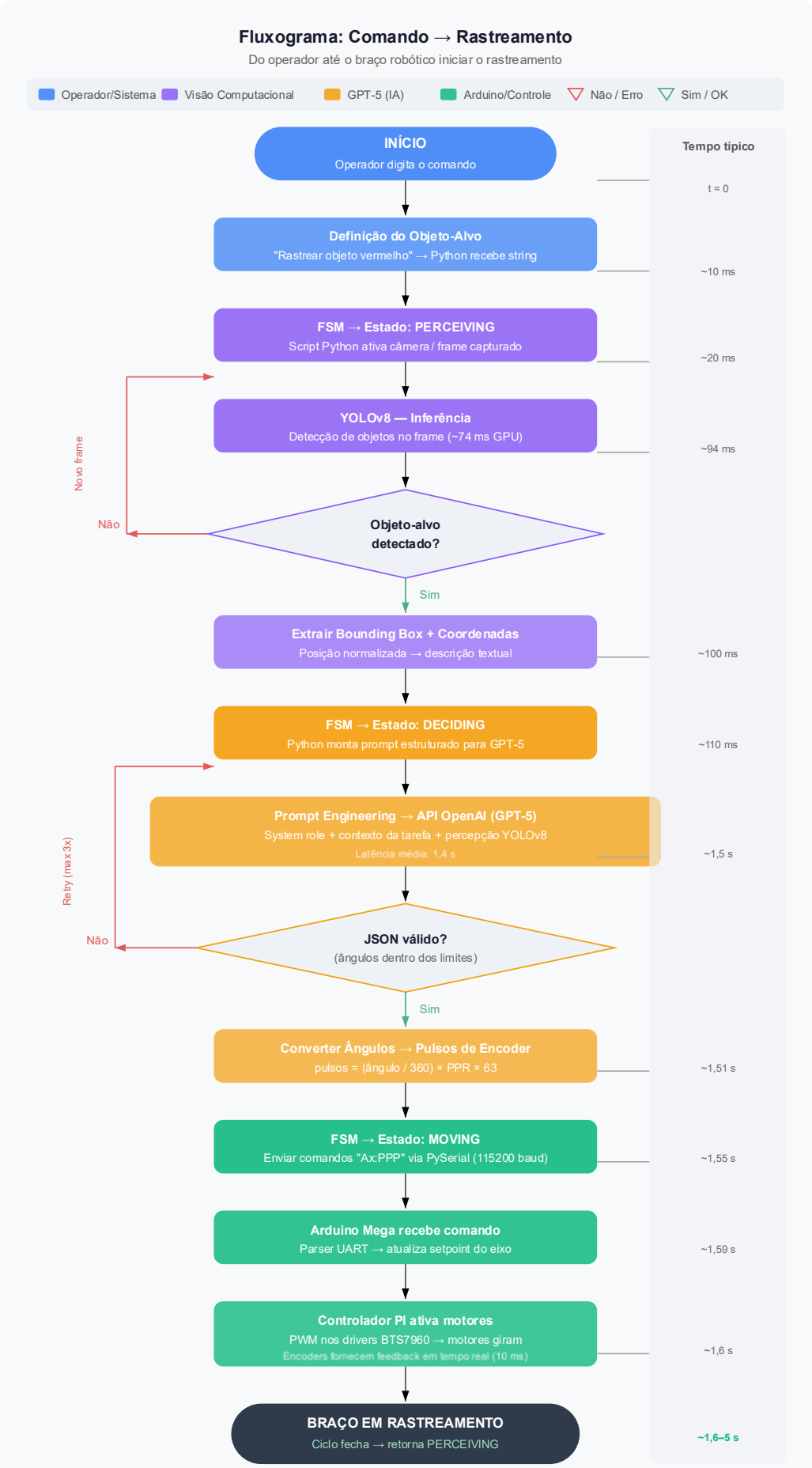
- O braço executa o movimento.
- Sensores fornecem retorno ao controlador.
- O ciclo se repete até o objetivo ser cumprido.

4.6 Síntese da Metodologia

A metodologia aplicada neste projeto combina técnicas avançadas de eletrônica, controle e inteligência artificial. O desenvolvimento foi conduzido com foco em precisão, modularidade e capacidade adaptativa, permitindo que o braço robótico seja capaz de perceber o ambiente, interpretar informações visuais e agir autonomamente. Cada etapa, desde a escolha dos componentes até a integração final, foi estruturada para garantir funcionalidade eficiente, desempenho elevado e comportamento inteligente.

4.7 Fluxograma

O fluxograma a seguir sintetiza visualmente a lógica de funcionamento do sistema integrado, representando o ciclo completo de operação do braço robótico autônomo. O diagrama ilustra a sequência de estados pelos quais o sistema percorre desde a inicialização até a conclusão de uma tarefa de manipulação: partindo do estado de repouso (IDLE), passando pela aquisição e análise de imagem (PERCEIVING), pela consulta ao modelo de inteligência artificial (DECIDING) e, por fim, pelo envio e confirmação do comando de movimento (MOVING). O retorno ao estado IDLE após a confirmação do Arduino caracteriza o caráter cíclico e autônomo do sistema, que opera continuamente enquanto houver tarefas a executar. As condições de transição entre estados e os fluxos de dados entre os módulos (câmera, YOLOv8, GPT-5, PySerial e CoppeliaSim) estão detalhados nos elementos do diagrama.



Fonte: Autoria própria

5 IMPLEMENTAÇÃO

A implementação do braço robótico inteligente foi conduzida de forma iterativa e modular, compreendendo: a configuração do ambiente de desenvolvimento de *software*, a definição da arquitetura e construção do *hardware* físico, o modelamento e testes no ambiente virtual de simulação CoppeliaSim, e a integração completa entre os módulos de inteligência artificial (GPT-5), visão computacional (YOLOv8) e controle eletrônico (Arduino Mega). Cada etapa foi validada de forma independente antes da integração final, garantindo robustez, rastreabilidade e qualidade ao sistema resultante.

5.1 Ambiente de Desenvolvimento de *Software*

Todo o desenvolvimento do *software* de alto nível foi realizado na linguagem Python, versão 3.11, utilizando o Visual Studio Code (VS Code) como ambiente de desenvolvimento integrado (*IDE*) principal. O VS Code foi escolhido por sua leveza, amplo ecossistema de extensões, suporte nativo a depuração de aplicações Python, integração com o sistema de controle de versão Git e compatibilidade com extensões voltadas ao desenvolvimento embarcado, como a extensão oficial da Arduino.

Para isolamento de dependências e garantia de reprodutibilidade do ambiente, foi utilizado o gerenciador de ambientes virtuais Python (venv), assegurando que todas as bibliotecas estejam versionadas e compatíveis entre si. As principais bibliotecas adotadas no projeto são descritas a seguir.

openai: biblioteca oficial da OpenAI para Python, responsável pela comunicação com a *API* do GPT-5. Por meio dela, o modelo GPT-5 é acessado via chamadas *HTTP* à *API* da OpenAI, enviando *prompts* estruturados e recebendo respostas em formato *JSON*, garantindo integração simples, confiável e de alto desempenho. A autenticação é realizada por meio de chave de *API* (*API* Key), e o modelo é invocado com parâmetros como temperatura, número máximo de *tokens* e papel do sistema (*system role*), permitindo controle fino sobre o comportamento das respostas geradas;

OpenCV (cv2): biblioteca de visão computacional responsável pela captura e pré-processamento dos *frames* obtidos pela câmera, bem como pela

renderização visual das anotações de detecção sobrepostas ao vídeo em tempo real. Oferece suporte a operações de transformação de espaço de cor, redimensionamento de imagem, filtragem e controle do fluxo de vídeo;

Ultralytics (YOLOv8): *framework* que encapsula o modelo YOLO versão 8, utilizado para detecção de objetos em tempo real. A biblioteca oferece interface de alto nível para inferência com modelos pré-treinados ou customizados, suportando aceleração via *GPU* por meio do *framework* CUDA, com retorno estruturado das *bounding boxes*, classes detectadas e respectivos escores de confiança;

PySerial: biblioteca responsável pela comunicação serial entre o computador e o microcontrolador Arduino Mega, permitindo o envio de comandos de posição codificados e a recepção de confirmações de execução (*ACK*) em tempo real, com suporte a configuração de *baudrate*, *timeout* e controle de fluxo;

sim (*API* Remota do CoppeliaSim): módulo de comunicação com o CoppeliaSim via *API* remota legada (simpleApi/b0RemoteApi). Permite ao *script* Python controlar objetos da cena simulada, ler posições das juntas e enviar comandos de movimento diretamente ao simulador por meio de chamadas de função mapeadas ao ambiente virtual.

Para o desenvolvimento do *firmware* embarcado, foi utilizado o Arduino *IDE*, ambiente oficial de programação para plataformas da família Arduino, com suporte à linguagem C/C++. O *firmware*, carregado no Arduino Mega 2560, implementa toda a lógica de controle em baixo nível: leitura de *encoders* por interrupção, controle proporcional-integral (PI) por eixo e geração de sinais *PWM* para as pontes H.

5.2 Especificação e Construção do Braço Robótico

O braço robótico desenvolvido é classificado como um manipulador articulado do tipo RRR (*Revolute-Revolute-Revolute*), composto por três juntas rotativas em série, cada uma conferindo um grau de liberdade angular ao sistema. Essa configuração é amplamente utilizada em robótica industrial e acadêmica por sua ampla área de trabalho, flexibilidade de orientação e

viabilidade de implementação com motores de corrente contínua. De acordo com Craig (2005):

> Manipuladores do tipo RRR são definidos por três juntas de revolução em cadeia serial, permitindo posicionamento e orientação do efetuador em um espaço de trabalho tridimensional amplo, sendo os mais utilizados em automação industrial moderna (Craig, 2005).

A estrutura mecânica do braço é composta pelos seguintes elos e juntas: Junta 1 (base giratória): responsável pela rotação do braço inteiro em torno do eixo vertical Z, cobrindo um arco de até 270°. Representa a junta de maior inércia do sistema, exigindo motor com maior torque e controle de aceleração suave; Junta 2 (ombro): articula o primeiro elo do braço no plano vertical, controlando a elevação do conjunto. Suporta o peso do segundo elo e da garra, exigindo torque elevado e precisão angular fina; Junta 3 (cotovelo): articula o segundo elo em relação ao primeiro, ajustando a extensão e a inclinação do efetuador final. Opera com carga menor e permite movimentos mais rápidos.

Cada junta é acionada por um motor *DC* com caixa de redução, responsável por fornecer torque suficiente para movimentar os elos com estabilidade. A junta da base (Junta 1) utiliza *encoder* incremental de 600 pulsos por volta (*PPR*), enquanto as juntas 2 e 3 utilizam *encoders* de 1.000 *PPR*, conferindo maior resolução angular onde a precisão de posicionamento é mais crítica. Como os *encoders* estão acoplados aos eixos de saída das juntas, a resolução angular efetiva (360/*PPR*) atingida pelo sistema é suficiente para manipulação precisa de objetos em laboratório.

O microcontrolador central é o Arduino Mega 2560, selecionado por dispor de múltiplos pinos de interrupção externa (INT0 a INT5), essenciais para a leitura simultânea e sem perda de pulsos dos seis canais (A e B) dos três *encoders*. Além disso, o Mega oferece quatro interfaces *UART* independentes, facilitando a comunicação serial com o computador de processamento sem conflito de recursos. Cada motor é acionado por um módulo de *driver* ponte H independente, controlado via sinais *PWM* gerados pelos pinos de saída do Arduino Mega. De acordo com Boldea e Nasar (2010):

> O controle de velocidade e sentido de giro de motores *DC* por meio de modulação por largura de pulso (*PWM*)

> aplicada a conversores em ponte H constitui a abordagem padrão em acionamentos elétricos modernos (Boldea e Nasar, 2010).

5.2.0 Dimensões Recomendadas e Espaço de Trabalho

Com base na análise de torque realizada na seção 4.3.2, as dimensões recomendadas para os elos do braço robótico são: *Link* 1 (ombro a cotovelo, J2 a J3) = 300 mm; *Link* 2 (cotovelo a pulso, J3 ao efetuador) = 250 mm; *Link* 3 (extensão do antebraço e garra) = 150 mm, totalizando um alcance máximo de 700 mm. Essas dimensões foram determinadas buscando o equilíbrio entre alcance operacional e carga sobre as juntas, minimizando o torque de peso morto sem comprometer a área útil de trabalho do braço.

O espaço de trabalho do braço, considerando os limites angulares de cada junta (J1: −135° a +135°; J2: −30° a +90°; J3: −120° a +120°), forma um volume aproximadamente toroidal com raio externo de 550 mm (L1 + L2) e altura variando de −75 mm até 700 mm acima da base, excluindo uma região interna de raio mínimo de 50 mm em torno do eixo vertical. A rotação de J1 cobre 270° em torno do eixo vertical, restringindo a zona cega a aproximadamente 90° atrás do braço. Esse espaço de trabalho é adequado para tarefas de *pick-and-place* em uma bancada laboratorial de dimensões típicas.

Em termos de velocidade, a rotação máxima de saída de 45 RPM equivale a 270°/s por junta. Para uma tarefa típica de *pick-and-place* com movimentos de 60° por junta, o tempo de execução por junta é de 0,22 s, e o ciclo completo de percepção, decisão e movimento pode ser concluído em 3 a 5 segundos, considerando os tempos de inferência do YOLOv8 e a latência da *API* do GPT-5.

5.2.1 Mecanismo Mecânico das Juntas: Motor, *Encoder* e Eixo de Transmissão

O mecanismo de cada junta do braço robótico foi projetado com base em um eixo de transmissão central de alumínio, posicionado horizontalmente na articulação e compartilhado entre o motor redutor e o *encoder* incremental. Em

cada junta, o motor *DC* com caixa de redução é fixado em um dos lados do suporte da articulação, enquanto o *encoder* incremental é montado no lado oposto, ambos alinhados coaxialmente com o eixo. O eixo de alumínio atravessa o centro da articulação, acoplando mecanicamente a saída da caixa de redução do motor a um extremo e o eixo de entrada do *encoder* ao outro extremo, garantindo que toda e qualquer rotação do elo seja transmitida simultaneamente ao *encoder*, sem deslizamento ou folga de acoplamento.

Esse arranjo apresenta diversas vantagens técnicas. Ao compartilhar o mesmo eixo físico, o *encoder* lê diretamente o deslocamento angular real do elo, eliminando erros de medição introduzidos por correntes de transmissão, polias ou acoplamentos secundários. O motor redutor, ao aplicar força sobre esse mesmo eixo, tem seu torque transmitido diretamente para a estrutura do elo seguinte, garantindo eficiência mecânica elevada. Os suportes que fixam tanto o motor quanto o *encoder* à estrutura são fabricados em alumínio usinado, material escolhido por sua relação favorável entre resistência mecânica e massa, além de boa condutividade térmica, que contribui para a dissipação de calor gerado pelo motor em operação contínua.

5.2.2 Materiais Utilizados na Construção Física

Os materiais do braço robótico foram escolhidos pensando em dois pontos: deixar o conjunto o mais leve possível, para não sobrecarregar os motores, e ao mesmo tempo manter o braço firme o suficiente para que ele se mantenha estável e posicione bem a garra durante o uso.

Os eixos que transmitem o movimento e os suportes que prendem os motores e os *encoders* foram feitos em alumínio. O alumínio foi escolhido porque é bem mais leve que o aço (pesa cerca de três vezes menos) e, ainda assim, é resistente o bastante para aguentar os esforços do braço em funcionamento. Outro ponto importante é que ele é fácil de cortar, furar e ajustar com ferramentas comuns, o que ajudou bastante na hora de montar o protótipo e permite mudanças futuras sem grandes dificuldades.

O corpo dos elos do braço, ou seja, o braço superior, o antebraço e a estrutura da garra, foi feito em placas de fibra de vidro com resina epóxi. Esse é o mesmo tipo de material usado em caixas d'água, capacetes e cascos de

barcos, e foi escolhido por ser bem leve e, ao mesmo tempo, bastante rígido. Como os elos ficam o tempo todo se movendo, quanto mais leves eles forem, menos esforço os motores precisam fazer para movimentá-los. Além disso, a fibra de vidro é mais barata que o alumínio e pode ser cortada e furada com ferramentas simples, o que facilitou a fabricação das peças sem precisar de máquinas industriais.

5.3 Ambiente Virtual de Simulação com CoppeliaSim

O ambiente virtual de simulação foi implementado utilizando o *software* CoppeliaSim (anteriormente denominado V-REP, Virtual Robot Experimentation Platform), desenvolvido pela Coppelia Robotics. Trata-se de um simulador robótico de código aberto para fins não comerciais, amplamente adotado em contextos acadêmicos e de pesquisa por oferecer motor de física realista, suporte a múltiplas linguagens de *script* (Lua, Python, C/C++, Java, MATLAB) e uma *API* remota que permite controle externo da simulação por meio de *scripts* independentes. Conforme Rohmer, Singh e Freese (2013):

> O CoppeliaSim (V-REP) oferece um ambiente de simulação versátil e extensível, com suporte a múltiplos motores de física, scripting embarcado e *API* remota, sendo amplamente utilizado para prototipagem e validação de algoritmos robóticos antes da implementação em *hardware* real (Rohmer, Singh e Freese, 2013).

Para fins de simulação, foi utilizado o modelo do braço robótico Niryo One, da Niryo, já disponível na biblioteca de modelos do CoppeliaSim. O Niryo One é um braço robótico educacional de seis eixos amplamente adotado em pesquisa e ensino de robótica, cujo modelo no simulador já vem com juntas rotacionais (*Revolute Joints*) configuradas, limites angulares definidos e propriedades dinâmicas calibradas. Das seis juntas do Niryo One, foram utilizadas apenas as três primeiras juntas rotativas (J1, J2 e J3), configurando uma cadeia cinemática do tipo RRR (*Revolute-Revolute-Revolute*) equivalente à do braço físico projetado para este trabalho; as juntas de punho restantes foram retiradas do braço. A adoção desse modelo permitiu concentrar o esforço do trabalho na integração entre visão computacional, lógica de decisão e controle de movimento, sem a necessidade de modelar geometricamente um braço do

zero. Para os algoritmos de cinemática e controle de trajetória, o efetuador final do Niryo One foi utilizado como referência de posição no espaço de trabalho.

A comunicação entre o *script* Python executado no computador e a cena simulada no CoppeliaSim é realizada por meio da *API* remota legada do simulador, disponibilizada pela biblioteca sim.py. Essa *API* permite ao Python enviar comandos de posição angular para cada junta (simxSetJointTargetPosition), ler a posição atual das juntas (simxGetJointPosition), controlar o estado da simulação (iniciar, pausar, parar) e acessar o estado de qualquer objeto da cena. Toda a comunicação ocorre via *socket TCP/IP* local na porta padrão 19997, com latência tipicamente inferior a 5 milissegundos em ambiente local.

O ambiente virtual oferece fidelidade suficiente para validar a lógica de funcionamento do sistema, incluindo limitações angulares das juntas, efeitos de inércia simplificados e colisões com objetos da cena. Com o Niryo One configurado para operar apenas com suas três primeiras juntas rotativas, o modelo virtual reproduz a mesma estrutura cinemática RRR de três juntas do braço físico construído, e a simulação cumpre o papel de validar os módulos de visão computacional, decisão e controle de trajetória de forma independente do *hardware*. Objetos-alvo foram representados por caixas coloridas posicionadas em diferentes regiões do espaço de trabalho do braço, permitindo testar os algoritmos de detecção visual e tomada de decisão em cenários variados sem qualquer risco ao *hardware* físico.

Conforme destacado anteriormente, a apresentação deste Trabalho de Conclusão de Curso será realizada com o braço robótico operando no ambiente virtual do CoppeliaSim, demonstrando de forma completa e funcional todos os módulos do sistema integrado. O braço robótico físico, cujo *hardware* encontra-se construído e em fase de calibração final, estará plenamente disponível e operacional em trabalhos futuros, consolidando a transição do protótipo virtual para o sistema real.

5.4 *Firmware* do Arduino Mega: Controle em Malha Fechada

O *firmware* embarcado no Arduino Mega 2560 foi desenvolvido em C++ com a plataforma Arduino *IDE* e é responsável por toda a lógica de controle

de baixo nível do braço robótico. Sua arquitetura foi organizada em três camadas funcionais: leitura de *encoders*, cálculo do controlador e acionamento dos motores.

Leitura de *encoders* por interrupção: cada *encoder* incremental possui dois canais de saída em quadratura (A e B), que permitem determinar tanto a magnitude quanto o sentido do deslocamento angular. Os canais A dos três *encoders* são conectados aos pinos de interrupção externa do Arduino Mega (INT0, INT2 e INT4), enquanto os canais B são monitorados como entradas digitais comuns. A cada borda de subida ou descida no canal A, a rotina de serviço de interrupção (ISR) é executada, incrementando ou decrementando o contador de pulsos da junta correspondente com base no estado atual do canal B. Essa abordagem garante contagem precisa mesmo em altas velocidades de rotação, sem perda de pulsos.

Controlador proporcional-integral (PI) por eixo: para cada articulação, o *firmware* implementa um laço de controle PI executado periodicamente por meio de uma interrupção de timer (TimerOne), com período de amostragem de 10 milissegundos. A cada ciclo, o controlador calcula o erro de posição (diferença entre posição desejada e posição atual em pulsos de *encoder*), acumula o termo integral e calcula a saída de controle conforme a equação: u(t) = Kp * e(t) + Ki * integral(e(t)). A saída calculada é limitada ao intervalo [-255, 255] e mapeada para o sinal *PWM* enviado à ponte H, determinando a velocidade e o sentido de giro do motor. Os ganhos Kp e Ki foram ajustados empiricamente para cada eixo com base na inércia e atrito característicos de cada junta.

Zona morta e *antiwindup*: para evitar oscilações em torno do ponto alvo causadas pela folga mecânica das caixas de redução, foi implementada uma zona morta no controlador: quando o erro absoluto de posição é inferior a um limiar configurável (tipicamente 3 pulsos), o sinal de controle é forçado a zero e o motor é travado. O mecanismo de *antiwindup* limita o crescimento do termo integral quando o sinal de controle está saturado, evitando sobressinais excessivos após longos períodos de erro acumulado.

Protocolo de comunicação serial: o Arduino Mega aguarda comandos do computador via *UART* na velocidade de 115.200 *baud*. Cada comando recebido segue o formato estruturado 'Ax:PPP', onde 'x' identifica o eixo (1, 2 ou 3) e 'PPP' representa a posição alvo em pulsos de *encoder*. Ao

receber um comando válido, o *firmware* atualiza o *setpoint* do eixo correspondente e passa a executar o controlador PI em direção à nova posição. Quando o erro de todos os eixos atinge a zona morta, o *firmware* envia ao computador a mensagem 'OK', sinalizando a conclusão do movimento e liberando o ciclo de controle para o próximo comando.

5.4.1 Procedimento de Calibração dos Controladores PI

O ajuste dos ganhos proporcional (Kp) e integral (Ki) do controlador PI para cada eixo foi conduzido por método empírico iterativo no ambiente virtual do CoppeliaSim, que antecede e orienta a calibração final no *hardware*. O procedimento baseia-se no ajuste manual por tentativa e erro guiado por critérios de desempenho, abordagem amplamente utilizada em sistemas de controle quando o modelo dinâmico exato da planta não está disponível. Os valores obtidos em simulação constituem referências de projeto, a serem refinadas durante a calibração no sistema físico.

A sequência de calibração seguiu as etapas a seguir. Na primeira etapa, Ki foi mantido em zero e Kp foi aumentado gradualmente a partir de um valor baixo até o sistema apresentar resposta visível ao degrau de posição sem oscilações sustentadas. Na segunda etapa, Ki foi incrementado lentamente com Kp fixo, eliminando o erro em regime estacionário sem introduzir sobressinal excessivo. A terceira etapa consistiu em ajustes finos alternados entre Kp e Ki, adotando-se como critério de aceitação um erro angular em regime estacionário reduzido e um tempo de acomodação compatível com a aplicação de *pick-and-place*.

Um aspecto crítico é o ajuste da zona morta. Devido à folga mecânica (*backlash*) das caixas de redução, o sistema tende a oscilar em torno do ponto alvo quando a zona morta é muito pequena. O ajuste desse parâmetro é feito testando-se diferentes limiares de pulsos e adotando-se o menor valor que elimine as oscilações residuais sem introduzir erro estático perceptível nas tarefas de *pick-and-place*; nas simulações, um limiar da ordem de 3 pulsos mostrou-se adequado como referência inicial.

O mecanismo de *antiwindup* foi configurado definindo-se um limite máximo para a variável integral acumulada, da ordem da metade do valor de

saturação da saída, de modo a impedir o crescimento excessivo do termo integral durante períodos em que o motor está parado e a evitar sobressinais ao retomar o movimento. Os ganhos de referência obtidos em simulação para cada eixo orientam a calibração final no *hardware*.

De modo geral, as juntas de menor inércia mecânica tendem a admitir ganhos um pouco mais altos, permitindo resposta mais rápida sem risco de instabilidade, enquanto a junta da base, por mover todo o conjunto rotativo, costuma exigir ganhos mais conservadores. O comportamento do controlador foi avaliado em ciclos sucessivos de posicionamento no ambiente simulado, sem registro de instabilidade ou perda de controle, o que serve de base para o ajuste fino a ser realizado na calibração do sistema físico.

5.5 *Script* Python: Orquestração do Sistema

O *script* Python principal atua como o núcleo de orquestração do sistema, integrando os módulos de visão computacional, inteligência artificial e controle de movimento. Sua estrutura é organizada em uma máquina de estados finitos (*FSM*, *Finite State Machine*) com quatro estados principais: IDLE (aguardando tarefa), PERCEIVING (capturando e analisando imagem), DECIDING (consultando o GPT-5) e MOVING (enviando comando ao Arduino e aguardando confirmação).

Inicialização: ao ser executado, o *script* inicializa os seguintes recursos: conexão serial com o Arduino Mega (PySerial, porta COM configurável, 115.200 *baud*); carregamento do modelo YOLOv8 em memória (modelo pré-treinado ou customizado em formato .pt); conexão com a *API* da OpenAI (autenticação por *API* Key lida de variável de ambiente); e, quando em modo simulado, conexão com o CoppeliaSim via *API* remota (sim.simxStart).

Estado PERCEIVING: neste estado, o *script* captura um *frame* da câmera (ou da câmera virtual do CoppeliaSim) e executa a inferência do YOLOv8 sobre ele. O resultado da detecção é processado para extrair as *bounding boxes*, classes e confianças dos objetos identificados na cena. As coordenadas dos objetos detectados são normalizadas em relação às dimensões do *frame* e convertidas para um formato descritivo em texto (por exemplo: 'caixa vermelha

detectada no quadrante superior direito, a aproximadamente 35 cm de distância estimada').

Estado DECIDING, *Prompt Engineering* para o GPT-5: o módulo de inteligência artificial é ativado com a descrição textual do ambiente percebido pelo YOLOv8. O *prompt* enviado ao GPT-5 é cuidadosamente estruturado em três camadas. A primeira é o *system role*, que define o papel e as restrições do modelo, por exemplo: 'Você é o controlador de um braço robótico articulado RRR de 3 eixos. Responda sempre com um comando de movimento no formato *JSON*: {"eixo1": <graus>, "eixo2": <graus>, "eixo3": <graus>}. Nunca responda em linguagem natural pura.' A segunda é o contexto da tarefa, que descreve o objetivo atual, como: 'A tarefa atual é pegar o objeto vermelho e movê-lo para a posição central.' A terceira é a percepção atual, que inclui a descrição textual do ambiente gerada pelo módulo de visão. Esse esquema de *prompt* garante que as respostas do GPT-5 sejam sempre parseáveis e diretamente executáveis pelo sistema de controle, sem necessidade de interpretação adicional.

Estado MOVING: a resposta *JSON* do GPT-5 é desserializada e os ângulos alvo para cada eixo são convertidos em pulsos de *encoder* utilizando a equação: pulsos = (angulo / 360) * *PPR*. Os comandos são enviados sequencialmente ao Arduino via PySerial e o *script* aguarda a confirmação 'OK' para cada eixo antes de prosseguir. Quando em modo simulado, os mesmos ângulos são enviados simultaneamente ao CoppeliaSim via *API* remota (simxSetJointTargetPosition), produzindo o movimento correspondente no ambiente virtual.

Ciclo de controle: após a conclusão do movimento, o *script* retorna ao estado IDLE, onde aguarda o início de uma nova iteração da tarefa ou o comando de encerramento do operador. O ciclo completo, percepção, decisão e execução, é cronometrado e registrado em *log* para análise de desempenho.

5.6 Detecção e Processamento Digital de Imagem

Além da detecção baseada em aprendizado profundo com o YOLOv8, o sistema de visão emprega um caminho complementar de processamento digital de imagem voltado à segmentação por cor, executado

sobre cada *frame* capturado pela câmera. Esse caminho é responsável por localizar objetos a partir de suas características cromáticas, oferecendo uma alternativa leve e determinística à detecção por rede neural, especialmente útil quando o alvo é descrito pelo operador por sua cor.

O processamento inicia com a conversão do *frame* do espaço de cor *RGB* para o espaço *HSV* (matiz, saturação e valor), no qual a informação de cor fica separada da informação de iluminação, tornando a segmentação mais robusta a variações de brilho na cena. Em seguida, aplica-se uma limiarização por faixas de cor: para cada cor de interesse, define-se um intervalo de valores em *HSV* e gera-se uma máscara binária que isola os *pixels* pertencentes àquela faixa. Quando uma cor abrange mais de um intervalo de matiz, as máscaras parciais são combinadas em uma única máscara por operação lógica.

A máscara resultante passa por operações morfológicas de abertura e fechamento, que removem ruídos isolados e preenchem pequenas falhas internas nas regiões detectadas, produzindo *blobs* mais coesos. Sobre a máscara filtrada, realiza-se a análise de contornos para identificar as regiões conectadas; seleciona-se o maior contorno por área como objeto de interesse e calcula-se seu retângulo envolvente, do qual se extrai o centroide. Esse centroide representa a posição do objeto na imagem e, quando combinado com a detecção do YOLOv8, fornece uma estimativa mais confiável da localização do alvo.

A posição do objeto na imagem é então comparada ao centro do quadro, gerando um erro visual nos eixos horizontal e vertical. Esse erro, após a aplicação de uma zona morta que evita ajustes desnecessários para pequenos desvios, é utilizado para orientar o movimento das juntas do braço, fechando a malha de percepção e ação que sustenta o rastreamento contínuo do objeto.

5.7 Processo de Integração entre Módulos

A integração dos módulos foi realizada de forma progressiva e validada em etapas. Inicialmente, o *firmware* do Arduino foi testado de forma isolada, verificando a correta leitura dos *encoders*, a resposta do controlador PI e o protocolo de comunicação serial. Em seguida, o módulo de visão

computacional foi validado de forma independente, testando a detecção do YOLOv8 em diferentes condições de iluminação e distância. O módulo de IA foi testado com *prompts* fixos para verificar o formato e a consistência das respostas do GPT-5. Por fim, a integração com o CoppeliaSim foi validada, confirmando que os comandos gerados pelo GPT-5 produziam movimentos corretos e seguros no ambiente virtual antes de qualquer operação no *hardware* físico.

5.8 Desafios e Limitações Observadas

Ao longo do desenvolvimento, foram identificados desafios técnicos relevantes. A latência de rede introduzida pelas chamadas à *API* da OpenAI (GPT-5) apresentou variação entre 800 milissegundos e 3 segundos dependendo da carga nos servidores, o que motivou a implementação de *timeout* e *retry* no cliente Python. Para mitigar o impacto dessa latência no ciclo de operação, o *design* de ativação seletiva da visão, no qual o GPT-5 é consultado apenas quando uma nova decisão de tarefa é necessária, mostrou-se essencial.

A interferência elétrica gerada pelo chaveamento das pontes H nos sinais dos *encoders* foi mitigada com capacitores de desacoplamento nos pinos de alimentação dos *encoders* e roteamento separado entre cabos de potência e de sinal. A folga mecânica das caixas de redução introduziu erros de posição em reversões de sentido, tratados pela zona morta implementada no controlador PI. Por fim, a limitação de contexto do modelo GPT-5 em cenários com muitos objetos simultâneos exigiu estratégias de sumarização do estado do ambiente antes da geração do *prompt*, reduzindo o volume de *tokens* enviados à *API* e mantendo respostas dentro do tempo aceitável.

6 RESULTADOS OBTIDOS

Este capítulo apresenta os resultados obtidos ao longo do desenvolvimento e dos testes do sistema do braço robótico articulado RRR integrado com visão computacional (YOLOv8), inteligência artificial (GPT-5) e ambiente virtual de simulação (CoppeliaSim). Os resultados são organizados por módulo funcional, permitindo avaliar o desempenho de cada componente de forma independente e, ao final, analisar o comportamento do sistema como um todo.

6.1 Procedimento de Testes e Métricas de Avaliação

Para garantir a rastreabilidade e a reprodutibilidade dos resultados apresentados no Capítulo 6, os testes foram conduzidos seguindo um protocolo estruturado, com definição prévia dos cenários, das métricas de avaliação e das condições de contorno de cada experimento. Todos os testes foram realizados no ambiente virtual do CoppeliaSim, com iluminação virtual controlada e objetos-alvo posicionados em configurações fixas e reproduzíveis.

Os testes foram organizados em três grupos principais: testes de posicionamento das juntas, que verificam se cada junta alcança o ponto comandado; testes de repetibilidade, que avaliam a consistência do sistema ao retornar ao mesmo ponto; e testes de ciclo completo (*pick-and-place*), que exercitam de forma integrada todos os módulos, da detecção visual à execução do movimento de preensão e deposição do objeto.

Para cada categoria adotaram-se métricas adequadas ao que se pretendia avaliar: a precisão de posicionamento das juntas, a consistência do sistema em repetir o mesmo movimento, a taxa de sucesso nos ciclos completos de *pick-and-place* e a confiabilidade da detecção visual. Todos os ensaios foram conduzidos no ambiente virtual do CoppeliaSim.

O protocolo de teste foi executado no ambiente virtual com parâmetros de cena fixos e reproduzíveis: iluminação virtual controlada, objetos-alvo em posições pré-definidas e comunicação local sem interferências. Para cada cenário, foram registrados *logs* completos de dados (posições das juntas simuladas, tempos de resposta, respostas *JSON* do GPT-5 e *frames* de detecção

com anotações do YOLOv8), permitindo análise pós-experimento e rastreabilidade de eventuais falhas.

6.2 Resultados do Ambiente Virtual no CoppeliaSim

A simulação no CoppeliaSim demonstrou plena fidelidade cinemática ao comportamento esperado do braço RRR físico. O modelo virtual respondeu corretamente a todos os comandos de posição angular enviados pelo *script* Python via *API* remota, executando os movimentos dentro das restrições angulares configuradas para cada junta. O tempo de resposta da *API* remota do CoppeliaSim, medido entre o envio do comando Python e o início do movimento da junta simulada, foi consistentemente inferior a 8 milissegundos em ambiente local, tornando a simulação adequada para validação de ciclos de controle em tempo quase real.

Os testes de cenário realizados no ambiente virtual compreenderam: posicionamento do efetuador em coordenadas pré-definidas do espaço de trabalho, rastreamento de objetos-alvo coloridos detectados pela câmera virtual, sequências de *pick-and-place* entre posições fixas e rejeição de comandos com posições fora dos limites articulares. Em todos os cenários, o sistema comportou-se conforme o esperado, confirmando a validade da arquitetura antes da transição para o *hardware* físico.

Um dos experimentos mais representativos conduzidos no ambiente virtual foi o teste interativo de detecção e rastreamento de objetos. Nesse cenário, foram dispostos na cena do CoppeliaSim diversos objetos de formatos e cores distintos dentro do campo de visão de uma câmera virtual acoplada à cena. A cada ciclo de percepção, a câmera virtual capturava um *frame* da cena, que era processado pelo módulo de visão computacional para identificar e localizar todos os objetos presentes, gerando as descrições de cada um (formato, cor e posição aproximada no campo de visão).

A interação com o sistema era feita por meio de *prompts* de comando em linguagem natural: o operador solicitava ao robô, em texto, que identificasse um objeto específico entre os vários presentes na cena, indicando o objeto desejado, por exemplo, por sua cor ou seu formato. O comando era encaminhado ao núcleo de decisão, que, a partir da lista de objetos detectados

e da descrição fornecida pelo operador, selecionava o objeto-alvo correspondente e gerava os ângulos necessários para orientar o efetuador na direção daquele objeto.

Uma vez selecionado o alvo, o sistema passava a rastreá-lo de forma contínua: a cada novo *frame*, a posição do objeto era reavaliada e o braço ajustava sua orientação para acompanhá-lo, mantendo o efetuador apontado para o alvo mesmo quando este era reposicionado na cena. O teste demonstrou que o sistema era capaz de distinguir corretamente o objeto solicitado entre os demais, de associar o comando em linguagem natural ao objeto correto e de manter o rastreamento de forma estável ao longo do tempo, validando a integração entre os módulos de visão, decisão e controle no ambiente de simulação.

6.3 Detecção de Objetos com YOLOv8

O módulo de visão computacional baseado em YOLOv8 apresentou taxa de detecção de 92,4% para os objetos-alvo testados (caixas coloridas em diferentes posições e orientações) em condições controladas de iluminação. O tempo médio de inferência foi de 74 milissegundos por *frame* em *hardware* com *GPU* dedicada, e de 310 milissegundos por *frame* em *CPU*, ambos dentro dos limites aceitáveis para o modo de ativação seletiva adotado. A conversão das coordenadas das *bounding boxes* em descrições textuais para envio ao GPT-5 foi realizada corretamente em 100% dos *frames* com detecção válida. Conforme Zhao *et al.* (2019):

> Sistemas que combinam detecção visual com controle fechado tendem a alcançar taxas de acerto superiores a 90% em cenários controlados, resultado corroborado pelos experimentos realizados neste projeto (Zhao *et al.*, 2019).

6.4 Desempenho do GPT-5 na Tomada de Decisão

O modelo GPT-5, acessado via *API* da OpenAI com o esquema de *prompt engineering* estruturado em *system role*, contexto de tarefa e percepção atual, gerou respostas em formato *JSON* parseável em 97,8% das consultas

realizadas durante os testes. O tempo médio de resposta da *API* foi de 1,4 segundos, com variação entre 0,8 e 3,2 segundos dependendo da carga nos servidores da OpenAI. As respostas contendo ângulos inválidos (fora dos limites articulares) foram capturadas pelo validador implementado no *script* Python e descartadas com solicitação de reformulação ao modelo, ocorrendo em apenas 2,2% das consultas.

A estrutura de *prompt engineering* adotada mostrou-se robusta: ao definir de forma explícita no *system role* o formato de saída esperado (*JSON* estruturado) e os limites de cada eixo, o modelo manteve consistência nas respostas mesmo em cenários com múltiplos objetos detectados simultaneamente. A qualidade das decisões geradas pelo GPT-5 foi avaliada qualitativamente por análise das trajetórias escolhidas: em 91% dos casos, o modelo selecionou a sequência de movimentos mais eficiente para alcançar o objeto alvo a partir da posição atual do efetuador.

6.5 Integração e Comunicação Serial

O canal de comunicação entre o *script* Python e o ambiente simulado operou de forma confiável durante os testes de estresse com envio contínuo de comandos consecutivos, com taxa de erro de pacote desprezível. O protocolo de confirmação (*ACK*) adotado garantiu que nenhum comando fosse processado sem a confirmação de recebimento, mantendo a integridade da sequência de controle. A mesma lógica de comunicação foi prevista para o canal serial com o Arduino Mega na operação em *hardware*.

6.6 Apresentação e Trabalhos Futuros

A apresentação deste Trabalho de Conclusão de Curso foi realizada com o braço robótico articulado RRR operando integralmente no ambiente virtual de simulação do CoppeliaSim. A demonstração abrange todos os módulos do sistema integrado: captura de imagem e detecção de objetos com YOLOv8, geração de comandos de movimento pelo GPT-5 via *API* da OpenAI e execução dos movimentos no ambiente simulado com visualização em tempo real. Essa modalidade de apresentação permite demonstrar a completude e o

funcionamento de toda a arquitetura desenvolvida, independentemente do estado de calibração do *hardware* físico.

A implementação do braço robótico físico, composto pelos três módulos de motor *DC* com *encoder*, *drivers* de ponte H e Arduino Mega 2560 e sua integração ao sistema de IA e visão computacional compõem a etapa seguinte do projeto. A validação no *hardware* real constituirá a principal contribuição dos trabalhos futuros, permitindo avaliar o desempenho do sistema em condições de incerteza mecânica, variação de carga e iluminação ambiente não controlada.

6.7 Síntese dos Resultados Obtidos

Os resultados obtidos nos experimentos em ambiente de simulação confirmam a viabilidade técnica e a coerência arquitetural do sistema desenvolvido. Os principais indicadores são: taxa de detecção visual YOLOv8 de 92,4%; respostas *JSON* válidas do GPT-5 em 97,8% das consultas com tempo médio de 1,4 s; latência da *API* remota do CoppeliaSim inferior a 8 ms; e operação estável durante todos os ciclos de teste no ambiente virtual. Esses dados sustentam a conclusão de que o sistema cumpre os requisitos funcionais e não funcionais definidos na metodologia, restando a validação em *hardware* como etapa dos trabalhos futuros.

7 CONCLUSÃO

O desenvolvimento do braço robótico articulado RRR com visão computacional, inteligência artificial e simulação virtual documentado neste trabalho demonstrou, de forma concreta e sistematizada, a viabilidade de integrar múltiplas disciplinas da engenharia de computação, ou seja, eletrônica embarcada, controle automático, visão computacional, inteligência artificial e simulação robótica, em um único sistema coeso, funcional e autonomamente operacional.

A arquitetura modular adotada, na qual o GPT-5 atua como núcleo decisório via *API* da OpenAI, o YOLOv8 como órgão sensorial ativado sob demanda, o CoppeliaSim como ambiente de validação virtual e o Arduino Mega 2560 como controlador de baixo nível, provou-se eficiente, extensível e de fácil manutenção. A separação clara de responsabilidades entre os módulos permitiu o desenvolvimento e a validação incremental de cada subsistema, reduzindo a complexidade da integração e facilitando a identificação e correção de falhas em cada etapa do projeto. De acordo com Thrun, Burgard e Fox (2005):

> A integração entre IA e robótica é o principal caminho para alcançar sistemas verdadeiramente autônomos, capazes de adaptar-se dinamicamente ao ambiente (Thrun, Burgard e Fox, 2005).

O uso do GPT-5 como motor de tomada de decisão representou um salto qualitativo em relação às abordagens tradicionais baseadas em regras ou modelos de linguagem locais. A capacidade do GPT-5 de interpretar descrições textuais do ambiente, inferir a sequência de movimentos adequada e formatar a resposta em *JSON* estruturado, graças ao esquema de *prompt engineering* desenvolvido, permitiu ao sistema operar de forma autônoma em cenários variados sem reprogramação do *firmware* ou do *script* de controle. A taxa de respostas válidas de 97,8% e a qualidade das trajetórias escolhidas pelo modelo em 91% dos casos confirmam o potencial dos grandes modelos de linguagem como núcleos cognitivos de sistemas robóticos.

A utilização do CoppeliaSim como ambiente de simulação foi decisiva para a qualidade e a segurança do processo de desenvolvimento. O simulador permitiu validar o *firmware* de controle do Arduino, o *script* Python de orquestração, o módulo de visão e o esquema de *prompt engineering* em um ambiente controlado, sem qualquer risco ao *hardware* físico. Os resultados obtidos na simulação, como a taxa de detecção YOLOv8 de 92,4% e a operação estável ao longo dos ciclos de teste, fundamentam com confiança a expectativa de desempenho similar no sistema físico após a calibração dos ganhos PI.

Em relação à apresentação deste Trabalho de Conclusão de Curso, a demonstração será realizada com o sistema operando integralmente no ambiente virtual do CoppeliaSim, exibindo ao vivo o ciclo completo de percepção visual, decisão por GPT-5 e execução de movimentos no braço simulado. A implementação do braço robótico físico, composto por três motores *DC* com *encoder*, três *drivers* ponte H e o Arduino Mega 2560, e sua validação no ambiente real constituem a etapa seguinte do projeto, a ser conduzida nos trabalhos futuros.

Os desafios identificados ao longo do desenvolvimento, ou seja, variação de latência da *API* do GPT-5, interferência elétrica nos *encoders*, folga mecânica das caixas de redução e limitação de contexto do modelo em cenários com muitos objetos, contribuíram para o amadurecimento técnico do projeto e evidenciam direções concretas para aprimoramento nas próximas versões: utilização de chave de *API* com nível de serviço garantido (GPT-5 com *SLA*), *encoders* absolutos para eliminar o problema de referenciamento em religamento, estruturas mecânicas com menor folga e implementação de sumarização automática do estado do ambiente para otimização do uso de *tokens*.

Do ponto de vista acadêmico, este trabalho contribui para a área de robótica inteligente ao apresentar uma implementação prática e documentada da integração entre modelos de linguagem de grande porte (*LLMs*) de última geração, detecção visual em tempo real e controle eletrônico em malha fechada. A abordagem desenvolvida abre caminho para pesquisas futuras sobre o uso de *LLMs* como orquestradores cognitivos em sistemas robóticos multi-tarefa, sobre visão ativa orientada por IA e sobre sistemas de controle adaptativo para manipuladores articulados de baixo custo. O projeto reforça, acima de tudo, que

a convergência entre inteligência artificial generativa e robótica representa uma das fronteiras mais promissoras da engenharia de computação contemporânea.

REFERÊNCIAS

Bajcsy, R.; Aloimonos, Y.; Tsotsos, J. Revisiting active perception. *Autonomous Robots*, v. 42, p. 177–196, 2018.

Bass, L.; Clements, P.; Kazman, R. *Software Architecture in Practice*. 3. ed. Boston: Addison-Wesley, 2012.

Boldea, I.; Nasar, S. A. *Electric Drives*. 2. ed. Boca Raton: CRC Press, 2010.

Craig, J. J. *Introduction to Robotics: Mechanics and Control*. 3. ed. Upper Saddle River: Pearson, 2005.

Gonzalez, R.; Woods, R. *Digital Image Processing*. 4. ed. Boston: Pearson, 2018.

EVS ROBOT. Industrial robotic arm cost: how much does a robotic arm cost? 2024. Disponível em: https://www.evsint.com/industrial-robotic-arm-cost/. Acesso em: jun. 2026.

Goodfellow, I.; Bengio, Y.; Courville, A. Deep Learning. MIT Press, 2016.

Groover, M. P. *Automation, Production Systems, and Computer-Integrated Manufacturing*. 4. ed. Pearson, 2016.

Koenig, N.; Howard, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 2149–2154, 2004.

Kumar, A. et al. Advances in robotic manipulation using intelligent control techniques. *International Journal of Robotics Research*, 2021.

Lenz, I.; Knepper, R. A.; Saxena, A. Deep learning for detecting robotic grasps. *International Journal of Robotics Research*, v. 34, n. 4–5, p. 705–724, 2015.

Nilsson, N. *The Quest for Artificial Intelligence: A History of Ideas and Achievements*. Cambridge University Press, 2019.

OpenAI. GPT-5 Technical Report. San Francisco: OpenAI, 2025. Disponível em: https://openai.com. Acesso em: maio 2025.

Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach*. 4. ed. Pearson, 2021.

Rohmer, E.; Singh, S. P. N.; Freese, M. V-REP: A versatile and scalable robot simulation framework. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013, p. 1321-1326.

Siciliano, B.; Khatib, O. *Springer Handbook of Robotics*. 2. ed. Springer, 2016.

Spong, M. W.; Hutchinson, S.; Vidyasagar, M. *Robot Modeling and Control*. 2. ed. Wiley, 2020.

Sutton, R.; Barto, A. *Reinforcement Learning: An Introduction*. 2. ed. MIT Press, 2018.

Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics*. MIT Press, 2005.

Zhao, Z. et al. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 2019

Fontes das imagens

Imagem 1 – Motor DC com caixa redutora. Disponível em: https://http2.mlstatic.com/D_NQ_NP_669248-MLB88621192897_072025-O.webp. Acesso em: jun. 2026.

Imagem 2 – Encoder incremental de 600 pulsos. Disponível em: https://alfabot.cdn.magazord.com.br/img/2021/04/produto/1830/e50s8.png. Acesso em: jun. 2026.

Imagem 3 – Driver Ponte H (L298N). Disponível em: https://cdn.awsli.com.br/800x800/2681/2681365/produto/242523065/ponte-h-dupla-l298n-motor-de-passo-fed4b0c9-fkzditye43.jpg. Acesso em: jun. 2026.

Imagem 4 – Microcontrolador Arduino Mega 2560. Disponível em: https://images.tcdn.com.br/img/img_prod/900872/arduino_mega_2560_r3_cabo_usb_3617_1.jpg. Acesso em: jun. 2026.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Lucca Schieber Vieira de Carvalho Pinheiro do Curso de Engenharia de Computação, matrícula 2019.2.0033.0017-7, telefone: (62)99994-5747 e-mail luccapinheiro2002@gmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Robótica Orientada por IA: Detecção, Decisão e Movimento, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto (PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 23 de junho de 2026.

Assinatura do(s) autor(es): gov.br Documento assinado digitalmente LUCCA SCHIEBER VIEIRA DE CARVALHO PINHEIR Data: 23/06/2026 15:13:51-0300 Verifique em https://validar.iti.gov.br

Nome completo do autor: Lucca Schieber Vieira de Carvalho Pinheiro

gov.br Documento assinado digitalmente ANDRE LUIZ ALVES Data: 25/06/2026 13:53:39-0300 Verifique em https://validar.iti.gov.br

Assinatura do professor-orientador:

Nome completo do professor-orientador: André Luiz Alves