

Pedro Luís Mendonça Silva

**Sistema de Visão Computacional
Embarcado para Previsão de Trajetória
Plana e Interceptação de Projétil em
Tempo Real**

UBERLÂNDIA

2025

Pedro Luís Mendonça Silva

Sistema de Visão Computacional Embarcado para Previsão de Trajetória Plana e Interceptação de Projétil em Tempo Real

Trabalho de Conclusão de Curso da
Engenharia de Controle e Automação da
Universidade Federal de Uberlândia - UFU
- Câmpus Santa Mônica, como requisito
para a obtenção do título de Bacharel em
Engenharia de Controle e Automação

Universidade Federal de Uberlândia – UFU

Faculdade de Engenharia Elétrica

Orientadora: Profa. Dra. Gabriela Vieira Lima

UBERLÂNDIA

2025

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

S586 2025	Silva, Pedro Luis Mendonca, 2001- Sistema de Visão Computacional Embarcado para Previsão de Trajetória Plana e Interceptação de Projétil em Tempo Real [recurso eletrônico] / Pedro Luis Mendonca Silva. - 2025. Orientadora: Gabriela Vieira Lima. Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Uberlândia, Graduação em Engenharia de Controle e Automação. Modo de acesso: Internet. Inclui bibliografia. Inclui ilustrações. 1. Processos de fabricação - Automação. I. Lima, Gabriela Vieira, 1990-, (Orient.). II. Universidade Federal de Uberlândia. Graduação em Engenharia de Controle e Automação. III. Título. CDU: 67.02:681.3
--------------	---

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

Agradecimentos

A Deus, o Alfa e o Ômega, o princípio e o fim, o Todo-Poderoso, Aquele que é, que era e que há de vir. Por Ele, todas as coisas foram feitas. Em Cristo, como sinal de amor, entregou-se para nossa salvação. A Ele, toda honra e toda glória, por todos os séculos dos séculos.

Ao meu núcleo familiar, meu alicerce, pura manifestação do amor em minha vida. Em especial, aos meus pais, que ofereceram aos filhos o suporte que não tiveram, fazendo o possível e o impossível por nós.

Aos colegas e amigos que, direta ou indiretamente, tornaram a experiência da graduação mais leve, divertida e completa.

Aos trabalhadores da Universidade Federal de Uberlândia, que atuam nos bastidores com empenho diário para que os estudantes possam se dedicar plenamente ao processo de formação.

Aos professores, pelos ensinamentos, pelas correções e pelos diálogos que tanto contribuíram para meu amadurecimento pessoal e profissional.

À minha orientadora, por ter acreditado no meu projeto, pelo companheirismo, pela constante disponibilidade pela compreensão demonstrada ao longo da jornada.

“É justo que muito custe, o que muito vale.”

Santa Teresa D'Ávila

Resumo

A visão computacional tem se consolidado como uma tecnologia essencial no desenvolvimento de sistemas autônomos, com aplicações significativas em áreas como medicina, esportes, robótica e agricultura. Com o avanço do poder computacional e da eficiência energética em componentes eletrônicos, o processamento de dados em sistemas embarcados tem se mostrado útil para aumentar a autonomia e a descentralização das soluções baseadas em visão computacional. Dentre as funções que essa tecnologia pode desempenhar, destacam-se a identificação de objetos e a predição de suas trajetórias em tempo real. Nesse contexto, este trabalho propõe o desenvolvimento de um sistema físico composto por um interceptador inteligente, capaz de bloquear a trajetória de um projétil em um ambiente plano. Através da integração de algoritmos de visão computacional e de controle, o sistema será capaz de identificar e prever a trajetória de uma esfera, deslocando um atuador linear até o ponto de colisão calculado, a fim de impedir o avanço do projétil. O processamento dos dados visuais será realizado por uma placa BeagleBone Black, que operará sob o sistema operacional Linux. A aquisição de informações sensoriais e o envio de sinais de controle para um motor de corrente contínua serão responsabilidade de um ESP32. Assim, o projeto aborda aspectos multidisciplinares, envolvendo eletrônica, programação, acionamentos elétricos, controle e inteligência artificial, oferecendo uma solução compacta e eficiente para desafios de alta precisão e resposta em tempo real. Os testes iniciais foram realizados em um computador pessoal e validaram a abordagem dos algoritmos de visão e previsão de trajetória. Ao transferir essa responsabilidade para a BeagleBone Black, foi verificada uma queda de desempenho em função do menor poder de processamento, mas o sistema conseguiu interceptar o projétil de forma satisfatória. A adoção de técnicas simples e robustas, como cálculos geométricos, foi essencial para a acurácia das previsões realizadas. Os resultados atingidos foram compatíveis com os objetivos propostos.

Palavras-chave: Rastreamento de Objetos, Controle de Posição, Sistemas Embarcados.

Abstract

Computer vision has become an essential technology in the development of autonomous systems, with significant applications in fields such as medicine, sports, robotics and agriculture. Due to advancements in computational power and energy efficiency in electronic components, data processing in embedded systems has proven useful in increasing the autonomy and decentralization of computer vision-based solutions. Among the tasks that this technology can perform, object identification and real-time trajectory prediction stand out. In this context, this work proposes the development of a physical system composed of an intelligent interceptor capable of blocking the trajectory of a projectile in a planar environment. Through the integration of computer vision and control algorithms, the system will be able to identify and predict the trajectory of a sphere, moving a linear actuator to the calculated collision point to prevent the projectile from advancing. The processing of visual data will be performed by a BeagleBone Black board, which will operate under the Linux operating system. The acquisition of sensory information and the sending of control signals to a direct current motor will be the responsibility of an ESP32. Thus, the project addresses multidisciplinary aspects, involving electronics, programming, electric drives, control, and artificial intelligence, providing a compact and efficient solution for high-precision, real-time response challenges. Initial tests were conducted on a personal computer and validated the proposed vision and trajectory prediction algorithms. When processing was transferred to the BeagleBone Black, a decrease in performance was observed due to its lower computational capacity; however, the system was still able to intercept the projectile in a satisfactory manner. The adoption of simple and robust techniques, such as geometric calculations, was essential to achieving accurate predictions. The results obtained were consistent with the proposed objectives.

Keywords: Object Tracking, Position Control, Embedded Systems.

Lista de Ilustrações

Figura 1 – Esboço estrutural do projeto.....	13
Figura 2 – BeagleBone Black	22
Figura 3 – Esquema de Pinos BeagleBone Black.	22
Figura 4 – Sistema de controle em malha aberta.	24
Figura 5 – Sistema de controle em malha fechada.	24
Figura 6 – Sistema de controle digital com microcontrolador.	25
Figura 7 – Circuito equivalente simplificado de um motor CC.	27
Figura 8 – Módulo Ponte H.....	28
Figura 9 – Diagrama de blocos do controlador PID.....	29
Figura 10 – Desenho técnico do campo de colisão.	31
Figura 11 – Regiões do campo de colisão.....	31
Figura 12 – Campo de colisão.....	32
Figura 13 – Câmera instalada no suporte suspenso.	33
Figura 14 – Sistema do interceptador.....	33
Figura 15 – Foto do campo pela câmera suspensa.....	34
Figura 16 – Foto binária do campo a partir de segmentação HSV.	35
Figura 17 – Frame após slicing.	36
Figura 18 – Projétil esférico encontrado.	37
Figura 19 – Pontos de interesse.....	38
Figura 20 – Versão Debian pré atualização.	39
Figura 21 – Imagens Linux BBB.	40
Figura 22 – Flash da imagem no cartão com balenaEtcher.	41
Figura 23 – Login após atualização do sistema operacional.	41
Figura 24 – Nova versão Debian BBB.....	41
Figura 25 – Pulsos encoder de quadratura.....	43
Figura 26 – Arquitetura do projeto.	44
Figura 27 – Jumper de ativação do regulador do módulo L298N.	46
Figura 28 – Diagrama esquemático do projeto.....	48
Figura 29 – Eixos do sistema ordenado dos pixels.	49
Figura 30 – Trajetórias estimadas a partir do algoritmo desenvolvido.....	53
Figura 31 – Fluxograma da rotina de tracking.	54
Figura 32 – Diagrama para testes iniciais.....	57
Figura 33 – Bolas alternativas.	58

Lista de Tabelas

Tabela 1 – Aplicações de rastreamento de objetos.	18
Tabela 2 – Valores dos parâmetros das cores por modelo.	19
Tabela 3 – Equações fundamentais controlador PID	28
Tabela 4 – Parâmetros HSV para segmentação.....	35
Tabela 5 – Funções das barreiras.	38
Tabela 6 – Pinos para comunicação UART.....	45
Tabela 7 – Ganhos do PID.....	48

Lista de Abreviaturas e Siglas

<i>BBB</i>	<i>BeagleBone Black</i>
<i>CC</i>	<i>Corrente Contínua</i>
<i>CLP</i>	<i>Controlador Lógico Programável</i>
<i>DL</i>	<i>Deep Learning</i>
<i>FIFO</i>	<i>First In, First Out</i>
<i>GPIO</i>	<i>General Purpose Input/Output</i>
<i>GPU</i>	<i>Graphics Processing Unit</i>
<i>IoT</i>	<i>Internet of Things</i>
<i>MDF</i>	<i>Medium-density Fiberboard</i>
<i>PID</i>	<i>Controlador proporcional integral derivativo</i>
<i>PWM</i>	<i>Pulse Width Modulation</i>
<i>RTOS</i>	<i>Real Time Operating System</i>
<i>SBC</i>	<i>Single Board Computer</i>
<i>SO</i>	<i>Sistema Operacional</i>
<i>SP</i>	<i>Setpoint</i>
<i>SSH</i>	<i>Secure Shel</i>
<i>UART</i>	<i>Universal Asynchronous Receiver-Transmitter</i>
<i>USB</i>	<i>Universal Serial Bus</i>
<i>YOLO</i>	<i>You Only Look Once</i>

Sumário

1. INTRODUÇÃO	11
1.1. JUSTIFICATIVA	13
1.2. OBJETIVO.....	14
1.2.1. <i>Objetivos Específicos</i>	14
2. REFERENCIAL TEÓRICO	16
2.1. VISÃO COMPUTACIONAL	16
2.1.1. <i>Segmentação por Cores</i>	18
2.1.2. <i>OpenCV-Python</i>	19
2.2. LINUX EMBARCADO	20
2.2.1. <i>BeagleBone Black</i>	21
2.3. SISTEMAS DE CONTROLE.....	23
2.3.1. <i>Malha Aberta</i>	23
2.3.2. <i>Malha Fechada</i>	24
2.3.3. <i>Controle Digital e Microcontrolador ESP32</i>	24
2.4. MOTORES DE CORRENTE CONTÍNUA	26
2.4.1. <i>Ponte H</i>	27
2.4.2. <i>Algoritmos de Controle</i>	28
3. METODOLOGIA.....	30
3.1. CONSTRUÇÃO MECÂNICA	30
3.2. SISTEMA DE VISÃO COMPUTACIONAL.....	34
3.3. BEAGLEBONE BLACK <i>DEPLOYMENT</i>	39
3.4. INTERCEPTADOR.....	43
3.5. PROJETO ELETROELETRÔNICO	48
4. RESULTADOS E DISCUSSÕES	49
4.1. MODELAGEM DA COLISÃO	49
4.1.1. <i>Transformação de domínio e setpoint final</i>	55
4.2. DESEMPENHO DO SISTEMA	56
5. CONCLUSÃO.....	60
REFERÊNCIAS BIBLIOGRÁFICAS	62

1. INTRODUÇÃO

Sendo uma área relativamente recente em comparação com outras disciplinas das engenharias, haja vista as suas raízes históricas que apontam aproximadamente para a década de 70, a visão computacional é a ciência que busca fornecer às máquinas a capacidade de extrair os conteúdos visuais e interpretá-los em sequência (de Milano; Honorato, 2010). Integrando áreas como inteligência artificial e processamento de imagens, as utilizações práticas dessa tecnologia evidenciam o seu caráter multidisciplinar. Setores como agricultura, medicina e astronomia se beneficiam amplamente de sua capacidade de identificar padrões, analisar imagens em tempo real e auxiliar no controle inteligente e adaptativo de diversos processos (Neves; Vieira Neto; Gonzaga, 2012).

No ambiente industrial, por sua vez, a visão computacional também tem aplicabilidade prática validada. Ainda na década de 80, já existiam estudos que abordavam sobre a perspectiva das evoluções em processos de manufatura devido a essa tecnologia, principalmente nos campos de inspeção e de robótica (Rossol, 1983). Seguindo tal panorama, dentre as aplicações industriais que contam com a contribuição de sistemas de visão computacional, podem ser citadas a inspeção automatizada para controle de qualidade, o controle de deslocamento e orientação de robôs e a automatização de processos (Rudek; Coelho; Canciglieri Jr, 2001). Mostrando versatilidade, com o advento e consolidação da quarta revolução industrial, os sistemas de visão de máquina têm sido adotados com crescente frequência em muitas fases e em variados setores de processos industriais, como em sistemas de medição e instrumentação. Esse aumento de popularidade tem relação direta com a evolução dos sistemas computacionais e eletrônicos (Fonseca, 2017).

Seguindo a mesma tendência, com os avanços significativos em poder de processamento computacional, o aumento da eficiência energética de dispositivos microeletrônicos e da concomitante redução de tamanho dos componentes, tecnologias de visão computacional começaram a ser colocadas em sistemas embarcados. Principalmente com a atual descentralização evidenciada pela *Edge Computing* e pela *Internet of Things* (IoT), realizar o processamento dos dados de maneira local é essencial para habilitar inteligência e autonomia em sistemas de visão computacional, garantindo confiabilidade, reduzindo latência, aumentando a

privacidade dos dados e otimizando o uso da largura de banda (Nadeski, 2019). Para tanto, atualmente é possível encontrá-los em robôs móveis, em aplicações de segurança, em automóveis, em drones, em inspeção de qualidade, em agricultura de precisão e em diversas outras aplicações, sejam elas específicas para ambientes industriais ou não.

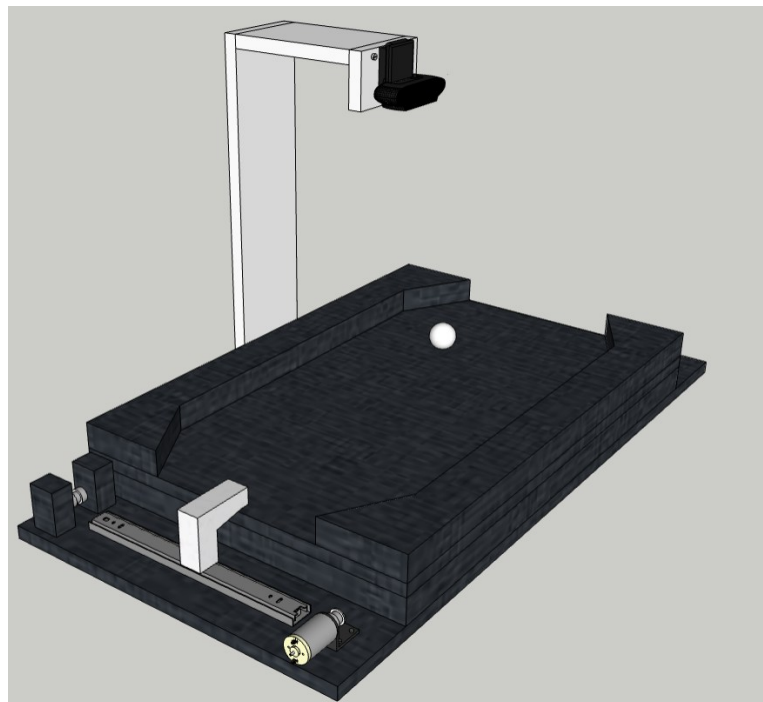
Mais do que apenas bem estabelecida, pode-se dizer que a visão computacional embarcada está em ascensão, sendo que hardwares compatíveis com sistemas operacionais de tempo real (RTOS – *Real Time Operating System*) e que contam com unidades dedicadas ao processamento gráfico (GPU – *Graphics Processing Unit*) têm ficado mais acessíveis, permitindo melhorias de desempenho e de confiabilidade (Curvello, 2016). Além dos processadores de alto desempenho, as evoluções nas técnicas de processamento de imagem e a integração de técnicas avançadas de aprendizagem de máquina, como *Deep Learning* (DL), nos dispositivos microeletrônicos também impulsionam a incorporação de visão computacional em embarcados (Udendhran *et al.*, 2020). Portanto, por mais que existam desafios a serem superados, evidencia-se que essa área da engenharia está consolidada em variados setores e pesquisadores têm buscado alcançar incrementos de velocidade e de robustez nesses sistemas (Chukwu, 2024).

Dentre as diversas tarefas possíveis de serem realizadas por meio do uso da visão computacional, destaca-se a identificação de objetos das mais variadas naturezas, aplicação a qual tem tido avanços significativos nos últimos anos. Grande parte dos desenvolvimentos alcançados nessa tecnologia se devem à crescente necessidade de sistemas inteligentes em robótica industrial, em medicina, na indústria automotiva, em segurança e em vigilância. Com a integração de DL em sistemas de visão computacional e o advento de modelos avançados como YOLO (*You Only Look Once*), progressos expressivos ocorreram na área (Rahmaniar; Hernawan, 2021). Em consequência, a partir da correta localização do objeto em uma sequência de imagens, é comum desejar identificar a trajetória percorrida por ele e, a partir do modelo de percurso encontrado, antecipar a sua posição futura (Simões; Costa; Garcia, 2002). Tal predição pode ser utilizada em paralelo com sistemas de controle, seja objetivando evitar colisões como na direção assistida e autônoma de veículos (Dixit; Chidambaram; Allam, 2021), ou almejando classificar frutas de maneira automática (Blasco *et al.*, 2009).

Tendo isso em vista, a proposta desse trabalho é construir um sistema físico

com um interceptador inteligente de um grau de liberdade que seja capaz de realizar a obstrução de um projétil que se desloca no mesmo plano que o seu. Isso será feito a partir da aplicação de algoritmos de visão computacional capazes de identificar a posição e prever a trajetória de uma esfera. Por sua vez, um mecanismo associado a um atuador linear e controlado por um motor de corrente contínua será deslocado para o ponto de encontro calculado previamente com o objetivo de não permitir a passagem do projétil do seu eixo, rebatendo-o. A concepção inicial do que será construído pode ser observado na Figura 1. Todo o processamento visual será responsabilidade de um embarcado com as aplicações necessárias funcionando em um sistema operacional (SO) Linux. A integração de dados sensoriais e o envio de sinais de controle serão realizados por meio de um ESP32.

Figura 1 – Esboço estrutural do projeto.



Fonte: Autor (2025).

1.1. Justificativa

A robótica autônoma tem desempenhado um papel fundamental no desenvolvimento de sistemas capazes de operar em cenários desafiadores, especialmente em aplicações de tempo real que demandam alta precisão e velocidade de resposta. Nesse contexto, a análise de trajetórias em tempo real se

apresenta como um tema de grande relevância, com empresas que fornecem soluções para aplicações que vão desde sistemas de defesa e segurança até automação industrial e esportiva (Lowe, 2015). A capacidade de prever e reagir a trajetórias com precisão é um desafio que requer o domínio de técnicas de processamento de imagens, algoritmos de controle dinâmico e cálculos preditivos.

Esse trabalho se destaca por sua forte multidisciplinaridade, integrando conhecimentos em eletrônica, programação, acionamentos elétricos, controle e inteligência artificial. Tal abordagem não apenas reflete a essência do curso de Engenharia de Controle e Automação, mas também proporciona uma base robusta para o desenvolvimento futuro de soluções aplicáveis a diferentes contextos. Além disso, a proposta supera as limitações de ambientes de simulação ao visar à construção de uma implementação física, conferindo maior tangibilidade e relevância ao projeto.

Por fim, o uso de sistemas embarcados reforça ainda mais a importância do estudo, pois demonstrará a viabilidade de soluções compactas e eficientes para desafios complexos. Sendo assim, espera-se que os resultados contribuam tanto para o avanço acadêmico quanto para aplicações práticas em cenários onde a precisão e a agilidade são fatores importantes.

1.2. Objetivo

O objetivo geral deste trabalho é projetar e construir um sistema capaz de prever a trajetória plana de uma esfera em um campo de colisão e interceptá-la no ponto calculado, por meio da integração algoritmos de visão computacional e de controle de posição em sistemas embarcados.

1.2.1. Objetivos Específicos

Inseridas dentro do objetivo principal, diversas outras metas buscam ser atingidas durante a execução do projeto. Dentre elas, podem ser citadas:

- desenvolvimento do projeto mecânico tridimensional, com o tamanho e a disposição das peças do sistema devidamente especificados;

- elaboração de documentação das partes elétrica, como o diagrama esquemático, e eletrônica, como a relação de componentes de hardware;
- especificação de um ou mais algoritmos de visão computacional que serão usados para identificar a posição, a velocidade e a trajetória no plano cartesiano da esfera e implementação em linguagem Python;
- construção física do campo de colisão, respeitando os projetos mecânico e elétrico do sistema;
- desenvolvimento da estratégia de controle a ser implementada no interceptador e subsequente sintonização;
- validação dos algoritmos de detecção de trajetória e de controle, executados de maneira paralela, em sistemas embarcados.

2. REFERENCIAL TEÓRICO

Este capítulo apresenta o embasamento científico necessário para sustentar o desenvolvimento do trabalho, explorando as teorias e os aspectos técnicos que fundamentam suas temáticas principais. Tendo em vista a natureza multidisciplinar do projeto, serão abordados tópicos como visão computacional, controle de sistemas, motores de corrente contínua e sistemas embarcados. Portanto, a intenção é oferecer uma base sólida que permita compreender de forma clara e objetiva os conceitos associados ao funcionamento do sistema de interceptação de projéteis proposto, integrando as áreas envolvidas e reforçando a relevância técnica do estudo.

2.1. Visão Computacional

A visão computacional é a ciência que busca desenvolver métodos e algoritmos capazes de extrair informações significativas de uma ou mais imagens, ou até de uma sequência delas. Dado esse cenário, pela própria natureza do problema, algumas dificuldades foram surgindo durante o desenvolvimento da tecnologia, visto que os dados visuais são muitas vezes insuficientes para determinar uma solução única, como definir características de profundidade a partir da análise de um elemento bidimensional. Então, para mitigar essas limitações, comumente são empregados de forma conjunta diferentes modelos baseados em física, estatística e aprendizado de máquina, com o objetivo de reconstruir com precisão algumas propriedades de interesse, como formas e cores de objetos (Szeliski, 2010).

No contexto de fornecer aos computadores a capacidade de captar informações a partir de imagens, o processamento de sinais e a identificação de padrões surgiram como pilares para a consolidação, avanço e sucesso de uma ciência que já apresentava progressos significativos no final do século passado. Devido à grande quantidade de dados gerados por dispositivos de captura visual, é necessário que seja levado em consideração o tempo de execução e a complexidade computacional dos algoritmos, visto que eles devem conseguir tratar e extrair características de sinais muitas das vezes multidimensionais. Sendo assim, como forma de avaliar a performance de um sistema de visão computacional, além da velocidade, podem ser analisadas a precisão, em função de erros aleatórios e

sistemáticos, e o sucesso em resolver o problema de interesse (Jähne; Haussecker; Geissler, 1999).

Com a associação de algoritmos de *Machine Learning* aos sistemas de visão computacional e com o contínuo aprimoramento dessas tecnologias, muitos desafios relacionados ao processamento e à extração de características foram superados, auxiliando na resolução de problemas complexos. Nesse sentido, áreas como medicina, engenharia, agricultura, esportes e educação têm se beneficiado de aplicações inovadoras baseadas nessa combinação (Khan; Al-habsi, 2020). Tal cenário se acentuou com os avanços em *Deep Learning*, um conjunto de métodos de aprendizagem de máquina, como redes neurais, e de modelos probabilísticos, dado que os computadores passaram a processar os dados com certo grau de abstração, inspirando-se no funcionamento do cérebro humano. Como resultado, o aprendizado profundo, impulsionado pelo acesso facilitado a grandes bases de dados e pelo processamento paralelo em GPU's, tem alcançado um desempenho superior em uma ampla gama de tarefas quando comparado a técnicas anteriores, gerando progressos expressivos em diversas aplicações (Voulodimos *et al.*, 2018).

Dentre os principais objetivos da visão computacional, destaca-se a obtenção de informações sobre a posição e o formato de objetos específicos, uma vez que o seu conhecimento serve como base sólida para o desenvolvimento de raciocínios e a execução de ações, aspectos os quais são essenciais para sistemas inteligentes (Vidmar; Solina, 1992). De maneira semelhante, o rastreamento desses objetos de interesse — isto é, a estimação de suas trajetórias a partir de uma sequência de imagens — é uma das funções mais relevantes da área. Isso porque, motivada pelos avanços tecnológicos nos computadores e pela maior acessibilidade a câmeras de alta qualidade, a demanda por análises automatizadas de vídeo tem crescido. Contudo, essa tarefa apresenta desafios significativos, como padrões de movimento complexos, presença de ruídos nos quadros analisados, variações na iluminação da cena e a projeção tridimensional em um espaço bidimensional (Yilmaz; Javed; Shah, 2006).

No contexto do rastreamento por visão computacional, o termo 'objeto' diz respeito a qualquer elemento presente em uma imagem que seja relevante para análise, podendo ser uma pessoa, um animal ou um item. Os objetos em uma imagem podem ser representados como pontos, formas geométricas ou contornos. Para identificá-los, propriedades visuais como cor, borda e textura se mostram válidas.

Tendo isso em vista, existem diversas técnicas que se diferem na forma de abordar o problema, sendo a seleção da mais adequada dependente do contexto e do cenário observado (Yilmaz; Javed; Shah, 2006). Sendo assim, a partir da escolha da metodologia adequada, diversas áreas podem se beneficiar de aplicações dessa tecnologia, como pode ser observado na Tabela 1.

Tabela 1 – Aplicações de rastreamento de objetos.

Aplicação	Descrição
Vigilância	Deteção de atividades suspeitas
Interação Homem-máquina	Reconhecimento de gestos e rastreamento de olhar
Trânsito	Direcionamento de tráfego em tempo real
Carros autônomos	Evasão de obstáculos e planejamento de trajetória

Fonte: Yilmaz; Javed; Shah (2006) – adaptado.

2.1.1. Segmentação por Cores

A segmentação de imagens é uma parte fundamental da visão computacional. É a primeira etapa do processamento e, portanto, interfere diretamente nos resultados atingidos. O seu objetivo é dividir uma imagem agrupando pontos que compartilham características em comum. Ou seja, de maneira formal, essa fase pode ser definida como o processo de dividi-las em regiões uniformes que se distinguem das adjacentes. Isto é, dados um conjunto de regiões $\{S_1, S_2, S_3, \dots, S_n\}$ que formam uma imagem I em sua totalidade, um operador de homogeneidade H e um identificador de vizinhança V , tem-se que $\bigcup_{i=1}^n S_i = I$, $S_i \cap S_j = \emptyset$, $H(S_i) = \text{true} \forall i$, $V(S_i, S_j) = \text{true} \Rightarrow H(S_i \cup S_j) = \text{false}$ desde que $i \neq j$ (Cheng *et al.*, 2001).

Para realizar a segmentação, existem duas abordagens bem conhecidas que examinam propriedades como cor, intensidade e textura dentro de uma imagem. A primeira delas procura por contornos, visando à identificação de descontinuidades ou de transições entre os locais analisados. A outra, por sua vez, busca gerar regiões bem definidas por meio de agrupamentos de pixels que apresentam similaridades em algum tipo de atributo. A busca por contornos dificilmente consegue estabelecer fronteiras completamente fechadas, o que não resulta em uma segmentação de imagem direta e gera a necessidade de tratamentos posteriores. Em contrapartida, a abordagem por região leva a uma partição objetiva, em que cada ponto fica

devidamente isolado em sua respectiva área (Elbalaoui *et al.*, 2013).

Dentre as formas de se abordar o problema de segmentação, a análise por cores ganhou destaque visto que a coloração de uma imagem pode fornecer informações importantes sobre objetos de interesse que se encontram nela. Nesse sentido, tecnologias que contam com esse tipo de metodologia são utilizadas em aplicações de medicina, da indústria alimentícia e de outras áreas. Antes de entrar no processo desse tipo de segmentação, para que as características de coloração sejam extraídas adequadamente, deve-se selecionar uma representação matemática que consiga especificar parâmetros como cromaticidade e intensidade de cada cor. Para tanto, diferentes espaços de cores podem ser escolhidos. Na Tabela 2, para exemplificar, os parâmetros referentes às cores preto, branco, azul, vermelho, verde, amarelo e cinza são exibidos considerando diferentes padrões (Garcia-Lamont *et al.*, 2018).

Tabela 2 – Valores dos parâmetros das cores por modelo.

	RGB			HSV			L*a*b		
	R	G	B	H	S	V	L*	a*	b*
Amarelo	1	1	0	$\pi/3$	1	255	100	0	128
Azul	0	0	1	$4\pi/3$	1	255	100	-127	-127
Branco	1	1	1	-	0	255	100	0	0
Cinza	1/2	1/2	1/2	-	0	127	50	0	0
Preto	0	0	0	-	0	0	0	0	0
Verde	0	1	0	$2\pi/3$	1	255	100	-127	128
Vermelho	1	0	0	0	1	255	100	128	128

Fonte: Garcia-Lamont *et al.* (2018) – adaptado.

2.1.2. OpenCV-Python

Desenvolvida em meados do final da década de 80 e do começo dos anos 90 por Guido Van Rossun no Instituto Nacional de Pesquisa em Matemática e Ciência da Computação dos Países Baixos, Python é uma linguagem de programação interpretada de alto nível, multiplataforma, extensível, multiparadigma e de código aberto. Sendo altamente reconhecida por sua confiabilidade e pela sintaxe simplificada, a linguagem é amplamente utilizada em áreas como aprendizado de máquina, big data, desenvolvimento web e visão computacional, possuindo uma

comunidade ativa de desenvolvedores. Além disso, sua popularidade também se deve à extensa biblioteca padrão, que oferece, dentre outras, ferramentas para processamento de texto, manipulação de dados e operações matemáticas. Como consequência, programadores novatos e experientes exploram o seu potencial em aplicações acadêmicas, comerciais e científicas (Saabith; Fareez; Vinothraj, 2019).

Além da sua biblioteca padrão, Python se destaca pela ampla gama de bibliotecas *open source* que a comunidade disponibiliza e dá suporte, sendo úteis para diversos propósitos. Dentre elas, a OpenCV ganha prestígio no âmbito da visão computacional, visto que o seu código aberto foi desenvolvido para acelerar o desenvolvimento desse tipo de aplicação, oferecendo mais de dois mil algoritmos para tarefas como reconhecimento de objetos, rastreamento de movimento e realidade aumentada, os quais são expandidos continuamente. Originalmente desenvolvida em C++, para ser utilizada em Python, essa biblioteca utiliza de *wrappers* (encapsuladores) gerados de forma automática a partir de *headers* C++. Como resultado, as funções são executadas com a velocidade e eficiência da linguagem compilada, enquanto o processo de programação se beneficia da sintaxe simplificada do Python. Além disso, por estruturar os arrays e basear as suas operações numéricas na biblioteca Numpy, as integrações com outras bibliotecas como Scipy e Matplotlib são facilitadas (OpenCV, 2025).

2.2. Linux Embarcado

Anteriormente, era comum realizar o projeto integral do hardware de um dispositivo eletrônico apenas visando à execução de uma tarefa específica. No entanto, caso fosse necessário atualizar ou modificar o aparelho, poderia ser preciso reestruturar grande parte do projeto inicial, consumindo tempo e recursos consideráveis. Como solução para esse desafio, o uso de componentes genéricos de *hardware* multipropósito com uma camada de *software* implementada tornou-se uma alternativa viável para executar as mesmas funções específicas. Dessa forma, os sistemas embarcados surgiram a partir desse conceito, associando microcontroladores, ou até mesmo processadores, a uma aplicação especializada a nível de *software*. Dependendo da complexidade das atividades realizadas pelo sistema, a programação pode ser feita por meio de laços e rotinas simples ou exigir a utilização de sistemas operacionais (Zorzo *et al.*, 2002).

Nesse cenário, o uso do Linux como sistema operacional em embarcados se mostra possível, dado que o avanço tecnológico aumenta a qualidade dos processadores e das memórias torna os custos mais acessíveis. Ainda no início dos anos 2000, já era possível encontrar, a preços competitivos, microprocessadores de 32 bits com vários megabytes de memória. Tal contexto viabiliza a adoção de sistemas operacionais avançados como o Linux, um núcleo que oferece confiabilidade, estabilidade e eficiência, ao mesmo tempo em que conta com uma comunidade ativa e extensa de usuários. Sendo compatível com diversas arquiteturas, como x86 e ARM, a sua natureza de código aberto permite constantes aprimoramentos em desempenho, funcionalidades e suporte a drivers (Hollabaugh, 2002). Além disso, tal panorama surge como uma oportunidade de personalização, de acordo com as necessidades específicas de cada aplicação, e de ganho econômico, visto que não é necessário lidar com custos de licenciamento do produto (Lombardo, 2001).

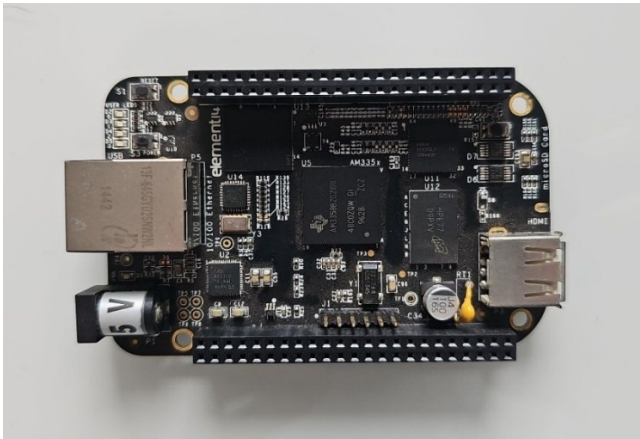
2.2.1. BeagleBone Black

A BeagleBone Black (BBB), apresentada na Figura 2, é uma *Single-board Computer* (SBC) acessível de alto poder de processamento, cujo projeto de *hardware* é *open source* e, portanto, pode ser adaptada de acordo com as necessidades específicas do projetista. Com ela, é possível integrar circuitos eletrônicos do mais baixo nível com aplicações avançadas que rodam em Linux embarcado, como um servidor Web e protocolos de acesso remoto como SSH (*Secure Shell*). A partir dessa junção, o desenvolvimento de soluções de Internet das Coisas e robótica é facilitado. Além disso, o sistema operacional já providencia suporte para interfaces USB (*Universal Serial Bus*), o que permite que a BBB possa trabalhar conectada a periféricos, como câmeras e adaptadores wifi (Molloy, 2014).

A placa possui um ARM Cortex A8 1GHz como processador, especificamente o AM3358BZCZ100, uma memória RAM de 512 MB, que oferece uma largura de banda de 1,6 GB por segundo, uma memória flash de 4 GB e um slot para cartão microSD. Além da porta USB *host* com carga máxima de 500 mA, a BBB possui uma porta miniUSB, uma interface RJ45 para conexão Ethernet 10/100, uma saída HDMI integrada e 65 pinos GPIO (*General Purpose Input/Output*) configuráveis, os quais são encontrados em dois conectores, P8 e P9, conforme apresentado na Figura 3, que contam com 46 pinos cada. O *boot* por padrão é feito por meio da memória eMMC de

4 GB, mas a placa também tem suporte à inicialização pelo cartão, pela porta USB ou por porta serial (BeagleBoard, 2024).

Figura 2 – BeagleBone Black



Fonte: Autor (2025).

Figura 3 – Esquema de Pinos BeagleBone Black.

P9				P8			
DGND	1	2	DGND	DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3	GPIO_38	3	4	GPIO_39
VDD_5V	5	6	VDD_5V	GPIO_34	5	6	GPIO_35
SYS_5V	7	8	SYS_5V	TIMER4	7	8	TIMER7
PWR_BTN	9	10	SYS_RESETN	TIMER5	9	10	TIMER6
GPIO_30	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
GPIO_31	13	14	EHRPWM1A	EHRPWM2B	13	14	GPIO_26
GPIO_48	15	16	EHRPWM1B	GPIO_47	15	16	GPIO_46
GPIO_5	17	18	GPIO_4	GPIO_27	17	18	GPIO_65
I2C3_SCL	19	20	I2C3_SDA	EHRPWM2A	19	20	GPIO_63
EHRPWMOB	21	22	EHRPWMOA	GPIO_62	21	22	GPIO_37
GPIO_49	23	24	GPIO_15	GPIO_36	23	24	GPIO_33
GPIO_117	25	26	GPIO_14	GPIO_32	25	26	GPIO_61
GPIO_115	27	28	ECAPPWM2	GPIO_86	27	28	GPIO_88
EHRPWMOB	29	30	GPIO_112	GPIO_87	29	30	GPIO_89
EHRPWMOA	31	32	VDD_ADC	GPIO_10	31	32	GPIO_11
AIN4	33	34	GNDA_ADC	GPIO_9	33	34	EHRPWM1B
AIN6	35	36	AIN5	GPIO_8	35	36	EHRPWM1A
AIN2	37	38	AIN3	GPIO_78	37	38	GPIO_79
AIN0	39	40	AIN1	GPIO_76	39	40	GPIO_77
GPIO_20	41	42	ECAPPWMO	GPIO_74	41	42	GPIO_75
DGND	43	44	DGND	GPIO_72	43	44	GPIO_73
DGND	45	46	DGND	EHRPWM2A	45	46	EHRPWM2B

Fonte: BeagleBoard (2024).

Diversos aspectos fazem da BeagleBone Black um ótimo SBC para desenvolvimento de sistemas embarcados. Por meio do uso do seu ambiente de desenvolvimento, a configuração de sensores e o controle de atuadores são feitos sem maiores complicações. Além disso, é possível integrar periféricos complexos por meio de vários tipos de rede, a partir do uso de programação C/C++/Python/JavaScript

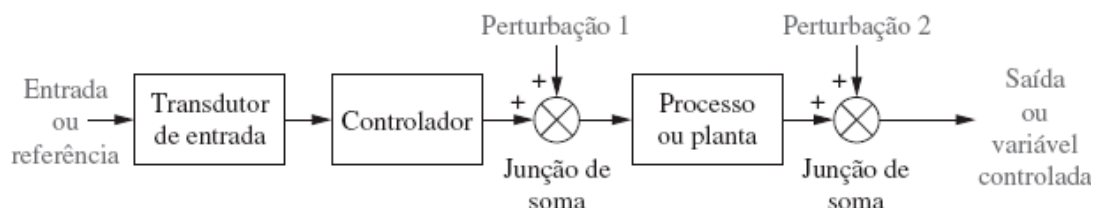
e suas bibliotecas. Dessa maneira, a BBB mostra grande capacidade para ser usada em projetos simples, como os de microcontroladores 8 bits, e em mais desenvolvidos, como os que são executados em cima de sistemas operacionais como o Linux. Tendo essas características em vista, a placa demonstra qualificação para aplicações de robótica e de *Internet of Things* (He; Qian; Huang, 2016).

2.3. Sistemas de Controle

Diversas aplicações do cotidiano fazem uso de sistemas de controle, que são, para Nise (2023), “subsistemas e processos (ou plantas) construídos com o objetivo de obter uma saída desejada com um desempenho desejado, dada uma entrada especificada”. As possibilidades de fornecer ganhos de potência ao processo, de controlar os sistemas de maneira remota, de atribuir convenientemente o formato da variável de entrada e de compensar distúrbios que desestabilizam a saída fazem com que os sistemas de controle sejam amplamente projetados. Sendo assim, atualmente esses mecanismos reguladores estão associados a computadores digitais que são encontrados em manipuladores robóticos, em indústrias de processo e em tecnologias de veículos terrestres, aquáticos, aéreos e espaciais (Nise, 2023).

2.3.1. Malha Aberta

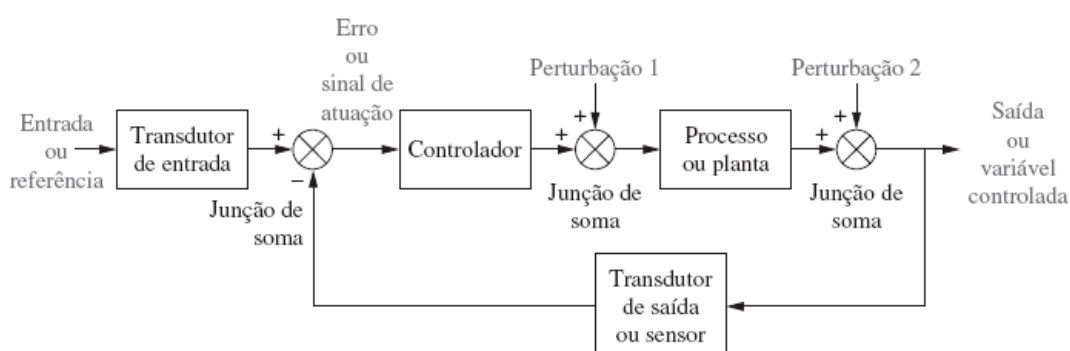
Os sistemas de controle em malha aberta são aqueles que não possuem uma realimentação do sinal de saída. Sendo assim, ele não consegue compensar distúrbios que podem ocorrer tanto no sinal que sai do controlador quanto na própria saída do processo, pois ele não é capaz de mensurar o valor real da variável controlada. Isto é, o sinal de regulação enviado pelo controlador é baseado unicamente pelo sinal de entrada que determina a referência a ser seguida, impossibilitando correções que podem ser necessárias em função de possíveis perturbações que influenciem no processo. Ou seja, nesse tipo de metodologia, cuja representação em diagrama de blocos é apresentada na Figura 4, não se sabe se a atuação realizada está sendo precisa, insuficiente ou exagerada, podendo levar a variável controlada a valores indesejados em função da sensibilidade às perturbações (Nise, 2023).

Figura 4 – Sistema de controle em malha aberta.

Fonte: Nise (2023).

2.3.2. Malha Fechada

Sistemas em malha fechada (Figura 5), também chamados de sistemas de controle com realimentação, dispõem de artifícios, como sensores, que permitem que a resposta obtida na saída seja medida e analisada. A partir dessa informação, portanto, a variável controlada que está sendo observada no processo é comparada com o valor de referência da entrada, a partir de transdutores, possibilitando que o sinal de atuação seja regulado de forma a corrigir quaisquer distúrbios ou erros existentes na planta. Como resultado, o uso dessa metodologia gera uma maior exatidão. Além disso, sistemas retroalimentados são mais flexíveis, se adaptando em função dos ruídos, das mudanças de comportamento do ambiente e das perturbações que afetam o processo e o sinal de controle (Nise, 2023).

Figura 5 – Sistema de controle em malha fechada.

Fonte: Nise (2023).

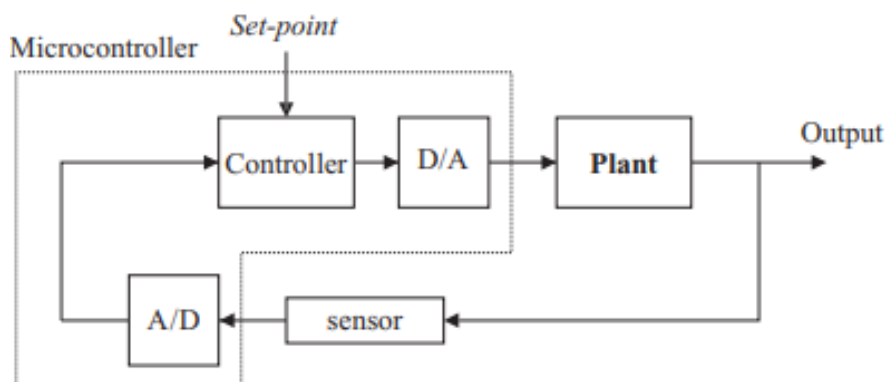
2.3.3. Controle Digital e Microcontrolador ESP32

A maior parte das aplicações de controle atualmente é realizada por meio de dispositivos computadorizados, sendo que equipamentos eletrônicos como

microcontroladores e Controladores Lógicos Programáveis (CLPs) são comumente utilizados. Para interagir com os sinais da planta, que geralmente são analógicos, é necessário o uso de conversores A/D (analógico para digital) e D/A (digital para analógico). Dessa forma, o sistema primeiramente amostra o sinal contínuo no tempo com o objetivo de transformá-lo em uma variável digital, pronta para processamento. A partir desse parâmetro, o controlador realiza os cálculos necessários para definir a ação de controle e envia um sinal de resposta para atuar na planta, o qual normalmente precisa ser convertido de volta para a forma analógica (Ibrahim, 2006).

Certos microcontroladores contam nativamente com conversores A/D e D/A internos, permitindo uma interação direta com os sinais analógicos do sistema sem o uso de dispositivos externos auxiliares e centralizando o processamento. Uma malha de controle digital dessa natureza é apresentada na Figura 6, sendo que o *setpoint* é definido no próprio sistema computadorizado.

Figura 6 – Sistema de controle digital com microcontrolador.



Fonte: Ibrahim (2006).

Nesse contexto, o desenvolvimento dos microcontroladores, que podem ser definidos como computadores miniaturizados de único *chip*, impulsionou o interesse em sistemas de controle digital (Ibrahim, 2006). Dentre os microcontroladores de diversas famílias, os quais possuem suas particularidades e vantagens, o ESP32 se mostra bastante versátil. Com arquitetura baseada no microprocessador Xtensa LX6, o dispositivo conta com uma variedade de características úteis para o desenvolvimento de sistemas de automação e robótica. Dentre tais propriedades, a presença de conversores A/D e D/A, de pinos capazes de gerar sinais modulados por largura de pulso (PWM - *Pulse Width Modulation*) e de variadas interfaces de

comunicação fazem os microcontroladores da família ESP32 indicados também para aplicações de controle, dado a sua capacidade de integrar sensores e atuadores, como motores (Samiullah; Irfan; Rafique, 2023).

2.4. Motores de Corrente Contínua

Em sistemas robóticos, os atuadores que se valem de fenômenos físicos eletromagnéticos, como motores, são os mais utilizados, devido, dentre outros motivos, à facilidade de encontrá-los, à facilidade de integração com sensores e à possibilidade de controle de movimento por meio de sinais elétricos (Romano; Dutra, 2002). Nesse sentido, existem uma variedade de motores, os quais possuem fundamentos muito semelhantes, visto que o seu torque é gerado pela interação entre campos magnéticos que podem ser gerados pela circulação de corrente em bobinas, por ímãs, eletroímãs e correntes induzidas (Hunter; Hollerbach; Ballantyne, 1991).

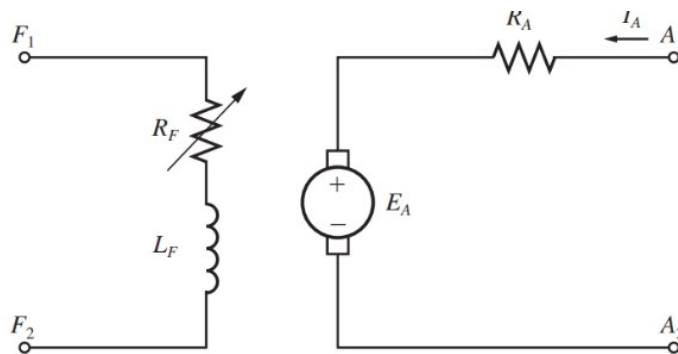
Por mais que os motores de passo também sejam aplicados em diversos sistemas mecatrônicos que necessitam de posicionamento de precisão, como robôs e outras aplicações industriais, os motores CC (corrente contínua) se mostram como alternativas viáveis, sendo amplamente utilizados nesses cenários (Bacac *et al.*, 2014). Esse tipo de atuador se destaca por, além de ser compacto, possuir um valor de torque estável mesmo que a velocidade seja alterada. Mesmo assim, com o objetivo de aproveitar a máxima eficiência mecânica dos motores CC, que ocorre em altas rotações, é comum associá-los a sistemas redutores, como conjuntos de engrenagens, para reduzir a sua velocidade ao mesmo tempo em que o seu momento de força é aumentado. Como ponto de atenção, para que um processo regulatório preciso de posição seja possível, esse tipo de dispositivo precisa ser associado a medidores, como encoders e tacômetros, para que sua malha de controle seja fechada (Romano; Dutra, 2002).

Em relação às características de funcionamento de um motor de corrente contínua, existem várias proporcionalidades que descrevem o seu desempenho. Uma das relações mais importantes é entre a tensão interna gerada na máquina e a frequência angular do rotor. Essa tensão induzida é diretamente proporcional à velocidade de rotação e ao fluxo magnético presente no motor, fazendo com que a diferença de potencial cresça conforme o aumento na velocidade de rotação. O conjugado induzido, por sua vez, é diretamente proporcional à corrente da armadura

e ao fluxo magnético, ou seja, para gerar mais torque, é necessário aumentar o valor de alguma dessas duas grandezas (Chapman, 2013).

Além disso, o circuito equivalente do motor de corrente contínua, apresentado na Figura 7, mostra que a tensão aplicada nos terminais F_1 e F_2 gera uma corrente no enrolamento de campo, representado pela associação em série de R_F e L_F , a qual estabelece o fluxo magnético da máquina. Do lado do rotor, a força contraeletromotriz E_A , induzida devido à rotação do motor, se opõe à tensão aplicada nos terminais A_1 e A_2 e, em conjunto com a resistência R_A , afeta a intensidade da corrente de armadura I_A . Isso, em concomitância com as relações anteriormente abordadas, reforça a ideia de que o controle de velocidade e torque do motor CC está diretamente ligado a variáveis como tensão e corrente, permitindo ajustes no desempenho conforme a necessidade da aplicação.

Figura 7 – Circuito equivalente simplificado de um motor CC.



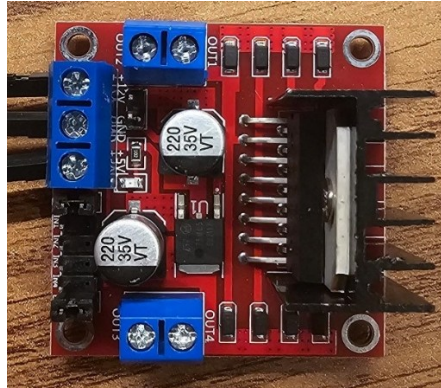
Fonte: Chapman (2013).

2.4.1. Ponte H

A ponte H, apresentada na Figura 8, é um dispositivo eletrônico essencial no controle de motores de corrente contínua, permitindo o ajuste dinâmico da direção, do torque e da velocidade da máquina. Estruturalmente, ela é composta por quatro chaves (geralmente transistores) dispostas de forma a possibilitar a comutação da corrente no motor, permitindo movimentos rotativos tanto no sentido horário quanto anti-horário. Além disso, ao incorporar sinais modulados por largura de pulso (PWM) em seu circuito, é possível regular a velocidade do motor, visto que a variação no ciclo de trabalho do sinal ajusta o valor médio da tensão fornecida, impactando diretamente na potência entregue. Dessa forma, as limitações impostas pelo uso de uma fonte de

tensão constante são superadas, possibilitando a manipulação direta das características do motor CC (Ranadev; Chakrasali, 2014).

Figura 8 – Módulo Ponte H.



Fonte: Autor (2025).

2.4.2. Algoritmos de Controle

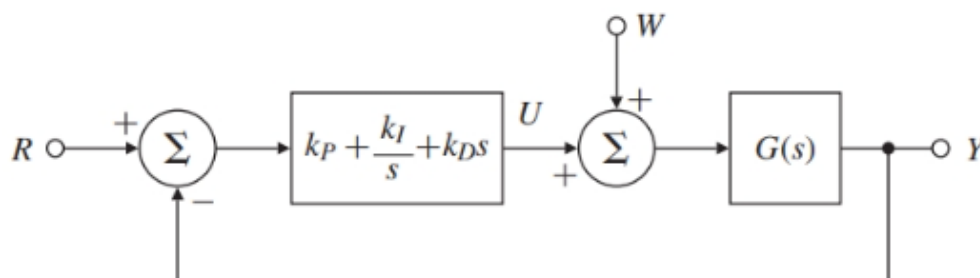
Dado que um motor CC está associado a um sensor capaz de providenciar um sinal de retroalimentação de posição, como um encoder, existem diferentes abordagens que podem ser utilizadas a fim de que o controle de posição do dispositivo seja efetivado. Dentre elas, o controlador proporcional integral derivativo (PID), um dos algoritmos mais comuns em sistemas de controle, cujo diagrama no domínio de Laplace é apresentado na Figura 9, se mostra uma boa alternativa. Nele, o sinal de controle enviado é proporcional ao erro, à integral do erro e à derivada do erro (Bacac *et al.*, 2014). As equações fundamentais desse algoritmo em tempo contínuo e discreto são apresentadas na Tabela 3, onde K_p é a constante de proporcionalidade, T_I é o tempo integrativo, T_D é o tempo derivativo e T_s é o tempo de amostragem.

Tabela 3 – Equações fundamentais controlador PID

	Ação de controle
Tempo contínuo	$u(t) = K_p \cdot \left(e(t) + \frac{1}{T_I} \cdot \int_0^t e(\tau) d\tau + T_D \cdot \frac{d}{dt} e(t) \right)$
Tempo discreto	$u[n] = K_p \cdot \left(e[n] + \frac{T_s}{T_I} \cdot \sum_{i=0}^n e[i] + \frac{T_D}{T_s} \cdot (e[n] - e[n-1]) \right)$

Fonte: Vatansever; Haciiskenderoglu (2022) – adaptado.

Figura 9 – Diagrama de blocos do controlador PID.



Fonte: Franklin; Powell; Emami-Naeini (2013).

Outra maneira de realizar o controle de posição de um motor de corrente contínua é a partir do controle cascata, em que existe uma malha responsável por controlar a velocidade angular do motor e outra encarregada de regular a posição. Tais malhas, que contam com ganhos iguais aos do PID, são associadas, sendo que uma define o setpoint da outra. Portanto, o método em cascata, em função da maior quantidade de parâmetros, exige uma calibração de maior complexidade, mesmo que o termo derivativo não seja utilizado (Bacac *et al.*, 2014).

Por fim, por mais que a descrição de sistemas por meio de variáveis de estado apresente vantagens mais relevantes para sistemas que trabalham com múltiplas entradas e múltiplas saídas, é possível realizar o controle de posição do motor CC, que possui uma entrada e uma saída, por espaço de estados. O seu ponto forte é a robustez em relação às perturbações dinâmicas que possam ocorrer (Bacac *et al.*, 2014).

3. METODOLOGIA

O desenvolvimento do trabalho foi estruturado em cinco grandes etapas. Na primeira, focou-se na construção mecânica do campo de colisão e dos mecanismos essenciais para seu funcionamento integral. O segundo estágio consistiu no desenvolvimento do código em Python do sistema de visão computacional responsável por rastrear o projétil esférico no campo de colisão. Na terceira parte foi realizada toda a configuração de hardware para que o código com a rotina de *tracking* pudesse ser executado em uma plataforma embarcada com sistema operacional Linux. Na quarta fase, o sistema do interceptador foi priorizado considerando os aspectos de sensoriamento, acionamento e controle. Por fim, na última etapa foi feito o design do projeto eletroeletrônico e, para tanto, foi levantada a relação de todos componentes necessários, os quais foram adquiridos e o sistema final foi montado. Sendo assim, nas próximas seções serão descritas com maiores detalhes todas as informações pertinentes a cada uma das fases citadas.

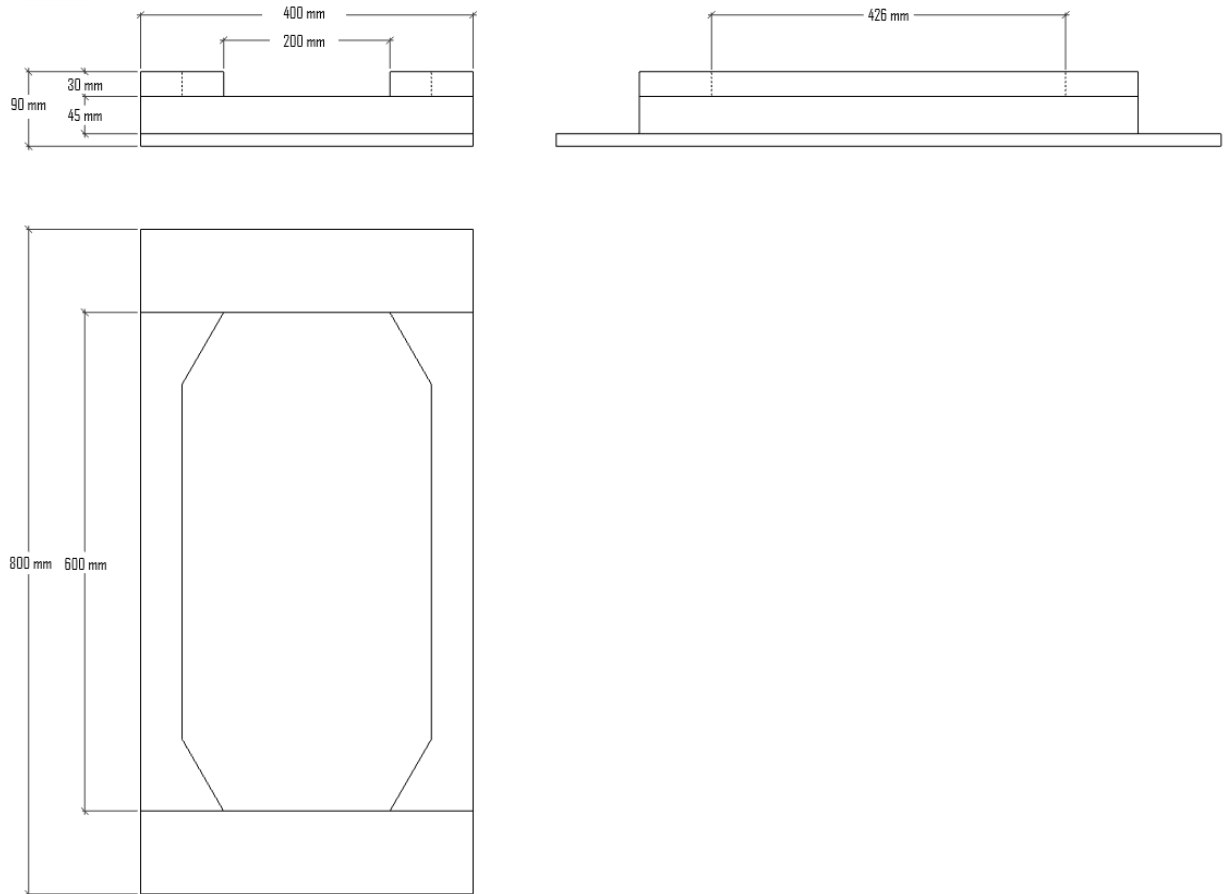
3.1. Construção Mecânica

Para que fosse possível ultrapassar o ambiente de simulação no projeto proposto, foi necessário realizar a construção de um campo de colisão que permitisse o deslocamento de um projétil esférico em um ambiente plano. O seu espaço deveria ser limitado nas direções que não fossem paralelas ao eixo de movimento do interceptador para que a bola pudesse rebater nas barreiras, gerando trajetórias variantes no tempo. Assim, o sistema de visão computacional desenvolvido poderia contar com algoritmos de previsibilidade, em que as possíveis colisões nas partes laterais seriam identificadas previamente, com o objetivo de calcular de maneira aproximada a posição em que a interceptação do projétil deveria ocorrer.

Desenvolveu-se um esboço do campo de colisão que se almejava alcançar, conforme foi apresentado na Figura 1. O projeto real seguiu a ideia inicial, sendo que algumas das dimensões do sistema foram definidas baseadas nas especificações de chapas de madeira MDF (*Medium-density Fiberboard*), material que foi escolhido para confecção. A Figura 10 mostra o desenho técnico do campo de colisão, em suas vistas frontal, superior e lateral. Vale dizer que o espaço útil de movimento da bolinha possui

aproximadamente 1700 cm^2 , correspondentes à região mais interna do campo que está indicada em verde na Figura 11.

Figura 10 – Desenho técnico do campo de colisão.



Fonte: Autor (2025).

Figura 11 – Regiões do campo de colisão.



Fonte: Autor (2025).

Além disso, um sistema de deslocamento linear foi desenvolvido, o qual deve ocupar uma das laterais em cinza da Figura 11. A amplitude útil de movimento do

interceptador, conforme projeto do campo de colisão, deve ser de 200 mm. Em função do menor preço quando comparado a guias lineares, foi decidido adaptar uma corrediça telescópica de 30 cm, usualmente utilizada na instalação de gavetas, para a função. Para ser a força motriz do sistema, um motor CC de ímã permanente com eixo de 4 mm, alimentação 24 VCC, 620 rotações por minuto com encoder incremental embutido foi escolhido. Adicionalmente, definiu-se o seguinte conjunto de peças para completar kit de transmissão:

- correia GT2 largura 6 mm
- polia de acionamento 20 dentes GT2 6 mm e furo 4 mm
- polia tensora com rolamento 20 dentes GT2 6 mm e furo 4 mm
- suporte em L para fixação do motor

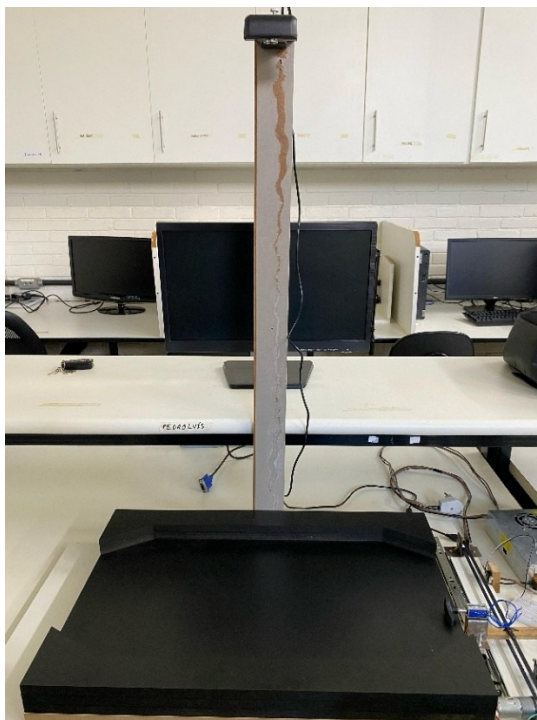
Para posicionar a câmera de forma a conseguir ter em seu alcance toda a extensão do campo de colisão, foi feita uma peça em MDF que permite a fixação do dispositivo de filmagem a uma altura de 1 metro, a qual foi anexada à parte lateral do campo de colisão. Também foram fabricados dois elementos a partir de chapas metálicas finas, um com o objetivo de fixar a polia tensora na mesma altura da polia de acionamento e outro para funcionar como carro deslizante. Por fim, dois sensores de fim de curso foram posicionados nos pontos extremos do deslocamento do interceptador com o auxílio de pequenos blocos de madeira. Sendo assim, após aquisição de todos os materiais necessários, a montagem física do campo de colisão foi finalizada e o resultado pode ser verificado nas Figuras 12, 13 e 14.

Figura 12 – Campo de colisão.



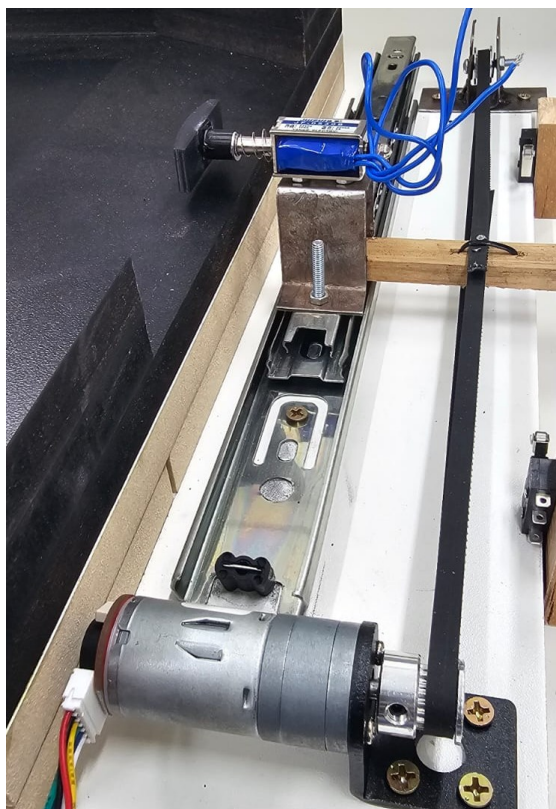
Fonte: Autor (2025).

Figura 13 – Câmera instalada no suporte suspenso.



Fonte: Autor (2025).

Figura 14 – Sistema do interceptador.



Fonte: Autor (2025).

Observa-se que carro deslizante foi associado a uma pequena tábua de madeira, a qual serve de eixo de fixação para que as extremidades da correia GT2 sejam presas. Portanto, a partir dessa associação e do sistema de transmissão entre as polias e a correia, o movimento angular do motor se transforma em linear. Assim, por ter liberdade de deslocamento sobre o trilho de gaveta, o carro se movimenta concomitantemente às rotações da máquina elétrica. Além disso, percebe-se que para executar a função de interceptador foi escolhido um mini atuador solenoide, com o objetivo de possibilitar, em trabalhos futuros, a associação de um sistema de rebatimento ao projeto. Ou seja, ao se deslocar até o ponto de encontro com o projétil esférico, o sistema poderá reagir. Por fim, a cor preta da madeira foi escolhida para facilitar o processo de detecção e rastreamento de uma bola de cor branca.

3.2. Sistema de Visão Computacional

Com o objetivo de possibilitar o processo de rastreamento do projétil, foi adquirida uma câmera de vídeo USB com resolução *full HD* e rosca tripé $\frac{1}{4}$ de polegada para fixação. O campo de visão do dispositivo, quando elevado na altura do suporte de MDF desenvolvido, era capaz de capturar toda a região de deslocamento da bola no campo de colisão com certa sobra, conforme mostrado na Figura 12. O equipamento é do modelo “General Webcam”, o qual não requer instalação adicional de *drivers* para o seu correto funcionamento.

Figura 15 – Foto do campo pela câmera suspensa.



Fonte: Autor (2025).

Após obter as informações visuais da câmera de maneira eficiente e confiável, é necessário criar o algoritmo de visão computacional responsável pelo rastreamento e previsão de trajetória do projétil no plano do campo de colisão. Dessa forma, utilizando como base o projeto “*Basket Predictor*” (Souza, 2022) e as bibliotecas OpenCV e cvzone, um programa em Python foi desenvolvido.

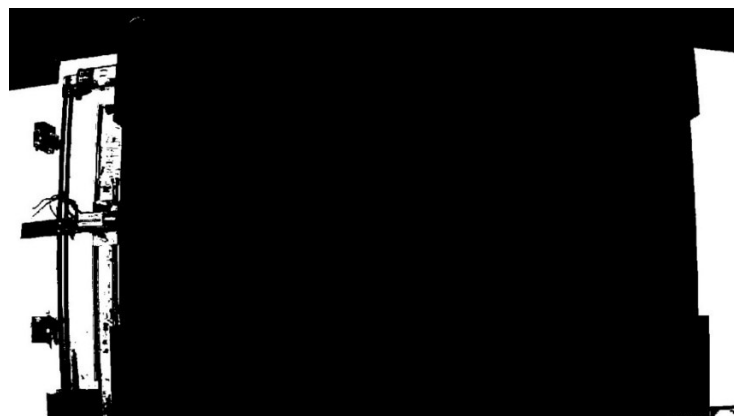
No processo de rastreamento de objetos por visão computacional, o primeiro passo é identificar os objetos por meio da diferenciação entre eles e o ambiente ao redor. Para isso, como discutido anteriormente, o campo de colisão foi projetado em preto, a fim de facilitar o contraste com um projétil de cor clara. Dessa forma, ao processar o frame capturado pela câmera e realizar a segmentação de cores utilizando o código HSV da bola, é possível aplicar um algoritmo de detecção de contornos. Esse algoritmo recebe uma imagem binária, como a mostrada na Figura 13, na qual são destacadas as áreas das cores desejadas (com os parâmetros definidos na Tabela 4) em branco, enquanto o restante é representado em preto. A partir dessa máscara, o algoritmo identifica todos os contornos externos da figura, localizando as bordas (regiões de transição abrupta de intensidade entre pixels adjacentes).

Tabela 4 – Parâmetros HSV para segmentação.

	H	S	V
Mínimo	0	0	170
Máximo	100	135	255

Fonte: Autor (2025).

Figura 16 – Foto binária do campo a partir de segmentação HSV.



Fonte: Autor (2025).

Devido ao fato de que o campo de visão da câmera supera os limites do espaço útil de movimento da bolinha, a base do campo, adquirida em MDF branco em função do melhor custo-benefício, estava influenciando negativamente no sistema. Isso porque a função de busca de contornos por segmentação HSV mostrou sensibilidade à sua cor, que é similar à cor do projétil. Tendo isso em vista, a primeira estratégia adotada para superar o problema foi a redução da área de análise por meio de um *slicing* matricial da imagem, selecionando uma subparte retangular que minimiza a presença da madeira de fundo branco. É válido ressaltar que os valores utilizados no recorte foram identificados de maneira empírica e são dependentes da orientação da câmera e, portanto, devem ser redefinidos em caso de quaisquer mudanças de posição ou de ângulo de inclinação no dispositivo. O resultado pode ser observado na Figura 17.

Figura 17 – Frame após *slicing*.



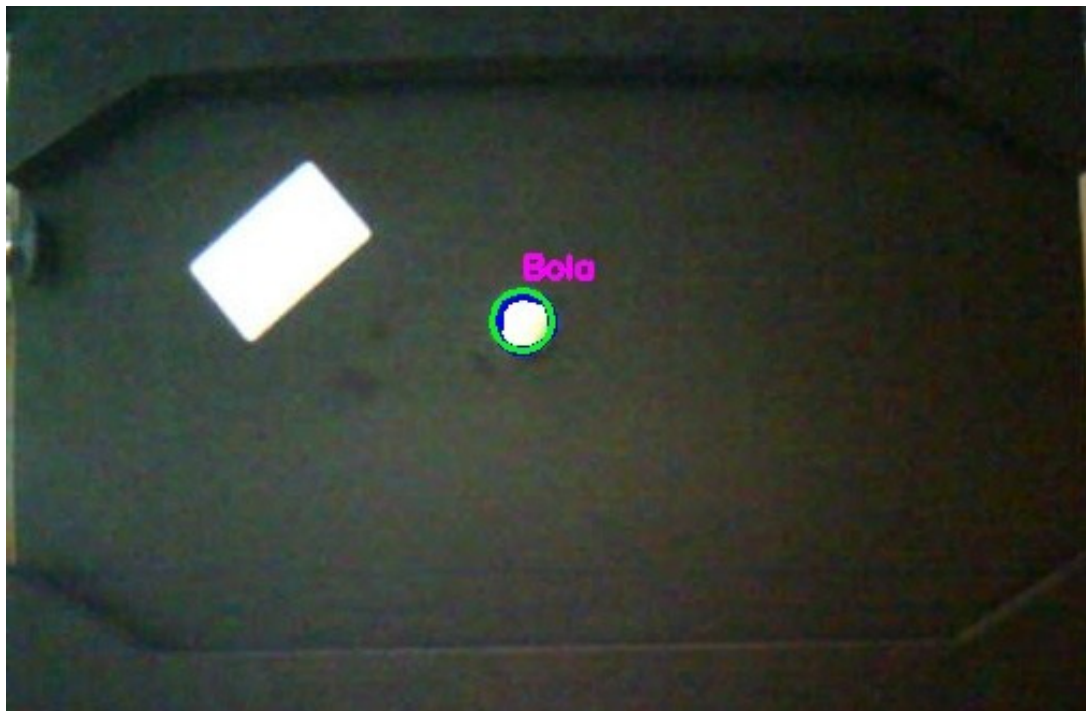
Fonte: Autor (2025).

Contudo, por mais que na imagem recortada a região branca da base não fosse tão expressiva, ela ainda gerava pequenos contornos que não correspondiam à bola. Dessa maneira, acrescentou-se ao código um filtro com áreas máxima e mínima, além de uma função de verificação de circularidade dos contornos encontrados. Assim, seja A a área de uma forma e P o seu perímetro, o índice de circularidade C , originalmente definido por Cox (1927), é dado pela Equação 1.

$$C = \frac{4 \cdot \pi \cdot A}{p^2} \quad (1)$$

Em um círculo perfeito, portanto, o valor de C é unitário. Então, ao encontrar um contorno, é checado se o seu valor de circularidade é próximo de um para que seja classificado como a fronteira do projétil de desejo, o qual possui geometria esférica. Especificamente no código desenvolvido, foi adotada uma tolerância para que contornos com parâmetro C entre 0,7 e 1,2 sejam considerados circulares e suas coordenadas centrais sejam armazenadas. Tal margem é importante para que pequenas imperfeições oriundas de iluminação ou quaisquer outros detalhes não impossibilitem a identificação da bola. Dessa maneira, mesmo que existam outros objetos na imagem com a mesma cor, o algoritmo focará na esfera, conforme observado na Figura 18.

Figura 18 – Projétil esférico encontrado.

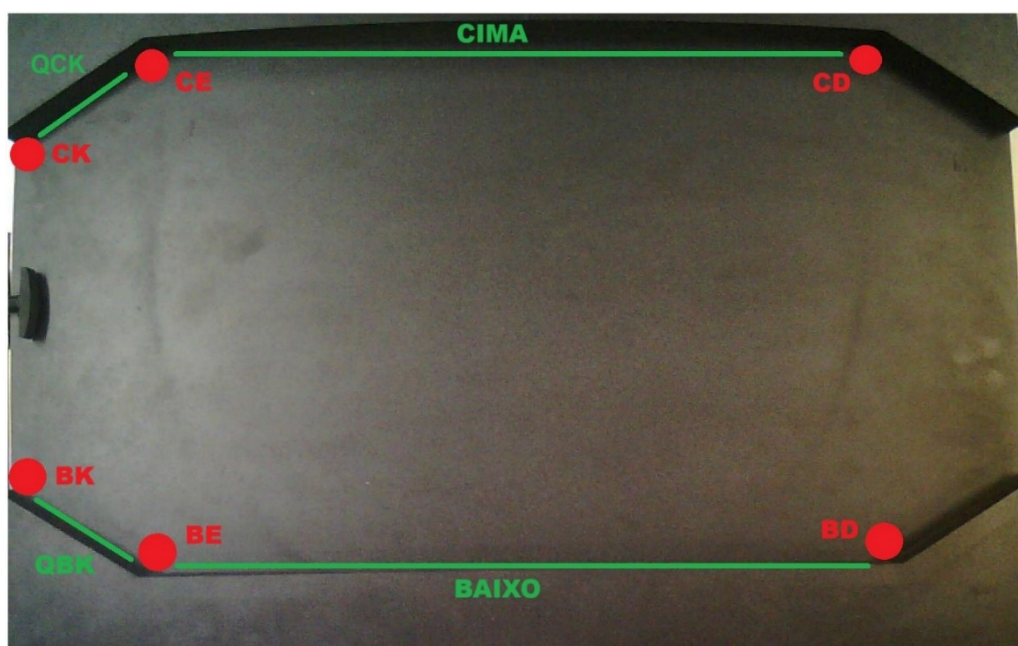


Fonte: Autor (2025).

Posteriormente, dado que é possível diferenciar o contorno do projétil de quaisquer outros, foi realizada a modelagem do campo de colisão. Para tanto, utilizou-se da função que armazena o centro do projétil esférico para identificar os pontos de interesse no campo de colisão. Sendo assim, posicionou-se a esfera nas quinas de interesse, apresentadas na Figura 19 em vermelho, e as coordenadas centrais foram

armazenadas. Sendo assim, a partir dos pontos adquiridos, foram definidas as funções que descrevem os limites físicos, demonstrados em verde na Figura. Na Tabela 5 estão especificadas as equações para cada barreira. Destaca-se uma simplificação do projeto na qual as quinas da parte direita do campo foram ignoradas, visto que o interceptador foi instalado do lado esquerdo e o projétil a priori o projétil não será atirado nas suas direções.

Figura 19 – Pontos de interesse.



Fonte: Autor (2025).

Tabela 5 – Funções das barreiras.

Barreira	Função 1º Grau	Domínio
Superior (Cima)	$y = 0,0139x + 51$	$75 \leq x \leq 435$
Inferior (Baixo)	$y = 0,0266x + 292$	$70 \leq x \leq 445$
Quina de Cima (QCK)	$y = -0,6666x + 100$	$10 \leq x \leq 74$
Quina de Baixo (QBK)	$y = 0,6666x + 243$	$10 \leq x \leq 69$

Fonte: Autor (2025).

A união das abscissas e ordenadas das funções gera o conjunto de pontos que descreve a fronteira que delimita o modelo do campo de colisão. Todo o processo de detecção dos limites do sistema físico foi feito em função da posição da câmera e, sendo assim, da mesma forma que no processo de *slicing* da imagem, quaisquer mudanças na orientação do dispositivo afetam os resultados e, caso ocorram, as

funções e os seus respectivos domínios devem ser redefinidos.

3.3. BeagleBone Black *Deployment*

Toda a preparação do sistema de visão computacional foi feita em um computador pessoal. Contudo, o escopo do projeto foi definido para que o processo pudesse ser realizado em um dispositivo embarcado. Sendo assim, conforme abordado previamente, a BeagleBone Black é um *hardware* que permite a instalação de um sistema operacional Linux e possui interfaces úteis para o problema em questão, como uma interface USB para comunicação com a câmera. Portanto, para que seja possível executar os algoritmos de visão computacional na SBC, é necessário configurar o *hardware*.

Grande parte dos procedimentos básicos para começar a utilizar a BBB podem ser encontrados no próprio site da BeagleBoard. O primeiro passo para conseguir acesso ao *hardware* em um computador pessoal é possuir um cabo mini USB e instalar os respectivos *drivers* para estabelecer uma rede via interface serial. A partir disso, ao conectar a placa ao computador, é possível acessar a placa por meio de um endereço IP fixo estabelecido na rede USB, que por padrão é 192.168.7.2. Quando a imagem instalada é de versões mais antigas, o ambiente de desenvolvimento *Cloud 9* pode ser acessado diretamente pelo navegador como uma *webpage* a partir do seu IP (entretanto, tal *software* foi descontinuado e em imagens novas não é possível acessá-lo).

Figura 20 – Versão Debian pré atualização.

```
root@beaglebone:~# lsb_release -da
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 7.4 (wheezy)
Release:        7.4
Codename:       wheezy
```

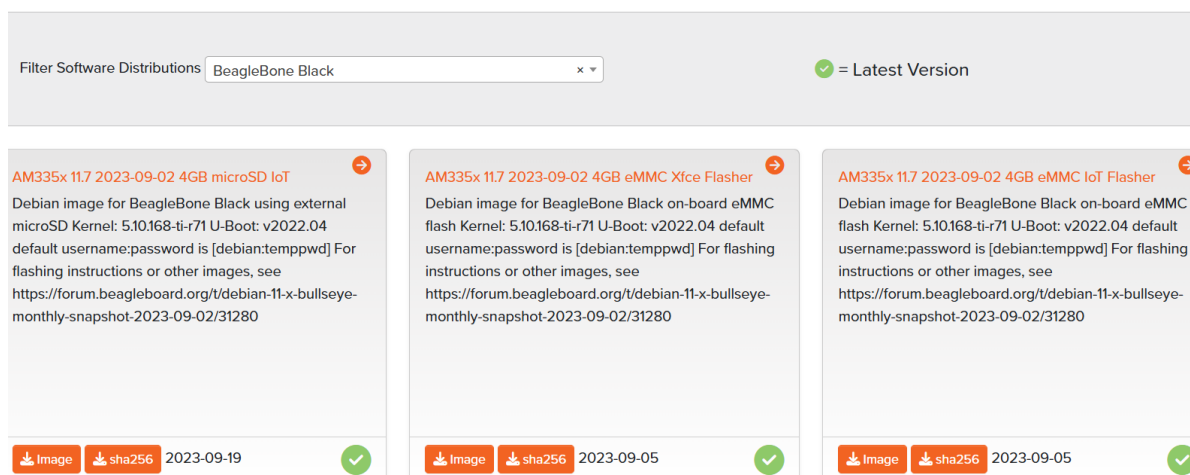
Fonte: Autor (2025).

Ao realizar o primeiro acesso via *Cloud 9*, conectou-se a porta ethernet da placa ao roteador para que o acesso à internet fosse possibilitado. Para funcionar, foi essencial garantir que a interface de rede da BeagleBone Black estava configurada

para aceitar DHCP (*Dynamic Host Configuration Protocol*). Assim, a placa recebeu automaticamente um IP ao ser conectada na rede e o *download* de diversos pacotes foi possibilitado. Contudo, ao tentar realizar a configuração das bibliotecas Python necessárias, diversas limitações foram surgindo, as quais estavam associadas à versão ultrapassada do kernel do Linux que estava instalado na SBC, a qual pode ser verificada na Figura 20.

Tendo isso em vista, decidiu-se formatar o sistema da BeagleBone Black com uma imagem Linux mais atualizada para a placa em específico. A última versão disponibilizada para a BBB encontrada no site da BeagleBoard foi uma distribuição Debian, conforme apresentado na Figura 21. Como metodologia para atualizar o kernel, escolheu-se utilizar um cartão microSD externo bootável. Para tanto, foi realizado o *download* da imagem ‘AM335x 11.7 2023-09-02 4GB microSD IoT’ e, com o auxílio da ferramenta balenaEtcher (Figura 22), o sistema operacional foi gravado no dispositivo de mídia. A versão IoT foi escolhida por ser mais compacta e leve, visto que não inclui interface gráfica.

Figura 21 – Imagens Linux BBB.

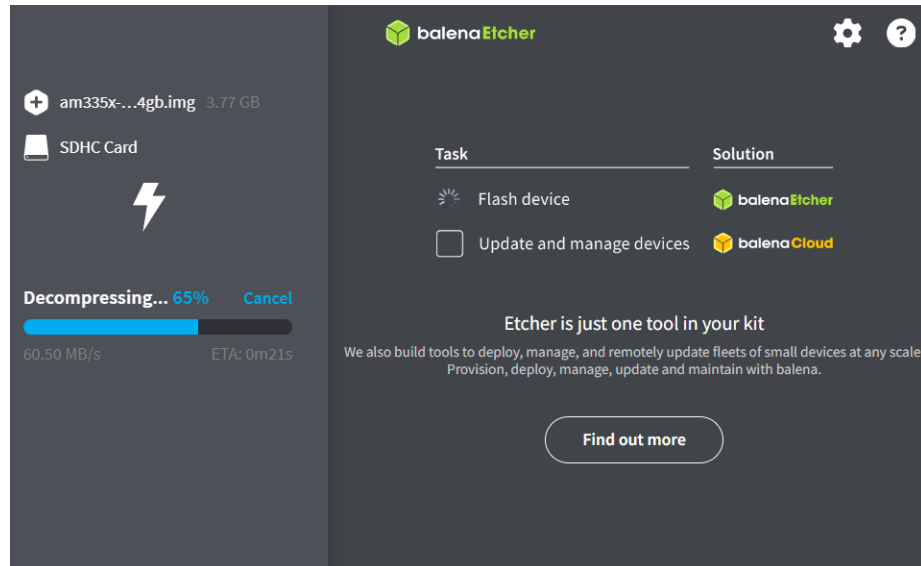


Fonte: Autor (2025).

A partir da configuração do cartão microSD com a imagem Debian, foi necessário realizar o *boot* por meio dele. Para tanto, o procedimento consiste em inserir o dispositivo de mídia na BeagleBone Black ainda desligada e energizar o *hardware* enquanto o botão S2 é pressionado. Para acessar a SBC, a partir da atualização, foi escolhido utilizar o *software* Putty, útil para estabelecer conexões seriais, por SSH e por telnet. Dessa forma, ao conectar à BBB, verificou-se que o

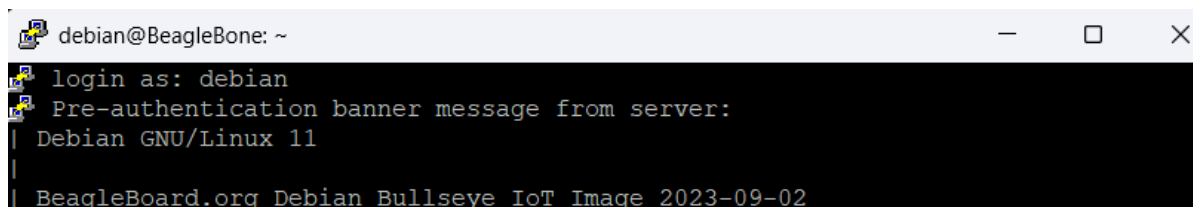
sistema operacional resultante do processo de inicialização era da versão atualizada, como pode ser observado nas Figuras 23 e 24.

Figura 22 – Flash da imagem no cartão com balenaEtcher.



Fonte: Autor (2025).

Figura 23 – Login após atualização do sistema operacional.



Fonte: Autor (2025).

Figura 24 – Nova versão Debian BBB.

```
debian@BeagleBone:~$ lsb_release -da
No LSB modules are available.
Distributor ID: Debian
Description:   Debian GNU/Linux 11 (bullseye)
Release:      11
Codename:     bullseye
```

Fonte: Autor (2025).

Contudo, para que o sistema operacional seja atualizado de maneira definitiva, com o objetivo de que não seja necessário manter o cartão conectado e realizar a mesma sequência de boot citada toda vez que o hardware é inicializado, é necessário carregá-lo para a memória flash. Para tanto, existe um *script* padrão na imagem

Debian instalada denominado *enable-beagle-flasher*, o qual deve ser executado antes de reiniciar a BeagleBone Black, resultando na cópia permanente do SO para a memória interna da placa. É válido citar que tal procedimento apaga todos os dados prévios e requer a realização de *backup* caso existam informações importantes.

Por fim, foi necessário instalar a biblioteca Python para visão computacional. Nesse sentido, utilizando o sistema de gerenciamento de pacotes *pip*, instalou-se o módulo OpenCV na sua versão *headless*, a qual não possui funções de *display* dado que nela não há suporte para interfaces gráficas. A partir dessa biblioteca, é possível realizar todos os cálculos e manipulações nas imagens, não permitindo, contudo, visualizá-las em tempo real. Essa abordagem é particularmente útil para a aplicação uma vez que a BeagleBone Black não será associada a um monitor, evitando a instalação de pacotes que não serão utilizados.

A partir da instalação do *Opencv-headless* e da subsequente tentativa de uso, verificou-se a necessidade da instalação de pacotes adicionais de sistema para o Linux que não foram incluídos nativamente na imagem Debian utilizada. Tais pacotes atribuíam funcionalidades para que a biblioteca fosse capaz de interagir com diversos formatos de arquivos multimídia, especialmente imagens e vídeos, além de integrar ferramentas para a realização de operações algébricas lineares. Dentre as bibliotecas de sistema instaladas, podem ser citadas:

- *libavcodec-dev*.
- *libavformat-dev*.
- *libswscale-dev*.
- *libavutil-dev*.
- *libwebp-dev*.
- *libjpeg-dev*.
- *libpng-dev*.
- *libopenjp2-7-dev*.
- *libblas-dev*.
- *libatlas-base-dev*.

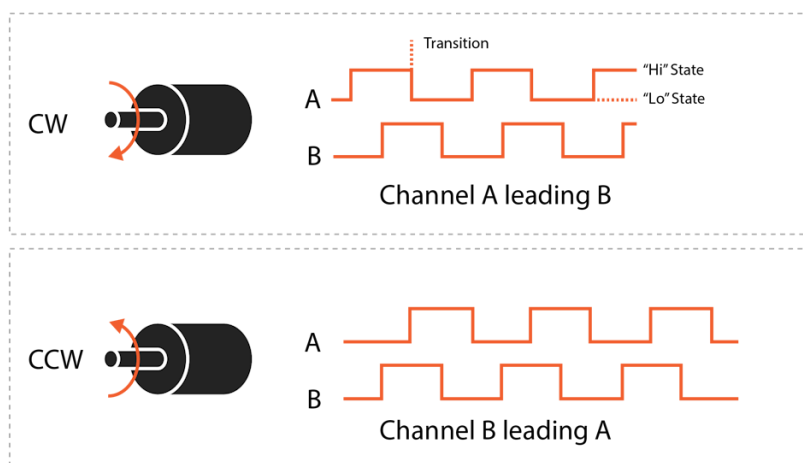
Após todas essas configurações iniciais para permitir a execução da rotina de visão computacional na BeagleBone Black, o *hardware* teve sua memória não volátil ocupada em 93%. Sendo assim, grande parte do seu armazenamento foi requerida durante o processo de preparação básica, mesmo tendo evitado a instalação de

recursos gráficos na SBC. Contudo, tal cenário não se demonstrou muito preocupante dado que não serão criados novos volumes durante o funcionamento do sistema de visão, que demandará apenas recursos de processamento e da memória volátil.

3.4. Interceptador

Para controlar o sistema do interceptador, foi decidido utilizar um ESP32 WROOM em função da sua acessibilidade, dado que ele apresenta uma boa relação custo-benefício, e da facilidade nos processos de configuração e de programação das suas aplicações, que podem ser realizados em ambientes de desenvolvimento simples e consolidados, como o Arduino IDE e PlatformIO. A responsabilidade do microcontrolador, portanto, é controlar o motor para atingir a posição de interceptação desejada, a partir dos sinais de *feedback* do encoder. Nesse sentido, como o sensor de rotação utilizado é incremental de quadratura, vários pulsos referentes às revoluções são gerados em dois canais distintos, conforme demonstrado na Figura 25. Sendo assim, para trabalhar com eles e interpretá-los de maneira confiável, é essencial que o dispositivo eletrônico seja capaz de trabalhar com bordas de subida e descida nos seus pinos digitais em uma frequência de aquisição elevada.

Figura 25 – Pulsos encoder de quadratura.



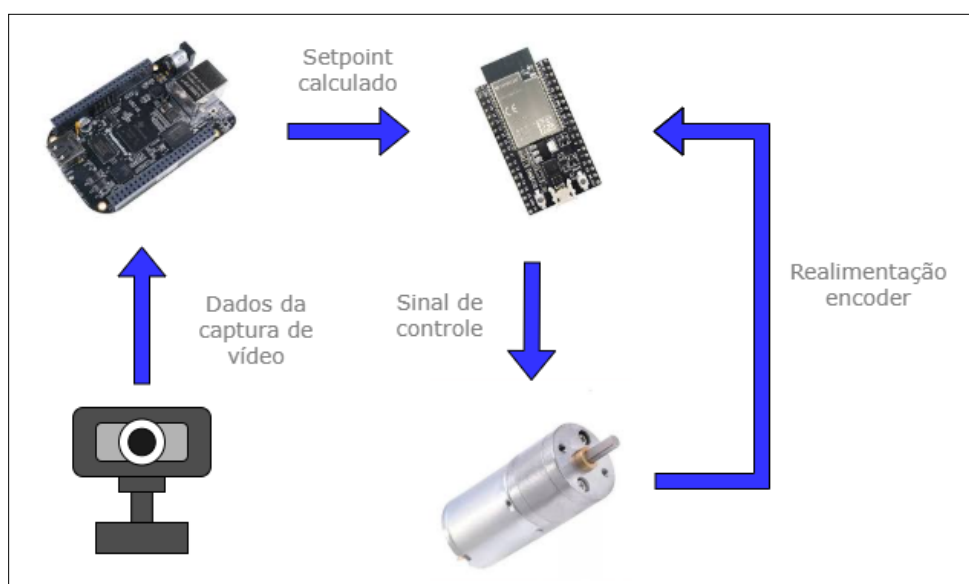
Fonte: Smith (2023) - adaptado.

A priori, foi ponderada a possibilidade de realizar o controle do interceptador também pela BeagleBone Black. Contudo, é necessário considerar que a tarefa de visão computacional exige um alto processamento e considerável parte da memória

de acesso da BBB, fazendo com que um sistema adicional com acionamento e sensoramento, os quais deveriam ocorrer em baixos intervalos de tempo, pudesse acarretar em uma grande perda de desempenho. Além disso, foram encontradas certas dificuldades para trabalhar com o encoder na SBC, visto que se mostrou necessário alterar algumas configurações na *device tree* do sistema com drivers específicos, os quais impactam diretamente na relação de hardware do kernel Linux de forma a lidar com interrupções nos seus pinos digitais.

Tendo tais aspectos em vista e com o objetivo de simplificar essa etapa, decidiu-se dividir as funções do projeto em dois dispositivos: a BeagleBone Black assume a responsabilidade pela rotina de captura de posição e de trajetória do projétil de maneira integral enquanto o ESP32 lida com a leitura do encoder e controla o movimento do interceptador. Para integrar ambos sistemas, portanto, é necessário estabelecer uma comunicação entre os processos que são executados em cada *hardware*, permitindo a troca de informações entre os dois para que a interceptação em tempo real ocorra. O diagrama simplificado que descreve essa estrutura é apresentado na Figura 26.

Figura 26 – Arquitetura do projeto.



Fonte: Autor (2025).

Sendo assim, para que o ESP32 seja capaz de receber os dados relevantes do sistema de visão computacional da BBB, configurou-se uma interface serial UART (*Universal Asynchronous Receiver-Transmitter*) para estabelecer a comunicação

entre a SBC e o microcontrolador. Nesse protocolo, os dados são transmitidos bit a bit em dois fios sem a necessidade de um sinal de *clock*, sendo que basta estabelecer a mesma taxa de transmissão (*baud rate*) entre os dispositivos com uma mesma referência de tensão. A ideia é enviar o *setpoint* final, referente ao ponto em que a bola deve cruzar o eixo do interceptador a partir dos cálculos realizados, apenas nos momentos em que o seu valor sofre alguma alteração, maximizando a eficiência em termos de latência e de largura de banda. Os pinos definidos como transmissor (TX) e receptor (RX) em ambos os hardwares podem ser verificados na Tabela 6. O *baud rate* definido para troca de informações no barramento serial foi de 115200 bits por segundo.

Tabela 6 – Pinos para comunicação UART.

	BeagleBone Black	ESP32
Receptor (RX)	P9-26	16
Transmissor (TX)	P9-24	17

Fonte: Autor (2025).

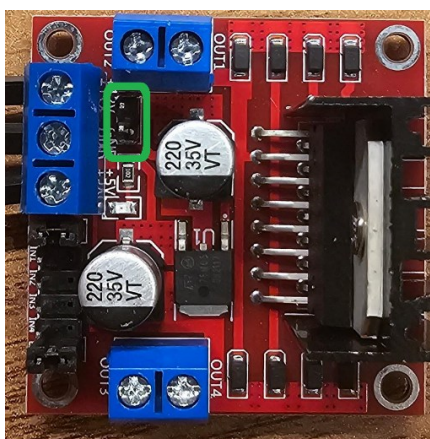
No lado da BeagleBone Black, foi acrescentada ao sistema, com o uso da biblioteca *pyserial*, a funcionalidade de envio de dados na porta serial TTYs1, a qual corresponde à UART1, que possui como interface física os pinos P9-24 e P9-26 da SBC. O *setpoint* que deve ser enviado é um *unsigned short* int, que possui 2 bytes, e a ordem de envio deve seguir o formato *little endian* (byte menos significativo é enviado e recebido primeiro), visto que é o padrão de arquitetura usado pelos processadores de cada um dos *hardwares* (ARM-Cortex A8 e Xtensa LX6). No ESP32, aproveitou-se os seus dois núcleos para separar as tarefas de comunicação e de controle. Sendo assim, a *task* do *core* zero realiza a leitura do barramento serial constantemente, atualizando o valor da variável de *setpoint* sempre que recebe dois bytes.

Existe um detalhe relacionado ao uso dos terminais P9-24 e P9-26 da BeagleBone Black como interface para comunicação serial. Para que o canal de comunicação funcione corretamente, é necessário alterar a configuração de ambos os pinos de GPIO para UART. Essa alteração é realizada por meio dos comandos ‘config-pin P9.24 uart’ e ‘config-pin P9.26 uart’. No entanto, sempre que a BBB é reiniciada e o processo de *boot* ocorre, o modo dos pinos retorna à configuração

padrão. Com isso em mente, foi desenvolvido um script com esses comandos que deve ser executado pelo shell do Linux. Para automatizar esse procedimento de configuração, foi adicionado ao crontab do sistema o agendamento de execução do script sempre que o sistema for inicializado.

Após a validação da interface UART de comunicação entre a BeagleBone Black e o ESP32, focou-se no sistema de movimento do interceptador. Para acionar o motor, utilizou-se um módulo ponte H dupla L298N, igual à apresentada anteriormente na Figura 8, alimentada com uma fonte chaveada com saída igual à tensão nominal do motor (24 V) e potência 360 W. Esse módulo possui um regulador interno para alimentar o seu circuito lógico a partir da tensão de alimentação fornecida. Contudo, o valor de 24 V já não está na margem segura de uso desse dispositivo eletrônico, fazendo com que seja prudente isolar os dois circuitos para que o módulo não estrague. Isso deve ser feito removendo o jumper que realiza tal conexão, o qual é destacado em verde na Figura 27. Consequentemente, é necessário fornecer a alimentação lógica do sistema, que pode ser feita em 3,3 V ou em 5 V, facilitando o uso do próprio microcontrolador para essa função.

Figura 27 – Jumper de ativação do regulador do módulo L298N.



Fonte: Autor (2025).

Para conseguir controlar a posição do interceptador a partir da leitura dos sinais do encoder, é necessário passar por um estágio de calibração. Nessa etapa, o carro central é levado às duas extremidades do eixo de interceptação e o número total de pulsos para percorrer o trajeto completo é armazenado na memória do microcontrolador. Isto é, durante a execução do programa, o ESP32 manda um sinal de acionamento para o motor a partir da ponte H até identificar que atingiu o primeiro

sensor fim de curso. A seguir, a contagem de pulsos é zerada e o sentido de giro do motor é invertido até atingir o segundo sensor, momento no qual a quantidade de pulsos geradas pelo encoder é armazenada. Sendo assim, dado que a distância entre as duas pontas é medida em pulsos, é possível definir o deslocamento desejado do interceptador também em função de pulsos.

Conforme abordado anteriormente, é possível manipular o valor e o sinal da tensão de saída do módulo ponte H a partir da aplicação de sinais PWM nos seus pinos. A tensão de saída, portanto, é aplicada nos terminais do motor e a sua variação implica na alteração da velocidade de rotação da máquina. A velocidade de rotação, por sua vez, está diretamente relacionada à posição do interceptador no eixo linear. Tendo isso em vista, é possível fechar uma malha de controle de posição para o interceptador, onde a variável de processo é a posição do interceptador, medida a partir dos pulsos de *feedback* do encoder, e a variável manipulada é o sinal de saída PWM.

A partir disso, a estratégia escolhida para controlar a posição do dispositivo foi o algoritmo PID, devido a capacidade de ser regulado de forma simples para atingir a performance necessária para a aplicação. É válido ressaltar que o controle de posição a partir do motor não é um processo *self-regulating*, ou seja, a saída (posição) é alterada continuamente desde que a entrada não seja nula. Em outras palavras, não existe um ponto de equilíbrio natural que faz com que a resposta do sistema se estabilize mesmo que exista estímulo na sua entrada (enquanto a rotação do motor ocorrer, em função do PWM, a posição do interceptador continuará sendo alterada). O método para realizar o *tunning* dos parâmetros da malha de controle foi experimental, sem modelagens prévias ou simulações computacionais.

O sinal de PWM do ESP32 vai de 0 a 255, mas observou-se que abaixo de 40 a alimentação não fornece ao motor um torque suficiente para conseguir vencer a inércia inicial, fazendo com que não haja rotação. Sendo assim, para que não exista uma *deadzone* forçada no sistema, o limite inferior do controle foi definido em 40, de forma que mesmo saídas pequenas do algoritmo de controle consigam interferir na posição. Como resultado dessa mudança e em função da dinâmica observada do sistema, os ganhos do algoritmo, os quais podem ser observados na Tabela 7, foram definidos, sendo que decidiu-se zerar a parcela derivativa e utilizar um pequeno termo integrativo.

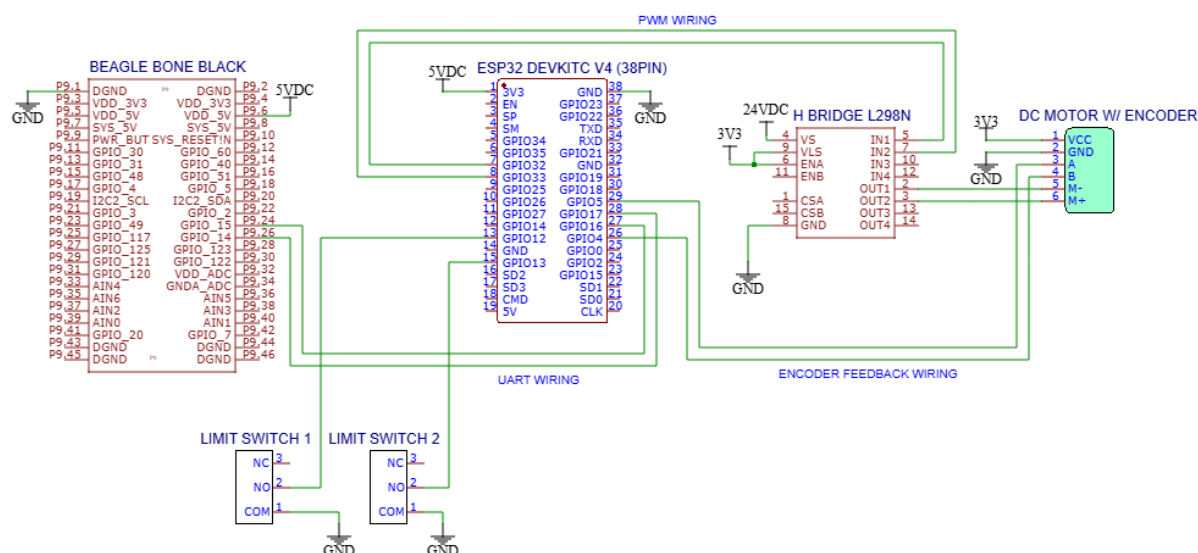
Tabela 7 – Ganhos do PID.

Termo	Ganho
Proporcional	0,150
Integral	0,001
Derivativo	0

Fonte: Autor (2025).

3.5. Projeto Eletroeletrônico

Tendo em vista todos os aspectos eletroeletrônicos que foram citados, foi construído o diagrama esquemático (Figura 28) que reúne todos os dispositivos utilizados, sendo que os pinos e as conexões refletem de maneira exata o projeto construído. É necessário garantir que todos os componentes compartilhem a mesma referência de terra (GND), pois isso evita flutuações de sinal, ruídos indesejados e problemas de funcionamento que podem ocorrer devido a diferenças de potencial entre os dispositivos

Figura 28 – Diagrama esquemático do projeto.

Fonte: Autor (2025).

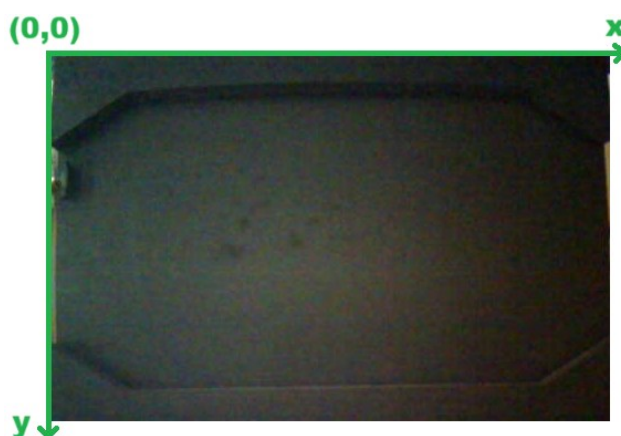
4. RESULTADOS E DISCUSSÕES

A partir da finalização do processo de construção do sistema e da validação do algoritmo de rastreamento do projétil, foi possível desenvolver o sistema final de predição de trajetória, o qual é capaz de calcular antecipadamente, a partir dos dados adquiridos em tempo real, o *setpoint* referente à posição que a interceptação deve ocorrer. Além disso, o projeto desenvolvido foi testado em diferentes condições para verificar a sua precisão, a sua robustez e as suas limitações. Os resultados alcançados, portanto, serão apresentados nesse capítulo juntamente a quaisquer observações ou discussões pertinentes.

4.1. Modelagem da Colisão

O sistema de visão computacional desenvolvido é capaz de identificar a posição da bola sequencialmente a partir dos *frames* recebidos da câmera. Sendo assim, para conseguir estimar o ponto futuro do projétil, é possível utilizar as suas posições anteriores, especialmente por se tratar de um movimento plano bidimensional, sem variações de profundidade. Para tanto, sempre que a bola é identificada na imagem, o pixel referente ao seu centro é armazenado em duas parcelas, uma referente ao eixo horizontal e outra ao eixo vertical. Tais coordenadas partem do mesmo sistema ordenado utilizado para identificar as quinas e as barreiras do campo de colisão, o qual é apresentado na Figura 29.

Figura 29 – Eixos do sistema ordenado dos pixels.



Fonte: Autor (2025).

A partir dessas coordenadas centrais, desde que mais de um ponto tenha sido armazenado, é possível aplicar uma regressão linear para estimar a trajetória do projétil. Para tanto, as funções `polyfit` e `poly1d` do pacote Numpy foram utilizadas, facilitando a integração com as funções da biblioteca OpenCV. Tal abordagem assume que, em intervalos curtos, o deslocamento da bola pode ser aproximado por um comportamento linear. Tal premissa é bastante razoável para o contexto do trabalho, mas, certamente, não contempla todos os cenários. Contudo, por mais que não seja integralmente precisa, trabalhar com retas é uma simplificação coerente no contexto de interceptação no campo plano.

Tendo isso em vista, trabalhou-se com uma regressão “móvel”, em que o número máximo de pontos amostrais para o cálculo dos coeficientes foi definido em cinco. Sendo assim, ao se adquirir uma sexta coordenada central do projétil, a primeira coordenada armazenada é descartada, caracterizando uma fila do tipo *First In, First Out* (FIFO). Dessa forma, o sistema aguarda por, no mínimo, dois pontos para identificar a reta que melhor se ajusta ao conjunto de coordenadas por meio do método dos mínimos quadrados. Tal reta, portanto, diz respeito à direção atual de deslocamento do projétil, que pode ser estendida para a trajetória futura. O tamanho limitado do conjunto de dados faz com que a regressão seja mais sensível às mudanças de comportamento da bola, o que resulta em uma previsão com resposta mais rápida, adaptativa e assertiva.

O campo foi construído de maneira que as suas laterais limitem o movimento do projétil, permitindo o seu escape apenas no eixo do interceptador. A depender da direção de deslocamento, portanto, a bola colide com as barreiras, fazendo com que o sentido do seu movimento seja alterado, similar ao que acontece com as bolas de bilhar. Sendo assim, quando tal cenário é identificado, o sistema esquece os pontos antigos para que a regressão não seja afetada pela direção anterior, o que também contribui para uma maior velocidade e adaptabilidade do sistema. Além disso, essa estratégia também é útil para as situações em que a bola tem sua dinâmica alterada, seja por efeitos de rotação ou por quaisquer instabilidades no sistema que resultam em mudanças no sentido da bola.

Além disso, com o objetivo de agilizar o processo de novas regressões, foi adicionado um fator de esquecimento ao sistema. A partir dele, quando 30 ciclos da rotina de visão computacional são executados sem sucesso na identificação do projétil, a fila de pontos centrais que estava previamente armazenada é reiniciada.

Esse comportamento é útil para encontrar rapidamente a nova regressão assim que o projétil volta a ser detectado no campo de colisão, visto que retira a interferência dos pontos antigos, que podem fazer com que o processo de detecção da reta atualizada fique com baixa precisão ou incoerente a depender do posicionamento das coordenadas anteriores.

Entretanto, identificar apenas a direção atual de deslocamento do projétil não é suficiente para prever em qual posição a interceptação deve ocorrer. Sendo assim, além de realizar a regressão do movimento presente, é necessário levar em consideração as possíveis colisões futuras nas laterais do campo para estimar a trajetória de maneira integral. Ou seja, a partir dos últimos pontos adquiridos, deve-se estimar em qual ponto a bola cruzará o espaço de movimento do interceptador. Para tanto, a partir da direção no momento, um algoritmo de projeção é executado para identificar onde acontecerão reflexões (desde que o sentido de deslocamento se mantenha) e para estimar os novos caminhos que o projétil percorrerá.

Tendo isso em vista, o algoritmo desenvolvido para avaliar toda a trajetória futura do sistema se baseia em uma premissa básica que considera que o sentido de deslocamento da bola sempre é da direita para a esquerda. Dessa forma, primeiramente é checado se a reta de regressão atual da bola leva diretamente ao eixo de interceptação, pois nesse cenário já se sabe em qual coordenada o interceptador deve ser posicionado para impedir o avanço do projétil. Caso isso não ocorra, busca-se encontrar o ponto de intersecção entre a reta de deslocamento do projétil e as funções das barreiras para identificar possíveis colisões que influenciam no percurso da esfera.

Sabe-se que, no caso de duas retas não paralelas, existirá somente um ponto em que elas se interceptam. Dessa forma, partindo da premissa de que os coeficientes angulares da reta da trajetória da bola (Equação 2) e da reta da barreira (Equação 4) são distintos entre si, é calculado o possível ponto de colisão do projétil igualando as duas expressões, conforme demonstrado na Equação 4. A partir dela, é possível colocar em evidência o ponto de intersecção que se deseja obter ao posicionar os termos que o acompanham no lado esquerdo da equação. Em seguida, basta isolar x para encontrar a expressão que fornece a coordenada de encontro entre as retas, a qual é apresentada na Equação 5. Portanto, o ponto x_{INTERSEC} é o quociente entre a diferença dos coeficientes lineares ($\beta_{\text{BARREIRA}} - \beta_{\text{BOLA}}$) e a diferença dos coeficientes angulares ($\alpha_{\text{BOLA}} - \alpha_{\text{BARREIRA}}$) das retas.

$$y_{BOLA} = \alpha_{BOLA} \cdot x + \beta_{BOLA} \quad (2)$$

$$y_{BARREIRA} = \alpha_{BARREIRA} \cdot x + \beta_{BARREIRA} \quad (3)$$

$$y_{BOLA}(x_{INTERSEC}) = y_{BARREIRA}(x_{INTERSEC}) \quad (4)$$

$$\therefore \alpha_{BOLA} \cdot x_{INTERSEC} + \beta_{BOLA} = \alpha_{BARREIRA} \cdot x_{INTERSEC} + \beta_{BARREIRA} \rightarrow$$

$$\alpha_{BOLA} \cdot x_{INTERSEC} - \alpha_{BARREIRA} \cdot x_{INTERSEC} = \beta_{BARREIRA} - \beta_{BOLA} \rightarrow$$

$$x_{INTERSEC}(\alpha_{BOLA} - \alpha_{BARREIRA}) = \beta_{BARREIRA} - \beta_{BOLA} \rightarrow$$

$$x_{INTERSEC} = \frac{\beta_{BARREIRA} - \beta_{BOLA}}{\alpha_{BOLA} - \alpha_{BARREIRA}} \quad (5)$$

A partir do valor encontrado por meio da Equação 5, é preciso verificar se ele está no *range* de valores válidos, ou seja, se ele está posicionado à esquerda da última posição da bola ($x_{INTERSEC} < x_{BOLA}$, considerando o eixo ordenado da Figura 29) e se ele pertence ao domínio da função afim da barreira analisada, conforme apresentado previamente na Tabela 5. Quando esses dois requisitos são positivos, o algoritmo considera como confirmada a colisão do projétil no ponto calculado, fazendo com que seja necessário estimar uma nova reta direcional para a bola após o choque com os limites físicos do campo.

Dessa forma, existem dois tipos de abordagem, uma para quando a colisão ocorre nas laterais e outra para quando ela ocorre nas quinas anguladas. Se a colisão identificada foi em alguma das duas quinas próximas ao interceptador, o algoritmo desenvolvido considera que a interceptação deve ocorrer no ponto limite do curso do interceptador mais próximo à respectiva quina. Tal consideração foi feita para simplificar a previsão de trajetória, levando em conta que essa dinâmica se aproximou do comportamento observado empiricamente no campo. Em contrapartida, caso a colisão prevista ocorra na barreira superior ou na inferior, a trajetória futura do projétil é estimada a partir da metodologia da física ótica para o cálculo dos raios refletidos em um espelho plano.

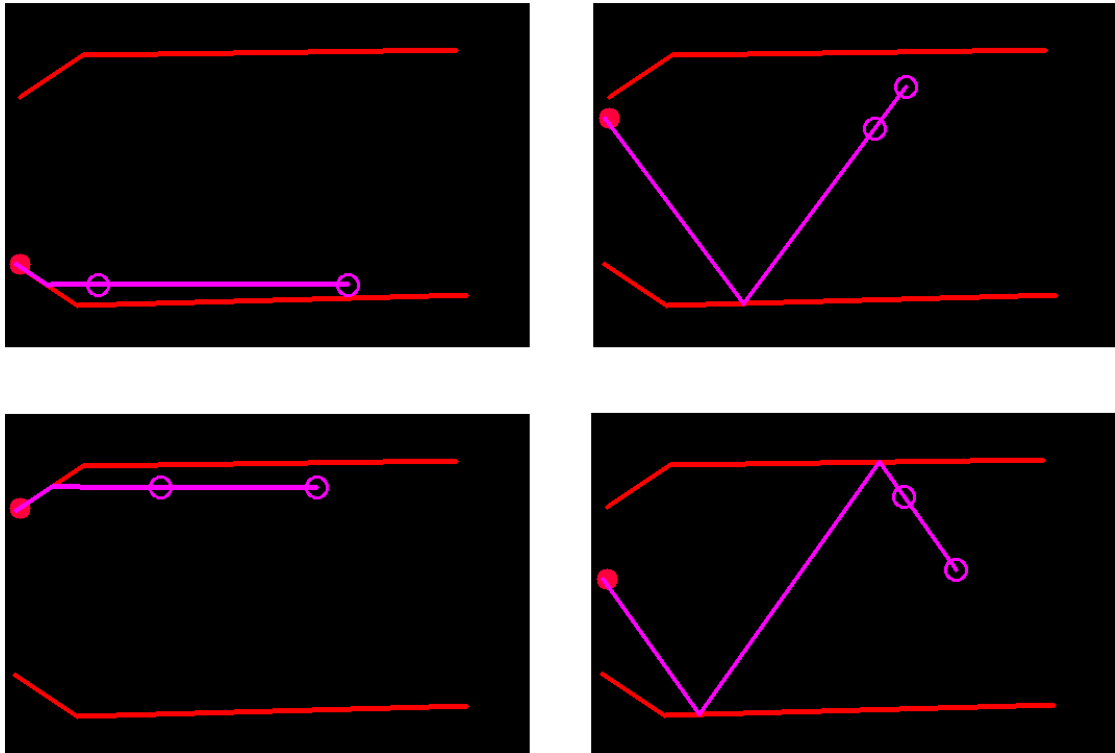
Isto é, a partir da direção atual da bola e da equação da barreira, os coeficientes da reta que aponta para a direção refletida são calculados. Tais parâmetros são encontrados a partir de uma abordagem simplificada puramente geométrica, em que se considera que o ângulo de incidência da bola na barreira é igual ao seu ângulo de saída. Sendo assim, para o coeficiente angular basta inverter o seu sinal, dado que as barreiras são praticamente horizontais ($\alpha \cong 0$). Contudo, para o coeficiente linear, deve-se encontrar o valor que valida a igualdade entre a reta refletida e a função da barreira no ponto de intersecção encontrado a partir da solução da Equação 5.

Tendo isso em vista, para definir a reta resultante da reflexão do projétil em alguma das barreiras laterais, dado que o coeficiente angular atualizado está definido, é preciso encontrar o novo coeficiente linear. Para tanto, inicialmente, é necessário encontrar a coordenada de intersecção no eixo y da imagem. Isso pode ser feito substituindo o valor de x_{INTERSEC} na equação da trajetória prévia da bola, como apresentado na Equação 6. Posteriormente, ao igualar os valores da equação antiga da bola e da nova que está sendo estimada no ponto x de intersecção, o termo β_{REFLET} pode ser isolado. Sendo assim, como resultado, associando os novos coeficientes, tem-se a Equação 7, que estima a reta da trajetória refletida da bola após a colisão.

$$\begin{aligned}\alpha_{\text{REFLET}} &= -\alpha_{\text{BOLA}} \\ y_{\text{INTERSEC}} &= y_{\text{BOLA}}(x_{\text{INTERSEC}}) \\ \therefore y_{\text{INTERSEC}} &= \alpha_{\text{BOLA}} \cdot x_{\text{INTERSEC}} + \beta_{\text{BOLA}} \\ y_{\text{REFLET}}(x_{\text{INTERSEC}}) &= y_{\text{INTERSEC}} \rightarrow \\ \alpha_{\text{REFLET}} \cdot x_{\text{INTERSEC}} + \beta_{\text{REFLET}} &= y_{\text{INTERSEC}} \rightarrow \\ \beta_{\text{REFLET}} &= y_{\text{INTERSEC}} - \alpha_{\text{REFLET}} \cdot x_{\text{INTERSEC}} \\ y_{\text{REFLET}} &= -\alpha_{\text{BOLA}} \cdot x + (y_{\text{INTERSEC}} - \alpha_{\text{REFLET}} \cdot x_{\text{INTERSEC}})\end{aligned}\tag{6}$$

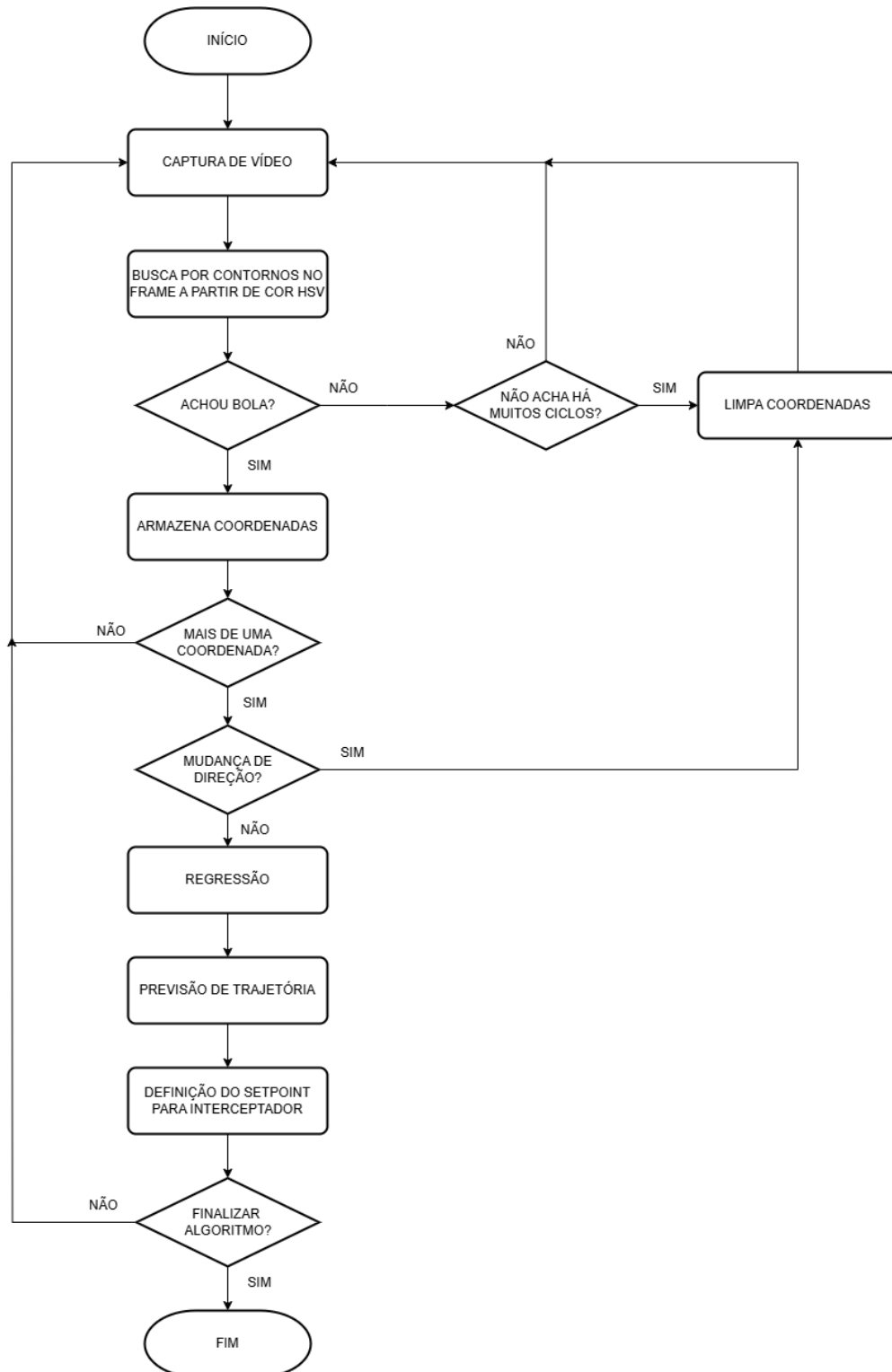
$$\tag{7}$$

Figura 30 – Trajetórias estimadas a partir do algoritmo desenvolvido.



Fonte: Autor (2025).

Figura 31 – Fluxograma da rotina de *tracking*.



Fonte: Autor (2025).

Dessa maneira, uma nova direção de deslocamento é estimada para o projétil, permitindo que a projeção futura de sua trajetória seja feita novamente, até que o ponto final no eixo do interceptador seja encontrado. A Figura 30 apresenta, nos

segmentos em rosa, a rota estimada em quatro situações hipotéticas, sendo que as linhas vermelhas são referentes às equações das barreiras, os círculos ocos em rosa são as duas últimas coordenadas da bola utilizadas para o cálculo e o círculo vermelho preenchido é a posição de interceptação prevista.

É válido ressaltar que essa abordagem para o cálculo antecipado do ponto de interceptação conta com um reajuste constante a partir da trajetória real do projétil. Isto é, mesmo que uma coordenada tenha sido estimada a partir da projeção geométrica, após a colisão física da bola na barreira, o sistema começa a analisar a rota efetiva que a esfera realiza após a reflexão para novamente tentar prever em qual região o interceptador deve ser posicionado. Dessa maneira, mesmo que o resultado prévio gerado pelo algoritmo apresente divergência em relação ao rebatimento real, devido às premissas e às aproximações utilizadas, o erro é corrigido.

Tendo isso em vista, quando o algoritmo consegue identificar a coordenada de interceptação, o seu valor no eixo vertical é enviado ao sistema de interceptação do ESP32 via interface serial. Esse envio ocorre toda vez que o valor estimado sofre uma alteração expressiva no seu valor, para não exigir uso constante da UART e para não ficar alterando constantemente o *setpoint* do controle de posição. O fluxograma de execução total do sistema de rastreamento e previsão é apresentado na figura 31

4.1.1. Transformação de domínio e *setpoint* final

A primeira etapa para conseguir controlar a posição do interceptador a partir do *setpoint* recebido do sistema de visão computacional é traduzir o valor da variável do domínio dos pixels, que foi transmitido via interface UART, para o domínio físico da guia linear. Isso é feito por meio de uma interpolação linear, que correlaciona a coordenada y do pixel de interceptação calculado na rotina de *tracking* com a quantidade de pulsos necessária para atingir esse mesmo ponto no campo de colisão. Para tanto, é necessário possuir os valores máximos e mínimos do curso de movimento do interceptador em ambos sistemas.

Sendo assim, o limiar de pulsos do encoder é definido durante a etapa de inicialização do sistema de controle de posição, enquanto os pixels limites foram obtidos empiricamente a partir da análise dos frames, cujo resultado depende do arranjo da câmera e do processo de *slicing* da imagem. Uma vez que tais valores foram adquiridos, é possível encontrar o *setpoint* (SP), correspondente à quantidade

de pulsos necessários para atingir a posição de interceptação, que deve ser utilizado no controle PID no ESP32. Sendo assim, o SP pode ser encontrado a partir da Equação 8, onde Y diz respeito às coordenadas no domínio da imagem e P_{MAX} ao número de pulsos no fim de curso da guia linear.

$$SP = \frac{(Y_{RECEBIDO} - Y_{MIN}) \cdot P_{MAX}}{Y_{MAX} - Y_{MIN}} \quad (8)$$

4.2. Desempenho do sistema

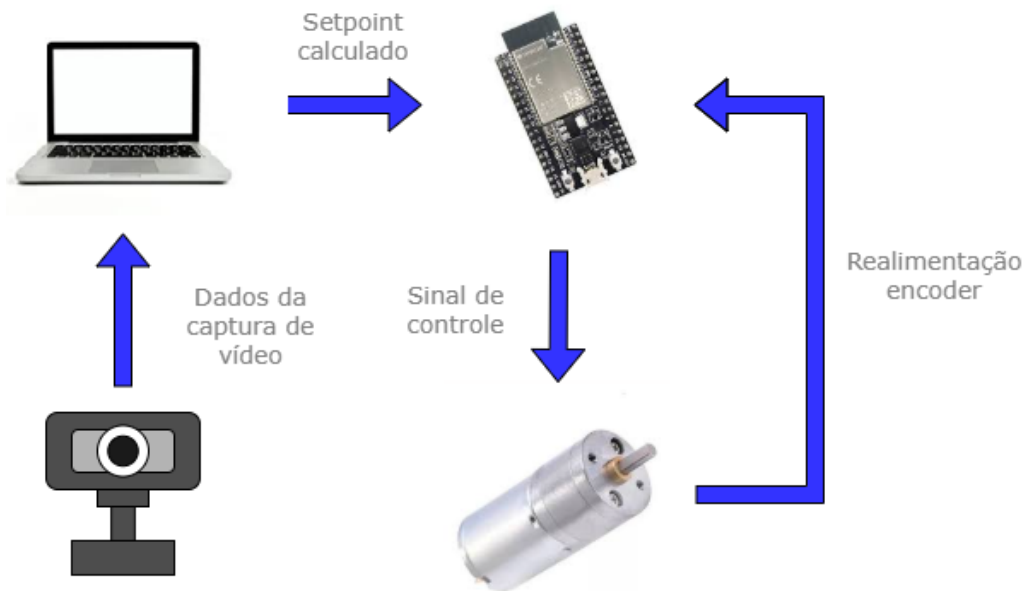
Uma vez finalizado o processo de integração entre o sistema computacional de previsão de trajetória e o sistema físico de controle de posição, foi iniciada a etapa de validação do projeto. Nessa parte, foram realizados vários lançamentos da bola em direção ao interceptador. Como critério para caracterizar os testes como sucesso ou falha, foi verificado em cada lançamento se o interceptador conseguiu se posicionar de forma a colidir com o projétil, reagindo ao movimento da bola no campo de colisão.

A partir do escopo do projeto, definiu-se a implementação do sistema de rastreamento e de previsão de trajetória em uma plataforma embarcada. Conforme abordado anteriormente, o *hardware* escolhido para tal função foi uma BeagleBone Black. Contudo, a execução dos algoritmos de visão desenvolvidos nos testes iniciais foi feita em um computador, devido ao seu maior poder de processamento, às ferramentas de desenvolvimento e ao ambiente mais flexível e estável. Essa estratégia permitiu iterar rapidamente, ajustar parâmetros e validar a lógica do algoritmo com maior agilidade. Uma vez que a solução está funcional e validada no ambiente de desenvolvimento, a transição para o sistema embarcado pode ser feita com foco nas particularidades da plataforma, reduzindo o tempo de desenvolvimento, facilitando a identificação de problemas e aumentando as chances de sucesso na implementação final do projeto embarcado.

Tendo isso em vista, os primeiros testes foram feitos com a câmera conectada ao computador, o qual, por sua vez, estava conectado ao ESP32 por meio da sua interface USB, com o objetivo de enviar o *setpoint* calculado ao microcontrolador. O diagrama do sistema com o fluxo de informações é apresentado na Figura 32. Percebe-se, portanto, que a mesma estrutura do projeto anteriormente descrita é respeitada por essa configuração, com a única diferença de que o papel da BBB está

sendo executado por uma máquina com maior poder computacional (16 GB de RAM, processador Intel Core i7 e placa de vídeo dedicada).

Figura 32 – Diagrama para testes iniciais.



Fonte: Autor (2025).

O interceptador reagiu com sucesso ao movimento do projétil em 100% dos primeiros dez lançamentos realizados. Contudo, com o passar do tempo, quando muitos testes (aproximadamente mais de quinze) foram feitos em sequência, o sistema aparentou acumular um erro expressivo relativo ao encoder, de forma que a realimentação do sistema começou a divergir consideravelmente da posição real do interceptador. Nesses cenários, se fez necessário reinicializar o microcontrolador para que a etapa de calibração, realizada no início do programa, fosse executada novamente, fazendo com que o funcionamento retornasse ao normal.

O grande aspecto positivo observado foi que o sistema conseguia obter um posicionamento mais ágil devido à parcela preditiva do algoritmo, que busca extrapolar a trajetória presente identificada para os momentos futuros, descartando uma resposta puramente reativa. Entretanto, observou-se que a trajetória calculada não teve uma precisão muito alta, principalmente devido ao fato de que a metodologia para o cálculo é estritamente geométrica. Por meio dela, fatores como atrito de rolamento, movimentos não lineares e colisões não ideais foram desconsiderados, acumulando parcelas de erro. Contudo, mesmo diante desse cenário, a readaptação

do sistema em tempo real faz com que geralmente as correções necessárias sejam feitas antes do projétil chegar ao eixo do interceptador.

Antes de transferir a responsabilidade de execução da rotina de rastreamento para a BeagleBone Black, foram feitos testes com bolas de outras cores e diâmetros maiores, conforme mostrado na Figura 33, com o objetivo de observar o comportamento do sistema em circunstâncias que não foram projetadas inicialmente. Sendo assim, observou-se que, a depender do padrão de iluminação do ambiente de teste, as esferas maiores também eram identificadas pelo sistema de segmentação HSV. Ou seja, especialmente quando as luzes do ambiente estavam integralmente acesas, as suas cores (laranja, branco e bege) capturadas pela câmera estavam no *range* de cores definido, possibilitando a identificação de todas.

Figura 33 – Bolas alternativas.



Fonte: Autor (2025).

Entretanto, para que as bolas de maior diâmetro (4 cm) pudessem ser rastreadas e interceptadas, foi necessário alterar os limites de área definidos previamente no sistema de visão computacional, visto que a área máxima para classificar os contornos como o projétil de interesse estava em um valor insuficiente para validar as novas esferas. Quando essa mudança foi feita, o sistema conseguiu interceptar com sucesso 83% dos lançamentos realizados. No caso em que área máxima não foi aumentada, verificou-se que o sistema conseguiu reagir isoladamente ao projétil menor, de diâmetro 3 cm, mesmo quando as outras bolas foram mandadas simultaneamente no campo de colisão.

Sendo assim, os resultados preliminares atingidos por meio dos testes no computador pessoal validaram o algoritmo de interceptação. Para finalizar a etapa de verificação de desempenho do projeto, conforme a proposta inicial, utilizou-se a BeagleBone Black para executar a rotina de *tracking* do projétil. Como esperado, devido ao menor poder computacional da placa, a velocidade do processo de previsão

de trajetória foi reduzida consideravelmente. Nesse cenário, para que o sistema conseguisse reagir em tempo hábil, os lançamentos realizados foram um pouco mais lentos do que os feitos anteriormente.

Dessa forma, disparando a bola com uma menor velocidade inicial, o sistema com funcionamento totalmente embarcado interceptou o projétil com sucesso em 75% das tentativas, dentre as quais a esfera contou com trajetórias diretas (sem desvios ou rebatimentos) e indiretas (com colisões nas barreiras laterais ou nas quinas). A grande razão por trás dos erros do sistema nesse cenário específico é a lentidão de resposta, visto que a BBB tem menor capacidade de processamento e demora mais para interpretar os frames da câmera e para realizar os cálculos necessários.

Durante alguns testes realizados com as luzes artificiais do ambiente apagadas, ou seja, utilizando apenas a iluminação natural proveniente das janelas do laboratório, o sistema não conseguiu detectar a trajetória do projétil. Consequentemente, o interceptador não reagiu ao movimento da bola. Esse comportamento reforça a dependência da segmentação baseada no espaço de cores HSV e evidencia a necessidade de condições de iluminação consistentes para o bom desempenho do projeto. Vale destacar que esses experimentos foram conduzidos no período da tarde, em um dia nublado, o que impossibilita uma avaliação objetiva e quantitativa do desempenho do sistema com as lâmpadas do laboratório apagadas em cenários com condições luminosas externas significativamente diferentes.

5. CONCLUSÃO

O campo da robótica autônoma vem se expandindo de forma significativa nas últimas décadas, impulsionado pelo avanço de tecnologias como a visão computacional, a inteligência artificial, os sistemas embarcados e os algoritmos de controle em tempo real. Esse crescimento tem ampliado as aplicações em áreas como logística, manufatura, segurança, medicina e entretenimento. Nesse cenário, o presente trabalho buscou desenvolver um sistema capaz de prever e interceptar a trajetória de um projétil em movimento, em um ambiente plano, com o auxílio de visão computacional e controle de posição embarcados, integrando conceitos fundamentais da Engenharia de Controle e Automação.

Utilizando visão computacional e regressão linear como base para a modelagem do movimento, o sistema proposto demonstrou considerável eficácia. Ao aplicar técnicas simples, porém robustas, como a análise de pontos centrais em sequência e o uso de regressões móveis, foi possível estimar a direção do deslocamento da bola de forma adaptativa e com boa responsividade. A incorporação de mecanismos como a fila FIFO, o fator de esquecimento e a reconfiguração do modelo após colisões, foi essencial para garantir a acurácia das previsões mesmo em cenários com mudanças de comportamento, como reflexões nas bordas ou alterações na dinâmica do projétil utilizado.

Foi necessário implementar uma lógica de projeção da trajetória, capaz de considerar colisões futuras com as barreiras que compõem o campo para prever o ponto de interceptação. Para tanto, o sistema avalia se a reta estimada intercepta diretamente o eixo do interceptador e, caso contrário, é realizado o cálculo da intersecção com as equações das bordas. Quando uma colisão é prevista, uma nova trajetória é gerada a partir dos princípios da física ótica, assumindo que o ângulo de incidência é igual ao ângulo de reflexão. Essa abordagem geométrica permitiu simular os rebotes da bola com base na dinâmica de espelhos planos, tratando o comportamento das barreiras superiores e inferiores com fidelidade razoável.

O envio da posição estimada ao sistema de interceptação, via comunicação serial com o ESP32, integrou os sistemas de visão e de atuação, possibilitando a operação em tempo real. Embora o modelo se baseie em simplificações, os resultados obtidos foram compatíveis com os objetivos propostos. A arquitetura modular e a

escolha por algoritmos computacionalmente leves tornam o sistema viável para aplicação em plataformas embarcadas com capacidade de processamento limitada, como a BeagleBone Black utilizada, mas com desempenho inferior. Além disso, as grandes limitações observadas foram relacionadas à sensibilidade à iluminação do ambiente e aos erros do *feedback* de rotação do motor.

O trabalho desenvolvido traz contribuições relevantes, especialmente no contexto de sistemas autônomos e inteligência computacional aplicada. A abordagem proposta pode ser estendida a diversas áreas que envolvem a interação de máquinas com o ambiente, como esportes, jogos e até em ambientes industriais, onde o controle de objetos em movimento é essencial. Além disso, ao integrar técnicas de visão computacional e controle de posição, o trabalho oferece um exemplo prático de como tecnologias emergentes podem ser utilizadas para melhorar a eficiência e a precisão em tarefas complexas. Os mesmos princípios podem ser adaptados para aplicações como robôs de resgate, veículos autônomos, ou sistemas de monitoração e intervenção em tempo real, trazendo benefícios na segurança, acessibilidade e automação.

Como perspectivas futuras, destacam-se a possibilidade de incorporar modelos mais avançados de predição de trajetória, como filtros de Kalman ou redes neurais. Além disso, é possível incorporar ao sistema um mecanismo de rebatimento do projétil, a partir da bobina solenoide utilizada no interceptador, e instalar um sistema de interceptação simétrico do outro lado do campo de colisão para permitir o confronto entre duas estratégias distintas. Como melhorias, pode-se trocar a BeagleBone Black por outro *hardware* com maior poder de processamento para melhorar a velocidade de resposta e para integrar as partes de visão e de controle de posição no mesmo dispositivo. Também é possível adicionar ao projeto um algoritmo de identificação dinâmica dos limites do campo de colisão bem como uma rotina periódica de calibração do motor.

REFERÊNCIAS BIBLIOGRÁFICAS

BACAC, Nikola *et al.* Comparison of different DC motor positioning control algorithms. In: 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2014.

BEAGLEBOARD, Foundation. BeagleBone Black. [S. l.], 2024. Disponível em: <https://docs.beagleboard.org/boards/beaglebone/black/>. Acesso em: 11 dez. 2024.

BLASCO, José *et al.* Development of a machine for the automatic sorting of pomegranate (*Punica granatum*) arils based on computer vision. Journal of food engineering, v. 90, n. 1, p. 27-34, 2009.

CHAPMAN, Stephen J. Fundamentos de máquinas elétricas. 5. ed. Porto Alegre: AMGH, 2013.

CHENG, Heng-Da *et al.* Color image segmentation: advances and prospects. Pattern recognition, v. 34, n. 12, p. 2259-2281, 2001.

CHUKWU, Ikegwu Anayo. Recently Emerging Trends in Embedded Computer Vision for Object Detection, Recognition, and Tracking. International Conference on Technological Solutions for Smart Economy | SmartEco. p. 1, 2024.

COX, E. P. A method of assigning numerical and percentage values to the degree of roundness of sand grains. Journal of paleontology, v. 1, n. 3, p. 179-183, 1927.

CURVELLO, André Márcio de Lima. Proposta para aceleração de desempenho de algoritmos de visão computacional em sistemas embarcados. Tese de Doutorado. Universidade de São Paulo, 2016.

DE MILANO, Danilo; HONORATO, Luciano Barrozo. Visão computacional. Faculdade de Tecnologia, Universidade Estadual de Campinas, 2010.

DIXIT, Aditya; CHIDAMBARAM, Ramesh Kumar; ALLAM, Zaheer. Safety and risk

analysis of autonomous vehicles using computer vision and neural networks. *Vehicles*, v. 3, n. 3, p. 595-617, 2021.

ELBALAOUI, Abderrahmane *et al.* Review of Color Image Segmentation. *International Journal of Advanced Computer Science and Application (Special Issue on Selected Papers from Third international symposium on Automatic Amazigh processing)*, 2013.

FONSECA, Thiago dos Santos da. Desenvolvimento e validação de métodos de sensoreamento visual aplicados a instrumentação de processos no contexto da indústria 4.0. 2017.

FRANKLIN, Gene F.; POWELL, J. David; EMAMI-NAEINI, Abbas. *Sistemas de controle para engenharia*. Bookman Editora, 2013.

GARCIA-LAMONT, Farid *et al.* Segmentation of images by color features: A survey. *Neurocomputing*, v. 292, p. 1-27, 2018.

HE, Nannan; QIAN, Ying; HUANG, Han-way. Experience of teaching embedded systems design with BeagleBone Black board. In: 2016 IEEE international conference on electro information technology (EIT). IEEE, 2016.

HOLLABAUGH, Craig. *Embedded Linux: hardware, software, and interfacing*. Addison-Wesley Professional, 2002.

HUNTER, Ian W.; HOLLERBACH, John M.; BALLANTYNE, John. A comparative analysis of actuator technologies for robotics. *Robotics Review*, v. 2, p. 299-342, 1991.

IBRAHIM, Dogan. *Microcontroller based applied digital control*. John Wiley, 2006.

JÄHNE, Bernd; HAUSSECKER, Horst; GEISSLER, Peter. *Handbook of computer vision and applications*. San Diego: Academic press, 1999.

KHAN, Asharul Islam; AL-HABSI, Salim. Machine learning in computer vision. *Procedia Computer Science*, v. 167, p. 1444-1451, 2020.

LOMBARDO, John. Embedded linux. Sams Publishing, 2001.

LOWE, David. The computer vision industry. [S. l.], 2015. Disponível em: <https://www.cs.ubc.ca/~lowe/vision.html>. Acesso em: 3 dez. 2024.

MOLLOY, Derek. Exploring BeagleBone: tools and techniques for building with embedded Linux. John Wiley & Sons, 2014.

NADESKI, Mark. Bringing machine learning to embedded systems. Texas Instruments, p. 1-5, 2019.

NEVES, Luiz Antônio Pereira; VIEIRA NETO, Hugo; GONZAGA, Adilson. Avanços em visão computacional. Omnipax, Curitiba, PR, v. 1, 2012.

NISE, Norman S. Engenharia de Sistemas de Controle. 8. ed. Rio de Janeiro: Grupo Editorial Nacional, 2023.

OPENCV. OpenCV. [S. l.], 2025. Disponível em: <https://opencv.org/>. Acesso em: 12 jan. 2025.

RAHMANIAR, Wahyu; HERNAWAN, Ari. Real-time human detection using deep learning on embedded platforms: A review. Journal of Robotics and Control (JRC), v. 2, n. 6, p. 462-468, 2021.

RANADEV, Manjunath B.; CHAKRASALI, R. L. Speed-Torque Characteristic of Dc Motor Fed by H-Bridge Converter. International Journal of Energy and Power, v. 3, n. 4, p. 49-53, 2014.

ROMANO, Vitor Ferreira; DUTRA, M. Introdução a robótica industrial. Robótica Industrial. São Paulo, Brasil. 2002.

ROSSOL, Lothar. Computer vision in industry. In: Robot Vision. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983.

RUDEK, Marcelo; COELHO, Leandro dos Santos; CANCEGLIERI JR, O. Visão computacional aplicada a sistemas produtivos: fundamentos e estudo de caso. XXI Encontro Nacional de Engenharia de Produção-2001, Salvador, 2001.

SAABITH, A. S.; FAREEZ, M. M. M.; VINOTHRAJ, T. Python current trend applications-an overview. International Journal of Advance Engineering and Research Development, v. 6, n. 10, 2019.

SAMIULLAH, Muhammad; IRFAN, Muhammad Zohaib; RAFIQUE, Abid. Microcontrollers: A Comprehensive Overview and Comparative Analysis of Diverse Types.

SIMÕES, Alexandre da Silva; COSTA, Anna Helena Reali; GARCIA, Cláudio. Regressão e predição de trajetórias: um comparativo das técnicas clássicas e sua aplicação para a visão computacional no domínio do futebol de robôs. Workshop de Computação, São José dos Campos, v. 1, p. 137-143, 2002.

SMITH, Grant Maloy. Medindo RPM, ângulo e velocidade usando sensores digitais, codificadores e contadores: Codificadores rotativos incrementais. DEWESoft: Blog, 2023. Disponível em: <https://dewesoft.com/pt/blog/medir-digital-codificador-e-contrasensores#codificadores-rotativos>. Acesso em: 5 mar. 2025.

SOUZA, Wellington Isac. Prevendo Trajetória da Bola de Basquete com Visão Computacional e Estatística. 1 vídeo (15 min.). YouTube, 26 nov. 2022. Publicado pelo canal Dev Ideias. Disponível em: https://www.youtube.com/watch?v=HvKLK_SeKns. Acesso em: 7 ago. 2024.

SZELISKI, Richard. Computer vision: algorithms and applications. Springer Nature, 2010.

UDENDHRAN, R. *et al.* Enhancing image processing architecture using deep learning for embedded vision systems. Microprocessors and Microsystems, v. 76, p. 103094, 2020.

VATANSEVER, Fahri; HACIISKENDEROGU, Emre. PID Tuning with Up-to-date Metaheuristic Algorithms. *Uludağ Üniversitesi Mühendislik Fakültesi Dergisi*, v. 27, n. 2, p. 573-584, 2022.

VIDMAR, Andreja; SOLINA, Franc. Reconstruction of superquadrics from 2-D contours. 1992.

VOULODIMOS, Athanasios *et al.* Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, v. 2018, n. 1, p. 7068349, 2018.

YILMAZ, Alper; JAVED, Omar; SHAH, Mubarak. Object tracking: A survey. *Acm computing surveys (CSUR)*, v. 38, n. 4, p. 13-es, 2006.

ZORZO, Avelino Francisco *et al.* Uso de Linux como Sistema Embarcado. In: III Workshop sobre Software Livre, 2002, Brasil. 2002.