CENTRO PAULA SOUZA
Tecnologia em Eletrônica Automotiva

Joelton da Silva Santana
Rodrigo Cavalcanti Alexandrino

DESENVOLVIMENTO DE UM CARRO AUTONOMO EM ESCALA 1:10

São Paulo
2025

Joelton da Silva Santana
Rodrigo Cavalcanti Alexandrino

DESENVOLVIMENTO DE UM CARRO AUTONOMO EM ESCALA 1:10

Trabalho de Conclusão de Curso Apresentado ao Curso de Tecnologia em Eletrônica Automotiva da Fatec Santo André, orientado pelo Prof. Me. Weslley Medeiros Torres, como requisito parcial para obtenção do título de Tecnólogo em Eletrônica Automotiva.

São Paulo
2025

FICHA CATALOGRÁFICA

S232d
Santana, Joelton da Silva
Desenvolvimento de um carro autônomo em escala 1:10 / Joelton da Silva Santana, Rodrigo Cavalcanti Alexandrino. - Santo André, 2025. – 96f: il.

Trabalho de Conclusão de Curso – FATEC Santo André.
Curso de Tecnologia em Eletrônica Automotiva, 2025.

Orientador: Prof. Me. Weslley Medeiros Torres

1. Eletrônica. 2. Veículo autônomo. 3. Desenvolvimento. 4. Tecnologia. 5. Software. 6. Programação. 7. Automação. 8. Plataforma. I. Alexandrino, Rodrigo Cavalcanti. II. Desenvolvimento de um carro autônomo em escala 1:10.

629.2

Fatec CENTRO PAULA SOUZA GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Santo André

LISTA DE PRESENÇA

Santo André, 06 de dezembro de 2025.

LISTA DE PRESENÇA REFERENTE À APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO COM O TEMA: **"DESENVOLVIMENTO DE UM CARRO AUTÔNOMO EM ESCALA 1:10"** DOS ALUNOS DO 6º SEMESTRE DESTA U.E.

BANCA

PRESIDENTE:
PROF. WESLLEY MEDEIROS TORRES _____

MEMBROS:
PROF. PAULO TETSUO HOASHI _____

PROF. EDSON CAORU KITANI _____

ALUNOS:

JOELTON DA SILVA SANTANA _____

RODRIGO CAVALCANTI ALEXANDRINO _____

Dedicamos a todos que de alguma maneira contribuíram para o desenvolvimento desse trabalho.

AGRADECIMENTOS

Gostaríamos de expressar nossa mais profunda gratidão a todas as pessoas que, de maneira direta ou indireta, contribuíram para a realização deste trabalho.

Eu Joelton, agradeço a minha esposa, Agata Cristina da Silva Sacramento, a quem devo um agradecimento especial. Seu amor, paciência e apoio incondicional foram fundamentais para que eu prosseguisse firme em cada etapa desta jornada. Seu incentivo diário foi o alicerce que me manteve focado e determinado, e sou imensamente grato por sua presença constante ao meu lado.

Eu Rodrigo, agradeço a minha noiva, Viviane Salute Angelucci, que foi peça fundamental em toda esta caminhada. Desde o início, foi ela quem me motivou e acompanhou em cada desafio. Sua presença, apoio e palavras de encorajamento foram fundamentais para a conclusão deste trabalho, este trabalho é também fruto do seu amor e dedicação.

Agradecemos ao professor orientador Me. Weslley Medeiros Torres, por sua orientação, conhecimento e paciência ao longo deste processo. Sua contribuição acadêmica foi imprescindível, e sou muito grato pelo apoio contínuo, pela clareza nas orientações e pelo comprometimento em me ajudar a aprimorar meu trabalho.

Ao professor Fernando Garup Dalbo, pela dedicação nas aulas e pelo suporte nas etapas do trabalho. Seu conhecimento e experiência nos ajudaram a construir as bases sólidas para a conclusão deste projeto.

Aos nossos colegas, que estiveram conosco nos momentos de desafios e de celebrações. A amizade e o apoio de vocês foram fundamentais para a realização desse trabalho.

E, especialmente, ao nosso amigo Weslei Alves Silva. Mais do que um apoio, ele foi o verdadeiro mentor durante toda essa jornada. Ele nos ensinou e corrigiu quando necessário e nos desafiou a ir além dos nossos próprios limites. Sua amizade e generosidade com o conhecimento foram indispensáveis para que pudéssemos concluir este trabalho da melhor maneira possível. Não seria possível sem a sua colaboração incansável.

Por fim, ao Marcelo Alexander Kai, que teve um papel fundamental ao ajudar a financiar este projeto. Seu apoio financeiro permitiu que a ideia se concretizasse de forma mais ampla e com a qualidade necessária. Agradeço imensamente por acreditar no projeto e por proporcionar os meios para que ele fosse possível.

Este trabalho é a soma do apoio, da amizade e do conhecimento compartilhado por todos vocês. Não seria possível chegar até aqui sem a contribuição de cada um. Muito obrigado!

“Tornou-se aterradoramente claro que a nossa tecnologia ultrapassou a nossa humanidade.”

Albert Einstein

RESUMO

Este trabalho visa ao desenvolvimento de um veículo com propulsão e direção autônoma em escala 1:10, utilizando uma plataforma acessível com foco na aplicação de técnicas de visão computacional. O veículo é controlado por uma *Raspberry Pi*, que atua como unidade de controle principal, responsável pelo processamento de dados e pela tomada de decisões. Essa placa realiza o controle do servo de direção, bem como do motor de propulsão que movimenta o veículo. A captura de imagens é feita por uma câmera acoplada ao veículo, que alimenta o modelo de visão computacional, permitindo a condução autônoma em trajetos predefinidos. Este projeto integra conceitos de eletrônica, programação e automação, demonstrando a viabilidade de sistemas autônomos em pequena escala. Além disso, contribui para a disseminação do conhecimento sobre automação veicular, incentivando inovações e estudos futuros na área.

Palavras chaves: Veiculo autônomo. *Raspberry Pi*. Visão computacional. Automação veicular. Programação.

ABSTRACT

This work aims to develop a 1:10 scale vehicle with autonomous propulsion and steering, using an accessible platform focused on the application of computer vision techniques. The vehicle is controlled by a *Raspberry Pi*, which acts as the main control unit, responsible for data processing and decision-making. This board manages the steering servo as well as the propulsion motor that drives the vehicle. Image capture is performed by a camera mounted on the vehicle, which feeds a computer vision model, enabling autonomous navigation on predefined routes. This project integrates concepts from electronics, programming, and automation, demonstrating the feasibility of small-scale autonomous systems. Furthermore, it contributes to the dissemination of knowledge in vehicle automation, encouraging innovation and future research in the field.

Keywords**:** Autonomous vehicle. *Raspberry Pi*. Computer vision. Vehicle automation. Programming.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIAÇÃO E SIGLAS

A2	*Application Performance Class 2* (Classe de Desempenho de Aplicação 2)
ACC	*Adaptive Cruise Control* (Controle de Cruzeiro Adaptativo)
ACPS	*Automotive Cyber-Physical System* (Sistema Ciberfísico Automotivo)
ADAS	*Advanced Driver-Assistance System* (Sistema Avançado de Assistência ao Motorista)
ARM64	*Architecture 64-bit* (Arquitetura 64 bits)
BGR	*Blue Green Red* (Azul Verde Vermelho)
CLAHE	*Contrast Limited Adaptive Histogram Equalization* (Equalização de Histograma Adaptativo com Contraste Limitado)
CNN	*Convolutional Neural Network* (Rede Neural Convolucional)
ECU	*Electronic Control Unit* (Unidade de Controle Eletrônico)
ESC	*Electronic Speed Controller* (Controle Eletrônico de Velocidade)
EWMA	*Exponentially Weighted Moving Average* (Média Móvel Ponderada Exponencialmente)
FSD	*Full Self-Driving* (Direção Autônoma Total)
GPIO	*General Purpose Input/Output* (Entrada/Saída de Propósito Geral)
HSV	*Hue Saturation Value* (Matiz, Saturação, Valor)
I²C	*Inter-Integrated Circuit* (Circuito Interintegrado)
LiDAR	*Light Detection and Ranging* (Detecção e Localização de Luz)
MCP	*Model Predictive Control* (Controle Preditivo de Modelo)
MJPEG	*Motion JPEG* (JPEG em Movimento)
ODD	*Operational Design Domain* (Domínio de Design Operacional)
PID	*Proporcional, Integral, Derivativo*
PWM	*Pulse Width Modulation* (Modulação por Largura de Pulso)
RC	*Remote Control* (Controle Remoto)
RGB-D	*Red Green Blue – Depth* (Vermelho Verde Azul - Profundidade)

ROI *Region of Interest* (Região de Interesse)

SAE *Society of Automotive Engineers* (Sociedade de Engenharia Automotiva)

SLAM *Simultaneous Localization and Mapping* (Localização e Mapeamento Simultâneos)

SSD *Single Shot MultiBox Detector* (Detector MultiBox de Disparo Único)

UART *Universal Asynchronous Receiver-Transmitter* (Receptor-Transmissor Assíncrono Universal)

V2I *Vehicle-to-infrastructure* (Veículo para Infraestrutura)

V2V *Vehicle-to-vehicle* (Veículo para Veículo)

YOLO *You Only Look Once* (Só se olha uma vez)

Sumario

1 INTRODUÇÃO

A indústria automobilística tem passado por transformações significativas ao longo das últimas décadas, especialmente com a incorporação de tecnologias embarcadas que tornam os veículos cada vez mais inteligentes. A automação veicular surge como um dos avanços mais promissores nesse cenário, impulsionada pelo desenvolvimento de sensores, sistemas de controle, visão computacional e inteligência artificial.

Essa evolução tem permitido o surgimento dos chamados veículos autônomos, capazes de tomar decisões e se locomover com pouca ou nenhuma intervenção humana. A adoção crescente dessas tecnologias busca não apenas melhorar a experiência de condução, mas também aumentar a segurança, reduzir o número de acidentes e tornar o trânsito mais eficiente.

Neste contexto, o presente trabalho propõe o desenvolvimento de um veículo autônomo em escala reduzida, com o objetivo de explorar, de forma prática e acessível, os princípios que regem a automação veicular.

A Figura 1 apresenta a linha do tempo da evolução da indústria veicular, destacando marcos tecnológicos desde o primeiro automóvel até o início da automação veicular.

Figura 1 - Linha do tempo da evolução da industrial veicular

Fonte: Adaptado de SAE INTERNATIONAL, 2025

A Figura 2 apresenta o CaRINA 2, veículo autônomo desenvolvido pela USP São Carlos, como exemplo dos avanços recentes em sistemas de condução autônoma, que emprega sensores e algoritmos de inteligência artificial para realizar a navegação.

Figura 2 - Veículo autônomo da USP São Carlos

Fonte: USP São Carlos, 2015

1.1 Objetivo

O objetivo deste trabalho é desenvolver um veículo de propulsão e direção autônoma em escala 1:10, utilizando como base para a aplicação de técnicas de visão computacional e automação veicular. A proposta envolve a pesquisa e compreensão do funcionamento de ferramentas computacionais, com foco na correta programação, configuração, integração e calibração dos componentes eletrônicos essenciais, como o servo motor, câmera, controlador de Velocidade Eletrônico (ESC) e Unidade de Controle Eletrônica (ECU). Dessa forma, buscamos assegurar uma comunicação eficiente entre *hardware* e *software*, a fim de alcançar uma navegação autônoma estável e funcional.

1.2 Motivação

O desenvolvimento de sistemas autônomos tem crescido rapidamente e representa uma tendência importante para o futuro da mobilidade. Este projeto busca explorar essa área por meio da criação de um veículo autônomo em escala 1:10, utilizando ferramentas acessíveis a comunidade. O que nos motiva, é aplicar na prática,

conceitos de eletrônica, programação e visão computacional, facilitando o aprendizado e a experimentação de novas tecnologias que são usadas em veículos reais.

1.3 Conteúdo

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta a fundamentação teórica, abordando todos os conceitos utilizados no projeto para garantir o completo entendimento do leitor; o Capítulo 3 detalha a metodologia empregada, explicando os procedimentos e técnicas adotadas para o desenvolvimento do sistema; o Capítulo 4 exibe os resultados obtidos nos testes práticos, com uma análise crítica dos dados coletados; o Capítulo 5 traz as conclusões finais, sintetizando as descobertas e contribuições deste estudo; e finalmente, o Capítulo 6 propõe novas perspectivas e desafios para futuras pesquisas na área.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a base teórica necessária para o leitor entender os conceitos e tecnologias utilizadas no desenvolvimento do sistema veicular autônomo. Serão abordados a seguir, os principais elementos que sustentam o projeto, com a explicação do funcionamento de cada tecnologia essencial para a construção e operação do veículo autônomo.

2.1 Veículo Autônomo

Conforme o material desenvolvido e disponibilizado por Medeiros (2024), um veículo autônomo pode ser definido como um automóvel capaz de se movimentar de forma independente. Estruturado pela integração de sensores e atuadores com componentes elétricos e mecânicos. Esse veículo é capaz de interagir com o ambiente ao seu redor, utilizando algoritmos e processamento de dados para guiar sua locomoção.

A autonomia veicular, ainda que parcial, já é uma realidade. Esse avanço tecnológico no sistema de transporte automotivo demonstra que, por meio dessa inovação, é possível melhorar a qualidade do tráfego, aumentar a segurança e proporcionar maior confiabilidade nos deslocamentos, conforme aponta Silva (2022).

De acordo com Molina (2018), os veículos estão evoluindo de um sistema que se baseia nas leis da mecânica, física e química para um sistema mais sofisticado, denominado *Automotive Cyber-Physical System* (ACPS), que pode ser definido pela integração de sistemas físicos e digitais que interagem em tempo real, permitindo o controle e a otimização do comportamento físico do veículo por meio de sensores, atuadores e algoritmos de controle.

Esse contexto evidencia que os veículos estão se tornando cada vez mais capazes de desempenhar funções que, tradicionalmente, eram realizadas pelos humanos. Nesse sentido, Medeiros (2024) destaca um conjunto de capacidades essenciais para o

comportamento autônomo de um veículo, que devem estar presentes para garantir sua operação eficiente e segura. Entre essas capacidades, podemos citar:

- Percepção da posição atual na trajetória;
- Monitoramento do Estado do veículo;
- Compreensão das Regras de trânsito;
- Reconhecimento de sinalização semafórica;
- Controle e comportamento da direção;
- Controle e comportamento da velocidade;
- Capacidade de captar informações sobre o ambiente;
- Tomada de decisão em tempo real.

A Figura 3 ilustra um veículo autônomo da Waymo, uma das empresas pioneiras no desenvolvimento de tecnologias de direção autônoma. Esse exemplo visual reforça a aplicação prática das capacidades mencionadas por Medeiros (2024), demonstrando como esses sistemas são integrados em veículos reais para permitir uma condução segura, eficiente e sem intervenção humana direta. A Figura 3 evidencia a presença de sensores, câmeras e outros dispositivos embarcados que possibilitam ao carro perceber o ambiente, interpretar regras de trânsito e tomar decisões em tempo real.

Figura 3 - Veículo Autônomo da Waymo

Fonte: CNBC, 2024

2.1.1 Classificação dos níveis de automação

Desde 2014, a Sociedade de Engenharia Automotiva (SAE 2021), elaborou uma classificação para os níveis de automação de veículos, apresentada no documento denominado SAE J3016. Esse material descreve os sistemas de assistência e automação da direção veicular. A categorização, conhecida como Níveis de Automação SAE, define seis estágios de automação, do Nível 0 (sem automação) ao Nível 5 (automação completa), aplicando-se a veículos motorizados em ambientes rodoviários, a seguir está a definição básica para cada nível estipulado pela SAE 2021.

Nível 0

Todo o controle do veículo é responsabilidade exclusiva do condutor humano. Embora o veículo possa contar com sistemas de assistência, como alertas de colisão ou frenagem automática de emergência, essas tecnologias não assumem o controle de forma contínua, então não são caracterizados como autonomia veicular.

Nível 1

O sistema é capaz de auxiliar o motorista na condução, como frenagem, aceleração ou direção. Um exemplo típico é o Controle de Cruzeiro Adaptativo (ACC). Apesar da assistência, é necessário o motorista permanecer responsável pela operação do veículo durante todo o percurso do veículo.

Nível 2

O sistema é capaz de controlar simultaneamente a direção e a velocidade do veículo sob determinadas condições, por exemplo: rodovias com faixas claras e bem demarcadas e clima seco com boa visibilidade, ainda assim, o motorista deve estar pronto para intervir na autonomia veicular e assumir o controle.

Nível 3

O Veículo pode gerenciar todas as funções de direção e velocidade, monitorando o ambiente e verificando a necessidade de intervenção humana na condução do veículo, nessa etapa o veículo entende suas limitações e solicita intervenção humana quando necessário.

Nível 4

O veículo é capaz de realizar todas as funções de direção de forma autônoma em determinadas condições denominadas Domínio de Design Operacional (ODD). Se ocorrer um imprevisto durante o percurso, o sistema é capaz de lidar com a situação sem a necessidade de intervenção imediata do motorista.

Nível 5

O veículo é completamente autônomo em todas as condições. Não há necessidade de volante, pedais ou qualquer intervenção humana, permitindo que os ocupantes sejam apenas passageiros em todas as circunstâncias.

A tabela 1 apresenta um resumo dos níveis de automação veicular definidos pela SAE, destacando as funções do sistema e o papel do condutor em cada nível:

Tabela 1 - Classificação dos Níveis de Automação Veicular

NÍVEL	NOME	CONTROLE DO VEÍCULO	RESPONSABILIDADE PRINCIPAL
0	Sem Automação	Humano	Humano
1	Assistência ao Motorista	Sistema auxilia uma função	Humano
2	Automação Parcial	Sistema controla direção e velocidade	Humano
3	Automação Condicional	Sistema controla tarefas especificas	Sistema (com supervisão humana)
4	Alta Automação	Sistema é capaz de controla todas as tarefas	Sistema
5	Automação Total	Sistema controla todo o veículo	Sistema

Fonte: Adaptado do SAE J3016, 2025

Atualmente, nenhuma empresa foi capaz de desenvolver um Veículo Autônomo Nível 5, no entanto várias empresas, estão desenvolvendo e testando veículos com alto grau de automação:

- **Waymo**: A empresa, subsidiária da *Alphabet*, oferece serviços de táxi autônomo em áreas limitadas dos Estados Unidos, utilizando veículos que se aproximam do Nível 4 de automação.
- **Cruise**: Subsidiária da *General Motors*, realiza testes de veículos autônomos em ambientes urbanos, visando alcançar níveis altos de automação.
- **Tesla**: Oferece o sistema *Full Self-Driving* (FSD), que atualmente opera no Nível 2 de automação, exigindo supervisão constante do motorista.

2.1.2 Domínio de Design Operacional

De acordo com SAE o ODD, refere-se ao conjunto específico de condições sob as quais um sistema de condução automatizada é projetado para operar com segurança. Essas condições englobam fatores ambientais, geográficos, temporais e características específicas das vias e do tráfego.

De acordo com a definição formal:

> As condições operacionais sob as quais um determinado sistema de automação de condução, ou uma de suas funcionalidades, é projetado para funcionar, incluindo, mas não se limitando a restrições ambientais, geográficas, de horário e/ou a presença ou ausência de certas características de tráfego ou rodovia. (SAE J3016, 2021).

A definição precisa do ODD é essencial para assegurar que os sistemas de condução automatizada operem exclusivamente dentro dos cenários para os quais foram concebidos. Essa delimitação contribui significativamente para a segurança dos ocupantes, dos demais usuários da via e para a confiabilidade do sistema. Por exemplo, um veículo autônomo pode ter seu funcionamento restrito a rodovias com sinalização horizontal bem definida, fluxo de tráfego moderado e condições climáticas estáveis, evitando contextos para os quais seu sistema não foi projetado.

Além disso, o ODD tem papel central nos processos de validação, testes e certificação de veículos autônomos, servindo como referência tanto para fabricantes quanto para órgãos reguladores ao estabelecer os limites operacionais seguros de cada sistema automatizado.

2.1.3 Sistemas Ciberfisicos Automotivos

Os ACPS representam uma evolução na arquitetura dos veículos modernos, integrando de forma única o mundo físico e o digital. Esses sistemas combinam sensores, atuadores, controladores eletrônicos e *software* embarcado para monitorar, processar e responder em tempo real aos estímulos do ambiente e do próprio veículo.

A Figura 4 ilustra de forma conceitual a integração e a interdependência entre os componentes físicos e computacionais que compõem um sistema ciberfísico, destacando a interação dinâmica entre os domínios da computação, comunicação e controle, os quais são fundamentais para o processamento e a troca de informações dentro de sistemas complexos e inteligentes.

Figura 4 – Representação de um sistema ciberfísico

Fonte: E-Aware, 2016

Segundo Nousias (2021), os ACPS são fundamentais para o desenvolvimento de veículos autônomos, pois viabilizam a coordenação entre múltiplos subsistemas como direção, frenagem, navegação e percepção de maneira segura e eficiente. A capacidade desses sistemas de operar com respostas rápidas e inteligentes é o que possibilita níveis elevados de automação veicular.

Além disso, os ACPS permitem a implementação de algoritmos avançados de aprendizado profundo para o entendimento eficiente de imagens, essencial para a percepção do ambiente em veículos autônomos. Isso inclui a detecção e classificação de objetos, reconhecimento de sinais de trânsito e análise de comportamento de pedestres, contribuindo para a tomada de decisões em tempo real.

A integração dos ACPS com tecnologias de comunicação veicular, como a comunicação *vehicle-to-vehicle* (V2V) e *vehicle-to-infrastructure* (V2I), amplia ainda mais a capacidade dos veículos de interagirem com o ambiente e outros usuários da via, promovendo uma mobilidade mais segura e eficiente.

2.2 Visão Computacional no Veículo Autônomo

A visão computacional desempenha um papel essencial no funcionamento dos veículos autônomos, especialmente na etapa de percepção do ambiente. Essa tecnologia é responsável por interpretar as imagens captadas por sensores embarcados, como câmeras, permitindo que o veículo reconheça e compreenda o que está ao seu redor. Segundo Paula (2015), a visão computacional possibilita a detecção precisa de objetos, o reconhecimento de sinais de trânsito, faixas de rodagem, pedestres e outros elementos importantes para a navegação.

Essa capacidade é fundamental para que o veículo consiga tomar decisões em tempo real, como reduzir a velocidade ao identificar um obstáculo ou mudar de faixa com segurança. Grigoriou (2021) destaca que a visão computacional é uma das tecnologias-chave para a condução autônoma, pois permite que o sistema compreenda o ambiente e reaja de forma segura e eficiente.

Combinada a algoritmos de inteligência artificial e aprendizado profundo, a visão computacional aumenta significativamente a autonomia e a segurança dos veículos, tornando-os mais confiáveis em diferentes situações de tráfego e condições ambientais.

A Figura 5 apresenta uma representação simplificada do processo pelo qual o veículo autônomo percebe e interpreta o ambiente ao seu redor, possibilitando uma condução segura e precisa.

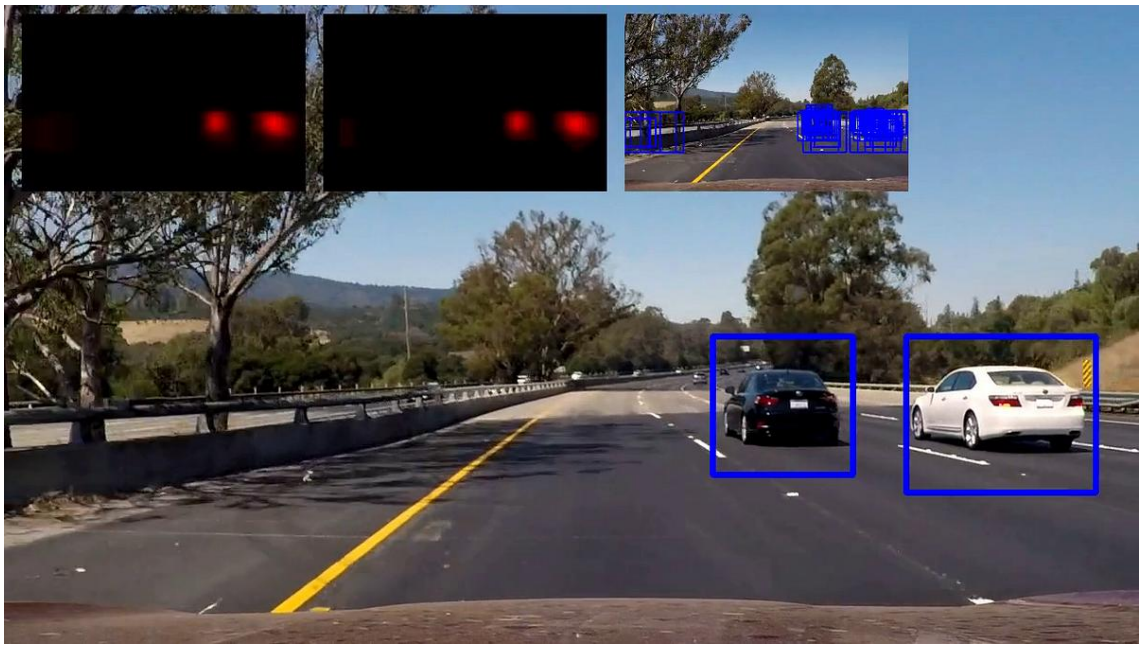
Além disso, é importante destacar que a visão computacional não apenas interpreta o ambiente, mas também fornece dados estruturados que servem de entrada para diferentes módulos do sistema autônomo, como planejamento de trajetória, controle de direção e prevenção de colisões. A precisão dessas informações influencia diretamente

a capacidade do veículo de manter-se na faixa correta, identificar obstáculos com antecedência e reagir adequadamente às mudanças do cenário.

Outro ponto relevante é que a visão computacional moderna está fortemente associada a técnicas de inteligência artificial, permitindo que modelos de detecção e classificação funcionem mesmo em condições adversas, como variações de iluminação, sombras, desgaste da pista ou presença de objetos inesperados. Essa robustez é essencial para garantir a confiabilidade do sistema em ambientes reais.

Assim, a visão computacional torna-se um dos principais pilares da condução autônoma, integrando informações visuais e transformando-as em conhecimento acionável para a tomada de decisões. A figura 5 apresenta um exemplo simplificado desse processo de percepção, ilustrando como a câmera identifica elementos importantes do ambiente rodoviário.

Figura 5 – Visão computacional automotivo



Fonte: Medium, 2017

2.2.1 Câmeras

As câmeras são sensores passivos que capturam imagens do ambiente e são um dos componentes principais em veículos autônomos. Paula (2015) descreve, que os veículos podem utilizar câmeras monoculares, estéreo e RGB-D, fornecendo informações visuais vitais para a identificação de obstáculos e a detecção de faixas de rodagem.

Esse tipo de sensor é indispensável para a funcionalidade do sistema de navegação autônoma, pois garante que o veículo tenha uma visão contínua e precisa de

seu entorno. Grigoriou (2021) também destacam que a integração de câmeras com outros sensores permite ao veículo realizar uma percepção mais robusta e confiável em tempo real.

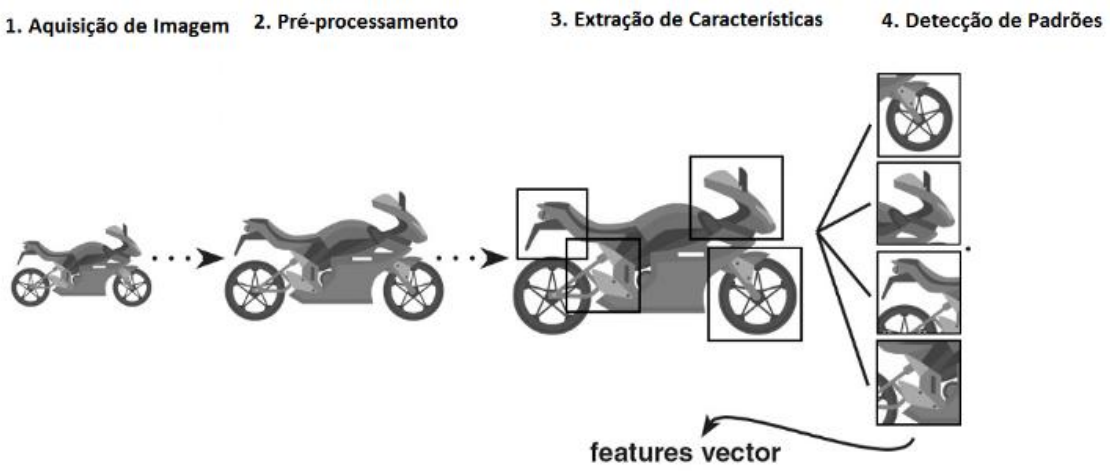
2.2.2 Processamento de imagem para navegação e detecção de obstáculos

O processamento de imagens é um dos principais desafios e contribuições para a navegação autônoma, conforme abordado por Paula (2015). Técnicas como segmentação, detecção de bordas e reconhecimento de padrões são essenciais para a análise e compreensão do ambiente ao redor.

Tais abordagens permitem que o sistema identifique e classifique objetos como outros veículos, pedestres e sinais de trânsito. Grigoriou (2021) também reforça a importância do processamento de imagens para garantir a segurança na condução autônoma, permitindo ao sistema reagir rapidamente a obstáculos e variações no cenário.

A Figura 6 apresenta o processamento de uma imagem, evidenciando as etapas de análise e interpretação dos dados visuais.

Figura 6 – Processamento de imagem



Fonte: Medium, 2020

2.2.3 Algoritmos comuns de visão computacional aplicados a carros autônomos

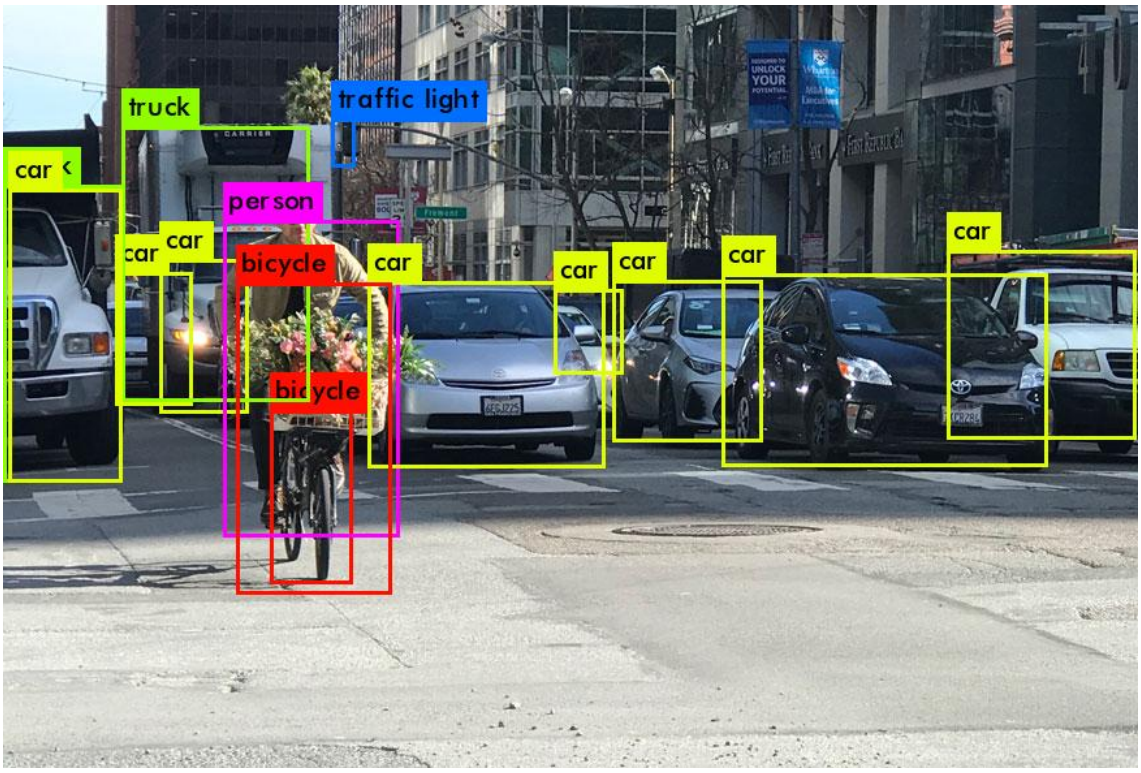
A aplicação de algoritmos de visão computacional em veículos autônomos é um dos pilares para a percepção do ambiente e tomada de decisões em tempo real. Entre os algoritmos mais utilizados, destacam-se aqueles baseados em aprendizado profundo, especialmente as redes neurais convolucionais (CNNs), que têm revolucionado a forma como os sistemas autônomos interpretam imagens.

Segundo Grigoriou (2021), as CNNs são amplamente empregadas para tarefas de detecção, classificação e reconhecimento de objetos em tempo real, possibilitando respostas rápidas e precisas diante de diferentes estímulos visuais. Essa capacidade é essencial para garantir a segurança e eficiência da navegação autônoma, sobretudo em ambientes urbanos, onde o cenário é dinâmico e repleto de obstáculos.

Dentre os modelos mais destacados, o *You Only Look Once* (YOLO) tem se consolidado como uma das abordagens mais eficazes para a detecção de objetos em tempo real. De acordo com Paula (2015) e com a análise apresentada em *Applications of Computer Vision in Autonomous Vehicles* (2023), o YOLO oferece uma combinação entre velocidade e precisão, sendo capaz de processar imagens e localizar múltiplos objetos em uma única varredura da imagem, o que o torna especialmente adequado para aplicações embarcadas com recursos computacionais limitados, como é o caso de muitos protótipos de veículos autônomos.

A Figura 7 apresenta o modelo YOLOv3, utilizada em veículos autônomos para identificar e localizar elementos no ambiente em tempo real.

Figura 7 – Modelo YOLOv3



Fonte: Medium, 2021

Assim, a adoção de algoritmos robustos de visão computacional, como o YOLO, representa um avanço significativo para o desenvolvimento de carros autônomos, viabilizando uma percepção confiável do ambiente e promovendo decisões mais assertivas por parte do sistema de controle veicular.

2.2.4 Sensores e atuadores no carro autônomo

Os sensores e atuadores são componentes fundamentais para o funcionamento de um carro autônomo. Os sensores permitem que o veículo visualize o ambiente ao seu redor, coletando informações como distância de obstáculos, faixas da pista, sinais de trânsito e outros veículos. Já os atuadores são responsáveis por transformar essas informações em ações, controlando a direção, aceleração e frenagem do carro. A seguir, vamos entender o papel de cada tipo de sensor e atuador e como eles trabalham juntos para tornar a condução autônoma possível.

2.2.5 Sensores ultrassônicos

Os sensores ultrassônicos são dispositivos fundamentais na arquitetura sensorial de veículos autônomos, especialmente em funções que requerem precisão na detecção de obstáculos próximos. Esses sensores funcionam por meio da emissão de ondas sonoras de alta frequência (ultrassom), que, ao colidirem com um objeto, retornam ao emissor. Com base no tempo de ida e volta do sinal, o sistema calcula a distância exata até o objeto detectado, fornecendo dados em tempo real ao sistema de controle do veículo.

Couto (2012) ressalta que essa tecnologia é especialmente eficaz em baixas velocidades, como nas manobras de estacionamento, manobras em garagens e em ambientes urbanos com espaço restrito. Nessas condições, a precisão é vital para garantir a segurança do veículo e de seu entorno, evitando colisões com obstáculos que, por vezes, não são visíveis ao condutor ou aos sensores ópticos.

Além da segurança, a simplicidade de operação, o baixo custo e a confiabilidade tornam os sensores ultrassônicos uma solução muito empregada em sistemas embarcados automotivos, tanto em carros de passeio quanto em protótipos de veículos autônomos em escala reduzida. Eles são frequentemente utilizados em conjunto com algoritmos de controle que possibilitam reações rápidas diante da presença de objetos próximos, funcionando como uma camada adicional de segurança.

Em contrapartida, para a detecção de objetos em maiores distâncias ou com maior nível de detalhe, tecnologias mais avançadas, como o LiDAR (*Light Detection and Ranging*), entram em cena. O LiDAR utiliza pulsos de laser para mapear o ambiente em três dimensões com alta precisão e riqueza de detalhes. Ele é capaz de gerar nuvens de pontos tridimensionais do entorno, o que o torna indispensável em ambientes complexos, como centros urbanos, onde a quantidade e diversidade de obstáculos exige uma percepção espacial refinada.

A imagem a seguir ilustra um sensor ultrassônico automotivo típico, geralmente instalado nos para-choques dianteiro e traseiro dos veículos. Sua função principal é identificar a presença de objetos a curta distância, informando o sistema de controle sobre a necessidade de frear, reduzir a velocidade ou ajustar a trajetória durante manobras. Como visto, o sensor atua como uma extensão sensorial do veículo, promovendo um controle mais preciso em situações de baixa velocidade e espaço reduzido.

Assim, os sensores ultrassônicos cumprem um papel estratégico no conjunto sensorial de um veículo autônomo. A Figura 8 apresenta esse tipo de sensor, responsável por fornecer dados essenciais para a condução segura. Quando integrados a outros sensores, como câmeras, radares e LiDARs, contribuem para uma percepção mais completa e confiável do ambiente, permitindo que o veículo navegue com segurança mesmo em situações desafiadoras.

Figura 8 - Sensor ultrassônico automotivo

Fonte: Mobiauto, 2024

2.2.6 Câmeras

As câmeras desempenham um papel central na percepção dos veículos autônomos e são classificadas em monoculares, estéreo e RGB-D, dependendo da necessidade do sistema. Couto (2012) explica que a configuração dessas câmeras varia de acordo com a aplicação específica, como a detecção de faixas de rodagem, o reconhecimento de sinais de trânsito ou o monitoramento do entorno.

A combinação de diferentes tipos de câmeras pode melhorar substancialmente a percepção do ambiente, aumentando a eficácia do sistema. De acordo com *Image Analysis in Autonomous Vehicles* (2024), as câmeras RGB-D, por exemplo, oferecem a vantagem de capturar informações de profundidade além das imagens tradicionais, complementando a percepção em 3D.

2.2.7 Atuadores

Os atuadores são responsáveis por converter os sinais de controle em movimentos físicos, sendo fundamentais para a operação de veículos autônomos. Couto (2012) destaca que os motores elétricos são utilizados para a propulsão do veículo, enquanto os sistemas de direção eletrônica controlam o ângulo das rodas para garantir que o veículo se desloque de maneira precisa e segura. A atuação desses sistemas de forma eficiente e precisa é crucial para a execução de manobras autônomas de maneira segura e sem falhas.

2.2.8 Controlador eletrônico de velocidade e seu papel no controle do motor

O ESC tem um papel fundamental no controle dos motores elétricos dos veículos autônomos, regulando sua velocidade ao interpretar sinais de controle e ajustar a potência fornecida aos motores. Couto (2012) explica que o ESC é essencial para manter a velocidade desejada, permitindo ao veículo responder de forma eficiente às mudanças nas condições de condução e mantendo a estabilidade durante as operações autônomas.

2.2.9 Servo motores

Os servomotores são componentes críticos em veículos autônomos, responsáveis por controlar partes do veículo que requerem movimentos precisos, como a direção e a aceleração. Couto (2012) enfatiza que esses motores recebem sinais de controle e ajustam sua posição de acordo com as instruções do sistema de navegação. Isso permite que o veículo execute movimentos rápidos e precisos, essenciais para manobras complexas e para a adaptação constante a um ambiente dinâmico.

2.3 Algoritmos de controle e navegação

Para que um carro autônomo se mova de forma segura e eficiente, ele precisa de algoritmos capazes de tomar decisões em tempo real. Os algoritmos de controle e navegação são responsáveis por interpretar os dados dos sensores e planejar os movimentos do veículo, como manter-se na pista, fazer curvas e evitar obstáculos.

2.3.1 Técnicas de controle de movimento para carros autônomos

O controle de movimento de veículos autônomos envolve o ajuste preciso da velocidade e direção para seguir uma trajetória específica. Dias (2016) destaca a importância das técnicas de controle, como o controle Proporcional, Integral, Derivativo (PID), que são amplamente utilizadas devido à sua simplicidade e eficácia, especialmente em sistemas lineares. *Segundo Design and Implementation of PID-Based Steering*

Control for 1/10-Scale Autonomous Vehicle (2021), o controle PID é uma escolha popular para ajustar a direção e a velocidade de veículos em escala reduzida, validando sua aplicabilidade em modelos autônomos. Essa técnica permite a manutenção de uma resposta equilibrada e estável, crucial para garantir que o veículo siga a trajetória desejada com alta precisão.

2.3.2 Controle PID

O controlador PID ajusta a saída do sistema com base na diferença entre o valor desejado e o valor atual. Dias (2016) explica que o controle PID leva em consideração três aspectos: a magnitude do erro (proporcional), a acumulação de erros passados (integral) e a taxa de mudança do erro (derivativo), o que resulta em uma resposta controlada, eficiente e estável. Esse método é amplamente adotado por sua capacidade de oferecer uma resposta rápida e controlada a mudanças no ambiente e no comportamento do veículo, conforme também é detalhado em *Design and Implementation of PID-Based Steering Control for 1/10-Scale Autonomous Vehicle* (2021).

2.3.3 Modelos de aprendizado de máquina para melhorar o comportamento autônomo

Modelos de aprendizado de máquina, como redes neurais e aprendizado por reforço, são utilizados para aprimorar o comportamento de veículos autônomos ao longo do tempo. Dias (2016) menciona que esses modelos são essenciais para melhorar a capacidade de adaptação dos veículos a ambientes dinâmicos e imprevisíveis, permitindo que o veículo aprenda com experiências passadas. Em *New control model for autonomous vehicles using integration of MPC and Stanley controllers* (2024), é discutido como a combinação de aprendizado de máquina com modelos de controle avançados, como o *Model Predictive Control* (MCP), pode otimizar o desempenho de veículos autônomos em condições desafiadoras, tornando-os mais eficazes em ambientes complexos.

2.3.4 Implementação de estratégias de navegação

A navegação autônoma exige a geração de trajetórias seguras e eficientes, para o que diversos algoritmos são utilizados. Dias (2016) aponta a relevância de algoritmos como o *Simultaneous Localization and Mapping* (SLAM), que permite ao veículo construir mapas do ambiente e se localizar dentro deles em tempo real. Além disso, New control model for autonomous vehicles using integration of MPC and Stanley controllers (2024) discute como técnicas de planejamento de caminho, como o controle baseado em Stanley, são usadas para determinar a melhor rota a seguir, garantindo uma navegação segura e otimizada. Esses algoritmos são cruciais para a tomada de decisões de navegação em tempo real, permitindo que o veículo se mova de forma eficiente e sem risco.

2.4 Estudos e pesquisas relacionados ao tema adas

O uso de veículos autônomos em escala reduzida tem se mostrado uma abordagem eficaz para pesquisas e testes de sistemas avançados de assistência ao condutor (ADAS). Esses modelos permitem a experimentação de algoritmos de controle, percepção e navegação em ambientes controlados e de baixo custo.

Plataformas como o *Donkey Car* oferecem um base acessível e modular para o desenvolvimento de veículos autônomos em pequena escala. Utilizando *hardware* como *Raspberry Pi* e *frameworks* de aprendizado profundo, como *TensorFlow*, o *Donkey Car* permite a implementação de diferentes tipos de pilotos automáticos, incluindo aprendizado profundo, seguimento de caminho e visão computacional.

Outra plataforma notável é o *RoboCar*, desenvolvido na Universidade de Luxemburgo. Trata-se de uma plataforma de pesquisa de código aberto para direção autônoma, que integra uma arquitetura de *hardware* e *software* robusta, alinhada aos sistemas existentes do veículo, minimizando a necessidade de modificações extensas. O RoboCar suporta várias funções de direção autônoma e foi testado em estradas públicas em Luxemburgo.

2.4.1 Comparação com outras plataformas como o *Robocar* e o *Tamiya*

Além do *Donkey Car* e do RoboCar, outras plataformas têm sido utilizadas para pesquisas em ADAS. O Tamiya, por exemplo, é uma plataforma de modelo reduzido amplamente utilizada em competições e pesquisas acadêmicas. Embora não seja especificamente projetado para direção autônoma, o Tamiya pode ser adaptado com sensores e controladores para experimentar algoritmos de navegação e controle.

Comparativamente, o *Donkey Car* é mais focado em aprendizado de máquina e visão computacional, enquanto o RoboCar oferece uma integração mais profunda com sistemas veiculares reais, sendo adequado para testes em ambientes urbanos. O Tamiya, por sua vez, é mais acessível e flexível para experimentações básicas e competições educacionais.

2.4.2 Inovações e tendências no desenvolvimento de veículos autônomos

As pesquisas em veículos autônomos têm avançado significativamente, com inovações em sensores, algoritmos de controle e integração de sistemas. Tendências atuais incluem o uso de aprendizado por reforço para melhorar a tomada de decisões em tempo real, a integração de múltiplos sensores para percepção mais precisa do ambiente e o desenvolvimento de plataformas de código aberto para facilitar a colaboração entre pesquisadores.

Além disso, há um crescente interesse em plataformas de simulação e testes virtuais, que permitem a avaliação de algoritmos em cenários complexos sem os riscos associados a testes em ambientes reais. Essas inovações estão contribuindo para o

desenvolvimento de sistemas ADAS mais robustos e eficientes, aproximando a realidade de veículos totalmente autônomos.

3 METODOLOGIA

A metodologia adotada neste projeto teve como foco principal o desenvolvimento prático de um veículo autônomo em escala reduzida, utilizando ferramentas acessíveis e de código aberto. A abordagem escolhida foi a de prototipagem incremental, onde nosso projeto final é constituído de pequenos pedaços aprimorados progressivamente, permitindo construir, testar e ajustar o sistema de forma contínua ao longo do processo.

O projeto foi estruturado em etapas: definição dos componentes eletrônicos e mecânicos, *software*, montagem da estrutura do carro, configuração do *hardware* principal (*Raspberry Pi* Modelo 4B), definição, instalação e personalização de *frameworks*, e, por fim, a realização de testes e melhorias com base no desempenho obtido.

Com isso, o projeto buscou equilibrar teoria e prática, priorizando o aprendizado aplicado e a validação contínua das soluções implementadas, sempre com foco na funcionalidade do sistema e nos objetivos do TCC.

3.1 Sistema Operacional *Raspberry* OS (64 bits)

A escolha por utilizar o sistema operacional *Linux*, mais especificamente a versão de 64 bits do *Raspberry Pi* OS, foi motivada por uma combinação de fatores técnicos e estratégicos que favorecem o desempenho e a estabilidade do sistema embarcado. O *Linux* é amplamente adotado em projetos de sistemas embarcados e robótica devido à sua natureza de código aberto, grande comunidade de suporte, flexibilidade na customização e compatibilidade com bibliotecas e *frameworks* essenciais, como *OpenCV, TensorFlow*, *Donkey Car* e outros.

A versão de 64 bits do sistema foi selecionada para garantir o suporte pleno a aplicações computacionalmente intensivas, como o processamento contínuo de vídeo capturado pela câmera embarcada e o controle em tempo real dos atuadores do veículo autônomo. Diferentemente das versões de 32 bits, que impõem limites de memória endereçável e utilizam de forma menos eficiente arquiteturas *multicore*, a versão de 64 bits permite um aproveitamento mais robusto do *hardware*, otimizando o desempenho do sistema durante as sessões de condução autônoma.

Além disso, o ambiente *Linux* oferece estabilidade, segurança e controle granular sobre os recursos do sistema, o que é fundamental em aplicações em tempo real e em sistemas ciberfísicos automotivos. Essas características tornam o *Linux* a plataforma ideal para o desenvolvimento e execução confiável do projeto.

3.2 *Framework* utilizado

O desenvolvimento de um veículo autônomo em escala reduzida demanda uma plataforma de *software* que seja simultaneamente compatível com as restrições computacionais da *Raspberry Pi* e suficientemente robusta para integrar sensores, atuadores e algoritmos de visão computacional.

Entre as diversas opções disponíveis, o *framework Donkey Car* foi escolhido devido à sua adequação técnica, flexibilidade e forte enfoque educacional.

O *Donkey Car* é um projeto *open source* consolidado, amplamente utilizado em ambientes acadêmicos, de pesquisa e projetos amadores. Sua arquitetura modular, baseada principalmente em *Python*, facilita a prototipagem ágil de sistemas autônomos, permitindo que usuários com conhecimentos básicos em programação e sistemas embarcados possam desenvolver soluções funcionais e escaláveis.

Além da compatibilidade com *hardware* acessível, o *framework* oferece suporte nativo para funcionalidades essenciais, tais como captura de imagens em tempo real, controle de atuadores via sinais PWM, treinamento supervisionado de redes neurais para condução e execução autônoma do veículo. Outro diferencial relevante é a interface web integrada, que simplifica a configuração de parâmetros, coleta de dados, execução de testes e monitoramento em tempo real, tudo acessível diretamente pelo navegador.

A escolha do *Donkey Car* também foi fundamentada na qualidade de sua documentação oficial e na existência de uma comunidade técnica ativa, especialmente em plataformas como *Discord*. Esse suporte colaborativo se mostrou fundamental para a resolução de desafios técnicos e para a atualização constante das práticas e ferramentas utilizadas ao longo do desenvolvimento do projeto.

3.2.1 Donkey Car

O *Donkey Car* é um *framework* de código aberto amplamente utilizado no desenvolvimento de veículos autônomos em escala reduzida. Ele integra ferramentas de aprendizado de máquina, como *TensorFlow* e *Keras*, com bibliotecas de controle embarcado e visão computacional, fornecendo uma base robusta para a construção e operação de sistemas de navegação autônoma.

Seu principal modo de operação baseia-se no aprendizado supervisionado por imitação, onde o veículo é conduzido manualmente enquanto a câmera registra imagens e o sistema armazena os comandos correspondentes de direção e aceleração. Esse conjunto de dados, chamado de *dataset*, é posteriormente utilizado para o treinamento de uma rede neural, capaz de inferir comandos de direção com base apenas nas imagens capturadas em tempo real.

Além do modo supervisionado, o *Donkey Car* também oferece suporte a outras abordagens de condução autônoma, o que o torna uma plataforma altamente versátil:

- Navegação baseada em GPS: com o uso de um receptor GPS e módulos de *software* específicos, é possível registrar coordenadas durante a condução manual e reproduzir automaticamente a rota. Essa abordagem é recomendada para ambientes externos, onde o sinal de GPS é confiável.
- Visão Computacional Tradicional: neste modo, o controle do veículo é realizado com base em algoritmos clássicos de processamento de imagem, como limiarização, segmentação de cor e detecção de contornos, eliminando a necessidade de modelos de aprendizado profundo. Essa abordagem é ideal para sistemas com restrições de processamento e necessidade de respostas mais determinísticas e em tempo real.

A Tabela 2 apresenta um resumo dos modos de desenvolvimento do Donkey Car, consolidando as principais características e funcionalidades de cada modo.

Tabela 2 - Modos de Desenvolvimento Donkey Car

MODO DE CONDUÇÃO	DESCRIÇÃO	VANTAGENS	LIMITAÇÕES
Aprendizado Supervisionado (Imitação)	O carro é conduzido manualmente enquanto grava imagens e comandos. Após o treinamento, a rede neural controla o veículo com base nas imagens.	• Baseado em redes neurais reais • Aprendizado automático de padrões • Ótimo para pistas complexas	• Exige grande volume de dados • Pode não generalizar bem • Treinamento consome tempo e recursos
Navegação por GPS	O veículo registra coordenadas GPS durante a condução manual e depois segue o mesmo percurso de forma autônoma.	• Ideal para ambientes externos • Não depende de visão computacional • Fácil de configurar com módulo GPS	• Baixa precisão em ambientes fechados • Não responde bem a obstáculos ou mudanças de percurso
Visão Computacional Tradicional	Utiliza algoritmos clássicos (ex: binarização, detecção de bordas) para seguir linhas ou trilhas visuais no solo.	• Rápida resposta • Baixo uso de recursos • Não requer treinamento	• Sensível a variações de iluminação • Limitado a pistas com alto contraste • Menor adaptabilidade

Fonte: Adaptado dos documentos oficiais Donkey Car, 2025

Essa comparação tem como objetivo auxiliar na compreensão das diferentes abordagens suportadas pela plataforma, permitindo uma escolha mais informada de acordo com os objetivos e restrições de cada projeto.

Dessa forma, a flexibilidade do *Donkey Car* o torna adequado tanto para aplicações educacionais quanto para projetos de pesquisa e prototipagem de soluções autônomas mais sofisticadas. No projeto descrito neste trabalho, foi adotado o modo de Visão Computacional Tradicional, por ser mais adequado aos recursos computacionais disponíveis e permitir maior controle sobre os algoritmos de percepção e decisão.

O sistema é dividido em módulos independentes que facilitam a manutenção e expansão do código. Entre suas principais funcionalidades estão:

- Captura e processamento de imagens em tempo real;
- Controle de direção (servo motor) e velocidade (ESC) via PWM;
- Treinamento e execução de modelos de aprendizado supervisionado;
- Interface web para monitoramento e configuração do sistema.

No contexto deste trabalho, o *Donkey Car* foi utilizado como plataforma principal de integração e pesquisa entre *hardware* e *software*, viabilizando o controle do veículo com base em visão computacional. Sua adoção permitiu acelerar o desenvolvimento e validar, na prática, os conceitos estudados sobre veículos autônomos, redes neurais e sistemas embarcados.

Módulos Donkey Car

Durante o desenvolvimento do veículo autônomo, foram utilizados diversos módulos disponíveis no *framework Donkey Car*, cada um responsável por uma funcionalidade essencial para a operação do carro. Esses módulos funcionam de forma integrada e são configuráveis por meio de arquivos *Python*, facilitando adaptações conforme o *hardware* utilizado no projeto.

Módulo de câmera

A câmera é o principal sensor do sistema autônomo, responsável por capturar as imagens da pista que servirão como entrada para os modelos de direção. No projeto, foi utilizada uma câmera USB, compatível com a *Raspberry Pi* 4B, conectada diretamente à porta USB da placa. O *Donkey Car* oferece suporte nativo a diferentes tipos de câmeras, facilitando a estratégia de configuração e captura de imagens em tempo real.

Módulo de controle de direção

O controle de direção foi realizado por meio de um servo motor conectado ao controlador PCA9685, que é uma placa de controle PWM I²C. O *Donkey Car* possui um módulo específico que permite controlar o ângulo de direção do servo, com ajustes finos feitos via *software*, garantindo uma resposta suave e precisa do sistema direcional, sendo extremamente funcional para validação do funcionamento da direção.

Módulo de controle de aceleração

Para o controle da velocidade, foi utilizado um ESC ligado ao mesmo PCA9685, em um canal separado. O módulo de *throttle* (aceleração) do *Donkey Car* permite ajustar a intensidade do sinal PWM enviado ao ESC, controlando a aceleração e frenagem do motor de forma eficiente e segura, permitindo conhecer por meio de ferramentas como osciloscópio o real funcionamento do ESC.

Módulo de gravação de dados

Um dos recursos mais importantes do *Donkey Car* é o módulo de gravação de dados, que permite registrar, durante a condução manual (modo teleoperado), as imagens capturadas pela câmera juntamente com os comandos de direção e aceleração aplicados. Esses dados são armazenados em arquivos estruturados, que posteriormente são utilizados no processo de treinamento do modelo de condução autônoma.

Esse processo de coleta e gravação é essencial para que o modelo de inteligência artificial aprenda os padrões corretos de direção, baseando-se em exemplos reais de pilotagem humana.

A Tabela 3 apresenta um resumo dos principais módulos do Donkey Car que são utilizados, destacando suas funções e contribuições para o funcionamento do veículo.

Tabela 3 - Módulos Donkey Car mais utilizados

MÓDULO	FUNÇÃO PRINCIPAL	COMPONENTE UTILIZADO
Câmera	Captura de imagens da pista	Câmera USB
Controle de Direção	Ajuste do ângulo de direção	Servo Motor + PCA9685
Controle de Aceleração	Comando de velocidade e frenagem	ESC + PCA9685
Gravação de Dados	Armazenamento de imagens e comandos	Armazenamento via Raspberry Pi

Fonte: Autor, 2025

Com a infraestrutura de *software* definida por meio do *Linux* e *Donkey Car*. A seguir, serão detalhados os elementos físicos utilizados no desenvolvimento do protótipo.

3.3 *Hardware* mecânica utilizado

A estrutura do veículo autônomo foi baseada em um modelo comercial de carro de controle remoto (RC) elétrico, em escala 1:10, originalmente projetado para corridas.

Esse tipo de chassi oferece flexibilidade na fixação de módulos adicionais, como câmeras, placas controladoras, servomotores e controladores eletrônicos de velocidade (ESC), o que facilita o desenvolvimento iterativo e a integração de *hardware* em ambientes de testes reais. A estrutura também possui dimensões e proporções próximas às de um veículo real em escala reduzida, o que permite a simulação mais fiel de cenários de navegação, curvas e obstáculos.

A adoção de uma plataforma RC pronta também contribui para a redução de custos e do tempo de desenvolvimento mecânico, permitindo que o foco do projeto seja direcionado à implementação e otimização dos sistemas embarcados de percepção, controle e tomada de decisão autônoma.

A Figura 9 apresenta o modelo do carro RC utilizado como base para o desenvolvimento do protótipo autônomo.

Figura 9 - Modelo comercial RC

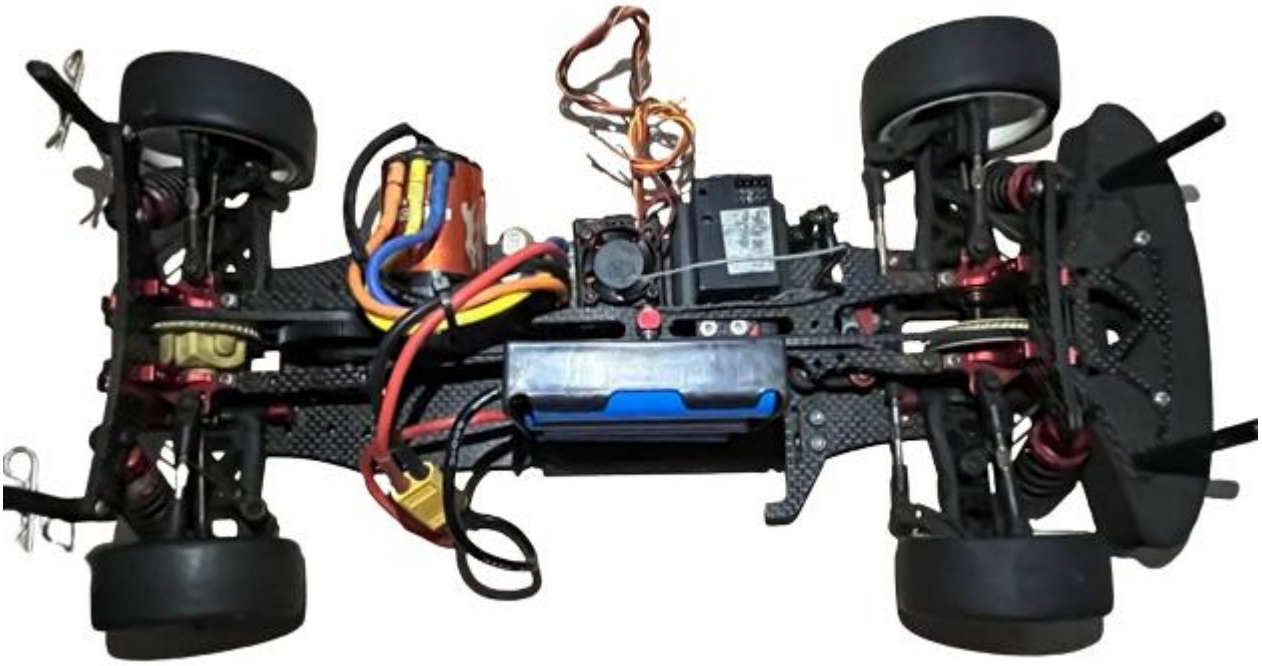
Fonte: Autor, 2025

Após a aquisição do veículo em escala, foi realizada a remoção da bolha superior (carenagem) e do sistema original de controle remoto. Em seguida, o sistema de ligação elétrica entre o ESC e o rádio transmissor, responsável pelo envio dos sinais PWM que controlavam a direção e a velocidade, foi desconectado. Essa modificação teve como objetivo transferir o controle da tração para a *Raspberry Pi*, permitindo, assim, a automação total do movimento do veículo.

Cabe destacar que o veículo não foi adquirido com sua bateria original. Diante disso, foi necessário desenvolver uma nova solução para a alimentação elétrica do motor e do ESC, garantindo que esses componentes pudessem ser operados de forma eficiente e compatível com o novo sistema embarcado. Todo o processo de desenvolvimento do sistema de alimentação será detalhado nos capítulos seguintes.

A Figura 10 ilustra o chassi do veículo após a remoção do controle via rádio transmissor, evidenciando o ESC e o servo motor desconectados do sistema original e já integrados à nova estrutura de alimentação elétrica projetada para o projeto de automação.

Figura 10 - Chassi com Alimentação, ESC e Motor



Fonte: Autor, 2025

Como o modelo não possuía suporte específico para os módulos eletrônicos, uma base de acrílico foi instalada na parte superior do chassi, ocupando o espaço anteriormente destinado à carenagem. Essa base serviu como plataforma para fixar a *Raspberry Pi*, o controlador PCA9685 e o *pack* de baterias. Inicialmente, os componentes foram posicionados de forma funcional sobre essa base para facilitar o acesso durante os testes e ajustes do sistema.

3.4 *Hardware* eletrônico utilizado

A construção do carro autônomo em escala, envolveu a integração de diversos componentes eletrônicos, controladores e atuadores que são fundamentais para garantir o funcionamento do sistema de direção, propulsão, processamento e controle. A seguir, são detalhados os principais elementos utilizados.

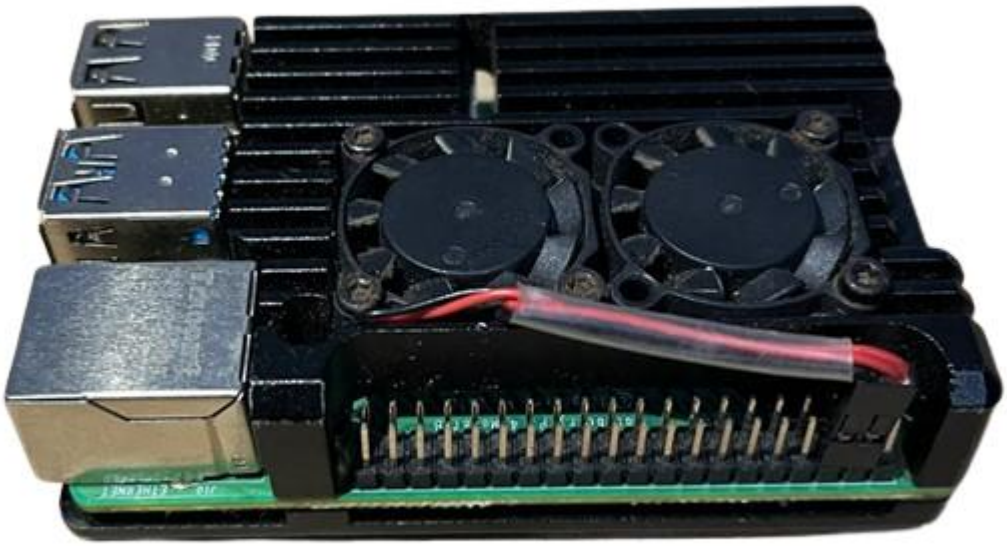
3.4.1 Raspberry Pi

Uma *Raspberry Pi* 4 Modelo B, foi utilizada como unidade central de processamento do sistema. A versão conta com 4 GB de memória RAM oferecendo

desempenho satisfatório para a execução simultânea das tarefas de visão computacional, controle dos atuadores e operação do modelo de direção autônoma.

A *Raspberry Pi* pode aquecer significativamente quando utilizado para tarefas exigentes (como processamento de imagem e automação), ela reduz automaticamente seu desempenho para evitar danos, isso é chamado de *thermal throttling.* Por esse fato, foi instalado um dissipador de calor, que integra toda a placa e integra dois coolers, garantindo o máximo de desempenho do processador. A figura 11 apresenta a *Raspberry Pi* com o sistema de dissipação de calor instalado.

Figura 11 - Raspberry Pi 4 Modelo B 4GB



Fonte: Autor, 2025

O *datasheet* da *Raspberry Pi* encontra-se listado nas referências deste trabalho.

3.4.2 Cartão MicroSD

O cartão microSD tem a função principal de armazenar o Sistema Operacional e todos os arquivos necessários para o funcionamento da placa. Ele funciona como o *Hard Disk* ou *Solid State Drive* de um computador, pois a *Raspberry Pi* não possui memória de armazenamento interno.

Utilizou-se um cartão microSD de 128 GB, classe U3 A2 V30 1066x, essencial para o funcionamento do sistema. A escolha desse modelo se justifica pela alta capacidade de armazenamento necessária para comportar o sistema operacional, os dados de treino, logs e o próprio modelo de rede neural, além do elevado desempenho

em leitura e gravação. A classificação U3 assegura uma taxa mínima de gravação sequencial de 30 MB/s, a classe V30 garante desempenho contínuo para gravação de vídeo, enquanto a *Aplication Performance Class 2* (A2) oferece maior velocidade em operações de leitura e escrita aleatória, ideal para a execução de aplicações diretamente no cartão. A velocidade 1066x corresponde a uma taxa de leitura teórica de aproximadamente 160 MB/s, o que contribui para a agilidade do sistema. Esses atributos são fundamentais para manter a estabilidade e a resposta em tempo real exigidas pelo sistema embarcado. A comunicação com os demais módulos eletrônicos, como o controlador PWM PCA9685, foi realizada por meio dos pinos GPIO e da interface I2C, possibilitando o controle preciso dos atuadores com baixo consumo de recursos computacionais. A Figura 12 apresenta o MicroSD utilizado.

Figura 12 - MicroSD

Fonte: Lexar, 2024

O processo de instalação do Sistema Operacional e a configuração do cartão microSD, bem como o *datasheet* do modelo utilizado, encontram-se disponíveis nas referências deste trabalho.

3.4.3 Célula de Lítio 18650

Para a alimentação da *Raspberry Pi* e do Motor Elétrico, foram utilizadas células de íon de lítio 18650 devido à sua alta densidade energética, capacidade de fornecimento de corrente contínua, baixo custo e boa relação entre tamanho e desempenho. Essas características tornam esse tipo de célula ideal para aplicações embarcadas e móveis, como o carro autônomo desenvolvido neste projeto. A figura 14 apresenta as Células utilizadas.

Para o fornecimento de energia elétrica à *Raspberry Pi* e ao motor elétrico utilizados neste projeto, optou-se pela utilização de células recarregáveis do tipo íon de lítio modelo 18650. Essa escolha se justifica por uma série de vantagens técnicas que tornam essas células especialmente adequadas para sistemas embarcados e aplicações móveis, como é o caso do veículo autônomo em escala 1:10 desenvolvido neste trabalho.

As células 18650 se destacam principalmente pela sua alta densidade energética, o que significa que elas conseguem armazenar uma grande quantidade de energia em

um volume reduzido. Essa característica é fundamental em projetos onde há limitação de espaço físico e peso, como em plataformas móveis que precisam de autonomia energética sem comprometer a leveza e a compactação dos componentes.

Além disso, essas células apresentam capacidade elevada de fornecimento de corrente contínua, sendo capazes de alimentar, de forma estável e eficiente, tanto circuitos eletrônicos sensíveis (como o microcontrolador e a placa *Raspberry Pi*) quanto atuadores de maior consumo, como motores elétricos. A capacidade de manter a tensão de saída dentro de uma faixa estável também contribui para a confiabilidade do sistema, mesmo sob variações de carga durante o funcionamento do carro. A figura 13 ilustra as células.

Figura 13 - Células de Li ion

Fonte: WinPow, 2025

O *datasheet* das células encontra-se listado nas referências deste trabalho.

3.4.4 Servo Motor

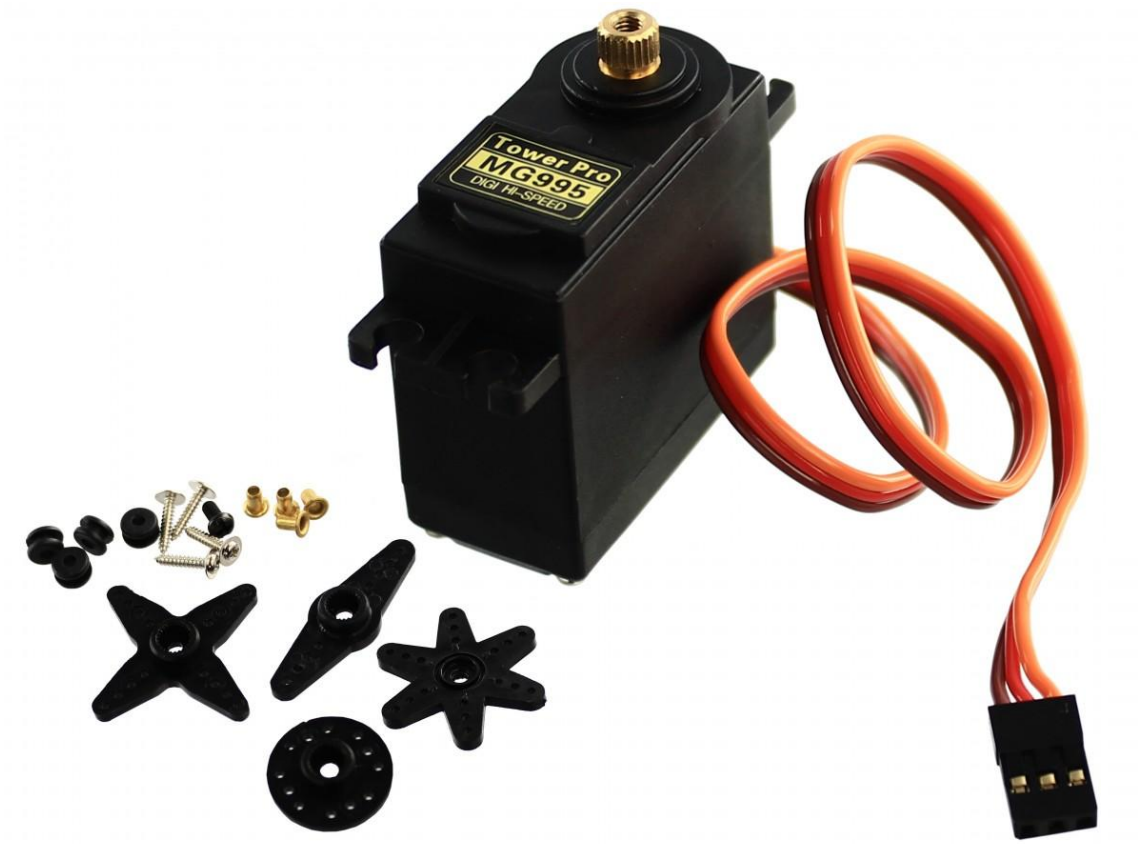
A direção do veículo foi implementada por meio de um servo motor, que atua diretamente sobre as rodas dianteiras, permitindo a realização dos movimentos de esterçamento. Esse servo foi conectado ao canal 0 de um módulo controlador de sinais PWM do tipo PCA9685, amplamente utilizado em sistemas embarcados devido à sua capacidade de gerar até 16 canais PWM com alta precisão e estabilidade.

O controle do ângulo de direção é realizado por meio de sinais PWM, cuja geração é feita pela *Raspberry Pi*. A comunicação entre a *Raspberry Pi* e o módulo PCA9685 ocorre via protocolo I2C, o que garante uma conexão estável, com baixo consumo de pinos GPIO e alta eficiência na transmissão de dados.

Essa configuração possibilita o envio de comandos de posição para o servo motor com alta resolução, permitindo ajustes finos no ângulo de direção, fator essencial para a estabilidade e o desempenho do sistema de navegação autônoma do veículo, especialmente em curvas e mudanças rápidas de trajetória.

A Figura 14 apresenta um modelo de servo motor semelhante ao empregado no projeto.

Figura 14 - Servo motor



Fonte: A2robotcs, 2025

3.4.5 Controlador Eletrônico de Velocidade

O ESC é responsável pelo acionamento e controle da velocidade do motor elétrico de tração. Ele opera por meio de sinais *Pulse Width Modulation* (PWM), que determinam a intensidade da aceleração enviada ao motor. Essa abordagem permite uma resposta precisa e proporcional à variação do sinal de controle, essencial para o comportamento dinâmico do veículo em operação autônoma.

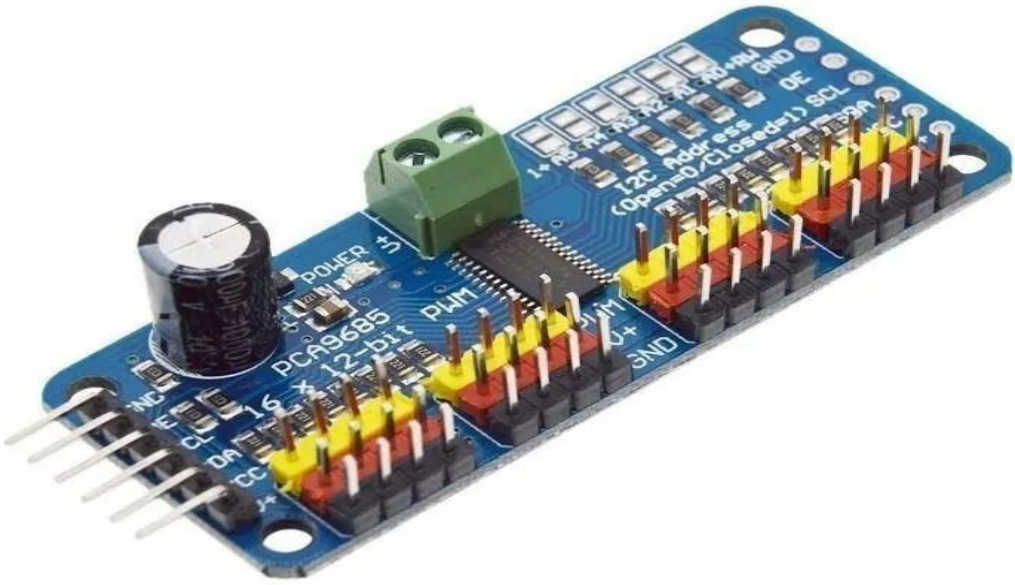
O *datasheet* da ESC encontra-se listado nas referências deste trabalho.

3.4.6 PCA9685

O PCA9685 é um módulo controlador de sinais PWM capaz de operar até 16 canais independentes com alta precisão. Ele foi utilizado para gerar os sinais necessários ao controle do servo motor de direção e do ESC de tração, oferecendo estabilidade e precisão no acionamento dos atuadores. Sua operação via interface I2C permite o controle simultâneo de múltiplos dispositivos com baixa carga de processamento na unidade central.

A figura 15 abaixo mostra o módulo PCA9685 utilizado no controle dos atuadores do veículo.

Figura 15 - PCA 9685



Fonte: Arducore, 2025

O *datasheet* da PCA9685 encontra-se listado nas referências deste trabalho.

3.4.7 Câmera USB

A câmera USB desempenhou o papel fundamental de sensor de visão**,** sendo responsável pela captura contínua de imagens do ambiente em tempo real. Essa funcionalidade é essencial para a implementação de sistemas de navegação autônoma baseados em visão computacional, uma vez que permite que o veículo perceba e interprete o cenário ao seu redor.

As imagens capturadas pela câmera foram processadas diretamente pela *Raspberry Pi* 4B, que executou o modelo de controle autônomo embarcado. Através

desse modelo, foi possível realizar o reconhecimento da trajetória, identificar padrões de pista e determinar comandos de direção adequados com base nos dados visuais obtidos. Esse processo de percepção e decisão baseados exclusivamente na visão simula, em escala, o funcionamento de sistemas avançados de assistência à condução (ADAS) e de veículos autônomos comerciais. A figura 16 ilustra a câmera utilizada.

Figura 16 - Câmera USB

Fonte: Yahboom, 2025

O *datasheet* da Câmera encontra-se listado nas referências deste trabalho.

3.4.8 Arquitetura do sistema

A arquitetura do sistema é composta por uma **Raspberry Pi 4 Modelo B 4G**, que atua como **unidade central de controle e processamento**. A câmera USB é responsável pela captura contínua de imagens, utilizadas no processamento de visão computacional para detecção da pista e tomada de decisão em tempo real.

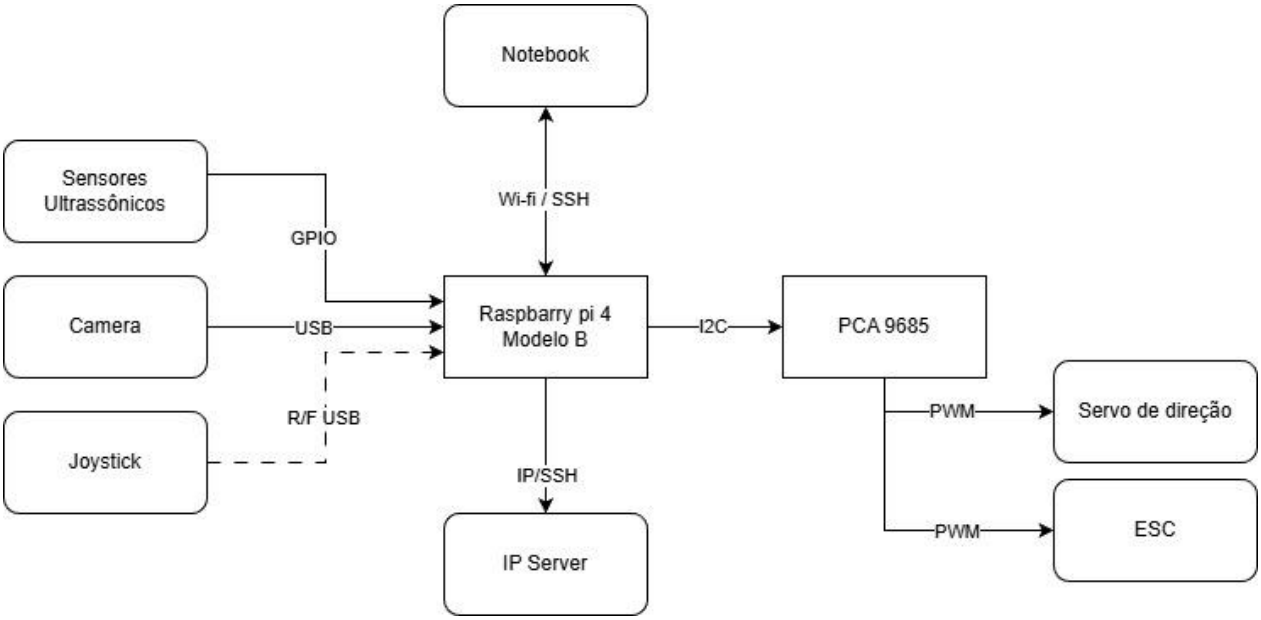
O **módulo PCA9685** é empregado para o **controle dos atuadores**, gerando sinais PWM destinados ao **servo motor de direção** e ao **ESC** que controla a velocidade

do motor de tração. Ambos os atuadores recebem comandos de controle definidos pelo algoritmo principal em execução na Raspberry Pi.

A comunicação entre o veículo e um **notebook** ocorre por meio de **conexão em rede IP (via Wi-Fi)**, possibilitando o **monitoramento, calibração e ajuste remoto** dos parâmetros do sistema durante os testes.

A **Figura 17** apresenta o **diagrama de blocos da arquitetura do sistema**, ilustrando a integração entre os componentes de hardware e as interfaces de comunicação. Essa configuração simplificada garante **baixa complexidade elétrica**, **redução de interferências** e **maior confiabilidade operacional**, mantendo todos os módulos integrados de forma eficiente e coerente com os requisitos do veículo autônomo em escala 1:10.

Figura 17 Diagrama de blocos do sistema



Fonte: Autor, 2025

3.5 Etapas de Desenvolvimento

O desenvolvimento desse projeto foi realizado de forma incremental, passando por diversas fases. A metodologia aplicada priorizou a experimentação prática, com testes constantes e ajustes baseados nos resultados obtidos em cada etapa. Esta abordagem permitiu compreender melhor os desafios técnicos envolvidos na montagem do sistema e na implementação da navegação autônoma.

As atividades foram organizadas em fases sequenciais, que incluíram planejamento, seleção e montagem dos componentes físicos, a configuração do ambiente de desenvolvimento, a calibração dos atuadores, e o desenvolvimento do

cérebro do veículo, por fim, os testes práticos de validação do sistema. Cada fase apresentou suas próprias particularidades e obstáculos, que foram solucionados com base em pesquisa, análise de documentação técnica e tentativa e erro.

A seguir, são detalhadas as principais etapas do processo de desenvolvimento, destacando os procedimentos adotados, e os desafios superados ao longo do projeto.

3.5.1 Planejamento e Levantamento de Requisitos

O planejamento deste projeto teve início com a definição clara do objetivo central: desenvolver um veículo autônomo em escala reduzida, capaz de percorrer uma pista de forma autônoma utilizando apenas uma câmera como sensor de percepção, além de um sistema embarcado para o processamento das informações e o controle da direção e velocidade. A inspiração para este trabalho partiu de Weslei Alves, ex-aluno da FATEC Santo André, cuja iniciativa serviu como referência para a concepção do protótipo.

A partir dessa definição, foi realizado um levantamento detalhado dos requisitos necessários para viabilizar a construção e o funcionamento do sistema, os quais foram organizados em duas categorias principais: requisitos de *hardware* e requisitos de *software*.

Durante essa etapa, ficou definido que o veículo utilizaria:

- um sistema de alimentação baseado em baterias recarregáveis;
- um servo motor para o controle da direção;
- um motor elétrico controlado por um ESC para a tração;
- Uma *Raspberry Pi* como unidade de processamento principal;
- Uma Câmera para visão do veículo.

Esse mapeamento permitiu identificar os componentes essenciais, as ferramentas de desenvolvimento, as possíveis limitações técnicas e a necessidade de implementação de outros componentes, garantindo uma base sólida para as etapas seguintes do projeto.

3.5.2 Sistema operacional

Concluído o planejamento e estabelecidos os requisitos do sistema, iniciou-se a preparação do ambiente de *software* embarcado, executado sobre a *Raspberry Pi*. Foram iniciadas as etapas de instalação e configuração do sistema operacional *Ubuntu Budgie 64-bit* e do *framework Donkey Car*, que juntos compõem a base de *software* necessária para o desenvolvimento do veículo autônomo.

3.5.3 Instalação Sistema *Ubuntu Budgie 64-bit*

A *Raspberry Pi* requer um sistema operacional leve, compatível com a arquitetura ARM64 e com suporte a bibliotecas científicas e ferramentas de desenvolvimento em *Python*. Optou-se pela instalação do *Ubuntu Budgie 64-bit*, versão estável mantida

oficialmente pela *Raspberry Pi* Foundation. Essa distribuição, baseada na família Debian GNU/*Linux*, apresenta compatibilidade nativa com os principais recursos da placa e fornece suporte direto a aceleração gráfica por *hardware*, interfaces de comunicação (GPIO, I2C, SPI, UART) e periféricos USB, como teclados, câmeras e adaptadores Wi-Fi, necessitando de um cartão MicroSD, para servir de memória tanto para o sistema operacional quanto para os arquivos.

O microSD foi formatado e preparado por meio da ferramenta oficial *Raspberry Pi Imager,* disponível para sistemas operacionais Windows, macOS e *Linux*. Esse utilitário permite a gravação direta da imagem do sistema no cartão, além de opções avançadas de configuração como a ativação do acesso SSH, configuração da rede Wi-Fi, nome do host e definição de credenciais de login. Tais configurações foram aplicadas previamente à gravação da imagem, permitindo que a *Raspberry Pi* estivesse totalmente acessível remotamente logo após o primeiro boot, via conexão SSH.

Com todas as configurações aplicadas e a imagem devidamente gravada no cartão microSD, a *Raspberry Pi* torna-se totalmente funcional e pronta para ser utilizada como um computador autônomo, acessível remotamente via SSH. O processo completo de instalação, configuração do sistema e preparação da placa encontra-se detalhado no Apêndice deste relatório.

Instalação do Framework Donkey Car

Após a instalação e configuração do sistema operacional *Raspberry Pi* OS (descrita na seção anterior), foi iniciada a instalação do ambiente de *software* responsável por interpretar as imagens capturadas pela câmera, controlar os atuadores e executar os comandos necessários para que o veículo funcione de forma autônoma.

A instalação do *Donkey Car* exige alguns passos técnicos importantes, mas que foram cuidadosamente seguidos para garantir o funcionamento adequado do sistema. Primeiramente, foram instaladas todas as bibliotecas e ferramentas que o sistema precisa para rodar o *Donkey Car* corretamente. Isso inclui compiladores, bibliotecas de comunicação com dispositivos conectados via USB, suporte à câmera e drivers para leitura e escrita dos sinais que controlam os motores e o servo.

Na sequência, foi criado um ambiente virtual *Python*, um espaço separado dentro do sistema operacional onde é possível instalar os pacotes do *Donkey Car* sem interferir em outros programas da *Raspberry Pi*. Esse ambiente evita conflitos entre diferentes versões de bibliotecas e ajuda a manter o sistema mais organizado e seguro.

Com o ambiente virtual pronto, os pacotes do *Donkey Car* foram instalados utilizando o pip, ferramenta oficial de instalação de bibliotecas *Python*. Essa instalação trouxe, automaticamente, todos os módulos necessários para o funcionamento do

framework, incluindo aqueles que tratam da visão computacional, controle de *hardware* e comunicação entre os componentes do carro.

Por fim, foi criado um projeto local do *Donkey Car*, que funciona como a pasta principal onde ficam organizados todos os arquivos do veículo, como os dados coletados, as configurações e, futuramente, os modelos de controle. Esse projeto é essencial para que o carro possa ser calibrado, treinado e testado.

Com a *Raspberry Pi* devidamente configurada e o ambiente *Donkey Car* instalado e funcional, foi possível garantir que o sistema embarcado estivesse preparado para realizar o processamento de dados e o controle dos componentes eletrônicos do veículo. A partir desse ponto, a próxima etapa do projeto envolveu o desenvolvimento do sistema de alimentação elétrica, responsável por fornecer energia de forma estável e eficiente a todos os módulos do carro, incluindo a unidade de processamento, os motores e os demais periféricos.

3.5.4 Alimentação do sistema

O fornecimento de energia elétrica é uma etapa crítica no desenvolvimento de sistemas embarcados móveis, pois afeta diretamente a estabilidade, a autonomia e a segurança do funcionamento do veículo.

Neste projeto, optou-se por uma arquitetura de alimentação **segmentada em dois circuitos independentes**, com o objetivo de evitar interferências elétricas entre o motor e o sistema de processamento. Essa separação garante maior confiabilidade, principalmente durante picos de corrente ocasionados por acelerações ou mudanças bruscas de direção.

Pack de Baterias para o Motor e Servo

Para o sistema de tração e direção, foi utilizado um **pack composto por quatro células de íons de lítio modelo 18650** ligadas em série (4S). Cada célula possui tensão nominal de 3,7 V, resultando em uma tensão total nominal de aproximadamente 14,8 V e máxima de 16,8 V quando totalmente carregado.

Essa configuração atende plenamente à faixa de operação do **d**o motor **brushless**, que operam entre 9 V e 16,8 V. A capacidade total do pack é de **2 600 mAh**, com corrente de descarga contínua de até **2,7 A**, o que permite suportar o consumo elevado do motor durante acelerações e manobras prolongadas.

O conjunto foi testado sob carga real, mantendo estabilidade de tensão e temperatura inferior a 45 °C após 20 minutos de operação contínua, dentro dos limites seguros recomendados pelo fabricante das células.

Power Bank para Alimentação da Raspberry Pi 4B

Para o sistema lógico e de processamento de imagem, a alimentação é fornecida por um **power bank de 22,5 W** e **capacidade nominal de 20 000 mAh**, originalmente projetado para recarga rápida de dispositivos móveis.

A escolha desse dispositivo foi motivada por sua **estabilidade de saída** (5V e 3A), **proteção interna contra sobrecorrente e subtensão**, além da **facilidade de recarga via porta USB-C**. O uso do power bank elimina a necessidade de um segundo pack de células 18650 e de conversores DC-DC adicionais, simplificando a arquitetura elétrica e reduzindo o peso total do veículo.

Durante os testes de campo, o power bank manteve alimentação contínua da Raspberry Pi por períodos superiores a 5 horas, sem quedas de tensão perceptíveis, garantindo operação confiável do sistema de visão computacional e controle.

Validação do Sistema de Alimentação

Foram realizados ensaios com o conjunto completo, envolvendo:

- **Medição de tensão e corrente** com multímetro digital sob carga real (motor e câmera em operação simultânea);
- **Verificação de estabilidade térmica** por meio de termômetro infravermelho, mantendo temperaturas dentro da faixa segura;
- **Ensaios de autonomia**, comprovando a capacidade do power bank e do pack 4S em suportar os requisitos de energia do veículo durante testes prolongados.

A solução final proporcionou **simplicidade construtiva**, **redução de peso**, e **robustez operacional**, atendendo plenamente às necessidades energéticas do veículo autônomo em escala 1:10.

Na seção seguinte, é apresentado o desenvolvimento do sistema de controle do servo motor de direção, responsável por definir a trajetória do veículo autônomo em resposta ao ambiente captado pela câmera.

3.5.5 Direção do Veículo

O sistema de direção do veículo autônomo utiliza um servo motor de precisão acionado por um controlador PWM (PCA9685) comandado pela unidade de processamento (Raspberry Pi 4B). A escolha da PCA9685 permitiu gerar sinais PWM estáveis sem sobrecarregar o processador da Raspberry Pi, garantindo respostas rápidas e repetíveis do atuador durante as manobras.**Estrutura de Controle Utilizada**

Para o controle do servo motor, foi adotada uma arquitetura que combinava os seguintes componentes:

- ***Raspberry Pi* 4B**: Unidade de controle principal do sistema;
- **Placa PCA9685**: Controladora PWM de 16 canais, conectada à *Raspberry Pi* via protocolo I2C;
- **Servo motor de direção**: Responsável pela movimentação das rodas direcionais, conectado ao canal 1 da PCA9685.

3.5.5.1 Componentes e arquitetura

- **Unidade de controle:** Raspberry Pi 4B (controlador principal).
- **Controladora PWM:** Adafruit PCA9685 (16 canais, interface I2C).
- **Atuador:** Servo motor de direção, conectado ao canal 1 da PCA9685.
- **Comunicação:** I2C (barramento 1) entre Raspberry Pi e PCA9685.

3.5.5.2 Conexões elétricas

As ligações físicas adotadas foram as seguintes:

- **I2C:** SDA → pino 03 (GPIO 2); SCL → pino 05 (GPIO 3).
- **Alimentação da PCA9685:** VCC e GND conforme esquema do projeto; GND comum entre Raspberry Pi, PCA9685 e fonte do servo.
- **Ligação do servo ao canal 1 da PCA9685:** fio de sinal (PWM), alimentação positiva e terra.

3.5.5.3 Configuração e calibração no *Donkey Car*

O processo de calibração iniciou-se com a edição do arquivo de configuração do *Donkey Car*:

nano ~/mycar/myconfig.py

Em seguida executou-se o utilitário de calibração:

donkey calibrate --channel 1 --bus=1

Durante os testes iniciais observou-se que os valores numéricos aceitos pelo utilitário (escala 0–1500) não correspondiam diretamente ao tempo de pulso requerido pelo servo, resultando em falta de resposta com valores testados inicialmente (600–800). Para diagnosticar, foi empregado um osciloscópio.

3.5.5.4 Medições com osciloscópio e mapeamento PWM

Com o osciloscópio comparou-se o pulso do rádio transmissor original e os pulsos gerados pela PCA9685. As medições indicaram os tempos de pulso típicos para o servo:

- 1,0 ms → direção totalmente à direita

- 1,5 ms → centro
- 2,0 ms → direção totalmente à esquerda

A correspondência entre os valores de configuração e os pulsos medidos foi mapeada como:

- 250 → 1,0 ms
- 325 → 1,5 ms (centro)
- 400 → 2,0 ms

Com base nesse mapeamento os parâmetros de direção foram definidos no myconfig.py como:

STEERING_LEFT_PWM = 400

STEERING_RIGHT_PWM = 250

STEERING_CENTER_PWM = 325

3.5.5.5 Resultados e considerações finais

Após a inserção dos novos parâmetros o servo passou a responder corretamente aos comandos enviados via PCA9685. Observou-se: manobrabilidade adequada para pista, controle de direção estável e eliminação de cargas computacionais desnecessárias da Raspberry Pi. A utilização do osciloscópio foi determinante para mapear corretamente a relação entre os valores do utilitário de calibração e os tempos de pulso efetivos do servo. (Detalhes e registro das medições encontram-se no relatório de calibração no github.)

3.5.6 Controle de Tração (ESC)

O controle de tração do veículo autônomo foi realizado por meio de um ESC, responsável por regular a velocidade de rotação do motor elétrico. Para garantir que os comandos de aceleração, parada e marcha à ré enviados pelo sistema embarcado fossem interpretados corretamente, foi necessário realizar um processo detalhado de calibração. O objetivo foi assegurar a compatibilidade entre os sinais PWM gerados pelo módulo PCA9685, controlado pela *Raspberry Pi* 4B, e o comportamento esperado pelo ESC.

As etapas detalhadas deste procedimento encontram-se descritas no *Relatório de Calibração do ESC* que consta no repositório github deste trabalho listado nas referências.

3.5.6.1 Arquitetura de Controle de Tração

A arquitetura utilizada para o controle de tração foi composta pelos seguintes elementos:

- *Raspberry Pi* 4B: unidade computacional principal;
- Módulo PCA9685: responsável por gerar sinais PWM com precisão;
- ESC (*Electronic Speed Controller*): conectado ao canal 0 da PCA9685;
- Motor DC de tração: alimentado por uma bateria de lítio independente;
- Fonte USB: responsável por alimentar a *Raspberry Pi* separadamente.

3.5.6.2 Conexões eletricas

As conexões foram feitas da seguinte forma:

- **Sinal PWM**: fio laranja do ESC → canal 0 da PCA9685;
- **Alimentação positiva**: fio vermelho do ESC;
- **Terra (GND)**: fio preto ou marrom do ESC.

Essa conexão permitiu a emissão de sinais PWM estáveis sem sobrecarregar o processamento da *Raspberry Pi*.

3.5.6.3 Integração com o *Donkey Car*

Para a calibração do ESC, utilizou-se o *framework Donkey Car*, configurado por meio do arquivo myconfig.py. Nele, foram definidos os valores mínimos e máximos de PWM correspondentes às ações de aceleração, parada e marcha à ré. A calibração foi iniciada com o comando:

donkey calibrate --channel 0 --bus=1

Esse comando habilitou o envio de sinais PWM entre os valores 0 e 1500, permitindo a identificação dos pulsos exatos que ativam cada função do ESC.

3.5.6.4 Desafios Iniciais de Comunicação

Nas primeiras tentativas de calibração direta, o motor não respondia aos valores de PWM enviados (por exemplo, 300, 350, 400), o que indicava que o ESC ainda se encontrava em modo de controle via rádio. A solução encontrada foi realizar uma reconfiguração inicial do ESC utilizando o controle remoto original, seguindo as instruções do manual do fabricante.

3.5.6.5 Validação com Osciloscópio

Com o auxílio de um osciloscópio digital portátil, modelo **DS203**, foi possível verificar com precisão os pulsos gerados e recebidos pelo ESC. O DS203 é um

osciloscópio compacto de quatro canais, amplamente utilizado em aplicações didáticas e embarcadas, por oferecer portabilidade, boa resolução temporal e recursos adequados para análise de sinais digitais e analógicos de baixa e média frequência.

Sinais do controle remoto original:

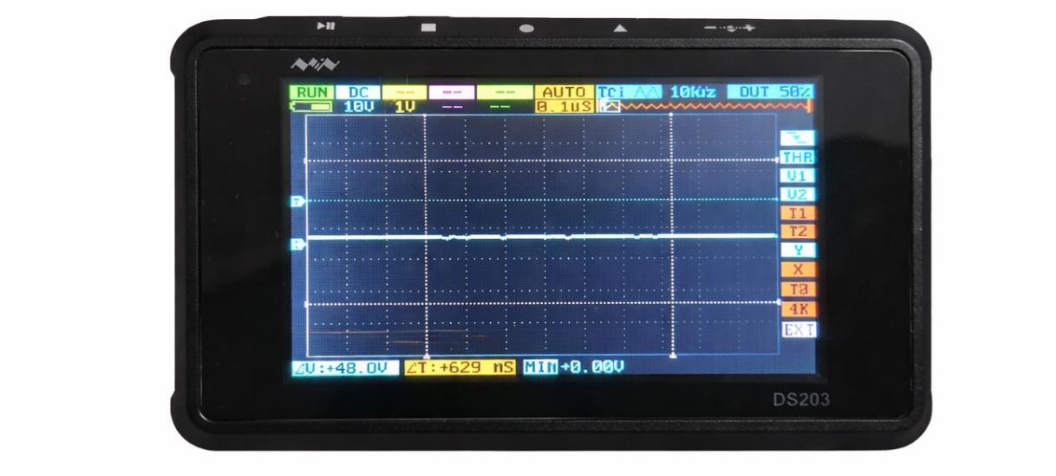
- Parado: 1,47 ms
- Aceleração: 1,48 a 1,81 ms
- Ré: 0,81 a 1,46 ms

Sinais da PCA9685:

- Valor 350: 1,47 ms (parado)
- Valor 370: 1,51 ms (aceleração)
- Valor 300: 1,4 ms (marcha à ré)

Apesar da correspondência entre os sinais, observou-se que a marcha à ré não era ativada diretamente, indicando a presença de uma lógica de segurança interna no ESC. A Figura 18 ilustra o osciloscópio utilizado.

Figura 18 – Osciloscópio DS203



Fonte: Autor, 2025

3.5.6.6 Solução Definitiva com Cartão de Configuração

Após análise do *datasheet* do ESC, identificou-se que ele operava com diferentes modos configuráveis. Utilizando um cartão de configuração fornecido pelo fabricante, foi possível alterar o modo de operação para normal, permitindo o acionamento direto da marcha à ré sem a necessidade de uma sequência de comandos específica.

Esse ajuste solucionou a principal limitação observada no sistema e viabilizou o controle completo da tração de forma direta e confiável.

3.5.6.7 Parâmetros Finais de Calibração

Com os testes concluídos, foram definidos os seguintes valores de PWM no arquivo de configuração do *Donkey Car* (myconfig.py):

- THROTTLE_STOPPED_PWM = 350 → Motor parado
- THROTTLE_FORWARD_PWM = 370 → Aceleração para frente
- THROTTLE_REVERSE_PWM = 300 → Marcha à ré

Todos os testes foram realizados com o veículo suspenso, de modo a evitar movimentos indesejados durante a calibração.

3.5.6.8 Conclusão da calibração

A calibração do ESC foi essencial para garantir o controle confiável da tração do veículo autônomo. O processo envolveu tanto testes práticos com osciloscópio quanto ajustes no modo de operação do ESC via cartão de configuração. A definição de valores finais de PWM assegurou que o sistema autônomo pudesse acelerar, parar e dar ré com precisão e sem a necessidade de intervenção manual. O relatório completo do procedimento consta no repositório github deste trabalho listado nas referências.

3.6 Codificação Inicial – Seguidor de Linha

Com o objetivo de estabelecer uma base sólida de conhecimento e aprendizado prático sobre codificação e o comportamento do sistema, optamos, inicialmente, pelo desenvolvimento de um protótipo de carro seguidor de linha. Essa etapa é fundamental para compreendermos o controle dos atuadores, a captura e interpretação de imagens pela câmera, bem como as respostas autônomas do sistema diante de estímulos visuais.

3.6.1 Objetivo e Justificativa

O desenvolvimento do seguidor de linha nos permite validar a comunicação entre os principais componentes da arquitetura do veículo. Por meio dessa integração inicial, podemos testar comandos básicos de direção e aceleração, utilizando algoritmos simples de processamento de imagem e tomada de decisão.

A escolha por iniciar com um seguidor de linha se justifica por sua eficácia como projeto introdutório, uma vez que envolve aspectos essenciais da robótica autônoma, como visão computacional, controle em tempo real e calibração de movimentos. Além disso, essa abordagem facilita a transição para fases mais avançadas, como o reconhecimento de pistas e curvas complexas, que serão implementadas nas etapas seguintes do projeto.

Diferentemente de projetos que utilizam sensores infravermelhos ou arrays de sensores analógicos para detecção de linha, neste protótipo optamos por utilizar exclusivamente uma câmera como sensor principal. Essa escolha, embora aumente a complexidade do sistema, oferece maior flexibilidade e realismo, simulando melhor as situações encontradas no mundo real e aproximando o funcionamento do veículo ao de carros autônomos modernos.

Grande parte do desenvolvimento inicial foi dedicada ao tratamento e análise das imagens capturadas pela câmera, utilizando técnicas de visão computacional para identificar a linha-guia da pista.

Essas etapas são executadas em tempo real, e os resultados obtidos são utilizados para ajustar dinamicamente o ângulo de direção, garantindo que o veículo permaneça na trajetória desejada.

A implementação dessa lógica foi realizada em Python, com o uso da biblioteca OpenCV, amplamente adotada em projetos de visão computacional. Essa base servirá de alicerce para o desenvolvimento de funcionalidades mais avançadas do projeto, como a detecção de curvas, entroncamentos e, futuramente, a tomada de decisões mais complexas.

3.6.2 Desenvolvimento do sistema

Este subcapítulo apresenta, em ordem cronológica, o processo de desenvolvimento do sistema de controle e navegação do carro autônomo seguidor de linha. O projeto foi construído de forma incremental, com testes práticos a cada etapa, permitindo ajustes contínuos de hardware e software até a obtenção de um comportamento estável e confiável do veículo.

3.6.3 Primeira etapa – Captura de vídeo

O primeiro passo no desenvolvimento do sistema foi validar o funcionamento da câmera USB conectada à Raspberry Pi, pois a leitura de imagens em tempo real é essencial para o carro autônomo. A câmera fornece os frames que permitem à visão computacional identificar a pista e gerar sinais de controle para o servo motor e o motor do veículo.

Para realizar a captura de vídeo, utilizou-se a biblioteca **OpenCV**, amplamente utilizada em projetos de visão computacional por sua eficiência e simplicidade. O processo envolve três etapas principais:

- **Inicialização da câmera** – Verifica-se se a câmera está conectada e acessível pelo sistema. Caso não seja possível estabelecer comunicação, o programa encerra a execução para evitar falhas posteriores.

- **Captura contínua de frames** – Em um loop principal, a câmera captura imagens constantemente. Cada frame representa um instante do ambiente à frente do carro, permitindo análises sequenciais para detectar a pista.
- **Visualização e encerramento seguro** – Para testes, os frames podem ser exibidos em uma janela de visualização. Além disso, são implementadas rotinas de encerramento que liberam os recursos da câmera e fecham todas as janelas, garantindo a integridade do sistema.

Essa etapa foi essencial para:

- confirmar a compatibilidade da câmera com a Raspberry Pi;
- avaliar a capacidade do hardware em processar imagens em tempo real;
- preparar o ambiente para integrar a captura de vídeo aos módulos de detecção de pista, cálculo de erro e controle de motores.

O código detalhado e comentado desta etapa está disponível no repositório do projeto no **GitHub**, possibilitando que qualquer interessado compreenda a implementação completa e execute os testes de captura de vídeo em seu próprio ambiente.

3.6.4 Segunda etapa – Controle dos atuadores

Após validar a captura de vídeo, a etapa seguinte foi desenvolver o controle dos **atuadores** do carro, especificamente o **servo de direção** e o **motor**, utilizando sinais PWM gerados pelo módulo **PCA9685** conectado à Raspberry Pi. Esse módulo permite criar sinais PWM precisos, essenciais para o controle exato de motores e servos em projetos de robótica.

O processo envolveu as seguintes etapas:

1. **Configuração da interface I2C** – A comunicação entre a Raspberry Pi e o PCA9685 é realizada via barramento I2C, usando os pinos SCL e SDA da placa. Essa interface permite enviar comandos digitais que determinam a largura de pulso do PWM em cada canal do módulo.
2. **Inicialização do PCA9685 e definição da frequência do PWM** – O módulo é configurado para operar com uma frequência de 50 Hz, padrão de servomotores, equivalente a um ciclo de 20 milissegundos. Cada canal do PCA9685 pode gerar sinais PWM independentes, permitindo controlar vários atuadores simultaneamente.
3. **Mapeamento dos canais para os atuadores** – O canal 0 foi destinado ao ESC que controla o motor, e o canal 1 ao servo de direção. Cada sinal PWM define a posição do servo ou a velocidade do motor de acordo com a largura do pulso, que

varia entre valores mínimos e máximos correspondentes ao intervalo de movimento permitido pelos atuadores.

4. **Definição dos valores de referência** – Valores específicos de duty cycle foram atribuídos como referência para os atuadores: por exemplo, para manter o motor parado ou posicionar o servo no centro. Esses valores representam a largura do pulso dentro de um ciclo PWM de 16 bits e servem como base para ajustes durante a operação do carro autônomo.

Essa etapa foi crucial para integrar o controle de hardware ao sistema de visão computacional. Com o PCA9685, a Raspberry Pi consegue enviar sinais confiáveis e precisos aos atuadores, garantindo que as decisões tomadas a partir da análise de vídeo sejam traduzidas em movimentos reais do carro.

O código detalhado e comentado desta etapa está disponível no repositório do projeto no **GitHub**, permitindo que interessados compreendam a implementação completa e realizem testes de controle dos atuadores em seus próprios ambientes.

3.6.4.1 Código para Identificação dos Valores de PWM para Controle

Durante a implementação do sistema de controle, foi observada uma **discrepância entre os valores de duty cycle esperados e os efetivamente interpretados pelo ESC**. Ou seja, os valores teóricos de PWM não resultavam nos comportamentos esperados do motor, enquanto o servo de direção respondeu de forma consistente. Essa diferença indica que diferentes modelos de ESC podem interpretar o sinal PWM de forma ligeiramente distinta.

Para garantir a operação confiável do motor, desenvolveu-se um **procedimento de calibração prática**, que consiste em variar sistematicamente os valores de PWM enviados ao ESC e observar a resposta do motor. O objetivo é identificar três pontos críticos:

1. **Neutro** – valor do duty cycle em que o motor permanece parado.
2. **Avanço** – valor mínimo que faz o motor girar para frente.
3. **Recuo** – valor mínimo que faz o motor girar no sentido inverso.

A lógica do procedimento consiste em enviar incrementalmente diferentes valores de PWM e monitorar o comportamento do motor, registrando os pontos em que ocorrem as respostas desejadas. Durante os testes, foi importante incluir tempos de espera entre alterações de sinal, permitindo que o ESC e o motor reajam de forma estável antes da próxima mudança.

Essa etapa de calibração demonstrou-se **crítica para a estabilidade do sistema**, garantindo que os comandos gerados pelo controle da Raspberry Pi sejam corretamente

interpretados pelo ESC. Além disso, evidencia a importância de documentar o comportamento específico de cada modelo de ESC, pois pequenas diferenças podem impactar diretamente no desempenho do projeto.

O código completo e comentado da calibração está disponível no repositório do projeto no **GitHub**, permitindo que interessados realizem testes semelhantes e adaptem os parâmetros para seus próprios sistemas.

3.6.4.2 Código para Validação da Largura do Pulso PWM

Após a calibração inicial descrita na Seção 3.6.4.1, tornou-se necessário **validar os valores de duty cycle identificados como representativos para os diferentes estados do motor**, incluindo parada, velocidade mínima, média e máxima. Essa etapa garante que os sinais PWM enviados pelo controlador correspondam de fato ao comportamento esperado do motor.

Para isso, foi desenvolvido um **procedimento de validação manual**, em que o operador insere valores de duty cycle dentro do intervalo permitido (0 a 65535) e observa a resposta física do motor em tempo real. Este método permite confirmar com precisão:

1. O valor correto para **parada do motor**.
2. Valores que geram **movimento em baixa rotação**.
3. Valores que produzem **aceleração plena**.
4. Identificação de possíveis **zonas mortas** ou insensibilidade nos extremos do intervalo de PWM.

O processo é interativo e experimental, oferecendo ao operador **feedback visual imediato**, essencial para ajustes finos do sistema. A estrutura simples do procedimento torna-o adequado para testes rápidos em bancada e diagnósticos do funcionamento do ESC.

Além disso, a **fixação da frequência do PCA9685 em 50 Hz** garante compatibilidade com a maioria dos ESCs comerciais, que esperam pulsos PWM com período de 20 ms. Essa configuração assegura que os sinais enviados sejam interpretados corretamente pelo controlador eletrônico, evitando comportamentos inesperados do motor.

A modularidade da abordagem permite que a validação seja adaptada para diferentes canais do PCA9685 ou outros tipos de atuadores, oferecendo flexibilidade para futuras expansões do sistema.

A validação prática do PWM complementa a calibração inicial, **assegurando que o carro autônomo opere de maneira confiável e previsível**, tanto em velocidade quanto em resposta a comandos de direção. O código completo e comentado do procedimento de validação está disponível no repositório do projeto no **GitHub**, permitindo que outros

pesquisadores ou desenvolvedores reproduzam e adaptem os testes conforme necessário.

3.6.5 Integração visão e controle

Com os dois sistemas principais devidamente testados e operacionais tanto o **controle de sinais PWM** para o motor e servo via PCA9685, quanto a **captura de imagem em tempo real** utilizando a câmera USB e a biblioteca OpenCV, foi possível avançar para a etapa de integração.

O objetivo desta fase foi verificar se a Raspberry Pi seria capaz de **executar simultaneamente o controle de atuadores e o processamento de vídeo**, mantendo desempenho adequado sem apresentar conflitos de hardware, travamentos ou perda significativa de frames.

Durante os testes, a Raspberry Pi manteve a comunicação **I²C ativa com o PCA9685** para o envio dos sinais de PWM, enquanto realizava a **captura contínua de vídeo** e a exibição dos frames. Essa experiência confirmou que a placa possui capacidade suficiente para tarefas concorrentes de hardware e processamento de imagem, mesmo com duas bibliotecas distintas em execução.

Apesar de ainda não existir controle em tempo real do veículo com base na análise visual nesta etapa, o teste de integração demonstrou que o sistema embarcado consegue operar de forma **estável e responsiva**. A ausência de travamentos ou latência perceptível reforça a viabilidade de implementar, em etapas posteriores, a **lógica de controle autônomo de direção e velocidade** baseada nas imagens capturadas.

Este sucesso na integração representa um **marco fundamental** para o desenvolvimento de um sistema autonomo autônomo para um protótipo em escala 1:10, pois garante que a infraestrutura de hardware e software pode suportar a complexidade do controle simultâneo de visão computacional e atuadores, servindo como base confiável para etapas mais avançadas do projeto.

Para aqueles interessados nos detalhes da implementação, incluindo inicialização da câmera, configuração do PCA9685 e loop de captura, o **código completo está disponível no repositório do projeto no GitHub**, permitindo acompanhamento prático, experimentação e adaptação por outros desenvolvedores ou pesquisadores.

3.6.6 Tratamento de Imagem para Detecção de Linha

Após a integração bem-sucedida entre os módulos de captura de imagem e controle PWM, decidiu-se concentrar, neste estágio, exclusivamente na etapa de percepção visual. A intenção foi aprofundar o estudo do processamento de imagem em

tempo real, isolando variáveis associadas ao movimento e permitindo foco total na interpretação da imagem capturada pela câmera.

3.6.6.1 Etapas do Algoritmo e Conceitos Utilizados

O algoritmo desenvolvido segue um pipeline clássico de visão computacional, adaptado às características do projeto e à necessidade de operação em tempo real:

1. **Captura de Imagem:**
2. **Pré-processamento:**
3. **Segmentação e Análise Multi-ROI:**
4. **Visualização para Depuração:**

Este pipeline, implementado em Python com OpenCV e NumPy, fornece uma **base robusta para a lógica de controle do veículo**, permitindo que o servo e o motor ajustem direção e velocidade de forma responsiva, com base exclusivamente nas informações obtidas pela visão computacional.

3.6.6.2 Conversão para Escala de Cinza

O primeiro passo no pré-processamento do frame capturado consiste em converter a imagem colorida do formato BGR, utilizado pelo OpenCV, para escala de cinza. Essa transformação reduz a complexidade da imagem, mantendo apenas a informação de intensidade luminosa dos pixels, essencial para técnicas de detecção de bordas e análise estrutural da pista.

A conversão para escala de cinza é realizada com a função cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY), que processa cada pixel, combinando os valores de azul, verde e vermelho em um único valor de intensidade. Essa abordagem simplifica subsequentemente a aplicação do filtro de desfoque e do detector de bordas Canny, além de reduzir o custo computacional em comparação com operações diretas em imagens coloridas.

Além disso, trabalhar em escala de cinza permite que o algoritmo de detecção de contornos e cálculo de centro das faixas seja mais robusto frente a variações de cor da pista ou do ambiente, concentrando-se exclusivamente nas variações de intensidade, que correspondem à presença de faixas brancas em fundo mais escuro.

Essa etapa é, portanto, crucial para garantir que o pipeline de visão computacional opere de maneira eficiente e estável, fornecendo uma base confiável para as operações subsequentes de detecção de bordas, filtragem de ruído e extração de informações geométricas da pista.

3.6.6.2 Desfoque Gaussiano

Após a conversão para escala de cinza, aplica-se o **desfoque Gaussiano** como etapa de pré-processamento. Essa técnica suaviza a imagem, reduzindo ruídos e pequenas variações de intensidade que poderiam interferir na detecção de bordas ou gerar falsos contornos.

O desfoque é implementado utilizando a função cv2.GaussianBlur(gray, (5, 5), 0), onde o primeiro parâmetro representa a imagem em escala de cinza, o segundo define o tamanho do kernel (5x5 neste caso) e o terceiro é o desvio padrão da distribuição Gaussiana, configurado para 0, permitindo que o OpenCV calcule automaticamente o valor ideal.

Essa operação possui dois objetivos principais:

1. **Redução de ruído:** elimina pequenas imperfeições e pixels isolados que não fazem parte da faixa da pista.
2. **Suavização de bordas:** prepara a imagem para a aplicação do detector de bordas Canny, tornando as transições de intensidade mais consistentes e evitando a detecção de múltiplos contornos artificiais.

O desfoque Gaussiano é fundamental para melhorar a precisão do módulo de visão, garantindo que apenas os contornos relevantes, aqueles correspondentes às faixas brancas da pista, sejam identificados nas etapas subsequentes. Essa etapa contribui significativamente para a estabilidade e confiabilidade do sistema de visão do veículo.

3.6.6.3 Detecção de Bordas com Canny

Após aplicar o desfoque Gaussiano, o próximo passo do pré-processamento consiste na **detecção de bordas utilizando o algoritmo Canny**. Essa técnica identifica regiões de transição intensa de brilho na imagem, correspondentes às bordas da faixa branca, fornecendo informações essenciais para o cálculo do centro da pista.

A detecção é realizada com a função cv2.Canny(blur, 50, 150), onde blur é a imagem suavizada pelo desfoque Gaussiano, e os valores 50 e 150 representam os thresholds mínimo e máximo, respectivamente. Esses thresholds determinam quais gradientes de intensidade serão considerados como bordas, controlando a sensibilidade do detector.

Para aumentar a robustez da detecção, especialmente frente a ruídos residuais ou pequenas imperfeições na pista, a máscara resultante é processada com operações morfológicas básicas.

O uso do Canny, aliado às operações morfológicas, aumenta a precisão da percepção visual do sistema, permitindo que o sistema identifique corretamente a posição da pista mesmo em condições de iluminação variáveis ou superfícies parcialmente desgastadas.

Essa combinação de detecção de bordas com pós-processamento morfológico gera uma **imagem binária limpa**, onde as bordas da faixa branca são realçadas e os ruídos removidos.

3.6.6.4 Operações Morfológicas (Dilatação e Erosão)

Após a detecção de bordas com o algoritmo Canny, a imagem binária resultante ainda pode conter pequenos ruídos ou lacunas nas regiões correspondentes à faixa branca.

Para refinar a máscara e torná-la mais confiável para a detecção de contornos, aplicam-se **operações morfológicas**, especificamente **dilatação** e **erosão**.

- **Dilatação (cv2.dilate)**: Expande os pixels brancos na imagem, preenchendo pequenas lacunas e conectando regiões fragmentadas da faixa. Isso é útil para garantir que as bordas detectadas formem contornos contínuos, facilitando a extração do centro geométrico da faixa.
- **Erosão (cv2.erode)**: Remove pixels isolados ou pequenos grupos de ruído branco que não pertencem à faixa, refinando a máscara e preservando apenas as regiões relevantes.

No código, essas operações são realizadas utilizando um **kernel de 3x3** e geralmente aplicadas em sequência (dilatação seguida de erosão ou abertura/fechamento), de acordo com a necessidade de remover ruídos ou preencher falhas na faixa.

O resultado é uma **máscara binária limpa e bem definida**, onde a faixa branca da pista está destacada de forma contínua. Essa máscara serve como entrada para a etapa seguinte de **detecção de múltiplos centros (Multi-ROI)**, permitindo calcular de maneira precisa o desvio lateral do veículo em relação à pista.

A aplicação dessas operações morfológicas garante maior **robustez e estabilidade** do sistema de visão computacional, permitindo que o carro mantenha a trajetória mesmo em condições adversas, como sombras, reflexos ou pequenas imperfeições na pista.

3.6.6.5 Definição das Regiões de Interesse (Multi-ROI)

Para tornar a detecção da faixa branca mais estável e confiável, cada frame capturado é subdividido em múltiplas **Regiões de Interesse (ROIs)** horizontais. Cada ROI cobre uma faixa do campo de visão, permitindo que o sistema analise diferentes profundidades da pista de forma independente.

Objetivo das Multi-ROIs:

- Monitorar a posição da faixa em diferentes distâncias à frente do veículo.
- Reduzir a influência de ruídos ou sombras localizadas em apenas uma região do frame.

- Permitir cálculos mais precisos do centro geométrico da pista para controle de direção.

Implementação:

- O frame processado é dividido em números de faixas horizontais de mesma altura.
- Cada ROI é processada individualmente para detectar contornos e calcular o **centroide da faixa**.
- Para aumentar a precisão, a função recebe como parâmetro o **centro da faixa no frame anterior**, limitando a busca em cada ROI a uma janela próxima do centro anterior (margem), tornando o acompanhamento mais estável.

Benefícios:

- Antecipação de reconhecimento: ROIs mais distantes permitem prever alterações na trajetória da pista.
- Robustez: caso uma ROI não detecte a faixa (por sombras ou interrupções), as outras ainda fornecem informações confiáveis.
- Estabilidade do controle: a média dos centros detectados em todas as ROIs fornece um **erro médio** mais consistente para ajustar o servo e o motor.

Essa abordagem de **detecção em múltiplas regiões** é fundamental para sistemas autônomos, garantindo que o algoritmo de direção mantenha o veículo alinhado à pista de forma precisa, mesmo em situações adversas de iluminação ou pequenas irregularidades na faixa.

3.6.6.6 Detecção de Contornos em Cada ROI

Após a definição das Regiões de Interesse (ROIs), o próximo passo é identificar os contornos correspondentes à faixa branca em cada região. Essa etapa é essencial para extrair informações espaciais precisas, que servirão para calcular a posição da pista em relação ao veículo.

Processo de Detecção de Contornos:

Aplicamos a função cv2.findContours(), onde cada ROI binarizada é analisada para identificar os contornos presentes, fazendo com que apenas **contornos externos (**enquanto utilizado o RETR_EXTERNAL) são considerados, evitando múltiplas detecções de pequenos ruídos ou fragmentos irrelevantes.

Seleção do contorno principal:

Entre todos os contornos detectados em uma ROI, seleciona-se o **maior contorno por área**, assumindo que ele representa a faixa principal da pista, consequentemente gerando maior precisão ao solicitar respostas dos atuadores para movimentação do veículo.

Cálculo do centroide:

Para o contorno selecionado, calcula-se o **centro geométrico** (centroide). O centroide representa a posição média da faixa dentro daquela ROI especifica, a figura 19 ilustra o cálculo do centroide utilizado.

Figura 19 - Cálculo do centroide

$$cx = \frac{M["m10"]}{M["m00"]}, \quad cy = \frac{M["m01"]}{M["m00"]}$$

Fonte: Adaptada pelo autor (2025)

Onde M são os momentos do contorno.

Vantagens da abordagem por contornos:

- **Precisão**: permite identificar exatamente onde a faixa está localizada em cada ROI.
- **Robustez a ruídos**: descarta pequenas manchas ou pixels isolados, mantendo apenas a faixa principal.
- **Base para cálculo do erro**: os centros detectados serão utilizados posteriormente para determinar o desvio lateral da pista em relação ao veículo, permitindo ajustes no servo de direção.

Essa etapa de **detecção de contornos em múltiplas ROIs**, tende a garantir que o sistema de visão tenha informações detalhadas e confiáveis sobre a posição da pista, mesmo em condições de iluminação variáveis ou com faixas parcialmente interrompidas.

3.6.6.7 Cálculo dos Centros das Faixas

Após a detecção dos contornos em cada ROI, o próximo passo é determinar os **centros geométricos das faixas**, que servirão como referência para o cálculo do desvio lateral da pista em relação ao veículo.

Processo de Cálculo dos Centros:

- **Extração dos centros por ROI**: Para cada contorno principal identificado em uma ROI, calcula-se o **centroide horizontal** (cx) usando os momentos da imagem, onde uma vez que foi identificado o contorno dominante, aplica-se o cálculo de momentos da imagem para determinar o **centroide**, que representa a posição horizontal média da faixa naquela região.

Os momentos proporcionam informações estatísticas sobre a distribuição dos pixels do contorno, permitindo calcular o centro geométrico mesmo em casos em que a

forma da faixa não é perfeitamente simétrica, a figura 20 ilustra o calculo dos centros de faixa.

Figura 20 – Fórmula do cálculo dos centros das faixas

$$cx = \frac{M_{10}}{M_{00}}$$

Fonte: Autor, 2025

Onde M10 e M00 são os momentos do contorno.

O valor de cx indica a posição horizontal média da faixa dentro daquela ROI.

- **Registro e armazenamento**: Todos os centros calculados nas diferentes ROIs são armazenados em uma lista ou vetor, mantendo a ordem das regiões do frame (dá mais próxima à mais distante).
- **Média dos centros**: Para aumentar a estabilidade da detecção podemos calcular a **média dos centros** das ROIs que possuem detecção válida utilizando o calculo ilustrado na figura 21.

Figura 21 – Fórmula do cálculo da média dos centros

$$centro\_medio = \frac{\sum cx_i}{N}$$

Fonte: Autor, 2025

Onde N é o número de ROIs com contornos detectados.

Este valor médio fornece uma estimativa robusta da posição lateral da pista em relação ao centro do veículo, suavizando variações bruscas que podem ocorrer em apenas uma ROI devido a sombras ou pequenos ruídos.

Benefícios do cálculo dos centros:

- **Robustez**: a média dos centros reduz o impacto de detecções isoladas ou ruídos.
- **Base para controle**: os centros servem diretamente como entrada para o algoritmo de controle do servo, permitindo ajustes precisos de direção.
- **Antecipação de curvas**: ao analisar centros em múltiplas ROIs, o sistema consegue prever mudanças de trajetória, garantindo condução mais estável e segura.

Essa etapa consolida a informação visual em **uma medida quantitativa**, que será utilizado no controle do carro autônomo para manter a faixa centralizada no campo de visão da câmera.

3.6.6.8 Cálculo do Erro em Relação ao Centro da Imagem

Após determinar os centros das faixas em cada ROI, o próximo passo é calcular o **erro lateral**, que representa o desvio da pista em relação ao **centro do campo de visão da câmera**. Esse erro constitui a variável de entrada principal para o controle do servo e do motor, permitindo que o carro se mantenha alinhado à pista.

Definição de Referência:

- **Centro da imagem**: posição horizontal central do frame capturado pela câmera. Para um frame de largura L, calculamos conforme ilustrado na figura 22:

Figura 22 – Fórmula do cálculo do centro da imagem

$$centro\_imagem = \frac{L}{2}$$

Fonte: Autor, 2025

- **Cálculo do Erro**: Para cada frame processado, calcula-se a diferença entre o centro médio da faixa (ou o centro detectado em cada ROI, se desejado) e o centro da imagem, como ilustrado na figura 23.

Figura 23 – Fórmula do cálculo do erro

$$erro = centro\_medio - centro\_imagem$$

Fonte: Autor, 2025

Um valor positivo indica que a faixa está deslocada para a direita do centro da câmera, enquanto um valor negativo indica deslocamento para a esquerda. Tais valores serão utilizados para movimentação da direção.

Considerações Adicionais:

Além do cálculo direto do erro, **podemos utilizar também** algumas estratégias complementares para aprimorar a estabilidade e a precisão do controle de direção:

- **Média ou filtro de erro**: pode-se utilizar uma média móvel ou fila de erros recentes para suavizar flutuações momentâneas, aumentando a estabilidade do controle.
- **Memória do erro**: em situações onde a faixa não é detectada temporariamente (frames perdidos), é possível manter o último erro válido para guiar o carro, evitando movimentos abruptos ou perda de pista.

- **Escalonamento para controle**: o valor de erro é mapeado para sinais PWM do servo motor, determinando o ângulo de direção necessário para centralizar a faixa.

Finalidade:

- Transformar a informação visual em **uma medida quantitativa de desvio lateral**, que pode ser diretamente utilizada pelo algoritmo de controle para ajustar a direção do veículo de forma contínua e responsiva.
- Permite que o carro antecipe curvas e mantenha alinhamento mesmo em condições adversas de iluminação ou em pistas com faixas parcialmente interrompidas.

3.6.6.9 Fila de Erros e Média Móvel para Estabilidade

Para garantir um comportamento mais suave e previsível na direção do veículo, foi implementado um **mecanismo de filtragem temporal** baseado em uma **fila circular de erros** (estrutura *deque* do Python). Essa técnica tem o objetivo de reduzir oscilações causadas por pequenas variações momentâneas na detecção da linha.

Princípio de Funcionamento:

A cada frame processado, o erro calculado entre o centro da faixa e o centro da imagem é armazenado em uma fila de tamanho fixo (por exemplo, 5 posições). Quando a fila atinge sua capacidade máxima, o valor mais antigo é automaticamente descartado, mantendo sempre os erros mais recentes.

Cálculo da Média Móvel:

A média dos valores armazenados na fila é utilizada como erro efetivo para o controle de direção, a figura 24 ilustra o calculo de media móvel.

Figura 24 – Fórmula do cálculo da média móvel

$$erro\_medio = \frac{\sum erro_i}{n}$$

Fonte: Autor (2025)

Onde n é o número de elementos atualmente na fila.

Essa abordagem reduz ruídos e suaviza as variações abruptas, permitindo que o servo motor realize correções progressivas, em vez de movimentos bruscos.

Vantagens da Estratégia:

- **Estabilidade** – A média móvel atenua flutuações causadas por imperfeições na detecção de contornos, sombras ou reflexos.

- **Robustez** – Em casos de perda momentânea da linha (por exemplo, quando a faixa é interrompida ou está parcialmente fora do campo de visão), o sistema mantém temporariamente o último erro válido.
- **Tempo de resposta controlado** – O tamanho da fila define o equilíbrio entre suavidade e rapidez de reação. Um valor pequeno responde mais rápido, enquanto um valor maior garante mais estabilidade.

Resultado Prático:

A utilização da média móvel proporciona **um controle de direção contínuo e equilibrado**, essencial para o deslocamento fluido do carro em pista, especialmente em curvas ou sob condições de iluminação variáveis.

3.6.6.10 Visualização para Depuração

Durante o desenvolvimento e validação do algoritmo de detecção de linha, foi implementado um **módulo de visualização em tempo real** com o objetivo de facilitar o processo de depuração e ajuste dos parâmetros de visão computacional. Essa visualização permite ao operador compreender de forma imediata como o sistema interpreta a imagem capturada e quais etapas do processamento estão sendo aplicadas.

Objetivo da Visualização:

O principal propósito dessa ferramenta é **fornecer feedback visual** sobre o comportamento do algoritmo, permitindo verificar, por exemplo:

- Se a segmentação da linha está ocorrendo corretamente.
- Se as regiões de interesse (ROIs) estão sendo delimitadas conforme o esperado.
- Se os contornos e centros das faixas estão sendo identificados adequadamente.
- Se o cálculo do erro em relação ao centro da imagem está coerente.

Elementos Exibidos:

A imagem final exibida na janela de depuração é o resultado da sobreposição de várias camadas de informação sobre o frame original, incluindo:

- As **regiões de interesse** delimitadas por retângulos coloridos.
- Os **contornos detectados** em cada ROI, geralmente representados em verde.
- Os **pontos centrais das faixas**, destacados por pequenos círculos.
- O **centro da imagem**, indicado por uma linha vertical amarela.
- O **erro calculado**, visualizado como a distância horizontal entre o centro da imagem e o centro da faixa detectada.

3.6.6.11 Integração com o Controle de Servo e Motor

Após o processamento da imagem e o cálculo do erro lateral em relação ao centro da pista, o valor obtido é utilizado para acionar os atuadores responsáveis pela direção e movimentação do veículo.

Essa etapa representa a **integração entre o módulo de visão computacional e o módulo de controle físico**, transformando as informações visuais em ações concretas que mantêm o carro autônomo alinhado à pista.

Lógica de Controle de Direção (Servo Motor):

O erro calculado pela visão computacional representa o desvio lateral da faixa em relação ao centro da imagem. Esse valor é fundamental para o controle do servo, pois indica se o veículo deve virar à direita ou à esquerda para manter-se alinhado com a pista. Para isso, aplica-se um controle proporcional, onde o ângulo de direção varia de acordo com a magnitude do erro detectado. A seguir a figura 25 apresenta a formula do erro convertido para o ângulo de direção.

Figura 25 - Fórmula do erro convertido para angulo de direção

$$\text{angulo\_direcao} = \text{angulo\_base} + (K_p \times \text{erro})$$

Fonte: Autor (2025)

Definição dos termos

- **angulo_base:** posição neutra do servo, correspondente ao carro alinhado em linha reta. Esse valor serve como referência inicial para o cálculo.
- **Kp:** constante de proporcionalidade que define o quanto o carro reage ao erro. Valores altos tornam o sistema mais responsivo (porém mais instável), enquanto valores baixos deixam o veículo mais suave (porém lento para corrigir desvios).
- **erro:** diferença entre o centro médio da faixa e o centro da imagem, conforme calculado na etapa anterior.

Conversão para PWM e atuação no servo

Após o cálculo do ângulo, o valor é convertido em **pulso PWM**, que é enviado para o canal configurado do módulo PCA9685. Esse módulo gera sinais precisos para controlar o servo, permitindo ajustes finos de direção.

O servo motor então gira proporcionalmente ao erro, realizando correções contínuas na trajetória e mantendo o veículo centralizado.

Lógica de Controle de Velocidade (Motor/ESC):

No sistema desenvolvido, a velocidade do veículo é mantida em um valor **baixo e constante**, garantindo estabilidade durante o seguidor de linha e reduzindo o risco de perda da pista. Essa abordagem simplifica o controle de tração e assegura que o foco principal do sistema seja a precisão da direção.

Mesmo operando em velocidade fixa, algumas condições lógicas são aplicadas para aumentar a segurança e evitar comportamentos inesperados

- **Faixa detectada corretamente:** quando a pista é identificada de forma estável, o veículo mantém sua velocidade constante, permitindo um deslocamento controlado e seguro.
- **Detecção incerta da faixa:** caso a visão computacional identifique falhas momentâneas como ruído, sombras ou perda parcial da linha, o sistema pode reduzir a velocidade ou manter o valor mínimo operacional, permitindo maior tempo para que o algoritmo recupere a trajetória.
- **Pista não detectada:** se nenhuma faixa for identificada, o sistema emite **um comando de parada**, evitando que o veículo siga sem referência visual.

O controle do ESC é realizado por meio de **sinal PWM** gerado pelo módulo PCA9685, utilizando os valores de *duty cycle* previamente calibrados para os estados de **marcha à ré, parado e frente**. Como a velocidade é fixa nesse projeto, o PWM correspondente à velocidade desejada é aplicado continuamente, exceto nos casos de redução ou parada descritos acima.

Ciclo de Controle em Tempo Real:

Todo o processo, da captura do frame até o acionamento dos atuadores, ocorre dentro de um **loop contínuo**, que se repete várias vezes por segundo.

O ciclo completo segue a seguinte sequência:

- Captura da imagem pela câmera.
- Processamento e detecção da faixa branca.
- Cálculo do erro em relação ao centro.
- Aplicação do controle proporcional no servo.
- Ajuste dinâmico da velocidade no motor.
- Retorno ao início do ciclo.

Essa repetição contínua garante que o veículo reaja de forma imediata às mudanças na pista, mantendo um comportamento autônomo fluido e estável.

Importância da Integração:

Essa integração entre visão e controle fecha o **ciclo de percepção–decisão–ação**, característica essencial dos sistemas autônomos. A precisão na comunicação entre o módulo de visão e os atuadores determina diretamente a estabilidade e a eficiência da navegação. A latência mínima entre a leitura da câmera e a atualização dos sinais PWM é fundamental para evitar atrasos perceptíveis, garantindo que o carro responda em tempo real às variações do trajeto.

Considerações Finais:

A integração do processamento de imagem com o controle dos atuadores consolidou o primeiro sistema autônomo funcional do projeto, capaz de detectar a faixa branca e ajustar sua trajetória automaticamente.

Esse estágio marca a transição entre o **sistema de percepção isolado** e o **sistema autônomo integrado**, em que a visão computacional passa a ter influência direta sobre o movimento do robô.

Essa arquitetura serviu de base para as versões seguintes, nas quais foram incorporados ajustes de estabilidade, filtragem do erro e aprimoramentos de resposta dinâmica, resultando em um comportamento de navegação mais natural e confiável.

Considerações sobre Robustez e Limitações

O pipeline de visão computacional desenvolvido demonstrou desempenho satisfatório nas condições controladas de teste, oferecendo boa estabilidade na detecção da faixa branca e resposta adequada aos comandos de direção e velocidade. No entanto, como qualquer sistema de percepção visual baseado exclusivamente em câmeras RGB, sua robustez apresenta limitações inerentes às variações do ambiente e às características físicas da pista.

Robustez em Condições Controladas

Durante os testes em pista com iluminação moderada e contraste bem definido entre a faixa branca e o fundo escuro, o sistema manteve **detecção estável** e **resposta consistente**.

A combinação das técnicas empregada garantiu:

- Resistência a pequenas variações de luminosidade e sombras suaves;
- Suavidade no controle de direção, reduzindo oscilações;

- Estabilidade nas curvas e em trechos retilíneos;
- Baixa taxa de falsos positivos durante a segmentação.

Esses resultados confirmam que o pipeline implementado é **robusto o suficiente para navegação autônoma em ambientes controlados**.

Limitações Identificadas

Apesar dos resultados positivos, algumas limitações foram observadas em situações mais desafiadoras:

- **Variações bruscas de luz:** reflexos intensos, luz solar direta ou sombras profundas podem alterar significativamente os valores de brilho e saturação, comprometendo a segmentação.
- **Faixas tracejadas ou desgastadas:** a ausência temporária de trechos contínuos da linha pode gerar perda momentânea de referência, mesmo com o uso da fila de erros.
- **Superfícies reflexivas ou brilhantes:** em alguns casos, o brilho do piso é interpretado erroneamente como parte da faixa branca.
- **Latência de processamento:** embora o sistema opere próximo de 30 fps, a execução de múltiplos filtros e operações morfológicas pode reduzir o desempenho em processadores menos potentes, como a Raspberry Pi 4, afetando o tempo de resposta em situações de curvas rápidas.
- **Dependência de parâmetros calibrados manualmente:** ajustes de thresholds HSV, intensidade de brilho e constantes de controle exigem calibração cuidadosa para cada ambiente.

Melhorias Futuras e Estratégias para Aumento de Robustez

Para mitigar essas limitações e tornar o sistema mais confiável em cenários variados, podem ser adotadas as seguintes estratégias:

- **Ajuste dinâmico de parâmetros:** implementação de um sistema de calibração automática de thresholds HSV com base na luminosidade ambiente.
- **Filtro temporal adaptativo:** ponderar o erro de detecção com base na confiança da segmentação, reduzindo oscilações em momentos de incerteza.
- **Uso de aprendizado de máquina leve:** redes neurais compactas (como MobileNet ou TinyML) podem ser treinadas para reconhecer a faixa independentemente da iluminação.

- **Integração com sensores adicionais:** o uso de sensores ultrassônicos ou de profundidade pode auxiliar na detecção de limites da pista e obstáculos, complementando a visão computacional.
- **Paralelismo no processamento:** otimizar o código para executar partes do pipeline em threads separadas, reduzindo latência e mantendo desempenho em tempo real.

Conclusão da Seção

O módulo de visão computacional, embora sujeito a limitações típicas de sistemas baseados em imagem, demonstrou ser **tecnicamente consistente e suficientemente robusto** para a aplicação proposta no estágio inicial do projeto.

Ele forneceu uma **base sólida para a integração com o controle de atuadores**, permitindo que o veículo autônomo execute navegação autônoma confiável em ambientes controlados e sirva como ponto de partida para aprimoramentos futuros.

A análise crítica das limitações e das possíveis melhorias orienta a evolução do sistema nas próximas etapas do desenvolvimento, consolidando o aprendizado adquirido e guiando a busca por maior **autonomia, adaptabilidade e estabilidade operacional**.

3.6.6.12 Limitações Observadas e Transição para o Sistema Final

Apesar de funcional, a lógica atual apresenta limitações frente a cenários mais complexos, como curvas muito fechadas, bifurcações ou ausência temporária da linha. Além disso, o controle não utiliza um modelo dinâmico completo (como um PID), o que pode causar oscilações em alta velocidade.

Esses fatores motivaram a evolução para um sistema de navegação mais sofisticado, apresentado na próxima seção, com foco em planejamento adaptativo, segmentação de pista com múltiplas classes e estratégias reativas mais inteligentes.

3.7 Código final do sistema – seguidor de pista

O desenvolvimento do código final representou uma evolução significativa em relação ao sistema básico descrito na Seção 3.6. Enquanto a versão inicial baseava-se na detecção de linhas brancas utilizando **detecção de bordas Canny, múltiplas regiões de interesse (Multi-ROI) e controle proporcional simples**, a versão final incorpora técnicas mais avançadas de visão computacional e estratégias de controle robustas, capazes de lidar com cenários de navegação mais complexos.

3.8 Principais avanços em relação ao sistema inicial:

- **Análise geométrica de contornos e cálculo refinado do centro da pista:** A detecção de bordas foi aprimorada para suportar situações de iluminação variável e pistas com faixas parcialmente interrompidas.

- **Implementação de máquina de estados:** Permite que o veículo adapte seu comportamento a diferentes situações, como curvas acentuadas, retas e perdas temporárias de linha.
- **Predição polinomial do trajeto:** Mantém a trajetória mesmo em períodos curtos de perda de detecção, suavizando o controle e prevenindo desvios abruptos.
- **Estratégias reativas para intersecções e recuperação de trajetória:** O sistema detecta rapidamente mudanças no padrão da pista e ajusta o movimento do carro sem intervenção externa.
- **Sistema de monitoramento via interface web:** Permite visualização em tempo real da detecção da linha, centros calculados e status do controle, facilitando depuração e ajustes.

3.8.1 Introdução e Objetivos do código final:

O desenvolvimento do código final representou a consolidação de todo o trabalho anterior, integrando protótipos e testes isolados em um sistema completo e robusto. Este capítulo descreve detalhadamente a implementação do software que permite ao veículo autônomo navegar de forma estável e segura em uma pista demarcada, mesmo em cenários com iluminação variável, faixas interrompidas ou intersecções complexas.

3.8.2 Objetivos principais do código final:

Os principais objetivos do sistema final é:

- Integrar todos os módulos desenvolvidos anteriormente em uma única aplicação coesa.
- Implementar algoritmos de visão computacional robustos, utilizando **detecção de cor, análise de contornos e predição polinomial**, garantindo detecção confiável da pista.
- Desenvolver estratégias de controle adaptativo baseadas em **máquina de estados, suavização EWMA e busca oscilatória**, permitindo respostas apropriadas a diferentes cenários operacionais.
- Garantir estabilidade e segurança do veículo, tratando perdas temporárias de linha e recuperações de trajetória.
- Prover ferramentas de monitoramento via interface web, exibindo em tempo real **o processamento de imagem, centros estimados e estados do sistema** para validação e depuração.

3.8.3 Arquitetura do Sistema e comparação com a versão inicial

A evolução do sistema entre a versão 3.6 e 3.7 demonstra avanços significativos em termos de robustez, capacidade de adaptação e complexidade do controle.

Além disso, a nova versão incorpora estratégias de filtragem, tratamento de casos extremos, mecanismos de redundância e ajustes mais finos na lógica de decisão, permitindo que o veículo se adapte melhor a variações de iluminação, trechos com baixa visibilidade e mudanças abruptas na geometria da pista. Essa ampliação no conjunto de capacidades resulta em um sistema mais robusto, menos suscetível a falhas e capaz de operar de forma mais confiável em situações reais. A tabela a seguir resume as principais diferenças entre as duas versões:

Tabela 4 – Diferença da versão inicial para a versão final do sistema

CARACTERISTICAS	VERSÃO 3.6 (BÁSICA)	VERSÃO 3.7 (AVANÇADA)
Detecção	Segmentação de bordas com Canny + limiarização	Máscara HSV para detecção de linhas brancas, análise de contornos e bounding rectangles para identificação de faixas
Controle	Controle proporcional simples	Máquina de estados que adapta a direção e velocidade, incluindo predição polinomial e ajuste baseado em EWMA
Robustez	Memória de erro simples	Predição polinomial de linhas, suavização EWMA, busca oscilatória para recuperação de trajetória quando a pista é temporariamente perdida
Cenários	Linha contínua	Curvas acentuadas, intersecções, faixas interrompidas e perda temporária de linha
Monitoramento	Visualização básica do frame	Streaming web em tempo real, mostrando frames processados, contornos detectados, centro da pista e estados do sistema

Fonte: Autor, 2025

3.8.3.1 Visão Geral da Arquitetura

O sistema final foi organizado em **camadas interdependentes**, que operam de forma sincronizada para garantir a navegação estável do veículo. Cada camada possui responsabilidades específicas:

Camada de Percepção (Entrada)

Responsável pela aquisição e pré-processamento das informações visuais capturadas pela câmera. Inclui:

- Captura contínua de frames em resolução adequada para processamento em tempo real;
- Pré-processamento de imagem, incluindo conversão para HSV, aplicação de máscaras e operações morfológicas;

- Detecção de features relevantes, como linhas e contornos, que representam as bordas da pista.

Camada de Decisão (Processamento)

Processa os dados da camada de percepção para determinar a ação mais adequada do veículo utilizando:

- Implementação de **máquina de estados** para lidar com diferentes cenários (linha dupla, perda temporária de pista, intersecções e curvas);
- Cálculo da trajetória com base em predição polinomial e médias suavizadas (EWMA);
- Ajuste adaptativo do controle, definindo a posição do servo e a velocidade do motor de acordo com o erro da trajetória.

Camada de Atuação (Saída)

Responsável por executar fisicamente as decisões tomadas pelo sistema:

- Controle de direção via servo motor, mapeando o erro de trajetória em ajustes precisos;
- Controle de velocidade via ESC, ajustando a aceleração e desaceleração do veículo conforme a complexidade do cenário.

Camada de Monitoramento (Feedback)

Fornece ferramentas de visualização e validação em tempo real:

- Interface web que exibe frames processados, detecção de contornos, estimativa do centro da pista e estados do sistema;
- Permite depuração e ajustes finos durante os testes práticos, aumentando a confiabilidade do sistema.

3.8.3.2 Fluxo do Sistema Final

O fluxo do sistema final foi projetado para integrar de forma robusta todos os módulos, mantendo a simplicidade da versão inicial quando possível, e introduzindo melhorias significativas para aumentar a confiabilidade e adaptabilidade do veículo.

Inicialização (similar à versão 3.6)

- Configuração do hardware PCA9685 para controle do servo e ESC;
- Inicialização da câmera para captura contínua de frames;
- Definição dos parâmetros PWM para atuação segura do servo e motor.

Loop Principal (evoluído)

- Captura de frames contínua, mantendo a estratégia da versão 3.6, garantindo atualização constante do estado da pista.

Pré-processamento (modificado)

- Seleção de **ROI inferior** de 180 pixels para foco na área relevante da pista;
- Conversão do frame para o espaço de cores HSV;
- Segmentação da cor branca para detecção das bordas da pista;
- Aplicação de **operações morfológicas** (abertura e fechamento) para reduzir ruídos e melhorar a qualidade da máscara.

Detecção Avançada (substitui Multi-ROI simples)

- Identificação de **contornos** das regiões brancas;
- Filtragem por **área e proporção** para eliminar falsos positivos;
- Classificação das bordas em **esquerda** e **direita**;
- Manutenção de **histórico de pontos** para predição polinomial e suavização de trajetória.

Máquina de Estados (novo)

Permite ao veículo lidar com diferentes cenários de forma adaptativa:

- **NORMAL:** ambas as bordas detectadas;
- **SINGLE_LEFT / SINGLE_RIGHT:** apenas uma borda detectada;
- **HOLD:** mantém a trajetória anterior durante perdas temporárias de detecção;
- **PREDICT:** utiliza regressão polinomial para estimar a posição da pista;
- **SEARCH:** busca oscilatória quando a pista está perdida por vários frames;
- **TRANSVERSAL:** identificação e tratamento de intersecções e faixas transversais.

Controle Adaptativo (evoluído)

- Suavização dos erros de trajetória usando **fila de erros**;
- **Mapeamento do erro para servo** mantém lógica proporcional da versão 3.6;
- **Velocidade adaptativa** ajusta a aceleração conforme a complexidade do cenário, reduzindo a velocidade em curvas, perdas de pista ou buscas.

Atuação (similar à versão 3.6)

- Execução das ordens de controle no servo e ESC conforme os cálculos de erro e predição.

Monitoramento (novo)

- **Streaming em tempo real** via Flask, mostrando frames processados, detecção de bordas, predição polinomial e estado da máquina, permitindo depuração e validação do sistema.

3.8.4 Transição da Detecção por Cor para Análise Geométrica

A migração do método baseado em bordas para a detecção por cor foi realizada de forma gradual, mantendo a mesma arquitetura de captura e controle do veículo. O principal objetivo foi aumentar a robustez do seguidor de pista em diferentes condições de iluminação e padrões de linha.

3.8.4.1 Motivação para a Mudança

A versão 3.6 do sistema utilizava **detecção de bordas via Canny**, combinada com técnicas básicas de filtragem de ruído. Apesar de eficiente em pistas simples, essa abordagem apresentou limitações durante os testes em condições reais:

- Sensibilidade a variações de iluminação que afetavam a detecção de bordas;
- Dificuldade em lidar com linhas tracejadas ou desgastadas, causando perda momentânea da faixa;
- Instabilidade em curvas fechadas, onde as bordas não eram contínuas;
- Limitações para distinguir intersecções ou bifurcações da pista.

Para superar essas limitações, a versão 3.7 adotou **detecção baseada em cor**, explorando a faixa branca da pista como referência principal. Essa mudança trouxe vantagens significativas:

- Robustez em diferentes condições de iluminação, desde que a faixa mantivesse contraste com o piso;
- Melhor manutenção da trajetória em curvas e trechos com linhas tracejadas;
- Capacidade de reconhecer intersecções e ajustar a direção do veículo de forma mais precisa;
- Redução de falsos positivos causados por ruídos de borda ou irregularidades do piso.

Assim, a transição do método baseado em bordas para a detecção por cor permitiu consolidar um sistema mais estável e confiável, preparando o carro para o desempenho esperado em pista real.

3.8.4.2 Nova Estratégia de Detecção

Para a versão final do seguidor de pista, a detecção de linhas foi completamente reformulada, passando da abordagem baseada em Canny (Seção 3.6) para uma análise por cor combinada com filtragem geométrica de contornos.

Vantagens da nova abordagem:

- **Maior discriminação de faixas:** A combinação de filtro por área e proporção vertical reduz falsos positivos.
- **Informação estrutural:** O uso de retângulos delimitadores (bounding rectangles) fornece dados geométricos precisos para o cálculo do centro da pista.
- **Base para predição:** As coordenadas das bases dos retângulos permitem estimar trajetórias futuras, mesmo em curvas ou situações de perda parcial de faixa.

O código do sistema de detecção e controle do carro autônomo está disponível no repositório GitHub do projeto.

3.8.5 Sistema de Predição e Máquina de Estados

Para superar as limitações da versão 3.6, que possuía apenas uma memória simples de erro, a versão final introduz um sistema de predição e máquina de estados. Esse sistema permite que o carro continue navegando corretamente mesmo em situações de perda parcial da faixa ou em curvas fechadas.

3.8.5.1 Superando as Limitações da Versão 3.6

A versão 3.6 não possuía mecanismos para estimar a trajetória futura da pista, tornando o carro vulnerável a perdas temporárias de sinal. Na versão final, implementou-se:

- **Histórico de pontos das bordas da pista:** Dois deques armazenam os últimos 20 pontos detectados de cada lado da faixa (direita e esquerda).
- **Predição polinomial:** A partir desses pontos históricos, realiza-se um ajuste polinomial de grau 2, estimando a posição futura da linha em uma coordenada Y desejada.
- **Integração com máquina de estados:** Dependendo da disponibilidade de pontos detectados, o sistema alterna entre estados normais, predição de lado único, busca suave e detecção transversal. Isso garante estabilidade na navegação mesmo em situações de perda parcial da faixa.

O código completo do sistema de predição, incluindo o ajuste polinomial e a manutenção do histórico de pontos, está disponível no repositório GitHub do projeto.

3.7.4.2 Máquina de Estados para Controle Robusto

Na versão **3.6**, o controle do veículo era baseado em uma lógica simples, composta apenas por dois estados principais: **“linha detectada”** e **“linha perdida”**. Essa estrutura binária se mostrou eficiente para testes iniciais em pistas bem iluminadas e com traçado contínuo, porém apresentou limitações quando aplicada em cenários mais desafiadores, como curvas acentuadas, trechos com sombra, reflexos ou intersecções.

Com o amadurecimento do sistema, a versão final evoluiu para uma **máquina de estados expandida**, projetada para garantir maior robustez e previsibilidade no comportamento do veículo. Essa máquina de estados foi desenvolvida com o objetivo de permitir **respostas diferenciadas** a cada situação da pista, evitando oscilações bruscas na direção e melhorando a capacidade de recuperação após perdas momentâneas de detecção.

Entre os principais avanços, destacam-se:

- **Maior resposta de decisão**, com múltiplos estados intermediários que representam condições específicas, como detecção parcial de faixas, manutenção temporária de trajetória e predição de movimento.
- **Inclusão de estados de busca e predição**, permitindo que o veículo continue se movendo de forma controlada mesmo na ausência de referências visuais momentâneas.
- **Capacidade de lidar com intersecções**, diferenciando situações de travessia à esquerda ou à direita, o que não era tratado na versão anterior.
- **Transições suaves entre estados**, reduzindo oscilações no comando do servo e garantindo estabilidade direcional.

Essa abordagem marca a transição de um controle reativo simples, característico da versão 3.6, para um **modelo de controle contextual e preditivo**, onde o comportamento do sistema é ajustado conforme a condição da pista.

O código completo que implementa essa máquina de estados, incluindo as condições de transição e estratégias de predição, encontra-se disponível no **repositório GitHub** do projeto.

3.8.5.2 Nova Funcionalidade: Detecção de Faixas Transversais

Na versão **3.6**, o sistema não possuía qualquer mecanismo para identificar **intersecções** ou **faixas transversais**, tratando esses cenários como perda de linha. Isso

fazia com que o veículo perdesse o rumo temporariamente em bifurcações, curvas em “T” ou cruzamentos, comprometendo a continuidade da navegação.

Na versão final, foi adicionada uma **rotina específica de detecção de faixas transversais**, que analisa a região superior do frame binarizado (após segmentação da pista) em busca de grandes concentrações de pixels brancos distribuídos horizontalmente. Essa abordagem permite identificar com precisão o início de uma intersecção, classificando-a em três tipos principais:

- **LEFT (Esquerda)** – intersecção localizada na porção esquerda da imagem;
- **RIGHT (Direita)** – intersecção à direita do campo de visão;
- **CENTER (Central)** – cruzamento ou bifurcação simétrica.

Ao detectar uma dessas condições, a máquina de estados é automaticamente atualizada para os modos **TURN_LEFT_TRANSVERSAL** ou **TURN_RIGHT_TRANSVERSAL**, permitindo que o veículo execute a manobra correspondente de forma segura e controlada.

Essa funcionalidade representou um salto importante na **autonomia e compreensão de contexto da pista**, tornando o sistema capaz de operar em ambientes mais complexos e realistas.

O trecho de código que implementa esse recurso está disponível no repositório do projeto no **GitHub**.

3.8.5.3 Estratégia de Busca Ativa

Na versão **3.6**, quando o sistema perdia completamente a linha por um longo período, o comportamento padrão era interromper o movimento do veículo até que a detecção fosse restabelecida. Embora essa solução fosse segura, ela resultava em **paradas desnecessárias** e comprometia a fluidez do deslocamento, especialmente em curvas acentuadas ou trechos com interrupções temporárias da linha.

A versão final introduziu uma **estratégia de busca ativa**, que permite ao carro **oscilar levemente a direção** enquanto tenta reencontrar a pista. Essa técnica cria um padrão de varredura controlada, baseado em uma **função senoidal**, que desloca o servo de um lado para o outro de forma progressiva e suave.

O princípio é simples: ao invés de permanecer parado, o sistema executa um **movimento oscilatório** centrado na última direção conhecida. Isso aumenta significativamente a chance de **readquirir a linha** sem a necessidade de reiniciar o sistema, tornando o comportamento do veículo mais dinâmico e autônomo.

Essa lógica está integrada à máquina de estados no modo **SEARCH**, que é ativado automaticamente quando nenhuma borda é detectada após um determinado número de

frames. Durante essa fase, a velocidade é reduzida e a amplitude do movimento é controlada para evitar manobras bruscas.

O código responsável por essa funcionalidade está disponível no repositório do projeto no **GitHub**.

3.8.6 Sistema de Monitoramento em Tempo Real

O sistema de monitoramento em tempo real foi desenvolvido com o objetivo de fornecer visibilidade contínua do processamento interno do veículo durante sua operação. Essa funcionalidade permite acompanhar, de forma remota e instantânea, as etapas de detecção de pista, tomada de decisão e atuação dos comandos, facilitando o diagnóstico de desempenho e o ajuste fino dos parâmetros de controle.

3.8.6.1 Evolução da Visualização

Na versão **3.6**, a visualização do sistema era limitada à janela local exibida pelo **OpenCV**, utilizada principalmente para depuração e calibração de parâmetros durante os testes. Embora funcional, essa abordagem exigia conexão direta com a Raspberry Pi via monitor ou acesso remoto, dificultando o acompanhamento em tempo real durante o deslocamento do veículo.

A versão final incorporou um **servidor Flask** que permite **transmitir o vídeo processado em tempo real via rede**, acessível por qualquer dispositivo conectado ao mesmo ponto de acesso Wi-Fi. Essa implementação transformou o processo de monitoramento, tornando possível visualizar o comportamento do carro — incluindo a linha detectada, os contornos e a região de interesse — diretamente em navegadores web, sem necessidade de interface gráfica local.

Essa arquitetura de streaming é baseada no formato **MJPEG (Motion JPEG)**, que envia quadros sequenciais comprimidos no padrão JPEG, proporcionando uma visualização fluida com baixa latência.

Além de ampliar a usabilidade, o novo sistema de monitoramento serviu como uma **ferramenta de diagnóstico fundamental**, permitindo observar o processamento visual e o estado do controle em tempo real durante as competições e testes de campo.

O trecho do código responsável pela criação do servidor Flask e do stream contínuo de imagens está disponível no **repositório GitHub** do projeto.

3.8.7 Integração e Compatibilidade com a Base Existente

Durante o desenvolvimento do código final, buscou-se garantir total compatibilidade com a estrutura e os componentes utilizados nas versões anteriores do sistema. Essa abordagem permitiu que o novo código evoluísse de forma contínua, preservando as partes já validadas e integrando novas funcionalidades sem comprometer a estabilidade geral.

3.8.7.1 Manutenção de Componentes Comprovados

Durante o desenvolvimento da versão final do sistema, buscou-se preservar os **módulos já validados na versão 3.6**, garantindo compatibilidade e estabilidade entre as versões. Essa estratégia reduziu o risco de falhas e permitiu concentrar os esforços em melhorias efetivas de desempenho e robustez.

Componentes mantidos da versão 3.6:

- Configuração do **PCA9685** e comunicação via barramento **I²C**;
- Valores **PWM** calibrados para controle do **servo** e do **ESC**;
- Estrutura de **captura de vídeo** e fluxo de aquisição de frames;
- Uso consolidado das bibliotecas **OpenCV** e **NumPy** para processamento e análise de imagens.

Componentes aprimorados:

- Estratégia de detecção: substituição do modelo baseado em **Canny** e múltiplas regiões por uma abordagem **única e colorimétrica**, mais precisa e eficiente;
- Lógica de controle: evolução de um **controle proporcional simples** para uma **máquina de estados adaptativa**, com resposta reativa a diferentes condições de pista;
- Mecanismos de recuperação: adoção de **predição polinomial e busca ativa**, em substituição à memória de erro simplificada da versão anterior.

Essa filosofia de desenvolvimento incremental garantiu que cada avanço fosse sustentado sobre uma base funcional e validada, consolidando o amadurecimento técnico do sistema.

3.8.7.2 Transição Suave entre Versões

A transição entre as versões 3.6 e 3.7 foi planejada de forma a preservar a compatibilidade com o hardware e o fluxo lógico existente, resultando em uma **evolução contínua**, e não em uma substituição abrupta.

Os principais pontos que caracterizam essa transição são:

- **Reutilização de conhecimento:** os conceitos, estruturas e calibragens da versão anterior foram aproveitados e refinados na nova implementação;
- **Compatibilidade de hardware:** todos os componentes físicos, câmera, PCA9685, servo e ESC, foram mantidos, reforçando a eficiência da arquitetura modular;

- **Base para expansões futuras:** a nova organização do código e a modularização dos módulos de visão, controle e monitoramento facilitam a incorporação de novos recursos, como telemetria e tomada de decisão autônoma.

Essa abordagem garantiu que a versão 3.7 pudesse ser executada no mesmo ambiente físico da versão anterior, preservando a confiabilidade já conquistada e agregando novas funcionalidades sem comprometer a estabilidade.

3.9 Conclusão da Evolução

O sistema final apresentado na versão 3.7 representa uma **evolução significativa** em relação à arquitetura inicial da versão 3.6. Enquanto o seguidor de linha básico serviu como uma ferramenta essencial para aprendizado e validação dos conceitos fundamentais de visão computacional e controle embarcado, a versão avançada consolidou o projeto como um sistema autônomo completo, robusto e escalável.

Principais conquistas da evolução:

- Transição bem-sucedida de um **protótipo educacional** para um **sistema robusto de navegação autônoma utilizando apenas uma câmera como sensor principal**;
- Implementação de **estratégias reativas e preditivas**, capazes de lidar com falhas momentâneas de detecção;
- Capacidade de operação em **cenários complexos**, incluindo curvas, intersecções e trechos de perda parcial de linha;
- Integração de um **sistema de monitoramento em tempo real**, que aprimora o desenvolvimento, a análise e a depuração do comportamento do veículo;
- Estrutura modular e escalável, preparada para **expansões futuras**, como controle inteligente e fusão sensorial.

Essa evolução demonstra a **maturidade técnica do projeto** e a consolidação de uma arquitetura funcional, confiável e pronta para aplicações mais complexas e realistas, estabelecendo um marco sólido para as próximas etapas de desenvolvimento do carro autônomo.

4 TESTES E VALIDAÇÕES EXPERIMENTAIS

O presente capítulo apresenta os testes e validações experimentais realizados com o carro autônomo seguidor de pista, utilizando o código final desenvolvido no capítulo anterior. O objetivo desta etapa é avaliar o desempenho do sistema em condições reais, analisando a capacidade de detecção da faixa, a precisão do controle

de direção e velocidade, a robustez frente a perdas temporárias da linha e a resposta do veículo em curvas de diferentes complexidades.

Além disso, busca-se identificar limitações operacionais, como velocidade máxima segura e impacto das variações de iluminação, de forma a validar a eficácia das soluções implementadas durante o desenvolvimento. Os resultados obtidos permitem não apenas a verificação funcional do sistema, mas também fornecem informações importantes para possíveis ajustes e futuras melhorias.

4.1 Preparação para os Testes

A etapa de preparação para os testes constitui um elo essencial entre o desenvolvimento do código final e a fase de validação prática em pista. Após a implementação dos módulos de visão computacional, controle de movimento, integração com os atuadores e interface de monitoramento, foi necessário garantir que todo o sistema estivesse devidamente ajustado para suportar as condições reais de operação.

Essa preparação envolveu não apenas a verificação de funcionamento individual dos módulos, mas também a calibração conjunta de parâmetros críticos, como limites de atuação do servo motor, faixas de velocidade do ESC, estabilidade do stream de vídeo processado e resposta temporal do algoritmo de detecção de pista. Além disso, foram definidos cenários de teste controlados (reta, curvas abertas e curvas fechadas) a fim de avaliar progressivamente a robustez do sistema em diferentes condições.

4.1.1 Configuração do Ambiente de Testes

Para a validação do sistema seguidor de linha, foi necessário preparar um ambiente controlado e, posteriormente, replicar condições mais próximas ao cenário real de operação. Três tipos de pistas foram utilizados, permitindo avaliar o comportamento do carro autônomo sob diferentes texturas e condições de iluminação:

- **Pista com papel de parede preto fosco:** constituída por uma base escura e lisa, com a faixa delimitada por **fita crepe branca**. Essa configuração apresentou o melhor desempenho geral, oferecendo alto contraste visual e mínima interferência luminosa. O acabamento fosco reduziu reflexos e sombras, proporcionando um ambiente ideal para validação do algoritmo de detecção por cor.
- **Pista em superfície epóxi:** composta por uma faixa branca sobre fundo azul, confeccionada manualmente. Esse ambiente foi pensado para reduzir interferências visuais externas e permitir uma análise inicial mais controlada do algoritmo.
- **Pista em piso asfáltico:** construída por meio da aplicação de fitas adesivas brancas sobre o chão, delimitando o trajeto. Essa configuração buscou reproduzir condições mais próximas às de um ambiente real, embora tenha introduzido desafios adicionais devido à textura e variação de cor do piso.

As dimensões das pistas (9m de comprimento, 13m de largura, 4 curvas), bem como o raio das curvas, foram definidas de modo a contemplar trechos retos, curvas abertas e curvas fechadas, possibilitando avaliar o desempenho do veículo em diferentes cenários de complexidade.

As condições de iluminação também foram objeto de análise:

- Na pista com papel de parede preto e na pista de epóxi, utilizou-se iluminação controlada com luzes de LED direcionadas ao solo, minimizando sombras e reflexos.
- Já na pista de piso asfáltico, os testes foram realizados sob iluminação natural (luz solar), expondo o sistema a variações de intensidade luminosa e à presença de sombras.

Outro aspecto relevante foi o ajuste prévio da câmera. Diversas posições e ângulos foram testados até se definir uma fixação estável que maximizasse a captura da pista sem distorções excessivas, garantindo uma boa proporção entre a área visível do trajeto e a frente do veículo.

4.2 Metodologia de Testes

Os testes realizados tiveram como objetivo verificar o funcionamento geral do carro autônomo seguidor de pista em condições reais, avaliando sua capacidade de manter-se dentro da faixa delimitada, reagir a curvas e lidar com variações de iluminação e textura do piso.

As observações foram feitas de forma prática, sem o uso de medições automatizadas, priorizando a análise visual do comportamento do veículo e o registro dos principais desafios encontrados durante a execução.

Critérios de Avaliação:

- **Manutenção na faixa:** verificação visual de quanto tempo o carro permaneceu dentro da pista durante o percurso.
- **Desempenho em curvas:** observação da resposta do sistema em curvas abertas e fechadas, avaliando a estabilidade da direção e eventuais perdas de trajetória.
- **Influência do tipo de piso:** testes realizados em diferentes superfícies (principalmente piso escuro e liso), para identificar impacto no desempenho da detecção de linha.
- **Efeitos da iluminação:** análise da sensibilidade do sistema à variação de luz e sombra, observando possíveis falhas na identificação da faixa.

- **Recuperação de trajetória:** observação do comportamento do carro ao perder momentaneamente a linha, avaliando sua capacidade de retornar à pista sem intervenção manual.

4.3 Resultados Obtidos

Os testes realizados com o carro autônomo permitiram avaliar o comportamento do sistema em condições reais de operação, identificando tanto seus pontos fortes quanto suas limitações práticas. Os resultados foram obtidos a partir da execução contínua do código final e da análise dos dados de erro, velocidade e estados operacionais registrados durante os ensaios.

4.3.1 Precisão e Estabilidade

O sistema demonstrou boa estabilidade em trechos retos e curvas suaves, mantendo o veículo centralizado dentro da faixa durante a maior parte do trajeto.

A filtragem dos erros e o uso da média móvel auxiliaram na suavização da resposta do servo, reduzindo oscilações bruscas no controle de direção.

4.3.2 Tempo de Resposta em Perdas Temporárias

Em situações de perda parcial da faixa, o sistema conseguiu restabelecer a trajetória utilizando os mecanismos de **predição polinomial** e **busca ativa**, recuperando o centro da pista. Esse tempo de resposta mostrou-se adequado em condições estáveis de iluminação, mas aumentou em ambientes de alto contraste ou superfícies irregulares, onde o ruído visual compromete a detecção dos contornos.

4.3.3 Desempenho em Diferentes Superfícies

Os testes comparativos entre diferentes tipos de pista evidenciaram que o desempenho do sistema é fortemente influenciado pelo tipo de superfície, textura e condições de iluminação do ambiente.

Entre os ambientes testados, destacou-se a pista construída com **papel de parede preto fosco**, onde a linha foi delimitada por **fita crepe branca**. Essa configuração proporcionou **o melhor desempenho geral**, com detecção de linha estável e resposta precisa do controle de direção. O acabamento fosco reduziu reflexos, permitindo que o sistema mantivesse o veículo centralizado na faixa com mínima oscilação.

Em contrapartida, os testes realizados em outras superfícies apresentaram variações de desempenho significativas:

- **Pista de epóxi (escura e lisa):** o desempenho foi satisfatório, com detecção consistente da faixa branca e raras falhas de segmentação, embora reflexos ocasionais pudessem interferir no contorno da faixa.

- **Piso asfáltico (texturizado e irregular):** o sistema demonstrou instabilidade na detecção, apresentando dificuldade em distinguir a faixa das variações naturais do piso. Reflexos e sombras causaram falsas detecções, resultando em desvios e perda temporária da trajetória.

Esses resultados confirmam que o sistema atual é **mais confiável em superfícies escuras, lisas e homogêneas**, nas quais o **contraste entre a linha branca e o fundo** é bem definido. Em superfícies com textura irregular ou coloração variável, o desempenho tende a degradar devido à limitação do método de segmentação baseado em cor.

4.3.4 Comportamento em Curvas e Intersecções

Em **curvas fechadas**, o sistema apresentou **limitações significativas**. Nessas situações, parte da faixa frequentemente saía do campo de visão da câmera, impossibilitando a detecção simultânea das duas bordas. Embora a máquina de estados tenha contribuído para manter o controle em algumas situações, o carro frequentemente apresentava sobrecorreção ou saía temporariamente da pista antes de se recuperar. Já nas **intersecções**, a função de detecção de faixas transversais obteve sucesso ao identificar bifurcações e reagir de forma adequada, embora em alguns casos o tempo de decisão tenha sido alto.

4.3.5 Observações Gerais

De modo geral, o carro autônomo apresentou desempenho satisfatório dentro das condições controladas, validando o funcionamento integrado da visão computacional, controle de movimento e interface de monitoramento.

No entanto, limitações importantes foram observadas:

- Sensibilidade a variações de cor e textura do piso;
- Dificuldade em curvas fechadas, especialmente quando a linha desaparece parcialmente do campo de visão;
- Dependência de iluminação uniforme para manter a detecção estável;
- Risco de saturação do servo em respostas rápidas a erros grandes.

Essas restrições indicam que, embora o sistema tenha atingido o objetivo de navegação autônoma básica, há amplo espaço para aprimoramentos em robustez, velocidade e capacidade de generalização.

4.4 Melhorias Futuras

Com os resultados obtidos e as limitações observadas nos testes práticos, várias melhorias podem ser projetadas para futuras versões do carro autônomo. Essas propostas visam não apenas ampliar a robustez do sistema, mas também aproximar o projeto de soluções mais avançadas em visão computacional e condução autônoma.

4.4.1 Aprimoramento da visão computacional:

O sistema atual baseia-se exclusivamente na detecção de cores em um único plano de visão. Futuras versões podem incorporar **múltiplas câmeras**, possibilitando **visão estéreo** e percepção de profundidade, o que ampliaria significativamente a capacidade do veículo de entender o ambiente ao seu redor. Além disso, o uso de **modelos de aprendizado profundo (Deep Learning)**, como **redes neurais convolucionais (CNNs)**, poderia permitir a detecção de faixas, curvas e intersecções com maior precisão, mesmo sob variações de iluminação ou textura do solo.

4.4.2 Adaptação automática às condições do ambiente:

Atualmente, a calibração da faixa HSV é fixa, o que limita o desempenho em ambientes com diferentes níveis de luminosidade. Uma evolução importante seria a implementação de **algoritmos de ajuste dinâmico de brilho e contraste**, ou mesmo **modelos de aprendizado adaptativo**, capazes de recalibrar automaticamente os parâmetros de detecção conforme o cenário muda, por exemplo, alternando entre ambientes internos e externos.

4.4.3 Integração de múltiplos sensores:

A fusão de dados de **sensores ultrassônicos, infravermelhos ou LIDAR** com a visão computacional aumentaria a segurança e confiabilidade do sistema. Essa abordagem multimodal permitiria ao veículo detectar obstáculos, calcular distâncias e até reagir a objetos fora do campo de visão da câmera, introduzindo noções básicas de **percepção 3D** e **planejamento de trajetória**.

4.4.4 Controle inteligente e redes neurais de direção:

O controle atual é baseado em lógica proporcional e estados discretos. Futuras versões podem empregar **redes neurais de controle**, treinadas com dados coletados durante as sessões de teste, para aprender padrões de direção ideais. Essa estratégia, inspirada em frameworks como o *DonkeyCar* e o *Duckietown*, permitiria que o carro aprendesse com sua própria experiência, otimizando automaticamente curvas, aceleração e frenagem.

4.4.5 Sistema de telemetria e análise de dados:

A implementação de um **módulo de telemetria em tempo real** permitiria acompanhar variáveis do sistema (como erros, velocidade e sinais PWM) diretamente via rede local

ou nuvem. Além disso, um **sistema de logging** estruturado possibilitaria armazenar e analisar o comportamento do carro durante os testes, servindo de base para **treinamento de modelos de IA** e ajustes automáticos de parâmetros.

4.4.6 Exploração de plataformas de hardware mais avançadas:

Embora a Raspberry Pi 4B tenha se mostrado adequada ao propósito experimental, futuras versões podem adotar **placas com GPU integrada**, como a **NVIDIA Jetson Nano ou Xavier**, que oferecem desempenho superior para aplicações de visão e aprendizado profundo, mantendo compatibilidade com o código-base existente.

5 CONCLUSÕES FINAIS

O desenvolvimento do carro autônomo em escala 1:10 permitiu compreender, na prática, conceitos de visão computacional, controle de atuadores e integração entre hardware e software. O sistema proposto mostrou-se funcional, sendo capaz de seguir uma faixa branca e deslocar-se entre faixas sobre diferentes superfícies, ainda que com limitações em curvas acentuadas e sob variações de iluminação.

Os testes realizados demonstraram que o uso de processamento de imagem baseado em cores é eficiente em ambientes controlados, porém sensível a reflexos, sombras e mudanças abruptas de luminosidade. A estrutura modular do código, aliada ao uso do controlador PCA9685 para o acionamento preciso do servo de direção e do motor, mostrou-se adequada e robusta, facilitando futuras expansões e ajustes do sistema.

Como validação prática do projeto, o veículo desenvolvido foi utilizado na competição **RoboCar Race 2025**, na qual obteve **primeiro lugar**, evidenciando a eficiência da arquitetura proposta e a confiabilidade do sistema em um ambiente competitivo e realista. Esse resultado reforça a aplicabilidade do projeto e a relevância das soluções adotadas ao longo do desenvolvimento.

Todos os registros fotográficos do projeto finalizado e da participação na competição, encontram-se disponíveis no repositório **GitHub** do projeto. Adicionalmente, a competição RoboCar está devidamente referenciada na seção de referências deste trabalho.

De forma geral, o projeto atingiu seus objetivos principais e estabeleceu uma base sólida para aprimoramentos futuros, como a implementação de redes neurais, a integração de sensores adicionais e a adoção de técnicas mais avançadas de visão computacional, os quais serão abordados no capítulo seguinte.

5.1 Propostas futuras

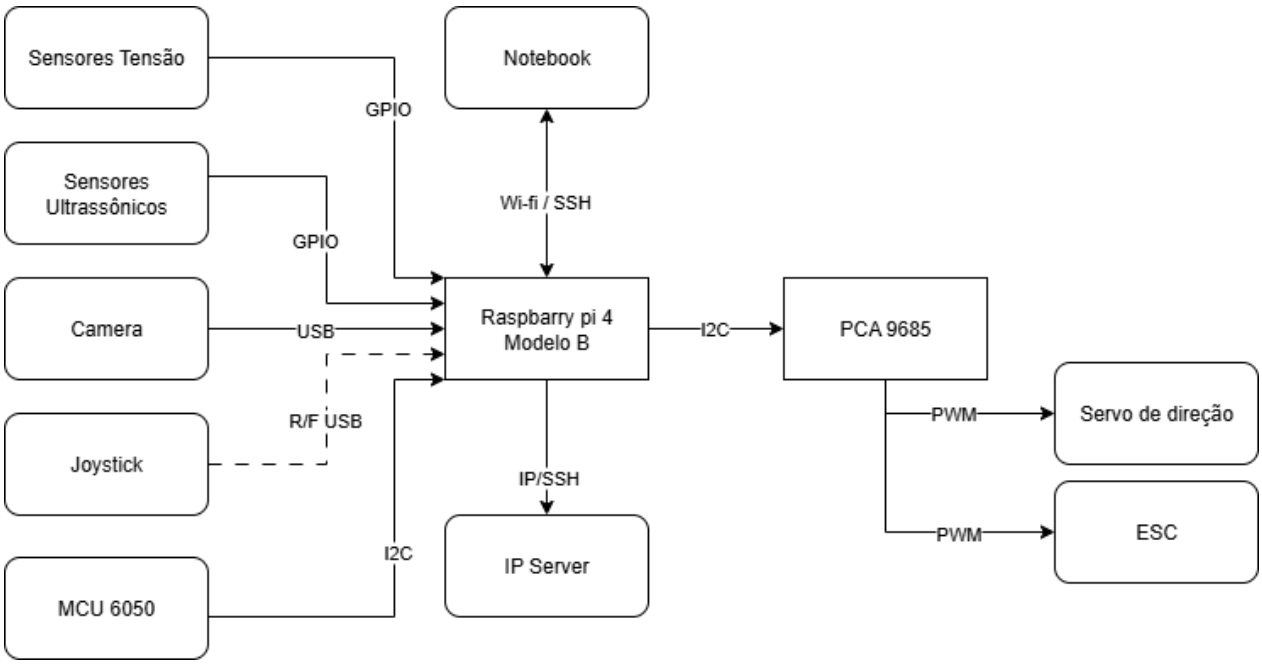
Com base nas conclusões e limitações identificadas, diversas melhorias podem ser implementadas para elevar o desempenho e a autonomia do veículo.

Uma das principais propostas é a integração de **redes neurais convolucionais (CNNs)** para detecção e segmentação de pista, substituindo a detecção baseada em cor por um modelo de aprendizado profundo capaz de se adaptar a diferentes condições de iluminação e superfícies. O uso de frameworks como **TensorFlow Lite** ou **PyTorch Mobile** permitiria a execução otimizada dessas redes diretamente na Raspberry Pi, viabilizando inferências em tempo real.

Além disso, sugere-se o uso de **múltiplas câmeras** — por exemplo, uma frontal e outra lateral — para ampliar o campo de visão e fornecer dados estereoscópicos, permitindo estimar distâncias e melhorar o controle em curvas e intersecções. Essa abordagem também abriria espaço para o uso de **visão 3D** e reconstrução de ambiente.

Outra evolução promissora é a incorporação de **sensores complementares**, como ultrassônicos, giroscópio, acelerômetro e módulos LIDAR. Esses sensores poderiam atuar em conjunto com a câmera, fornecendo redundância e maior confiabilidade em ambientes complexos, além de permitir a fusão sensorial (sensor fusion) para decisões mais seguras e precisas. A figura 26, ilustra um exemplo de uma futura arquitetura.

Figura 26 - Arquitetura futura proposta



No campo do controle, propõe-se o desenvolvimento de **algoritmos de predição baseados em aprendizado de máquina**, capazes de antecipar o comportamento do veículo em curvas e adaptar dinamicamente os parâmetros de direção e velocidade. Aliado a isso, um **sistema de registro de dados (data logging)** poderá ser adicionado para armazenar imagens, valores de sensores e comandos aplicados, criando uma base para futuros treinamentos e análises de desempenho.

Por fim, almeja-se a evolução do projeto para um **nível superior de automação**, incorporando conceitos de navegação autônoma completa, reconhecimento de obstáculos, e detecção de placas ou sinais visuais — transformando o veículo em uma plataforma de testes para pesquisa e desenvolvimento em mobilidade inteligente.

REFERENCIAS

A2 ROBOTICS. **Micro Servo Motor 9g SG90 TowerPro**. Disponível em: https://www.a2robotics.com.br/lhylxc812-micro-servo-motor-9g-sg90-towerpro. Acesso em: 21 maio 2025.

APPLICATIONS of computer vision in autonomous vehicles. **arXiv**, 2023. Disponível em: https://arxiv.org/html/2311.09093. Acesso em: 3 maio 2025.

APTIV. **What are operational design domains?** 2023. Disponível em: https://www.aptiv.com/en/insights/article/what-are-operational-design-domains. Acesso em: 29 abr. 2025.

ARDUCORE. **Controlador PWM Servo Driver 16 Canais I2C Arduino PCA9685**. Disponível em: https://www.arducore.com.br/controlador-pwm-servo-driver-16-canais-i2c-arduino-pca9685. Acesso em: 21 maio 2025. CNBC. **Autonomous vehicles are here, but the public is reluctant to trust them**. 2024. Vídeo. Disponível em: https://www.cnbc.com/video/2024/10/07/autonomous-vehicles-are-here-but-the-public-is-reluctant-to-trust-them.html. Acesso em: 21 maio 2025.

AXIS TECHNOLOGY. *AX-1080P-V2 USB Camera Module – Technical Specifications*. Disponível em: https://www.alibaba.com/product-detail/AX-1080P-V2-USB-Camera-Module_1600211627720.html. Acesso em: 12 maio 2025.

COUTO, Leandro Nogueira. **Sistema para localização robótica de veículos autônomos baseado em visão computacional por pontos de referência**. 2012. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. Disponível em: https://teses.usp.br/teses/disponiveis/55/55134/tde-04092012-110014/pt-br.php. Acesso em: 3 maio 2025.

DESIGN and implementation of PID-based steering control for 1/10-scale autonomous vehicle. **ResearchGate**, 2021. Disponível em: https://www.researchgate.net/publication/357722821_Design_and_Implementation_of_PID-Based_Steering_Control_for_110-Scale_Autonomous_Vehicle. Acesso em: 3 maio 2025.

DIAS, Mauricio Acconcia. **Sistema de hardware reconfigurável para navegação visual de veículos autônomos**. 2016. Tese (Doutorado em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. Disponível em: https://www.teses.usp.br/teses/disponiveis/55/55134/tde-13012017-164142/pt-br.php. Acesso em: 3 maio 2025.

DONKEY CAR. **Create an autopilot**. [S. l.]: Donkey Car, [202–]. Disponível em: <https://docs.Donkey Car.com/guide/train_autopilot/>. Acesso em: 3 maio 2025.]

ENCAUT, Rickson. **Detecção de objetos em tempo real pela câmera do celular com a YOLOV3**. 2024. Medium. Disponível em: https://ricksonencaut.medium.com/detec%C3%A7%C3%A3o-de-objetos-em-tempo-real-pela-c%C3%A2mera-do-celular-com-a-yolov3-178f84339578. Acesso em: 21 maio 2025.

GRIGORIOU, A.; VLASTARIS, A.; NIKOLAIDIS, N. **Computer vision for autonomous vehicles: a review**. **Sensors**, v. 21, n. 14, 2021. Disponível em: https://www.mdpi.com/1424-8220/21/14/4758. Acesso em: 3 maio 2025.

IMAGE analysis in autonomous vehicles: a review of the latest AI. **MDPI**, 2024. Disponível em: https://www.mdpi.com/2076-3417/14/18/8150. Acesso em: 3 maio 2025.

INFORMA. **Otonomo sources road data from BMW and Mini vehicles for mobility services**. Disponível em: https://wardsintelligence.informa.com/wi964745/otonomo-sources-road-data-from-bmw-and-mini-vehicles-for-mobility-services. Acesso em: 21 maio 2025.

LEXAR (Longsys Electronics Co., Ltd.). *Lexar PLAY microSDXC UHS-I Card – Especificações A2, U3*. Disponível em: https://www.amazon.com/Lexar-Compatible-Nintendo-Switch-Smartphones-LMSPLAY001T-BNNNU/dp/B08T8LL7G8. Acesso em: 5 setembro 2025.

MEDEIROS, W. T. **Aula 14 – Veículos autônomos**. 2024. Material de aula. Acesso em: 27 abr. 2025. MOBIAUTO. **Truque em VW Nivus fixa ACC sem precisar colar logo para evitar furto**. Disponível em: https://www.mobiauto.com.br/revista/truque-em-vw-nivus-fixa-acc-sem-precisar-colar-logo-para-evitar-furto/3986. Acesso em: 21 maio 2025.

MOLINA, Caroline Bianca Santos Tancredi. **Sistemas ciberfísicos veiculares: um estudo de arquitetura para veículos autônomos**. 2018. Dissertação (Mestrado em Engenharia Elétrica) – Universidade de São Paulo. Disponível em: https://www.teses.usp.br/teses/disponiveis/3/3141/tde-01112018-153824/publico/CarolineBiancaSantosTancrediMolinaCorr18.pdf. Acesso em: 27 abr. 2025.

MOUTAOUAKKIL, A.; EL KHATIB, K. **Operational design domain for automated driving systems: taxonomy, definition, and application**. **arXiv**, 2023. Disponível em: https://arxiv.org/abs/2304.11827. Acesso em: 29 abr. 2025.

MUSÉE DES ARTS ET MÉTIERS. **Le fardier à vapeur de Cugnot**. [S. l.]: Musée des Arts et Métiers, [s. d.]. Disponível em: https://www.arts-et-metiers.net/musee/fardier-vapeur. Acesso em: 6 jan. 2025.

NEW control model for autonomous vehicles using integration of MPC and Stanley controllers. **Nature**, 2024. Disponível em: https://www.nature.com/articles/s41598-024-69858-7. Acesso em: 3 maio 2025.

NOUSIAS, S. et al. **Accelerating deep neural networks for efficient scene understanding in automotive cyber-physical systems**. **arXiv preprint arXiv:2107.09101**, 2021. Disponível em: https://arxiv.org/abs/2107.09101. Acesso em: 1 abr. 2025.

NXP SEMICONDUCTORS. *PCA9685 – 16-channel, 12-bit PWM Fm+ I²C-bus LED controller – Product data sheet, Rev. 4 (16 April 2015)*. Disponível em: https://cdn-shop.adafruit.com/datasheets/PCA9685.pdf. Acesso em: 12 maio 2025.

PAULA, Maurício Braga de. **Visão computacional para veículos inteligentes usando câmeras embarcadas**. 2015. Tese (Doutorado em Engenharia Elétrica) – Universidade Federal do Rio Grande do Sul, Porto Alegre. Disponível em: https://www.lume.ufrgs.br/handle/10183/122511. Acesso em: 3 maio 2025.

RASPBERRY PI (TRADING) LTD. *Raspberry Pi 4 Model B Datasheet (versão 1.1, março 2024)*. Disponível em: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf. Acesso em: 12 maio 2025.

RESEARCHGATE. **An overview on LiDAR for autonomous vehicles**. 2024. Disponível em: https://www.researchgate.net/publication/379900256_An_Overview_on_LiDAR_for_Autonomous_Vehicles. Acesso em: 3 maio 2025.

RODRIGUES, Suzana. **O pipeline de visão computacional com Python, OpenCV**. 2024. Medium. Disponível em: https://suzana-svm.medium.com/o-pipeline-de-visao-computacional-com-python-opencv-adc70112f5ee. Acesso em: 21 maio 2025.

SAE INTERNATIONAL. **SAE J3016™: updated visual chart for driving automation systems**. 2021. Disponível em: https://www.sae.org/blog/sae-j3016-update. Acesso em: 28 abr. 2025.

SAE INTERNATIONAL. **Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles (J3016_202104)**. 2021. Disponível em: https://www.sae.org/standards/content/j3016_202104/. Acesso em: 29 abr. 2025.

SANTANA, Joelton. *TCC Carro Autônomo Raspberry Pi*. Disponível em: https://github.com/joeltonjc1/tcc_carro_autonomo_raspberry_pi. Acesso em: 05 out. 2025.

SANTOS, Giovanni Almeida. **Arquitetura funcional de veículos autônomos: uma proposição de técnicas para detecção de objetos, localização e interação humano-**

veículo. 2022. Tese (Doutorado em Engenharia Elétrica) – Universidade de Brasília, Brasília. Disponível em: https://repositorio.unb.br/jspui/handle/10482/44787. Acesso em: 3 maio 2025.

SILVA, Rômulo Ferreira de Araújo. **Sistema de percepção e controle para veículo autônomo utilizando visão computacional**. 2022. Trabalho de Conclusão de Curso – Universidade Federal do Ceará. Disponível em: https://repositorio.ufc.br/bitstream/riufc/65338/1/2022_tcc_rfdeasilva.pdf. Acesso em: 27 abr. 2025.

SKYRC. **Cheetah**. Disponível em: https://www.skyrc.eu/images/SkyRC/Cheetah/Cheetah2.jpg. Acesso em: 21 maio 2025.
TAMIYA USA. **Jr Roborace Devbot 2.0 (MA Chassis)**. [S. l.]: Tamiya USA, [s. d.]. Disponível em: https://www.tamiyausa.com/shop/132-pro/jr-roborace-devbot-20/. Acesso em: 3 maio 2025.

SHENZHEN PKCELL BATTERY CO., LTD. *ICR18650 2200 mAh 3,7 V — Technical Specification (QA.S.0221, edição A/0, 28 julho 2014)*. Disponível em: https://cdn-shop.adafruit.com/product-files/1781/C2253_-_ICR18650_2200mAh_3.7V_with_PCM_20140728_APPROVED_8.18.pdf. Acesso em: 12 maio 2025.

TESTOURI, M.; ELGHAZALY, G.; FRANK, R. **RoboCar: a rapidly deployable open-source platform for autonomous driving research**. **arXiv preprint arXiv:2405.03572**, 2024. Disponível em: https://arxiv.org/abs/2405.03572. Acesso em: 3 maio 2025.

UNIVERSIDADE DE SÃO PAULO. **Projeto CARINA 2 – Laboratório de Robótica Móvel (LRM)**. São Carlos: USP, [s. d.]. Disponível em: http://lrm.icmc.usp.br/web/index.php?n=Port.ProjCarina2Info. Acesso em: 6 jan. 2025.

WANG, Jimmy. **Teaching cars to see: vehicle detection using machine learning and computer vision**. 2023. Medium. Disponível em: https://medium.com/data-science/teaching-cars-to-see-vehicle-detection-using-machine-learning-and-computer-vision-54628888079a. Acesso em: 21 maio 2025.

YAHBOOM. **Camera-hfr**. Disponível em: https://category.yahboom.net/products/camera-hfr. Acesso em: 21 maio 2025.