PROJETO DE GRADUAÇÃO

CONTROLE, TELEMETRIA E INSTRUMENTAÇÃO DE UM AUTOMODELO

Lucas Koeler Somavilla Bomfim

Brasília, 11 de Maio de 2021

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

PROJETO DE GRADUAÇÃO

CONTROLE, TELEMETRIA E INSTRUMENTAÇÃO DE UM AUTOMODELO

POR,

Lucas Koeler Somavilla Bomfim

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro Mecânico.

Banca Examinadora

Prof. Jones Yudi Mori, UnB/ ENM (Orientador) _____

Prof. Walter de Britto, UnB/ ENM _____

Prof. Adriano Todorovic Fabro, UnB/ ENM _____

Brasília, 11 de Maio de 2021

Dedicatória

Este trabalho é dedicado ao jovem Lucas que entrou na UnB com grandes dúvidas e diversas expectativas que começam a se concretizar neste trabalho.

Lucas Koeler

Agradecimentos

Um Projeto de graduação entregue tem um significado grandioso na vida pessoal de um universitário. Representa o fim de um ciclo no qual ocorreram grandes mudanças em seu jeito de pensar, remete às decisões tomadas e reflete em novas portas abertas que deverão ser escolhidas.

Muito obrigado a todos os colegas de curso, amizades e família, que fizeram parte desse amadurecimento e não me deixaram perder o rumo durante as curvas mais sinuosas deste caminho!

Um agradecimento especial à minha mãe, Analucia, que me ajudou por horas com os testes e correções, ao meu pai, Airton, pelo suporte durante esta época e à minha irmã, Marina, que me ajudou com diversas correções e dicas.Um grande agradecimento à um grande amigo também, Bruno, que sempre gostou de discutir os assuntos técnicos que desencadearam em vários insights no projeto, assim como ajudas imprescindíveis nos programas de MATLAB e à Amanda, por suas correções e todo seu carinho, paciência e apoio emocional.

Lucas Koeler

RESUMO

Este trabalho tem o intuito construir, instalar e testar uma plataforma de telemetria, instrumentação e controle em um automodelo off-road de escala 1:10 com o intuito de estabelecer uma base de estudos futuros de autoguiagem e assistividade veicular pelos alunos da Universidade de Brasília (UnB). O veículo utilizado é originalmente acionado por controle remoto de radiofrequência com um chassi inspirado em modelos realísticos. Assim, o estudo é iniciado por uma fundamentação mostrando teoria e cálculos de modelos veiculares cinemáticos e dinâmicos simplificados para calcular a trajetória do veículo, assim como uma introdução ao funcionamento de componentes eletromecânicos e mecatrônicos utilizados no projeto como servomotores, sensores IMU, infravermelhos, ultrassônicos, magnetômetros, motores DC e microcontroladores adicionando uma breve explicação de sistemas de controle PID, métodos de prototipagem rápida e sistemas embarcados. Após isso é descrito o processo de implementação da plataforma com conexões eletrônicas, modelagem 3D, impressão 3D e fixação de todos os componentes do projeto, assim como a criação de um odômetro/velocímetro utilizando um sensor infravermelho, além de uma seção onde é feita a medição das características mecânicas principais do automodelo para a utilização dos modelos cinemáticos. Depois de obter a plataforma montada, é descrito uma série de testes realizados para validar o projeto, como ensaios de funcionamento dos sensores magnetômetro, e giroscópio do IMU, odômetro, assim como testes para realizar a calibração de sistemas PID para controlar o direcional e a velocidade do veículo. Ademais é feito um teste de performance comparando a obtenção de posição angular por um magnetômetro e por um acelerômetro. E por fim, é relatado também o teste de validação final do projeto, onde foi feito o cálculo de uma trajetória simples simulada na plataforma *MATLAB* levando em consideração o modelo simplificado cinemático da bicicleta e as características mecânicas do veículo de forma que seus ângulos de giro a cada instante foram exportados para o controlador seguir e assim realizar uma comparação entre a trajetória real e a simulada.

Palavras-chaves: Cálculo de trajetória veicular, modelo cinemático da bicicleta, implementação de controle PID, instrumentação de automodelo, veículo de controle remoto.

ABSTRACT

This study aims to build, install and test a telemetry, instrumentation, and control platform in a 1:10 scale off-road auto model in order to establish a basis for future studies of self-guided and vehicle assistance by the students of the Universidade de Brasília (UnB). The vehicle used is originally driven by a radio frequency remote controller and has a chassis inspired by realistic models. Thus, the study is initiated by showing the theory and calculations behind simplified kinematic and dynamic vehicle models to calculate its trajectory, as well as an introduction to the functioning of the electromechanical and mechatronic components used in the project such as servo motors, IMU modules, DC motors, microcontrollers and infrared, ultrasonic and magnetometer sensors, adding a brief explanation of PID control systems, rapid prototyping methods, and embedded systems. After that, the process of implementing the platform with electronic connections, 3D modeling, 3D printing, and the fixation of all project components are described, as well as the creation of an odometer/speedometer using an infrared sensor, in addition to a section where the measurement of the main mechanical characteristics of the vehicle used by kinematic models. After obtaining the assembled platform, a series of tests carried out to validate the project are described, such as the functioning of the magnetometer, IMU, gyroscope, and odometer, as well as tests to perform the calibration of PID systems to control the directional system and the vehicle speed. In addition, a performance test is made, comparing the achievement of the angular position by the magnetometer and the accelerometer. At last, the final project validation test is made, reporting the calculation of a simulated simple trajectory on the MATLAB platform, taking into account the simplified kinematic bicycle model and the mechanical characteristics of the vehicle, so that the program return angles at each instant, enabling to export this data to the controller via matrices to follow the path and, thus, perform a comparison between real and simulated trajectory.

Key words: Vehicle trajectory calculation, kinematic bicycle model, PID control implementation, instrumentation of a scaled-down model car, remote control vehicle.

SUMÁRIO

LISTA DE FIGURAS

Capítulo 4

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

a	Aceleração	[m/s^2]
A	Área	[m^2]
b	Distância entre os centros de cada roda lateral, bitola do veículo	[m]
$C_{\alpha f}$	Coeficiente de dureza lateral do pneu frontal do modelo de duas rodas	[N/rad]
C	Centro de massa do veículo estudado	[m]
$C_{\alpha r}$	Coeficiente de dureza lateral do pneu traseiro do modelo de duas rodas	[N/rad]
$C_{irc\ ef}$	Circunferência efetiva da roda	[m ou mm]
C_{mx}	Centro de massa referente ao eixo Y_v	[mm]
C_{my}	Centro de massa referente ao eixo X_v	[mm]
d	Distância no eixo cartesiano d do referencial móvel ou local da trajetória	[m]
D_C	Largura de pulso, *Duty Cycle* ou ciclo útil de trabalho	[µs ou %]
D_{lon}	Distância longitudinal percorrida pelo veículo	[m]
D_{lonPR}	Distância longitudinal percorrida com um cada giro completo do eixo traseiro	[m]
e	Distância no eixo cartesiano e do referencial móvel ou local da trajetória	[m]
F_x	Força resultante na direção X_V aplicada no centro de massa C do veículo	[N]
h	Altura do veículo	[m]
I_Z	Momento de Inércia do corpo com relação ao centro de massa C	[kg.m^2]
l	Distância do entre eixos do veículo	[m]
l_f	Distância entre o centro de massa C do veículo e seu eixo de rodas dianteiro	[m]
l_r	Distância entre o centro de massa C do veículo e seu eixo de rodas dianteiro	[m]
lim	Limitador do grau de curvatura utilizado nos programas de MATLAB	[-]
m	Massa do veículo	[kg]
$M_{vx,C}$	Momento de rotação do eixo X_v, aplicado no centro de massa C do veículo	[N.m]
$M_{vy,C}$	Momento de rotação do eixo Y_v, aplicado no centro de massa C do veículo	[N.m]
$M_{vz,C}$	Momento de rotação do eixo Z_v, aplicado no centro de massa C do veículo	[N.m]
n	Distância no eixo cartesiano n do referencial móvel ou local da trajetória	[m]
O	Centro de rotação de uma curva	[m]
P	Ponto de interesse no espaço	[m]
R	Raio da curva correspondente ao centro O e o centro de massa C do veículo	[m]
R_{ef}	Raio efetivo da roda	[m ou mm]
$\overrightarrow{r_P}(t)$	Vetor distância entre a origem s e um ponto P no espaço	[m]
s	Ponto de origem arbitrado onde o veículo começa a sua trajetória	[m]
t	Tempo	[s]
T_{total}	Período total de um ciclo inteiro da frequência PWM	[µs]
$T_{\text{sinal ativo}}$	Período do sinal ativo durante o ciclo (com presença de DDP)	[µs]
$T_{\text{sinal inativo}}$	Período do sinal inativo durante o ciclo (DDP nula)	[µs]
T_X	Densidade de fluxo magnético com relação ao eixo X_v	[µT]
$T_{X\_Corrigido}$	Densidade de fluxo magnético com relação ao eixo X_v com correções de inclinação	[µT]
T_Y	Densidade de fluxo magnético com relação ao eixo Y_v	[µT]
$T_{Y\_Corrigido}$	Densidade de fluxo magnético com relação ao eixo Y_v com correções de inclinação	[µT]
T_Z	Densidade de fluxo magnético com relação ao eixo Z_v	[µT]
V	Velocidade	[m/s]
V_{lat}	Velocidade local na direção Yv no centro de massa C do veículo	[m/s^2]
V_{lon}	Velocidade local na direção Xv no centro de massa C do veículo	[m/s^2]
$\dot{V}_{lat}$	Aceleração local na direção Yv no centro de massa C do veículo	[m/s^2]
$\dot{V}_{lon}$	Aceleração local na direção Xv no centro de massa C do veículo	[m/s^2]
X	Distância no eixo cartesiano X do referencial fixo ou global no solo	[m]
X_v	Distância no eixo cartesiano X do referencial móvel ou local no veículo	[m]
$\dot{X}$	Velocidade no eixo cartesiano X do referencial fixo ou global no solo	[m]

Y	Distância no eixo cartesiano Y do referencial fixo ou global no solo	[m]
$\dot{Y}$	Velocidade no eixo cartesiano Y do referencial fixo ou global no solo	[m]
Y_v	Distância no eixo cartesiano Y do referencial móvel ou local no veículo	[m]
Z	Distância no eixo cartesiano Z do referencial fixo ou global no solo	[m]
Z_v	Distância no eixo cartesiano Z do referencial móvel ou local no veículo	[m]
$\dot{Z}$	Velocidade no eixo cartesiano Z do referencial fixo ou global no solo	[m]

Símbolos Gregos

α	Ângulo de deriva ou *slip angle*	[°]
α_f	Ângulo de deriva dos pneus frontais aglutinados no modelo de duas rodas	[°]
α_r	Ângulo de deriva dos pneus traseiros aglutinados no modelo de duas rodas	[°]
β	Ângulo formado entre a direção da velocidade do veículo e o seu eixo XV	[°]
δ	Média das cotangentes entre os ângulos direcionais da roda interna e externa	[°]
δ_i	Ângulo direcional da roda interna	[°]
δ_{max}	Ângulo direcional médio máximo	[°]
δ_o	Ângulo direcional da roda externa	[°]
θ	Arfagem, ângulo entres os eixos *Z* e Z_v coincidente ao ângulo de rotação do eixo Y_v	[°]
φ	Rolamento, ângulo entres os eixos *Y* e Y_v coincidente ao ângulo de rotação do eixo X_v	[°]
Ψ	Guinada, ângulo entres os eixos *X* e X_v coincidente ao ângulo de rotação do eixo Z_v	[°]
$\dot{\Psi}$	Velocidade angular, também dada por ω	[rad/s]
Ψ_N	Ângulo do norte magnético com relação ao eixo Y_v	[°]
ω	Velocidade angular de rotação do eixo Z_v	[rad/s]
ω_δ	Velocidade angular de resposta do motor direcional	[°/s]
$\dot{\omega}$	Aceleração angular de rotação do eixo Z_v	[rad/s^2]

Grupos Adimensionais

N	Número de rotações
W_{e-r}	Relação de giros entre o eixo traseiro e a roda

Subscritos

amb	ambiente
ext	externo
in	entrada
ex	saída
v	veículo
P	Referente a um ponto P no espaço
lon	Referente a direção longitudinal do veículo, colinear ao eixo X_v
lat	Referente a direção lateral do veículo, colinear ao eixo Y_v
vert	Referente a direção vertical do veículo, colinear ao eixo Z_v
ef	Efetivo

Sobrescritos

•	Derivada de primeira ordem com relação ao tempo
••	Derivada de segunda ordem com relação ao tempo

Siglas

3D	Sistema de Três dimensões
ABNT	Associação Brasileira de Normas Técnicas
CAD	*Computer Aided Design* - Design Auxiliado por Computador
CH	*Chanel* – Canal
CNC	Controle Numérico Computadorizado
DARPA	Defense Advanced Research Projects Agency
DC	Direct Current – Corrente Contínua
DDP	Diferença de potencial elétrico
EDO	Equação diferencial ordinária
FDM	*Fused Deposition Modeling*
GPS	*Global Positioning System* – Sistema de Localização Global
I^2C	*Inter-Integrated Circuit* – Protocolo de conexão serial de circuto inter integrado
IMU	Inertial measure unit – Módulo com sensores que medem angulações e aceleração
ISO	*International Organization for Standardization* – Organização Internacional de Normalização
LIDAR	*Light Detection and Ranging* – Radar de rotativo por laser
MOT	*Moving Objects Detector* - Detector de Objetos em Movimento
SLA	*Stereolithography*
SLS	Selective Laser Sintering
SP	*Setpoint* – Valor Objetivo, Valor Destino
STL	*Stereolithography* ou *Standard Tessellation Language* – formato de arquivo
PG-1	Projeto de graduação 1
PG-2	Projeto de graduação 2
PV	*Process Variable* – Variável medida no Processo, Valor Real
PWM	Pulse width modulation – Modulação por controle de largura de pulso
VCC	*Voltage Common Collector* – usado para designer um cabo de alimentação de tensão contínua.
RRM	Receptor de radiofrequência e microcontrolador
TSD	*Traffic Signalization Detector* - Detector de Sinalização de Tráfico

1. INTRODUÇÃO

Automodelos, originalmente de controle remoto, apresentam uma plataforma simples, porém verossímil e de baixo custo para o estudo de testes e desenvolvimento de plataformas de guiagem autônoma.

1.1 CONTEXTUALIZAÇÃO DO TEMA E MOTIVAÇÃO

O fascínio de guiar automóveis sem a necessidade de um condutor dentro do veículo e automatizar suas funções está presente há gerações na sociedade. Apesar de empresas como a *Tesla Motors* estarem surpreendendo seu público com tecnologias de direção assistida, utilização de inteligência artificial e sensores de ponta em pleno ano de 2020, como é citado na introdução da tese [1], estas tecnologias vêm sendo estudadas e desenvolvidas desde o início do século XX.

Existem relatos na história de veículos controlados por rádio desde 1921, até carros conceito como o Firebird II, lançado pela General Motors (GM) na década de 50, que seriam controlados por sensores presentes nos carros e em estradas especiais. Mas, somente a partir da década de 80 que apareceram mudanças tecnológicas perceptíveis as quais impactaram significativamente a tecnologia utilizada nos dias atuais. O engenheiro alemão, Ernst Dickmanns, juntamente com sua equipe, segundo a revista Politico [2], desenvolveu um sistema denominado Vision Systems, que acoplado a uma van da marca Mercedes-Benz, foi capaz de percorrer trajetórias e perímetros urbanos com pouca interferência humana. A Van alterada é mostrada na Figura 1.

Figura 1. Van da Mercedes-Benz autônoma alterada por Ernst Dickmanns e sua equipe com o Vision System instalado.

Fonte: Foto do site de notícias ITIGIC [24]

Em 2004, a Agência de Projetos de Pesquisa Avançada de Defesa (DARPA) dos Estados Unidos da América, que já estava interessado em desenvolver tecnologias para carros autônomos com aplicação militar, deu início a um desafio com um prêmio inicial de um milhão de dólares. Nele, como é relatado na revista Discover Magazine [3], as equipes competidoras deveriam desenvolver um veículo off-road que fosse capaz de percorrer uma trajetória de 241 quilômetros sem nenhum tipo de interferência humana. Nessa primeira corrida, não houve nenhum vencedor, mas nos anos subsequentes, vários competidores conseguiram completar o percurso e desenvolveram grande parte das tecnologias que tornaram possível a realidade de direção assistida ou até autônoma que temos atualmente. Na Figura 2 é mostrada a imagem do veículo denominado "Stanley", o primeiro vencedor do Darpa Grand Challenge em 2005.

Figura 2. "Stanley", nome dado a um Volkswagen Touareg autônomo adaptado pela equipe da universidade de Stanford, vencedora do DARPA Grand Challenge de 2005.

Fonte: Site de notícias Popular Science [25].

Esse assunto tem despertado o interesse crescente da indústria, alguns governos de países estrangeiros já começaram a criar leis para fomentar o uso e desenvolvimento destes sistemas bem como o mercado privado vem reconhecendo cada vez mais sua importância. Isso é perceptível também pois novas pesquisas neste ramo aumentaram substancialmente em diversas universidades do mundo, como pode se observar nas teses [4], [5] e no artigo [6].

Além de deixar os meios de transporte mais eficientes, esse tipo de tecnologia pode ajudar a diminuir o número de mortes causadas por acidentes em estradas, que somam 1,35 milhão por ano, segundo a Organização Mundial da Saúde [7].

É importante destacar também a tese de Correia A. [36] realizada na Universidade de Brasília, a qual foi financiada pela Fiat usando um veículo de passeio do modelo "Palio" em que foi projetada,

desenvolvida, implementada e testada uma plataforma de controle e sensores para realizar o estacionamento do veículo em vagas paralelas de forma autônoma no ano de 2007.

Além disso é útil citar o artigo de BADUE, C. et al. [40] no qual é descrito como é composta a arquitetura geral do sistema de automação de veículos, mostrado na Figura 3, a qual esquematiza como os subsistemas que compõe os sistemas de Percepções e de Tomada de Decisões funcionam para deixar a guiagem do veículo autônoma.

Figura 3. Descrição dos Sistemas de percepção e tomada de decisões de um veículo autônomo.

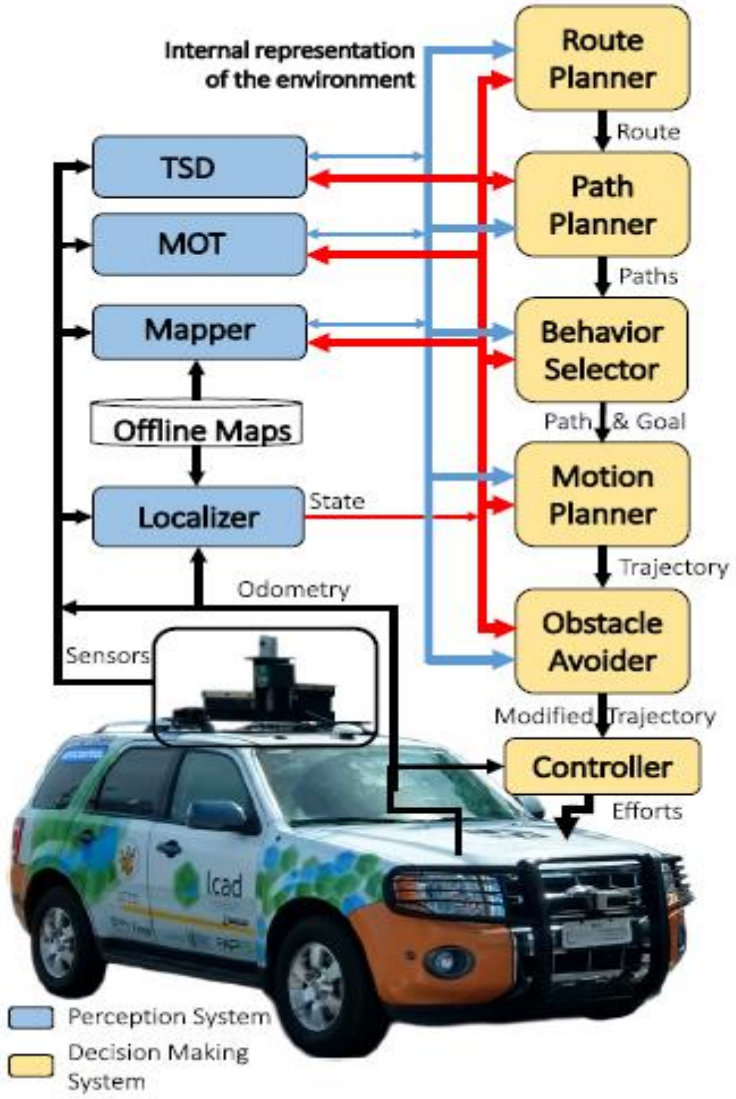


Figura 3. Descrição dos Sistemas de percepção e tomada de decisões de um veículo autônomo.

Fonte: Figura retirada do artigo de BADUE, C. et al. [40]

No artigo é explicado que o sistema de **Percepções** é responsável por estimar o estado do veículo, criando uma representação do ambiente a partir dos dados capturados por sensores presentes no veículo como LIDARs, IMUs, radares, câmeras, GPS, odômetros, dentre outros. Os dados recolhidos por estes sensores, serão enviados para o sistema de **Tomada de Decisões**, que irá processar tais informações e utilizá-las para poder navegar o veículo desde a sua posição inicial até a localização do

seu destino, a qual é definida previamente pelo seu usuário. Tudo isso é feito levando em conta as regras de trânsito, condições seguras de rodagem, assim como o conforto de seus passageiros.

O sistema de Percepções é composto por um *Localizer* (Localizador), que irá estimar posições, velocidades e acelerações do veículo em *Offline Maps* (Mapas Offline) que estão baixados no próprio sistema do veículo. Neles, as rotas e informações de trânsito já devem são computadas automaticamente antes do veículo começar a sua viagem. Assim o sistema de *Mapper* (Mapeador), recebe o *input* do estado do veículo e gera um *output online* do veículo no mapa, para atualizar a sua posição e ser alimentado com novas informações do caminho. Ademais, o *MOT-Moving Objects Detector* (Detector de Objetos em Movimento)e do *TSD-Traffic Signalization Detector* (Detector de Sinalização de Tráfego) servem para percepção de veículos, pedestres e outros obstáculos em movimento ao redor do veículo, assim como detecta sinalizações de trânsito como placas de pare e semáforos respectivamente.

O sistema de decisão é composto por um *Route Planner* (Planejador de Rotas) que computa a rota do início ao fim do destino, o *Path Planner* (Planejador de caminho) que irá utilizar as informações do primeiro, adicionadas de informações do status atual da localização do veículo, assim como o *Behavior Selector* (Seletor de Comportamento) que adiciona informações coletadas dos sensores do ambiente para computar uma localização objetivo, desviando de obstáculos e considerando a sinalização de trânsito. Assim, o *Motion Planner* (Planejador de Movimento) é responsável por computar a trajetória que o veículo de fato irá realizar, considerando todos as informações anteriores e considerando a dinâmica veicular, leis da física e conforto dos passageiros. O *Obstacle Avoider* (Sitsema de Evitar Obstáculos) pode mudar algum caminho do subsistema anterior se for necessário desviar de obstáculos percebidos na hora. E, por último, o controlador que recebe os dados do *Motion Planner* e traduz isso para controles dos atuadores (motores) necessários para mover o veículo.

AUTOMODELOS COMO PLATAFORMA DE ESTUDO

Um dos grandes desafios para o desenvolvimento destes sistemas é a necessidade de se obter grandes investimentos para comprar e manter os sensores, veículos, maquinário e unidades eletrônicas de processamento requeridas para viabilizar tais projetos.

Assim, visando dar início aos estudos de guiagem automática de veículos com uma baixa verba para pesquisa, os automodelos se apresentam como plataformas mais acessíveis e com grande potencial para o aprendizado, como desenvolver plataformas em uma escalas menores que servirá de base para empregar diversos estudos no âmbito de telemetria, instrumentação e controle de veículos, assim como foi feito e estudado diferentes modelos na tese [1].

Algumas marcas renomadas de automodelos como a “TRAXXAS” produzem unidades altamente confiáveis com chassis semelhantes aos de veículos automotores reais, além de sustentar horas de funcionamento com uma única carga de bateria. Por isso, foi escolhido o modelo 82056-4 como objeto de estudo deste projeto. Na Figura 4 é possível observar a estrutura do chassi, o sistema de suspensão e parte do *powertrain* do veículo. Além disso, este modelo conta com sistemas de diferenciais

dos eixos traseiro e dianteiro, um sistema de marcha com duas velocidas e tração nas quatro rodas. Mais detalhes sobre este serão abordados no capítulo 3, subeceção 5.

Figura 4. Chassi do automodelo "Model 82056-4" da marca Traxxas.

Fonte: Site da Traxxas, página de informações do modelo 82056-4 [23].

O automodelo da Figura 4 é comercializado com o sistema eletrônico-eletromecânico mostrado na Figura 5. Um rádio controle operado manualmente envia ao veículo os comandos via radiofrequência. Um receptor no veículo decodifica os comandos e gera ordens para os atuadores responsáveis por mudar o sistema de redução do motor (marcha), controlarem dois sistemas de travamento de sistemas diferenciais do eixo traseiro e dianteiro, assim como controlar o motor de propulsão e o sistema de direção. Uma das frentes de trabalho deste projeto é a substituição do conjunto radio-controle/receptor por um sistema embarcado capaz de comandar o veículo.

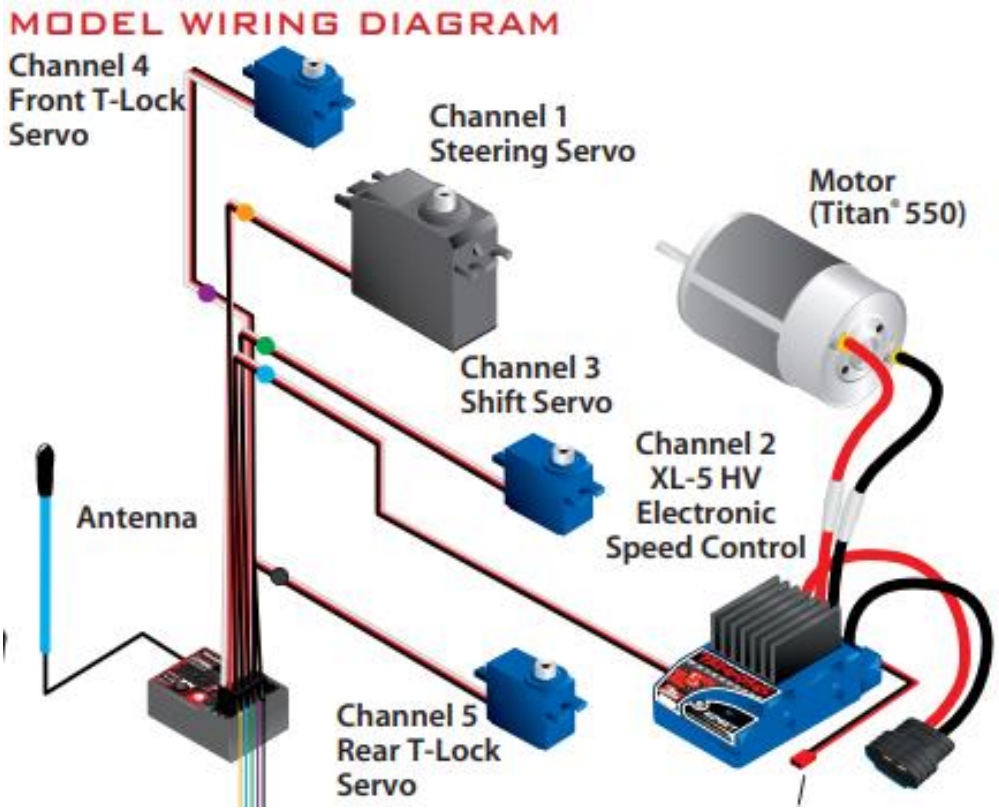


Figura 5. Diagrama de Subsistemas eletrônicas do automodelo Traxxas Model 82056-4

Fonte: Site da Traxxas, página de informações do modelo 82056-4 [23].

1.2 OBJETIVOS E ESTRUTURA DO TRABALHO

O presente trabalho tem o objetivo de instalar uma plataforma com sensores instrumentando parâmetros intrínsecos e extrínsecos ao veículo assim como um sistema eletrônico de arquitetura modular e programável/flexível que seja capaz de se conectar com todos os motores atuadores do automodelo e os sensores instalados. Desenvolvendo assim um software que seja capaz de receber comandos de trajetórias, adquirir dados do meio ambiente assim como realizar leituras de dados internos de acordo com sua dinâmica veicular aproximada. Estabelecendo assim uma plataforma de estudo base que sirva para futuros projetos que desenvolvam sistemas de automação e assistividade veicular.

Para a validação da plataforma, é proposto um estudo de caso com uma malha de controle de trajetória utilizando um PID e os sensores implementados.

Assim este documento será composto pelos seguintes capítulos:

- **Fundamentação teórica:** discorrendo sobre diferentes algoritmos para descrever a movimentação do veículo e outras definições necessárias para o correto entendimento das medições e conexões eletrônicas feitas durante o projeto e do funcionamento dos sensores e outros componentes.

- **Projeto e Instalação da Plataforma de Controle:** discorre sobre como foi montado cada componente da plataforma, os desafios encontrados, a realização do código e as medições características do veículo.
- **Testes Experimentais:** descrevendo os ensaios e resultados necessários para validar a instrumentação, trajetórias, assim como as análises dos resultados.
- **Conclusões:** mostrando um resumo do que se atingiu com este trabalho, fazendo uma análise compilada de todos os testes realizados, assim como uma discussão de possíveis estudos posteriores que poderão ser feitos a partir deste trabalho.

METODOLOGIA DE VALIDAÇÃO DO PROJETO E IMPLEMENTAÇÕES

Os testes realizados na seção 4 deste estudo foram feitos seguindo os princípios de prototipagem rápida (descritos na última seção do capítulo de revisão teórica) sendo compostos por pequenos ensaios com funcionalidades simples. A partir do sucesso de validação dos mesmos, as implementações eram iteradas, aumentadas em complexidade e aglutinadas para que se pudesse realizar testes mais completos até chegar ao ensaio final deste trabalho, que consiste em fazer o veículo percorrer uma trajetória previamente programada com correções ativas recebendo feedbacks de seus sensores.

O primeiro passo, porém, foi realizar a validação dos controles de todos atuadores do veículo por uma placa microcontroladora e desenvolver um sistema que pudesse contar a distância e velocidade longitudinal do veículo, implementar um sistema que pudesse fornecer informações de posição angular do veículo no espaço e por fim, desenvolver um código para que o veículo pudesse percorrer uma trajetória pré-programada.

Apesar de ser um estudo de implementação prática, com ensaios e medições, seu desenvolvimento foi feito durante a época de pandemia do COVID-19, onde se teve acesso restrito à laboratórios e instrumentações adequadas, sendo muitas vezes necessário o uso de instrumentos de medidas robustos como réguas, trenas e transferidores no lugar de goniômetros, paquímetros, dentre outros, assim como a realização de testes em lugares limitados.

Pode-se assim, colocar de forma objetiva os seguintes tópicos utilizados para validar o estudo:

1. Testes preliminares de compatibilidade dos atuadores já presentes no veículo para verificação de viabilidade de projeto;
2. Instalação de suporte/invólucro impresso em 3D para comportar sensores e eletrônicos;
3. Instalação de magnetômetro, módulo IMU, velocímetro/odômetro e sensores de proximidade;
4. Testes de controle dos atuadores de potência e direcional para percorrer trajetória em linha reta;
5. Testes de guiagem em linha reta para calibrar odômetro;
6. Testes de funcionamento do magnetômetro para aquisição de posição angular;
7. Testes de funcionamento do IMU para aquisição de posição angular;

8. Testes de performance entre sensor IMU e magnetômetro para a determinação de posição angular;
9. Teste de implantação de controle PID;
10. Teste de guiagem em linha reta com correções feitas por controle PID utilizando um magnetômetro;
11. Teste de controle de velocidade utilizando PID;
12. Teste de validação do estudo realizando a guiagem do automodelo por uma trajetória simples previamente simulada seguindo um modelo veicular estudado.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo faz um breve resumo das informações teóricas que foram utilizadas na parte de implementação e funcionamento da plataforma.

2.1 A FÍSICA POR TRÁS DA TRAJETÓRIA VEICULAR

Como é detalhadamente mostrado no artigo [7], existem vários métodos e modelos para descrever a trajetória de um veículo. Neste estudo, serão abordados os modelos cinemáticos e dinâmicos simplificados de duas rodas. Para ter um melhor entendimento da física por trás destes modelos de trajetória, podemos dividir os assuntos em duas grandes áreas: a cinemática e a dinâmica veicular.

- A cinemática veicular é responsável por fazer uma descrição geométrica dos movimentos no espaço. Engloba assuntos como diferentes referenciais e sistemas de coordenadas.
- Já a parte de dinâmica veicular, também chamada de cinética, versa sobre as leis físicas que causam o movimento, como os efeitos das forças e momentos nas equações de Newton-Euler.

2.1.1 CINEMÁTICA VEICULAR

SISTEMA REFERENCIAL

A cinemática veicular deve ser estudada com um sistema de referencial padrão. Como é necessário estudar a trajetória do veículo que se movimenta e rotaciona em um espaço 3D, definimos as referências de acordo com a norma ISO 8855 em [8], estabelecendo um sistema cartesiano referencial inercial composto pelos eixos **X**, **Y** e **Z** e um referencial móvel composto pelos eixos $\mathbf{X_V}$, $\mathbf{Y_V}$ e $\mathbf{Z_V}$ mostrado na Figura 6. Este sistema também é utilizado na literatura escrita por Jazar (2009) [9].

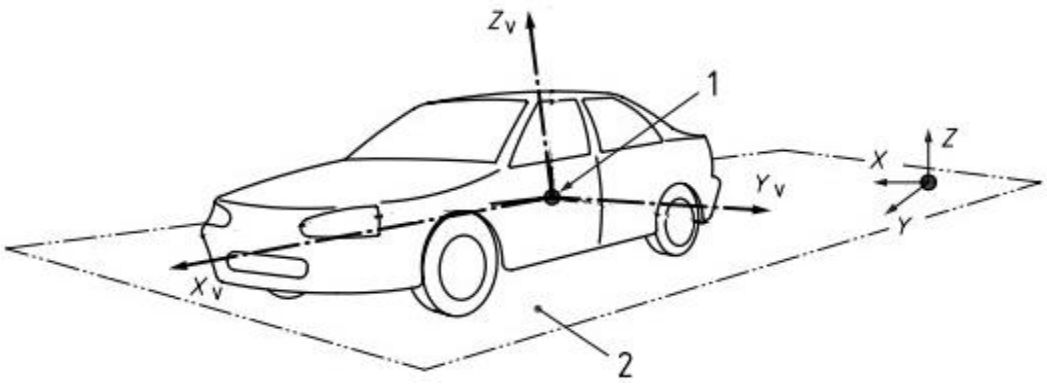


Figura 6. Sistema de coordenadas de acordo com a norma ISSO 8855. O número 1 mostra o ponto de referência do veículo e o 2 representa o plano horizontal do chão fixo onde o veículo se locomove.

Fonte: Elaborado pelo autor. Adaptado da norma ISO 8855 (2011) [8].

Referencial Inercial: Composto pelos eixos **X**, **Y** e **Z** fixado no plano horizontal do chão de modo que o plano formado por **X** e **Y** (longitudinal) se encontra perpendicular ao vetor aceleração da gravidade e o eixo **Z** é vertical com mesma direção e sentido oposto ao vetor aceleração gravitacional.

Referencial do Veículo: É o referencial móvel composto pelos eixos **XV** (apontando para a frente do veículo), **YV** (apontando para a lateral esquerda) e **ZV** (apontando para o topo). Este sistema tem a sua origem coincidente com o centro de massa do carro.

Referencial da Trajetória: É um referencial móvel complementar que tem a sua origem sempre coincidente com a trajetória descrita pelo veículo. Composta pelos eixos **d**, **e** e **n**. Os quais **d** sempre está na direção tangencial à trajetória, e está no plano horizontal do chão apontando para a esquerda da direção do primeiro e **n** é perpendicular ao plano horizontal do chão. A origem da curva da trajetória é representada pela letra **s**. As informações escritas estão ilustradas na Figura 7.

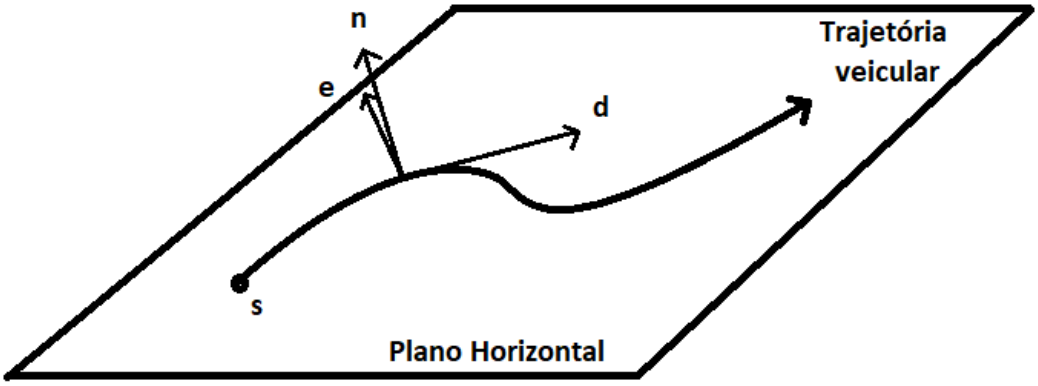


Figura 7. Sistema referencial de trajetória.

Fonte: Elaborado pelo autor.

MEDIDAS PRINCIPAIS DO VEÍCULO

Ainda de acordo com a terminologia estabelecida pela ISO 8855, é necessário definir as medidas principais que caracterizam cada veículo estudado. Estas são necessárias para calcular a trajetória percorrida de acordo com o ângulo esterçado. Na Figura 8 pode-se verificar a **distância entre eixos** dada por **l** e a **bitola** dada pela letra **b**.

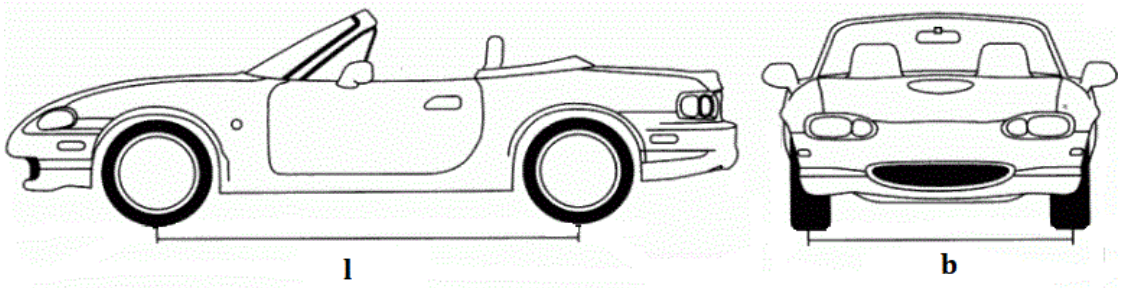


Figura 8. Medidas principais dos veículos. Entre eixos (l), Bitola(b) e altura(h)

Fonte: Elaborado pelo autor. Adaptado de site [10].

VELOCIDADE E ACELERAÇÃO DO VEÍCULO

Para se obter as relações de velocidade Eq. (1) e aceleração Eq. (2) do veículo, basta obter as derivadas de primeira e segunda ordem, respectivamente, da distância **rP(t)** de um ponto **P** nas coordenadas **X(t)**, **Y(t)** e **Z(t)** com relação ao tempo.

$$\vec{\mathrm{V}}(\mathrm{t}) = \dot{\vec{r}}_P(t) = \begin{bmatrix} \dot{X}(t) \\ \dot{Y}(t) \\ \dot{Z}(t) \end{bmatrix} \tag{1}$$

$$\vec{\mathrm{a}}(\mathrm{t}) = \ddot{\vec{r}}_P(t) = \begin{bmatrix} \ddot{X}(t) \\ \ddot{Y}(t) \\ \ddot{Z}(t) \end{bmatrix} \tag{2}$$

Ao se projetar as velocidades e acelerações obtidas no plano horizontal do veículo, obtemos as seguintes denominações de acordo com a Tabela 1 e Tabela 2.

Tabela 1. Velocidades no sistema de coordenadas móvel do veículo.

	Denominação da velocidade	**Símbolo**
Direção X:	Velocidade longitudinal	$V_{lon}(t)$
Direção Y:	Velocidade lateral	$V_{lat}(t)$
Direção Z:	Velocidade vertical	$V_{vert}(t)$

Tabela 2. Aceleração no sistema de coordenadas móvel do veículo.

	Denominação da aceleração	**Símbolo**
Direção X:	Aceleração longitudinal	$a_{lon}(t)$
Direção Y:	Aceleração lateral	$a_{lat}(t)$
Direção Z:	Aceleração vertical	$a_{vert}(t)$

ANGULOS ENTRE OS SISTEMAS DE COORDENADAS

Estabelece-se as denominações dos ângulos entre os sistemas de coordenadas inercial e horizontal do veículo como: **guinada** (ângulo entres os eixos $\boldsymbol{X}$ e $\boldsymbol{X_V}$ que coincide com o ângulo de rotação do eixo $\mathbf{Z_V}$, dado por $\boldsymbol{\Psi}$), **arfagem** (ângulo entres os eixos $\boldsymbol{Z}$ e $\mathbf{Z_V}$ que coincide com o ângulo de rotação do eixo $\mathbf{Y_V}$ dado por $\boldsymbol{\theta}$) e **rolamento** (ângulo entres os eixos $\boldsymbol{Y}$ e $\boldsymbol{Y_V}$ que coincide com o ângulo de rotação do eixo $\mathbf{X_V}$ dado por $\boldsymbol{\varphi}$). A imagem na Figura 9 ilustra os ângulos dispostos neste parágrafo.

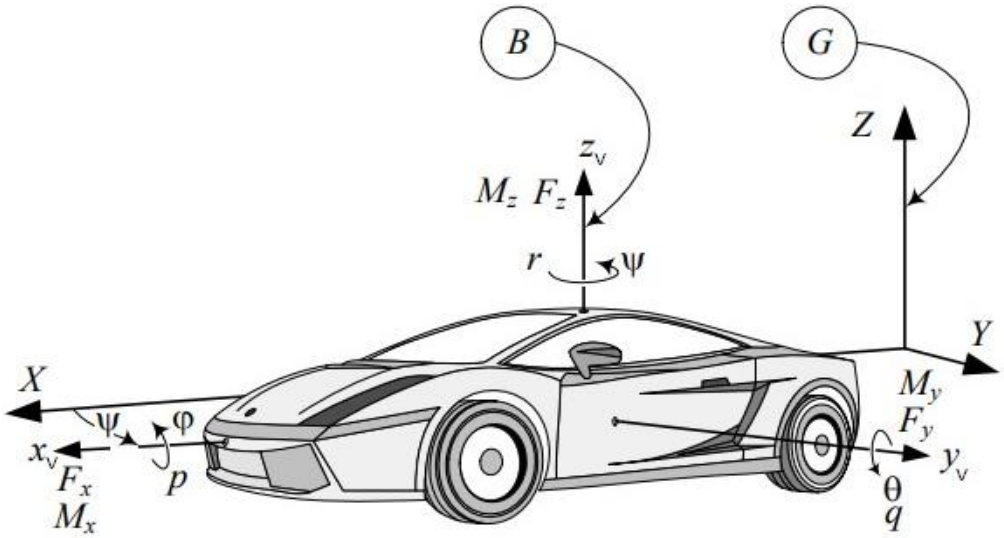


Figura 9. Ângulos de rotação do sistema fixo para o sistema móvel.

Fonte: Elaborado pelo autor. Adaptado de Jazar, Vehicle Dynamics, 2009 [9].

SISTEMAS DE DIREÇÃO: (PARALELO, ACKERMAN E ANTI-ACKERMAN)

Segundo Georg Schildbach (2020) [13], existem três tipos principais de sistemas geométricos de direção. Cada um tem suas características principais, vantagens e desvantagens.

O primeiro deles, e mais simples, é o **sistema de direção de rodas paralelas**. Este, que pode ser observado no centro da Figura 10, tem uma construção extremamente simples, porém não é utilizado na maioria dos veículos comerciais modernos, uma vez que não entrega uma boa estabilidade. Nos próximos parágrafos será explicado em mais detalhes os outros três tipos.

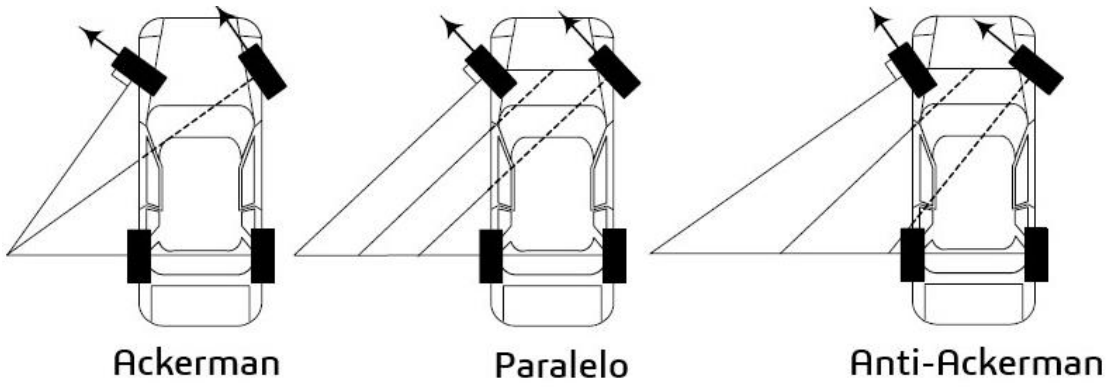


Figura 10. Os três tipos de sistemas de direção.

Fonte: Elaborado pelo autor. Adaptado de Jazar, Vehicle Dynamics, 2009 [9].

Isto ocorre pois, durante uma curva, cada uma das quatro rodas do veículo está percorrendo uma circunferência de raios diferentes, tendo em vista que o veículo tem o mesmo centro de rotação com relação ao seu centro de gravidade, porém cada roda está disposta em distâncias diferentes. E, desta forma, as rodas da frente, responsáveis pela direção do veículo, deveriam estar em ângulos diferentes para percorrer naturalmente o percurso sem haver deslizamento. Isto é demonstrado na Figura 11.

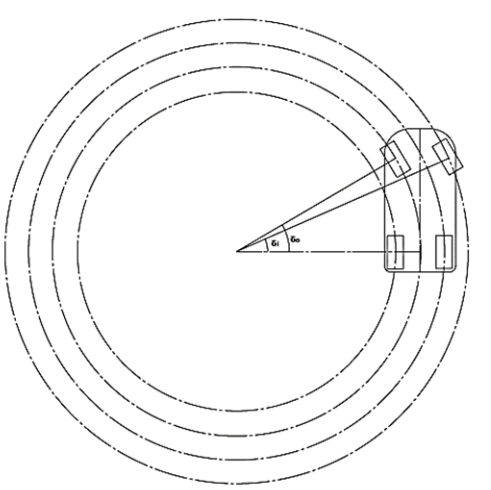


Figura 11. Os raios de curvatura de cada uma das rodas durante uma curva.

Fonte: Elaborado pelo autor.

O **sistema baseado na geometria direcional de Ackerman** corrige justamente o problema apresentado no parágrafo anterior. Segundo Jazar (2009) [9], um empreendedor Anglo-Alemão, chamado Rudolf Ackerman, foi responsável por patentear em 1816 um sistema direcional de carroças de 4 rodas e estudou a condição geométrica necessária para os ângulos das rodas. Esta condição, posteriormente conhecida como "condição de Ackerman" diz que sob baixas velocidades, para que um veículo possa realizar uma curva de Raio **R** (medida do centro de rotação da curva até o ponto coincidente com o centro de massa do veículo), os ângulos de esterçamento das rodas frontais interna e externa à curva (responsáveis pelo sistema direcional) devem ter valores diferentes entre si e proporcionais à diferença angular entre $\boldsymbol{X_V}$ e à direção perpendicular à linha que liga o centro da curva ao ponto que coincide com o centro de cada respectiva roda frontal. Esta condição é mostrada na Figura 12 e é também a utilizada no automodelo deste estudo.

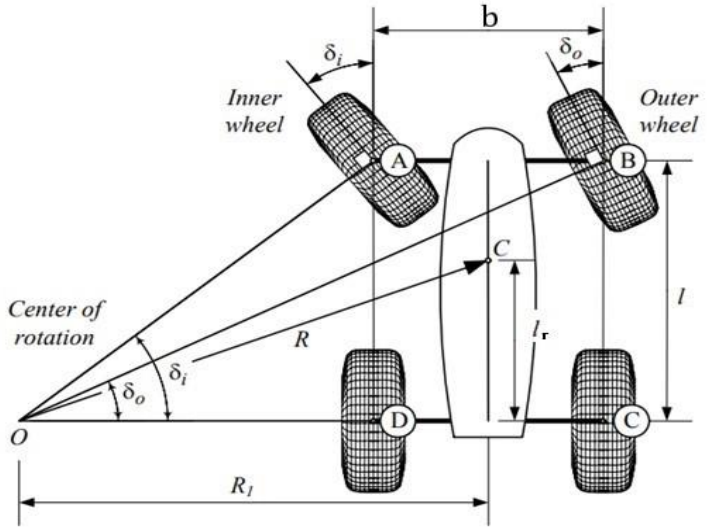


Figura 12. Ângulo de Ackerman das rodas frontais para o raio da curva e medidas principais do veículo.

Fonte: Elaborado pelo autor. Adaptado de Jazar, Vehicle Dynamics, 2009 [9].

O **centro de rotação da curva** recebe o símbolo de ***O***, enquanto o raio da curvatura ***R*** que o veículo descreve é calculado pela distância entre seu respectivo centro de massa ***C*** e ***O***.

Os ângulos $\boldsymbol{\delta_i}$ (direcional da roda interna) e $\boldsymbol{\delta_o}$ (direcional da roda externa) do sistema de Ackerman devem, portanto, respeitar a relação mostrada na Eq. (3).

$$cotg\ (\delta_o)\ -\ cotg\ (\delta_i)\ = \frac{b}{l} \tag{3}$$

Pode-se obter o **ângulo de esterçamento do veículo** $\boldsymbol{\delta}$ por meio de uma média com as tangentes de $\boldsymbol{\delta_i}$ e $\boldsymbol{\delta_o}$ que corresponde ao ângulo de uma roda frontal imaginária que estaria instalada em cima da linha longitudinal do veículo coincidente com o centro de massa e consequentemente colinear ao eixo $\mathbf{X_V}$. Calculamos $\boldsymbol{\delta}$ com a Eq. (4).

$$\delta = \ \mathrm{cotg}^{-1}\left(\frac{cotg\ (\delta_o) + \ cotg\ (\delta\_i\ )}{2}\right) \tag{4}$$

Assim, pode-se calcular facilmente o raio da curvatura **R** com relação às medidas do carro e do ângulo de esterçamento do veículo de acordo com a Eq. (5).

$$R = \sqrt{{l_r}^2 + l^2\ cotg^2(\delta)} \tag{5}$$

É importante notar que $\boldsymbol{l_r}$ corresponde à distância entre o centro do eixo das rodas traseiras e o centro de massa do veículo.

Na Eq. (5) pode-se também isolar $\boldsymbol{\delta}$ e deixar a equação em função de deste ângulo também. Trata-se de uma relação útil para o presente trabalho e o resultado disto está disposto na Eq. (6).

$$\delta = \frac{1}{\mathrm{arctg}\left(\sqrt{\frac{R^2 - l_r^2}{l^2}}\right)} \tag{6}$$

O último sistema de direção mencionado neste documento, tem uma **geometria anti-Ackerman**, na qual, de acordo com Milliken e Milliken (1995) [12], as rodas ao mudar o ângulo de esterçamento fazem uma compensação no sentido oposto ao do Ackerman. Os projetistas geralmente utilizam esta razão apenas em veículos de corrida, o qual é submetido a fortes forças laterais, a altas velocidades e a acelerações bruscas na maior parte do tempo de sua utilização causando um **ângulo de deriva** nas curvas. Assim, esta geometria compensa tal ângulo que ocorre devido à deflexão dos pneus ocasionada pelos altos esforços laterais que o carro sofre. Esta geometria é ilustrada na Figura 10, à

direita. Na seção 3.5.3 deste estudo, é possível verificar a direção do automodelo utilizado, sendo medida e comprovando seguir o padrão de geometria Ackerman.

O primeiro e último sistemas de direção citados nesta seção não são interessantes para o projeto e não serão abordados em muitos detalhes. É importante notar também que segundo Jazar (2009) [9], nenhum sistema real de direção consegue obedecer exatamente à condição de Ackerman, mas é possível obter uma aproximação boa o suficiente em vários pontos de esterçamento com mecanismos relativamente simples de múltiplos links.

MODELO CINEMÁTICO DE DUAS RODAS: EQUAÇÕES DE MOVIMENTO

O modelo cinemático de duas rodas, também conhecido como "Modelo da Bicicleta", pode ser entendido como uma simplificação, assim como descrito por Jazar (2009) [9], na qual as duas rodas da frente e as duas rodas traseiras do modelo de 4 rodas são aglutinadas no meio do veículo alinhado com a linha longitudinal, criando uma espécie de bicicleta imaginária colinear com o eixo $\mathbf{X_v}$ do veículo. Assim, mantém-se o mesmo centro instantâneo de rotação, mas é possível simplificar os cálculos de trajetória por diminuir o número de rodas e ângulos importantes.

Este modelo tem a vantagem de apresentar cálculos mais simplificados com relação aos modelos de quatro rodas apesar de que, ainda assim, apresenta um conjunto de equações diferenciais de primeiro grau. Suas limitações são que ele apresenta uma boa aproximação para condições de baixas velocidades, nas quais há pouca deformação dos pneus (ângulo de deriva $\boldsymbol{\alpha} \rightarrow 0$) e não há deslizamento dos pneus com o chão. Esta conversão pode ser observada na Figura 13.

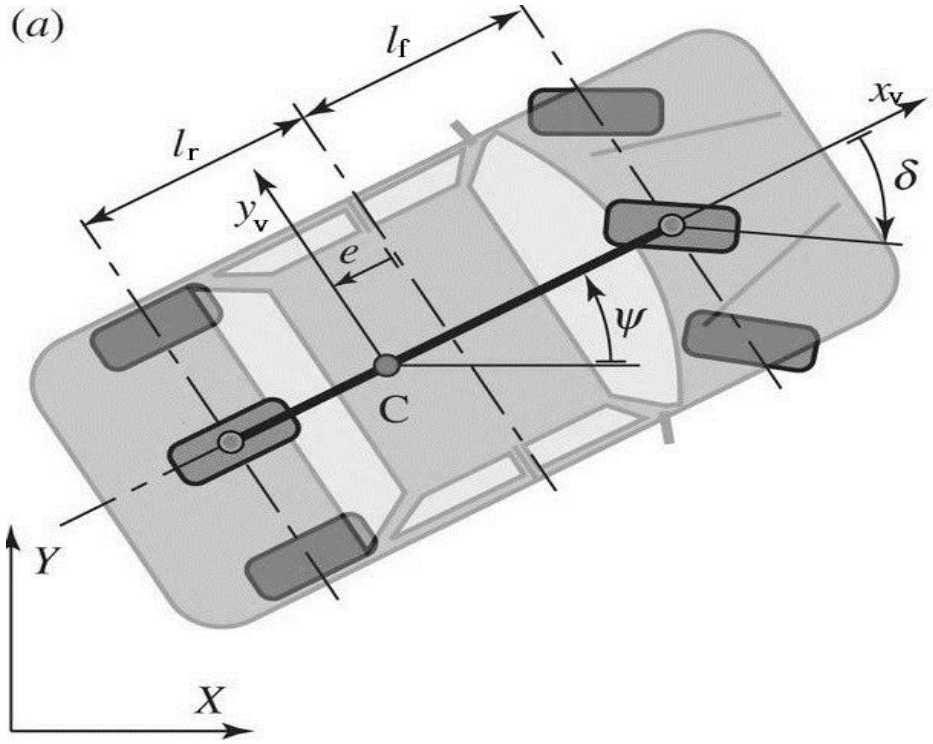


Figura 13. Aglutinação das rodas do modelo completo de 4 rodas para o modelo simplificado de bicicleta.

Fonte: Elaborado pelo autor. Adaptado de TAKÁCS D & STÉPÁN G, 2013 [11].

Assim, assumindo que o modelo de duas rodas supracitado esteja percorrendo uma curva de raio ***R*** a uma velocidade V_{lon} constante (para haver condição de curvas em estado estacionário) com relação ao seu centro de massa ***C*** e direção perpendicular ao raio da curva, como a mostrado na Figura 14, pode-se deduzir as equações de velocidade do modelo com relação ao referencial fixo no chão.

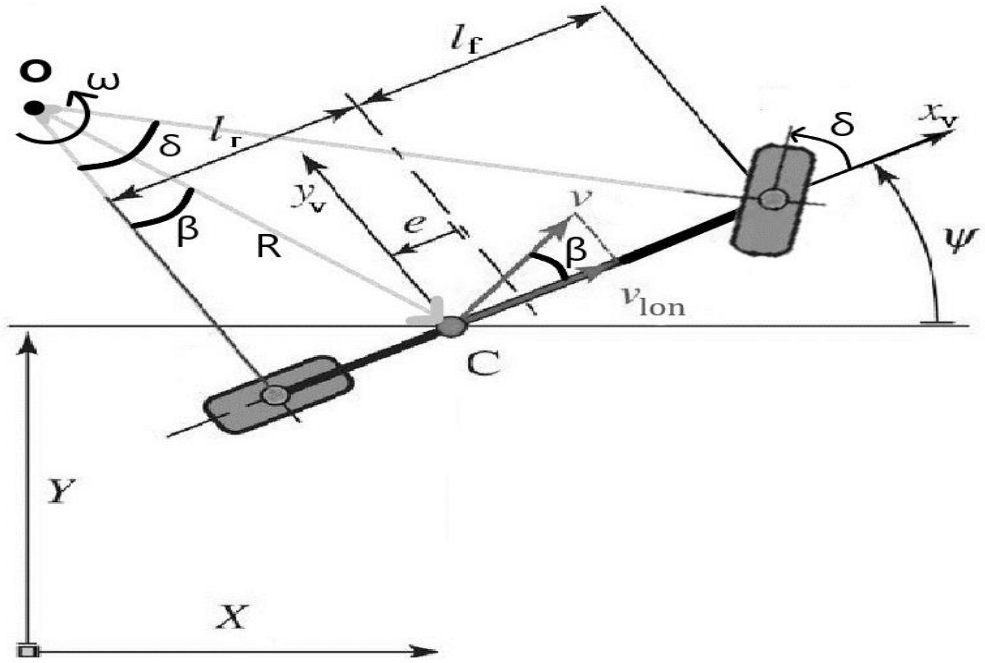


Figura 14. Modelo da bicicleta percorrendo um raio R do centro de rotação instantâneo até o centro de massa C.

Fonte: Elaborado pelo autor. Adaptado de TAKÁCS D & STÉPÁN G, 2013 [11].

As equações Eq. (7), Eq. (8) e Eq. (9) conseguem calcular as derivadas de primeira ordem de: ***X(t)***, ***Y(t)*** e do ângulo ***Ψ(t)***. Note que foi escolhido deixar a notação $\dot{\boldsymbol{X}}(\boldsymbol{t})$ e não outro símbolo de velocidade relacionado à letra "***v***" para enfatizar que neste caso, as velocidades obtidas estão no sistema de coordenadas fixo, também chamado de global.

$$\dot{X}(t) = \mathrm{V}\cos(\Psi + \beta) \tag{7}$$

$$\dot{Y}(t) = \mathrm{V}\,sen(\Psi + \beta) \tag{8}$$

$$\dot{\Psi}(t) = \omega = \frac{V}{R} \tag{9}$$

Sendo $\boldsymbol{\omega}$ a velocidade angular em torno do centro de rotação ***O*** e $\boldsymbol{\beta}$ sendo o ângulo formado entre a direção da velocidade do veículo e o seu eixo $\boldsymbol{X_V}$ ou pela a diferença entre a linha que forma o raio da curva entre o centro da curva ***O*** com o centro da roda traseira e o raio ***R.*** O valor de $\boldsymbol{\beta}$ pode ser definido na Eq. (10).

$$\beta = arctg\left(\frac{l_r}{l_r + l_f}\, tg(\delta)\right) \tag{10}$$

Considerando o valor de $\boldsymbol{\beta}$, podemos também, escrever a primeira derivada de $\boldsymbol{\Psi(t)}$ como está disposto na Eq. (11).

$$\dot{\Psi}(t) = \frac{V}{l_r + l_f}\cos(\beta)\, \mathrm{t}g(\delta) \tag{11}$$

Como no veículo que será estudado neste trabalho, não é possível obter o valor instantâneo da velocidade longitudinal do veículo $\boldsymbol{V_{lon}}$ (lembrando que esta é referente ao sistema móvel) por não haver nenhum sensor previamente instalado. Pode-se calcular facilmente o valor do vetor velocidade $\boldsymbol{V}$ utilizando a Eq. (12).

$$\mathrm{V} = V_{lon}\cos(\beta) \tag{12}$$

Existem também alguns estudos, nos quais são modificados alguns parâmetros desse modelo original como pode ser observado, por exemplo, no de Moritz Werling (2010) [15], que acrescentou *inputs* das taxas de velocidade angular do esterçamento entre outros parâmetros.

2.1.2 DINÂMICA VEICULAR

RESUMO SOBRE O QUE OCORRE EM UM PNEU

De acordo com o livro de Reimpell (2002) [16], a interação dos pneus com o solo é responsável pela parte mais significativa do comportamento dinâmico do veículo. Isto é confirmado por todos grandes nomes da literatura de dinâmica veicular que dedicam pelo menos um capítulo para abordar este tópico. Mas ainda assim, seus modelos são pouco utilizados no projeto de carros comuns, pois estes representam um dos maiores graus de complexidade na obtenção de equações que descrevem o movimento dos veículos de formas condizentes com a realidade, pelo fato de até hoje apresentarem uma série de dados variados, não lineares e de difícil medição.

Pequenas alterações em suas dimensões, técnicas construtivas e aerodinâmica podem alterar características significativas em um projeto de veículos de corrida. Neste trabalho, porém, será disposto uma pequena parcela do que é necessário entender sobre eles para que se possa compreender o modelo dinâmico abordado.

A primeira consideração que é preciso observar é a de que durante o contato do pneu com o solo, pode-se perceber que há uma pequena deformação devido à imposição da força peso do veículo distribuída neste material flexível e, dessa forma, o **raio efetivo da roda $\boldsymbol{R_{ef}}$** é diferente do **raio nominal**

R. Assim, durante os cálculos de distância e velocidade através de suas rotações é necessário medir e utilizar o raio efetivo. Esta medida é ilustrada na Figura 15.

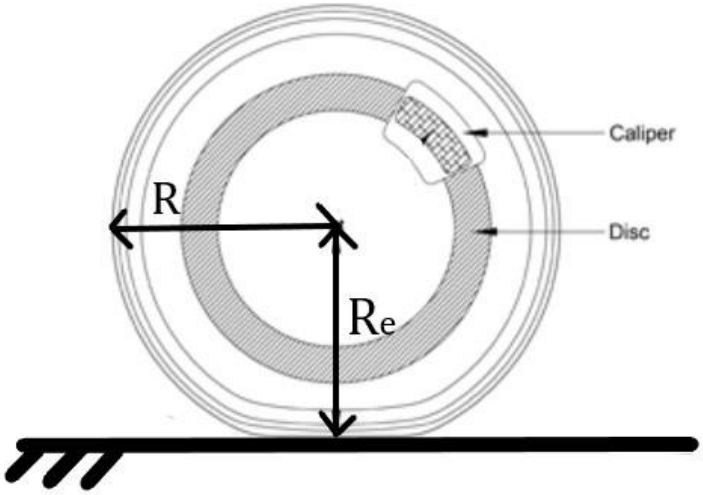


Figura 15. Raio efetivo *(Ref)* versus raio nominal *(R).*

Fonte: Elaborado pelo autor. Adaptado do site Grove (2020) [22].

Na seção 3.5.4 é mostrada a diferença visual entre as rodas do veículo com a presença e sem a presença de carga. Seu raio efetivo é calculado pela primeira vez com uma régua. Após isto, nas seções de testes, são feitos novos cálculos.

Segundo Milliken & Milliken (1995) [12], durante a movimentação de um veículo, principalmente em altas velocidades, os pneus também podem sofrer a ação de acelerações laterais e longitudinais. Levando em conta que os mesmos são feitos de borracha, é muito comum haver deformações em volta da sua área de contato com o solo. A deformação mais importante para este modelo é causada pelas forças laterais durante uma curva, denominada de **ângulo de deriva** α ou do inglês, *slip angle* e *side slip angle*. Esta deformação é responsável por mudar ligeiramente a direção que o pneu está de fato impondo sobre o solo, o que ocasiona uma diferença no ângulo de esterçamento real do veículo como pode ser explanado na Figura 16.

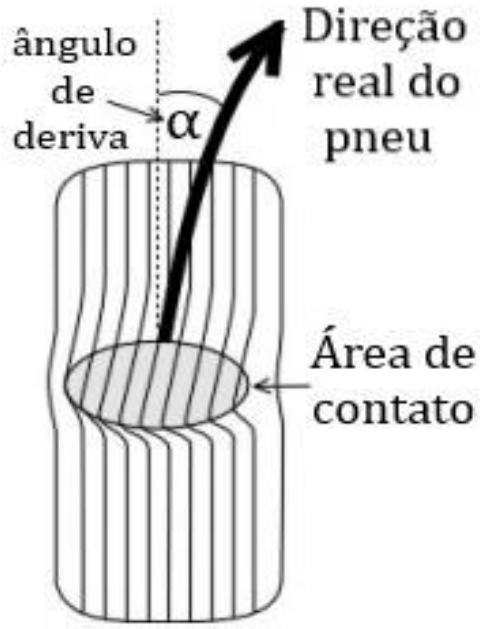


Figura 16. Ilustração do ângulo de deriva, também chamado de slip angle.

Fonte: Elaborado pelo autor. Adaptado da dissertação de Diego Bravo (2013) [19].

O ângulo de deriva geralmente é dividido entre cada um dos pneus. Como no modelo estudado neste documento considera a simplificação de duas rodas aglutinadas, separamos em $\boldsymbol{\alpha_f}$ (ângulo de deriva nos pneus frontais) e $\boldsymbol{\alpha_r}$ (ângulo de deriva nos pneus traseiros).

Outro parâmetro importante que será introduzido nas próximas equações é o coeficiente de dureza lateral dos pneus $\boldsymbol{C_{\alpha f}}$ (frontal) e $\boldsymbol{C_{\alpha r}}$ (traseiro). Estes são medidos, em N/rad e podem ser obtidos por modelos analíticos ou experimentais. Após obter as informações necessárias, tenta-se fazer uma aproximação para transformar os resultados em dados lineares que funcionam relativamente bem para velocidades baixas a moderadas (velocidades que não causem uma força grande o suficiente para ultrapassar a força de atrito estática entre o pneu e o solo, de modo que se utilize sempre o coeficiente de atrito estático).

Este coeficiente faz parte de uma aproximação linear de modelagem da sua deformação, que leva em conta apenas a deformação causada por esforços laterais sem considerar deslizamento entre a borracha e o solo. Trata-se do **modelo linearizado de pneu** e foi o escolhido para ser utilizado no modelo dinâmico do veículo que será apresentado posteriormente.

Existem diversos outros modelos, como os empíricos estudados no livro de Pacejka (2002) [17], etc. No artigo de Hongbin Ren, et al. (2014) [18], porém, modelos lineares e não-lineares são comparados, no qual, o primeiro mostra bons resultados em baixas velocidades, que é o caso do automodelo estudado neste documento.

MODELO DINÂMICO DE DUAS RODAS (BICICLETA)

Para que seja possível determinar uma equação do movimento a partir das forças e momentos que atuam sobre o veículo, é necessário definir alguns conceitos prévios e condições onde este modelo pode atuar. Na Figura 17 são mostradas todas as forças e momentos atuantes nos pneus e rodas. O modelo dinâmico da bicicleta, basicamente aglutina as quatro rodas em duas e transpõe a somatória de todas as forças e momentos atuantes nos pneus e corpo para o centro de massa do veículo para facilitar os cálculos.

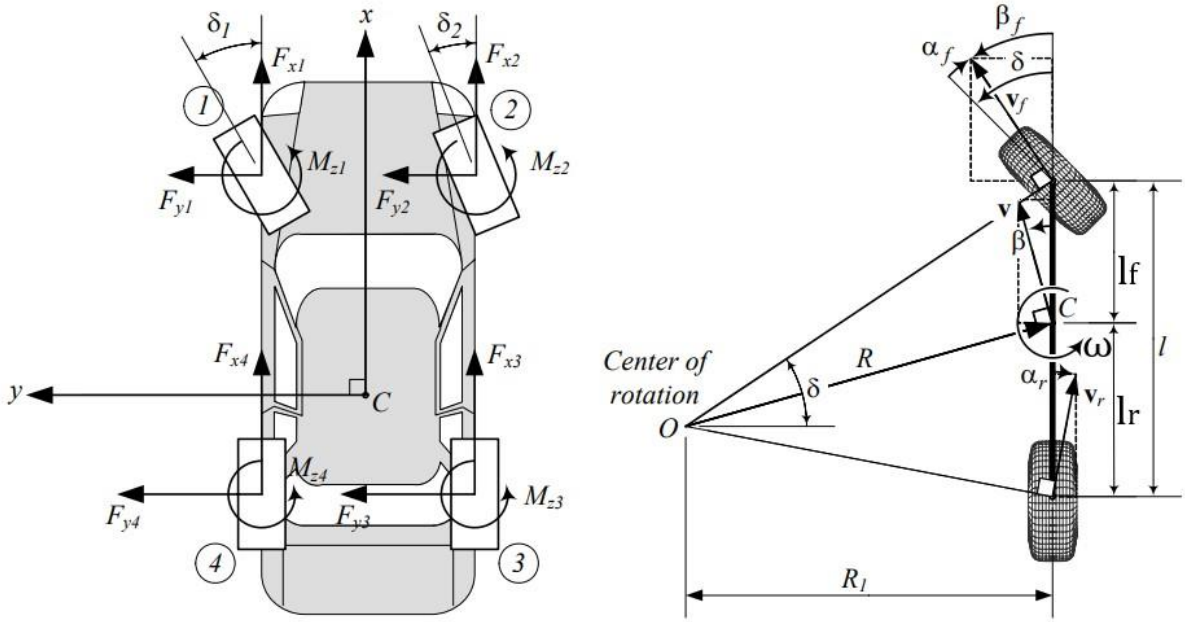


Figura 17. Ilustração das forças e momentos atuantes em um veículo no plano X-Y à esquerda e a representação do modelo dinâmico linear à direita com seu centro de referência local.

Fonte: Elaborado pelo autor. Adaptado de Jazar, Vehicle Dynamics, 2009 [9].

A equação de resolução deste assunto é determinada por uma equação diferencial, porém esta, exige alto poder de processamento o que consequentemente geraria um grande tempo de resposta dos microcontroladores responsáveis por guiar o veículo. Assim, uma solução encontrada é uma aproximação linearizada desta equação, que podemos conseguir se limitarmos algumas condições de funcionamento como ângulos de esterçamento pequenos e baixas velocidades.

Como é possível, implementar sensores relativamente baratos que conseguem disponibilizar algumas informações dessas equações de modo instantâneo, existe a possibilidade de que alguns cálculos possam ser facilitados. Ainda assim, a equação dinâmica, mesmo sendo simplificada para duas rodas, apresenta uma complexidade de implementação muito mais complexa que a cinemática.

Assim, segundo Jazar (2009) [9], as considerações que devemos fazer para continuar com o modelo dinâmico do veículo, são:

- O Chassi do veículo, assim como sua estrutura, é considerado um corpo rígido;
- O ponto **C** que marca a origem do sistema referencial móvel, ou local, está localizado no centro de massa do veículo;
- O tipo de movimento é considerado 2D e quasi-planar no plano **X-Y**, portanto as rotações $\boldsymbol{\theta}$ e $\boldsymbol{\varphi}$ são nulas assim como os momentos causados por estes ângulos. Ou seja, as rotações dos planos $\boldsymbol{X_v}$ e $\boldsymbol{Y_v}$ com relação ao sistema local do veículo aplicadas no centro de massa $\boldsymbol{C}$ do veículo, denotados como $\boldsymbol{M_{vx,C}}$ e $\boldsymbol{M_{vy,C}}$ respectivamente;
- Consideram-se apenas forças laterais no pneu calculadas por um modelo linear;
- A velocidade longitudinal $\mathbf{V_{lon}}$ do veículo é constante;
- Apenas as rodas dianteiras são esterçantes.

Assim, as Eq. (13) à (15) correspondem às três equações do modelo dinâmico bidimensional de duas rodas. Note que todos os vetores inclusos nela estão referenciados ao sistema local.

$$\dot{V}_{lon} = \mathrm{F_x} + w\, V_{lat} \tag{13}$$

$$\dot{V}_{lat} = \frac{1}{m\, V_{lon}} \left(-l_f\, C_{\alpha f} + l_r\, C_{\alpha r}\right)\omega - \frac{1}{m\, V_{lon}} \left(C_{\alpha f} + C_{\alpha r}\right) V_{lat} + \frac{1}{m} \left(C_{\alpha f}\delta_f\right) + \frac{1}{m} \left(C_{\alpha r}\delta_r\right) - \omega\, V_{lon} \tag{14}$$

$$\dot{\omega} = \frac{1}{I_Z\, V_{lon}} \left(-l_f^2\, C_{\alpha f} - l_r^2\, C_{\alpha r}\right)\omega - \frac{1}{I_Z\, V_{lon}} \left(l_f\, C_{\alpha f} - l_r\, C_{\alpha r}\right) V_{lat} + \frac{1}{I_Z} \left(l_f C_{\alpha f}\delta_f\right) - \frac{1}{I_Z} \left(l_r C_{\alpha r}\delta_r\right) \tag{15}$$

Para facilitar o entendimento, todos os parâmetros destas equações estão mostrados na Tabela 3.

Tabela 3. Componentes utilizados nas Eq. 13, 14 e 15.

Símbolos:	**Descrição:**
$\boldsymbol{F_x}$	Força resultante na direção $\boldsymbol{X_V}$ do sistema de coordenadas local aplicada no centro de massa $\boldsymbol{C}$;
$\boldsymbol{\dot{V}_{lat}}$	Aceleração local na direção $\boldsymbol{Y_V}$ do sistema de coordenadas local aplicada no centro de massa $\boldsymbol{C}$;
$\boldsymbol{\dot{V}_{lon}}$	Aceleração local na direção $\boldsymbol{X_V}$ do sistema de coordenadas local aplicada no centro de massa $\boldsymbol{C}$;
$\boldsymbol{\dot{\omega}}$	Aceleração angular em torno do eixo $\boldsymbol{Z_V}$ do sistema de coordenadas local aplicada no centro de massa $\boldsymbol{C}$;
$\boldsymbol{m}$	Massa do veículo;
$\boldsymbol{V_{lon}}$	Velocidade longitudinal, local na direção $\boldsymbol{X_V}$ do sistema de coordenadas local aplicada no centro de massa $\boldsymbol{C}$;
$\boldsymbol{V_{lat}}$	Velocidade lateral, local na direção $\boldsymbol{Y_V}$ do sistema de coordenadas local aplicada no centro de massa $\boldsymbol{C}$;
$\boldsymbol{\omega}$	Velocidade angular em torno do eixo $\boldsymbol{Z_V}$ do sistema de coordenadas local aplicada no centro de massa $\boldsymbol{C}$;
$\boldsymbol{C_{\alpha f}\ e\ C_{\alpha r}}$	Coeficientes de dureza lateral dos pneus frontal e lateral respectivamente
$\boldsymbol{l_f\ e\ l_r}$	Distâncias entre o centro de massa $\boldsymbol{C}$ e os eixos das rodas frontal e traseiro respectivamente;
$\boldsymbol{I_Z}$	Momento de Inércia do corpo com relação ao centro de massa $\boldsymbol{C}$.

As Eq. (14) e (15) podem ser dispostas em um formato matricial também. A Eq. (16) demonstra essas equações de uma forma mais organizada para facilitar o entendimento.

$$\begin{bmatrix} \dot{V}_{lat} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} -\dfrac{C_{\alpha f} + C_{\alpha r}}{m\, V_{lon}} & \dfrac{-l_f\, C_{\alpha f} + l_r\, C_{\alpha r}}{I_Z\, V_{lon}} - V_{lon} \\ -\dfrac{l_f\, C_{\alpha f} - l_r\, C_{\alpha r}}{I_Z\, V_{lon}} & -\dfrac{l_f^2\, C_{\alpha f} + l_r^2\, C_{\alpha r}}{I_Z\, V_{lon}} \end{bmatrix} \begin{bmatrix} V_{lat} \\ \omega \end{bmatrix} + \begin{bmatrix} \dfrac{C_{\alpha f}}{m} & \dfrac{C_{\alpha r}}{m} \\ \dfrac{l_f\, C_{\alpha f}}{I_Z} & -\dfrac{l_r\, C_{\alpha r}}{I_Z} \end{bmatrix} \begin{bmatrix} \delta_f \\ \delta_r \end{bmatrix} \tag{16}$$

A Eq. (17) mostra outra forma de escrever, porém, com o parâmetro da velocidade angular do ângulo de deriva, dada por $\boldsymbol{\dot{\beta}}$:

$$\begin{bmatrix}\dot{\beta}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix} -\frac{C_{\alpha f} + C_{\alpha r}}{m\, V_{lon}} & \frac{-l_f\, C_{\alpha f} + l_r\, C_{\alpha r}}{I_Z\, V_{lon}^2} - 1 \\ -\frac{l_f\, C_{\alpha f} - l_r\, C_{\alpha r}}{I_Z} & -\frac{l_f^2\, C_{\alpha f} + l_r^2\, C_{\alpha r}}{I_Z\, V_{lon}} \end{bmatrix} \begin{bmatrix}\beta\\ \omega\end{bmatrix} + \begin{bmatrix} \frac{C_{\alpha f}}{m} & \frac{C_{\alpha r}}{m} \\ \frac{l_f\, C_{\alpha f}}{I_Z} & -\frac{l_r\, C_{\alpha r}}{I_Z} \end{bmatrix} \begin{bmatrix}\delta_f\\ \delta_r\end{bmatrix} \quad (17)$$

Ao se observar a Eq. (16), pode-se simplifica-la também como uma simples equação explicitada como Eq. (18):

$$\dot{\boldsymbol{q}} = [A]\, \boldsymbol{q} + \boldsymbol{u} \quad (18)$$

Na qual $[\boldsymbol{A}]$ é uma matriz com valores constantes, **q** são as variáveis de controle e $\boldsymbol{u}$ é o vetor de entradas da equação.

2.2 COMPONENTES ELETRÔNICOS E ELETROMECÂNICOS

Nesta seção serão mostrados os principais componentes eletrônicos e eletromecânicos utilizados neste projeto.

2.2.1 PROCESSAMENTO EMBARCADO

Um sistema embarcado, ou embutido, é um sistema microprocessado, no qual o computador se encontra embutido em um aparelho e desempenha tarefas específicas para este, isto os diferencia de computadores de propósitos gerais como desktops. No geral, para sistemas embarcados possuem uma capacidade de processamento reduzida e, ao invés de utilizarem microprocessadores, utilizam microcontroladores, uma vez que o último possui diversos periféricos integrados no mesmo chip.

Atualmente, o uso de microcontroladores reprogramáveis de baixo custo vem sendo extensivamente utilizados no desenvolvimento de projetos, e até em alguns produtos finais. São exemplos desses, as placas das famílias Arduino, ESP, entre outras presentes no mercado.

Suas vantagens são ter tamanhos pequenos, baixo consumo de energia e custos acessíveis. Por outro lado, possuem uma capacidade mais limitada de processamento, geralmente executando uma tarefa por vez. Assim, ao se aumentar o número de sensores e atuadores ligados a esses controladores e quanto mais complexa a inteligência embarcada, mais demorado será o seu processamento e mais otimizado deverá ser o conjunto *hardware/software* do sistema de processamento.

Na seção 3.1 deste estudo é explicado com mais detalhes o processo de escolha do microcontrolador deste projeto.

2.2.2 MOTORES DC E CONTROLADORES ESC

Os motores de corrente contínua ou DC (*Direct Current*) são componentes eletromecânicos rotativos que convertem corrente elétrica contínua em energia mecânica de rotação. Neste projeto, esse dispositivo é utilizado como mecanismo de propulsão, entregando torque às rodas.

Os motores DC têm as vantagens de serem controlados facilmente mudando-se a DDP entre seus pólos de alimentação, de poderem ser revertidos, freados ou acelerados de forma rápida e de possuírem altas forças de arranque. Suas desvantagens são: geralmente são um pouco mais volumosos e caros que motores de corrente alternada, possuem maior manutenção devido ao desgaste das escovas e não podem ser operados em ambientes explosivos, devido à possibilidade de ocorrerem centelhas entra as escovas e os comutadores.

Como relatado por Mattede R. em [41], um motor DC simples é composto pelos seguintes componentes:

- **Enrolamento de Campo:** localizado na parte fixa do motor, denominada de estator. Nele existem fios enrolados ou ímãs que vão formar polos magnéticos Norte e Sul que constituem um campo de excitação que irá interagir com o rotor de forma a move-lo;
- **Enrolamento de armadura:** localizado na parte móvel do motor denominada de rotor. São fios enrolados por onde se passa corrente elétrica, gerando torque elétrico que movimenta o rotor.
- **Comutador:** responsável por fazer a corrente seguir pelas bobinas certas enroladas no rotor, de forma que o torque gerado esteja sempre no mesmo sentido;
- **Escovas:** geralmente construídas a partir de carvão, tem a função de fazer o contato do enrolamento de armadura com a fonte de alimentação DC.

Desta forma, a corrente DC flui pelo comutador onde é entregue às boninas certas do rotor, de forma que elas interajam com o campo do estator e gerem um torque resultante contínuo que faça o rotor girar. Os componentes do motor podem ser observados na Figura 18.

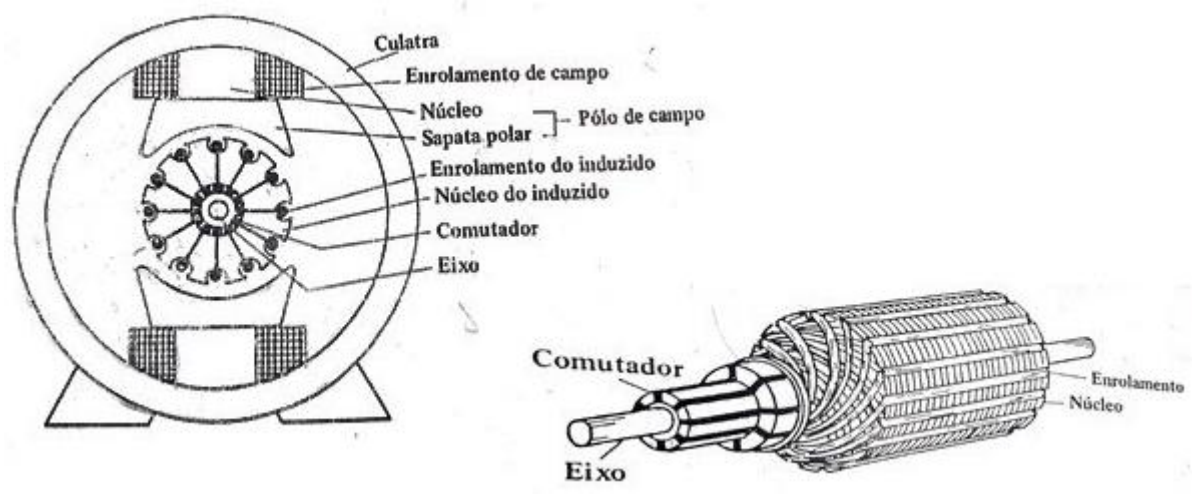


Figura 18. Componentes do Motor DC.

Fonte: Site Mundo da Elétrica [41]

Os sistemas ESC (*Electronic Speed Control*) são circuitos eletrônicos que controlam e regulam a velocidade de diferentes tipos de motores elétricos. Alguns desses dispositivos também podem fazer funções de reversão e de frenagem dinâmica.

Os controladores ESC aplicados em modelos de controle remoto, geralmente recebem um sinal PWM e energia de uma fonte de energia. Assim, de acordo com o PWM interpretado, o sistema varia a tensão de saída para entregar diferentes velocidades de rotação ou até mudar a polaridade para cambiar a direção de rotação.

2.2.3 SERVOMOTORES (OU SERVOMECANISMOS)

Servomecanismos são dispositivos eletromecânicos controlados por um sistema que é composto por sensores e moduladores que realizam leituras de grandezas, como posição, velocidade ou aceleração dos atuadores, enviando feedbacks contínuos de correção dos possíveis erros dessas grandezas para os moduladores. Este sistema é denominado de “malha fechada”.

Dentro de um servomotor é possível encontrar pelo menos um motor de corrente contínua, geralmente com uma redução integrada, possibilitando alto torque, porém velocidades baixas. Existem dois tipos: os servomotores de rotação (múltiplas voltas) e os de posição (precisão angular), sendo a última opção, a que é utilizada no atual projeto.

Todos os servomecanismos possuem pelo menos um dispositivo de controle, um de comando, um detector de erro e um amplificador do sinal de erro, assim como o dispositivo que irá de fato realizar a movimentação do mecanismo para corrigir os erros existentes.

Os primeiros servomotores foram desenvolvidos e utilizados para sistemas de artilharia e navegação marinha. Hoje, porém, com a difusão da tecnologia, podem ser encontrados em máquinas de

precisão, como as de usinagem, as de controle de antenas rastreadoras de satélites, mas também em formatos simples e de baixo custo como em carros de controle remoto utilizados por crianças e robistas.

Este documento abordará os dispositivos de mais simples construção, utilizado por robistas e pequenos projetos de prototipagem, que serão mencionados no Capítulo 4. Para serem controlados, estes sistemas devem ser ligados a três canais. Os dois primeiros são responsáveis por criarem uma DDP e fornecer uma corrente, entregando a energia necessária para o correto funcionamento do motor. O terceiro canal entrega um sinal, que no caso, indica a **posição objetivo** ou *setpoint*, via sinal PWM que o servomotor deve se encontrar. Um exemplo pode ser observado na Figura 19.

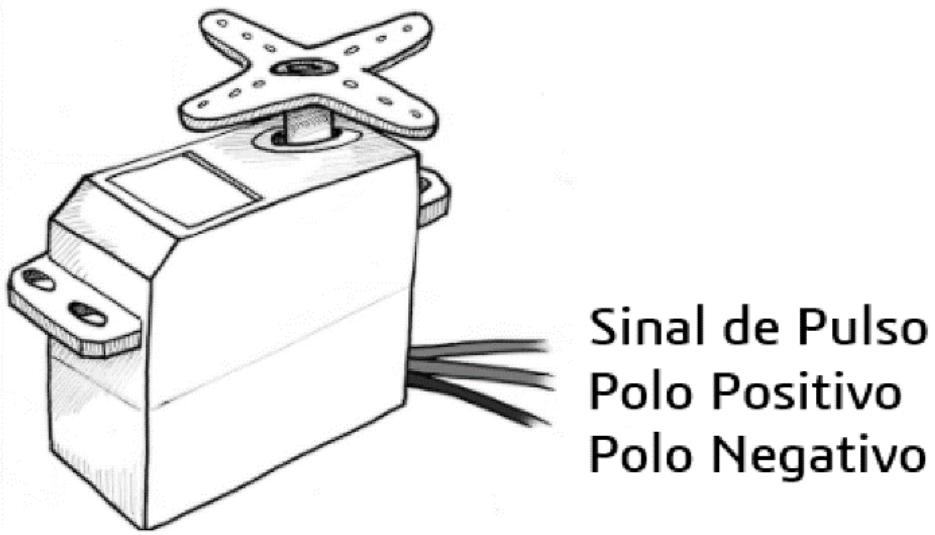


Figura 19. Os três canais de um servomotor robista.

Fonte: Elaborado pelo autor. Adaptado do site Backyard Brains [27].

Dentro do invólucro externo do dispositivo mostrado há um sensor e uma unidade de processamento que mede constantemente sua **posição real**, que é comparada com a **posição objetivo,** sendo enviados sinais de correção para a posição angular correta. Assim, ao se enviar um sinal de **posição objetivo** angular de 33°, por exemplo, e posteriormente, ao se aplicar uma força externa no eixo do servomotor para tirá-lo deste posicionamento, será observado que o motor fará uma força oposta procurando manter sempre a posição previamente ajustada.

Na grande maioria dos servomotores de robistas, o canal de sinal é enviado em forma de pulso elétrico do tipo PWM, explicado no tópico anterior deste documento. Estes são lidos por seu dispositivo de comando e passados para um pequeno motor DC que gira o suficiente para posicionar o eixo na angulação desejada. Isto, os tornam extremamente versáteis e fáceis de se controlar com placas microcontroladoras de prototipagem, além de representarem uma ótima escolha para sistemas de direção de automodelos guiados por controle remoto.

2.3 FUNCIONAMENTO DE PWM

“Pulse width modulation" (PWM), ou "modulação por largura de pulso", é uma técnica empregada comumente para realizar o controle de componentes eletrônicos, como servomotores, luminosidade, dentre outros. É formada basicamente por uma onda quadrada de frequência e consequentemente, períodos constantes, porém com uma largura de pulso que varia.

Esta técnica de Modulação é utilizada para fundir dados com uma onda de potência denominada portadora. Cada tipo de modulação possui vantagens e desvantagens com relação a características como atenuação, robustez, custo, etc. As ondas AM, que possuem modulação por amplitude, e FM, que possuem modulação por frequência, são exemplos clássicos. Já o PWM é adotado geralmente para transmitir informações simples em sistemas não-críticos.

O funcionamento de suas ondas funcionam, portanto, da seguinte forma: o dispositivo eletrônico controlador que envia a frequência, cria tais ondas quadradas com um período $\mathbf{T_{total}}$ de valor constante (geralmente em torno de 10.000µs para pequenos servomotores, como os utilizados neste trabalho) no qual a largura do pulso mostrada na Figura 20 de exemplo, varia de acordo com faixas estabelecidas pelos fabricantes.

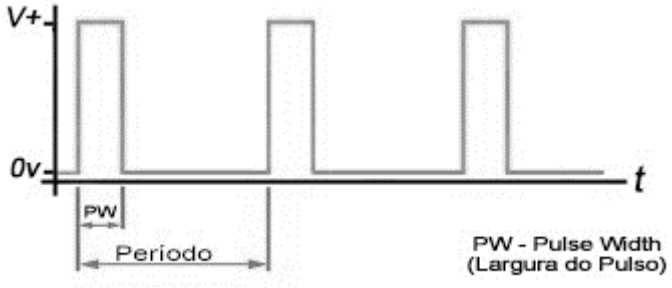


Figura 20. Representação gráfica de uma onda PWM

Fonte: Site Mecaweb [21].

Esta largura de Pulso $\boldsymbol{D_C}$ é comumente chamada de “*duty cycle*” ou ciclo de trabalho e corresponde à quantidade de tempo na qual a tensão está no seu valor máximo (sinal ativo) durante um período completo. Esse valor é comumente medido também em percentagem de acordo com a Eq. 19. ou simplesmente em μs calculado pela Eq. 20.

$$D_C = \frac{T_{sinal\ ativo}}{T_{total}} . 100 \tag{19}$$

$$T_{sinal\ ativo} = T_{total} - T_{sinal\ inativo} \tag{20}$$

Dessa forma, os componentes eletrônicos acessórios que recebem a onda elétrica possuem um circuito responsável por fazer a leitura do tamanho destas faixas e, então, traduzir tais valores em posições de motores, frações de uma potência disponível, quantidade luminosa, velocidade, etc.

No caso dos equipamentos utilizados neste estudo, os valores variam de aproximadamente 10 a 20% do tamanho de ***Dc***. No capítulo 3 este assunto é citado novamente para mostrar os resultados medidos nos servomotores que o veículo do presente documento utiliza.

2.4 SENSORES

Nesta seção é explicado o funcionamento e alguns cálculos matemáticos relacionados às operações realizadas com os dados adquiridos pelos sensores que compõe a instrumentação do projeto. Eles podem ser subdivididos em **intrínsecos** (responsáveis por medir os parâmetros do sistema) e **extrínsecos** (responsáveis por medir parâmetros externos ao sistema).

2.4.1 SENSORES INTRÍNSECOS

Os sensores extrínsecos utilizados neste projeto, são: acelerômetro (mede acelerações), giroscópio (mede velocidade angular), odômetro (construído com um sensor infravermelho, mede as rotações do eixo cardan de tração que são convertidas para distância percorrida).

MÓDULOS IMU

Os móduloa de Unidade de Medida Inercial (IMU ou *Inertial Measurement Unit*) são módulos eletrônicos que juntam um conjunto de sensores capazes de apresentar uma gama de graus de liberdade com dados relacionados ao posicionamento dele no espaço. Sua aplicação é ampla nos campos de engenharia aeroespacial e militar, como é reforçado no artigo de Arid Ainur [34], mas ficaram mais populares e acessíveis a partir de 2015, pois são muito utilizados em diversos tipos de drones, nos quais desempenham um papel importante para manter seu voo estável, fornecer informações complementares para sistemas de *GPS*, dentre outros.

Os principais sensores dentro de um IMU são:

- **Giroscópio** de 3 eixos, para medir a velocidade angular em torno dos 3 eixos **X**, **Y** e **Z**;
- **Acelerômetro** de 3 eixos, para medir a aceleração linear nos 3 eixos **X**, **Y** e **Z**.

Alguns ainda apresentam:

- **Magnetômetro** de 3 eixos, para medir a intensidade do fluxo magnético nos eixos **X**, **Y** e **Z**;
- **Barômetro** para determinação de altitude via pressão atmosférica;

- **Termômetro** para medir a temperatura pois ela pode influenciar na medição dos outros sensores.

Este módulo será muito interessante pois todas estas informações podem ser de grande valia para determinar informações de posição, que são importantes para os cálculos de trajetória do veículo. Dentre os quais será estudado com mais importância os sensores magnéticos e de giroscópio.

A maioria dos módulos IMU disponíveis no mercado de baixo custo, possuem sensores inerciais com tecnologia MEMS (Micro Electro-Mechanical Systems), que são implementações microscópicas dos sensores vistos acima em encapsulamentos de silício.

FUNCIONAMENTO DE UM GIROSCÓPIO ELETRÔNICO

Giroscópios eletrônicos, como explicado no artigo de Arif Rafiq [34], consistem de uma pequena massa composta por um material condutor e presa por pequenas molas que é movimentada dentro de um invólucro também condutor. Quando está sobre ações de rotação, é possível interpretar pelo controlador, desta forma, a intensidade de sua velocidade em radianos por segundo (rad/s). Uma imagem da construção da placa pode ser vista na Figura 21.

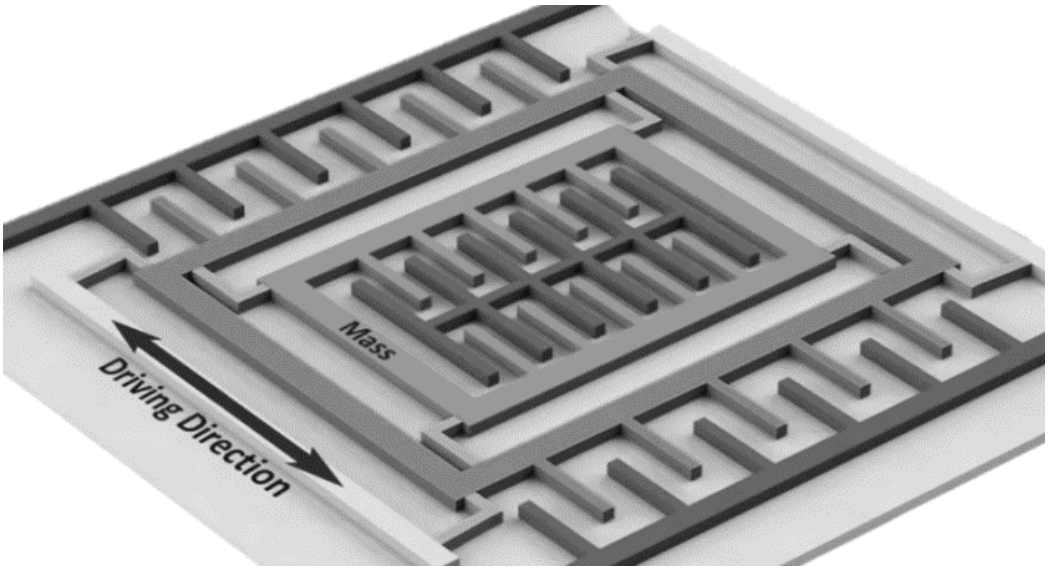


Figura 21. Ilustração dos componentes internos de um giroscópio eletrônico.

Fonte: Adaptado do site “How to Mechatronics” em [30].

A informação fornecida pelo giroscópio em radianos por segundo pode ser facilmente transformada em graus por segundo (°/s) de acordo com a Eq. (21.)

$$\omega_{\text{graus}} = \omega_{\text{rad}} \times \frac{180}{\pi} \tag{21}$$

Além de ser possível obter a velocidade angular, pode-se calcular a posição angular relativa do sensor ao se integrar a velocidade angular no tempo de acordo com a Eqs. (22) e (23).

$$\Psi = C_1 + \int_{t_0}^{t} \omega \, dt \tag{22}$$

$$\Psi = \Psi_0 + \omega \, (\mathrm{t} - \mathrm{t}_0) \tag{23}$$

Onde $\mathbf{C_1}$ corresponde a uma constante de integração. No sistema de instrumentação de um veículo, por exemplo, isso seria o ponto de partida pelo qual se começou a contar a somatória das integrações. Ou seja, o ângulo inicial de quando se começou a medir e depois os resultados sucessivos de uma medição antes da que está sendo calculada. Os limites de integração são dados por **t** e $\mathbf{t_0}$ que correspondem ao tempo no qual se iniciou e terminou cada uma das medições de $\boldsymbol{\omega}$, que por sua vez é a velocidade angular. Estas equações podem ser verificadas também na documentação de Romain Fetick [35].

FUNCIONAMENTO DE UM ACELERÔMETRO ELETRÔNICO

Acelerômetros eletrônicos, como explicado por Costa G. em [38] mostram as acelerações nos três eixos $\mathbf{X_v}$, $\mathbf{Y_v}$ e $\mathbf{Z_v}$ que são medidas por acelerômetros adequadamente posicionados. Como os acelerômetros contêm erros, as leituras devem ser compensadas removendo vieses fixos ou aplicando fatores de escala.

Cada um dos acelerômetros presentes funciona medindo as mudanças de capacitância ocorridas durante a sua movimentação e transformando sinais que serão interpretados em metros por segundo quadrado. Sua microestrutura possui uma massa com placas fixadas a molas pela qual ela se movimenta em uma região limitada por placas externas. Dessa forma, quando o sensor sofre uma aceleração, a massa se move e a capacitância varia. Na Figura 22 é demonstrada uma representação de como é a construção do sensor.

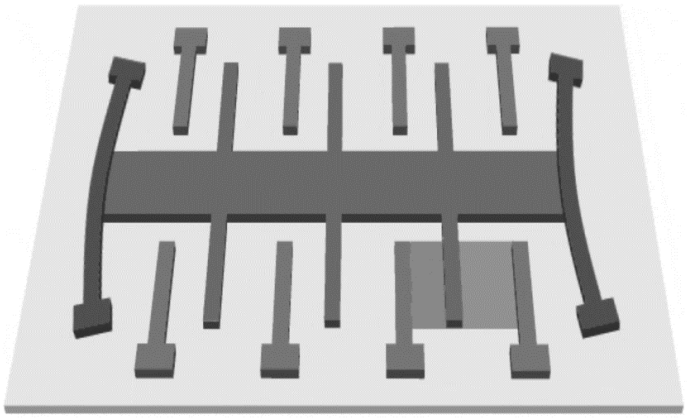


Figura 22. Representação construtiva de um acelerômetro MEMS.

Fonte: adaptado do site LME [42]

As medidas dos vetores de aceleração interpretadas podem ser integradas uma ou duas vezes pelo tempo para se obter a velocidade e distância vetorial também. Neste estudo, porém, esta não será

uma prioridade, visto que é muito comum o acúmulo de erros destas medições ao longo do tempo, deixando-as imprecisas.

FUNCIONAMENTO DE SENSORES INFRAVERMELHOS, ODÔMETRO E VELOCÍMETRO

Sensores infravermelhos, como explicado no artigo de Ardrsh S. [31], utilizam uma radiação eletromagnética na frequência infravermelha (uma luz fora do espectro eletromagnético da visão humana) que é irradiada, refletida em algum obstáculo e recebida por um receptor de luz infravermelha. O receptor de luz infravermelha consegue detectar a diferença de intensidade luminosa capturada por ele. Assim, pode-se obter a leitura convertida para uma diferença de potencial diretamente proporcional à distância com que certo obstáculo está do sensor, determinando assim a presença ou calculando a distância de um certo objeto ao sensor. Uma simplificação do seu funcionamento pode ser observada na Figura 23.

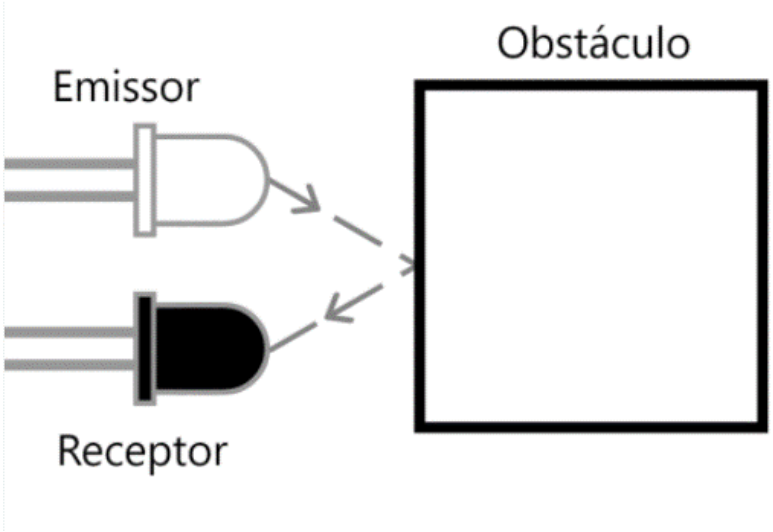


Figura 23. Funcionamento de um sensor infravermelho.

Fonte: Retirado do site “Mundo Projetado” em [32].

Apesar de permitir o cálculo de distâncias, a leitura deste sensor pode variar de acordo com a transparência, luz, temperatura de objetos e até mesmo com a cor do objeto de reflexão. Portanto, estas limitações fazem com que este sensor seja utilizado predominantemente em lugares mais controlados, onde não haja nenhum risco de perda de acurácia dos valores muito significativa, ou simplesmente utilizar um tratamento de dados compatível com informações mais grosseiras, como um sensor de presença.

Tais características geralmente são utilizadas como um sensor extrínseco, porém, neste trabalho, o sensor infravermelho foi usado para criar um odômetro para que se fosse possível contar as voltas do eixo do veículo e assim determinar sua distância percorrida. Como o sensor infravermelho é influenciado pela distância e reflexividade dos objetos (inclusive a cor), foi possível criar uma peça que usasse essas

características a favor da aquisição de sinais altos e baixos para transformar o sensor em um contador de voltas. Estes detalhes estão descritos na seção 3.3.2.

2.4.2 SENSORES EXTRÍNSECOS

Sensores extrínsecos medem parâmetros externos ao sistema e, neste projeto, pode-se citar o magnetômetro (que indica a direção do Norte Magnético da Terra) e o sensor de ultrassom (que mede a distância de obstáculos no ambiente).

FUNCIONAMENTO DE UM MAGNETÔMETRO

Magnetômetros (ou sensores magnéticos), segundo R. Grössingerr [28], são dispositivos que utilizam o princípio de efeito *hall* para detectar o campo magnético ao seu redor, que neste caso, o objetivo é quantificar o campo magnético da Terra. Estes sensores são amplamente utilizados em equipamentos das áreas de engenharia aeroespacial, de agricultura e militar, além de estarem presentes em grande parte dos *smartphones* utilizados pelo público civil hoje em dia.

O funcionamento dos sensores magnéticos abordados neste trabalho consiste em uma placa de FeNi (composto de níquel e ferro) que, de acordo com mudanças no fluxo magnético a que está exposta, muda a disposição dos elétrons em sua estrutura, alterando assim a sua resistividade. Uma breve ilustração pode ser vista na Figura 24.

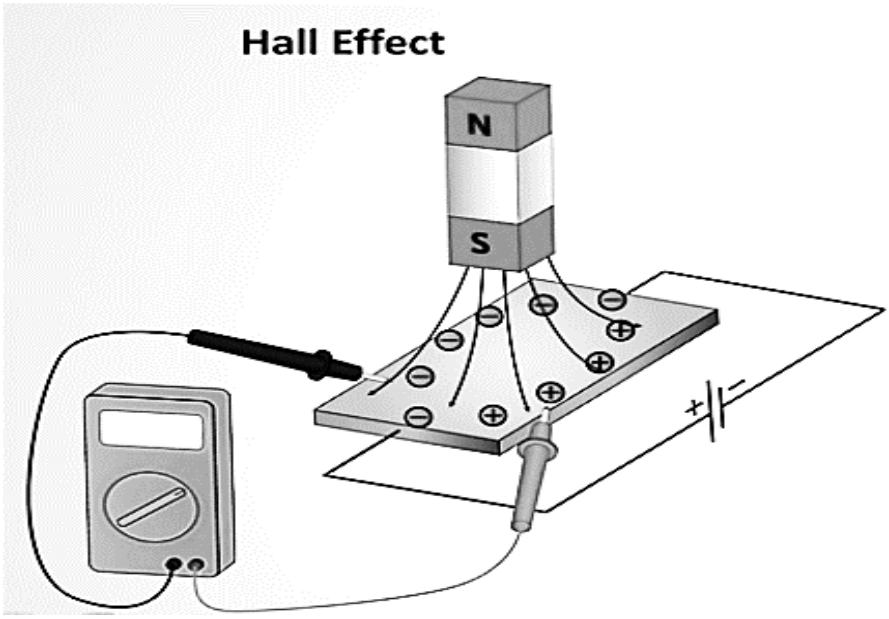


Figura 24. Funcionamento de um Magnetômetro.

Fonte: Site “How to Mechatronics” em [30].

Durante seu funcionamento, esses sensores conseguem fazer uma leitura do campo magnético ao seu redor nos três eixos cardiais (**X**, **Y** e **Z**) entregando, geralmente, a informação de saída que será interpretada em Microteslas (μT) pelo seu controlador para cada um destes eixos.

A principal informação obtida por estes sensores é a determinação da direção do norte magnético da Terra, que pode ser calculada por uma simples aplicação de fórmulas trigonométricas, como pode ser visto na Figura 25.

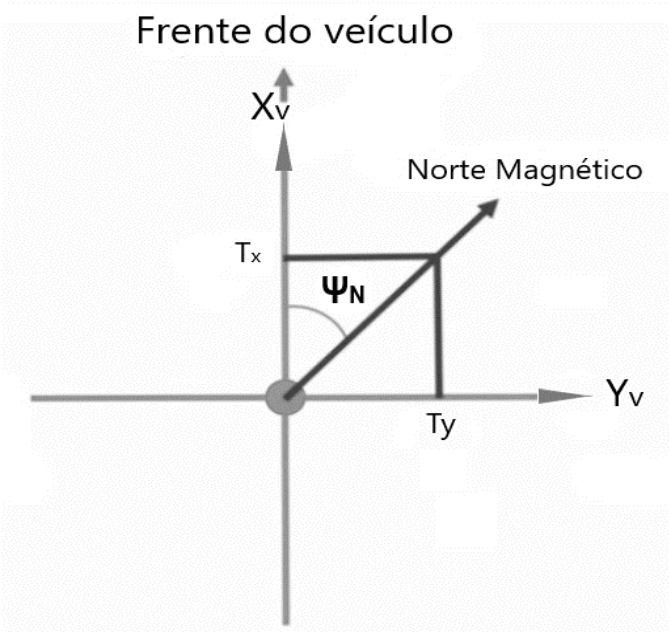


Figura 25. Cálculo do Norte Magnético.

Fonte: Elaborado pelo autor.

O ângulo da direção (em graus) do norte magnético com relação ao eixo móvel $\boldsymbol{X_v}$ (frente do veículo) é representado por $\boldsymbol{\Psi_N}$**.** As medidas de densidade do fluxo magnético (em Microteslas) medidas no eixo móvel $\boldsymbol{X_v}$ e $\boldsymbol{Y_v}$ são respectivamente representadas pelas letras $\boldsymbol{T_x}$ e $\boldsymbol{T_y}$**,** dadas na Eq. (24).

$$\Psi_N = \tan^{-1}\left(\frac{T_y}{T_x}\right) \tag{24}$$

Além disso, ao invés de calcular o norte magnético pela leitura "crua" dos sensores, como mostrado no artigo [29] da Honeywell sobre aplicações de sensores magnético-resistivos, é possível realizar uma compensação na leitura dos ângulos para o caso de o sensor sofrer inclinações nos outros eixos de rotação Assim, podemos calcular os componentes do $\boldsymbol{X}$ e $\boldsymbol{Y}$ do vetor Norte Magnético pelas Eq. (25) e (26):

$$T_{Y\_Compensado} = T_X\, cos(\theta) + T_Y\, sen(\varphi)\, sen(\theta) - T_Z\, cos(\varphi)\, sen(\theta) \tag{25}$$

$$T_{Y\_Compensado} = T_Y\, cos(\varphi) + T_Z\, sen(\varphi) \tag{26}$$

Onde $\mathbf{T_X}$, $\mathbf{T_Y}$ e $\mathbf{T_Z}$ são as componentes do vetor Norte Magnético em **X**, **Y** e **Z** respectivamente formadas pelas leituras "cruas" nestes mesmos eixos e os ângulos **φ** e **θ** são calculados pelo mesmo processo da Eq. (21). Os componentes podem ser verificados na Figura 26.

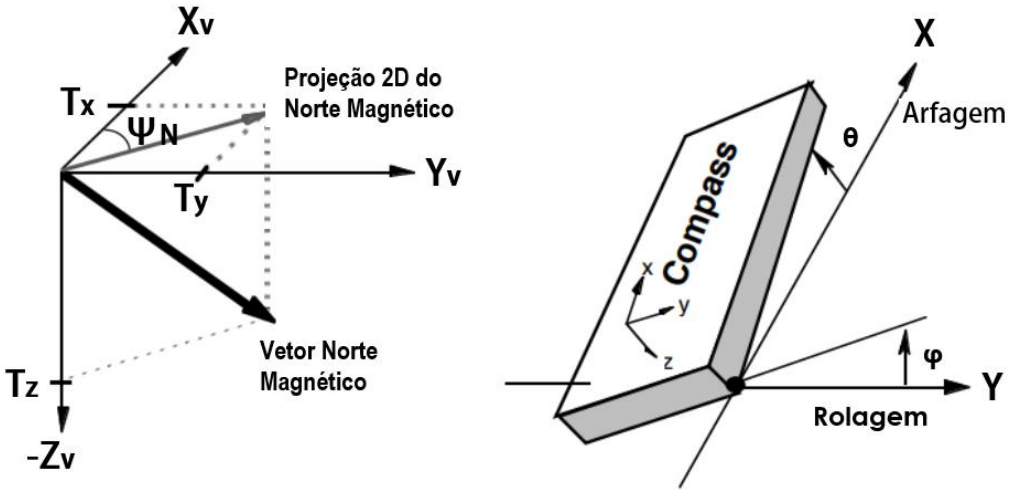


Figura 26. Componentes de leitura do magnetômetro para realizar compensação de inclinação.

Fonte: Elaborado pelo autor. Adaptado do artigo de Michael J. Caruso [29].

Além de poder indicar o norte magnético, uma aplicação que este sensor possui, que é extremamente útil e facilmente programável, é a de salvar a orientação dos 3 eixos inerciais de referência e estabelecer um eixo inicial como zero, de forma a se calcular a orientação do sistemas de coordenadas móvel do veículo com o inercial durante sua movimentação. Mais detalhes sobre isso serão discutidos na seção de "Projeto e Instalação da Plataforma de Controle".

SENSORES ULTRASSÔNICOS

Os sensores ultrassônicos, também descritos no artigo de de Ardrsh S. [31], medem distâncias ao enviar ondas sonoras de frequências mais altas que as audíveis por seres humanos. O transdutor do sensor age como uma espécie de microfone, para receber os sons, assim como um alto falante para emitir as ondas ultrassônicas. Assim que essas ondas são emitidas, elas refletem em algum objeto e voltam para o transdutor que capta a resposta de chegada e, de acordo com o tempo de demora, pode determinar a distância do som emitido. Um exemplo deste funcionamento pode ser visto na Figura 27.

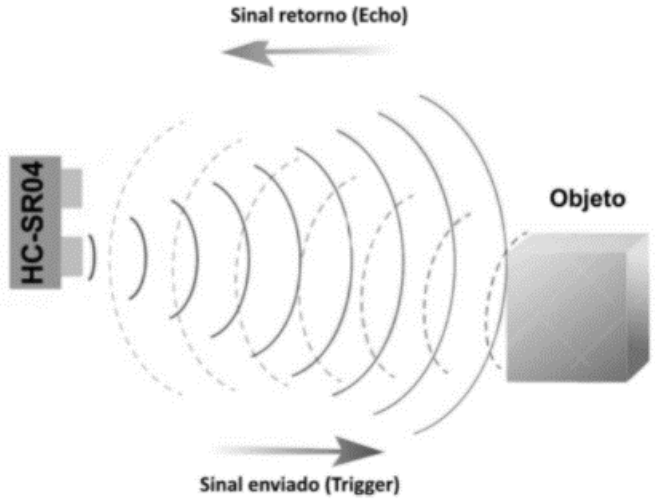


Figura 27. Funcionamento de um sensor ultrassônico.

Fonte: Retirado do site "Filipeflop" em [33].

Para calcular a distância, pode-se utilizar a fórmula apresentada na Eq. (27):

$$d_{medida\_ult} = \frac{1}{2} \times (337 \times t_{ult}) \quad (27)$$

Onde $\boldsymbol{d_{medida\_Ult}}$ é a distância medida em metros pelo sensor ultrassônico considerando a velocidade de propagação do som no ar de 337m/s à pressão de 1 atm e temperatura ambiente de 25°C multiplicada por $\mathbf{t_{ult}}$, que é o tempo de medição que demorou para a onda ser emitida e voltar ao receptor.

Esses sensores oferecem vantagens ao detectar objetos transparentes, nível de líquido ou superfícies altamente refletivas ou metálicas. Os sensores ultrassônicos também funcionam bem em ambientes úmidos já que um feixe óptico pode refrear as gotículas de água. No entanto, sensores ultrassônicos são suscetíveis a flutuações de temperatura ou vento. Além disso, são utilizados amplamente no setor automotivo para auxiliar no estacionamento do veículo, informando a proximidade e obstáculos.

2.5 BREVE INTRODUÇÃO À UM SISTEMA PID

O controle PID (Proportional-Integral-Derivative) é um mecanismo de controle com retroalimentação no qual se calcula continuamente o valor de erro correspondente à diferença entre um valor desejado, *Setpoint* (**SP**), e o valor real, *Process Variable* (**PV**), medido por algum sensor que faz parte do sistema. Dessa forma, fecha-se a malha de controle e, de acordo com esse erro, é empregada uma modulação contínua de correção baseada nos termos **Proporcional**, **Integral** e **Derivativo**. Na Figura 28 é possível ver a comparação entre os diagramas de funcionamento de um sistema de malha fechada e de um sistema de malha aberta.

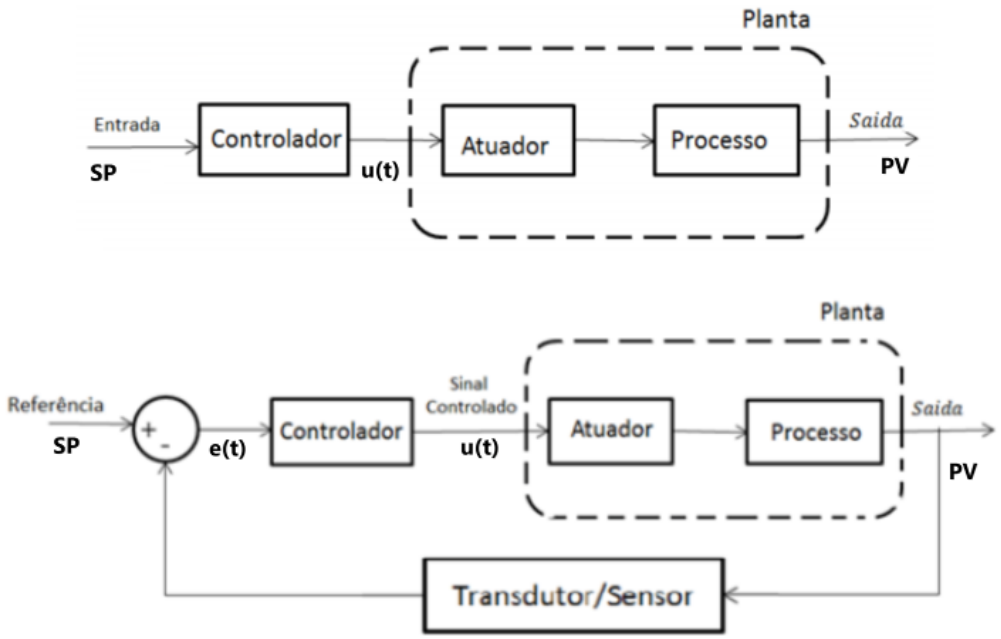


Figura 28. Comparação entre um controle de malha aberta (sitema à cima) e um exemplo de sistema de controle de malha fechada (à baixo).

Fonte: adaptado pelo autor de LUCENA, L. em [44]

Em termos práticos, esse processo consegue aplicar automaticamente uma correção de forma responsiva e acurada em uma função de controle. Por ter essas características é um dos sistemas de controle mais utilizados no mundo. Começou a ser implementado no controle automático direcional de navios em 1920 [43] e hoje pode ser encontrado, por exemplo em controles de temperatura de ar condicionado, no piloto automático de carros que acelera ou desacelera para manter uma velocidade desejada, entre outros.

Em termos matemáticos, podemos definir a equação de controle PID como uma equação diferencial-integral que tem como entrada o sinal de erro e(t) e o sinal de saída u(t) na Eq. 28:

$$\mathrm{u(t)} = \mathbf{K_P}\, e(t) + \boldsymbol{K_i} \int e(t)dt + \boldsymbol{K_d} \frac{de(t)}{dt} \tag{28}$$

Para facilitar o entendimento da fórmula, todos os parâmetros dessas equações estão mostrados na Tabela 4.

Tabela 4. Componentes utilizados na Eq. 26.

Símbolos:	**Descrição:**
$\mathbf{K_P}$	Ganho Proporcional
$\boldsymbol{K_i}$	Ganho Integral
$\boldsymbol{K_d}$	Ganho Derivativo
$\boldsymbol{t}$	Tempo
e(t)	Erro
$\boldsymbol{u(t)}$	Sinal de saída

Dessa forma, a Eq. 26 pode ser dividida em 3 partes que são somadas com pesos diferentes (que são calibrados de acordo com a resposta do projeto o sistema está sendo aplicado) para entregar a resposta que seja a mais rápida possível, mais suave e com menos erros.

Como explicado por Bertoncello S. D. em [43], para entender as influências de cada um desses termos, podemos citar como exemplo simplificado de um sistema de piloto automático de um automóvel responsável por manter a velocidade de um veículo a 30km/h que inicialmente estava a 0km/h.

O termo proporcional: é responsável por perceber o quanto o valor real ***PV*** está distante do seu valor destino ***SP*** e entregar uma resposta rápida acelerando-a para compensar o erro, porém, quando se lida com sistemas reais, é necessário levar em conta a inércia do sistema, o que causa o termo proporcional entregar uma sobre resposta e passar do valor ***SP*** e que este termo não garante tanta precisão. Desta forma, se fosse considerado um sistema PID com a predominância de um termo proporcional iríamos ver o veículo acelerando velozmente, passando dos 30km/h, retornando à uma velocidade mais baixa, acelerando novamente e mantendo esta oscilação até estabilizar próximo do valor objetivo. É interessante reparar que o sistema não atinge de fato o valor objetivo, fica apenas próximo. Tudo isso pode ser observado na Figura 29, onde se simulou um sistema PID com o ganho proporcional predominante.

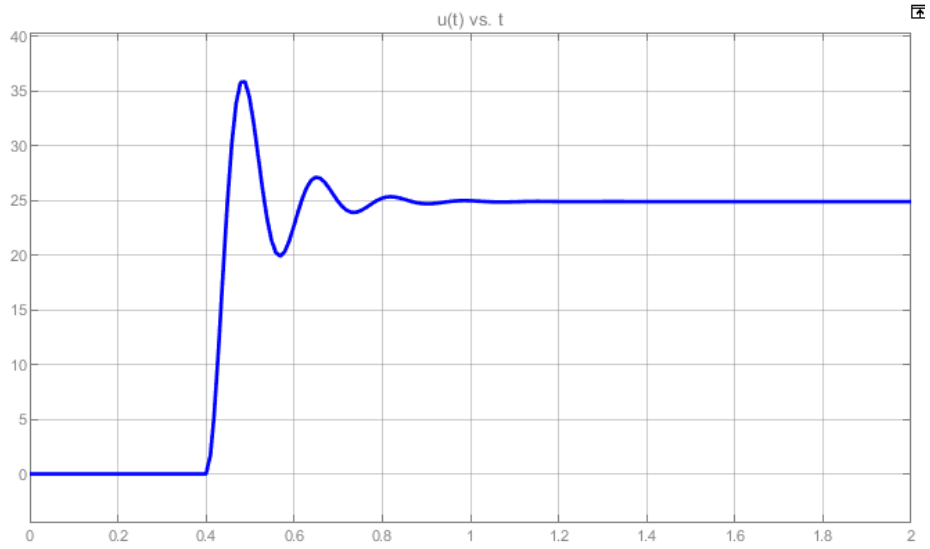


Figura 29. Resposta de sistema de controle PID com o ganho proporcional predominante.

Fonte: Elaborado pelo autor.

O termo Integral: como pode ser entendido no livro de Norman Nise [46] é responsável por ajudar a aplicar uma vertente adicional, mais forte ao longo do tempo, em casos de que o valor ***PV*** esteja próximo do ***SP***, mas não exatamente coincidentes, como se fosse uma calibração mais fina para se chegar o mais próximo possível do valor objetivo. Ao se acrescentar um ganho Integrador no sistema do exemplo, pode-se notar que o sistema demora um pouco mais para se estabilizar, aumentando as suas oscilações, porém ele estabiliza coincidente ao seu valor **SP**. Isto pode ser observado na Figura 30.

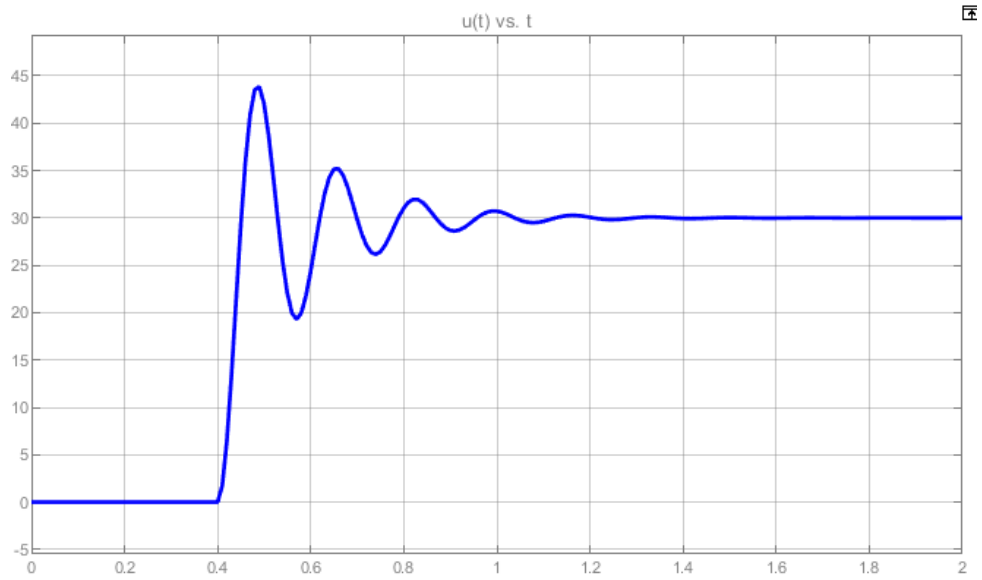


Figura 30. Sistema de controle PID com os ganhos Proprcional e Integrador equilibrados, porém sem a predominancia do ganho Derivativo.

Fonte: Elaborado pelo autor.

O termo Derivativo: é responsável por verificar a o quão rápido a resposta está se aproximando do valor objetivo, diminuindo a inclinação da resposta quando se chega próximo ao objetivo para que se alcance de forma suave. Desta forma, se considerarmos um sistema PID do exemplo anterior com um equilíbrio entre os três termos, teríamos uma resposta mais suave na qual se alcança o objetivo de forma mais rápida e com menos oscilações, como pode ser observado na Figura 31.

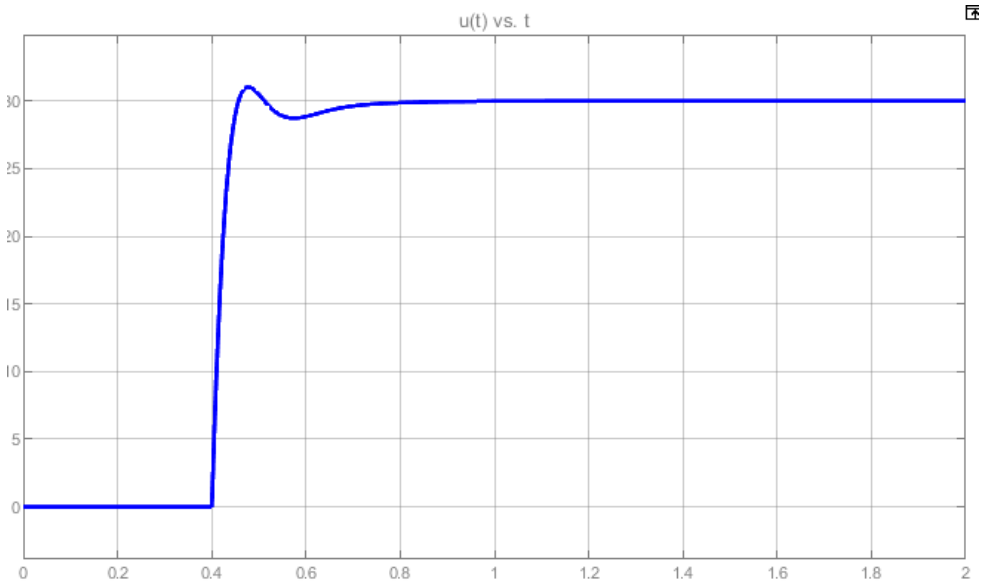


Figura 31. Sistema de controle PID com os 3 ganhos aproximadamente equilibrados.

Fonte: Elaborado pelo autor.

Apesar do termo derivativo ser muito importante, se deixado com um ganho muito alto, pode ampliar respostas desnecessárias do sistema a pequenas variações de leitura do input. Além disso, é interessante notar que em diversos sistemas que não necessitam de tanta acurácia ou velocidade, o termo integral é deixado de lado, sendo utilizados apenas um sistema "**PD**" ou "**PI**", entre outros como pode ser visto no livro de Ogata [45].

2.6 MÉTODOS DE PROTOTIPAGEM RÁPIDA

A prototipagem rápida, parafraseando o trabalho de Yan & Gu em [36], pode ser entendida como um grupo de técnicas que são utilizadas para fabricar rapidamente um modelo em escala de uma parte ou grupo de sistemas de um projeto final utilizando, geralmente, programas CAD e outras tecnologias de fabricação.

Dentre as principais tecnologias de prototipagem rápida, pode-se citar a impressão 3D por métodos aditivos como FDM, SLS, SLA, assim como métodos de subtração, como máquinas de corte e fresa por CNC.

Os principais ideais de prototipagem rápida consistem em tentar resolver vários problemas de um projeto atacando-os de forma isolada de modo que se possa reduzir e simplificar as dimensões e variáveis que esse problema possa ter. Após testes isolados, é necessário fazer testes juntando as soluções isoladas na montagem conjunta do projeto final. Isto economiza custos de testes e viabiliza um alcance mais rápido à versão final de um produto.

Além de ser utilizado com ferramentas de design e fabricação de estruturas, é muito comum aplicar os métodos de prototipagem rápida em produtos que contenham partes eletrônicas, nos quais se

faz a aplicação de placas eletrônicas reprogramáveis, que são extremamente versáteis para se testar a conexão com diferentes módulos, sensores e outros componentes, assim como se testar a aplicação de diferentes códigos em cima das opções construídas.

Neste projeto, todas as etapas tiveram grande influência destes métodos de desenvolvimento de soluções.

3. PROJETO E INSTALAÇÃO DA PLATAFORMA DE CONTROLE

Os detalhes construtivos, testes, assim como os resultados obtidos durante a instalação da plataforma estão presentes neste capítulo.

3.1 ASPECTOS GERAIS DO PROJETO

O projeto consiste basicamente em adaptar o automodelo “Traxxas Model 82056-4” inserindo um sistema embarcado modular e programável/flexível de tal forma que se possa executar comandos em todos os seus subsistemas sendo capaz de levar o automodelo a seguir uma trajetória previamente programada de forma autônoma. Isto é, sem que haja manipulação em seus direcionamentos comandados por seres humanos durante a execução deste percurso e com correção de efeitos de interferências externas e internas. A foto do automodelo original com sua bolha (revestimento externo de plástico que cobre o chassi) pode ser verificada na Figura 32.

Figura 32. Foto do automodelo Traxxas Model 84056-4 original com a bolha encaixada.

Fonte: Site da Traxxas, página de informações do modelo 82056-4 [23].

Este automodelo foi escolhido pois é capaz de continuar operante por aproximadamente duas horas com uma única carga de bateria, possui um chassi rígido, com sistemas de suspensão e *powertrain* inspirados em veículos reais construído em uma escala 1:10. Pode chegar a velocidades um pouco abaixo

de 20km/h e possui um ângulo de ataque de 56°, suficiente para ultrapassar a maior parte de obstáculos que um automodelo de testes pode encontrar.

Além disto, possui também a possibilidade de acionar ou travar um sistema de diferenciais presentes no eixo traseiro e outro no dianteiro, assim como um sistema de câmbio com duas velocidades. Tais configurações podem ser estudadas futuramente e programadas para auxiliar em diferentes situações. A foto do automodelo adaptado está na Figura 33.

Figura 33. Foto do automodelo adaptado.

Fonte: Elaborado pelo autor.

Para instalar a plataforma de controle no automodelo, foi necessário fazer algumas modificações nos circuitos e componentes eletrônicos, como a substituição do controle e receptor de radiofrequência e inserir um microcontrolador com conectividade remota para que se pudesse enviar sinais com valores precisos com uma duração de tempo controlada. Além de adicionar uma bateria exclusiva para alimentação do microcontrolador.

O projeto do modelo original do automodelo tem as suas funções e hierarquia de subsistemas de acordo com o diagrama apresentado na Figura 34.

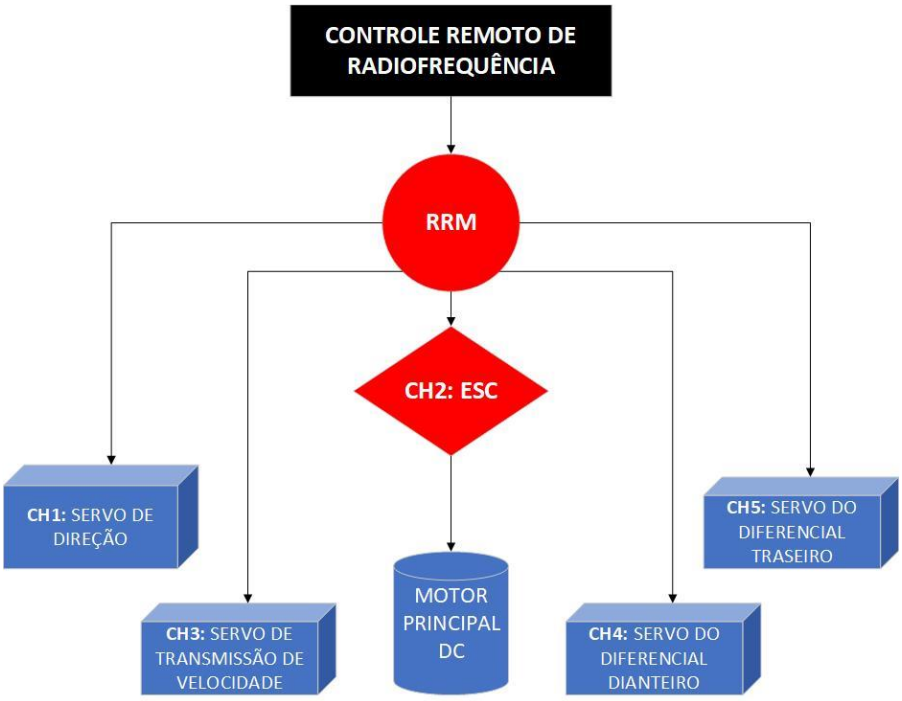


Figura 34. Diagrama de subsistemas de funcionamento automodelo original Traxxas Model 82056-4.

Fonte: Elaborado pelo autor.

Originalmente, para executar qualquer tipo de movimento precisa-se acionar o controle de radiofrequência, assim, o sinal é enviado para o receptor de radiofrequência e microcontrolador (RRM). Depois disso o sinal é processado e enviado em forma de PWM para um de seus 5 canais de saída. No caso dos canais 1, 2, 4 e 5, a frequência PWM é diretamente recebida por um servomotor que irá assumir a posição de acordo com o que foi comandado. No caso do canal 3, o sinal PWM é recebido por um controle eletrônico de velocidade (ESC) que irá processar esta frequência e enviar a corrente e voltagem proporcional para o motor principal DC.

Assim, para a construção da plataforma de controle do automodelo, foi-se excluído o controle remoto de radiofrequência e substituído o subsistema RRM por uma placa microcontroladora reprogramável. O novo diagrama de hierarquia de subsistemas está indicado na Figura 35, desta forma a placa controla diretamente os subsistemas por frequência PWM.

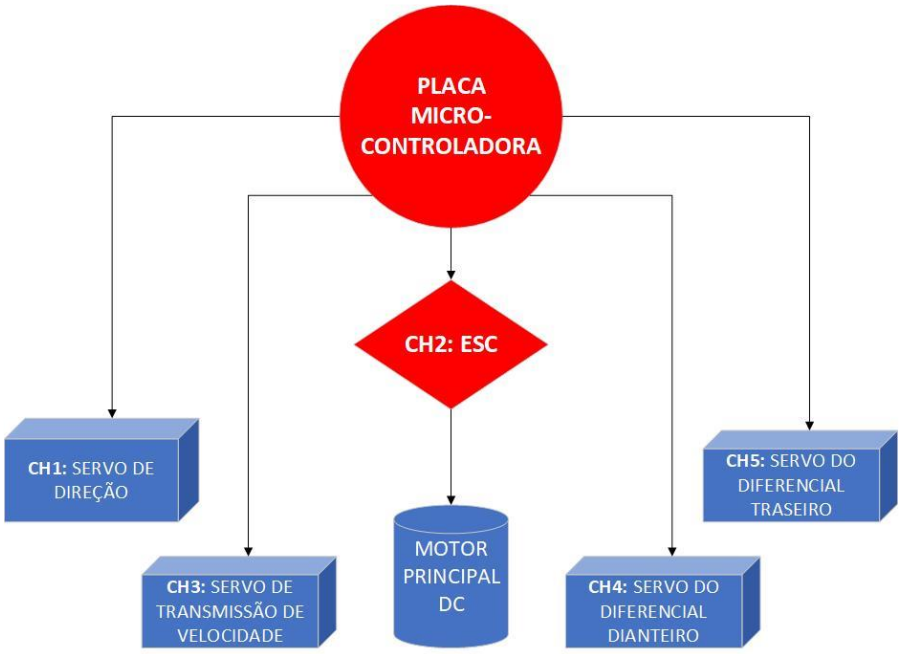


Figura 35. Diagrama de subsistemas funcionamento automodelo adaptado com microcontrolador reprogramável.

Fonte: Elaborado pelo autor.

Para selecionar o modelo de placa que faz mais sentido para o projeto, foi montada uma tabela com 5 modelos conhecidos de placas de prototipagem. Estas informações estão disponíveis na Tabela 5.

Tabela 5. Comparação de especificações técnicas entre placas controladoras.

	núcleos	**Arquitetura (Bits)**	**Frequência MHz**	**Memória RAM (KBytes)**	**Memória Flash (KBytes)**	**Entradas de GPIOs**	**Entradas analógicas**	**portas PWM**	**Conexões s/ fio**	**Preço**
arduino nano	1	8	16	2	32	14	6	6	-	55
Arduino mega	1	8	16		256	54	16	15	-	100
Node MCU ESP 8266	1	32	80	512	16000	17	8	17	Wifi	55
Node MCU ESP 32	2	32	160	160	16000	36	15	16+20	Wifi + Bluetooth	70
Rasbbery Pi 3 model B	4	ARM	1200	10^6	SD card	40	0	40	Wifi + Bluetooth	400

As duas opções de placas Arduino se mostraram um pouco inferiores em termos de desempenho quando comparadas preço a preço com os módulos Node MCU. Estes últimos apresentam conexões wireless, mais poder de processamento por um preço parecido. A placa Raspberry Pi 3 além de um microcontrolador, é um computador com entradas de tela, cabo de

internet, entre outros, é possível instalar um sistema Linux nela, porém tem um preço muito elevado e não possui entradas de leitura analógica, que é essencial para alguns sensores.

O modelo escolhido foi o "DOIT ESP32 DEVKIT V1", pois possui conectividade sem fio via WiFi e Bluetooth, podendo ter seu código reprogramado com facilidade sem a necessidade de ser desacoplada do automodelo e precisar ser conectada por um cabo serial ao computador. Isso garante agilidade para testar diferentes códigos e medir as trajetórias de forma a comparar os dados empíricos com os dados programados segundo os modelos matemáticos. Ela também tem uma construção com 2 núcleos separados e é 10 vezes mais rápida que o processador do Arduino.

Além disso, esta plataforma de desenvolvimento se mostrou extremamente versátil, possibilitando utilizar portas extras para acoplar outros sensores utilizados nas medições durante os testes. No total ela tem 30 pinos, sendo 15 conversores analógico para digital, compatibilidade com I^2C, saídas de 3.3V e 5V. Isto a tornou a escolha com melhor custo benefício para este projeto. A imagem da versão da plataforma pode ser encontrada na Figura 36.

Figura 36. Imagem da placa controladora "DOIT ESP32 DEVKIT V1"

Fonte: Site de tutoriais Eletronics Lab [26]

3.2 VERIFICAÇÃO DE VIABILIDADE DE PROJETO

O sistema do automodelo utilizado neste projeto é composto por 5 subsistemas principais:

- RRM (Receptor de radiofrequência e microcontrolador)
- Servo de direção
- Servos de funções auxiliares
- ESC XL-5 HV (controlador eletrônico de velocidade para o Motor Principal)
- Titan® 550 (Motor Principal DC)

Durante o funcionamento do automodelo, há um controle de radiofrequência que envia sinais para o receptor citado acima. Este receptor possui tanto a funcionalidade de receber tais sinais, mas também de processá-los e transmitir uma frequência PWM para todos os outros subsistemas seguintes. Tanto os servos auxiliares quanto o de direção recebem os sinais PWM traduzem isto para uma posição específica.

Já o motor principal DC (Titan® 550), necessita de um outro componente eletrônico para ser corretamente controlado, por isso, os sinais PWM são enviados primeiramente para uma Controlador de Velocidade Eletrônica onde são novamente processados, e assim a eletricidade é transmitida de forma adequada para motor principal DC. Todos os subsistemas estão representados na Figura 37.

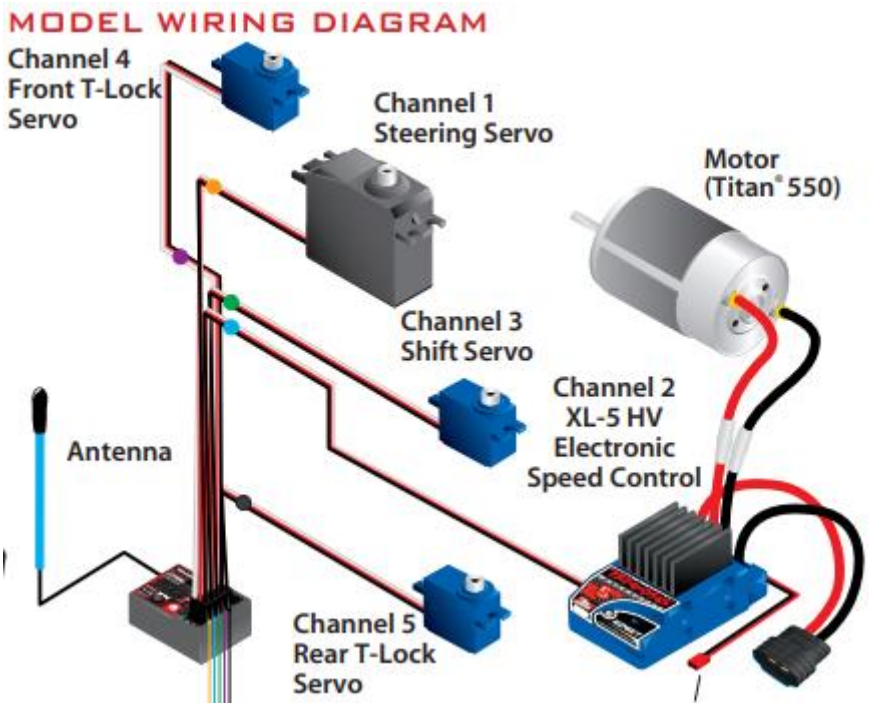


Figura 37. Diagrama de Subsistemas eletrônicas do automodelo Traxxas Model 82056-4

Fonte: Site da Traxxas, página de informações do modelo 82056-4 [23].

Para confirmar sobre a real viabilidade do projeto, foi necessário primeiramente, realizar testes para verificar se um microcontrolador reprogramável, como a ESP32, por exemplo, poderia realizar uma conexão com todos subsistemas. Para isso, foi realizado uma sequência de testes neste componente.

3.2.1 PROCEDIMENTO DE LEITURA DE SINAL DO RRM COM UM ESP32

Para ser capaz de medir a frequência PWM que o receptor de radiofrequência transmite, foi necessário mapear as saídas de cada canal e localizar seus respectivos servos, juntamente com suas funções. Esta informação foi facilmente encontrada no manual deste automodelo disponível no site da Traxxas.

Como pode ser observado na Figura 38, o RRM do automodelo em questão, após processar as informações recebidas pelo microprocessador, envia sinais PWM para 4 servomotores e 1 controlador eletrônico de velocidade que estão melhor explicados com suas respectivas funções na Tabela 6.

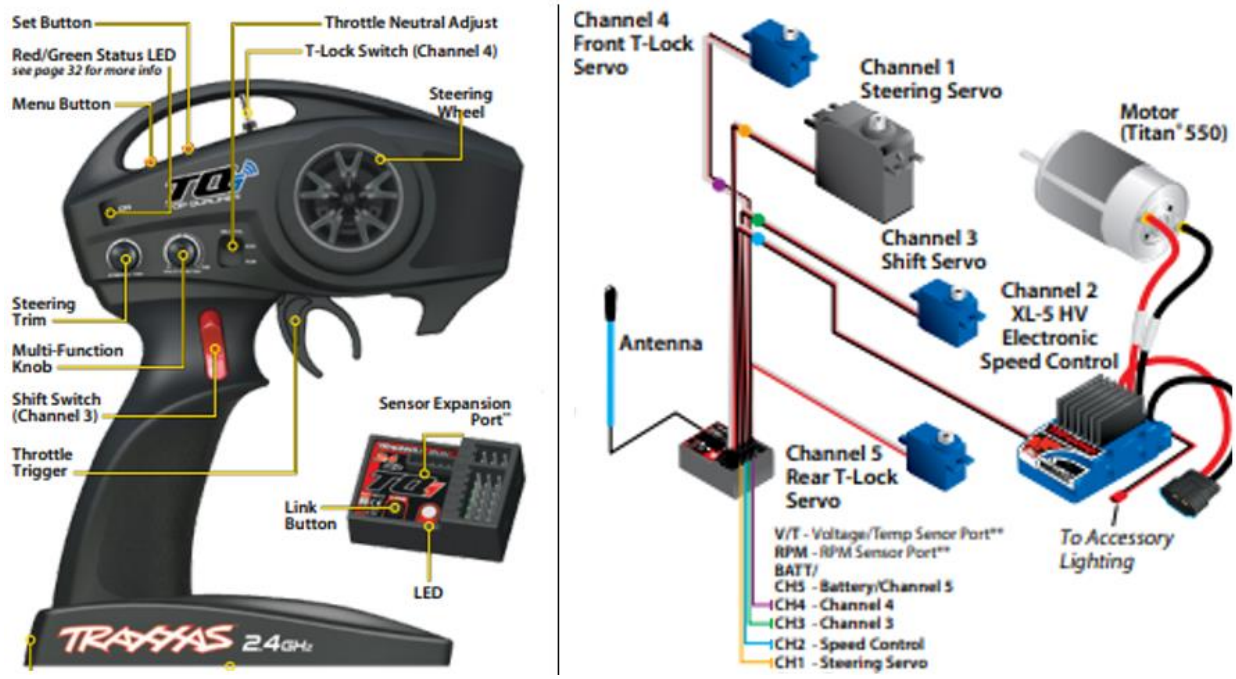


Figura 38. Imagem contendo controle de radiofrequência e o mapeamento de cada canal com seus respectivos sinais de subsistemas que o RRM envia.

Fonte: Site da Traxxas, página de informações do modelo 82056-4 [23].

Tabela 6. Canais eletrônicos existentes no automodelo Traxxas Model 82056-4 e suas respectivas funcionalidades

Canal	**Nome na Imagem**	**Nome**	**Função**
CH1	*Steering Servo*	Servo do subsistema de direção	Servomotor responsável por direcionar o sistema de direção das rodas dianteiras
CH2	*Speed Control*	Controle eletrônico de velocidade	Componente eletrônico responsável por controlar corretamente o motor principal DC
CH3	*Shift Servo*	Servo de Transmissão de velocidade	Servomotor responsável por posicionar o engrenamento em marcha trativa ou de velocidade
CH4	*Front T-Lock Servo*	Servo de controle do diferencial dianteiro	Servomotor responsável por acionar ou travar o diferencial das rodas dianteiras
CH5	*Rear T-Lock Servo*	Servo de controle do diferencial traseiro	Servomotor responsável por acionar ou travar o diferencial das rodas traseiras

Foi observado que em cada canal descrito na Tabela 6 é composto por 3 fios elétricos. Os de cores vermelha e preta são apenas responsáveis por formar uma DDP de 6V sendo eles positivo e negativo respectivamente. Já o terceiro, de cor branca, é responsável por enviar os sinais PWM que são traduzidos em uma posição angular pelo servomotor. Como os servos da Traxxas não possuiem

datasheet, provavelmente por questões de proteção de patentes e marca, foi realizada as medições das frequências de PWM que o RRM emitia aos servos para que se fosse possível replicá-las.

Para realizar a medição dos valores PWM foi necessário conectar o fio branco (responsável pelo envio de sinais) em uma das entradas analógicas da plataforma ESP32, que foi programada com o código presente no Apêndice A e criado um nó para conectar o fio negativo do automodelo com o terra da plataforma. O código foi compilado no software ARDUINO IDE e foi aberto, também, o monitor serial para conferir as medições.

O código do Apêndice A utiliza a função "*pulseIn()*" que contabiliza o tempo em microssegundos que um dado sinal está na posição "HIGH" (***sinal ativo - $T_{sinal\ ativo}$)*** ou "*LOW*" (***sinal inativo - $T_{sinal\ inativo}$***). Assim, é possível somar os dois valores para se obter o período total do sinal (T_{total} mostrado na Eq. (29) proveniente da Eq. (18) do capítulo 2) e obter o resultado do *duty cycle* (dado por $\boldsymbol{D_C}$, mostrado na Eq. 30 disposta no capítulo 2) em forma de porcentagem também.

$$T_{total} = T_{sinal\ ativo} + T_{sinal\ inativo} \tag{29}$$

$$D_C = \frac{T_{sinal\ ativo}}{T_{total}} . 100 \tag{30}$$

Durante o começo dos testes, foi notado que os sinais de PWM do controlador de rádio caíam numericamente à medida que se conectavam os outros terminais no RRM. Este fato se deve provavelmente devido a uma pequena queda da corrente fornecida pela bateria. Por isso, para realizar a medição de cada um dos canais da forma mais acurada possível, todos os fios dos componentes eletrônicos presentes no veículo de controle remoto foram conectados e foi retirado apenas o conector do canal que se desejava medir. Deste modo as condições seriam as mais próximas possíveis das que ocorrem durante o funcionamento real do automodelo.

3.2.2 RESULTADOS EXPERIMENTADOS EM CADA CANAL

Foram tomadas pelo menos 3 medidas para verificar a possibilidade de haver ruídos muito grandes. Como isto não aconteceu, o erro foi desprezado, uma vez que se mostrou irrisório com relação à precisão necessária dos valores para controlar os servos que será mostrada na sessão 3.3.

Todos os resultados obtidos se encontraram dentro das medidas 544μs e 1472μs o que já era esperado de acordo com o que foi explicado na sessão 2.2, responsável por mostrar o funcionamento dos sinais PWM para controle de servomotores. A interface do monitor serial do programa dispunha as medidas tomadas em μs (microssegundos) de cada uma das posições por servo de cada vez, como é mostrado na Figura 39.

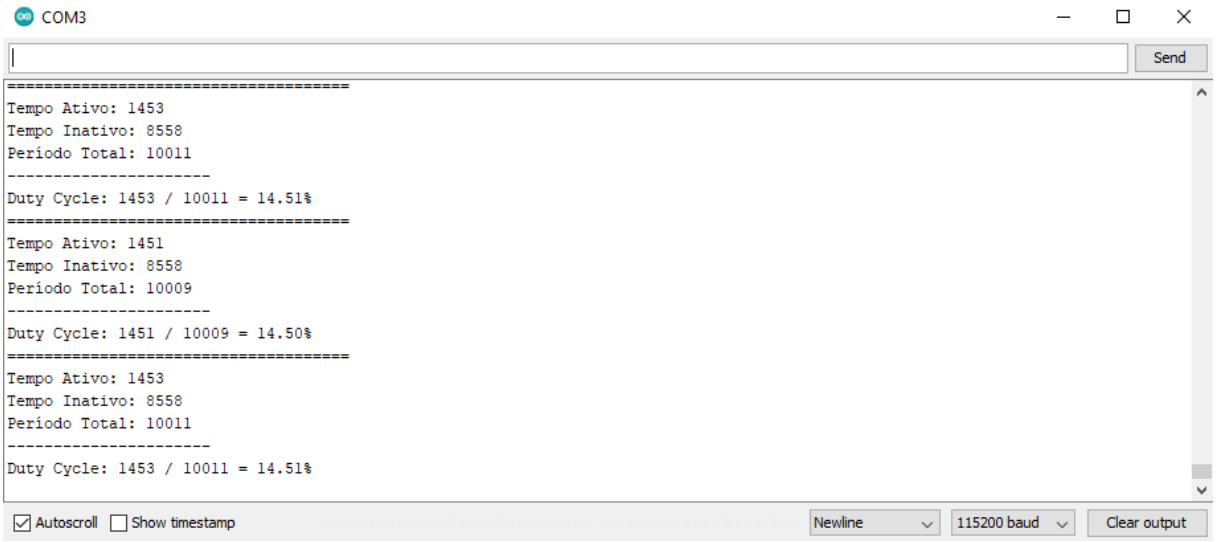


Figura 39: Valores do duty cycle em microssegundos do subsistema de direção com o sinal de direção em posição neutra (“volante” reto).

Fonte: Elaborado pelo autor.

Este processo foi repetido 12 vezes para cada uma das situações possíveis, os valores brutos obtidos foram anotados e estão dispostos no Apêndice B.

3.2.3 CONCLUSÃO DAS AFERIÇÕES

Concluindo as medições de PWM para cada componente, foi possível perceber que todos os *duty cycles* tem um período de aproximadamente 10.000µs, ou seja, 10^{-3}s e a faixa de operação do sinal ativo varia entre 10% e 20%. Foi possível observar também que o período corresponde a uma frequência de 100Hz que é uma frequência comum no controle de servos.

Como as diferenças dos resultados aferidos se demonstraram mínimas, um simples arredondamento foi suficiente para computar os dados que serão utilizados mais à frente no capítulo 3. Os resultados finais foram resumidos de uma forma mais objetiva e disponibilizados na Tabela 7 com os valores referentes a cada canal.

Tabela 7. Informações da faixa de trabalho de cada canal de saída do RRM.

Canal	Nome	Valor mínimo [μs]	Valor neutro [μs]	Valor mínimo [μs]	Posicionamentos
CH1	Servo do subsistema de direção	1001 (esquerda)	1451 (reto)	1942 (direita)	Posições contínuas dentro dos valores mínimos e máximos.
CH2	Controle eletrônico de velocidade	999 (trás)	1512 (parado)	1997 (frente)	Posições contínuas dentro dos valores mínimos e máximos.
CH3	Servo de Transmissão de velocidade	1000 (velocidade)	-	2001 (trativa)	Apenas duas posições: máxima e mínima.
CH4	Servo de controle do diferencial dianteiro	1000 (travado)	-	2001 (acionado)	Apenas duas posições: máxima e mínima.
CH5	Servo de controle do diferencial traseiro	1000 (acionado)	-	2001 (travado)	Apenas duas posições: máxima e mínima.

3.2.4 TESTES DE INTEGRAÇÃO DO SISTEMA DE ATUAÇÃO

Apesar de fazer a leitura do *duty cycle* com a plataforma ESP32 obtendo sucesso, foi observado que existiam algumas diferenças entre esta e o RRM já presente no automodelo. Dentre as diferenças, destacamos as DDPs do ESP32 (3.3V) e do RRM (6V), além de certas informações enviadas por código que poderiam impedir a possibilidade de substituir totalmente o RRM. Isso não foi um problema, pois tanto os servos quanto o subsistema ESC aceitam diversos níveis de tensão de comando, apesar de estarem sendo alimentados entre neutro e fase por uma tensão de 6V.

O código presente no Apêndice C foi rodado para verificar a viabilidade da plataforma ESP32 comandar independentemente a direção e a aceleração do automodelo. A imagem esquemática do circuito utilizado neste teste e nos posteriores está disponível na Figura 40.

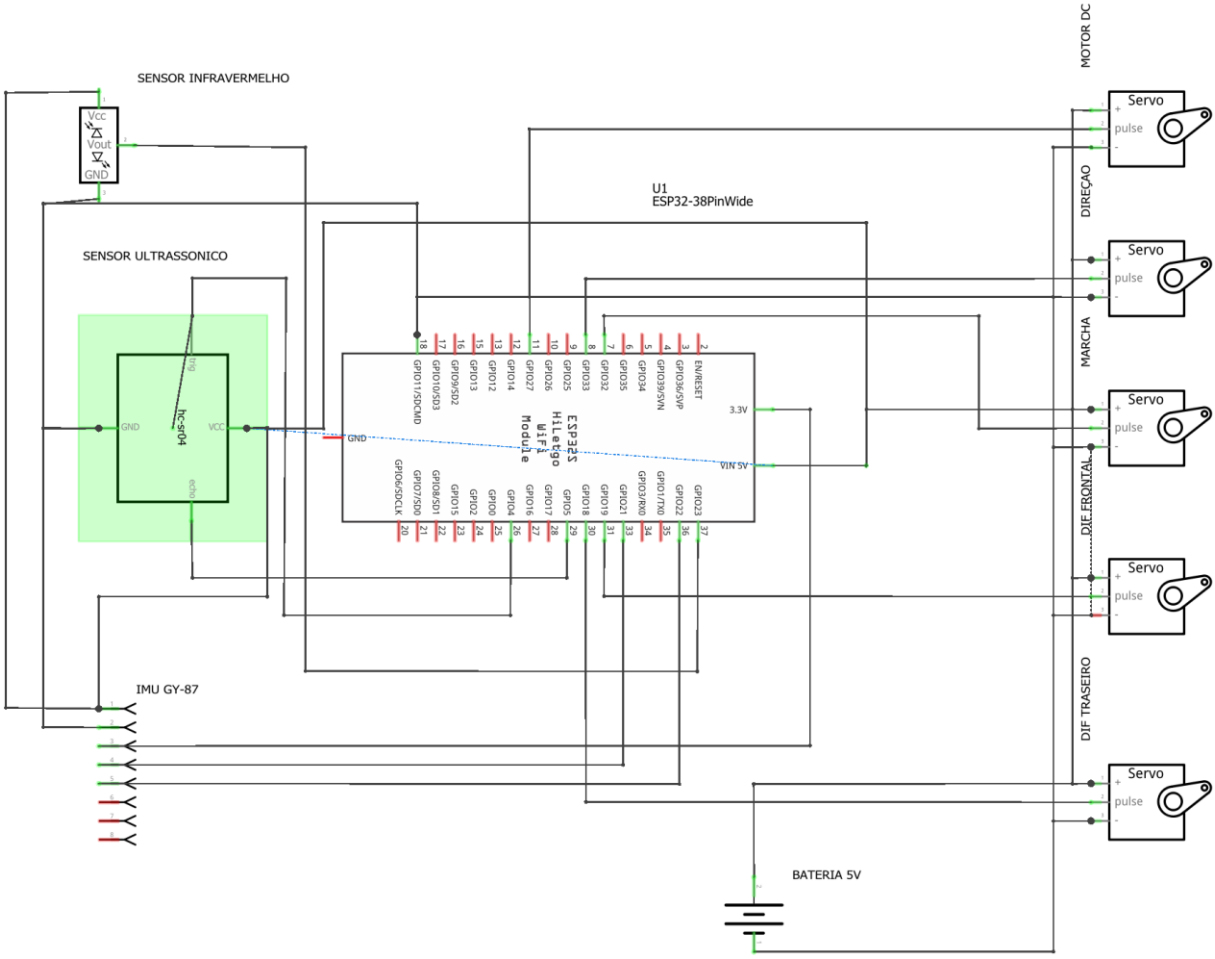


Figura 40: Montagem do circuito com cores dos cabos conectores.

Fonte: Elaborado pelo autor.

O desenho da montagem do circuito feita no próprio veículo ser encontrado também na Figura 41. É importante notar que no esquema, há um sensor que será abordado neste projeto com mais detalhes.

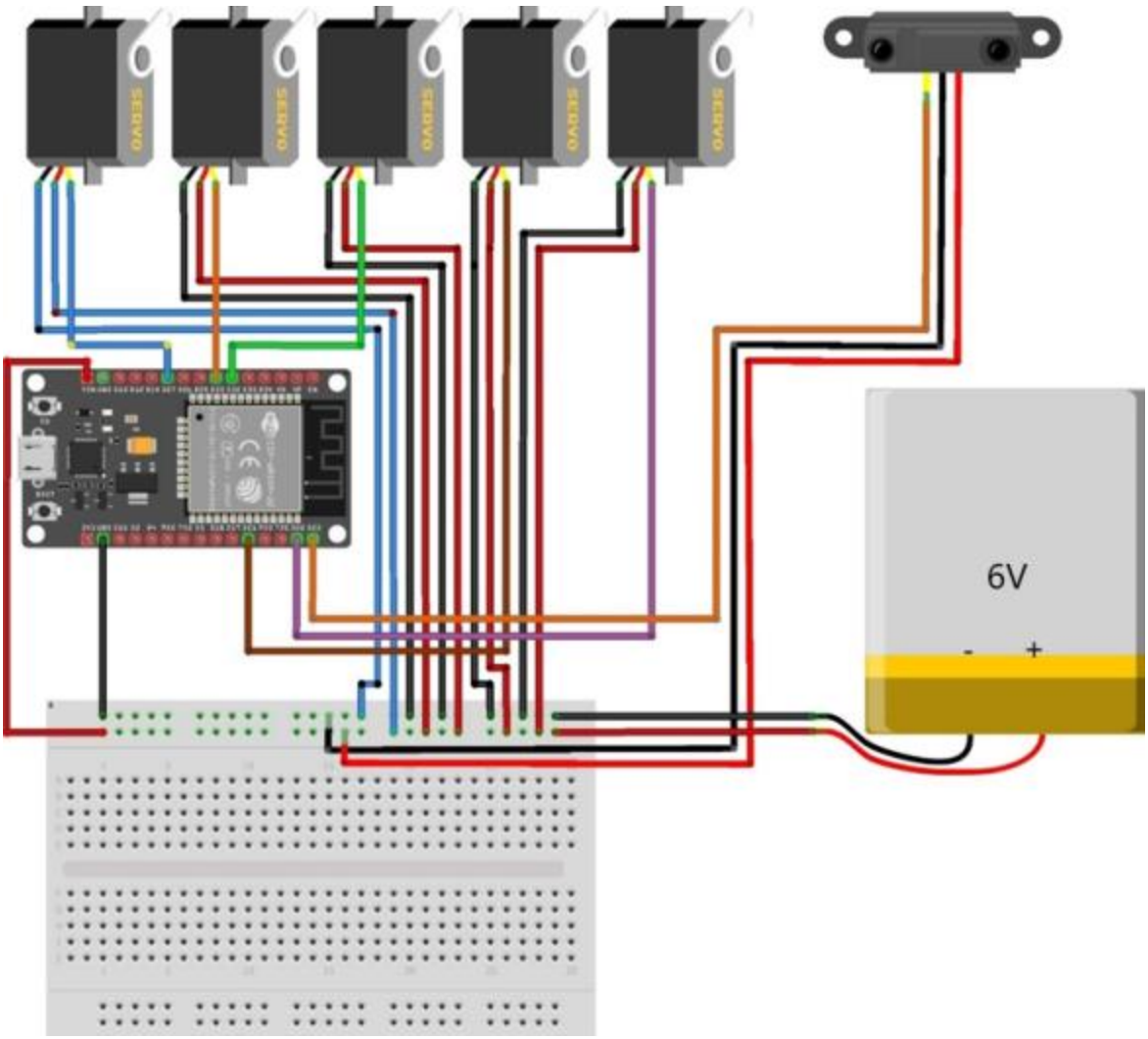


Figura 41: Desenho esquemático das conexões feitas nos testes realizados na parte 1 do projeto.

Fonte: Elaborado pelo autor.

Em conclusão, a implementação dos códigos foi bem-sucedida e os testes validaram a viabilidade do projeto.

Neste primeiro teste, as placas e sensores foram fixadas em protoboards com elásticos e fitas dupla-face de modo que não fossem feitas nenhuma modificação permanente na estrutura do automodelo. A montagem preliminar pode ser vista na Figura 42.

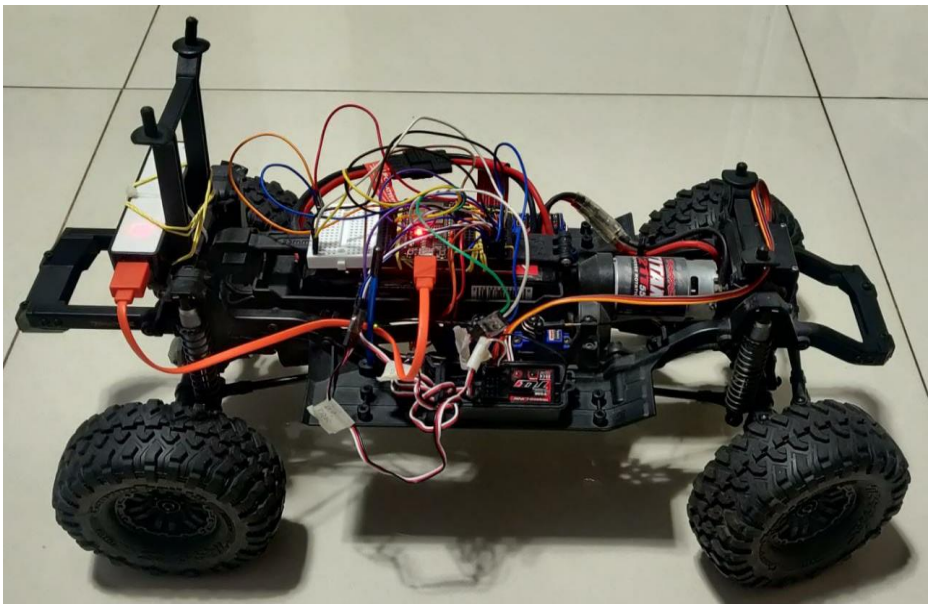


Figura 42. Montagem preliminar da plataforma e sensores.

Fonte: Elaborado pelo autor.

3.3 INSTALAÇÃO DA PLATAFORMA ELETRÔNICA DE CONTROLE

Para que seja possível realizar modificações no automodelo, instalar sensores, placas de controle, realizar testes onde se envolve movimentação e possíveis colisões, é necessário que todas as partes estejam bem fixas e organizadas no modelo, mas ainda assim de forma que seja fácil de fazer modificações posteriores.

Para isto acontecer, implementou-se uma estrutura física no automodelo que pudesse servir como um suporte principal em forma de plataforma adaptado especificamente para o modelo do veículo e, algumas adaptações foram realizadas para que fosse possível instalar as novas conexões de forma organizada e a suportar as forças causadas pelas acelerações impostas pelo movimento do automodelo.

Para realizar este subprojeto, foi retirada a “bolha” do veículo (capa de plástico externa acoplada ao chassis, que dá a forma de um automóvel real). Além disso, retirou-se algumas proteções laterais que poderiam possivelmente entrar no caminho da instalação do suporte principal, assim como os para-choques. Assim pôde ser observado uma imagem mais limpa do chassi e verificar com maior facilidade os pontos onde seria possível ancorar o suporte principal de forma estável.

Após isto tirou-se uma foto da vista superior e da vista lateral do veículo para que se pudesse ter uma imagem de referência para a realização de um design compatível com o automodelo real.

3.3.1 MODELAGEM 3D DO SUPORTE PRINCIPAL UTILIZANDO MÉTODOS DE PROTOTIPAGEM RÁPIDA

Para que fosse possível realizar um design proporcional e funcional do suporte principal utilizando métodos de prototipagem rápida foi necessário seguir os seguintes passos:

a) LEVANTAMENTO DE CONDIÇÕES DE CONTORNO E OBJETIVOS DA MODELAGEM

Realizou-se um estudo das condições de contorno e outras funcionalidades que o suporte deveria obedecer, as quais estão listadas na Tabela 8. É importante notar que muitas decisões feitas no design desta estrutura foi realizado pensando-se em comportar futuros estudos no veículo.

Tabela 8. Tabela de requisitos do suporte principal.

Nº:	**Requisitos de projeto para o suporte:**
1	O suporte deve ser preso de tal forma e ter integridade estrutural rígida o suficiente para que se possa levantar, mudar a direção e segurar todo o veículo com acelerações bruscas;
2	O suporte deve possuir algum tipo de alça para que seja fácil e rapidamente levantado, para o que o veículo possa ser puxado caso esteja em situações de possível colisão ou queda durante sua fase de testes;
3	O suporte deve ser fechado de forma a proteger os circuitos eletrônicos;
4	O suporte deve permitir uma abertura fácil para que possam ser feitas modificações e reparos rápidos no circuito;
5	O suporte deve ter espaço suficiente para acomodar futuros estudos que venham a ser realizados no veículo tenham a possibilidade de ser encaixados no mesmo, comportando placas eletrônicas de tamanhos comerciais conhecidos e outros sensores extras;
6	O suporte deve comportar sensores de distância à sua volta para que seja possível realizar um estudo posterior de proximidade de objetos em rota de colisão;
7	O suporte deve permitir que a bateria do veículo seja removida sem necessitar retirar o suporte do local;
8	O suporte deve conter um local de encaixe na sua parte frontal para que seja possível instalar um sistema de câmera para estudos futuros de leitura de imagens;
9	O suporte deve conter entradas de ventilação para que os eletrônicos não sobreaqueçam;
10	O suporte deve ter aberturas laterais, frontais e traseira para que seja possível a passagem de fios dos servos e outros eletrônicos do veículo;
11	O suporte deve conter um espaço grande o suficiente para que se instale pelo menos duas *protoboards* de tamanho médio e consiga acomodar de forma tranquila os fios, placas e outros sensores dentro do espaço principal.
12	O suporte deve ser modelado em 3D de modo que tenha uma fabricação fácil e eficiente em impressão 3D por método FMD, de modo que precise de poucos suportes poucas peças móveis (utilizando princípios de *Design for Manufactoring* e prototipagem rápida).

b) ANÁLISE DE FOTOGRAFIA DAS VISTAS DO VEÍCULO

Foi retirado fotos da vista lateral e superior do veículo de modo que se pudesse analisar o espaço disponível para a construção do suporte, quais as possibilidades de design e tamanho, melhores pontos de fixação, etc. Nas Figuras e é possível visualizar os pontos de ancoragem e as vistas citadas.

Figura 43: Vista lateral da estrutura do automodelo.

Fonte: Elaborado pelo autor.

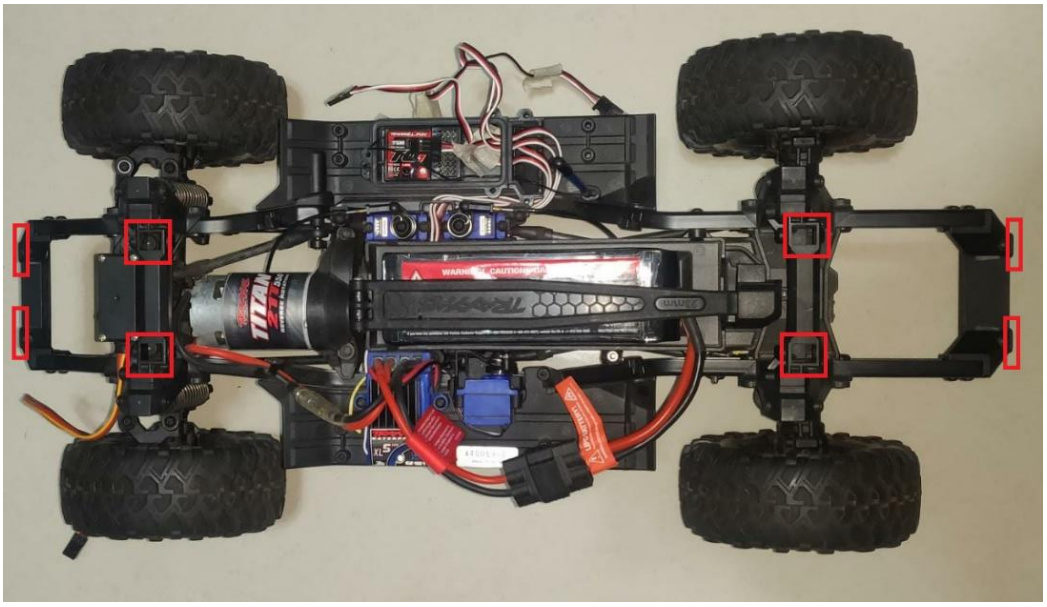


Figura 44. Vista superior com pontos de ancoragem identificados.

Fonte: Elaborado pelo autor.

Na Figura 44 foram identificados com retângulos vermelhos estruturas do chassi de alumínio de perfil em forma de tubo retangular que seriam extremamente positivos para prender os pontos de ancoragem do chassis. Suas entradas estão posicionadas na mesma altura e possuem o mesmo tamanho, o que tornou um local perfeito para de fazer a adaptação.

c) MODELAGEM 3D EM À PARTIR DAS FOTOS

No veículo físico, foi escolhida a distância entre as duas faces exteriores das entradas do tubo de alumínio de perfil retangular $\boldsymbol{d_{pa}}$ e tomadas medidas utilizando uma régua, disposta na Eq. (31).

$$d_{pa} = 300mm \tag{31}$$

A partir das fotos mostradas nas figuras do item b) foi possível inseri-las no programa *SOLIDWORKS* (programa CAD de modelagem 3D) e ajeita-las à sua devida escala de acordo com medida tomadas na Eq. (31). Assim, foi possível esboçar o que seria a base do design considerando todas as condições levantadas no item a). Esta etapa do processo pode ser visualizada na Figura 45.

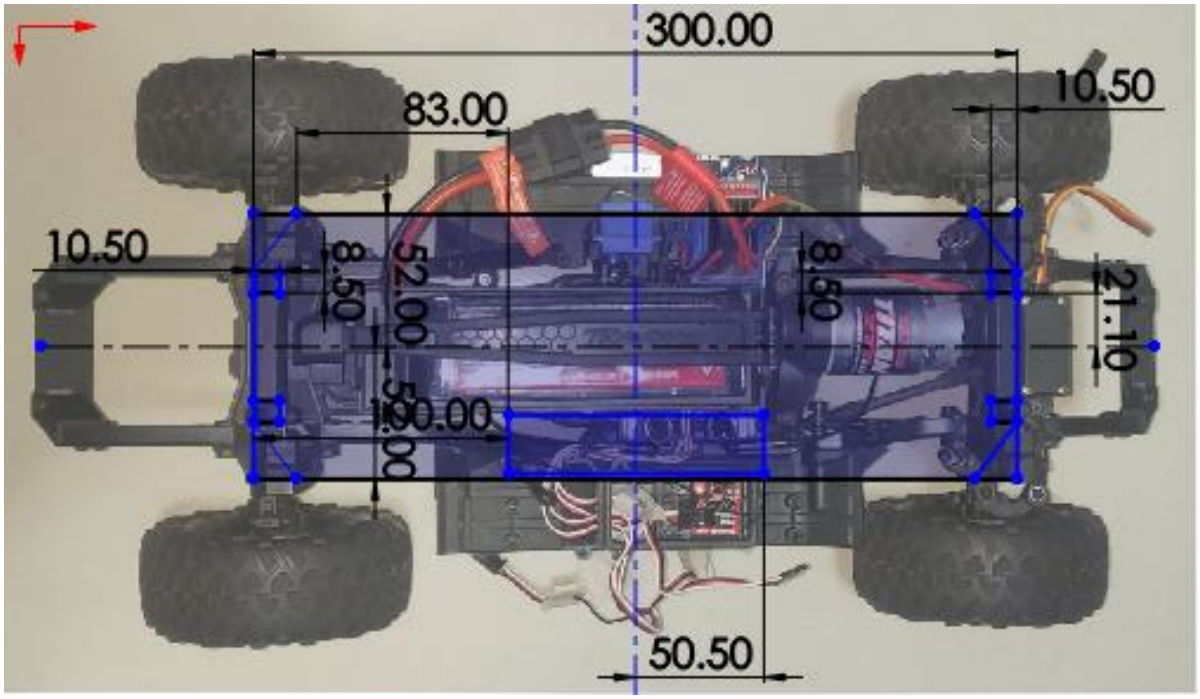


Figura 45. Esboço do suporte sobre imagem da vista superior.

Fonte: Elaborado pelo autor.

A partir da vista lateral, foi possível fazer o esboço do contorno da alça do suporte, respeitando a limitação de ter uma altura de no máximo 300mm do chão até a o ponto mais alto do suporte, mas ainda assim tendo espaço suficiente para que se fosse possível posicional uma mão confortavelmente na alça. Foi necessário garantir uma altura mínima da bateria de 18,60mm para que fosse possível fazer a sua retirada sem precisar desmontar o suporte e manter uma altura útil interna para os eletrônicos de pelo menos 40mm para que todos os cabos fossem conectados de forma organizada. O esboço pode ser verificado na Figura 46.

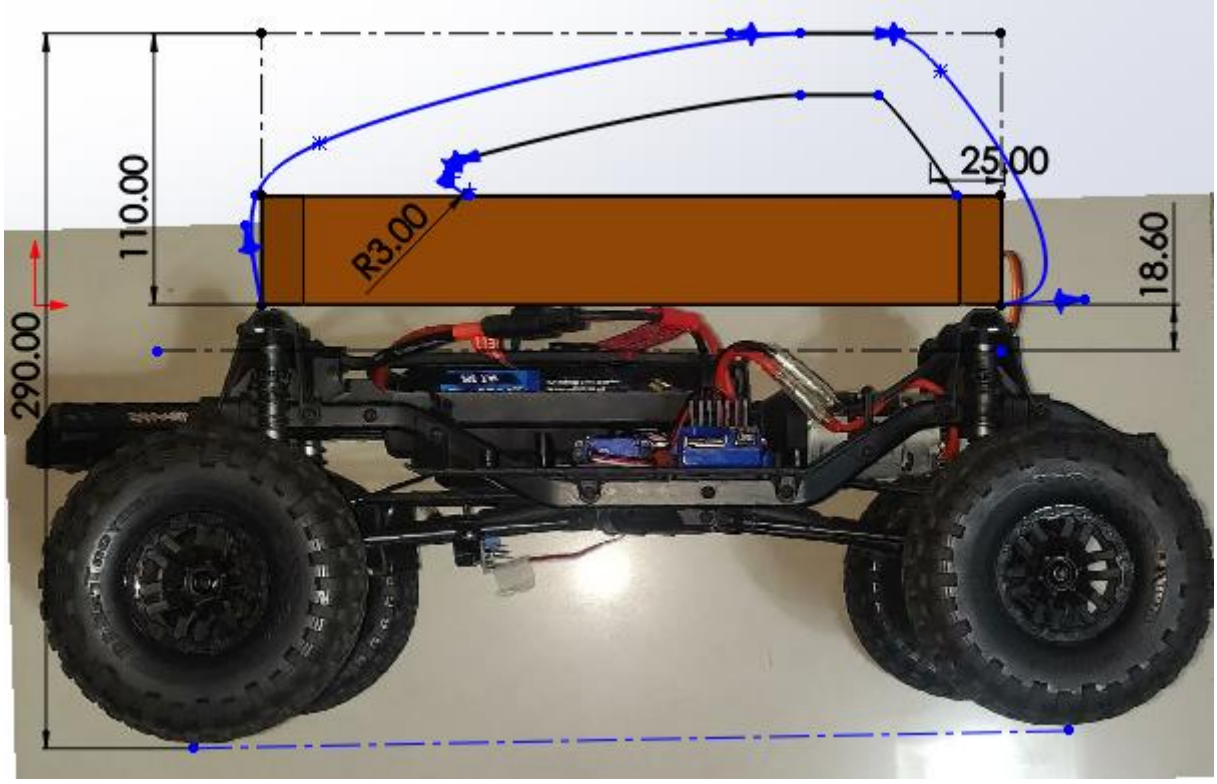


Figura 46. Esboço do suporte sobre imagem de vista lateral.

Fonte: Elaborado pelo autor.

Após os esboços base, a modelagem 3D pôde ser feita, ainda se pensando nas condições de contorno até se chegar no resultado final que pode ser visto na Figura 46.

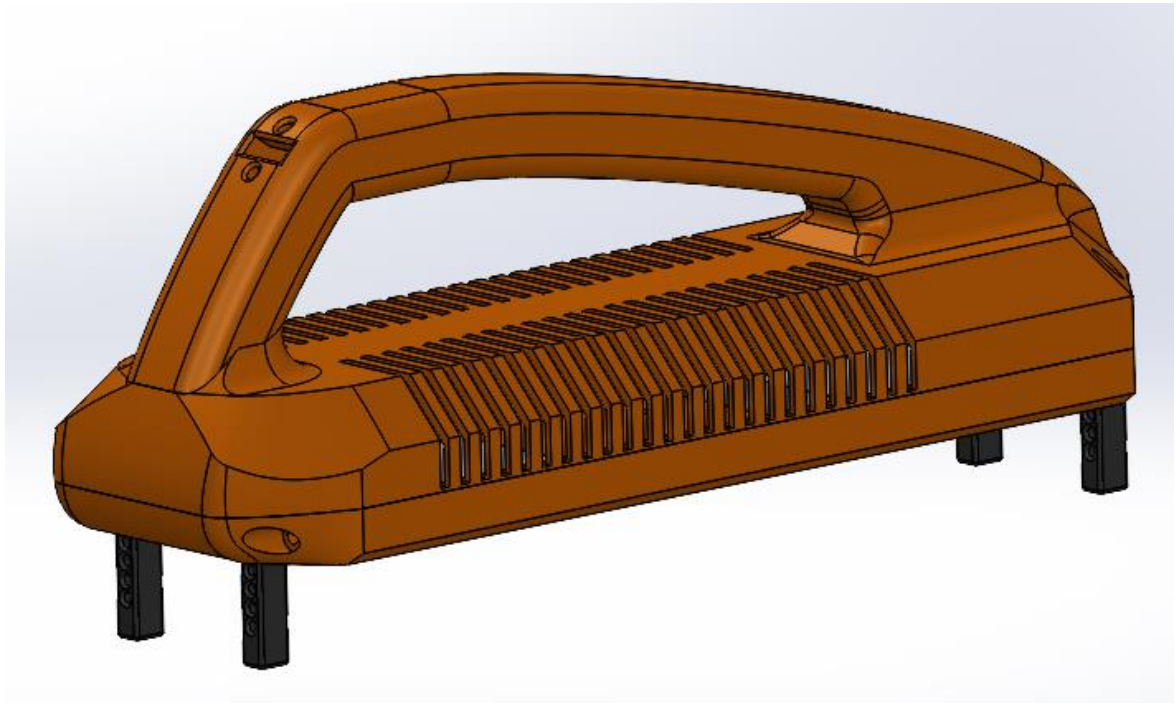


Figura 47. Imagem do suporte final modelado em 3D.

Fonte: Elaborado pelo autor.

A montagem final de suas partes foi feita de modo que a alça pudesse ser impressa separadamente e colada na tampa superior e a fixação da tampa superior com a base foi feita de forma que se pudesse utilizar parafusos e porcas M4. Para isto, utiliza-se um método de inserção de porcas na impressão 3D como pode ser visto na Figura 48.

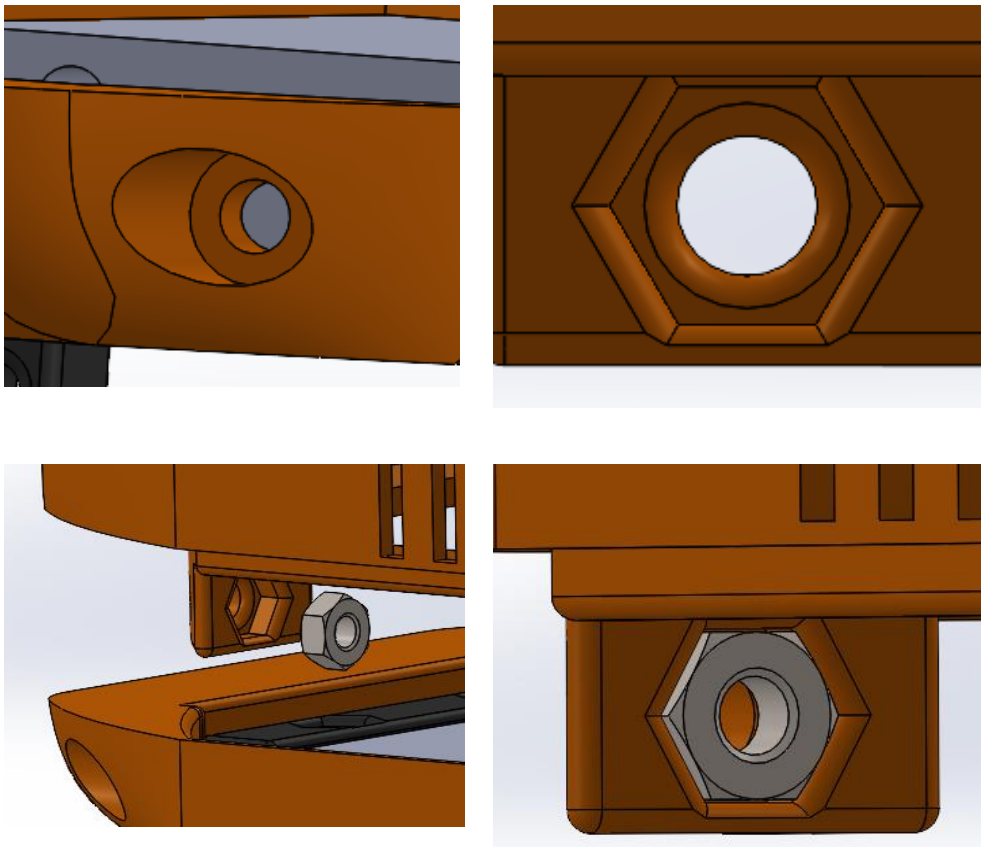


Figura 48. Conjunto de imagens mostrando o método de inserção das porcas.

Fonte: Elaborado pelo autor.

Uma imagem com a vista explodida da montagem do suporte pode ser observada na Figura 49. É interessante observar que foram adicionados 2 para-choques (dianteiro e traseiro) no conjunto para suportar os sensores ultrassônicos de distância. O método de modelagem e fabricação deles foi o mesmo do suporte principal.

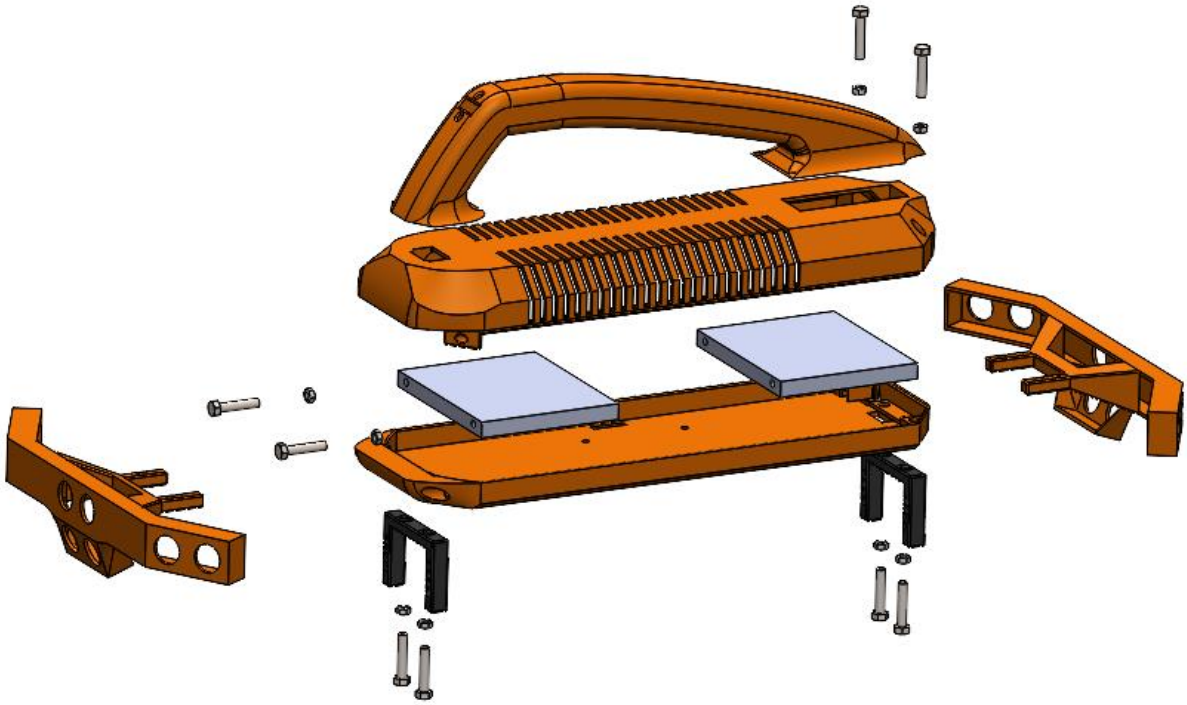


Figura 49. Vista Explodida da montagem completa do suporte principal e sues fixadores.

Fonte: Elaborado pelo autor.

d) IMPRESSÃO 3D E MONTAGEM

O suporte foi impresso em 3D utilizando filamento de PLA (poliácido láctico) que é um polímero amplamente utilizado na indústria de fabricação por métodos aditivos, acessível e capaz de aguentar os esforços sofridos pela peça.

A alça superior do suporte foi colada à tampa com uma cola à base de cianocrilato que reage muito bem com o material utilizado. Após isso foi utilizado 8 porcas M4 e 8 parafusos M4 de 30mm de comprimento para se fixar as partes removíveis do suporte. Pode-se verificar uma foto do suporte pronto e montado no automodelo na Figura 50.

Figura 50. Foto do suporte montado no automodelo.

Fonte: Elaborado pelo autor.

3.3.2 IMPLEMENTAÇÃO DE UM ODÔMETRO E VELOCÍMETRO COM UM SENSOR INFRAVERMELHO

Medir a distância percorrida e a velocidade do veículo de forma consistente, obtendo um erro menor que 1km/h foi um grande desafio. Para que se fosse possível obter esses dados, foi feito um sistema que pudesse contar os giros do eixo cardan que alimenta o eixo traseiro do veículo.

Para que se fosse escolhido o melhor sensor para isso, foi feita uma tabela comparando as principais características de quatro sensores encontrados no mercado compatíveis com a placa eletrônica reprogramável utilizada neste estudo. Os dados podem ser verificados na Tabela 9.

Tabela 9. Dados de sensores de proximidade para sistema de leitura de velocidade e odômetro.

	Alcance (cm)	Sofre interferência da luminosidade?	Sofre interferência de diferentes materiais?	Velocidade de resposta	Dimensões (mm)	Preço médio
Sensor ultrassônico hc-sr04	2 - 400	não	pouco	lento	45x20x15	R$ 10,50
Sensor ultrassônico Jsn-sr04t	20 - 600	não	pouco	lento	42x29x12	R$ 69,00
Sensor infravermelho TCRT5000	2 - 30	sim	médio	rápido	37x14x6	R$ 17,25
Sensor laser TOF10120	10 - 180	pouco	pouco	rápido	33x10,4x10.1	R$ 169,00

Ao analisar todas as características, escolheu-se então, inserir uma espécie de sistema de leitura infravermelho no sistema de transmissão do eixo utilizando o sensor infravermelho TCRT500. Este foi selecionado, uma vez que se necessitava de um sistema de leitura que pudesse captar altas velocidades, em uma distância próxima, sem a ocorrência de grandes erros na leitura que fosse menos custoso possível.

Para fazer isso, um módulo com o sensor ótico infravermelho reflexivo, modelo TCRT5000, foi posicionado em um pequeno suporte impresso em 3D que o aponta em direção ao eixo de transmissão de rotação para as rodas. O eixo por sua vez, recebeu uma peça que o envolve girando juntamente com ele, modelada especificamente para entregar a leitura com menor ruído possível, sua geometria é mostrada na Figura 51.

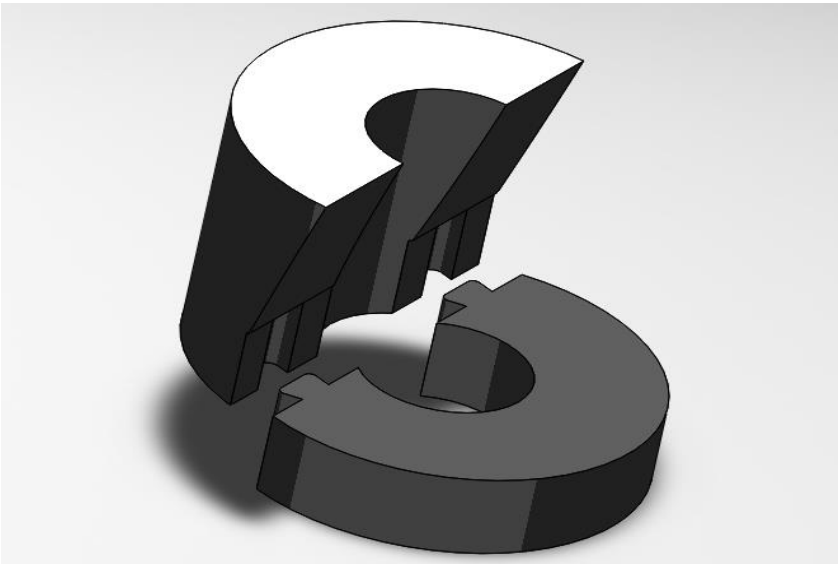


Figura 51. Peça cilíndrica do sensor de leitura de velocidade.

Fonte: Elaborado pelo autor.

É interessante que com esta peça, utilizou-se a característica do sensor sofrer diferenças de leitura com relação às cores à favor do objetivo do projeto, uma vez que a parte branca está mais próxima e reflete mais facilmente o leitor, enquanto a preta absorve a luminosidade ao mesmo tempo que está mais longe, entregando um maior contraste e consequentemente diminuindo a possibilidade de haver erros de leitura.

A Figura 52 mostra sua forma, onde é possível observar uma espécie de cilindro cortado com duas áreas planas voltadas para o sensor com alturas diferentes. A mais alta, de cor branca, reflete a luz infravermelha emitida que é captada de volta pelo sensor e contabilizado, enquanto a face negra, mais longe do sensor, absorve luz suficiente para que o sensor não seja acionado.

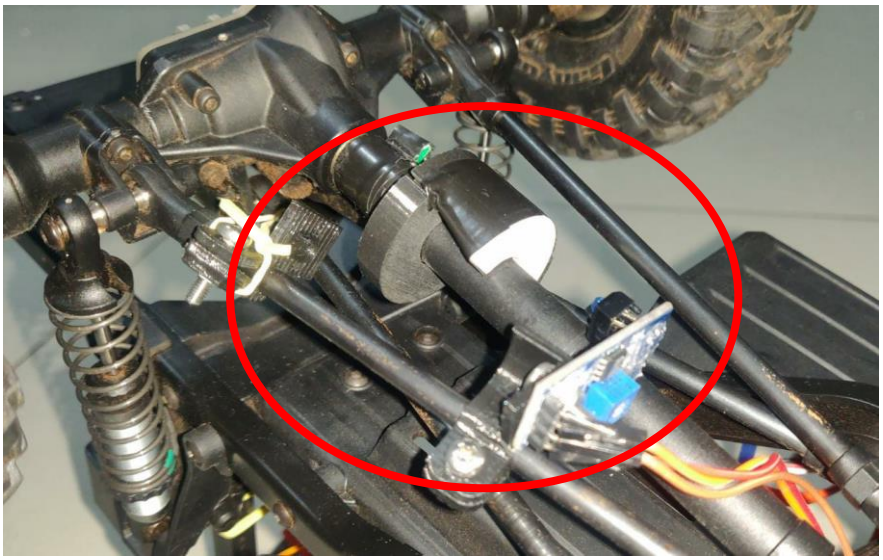


Figura 52. Sensor infravermelho com peça cilíndrica de duas cores para auxiliar a medição.

Fonte: Elaborado pelo autor.

Assim, quando o eixo gira, a luz infravermelha que é constantemente emitida reflete na área branca em direção ao sensor que é repetidamente acionado e desligado. Assim, cada variação desta, corresponde a um giro do eixo. Consegue-se assim, contabilizar a quantidade de vezes que isto ocorre durante o período de 1 segundo. Essa quantidade de vezes revela a velocidade de rotação do eixo e consequentemente a velocidade de giro das rodas do automodelo de acordo com a proporção de redução eixo-roda.

O módulo do sensor permite também, que seja regulada a permissividade do receptor de infravermelho por meio de um potenciômetro resistivo. Desse modo pode-se garantir que o mesmo terá o efeito esperado de acordo com a distância que o sensor foi posicionado do cilindro reflexivo acoplado ao eixo.

Por fim, a velocidade e distância percorrida do automodelo pôde ser obtida utilizando-se a relação de rotações do eixo para as rotações da roda, assim como a medida do raio efetivo $\boldsymbol{R_{ef}}$ da roda.

De acordo com os desenhos técnicos e tabelas apresentadas no manual disponibilizado no site da Traxxas [23], foi possível observar que o veículo possui dois estágios de redução do eixo para a roda.

O primeiro, presente no diferencial, que possui uma redução de *34:11* e o segundo, é uma redução por par de engrenagens simples que está localizada na extremidade externa do eixo da roda de *24:11*. Multiplicando os dois valores podemos chegar à relação de giro eixo-roda $\mathbf{W_{e\text{-}r}}$ disposta na Eq. (32).

$$W_{e-r} = 7{,}4181 : 1 \tag{32}$$

Com o raio efetivo $\boldsymbol{R_{ef}}$ medido na seção 3.5.4, foi possível obter então a distância percorrida de acordo com a Eq. (33). Esta equação mostra simplesmente a multiplicação do raio efetivo pelo valor de duas vezes ***pi***, para se encontrar o tamanho da circunferência e multiplicando tudo isto pelo número de de rotações dadas pelo eixo ***N***, e finalmente dividindo pelo número da relação de engrenagens, obtendo-se assim, a Distância longitudinal percorrida $\boldsymbol{D_{lon.}}$

$$D_{lon} = \frac{(2\pi.R_{ef}).N}{7{,}4181} \tag{33}$$

A velocidade instantânea é simplesmente obtida dividindo-se a distância percorrida por uma pequena fração de tempo contada pelo microcontrolador.

3.3.3 IPLEMENTAÇÃO DE UM SENSOR DE POSIÇÃO

Antes da instalação do sensor de posição, foram elencadas algumas possibilidades de sensores disponíveis no mercado. Porém durante o projeto, não houve uma opção ampla de possibilidades por conta dos recursos disponíveis e o módulo disponibilizado para fazer o projeto foi o módulo GY-87. Este, por sua vez, é equipado com os sensores:

- **MPU 6050**: é um IMU, responsável por fazer as leituras de giroscópio (velocidade angular instantânea nos 3 eixos) e de acelerômetro (aceleração linear instantânea nas 3 direções cartesianas).
- **HMC 5883L**: responsável por fazer as leituras de posição angular instantâneas nos 3 eixos a partir de um magnetômetro.
- **BMP 180**: responsável por medir a pressão relativa (barômetro) e com essa informação estimar a altitude do veículo. Este sensor não é utilizando neste estudo.

Para construir o código de cada um dos sensores, foi necessário procurar pelo datasheet deles separadamente. O módulo inteiro é conectado por 5 fios. Um de VCC (de 5V), outro fio fase de 3.3V, um *Ground* (polo negativo) e duas entradas responsáveis por fazer uma comunicação de protocolo I2C (DAS e SCL). Mais detalhes da ligação elétrica do sensor podem ser verificados na subseção 3.3.5 deste capítulo, onde pode se encontrar o diagrama do circuito elétrico.

3.3.4 INSTALAÇÃO DOS SENSORES ULTRASSÕNICOS

Para garantir que durante os percursos desenvolvidos e testes realizados o veículo não sofra colisões com objetos em sua rota, foi instalado dois sensores de proximidade. Para detectar o perigo e frear o veículo prevenindo assim a sua colisão.

Para conseguir fazer isso foi necessário selecionar um sensor que consiga detectar distâncias de pelo menos 20cm à 2 metros, que funcione com uma grande variedade de materiais e formas, que não seja afetado por diferentes tipos de luminosidade e que tenha um custo baixo.

Para que se fosse possível uma melhor decisão, foram levantados 4 tipos de sensores para medir a distância com suas principais características que tenham uma possibilidade de afetar o projeto, estas informações estão disponíveis na Tabela 10.

Tabela 10. Tabela de características de sensores para medir distância.

	Alcance (cm)	Sofre interferência da luminosidade?	Sofre interferência de diferentes materiais?	Sofre interferência de diferentes formas?	Dimensões (mm)	Preço médio
Sensor ultrassônico HC-SR04	2 - 400	não	pouco	médio	45x20x15	R$ 10,50
Sensor ultrassônico JSN-SR04T	20 - 600	não	pouco	médio	42x29x12	R$ 69,00
Sensor infravermelho TCRT5000	2 - 30	sim	médio	médio	37x14x6	R$ 17,25
Sensor laser TOF10120	10 - 180	pouco	pouco	médio	33x10,4x10.1	R$ 169,00

Analisando a tabela acima, foi possível chegar à conclusão de que os melhores sensores para o projeto seriam os de tecnologia ultrassônica. A principal vantagem das outras tecnologias é a velocidade de leitura, que não é um fator importante para este projeto, visto que os ultrassônicos mais lentos já suprem a necessidade do projeto. Além de já serem amplamente empregados para esta função no mercado automobilístico, estes não sofrem interferência de luminosidade nem do tipo de material com que o obstáculo é composto. Estes sensores também sofrem apenas pequena interferência para sensoriamento apenas de objetos com superfícies porosas ou gradeadas, o que é praticamente irrelevante considerando a aplicação e ambiente que o veículo irá ser utilizado.

Por conta destes reprogramadas com tal preço, o que encarece a sua utilização neste projeto, principalmente se for usado em maiores quantidades. Por isso a escolha final, foi do sensor HC-SR04 que apresenta uma funcionalidade muito similar e apesar de ter um tamanho maior, foi possível instalá-lo no com a modelagem de para-choques como mostrado na Figura 53.

Figura 53: Instalação física do sensor ultrassônico de proximidade.

Fonte: Elaborado pelo autor.

Com relação à instalação elétrica, o sensor pode ser conectado a partir de 4 fios. Um de VCC, de 5V, um *Ground* (polo negativo), um *Trigger* e um *ECHO* (ambos ligados à pinos digitais na placa eletrônica). O esquema elétrico pode ser visto na seção 3.3.5 deste trabalho.

Durante seu funcionamento, a placa eletrônica envia um pulso de *Trigger* de pelo menos 10µs em nível alto, ao receber o sinal de trigger, o sensor envia 8 pulsos de 40khz e detecta se há algum sinal de retorno ou não. A partir disto, se um sinal de retorno for identificado pelo receptor, o sensor gera um sinal de nível alto no pino de saída cujo tempo de duração é igual ao tempo calculado entre o envio e o retorno do sinal ultrassônico.

Por fim, a distância pode ser medida utilizando a Eq. (27) citada na seção 2.4 de fundamentação teórica sobre sensores ultrassônicos. Como o veículo fez testes em baixas velocidades neste trabalho, optou-se por acionar a frenagem de emergência quando a distância com um obstáculo iminente fosse menor que 50cm. Na seção de testes, é relatado um ensaio para testar a habilidade do sensor parar o veículo antes dele colidir com um objeto também.

O parachoque foi desenhado de forma que houvessem 4 sensores na frente e 4 atrás, sendo um para a parte frontal, dois laterais defasados com 15º do frontal e um frontal para detectar pequenos obstáculos baixos que possam ser transponíveis. Neste trabalho não é feito um estudo aprofundado do uso destes sensores (somente o frontal foi testado), porém o suporte em 3D foi desenhado para que se possa realizar estudos posteriores.

3.3.5 CIRCUITO ELÉTRICO FINAL E INSTALAÇÃO

Foi relativamente simples montar todo circuito elétrico no suporte após ele ter sido fabricado. Foi possível instalar duas *protoboards* de tamanho médio dentro do seu invólucro protegido de forma que ainda assim houvesse um espaço bom para acomodar todos os fios e sensores extras.

A diferença entre o circuito final e o apresentado na seção 3.2.4 é que este recebe um sensor de proximidade ultrassônico e um módulo IMU na conexão. Pode-se verificar o diagrama elétrico do circuito final na Figura 54.

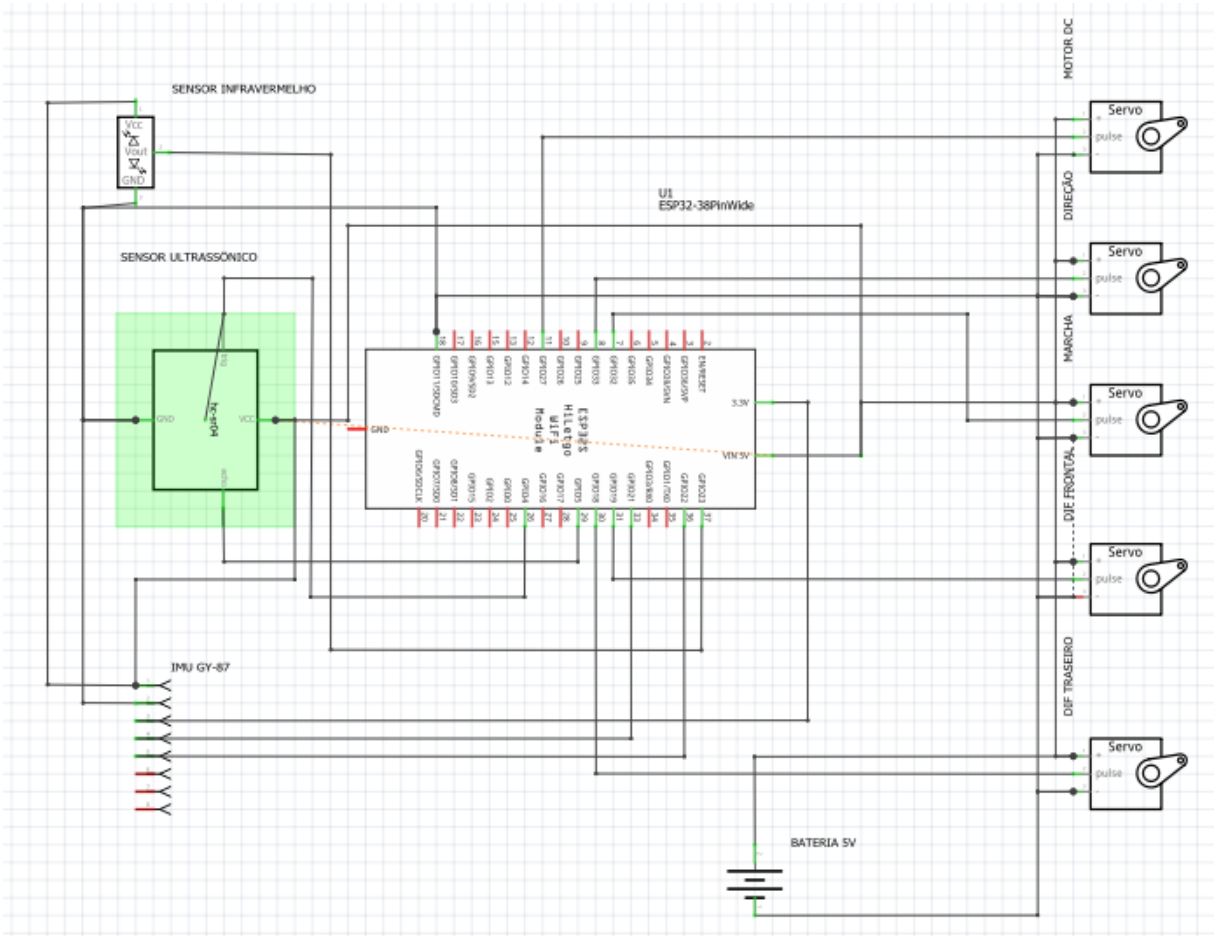


Figura 54. Diagrama do circuito final da plataforma de controle e instrumentação de um automodelo.

Fonte: Elaborado pelo autor.

Pode-se verificar abaixo uma simulação da montagem do circuito que foi feita previamente, para certificar-se de que não haveriam problemas, na Figura 55.

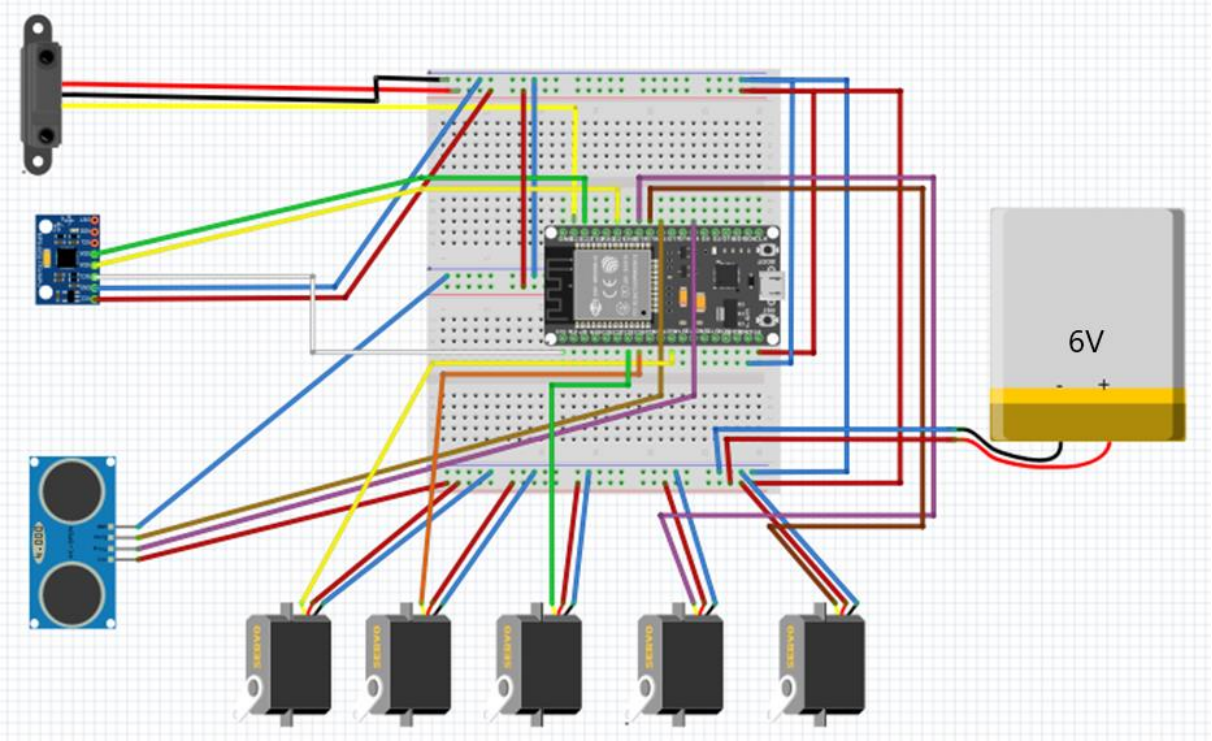


Figura 55. Simulação de circuito final da plataforma de controle e instrumentação de um automodelo.

Fonte: Elaborado pelo autor.

Pode-se verificar também, a montagem do circuito do circuito final já montado no suporte localizado na Figura 56.

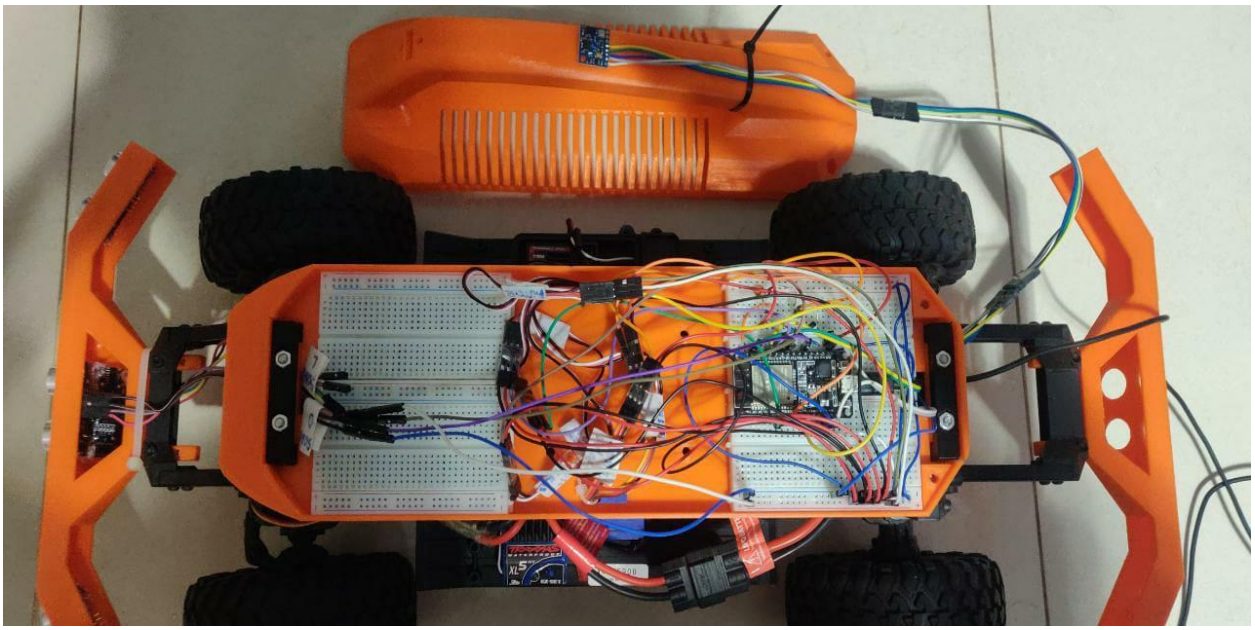


Figura 56. Foto do circuito final montado no automodelo.

Fonte: Elaborado pelo autor.

3.4 DESACOPLAMENTO DOS EIXOS DIANTEIROS

De acordo com a documentação estudada no site da Traxxas sobre o automodelo modelo 82056-4 em [23], é possível notar que apesar do veículo possuir dois sistemas de diferencial (um no eixo das rodas dianteiro e outro no traseiro), não existe um diferencial central que fizesse a compensação da rotação distribuída para as rodas dianteiras com as traseiras.

Esta característica é útil em um veículo 4x4 para que se haja uma boa tração em terrenos soltos como areia, cascalho e em situações de escalada onde se precisa ter todos os diferenciais travados para haver torque em todas as rodas. Porém, mesmo se tratando de um automodelo com medidas pequenas de bitola e entre eixos, ela impõe uma condição de deslizamento do pneu com o chão quando se realiza curvas, uma vez que as rodas estarão percorrendo distâncias diferentes, e os dois eixos estarão recebendo um número igual de rotações.

Por conta deste fator ser um possível motivo de entra em discordância com os modelos aplicados nos testes realizados no próximo capítulo, o eixo cardan que liga o motor ao eixo das rodas dianteiras foi retirado, deixando o veículo com tração traseira, apenas.

3.5 AFERIÇÃO DAS MEDIDAS CARACTERÍSTICAS MECÂNICAS DO VEÍCULO

3.5.1 BITOLA E ENTRE EIXOS

As medidas necessárias mais importantes para o cálculo da trajetória do veículo, como mostrado na seção teórica, são a bitola ***b*** e a distância entre eixos ***l***. Estas foram obtidas facilmente no site do fabricante do veículo de controle remoto na seção técnica, como pode ser observado na Eqs. (34) e (35) e na Figura 57.

$$b = 201\ mm \tag{34}$$

$$l = 324mm \tag{35}$$

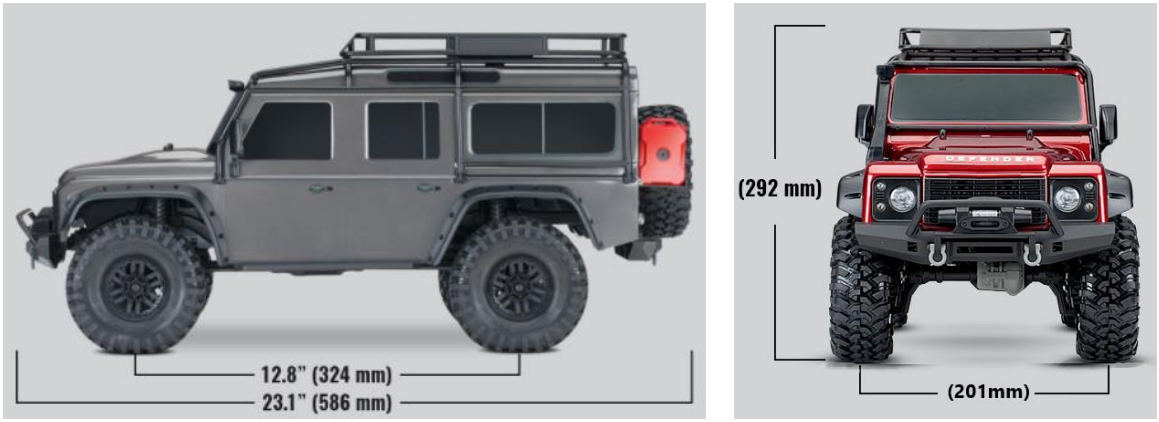


Figura 57: Distância entre eixos e bitola do automodelo Traxxas

Fonte: Adaptado pelo ator do site da Traxxas, página de informações do modelo 82056-4 [23].

3.5.2 CENTRO DE MASSA

Além da necessidade destas medidas características, foi necessário determinar o centro de massa do veículo. Como sua construção possui em sua grande parte simetria lateral (levando o eixo $\boldsymbol{X_v}$ como referência) foi considerado que seu centro de massa se encontra na medida zero do eixo $\boldsymbol{Y_v}$**.** Sendo assim, considerada a Eq. (36):

$$C_{My} = 0\ mm \tag{36}$$

Para medir o centro de massa longitudinal, apoiou-se o veículo em uma mesa plana, e passou-se um fio de barbante em baixo do suporte do veículo com todos os equipamentos, e sensores montados. Foi garantido que a linha do barbante estava reta e exatamente perpendicular com relação ao comprimento do veículo. Após isso foi-se fazendo sucessivas tentativas de levantar o automodelo passando a corda do barbante mais para a frente ou mais para trás observando a inclinação que o veículo pendulava com o movimento de levantamento.

Após um tempo, foi possível achar um ponto em que o veículo subia sem se pendular. Este ponto foi marcado e medido com uma régua. A distância entre o eixo frontal do veículo e o centro de massa no eixo $\boldsymbol{X_v}$ do veículo denominado $\boldsymbol{CM_X}$ foi de 142mm. Desta forma, obtemos que o centro de massa é o disposto na Eq. (37).

$$C_{Mx} = 142\ mm \tag{37}$$

É importante denotar que o centro de massa é justamente a origem do sistema de referencial móvel do veículo. Estas medidas foram tomadas com relação ao eixo dianteiro apenas para motivos de referência empírica, que será necessária para instalar o sensor IMU e utilizar nos modelos matemáticos. Uma foto ilustrando o procedimento com o barbante pode ser vista na Figura 58.

Figura 58. Procedimento para se descobrir o centro de massa longitudinal.

Fonte: Elaborado pelo autor.

Desta forma, pode-se deixar explícito as medidas $\boldsymbol{l_r}$ que corresponde à distância entre o centro do eixo das rodas traseiras e o centro de massa do veículo assim como e $\boldsymbol{l_f}$ que corresponde à distância entre o centro do eixo das rodas dianteiras e o centro de massa do veículo. Estes valores são informados nas Eq. (38) e (39) respectivamente.

$$l_r = 182\ mm \tag{38}$$

$$l_f = 142\ mm \tag{39}$$

3.5.3 SISTEMA DE DIREÇÃO

Foi necessário também, verificar o sistema geométrico de direção do automodelo utilizado no estudo. De acordo com a Eq. (3) apresentada na seção teórica sobre ângulos de Ackerman, é dito que a razão da bitola b pela distância entre eixos $\boldsymbol{l}$ deve ser igual à subtração das cotangentes dos ângulos externo $\boldsymbol{\delta_o}$ e interno $\boldsymbol{\delta_i}$ durante a esterção.

$$cotg\ (\delta_o)\ -\ cotg\ (\delta_i)\ = \frac{b}{l} \tag{3}$$

Onde a razão entre $\boldsymbol{b}$ e $\boldsymbol{l}$ mostrada na subseção 3.5.3, pôde ser calculada e disposta na Eq. (40):

$$\frac{b}{l} = \frac{201}{324} = 0{,}62 \tag{40}$$

Assim, na Figura 59, é mostrado uma foto perpendicular à altura do veículo mostrando sua vista superior, onde se fez uma linha a partir dos pontos de suporte da manga da roda e comparada com linhas horizontais paralelas à $\boldsymbol{X_v}$. Dessa forma foi possível adquirir os ângulos $\boldsymbol{\delta_i}$ (direcional da roda interna) e $\boldsymbol{\delta_o}$ (direcional da roda externa) utilizados no veículo da forma mais precisa possível.

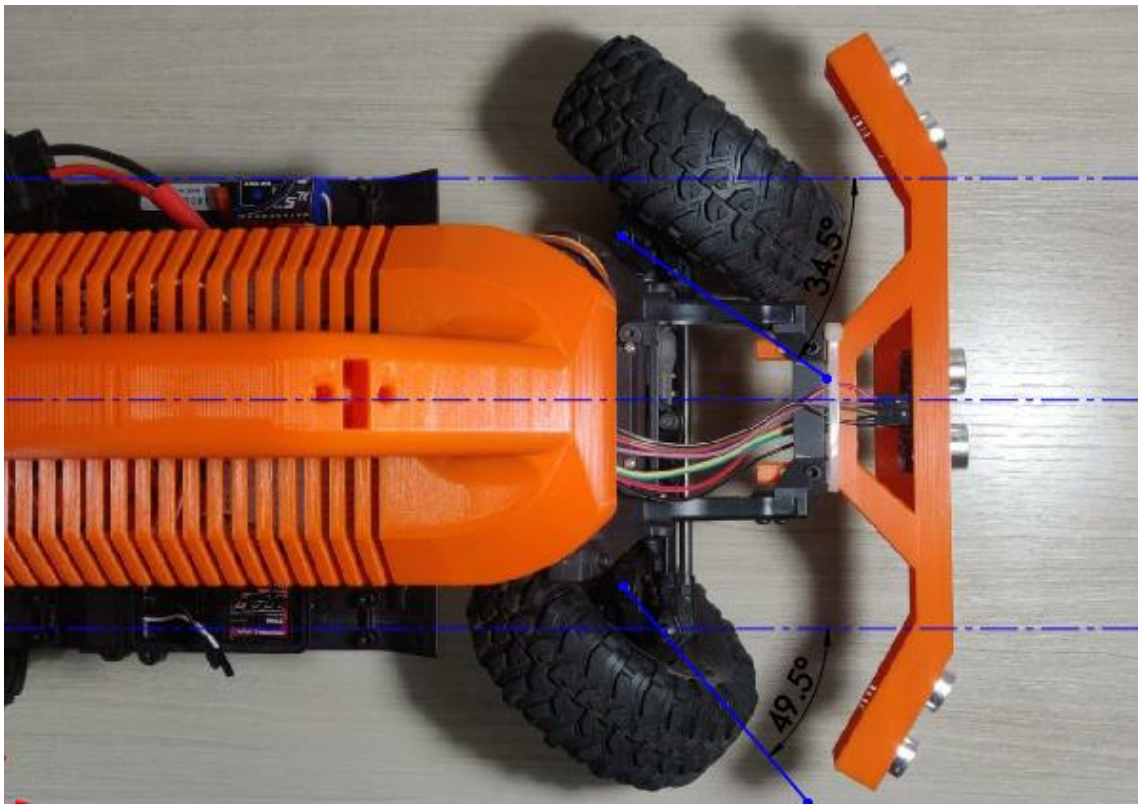


Figura 59. Foto mostrando o ângulo de Ackerman presente no sistema de direção do automodelo deste estudo.

Fonte: Elaborado pelo autor.

O próprio *software* utilizado, *SOLIDWORKS*, permitiu medir os ângulos entre as linhas projetadas pela foto da vista superior, sendo possível perceber o valor dos ângulos $\boldsymbol{\delta_i}$ = *49,5*° e $\boldsymbol{\delta_o}$ = 34,5° de acordo com a esterção total do veículo. A relação da subtração das cotangentes dos ângulos obtidos pode ser observada na Eq. (41).

$$cotg\,(\delta_o) - cotg\,(\delta_i) = cotg\,(34{,}5) - cotg\,(49{,}5) = 0{,}601 \tag{41}$$

Mesmo levando em conta que estas relações de ângulos sofrem uma pequena variação durante seu esterçamento, o resultado obtido na Eq. (41) foi muito parecido com a relação entre eixos da Eq. (40) provando que o automodelo segue um sistema de direção aproximado ao de Ackerman.

3.5.4 RAIO EFETIVO

O raio efetivo foi medido utilizando uma régua com resolução de 1 milímetro tentando capturar a medida entre a superfície do chão em que o pneu tocava e o centro do eixo da roda. A diferença entre ao tamanho da roda sem peso e com peso ode ser observada na Figura 60.

Figura 60. Comparação de roda do automodelo sem estar recebendo a carga do peso do veículo (na esquerda) e após receber o peso do veículo (na direita).

Fonte: elaborado pelo autor.

O raio da roda sem cargas de peso $\boldsymbol{R}$, declarado no site da Traxxas e o Raio efetivo $\boldsymbol{R_{ef}}$ da roda medido foi apresentado nas Eq. (42) e (43) respectivamente.

$$R = \ 59mm \tag{42}$$

$$R_{ef,1} = 55.3\ mm \tag{43}$$

Assim, pode-se calcular a circunferência efetiva $\boldsymbol{C_{irc\ ef}}$ da roda, de acordo com a Eq. (44) disposta:

$$C_{irc\ ef} = 2\pi . R_{ef} \tag{44}$$

Onde, substituindo-se os valores, acha-se o valor apresentado na Eq. (45). É importante denotar que este valor da circunferência efetiva, corresponderá à distância percorrida pelo veículo a cada 1 giro de roda.

$$C_{irc\ ef} = 347.46\ mm \tag{45}$$

Pode-se ainda, multiplicar o valor obtido na Eq. (45) pela taxa de giro roda-eixo anotada na Eq. (22) obtendo-se a distância longitudinal percorrida pelo veículo a cada volta completa que o eixo que contém o sensor gira denominada de $\boldsymbol{D_{lonPR}}$. O resultado está disposto na Eq. (46):

$$D_{lonPR} = 0{,}046839\ m \tag{46}$$

Ou simplesmente obter a medida da distância percorrida de acordo com a Eq. (47), sendo $\boldsymbol{N}$, o número de voltas do eixo contabilizadas.

$$D_{lon} = \frac{(2\pi.R_{ef}).N}{7{,}4181} \tag{47}$$

4 TESTES EXPERIMENTAIS

Aqui, o leitor pode encontrar a descrição e resultado dos testes feitos com o intuito de validar as tecnologias implementadas no capítulo anterior.

4.1 TESTES DE DISTÂNCIA LONGITUDINAL (MOTOR DC E DIRECIONAL GUIADOS POR CONTROLE REMOTO)

Após ter-se medido o raio efetivo da roda, foi criado um programa disposto no Apêndice D onde se utilizou a Eq. (47) para calcular a distância conhecida de 5m de comprimento de acordo com os giros da roda do automodelo. Neste teste, utilizou-se um suporte preliminar para conexão dos eletrônicos visto que o final não estava pronto ainda. Assim, conectou-se os terminais dos subsistemas de direção e motor DC no RRM e assim o veículo foi guiado utilizando o controle remoto em linha reta por uma distância previamente demarcada no chão. O experimento foi repetido 3 vezes e os resultados podem ser verificados na Tabela 11.

Tabela 11. Resultado dos testes de distância longitudinal, distância medida e números de rotações do eixo.

	Distância D_{exp} [m]	**Número de rotações do Eixo traseiro N**
Teste 1	5,29	113
Teste 2	5,29	113
Teste 3	5,29	113
Média	5,29	113
Desvio Padrão	0	0

Os resultados apresentaram uma ótima repetibilidade, porém, uma precisão ruim. Por causa disso, foi suposto que ao se fazer repetitivas somas da circunferência da roda do veículo, o erro da medida do raio efetivo da roda é acumulado, contribuindo para obtenção de medidas pouco precisas. Assim, para melhorar esta medida, foi testado outro método.

Ao se recolher o número de rotações do eixo, durante este percurso com uma distância real $\boldsymbol{D_{real}}$ de 5m, pôde-se calcular um novo raio efetivo $\boldsymbol{R_{ef.2}}$ de acordo com a Eq. (48).

$$R_{ef.2} = 7{,}4181\frac{D_{real}}{N\,.\,2\,\pi} \tag{48}$$

Substituindo-se os valores, obtém-se o resultado disponível na Eq. (49):

$$R_{ef.2} = 52.24014528\, mm \tag{49}$$

Para validar o novo raio efetivo, foi feito mais uma bateria de três repetições deste mesmo teste (código no Apêndice D), porém, com o novo raio efetivo e um percurso retilíneo previamente demarcado de 10m, obtendo-se os valores mostrados na Tabela 12.

Tabela 12. Resultado dos testes longitudinais utilizando o novo raio efetivo.

	Distância [m]	**Número de rotações do Eixo traseiro N**
Teste 1	10,00	226
Teste 2	10,00	226
Teste 2	10,00	226
Média	10,00	226
Desvio Padrão	0	0

Com estes resultados foi possível obter a acurácia desejada e o raio efetivo final do projeto foi considerado de acordo com o valor disposto na Eq. (49).

4.2 TESTE DE GUIAGEM EM LINHA RETA LIMITADA POR DISTÂNCIA (DIRECIONAL GUIADO POR CONTROLE REMOTO)

Este teste, teve o objetivo de concluir os estudos de modelagem de movimentação longitudinal, uma vez que validaria a utilização do contador de rotações para medição de velocidade longitudinal e distância de trajetória, assim como testaria os controles do motor estabelecidos pela placa controladora. Neste teste, utilizou-se um suporte preliminar para conexão dos eletrônicos visto que o final não estava pronto ainda.

Utilizando o programa disponível no Apêndice E, o veículo foi programado para percorrer uma linha reta com a distância já conhecida de 5m. O funcionamento deste, é composto por duas tarefas que rodam de forma independente: a primeira contabiliza a distância percorrida por meio do sensor infravermelho de velocidade instalado no veículo e a segunda é responsável por manter o motor do veículo ligado e entregando potência enquanto a distância objetivo não é atingida.

Apesar de o programa utilizado ter mantido o sistema de direção travado com a direção apontada para frente, foi observado que nas primeiras tentativas do teste, a guiagem do veículo se manteve instável, com dificuldades para se manter em linha reta. Isso ocorreu pelo fato de que o veículo possui algumas folgas, distribuição de massa heterogênea, entre outros fatores que devem ser compensadas futuramente instalando-se um sistema ativo que faça pequenas correções no sistema direcional.

O problema foi corrigido neste primeiro momento, ligando-se o subsistema de direção ao RRM e conduzindo a direção do veículo manualmente utilizando o controle remoto enquanto ele percorria o percurso lentamente da linha reta estipulada. Assim, foram realizadas 3 repetições deste teste, onde o veículo de fato começou o seu movimento e parou nas distâncias comentadas na Tabela 13 que foram medidas utilizando uma trena com resolução de 1mm.

Tabela 13. Resultado do teste de guiagem em linha reta limitado por distância.

	Distância [m]
Teste 1	5,08
Teste 2	5,10
Teste 2	5,08
Média	5,087
Desvio Padrão	0,01155

Como se pôde observar, o veículo teve uma boa performance no teste, apesar de ter percorrido uma distância um pouco maior do que ele realmente deveria. A hipótese mais provável é que isso tenha ocorrido pois o programa do teste do Apêndice E, não apresenta controles de aceleração e desaceleração durante o percurso. Assim, quando o veículo completa os 5m de distância, seus motores param de vez, fazendo com que ele percorra uma distância adicional devido à sua inércia, mesmo à baixas velocidades.

O fato de que o veículo não possui um sistema efetivo de freio, apenas o atrito do motor elétrico, também contribui para esse fator.

4.3 TESTE DE FUNCIONAMENTO DO MAGNETÔMETRO

O sensor presente dentro do módulo IMU GY-87 responsável por fazer a leitura do Norte magnético é o HMC 5883L. Este é capaz de fazer leituras de até 160Hz e tem uma acurácia de 1° à 2° segundo o seu *datasheet.*

Para testar a sua eficácia, foi montado um teste que conseguisse fazer a leitura do Norte magnético com constância durante meio minuto aplicando diferentes movimentos. Assim, ele foi

montado na placa ESP, utilizada neste estudo, e utilizou-se o código presente no Apêndice F**.** Neste, o magnetômetro recebe as leituras diferentes para cada um dos 3 eixos cartesianos (em μT), onde o vetor soma destas 3 magnitudes aponta para o Norte magnético. Como se está interessado, neste caso, apenas em um vetor de 2 dimensões, foi utilizada a Eq. (51) para se achar calcular a projeção em 2D e assim obter-se o ângulo de defasagem que o sensor está com o Norte denominado de $\boldsymbol{\Psi_N}$.

$$\Psi_N = \tan^{-1}\left(\frac{T_y}{T_x}\right) \tag{50}$$

Para garantir uma condição inicial e final igual, o automodelo foi suspenso e fixado em uma base que o mantinha estático. Desse modo, ao se iniciar o teste foram retiradas 10 leituras de posição com intervalos de meio segundo cada de modo que o sensor estava encaixado no seu suporte de forma estática. Os resultados foram anotados e retirada uma média. Logo após isso, o sensor foi retirado do suporte, movimentado em diversas direções aleatórias por um minuto e encaixado na mesma posição no mesmo suporte, onde foram anotadas mais 10 leituras finais e retirada a média. Para controle, foi girado o suporte em 180° e tomada as medidas no final também. O resultado destas leituras está anotado na Tabela 14.

Tabela 14. Medidas do teste de funcionamento do magnetômetro.

	Leitura Inicial [º]	**Leitura Final [°]**	**Leitura com 180° de defasagem após Leitura final [°]**
Medida 1	6,9	7,29	186,95
Medida 2	7,37	7,57	187,38
Medida 3	7,54	7,76	187,15
Medida 4	7,7	8,08	187,15
Medida 5	7,87	8,13	187,13
Medida 6	7,89	7,76	186,97
Medida 7	8,22	7,45	186,92
Medida 8	8,22	7,45	186,75
Medida 9	8,06	7,1	186,75
Medida 10	7,73	6,96	186,52
Média	7,750	7,555	186,967
Desvio Padrão	0,40631	0,38642	0,24958

Ao final do teste, foi possível observar uma diferença entre as médias inicial e final dentro do erro de acurácia informado no datasheet. Foi notado também, que durante os movimentos as leituras estavam se mostrando eficazes, uma vez que se movia o sensor em 180°, aparecia uma leitura de ângulo de defasagem acrescido de aproximadamente 179,5º do ângulo inicial, o que está dentro do erro esperado.

O sensor se mostrou eficaz com o seu funcionamento e assim, foi possível seguir para os próximos testes.

4.4 TESTE DE FUNCIONAMENTO DO GIROSCÓPIO

Para verificar se o giroscópio foi implementado corretamente no código, foi utilizado um teste parecido com o do primeiro para verificar se as leituras das velocidades angulares estavam acontecendo. Após verificar que haviam leituras de fato, foi implementado um código que integrava as velocidades no tempo utilizando o método de integração por retângulos, onde a placa recebia as leituras em uma amplitude de até 2.000°/s transmitindo amostras de leituras por protocolo I^2C à uma velocidade de até aproximadamente 2,6kHz.

A informação somente da velocidade angular não é tão interessante para esta parte do projeto, mas ao se integrar a velocidade angular no tempo, pode-se obter a posição angular da mesma forma que o magnetômetro. Assim, criou-se um código, utilizando a documentação de de Romain Fetick [35] para testar a qualidade da leitura de posição angular obtida pela Eq. (24), que mostra esta integral já calculada, que é discutida com mais detalhes na seção 2.4.2.

$$\Psi = \Psi_0 + \omega\,(t - t_0) \tag{24}$$

Desta forma, o mesmo procedimento do teste realizado na seção 4.3 foi repetido, mas com movimentos feitos de forma suave e lenta, os resultados foram plotados na Tabela 15. O código para este procedimento está anexado no Apêndice G.

Tabela 15. Medidas do teste de funcionamento do giroscópio com movimentos suaves e lentos.

	Leitura Inicial [º]	**Leitura Final [°]**	**Leitura com 180° de defasagem após Leitura final [°]**
Medida 1	0,1	-3,33	172,78
Medida 2	0,1	-3,01	172,78
Medida 3	0,09	-2,95	172,67
Medida 4	0,08	-2,95	172,61
Medida 5	0,08	-2,95	172,6
Medida 6	0,08	-2,96	172,6
Medida 7	0,08	-2,96	172,6
Medida 8	0,06	-2,96	172,6
Medida 9	0,06	-2,96	172,6
Medida 10	0,06	-2,97	172,6
Média	0,079	-3,000	172,644
Desvio Padrão	0,01524	0,11728	0,07486

Após estes resultados, foi repetido o mesmo experimento com movimentos mais bruscos e rápidos, os resultados estão anotados na Tabela 16.

Tabela 16. Medidas do teste de funcionamento do giroscópio com movimentos rápidos e bruscos.

	Leitura Inicial [º]	**Leitura Final [°]**	**Leitura com 180° de defasagem após Leitura final [°]**
Medida 1	0,21	-34,74	-216,77
Medida 2	0,2	-34,75	-216,77
Medida 3	0,19	-34,75	-216,77
Medida 4	0,18	-34,76	-216,78
Medida 5	0,17	-34,77	-216,79
Medida 6	0,16	-34,78	-216,8
Medida 7	0,16	-34,79	-216,8
Medida 8	0,15	-34,8	-216,81
Medida 9	0,14	-34,81	-216,81
Medida 10	0,13	-34,82	-216,82
Média	0,169	-34,777	-216,792
Desvio Padrão	0.02601	0,02751	0,01874

Ao final dos dois ensaios, foi possível perceber que o erro obtido em movimentos suaves e lentos, foi de aproximadamente 3°, o que é relativamente baixo quando comparado ao erro do ensaio de movimentos bruscos, que deu aproximadamente 35° de diferença. Isto provavelmente se deu ao fato de que em movimentos bruscos com fortes acelerações e rápidas mudanças de direção, fazem com que o leitor não capte com precisão todos os movimentos e estes erros vão se acumulando conforme se vai realizando a sequência dos próximos movimentos.

Outro fator que pode contribuir para os erros neste caso também, é o fato de o ângulo ser obtido por integração pelo método da soma de retângulos, que funciona basicamente somando a área de vários retângulos de espessura delgada **dt** e de comprimento correspondente ao valor da velocidade angular instantânea no começo desse curto período de tempo para se obter a posição angular acumulada. Como se pode visualizar na Figura 61 .

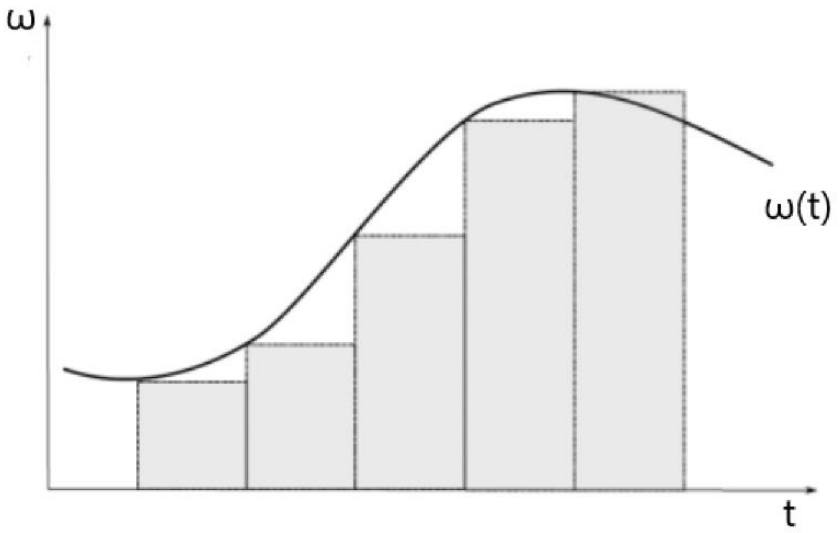


Figura 61. Demonstração de método de integração por retângulos.

Fonte: adaptado pelo autor de Paula, A. em [47]

Assim, por conta deste tipo de integração, ocorre também o acúmulo de erros ao longo do tempo, uma vez que nunca se é calculado a área exata, mas sim uma aproximação, como é explicado por Paula, A. em [47].

4.5 TESTE DE PERFORMANCE: MAGNETÔMETRO VS. GIROSCÓPIO

Como os testes deste estudo específico são feitos em baixa velocidade, foi feito um ensaio para entender melhor o motivo dos erros obtidos pelo giroscópio e comparar com os erros do magnetômetro em um único ensaio levando em conta a velocidade dos movimentos e o tempo decorrido. Dessa forma, foi realizado um estudo final, onde os dois sensores que compõe o módulo IMU foram instalados em um suporte giratório movida por um servo motor que ficou se movimentando por três minutos realizando movimentos não simétricos. O servo começou em uma posição inicial e parou na mesma após este período de tempo. Assim, foi possível observar a diferença de leitura dos dois sensores com relação ao mesmo movimento e comparar seus resultados finais.

Durante os testes, observou-se que o servo motor estava causando interferências nas leituras do magnetômetro. Isto foi resolvido criando um pequeno suporte 3D onde era possível posicionar o módulo IMU com doze centímetros de distância do motor, como é mostrado na Figura 62.

Figura 62. Plataforma de teste Giroscópio Vs. Magnetômetro, impressa para afastar o sensor do servo com o intuito de impedi-lo de sofrer interferências de leitura.

Fonte: elaborado pelo autor.

Este problema, foi verificado imediatamente no veículo também, onde descobriu-se que o motor DC gerava um campo capaz de interferir no magnetômetro se ele fosse posicionado no seu centro de massa no interior do suporte de plástico como pode ser observado na Figura 64. Para resolver este problema, o módulo IMU foi movido, ainda na linha do centro de massa do veículo referente aos eixos **X** e **Y** para a parte de cima do suporte de mão como é mostrado na Figura 63.

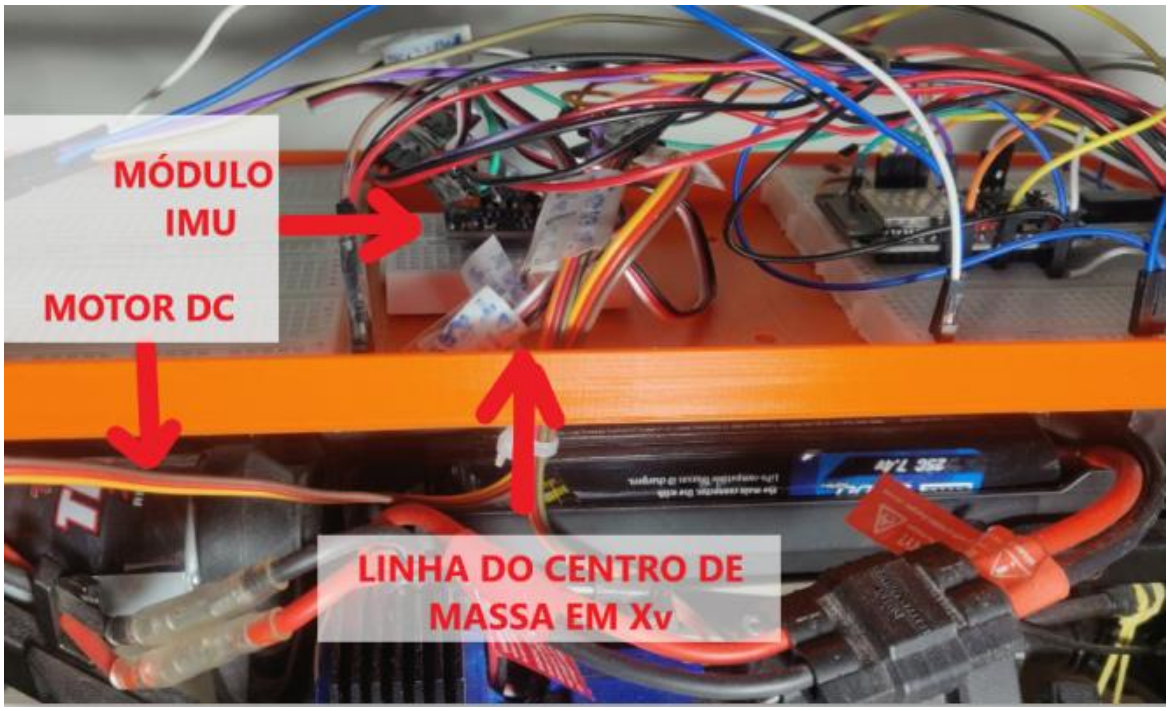


Figura 63. Sensor IMU instalado no Centro de massa do veículo referente aos eixos cartesianos Xv e Yv no interior do suporte de plástico.

Fonte: elaborado pelo autor.

Figura 64 Sensor IMU instalado no Centro de massa do veículo referente aos eixos cartesianos Xv e Yv no exterior para não sofrer influência dos motores.

Fonte: elaborado pelo autor.

O teste foi repetido sem a influência das interferências e as medidas foram tomadas durante todo o teste. Os resultados da posições iniciais e finais podem ser encontrados na Tabela 17.

Tabela 17. Medidas do teste de funcionamento do Giroscópio Vs. Magnetômetro em tempo prolongado.

	Leitura Inicial do Magnetômetro [º]	**Leitura Final do Magnetômetro [°]**	**Leitura Inicial das integrações do Giroscópio [º]**	**Leitura Inicial das integrações do Giroscópio [º]**
Medida 1	0,09	0,83	0,98	-7,00
Medida 2	0,09	0,84	1,03	-7,00
Medida 3	-0,37	0,84	0,87	-7,00
Medida 4	0,23	0,83	1,17	-7,00
Medida 5	-0,21	0,84	0,51	-7,00
Medida 6	-0,14	0,84	0,78	-7,01
Medida 7	0,16	0,84	1,07	-7,01
Medida 8	-0,24	0,83	0,32	-7,01
Medida 9	0,06	0,83	0,98	-7,01
Medida 10	-0,01	0,84	1,07	-7,01
Média	-0,034	0,836	0,878	-7,005
Desvio Padrão	0,03834	0,00516	0,27067	0.00527

Com base nos dados apresentados pode-se observar o cálculo do ângulo médio do giroscópio após os 3 minutos de ensaio apresentou uma defasagem de aproximadamente 8°, o que é um valor crítico para ser utilizado neste projeto. Além disto, foi feito um gráfico mostrando as posições angulares pelo tempo podem ser encontradas na Figura 65.

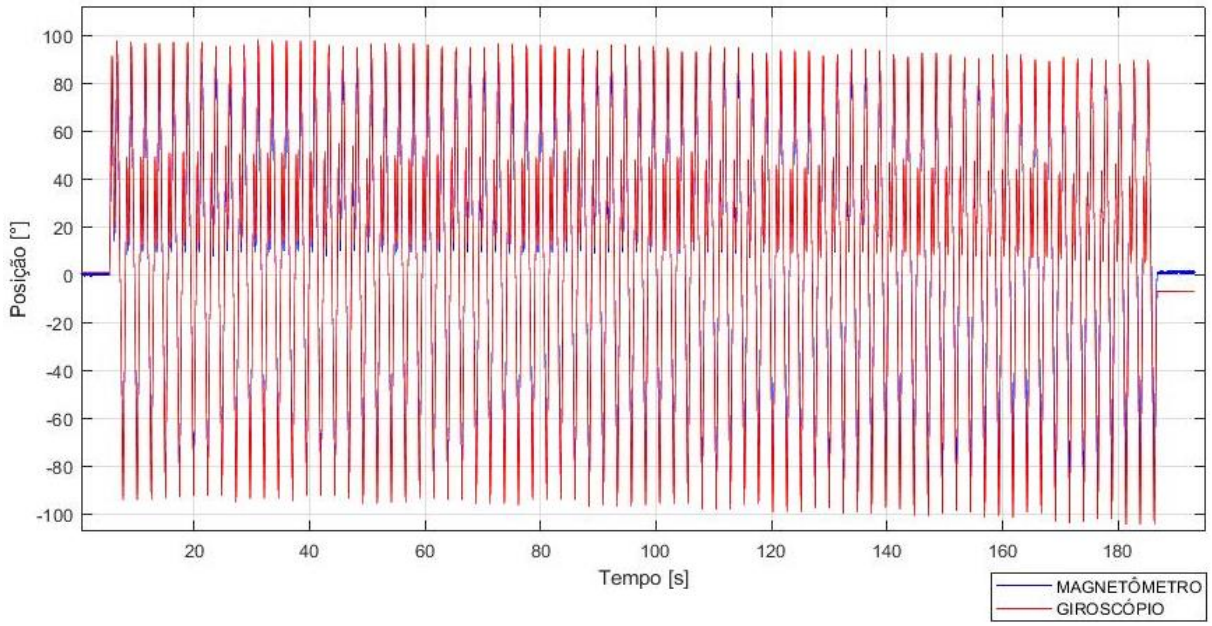


Figura 65. Gráfico posições angulares pelo tempo.

Fonte: elaborado pelo autor.

As leituras de posição angular calculadas pelos dados do giroscópio são representadas pela linha vermelha e as leituras de posição angular do magnetômetro são representadas pela linha na cor azul. Neste gráfico, é possível perceber que o giroscópio e o magnetômetro começam o ensaio parado lendo aproximadamente zero e no final, o magnetômetro permanece próximo do zero, enquanto a linha do giroscópio termina próxima do valor -7°. Além disto, pode-se observar que a linha vermelha mostrou movimentos com a amplitude ligeiramente maior, mas que ao mesmo tempo, suas medidas foram caindo, inclinando o gráfico ligeiramente para baixo, enquanto as linhas azuis permaneceram retas. Isto pode ser um forte indício que em um estudo mais longo a defasagem iria aumentar mais ao longo do tempo de movimentação.

Para se visualizar um pouco melhor, foi feito um gráfico com os últimos movimentos até o servo motor parar na mesma posição inicial, que se encontra na

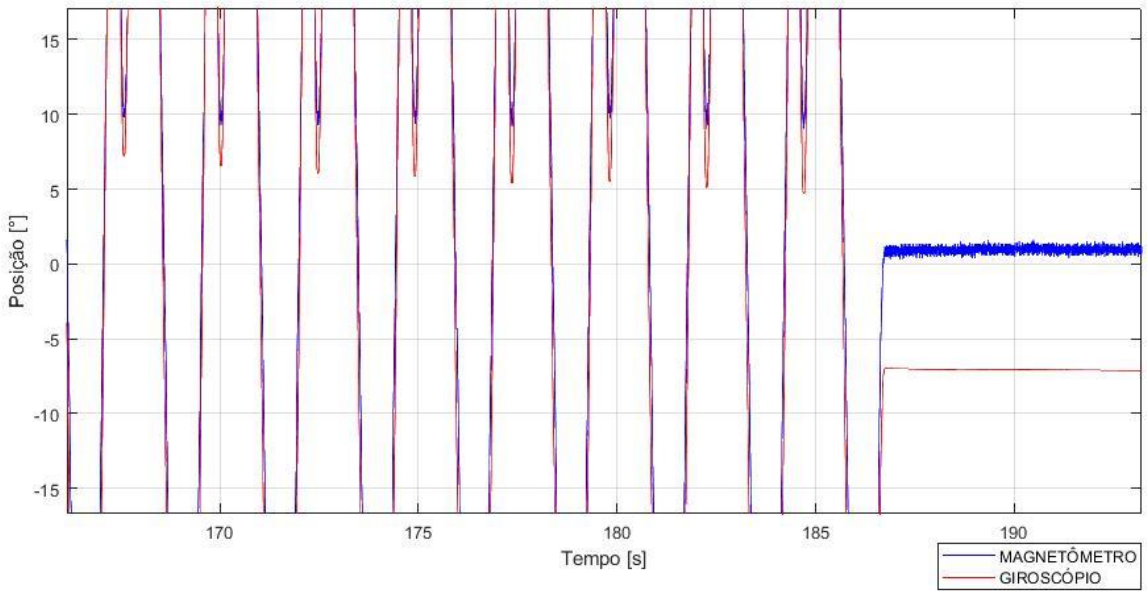


Figura 66. Gráfico mostrando o final da movimentação do ensaio do teste de performance do giroscópio Vs. Magnetômetro.

Fonte: elaborado pelo autor.

No Apêndice H podem ser encontrados dois códigos referentes a este teste, o primeiro utilizado na placa ESP com os sensores e o segundo, referente à um código utilizado para controlar o servo motor que estava movimentando o sensor.

Com base nos dados apresentados, é possível observar que ao longo do tempo e com velocidades maiores, o cálculo das posições angulares obtida pelas integrações do giroscópio vão acumulando erro e consequentemente transformando uma defasagem de leitura em um valor muito maior do que seria possível para utilizar em no sistema deste projeto. Assim, como este veículo pode sofrer vibrações, acelerações bruscas e aplicações de operações que exigem um tempo mais elevado, tornam o magnetômetro o a melhor opção para ser utilizado nesta aplicação.

4.6 TESTE DE PID EM PLATAFORMA GIRATÓRIA

Para testar um sistema de guiagem que consiga fazer autocorreções devido à terrenos irregulares, folgas no sistema de direção e outros eventos que possam tirar o veículo do percurso, um sistema de PID guiado para manter o veículo em linha reta seria um avanço importante para o projeto. Assim, foi feito novamente um pequeno projeto para se testar um código implementando a equação de um sistema PID discutida na seção 2.5 deste estudo.

Nele o módulo IMU foi posicionado em uma plataforma giratória, onde o giro era controlado por um servo também ligado ao controlador. O microcontrolador foi programado, então para fazer a

leitura da posição inicial da plataforma e utilizar um método de controle com PID para girar o servo motor sempre para a posição angular inicial caso uma força externa o girasse também.

O sistema deveria funcionar como se fosse uma espécie de *gimbal* estabilizador de um grau de liberdade. Para que isto acontecesse, foi necessário fazer várias repetições ajustando os valores dos ganhos do sistema PID. O código deste teste está disponível no Apêndice I. É possível observar uma foto do teste em execução na Figura 67.

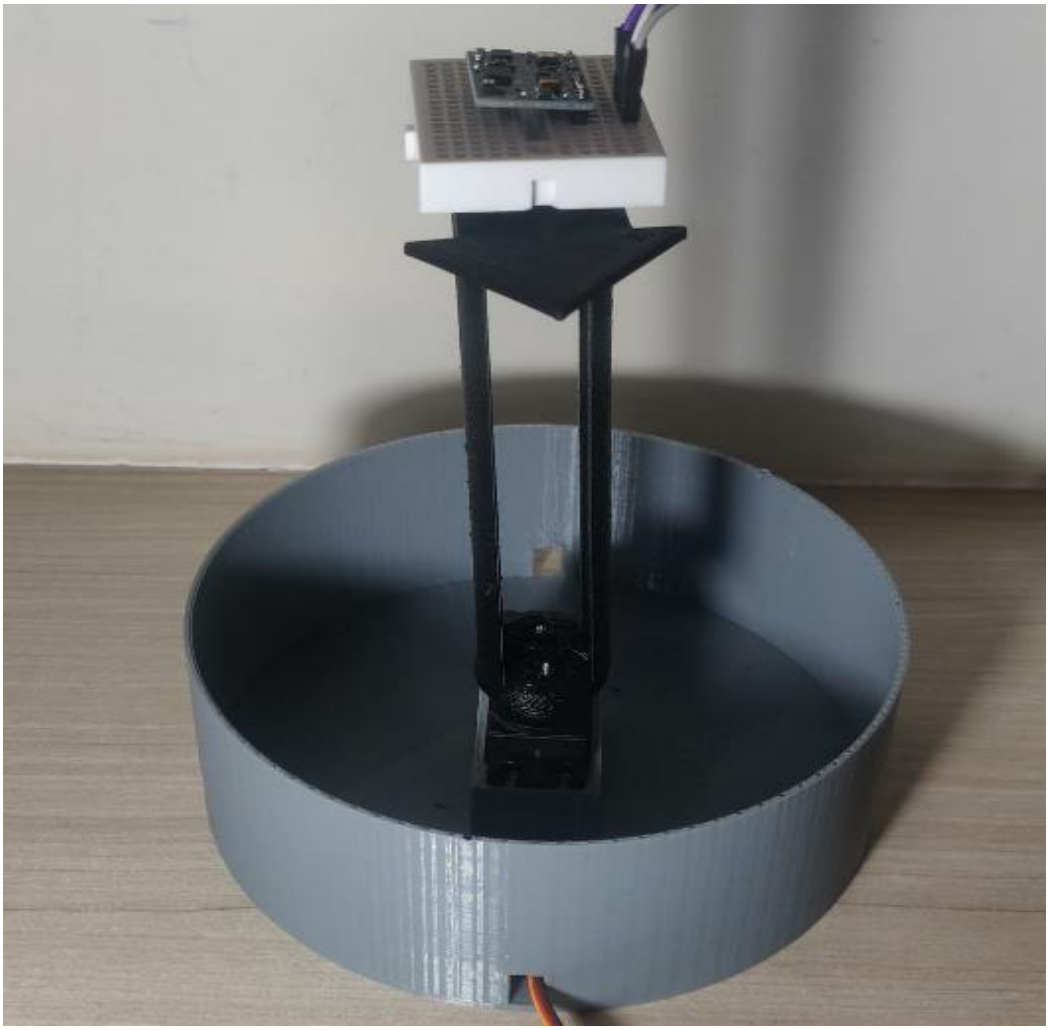


Figura 67. Teste de PID.

Fonte: elaborado pelo autor.

Desta forma, ao se ligar a placa microcontroladora, o magnetômetro lê a posição inicial, a salva e guarda como uma referência inercial, assim, ao se girar manualmente o suporte cinza, o magnetômetro reconhecia o movimento e mandava rapidamente um comando para que o servo girasse compensando o movimento e apontando a seta sempre para a mesma direção onde o código foi iniciado. No exemplo da foto na Figura 67, quando se gira o suporte cinza, o sistema irá compensar de forma que a seta aponte sempre para a direção da visão da câmera.

Para conseguir que o sistema tenha uma reação eficiente, é necessário calibrar o sistema testando a influência de diferentes valores de ganhos na equação PID. Após se chegar aos valores dados na Tabela 18, foi possível fazer movimentos com uma resposta livre de oscilações e baixo tempo de demora de resposta.

Tabela 18. Valores dos ganhos de PID do teste de PID.

Ganho:	**Valor:**
$\mathbf{K_p}$	1,0
$\mathbf{K_i}$	0,00001
$\mathbf{K_d}$	50

O código de PID foi implementado de forma bem-sucedida com respostas precisas e com baixo demora de resposta.

Para viabilizar uma testagem deste sistema e da calibração de PID, foi posicionado um transferidor com resolução de 1°, de forma que este ficasse alinhado com a seta da plataforma móvel, porém sem entrar em contato com a mesma. Na Figura 68 é possível visualizar a montagem feita.

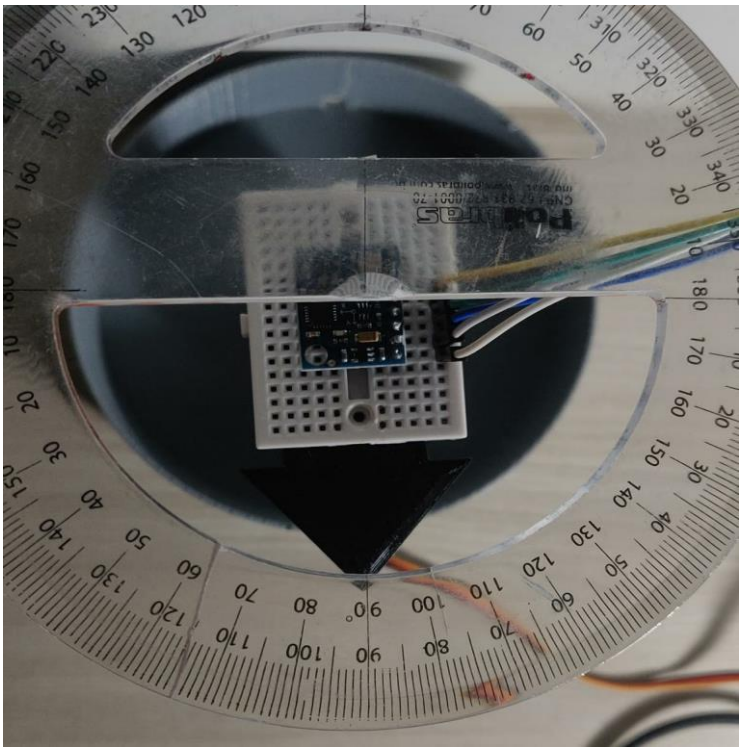


Figura 68. Imagem de teste de PID em plataforma rotativa.

Fonte: elaborada pelo autor

Assim, o procedimento ocorreu da seguinte forma: o sistema era iniciado, alinhado com o transferidor na posição angular inicial 0° e então realizavam-se movimentos sucessivos na base fixa

durante 10 segundos, após isso os movimentos eram sessados e era anotada a posição angular final com o transferidor. O sistema era então desligado.

Este ensaio foi repetido 10 vezes e seus resultados estão dispostos na Tabela 19.

Tabela 19. Dados do teste de PID.

	Leitura Inicial [º]	**Leitura Final [°]**
Medida 1	0°	1º
Medida 2	0°	-1º
Medida 3	0°	2°
Medida 4	0°	1º
Medida 5	0°	1º
Medida 6	0°	-1°
Medida 7	0°	-2º
Medida 8	0°	-1º
Medida 9	0°	1º
Medida 10	0°	1°
Média	0°	0,2
Desvio Padrão	0	1,2490

Apesar da baixa resolução do transferidor, ao se analisar os resultados, foi possível confirmar o que já se esperava, Respostas com erros entre 1° à 2° de angulação. A velocidade de resposta não foi possível ser quantizada por falta de equipamentos necessários, porém durante o experimento foi possível qualificá-la subjetivamente como rápida o suficiente para ser utilizada no veículo de controle remoto. Além do fato de que, o fator mais preponderante de demora de resposta é o tempo gasto pelo atuador, não o tempo de processamento nem o de leitura de sinais do sensor magnetômetro, uma vez que ambos conseguem entregar respostas em mais rápidas que 100Hz.

4.7 TESTE DE LINHA RETA NO VEÍCULO GUIADO POR PID UTILIZANDO UM MAGNETÔMETRO

Este teste, teve o objetivo de juntar os códigos de PID e de controle do veículo desenvolvidos nos testes anteriores e repetir o objetivo do teste de guiagem por linha reta com o veículo se corrigindo automaticamente sem nenhuma interferência feita pelo controle remoto.

Assim o mesmo procedimento do teste 4.2 foi repedido, porém desta vez, foi implementado o código disponível no Apêndice J que ao se ligar o veículo, o magnetômetro é acionado para fazer o

veículo fazer 4 leituras de cada ponto cartesiano, estabelecendo a média delas como o sistema referencial inercial. Após isto, o veículo recebe instruções para andar sempre com o seu sistema de reverencial móvel com o seu $\mathbf{X_v}$ sempre alinhado como **X** do referencial fixo utilizando controle PID para guiá-lo e sempre que houver uma defasagem no ângulo dos eixos, corrigi-lo.

Para fazer com que o sistema PID funcionasse corretamente, foi necessário fazer em torno de 20 tentativas de fazer o veículo seguir em linha reta, uma vez que a calibração de um sistema diferente exigiria ganhos diferentes de cada uma das variáveis utilizadas no teste anterior. Os valores dos novos ganhos foram anotados na Tabela 20.

Tabela 20. Valores dos ganhos de PID do teste linha reta guiada por PID.

Ganho:	**Valor:**
$\mathbf{K_p}$	0,01
$\mathbf{K_i}$	0,00002
$\mathbf{K_d}$	10

Uma vez que se chegou à um resultado satisfatório com os valores dos ganhos do PID, o teste foi repetido mais 5 vezes, o veículo foi guiado ao lado de uma linha demarcada no chão para se verificar os desvios e as correções do mesmo para mantê-lo reto. É interessante notar que ao se mudar poucas coisas no código, como a presença de alguns comandos de “Serial.print( )” foram suficientes para que fosse necessário recalibrar os sensores com novos ganhos para se manter uma resposta eficiente. Uma foto do teste pode ser vista na Figura 69, onde é possível observar a progressão da posição do veículo ao longo do teste.

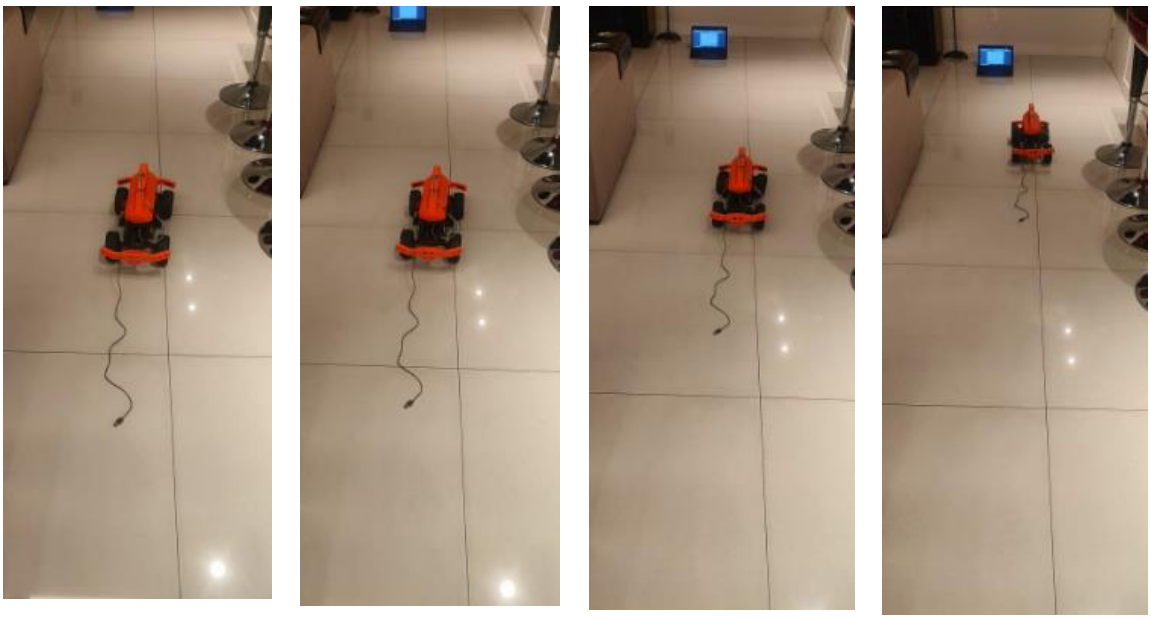


Figura 69. Teste de guiagem em linha reta com correção por PID.

Fonte: elaborado pelo autor.

Durante os 3 ensaios, os valores de distância real (medida com uma trena de resolução de 1mm), distância medida com o odômetro ultrassônico e número de voltas do eixo do veículo, foram anotados na Tabela 21.

Tabela 21. Distâncias obtidas no teste de guiagem em linha reta com correção por PID.

	Distância medida com o sensor [m]	**Nº de Voltas**	**Distância medida com a trena [m]**
Teste 1	5,31	120	5,275
Teste 2	5,26	119	5,210
Teste 3	5,18	117	5,156
Teste 4	5,22	118	5,197
Teste 5	5,18	117	5,146
Média	5,23	118,2	5,1968
Desvio Padrão	0,05568	1,30384	0,05132

Qualitativamente, o experimento mostrou que o sistema de correção aplicado pelo PID guiado pelo magnetômetro foi um sucesso. Ele manteve o veículo alinhado na maior parte do tempo à direção **X** do sistema inercial, definido na hora em que se ligava o veículo, apresentando pequenas oscilações que provavelmente podem ser diminuídas ainda mais se for feito um estudo de calibração mais detalhado.

Com relação aos resultados de distância, não foi obtida uma medida tão aceitável. Isso provavelmente se deve ao fato de que ao se adicionar a plataforma de suporte, mais sensores no veículo, e o fato de deixa-lo apenas com tração traseira, acarretou em uma adição de peso e afetou não só o raio efetivo (discutido no teste da subseção 4.1) quanto também aumentou a inércia do mesmo. Ocasionando um maior número de giros do eixo após os 5 metros, mas também uma diminuição da acurácia de medição do sensor com a trajetória real. Fatos que podem ter sido agravados ainda mais por haver uma oscilação do sistema PID durante a sua trajetória.

Assim, para se melhorar um pouco as medições, foi pego o valor da distância real $\boldsymbol{D_{real}}$ foi pego os valores de moda correspondente ao número de voltas durante o percurso correspondentes ao "Teste 3" e "Teste 5" e retirada a média das medições da trena e achado um valor de 5,151m assim como o número de rotações do sensor **N** de 117 voltas e pôde-se calcular um novo raio efetivo $\boldsymbol{R_{ef.3}}$ de acordo com a Eq. (51).

$$R_{ef.3} = 7{,}4181\frac{D_{real}}{N\,.\,2\,\pi} \tag{51}$$

Substituindo-se os valores, e convertendo para mm, obtém-se o resultado disponível na Eq. (52):

$$R_{ef.3} = 51.97787296\, mm \tag{52}$$

Assim, as medidas do sensor foram recalculadas de acordo com o número de voltas a partir do novo raio efetivo. Os dados foram plotados na Tabela 22. O algoritmo com o novo raio efetivo foi consertado para testes futuros também.

Tabela 22. Novas distâncias obtidas no teste de guiagem em linha reta com correção por PID consertando-se o raio efetivo.

	Distância medida com o sensor [m]	**Nº de Voltas**	**Distância medida com a trena [m]**
Teste 1	5,283	120	5,275
Teste 2	5,239	119	5,210
Teste 3	5,151	117	5,156
Teste 4	5,195	118	5,197
Teste 5	5,151	117	5,146
Média	5,2038	118,2	5,1938
Desvio Padrão	0,05737	1,30384	0,05132

Nota-se que apesar de se ter consertado o raio efetivo, o problema do veículo ter dado algumas voltas a mais em seu eixo após completar os 5m de distância objetivo mesmo depois de ter sua potência cortada mostra que a aceleração e desaceleração do veículo ainda devem ser consertadas, para que este movimento seja mais suave e que ele percorra uma distância mais precisa. Por isso, foi realizado outro teste para melhorar este fator.

4.8 TESTE DE IMPLEMENTAÇÃO DE CÓDIGO PID PARA VELOCIDADE CONSTANTE.

Para que haja mais facilidade de resolver as equações de movimento apresentadas no início do capítulo, é muito interessante se ter a condição de velocidade constante, que é difícil de se manter no veículo, sem a implementação de um sistema de controle de malha fechada, uma vez que ao fazer uma curva ou com diferentes estados de carga da bateria, a velocidade do veículo pode variar para um mesmo sinal de comando.

Assim, para resolver este problema, foi escrito um código e feito diversos testes para que fosse implementado um controle de PID para manter a velocidade constante durante as trajetórias realizadas. Além disso, exclusivamente nos arranques (saindo de velocidade zero) e paradas do veículo durante seu percurso, serão melhorados automaticamente ao se instalar o sistema de PID, o que pode contribuir para diminuir os erros achados no teste da subseção 4.7.

Durante o período de testes, foi também, descoberto um modo de condução do ESC denominado pelo fabricante como "*Crawling*", que diferente do modo de condução normal, toda vez que o motor deixa de receber potência, o ESC automaticamente ativa uma espécie de freio motor. Esta opção é primariamente para que o veículo fique parado no lugar em situações de escaladas e percursos inclinados, quando este não está recebendo potência, porém teve grande utilidade para frear o veículo de forma mais efetiva quando se acabava o percurso.

Durante a implementação do código, foi percebido que a velocidade de processamento da placa microcontroladora sofreu uma defasagem de tempo de resposta, suficiente para que fosse necessário ter que recalibrar o PID do sistema de direção juntamente com o novo sistema de PID para manter a velocidade constante.

Ao final de algumas repetições obteve-se os valores mostrados dos ganhos do sistema velocidade na Tabela 23 e do sistema de direção na Tabela 24. O código utilizado neste teste está disponível no Apêndice K.

Tabela 23. PID do teste de velocidade constante.

Ganho:	**Valor:**
$\mathbf{K_p}$	0,02
$\mathbf{K_i}$	0,0000001
$\mathbf{K_d}$	10

Tabela 24. PID do sistema de direção recalibrado para teste 4.7.

Ganho:	**Valor:**
$\mathbf{K_p}$	0,05
$\mathbf{K_i}$	0,00002
$\mathbf{K_d}$	60

Após ter sido obtido um resultado satisfatório, o veículo foi programado para percorrer uma linha reta de 5 metros e as distâncias lidas, reais e número de voltas foram anotados na Tabela 25.

Tabela 25. Resultados dos testes de guiagem em linha reta com PID de velocidade e direção.

	Distância medida com o sensor [m]	**Nº de Voltas**	**Distância medida com a trena [m]**
Teste 1	5,00	113	5,05
Teste 2	5,00	113	4,96
Teste 3	5,00	113	5,03
Teste 4	5,00	113	5,02
Teste 5	5,00	113	5,05
Média	5,00	113	5,022
Desvio Padrão	0	0	0,0331

Além dos dados terem apresentado uma melhora significante, diminuindo muito o seu erro, a velocidade lida instantânea no veículo durante os testes após seu arranque variou entre os números 0,94 e 1,03km/h. Assim, o teste foi concluído com sucesso pronto para o último ensaio deste estudo.

4.9 TESTE DE PERCURSO SEGUINDO TRAJETÓRIA PRÉ-CALCULADA

OBJETIVO

O objetivo do teste é fazer com que o veículo percorra a trajetória previamente simulada inserindo algumas coordenadas utilizando o Modelo da Bicicleta Cinemático para seja possível comprar a trajetória real com a simulada.

Em concordância com a conclusão tirada do artigo de Georg Schildbach (2015) [20], este estudo escolheu utilizar o modelo cinemático por sua simplicidade, por exigir menos poder de processamento e ter maior rapidez de resposta quando comparado com o dinâmico caso esta fosse calculada futuramente em um sistema embarcado. Além disso, este modelo necessita de medições mais simples das características do veículo utilizado quando comparado com o Modelo Dinâmico da Bicicleta.

ESTRATÉGIA

Para testar a guiagem do veículo de um percurso programado, foi-se estabelecido a seguinte estratégia de procedimentos:

1. Estabelece-se os pontos de passagem das coordenadas fixas **X** e **Y** para que o veículo trace uma rota;
2. Estabelece-se uma velocidade longitudinal $\mathbf{V_{lon}}$ constante medida em metros por segundo, uma velocidade angular $\boldsymbol{\omega_{\delta}}$ de resposta do motor do sistema de direção medida em graus por segundo, assim como um ângulo máximo $\boldsymbol{\delta_{max}}$ de esterçamento do sistema de direção medido em graus que esteja dentro do *range* do mesmo;
3. Executa-se o programa para calcular uma trajetória que utilize os pontos de coordenadas tanto como objetivos direcionais, como centro de suas curvaturas, para se orientar para o próximo ponto.
4. Extrai-se de *output* do programa tanto um gráfico **X** vs. **Y** mostrando sua rota desejada, como um arquivo de texto contendo uma matriz composta por uma coluna com as informações de ângulo **Ψ** de guinada medido. Em cada linha apresenta-se uma medida disposta em graus correspondente à cada centímetro percorrido na trajetória.
5. Insere-se este arquivo no código de controle do veículo e utilizado para que ele utilize um sistema de PID para controlar o seu direcional, cujo *setpoint* **SP** é o ângulo **Ψ** da matriz correspondente à uma certa distância percorrida $\mathbf{D_{lon}}$ e o *process variable* **PV** é dado pelas leituras instantâneas do magnetômetro.
6. Posiciona-se o automodelo em um ambiente aberto e executa-se o programa do microcontrolador, que irá, no momento de sua inicialização captar seu direcionamento e ponto inicial como a origem dos sistemas de coordenada inercial e móvel.
7. Assim, o percurso é filmado e analisado para que seja possível compará-lo com o simulado.

Este processo utilizando o controle PID permitiu com que se fosse possível manter o veículo na angulação certa sem que se fosse necessário haver uma preocupação grande com o ângulo de esterçamento médio **δ**. Uma vez que seria extremamente difícil e inacurado, medir seus ângulos de esterçamento relacionados com um sinal de input do servo.

Além disso, o controle PID é importante nesta implementação uma vez que ele é responsável por continuar corrigindo as imperfeições de trajetória geradas pela folga de suas peças móveis durante toda trajetória. Foi utilizado o mesmo tipo de controle também para se manter o veículo em uma velocidade constante também, o que foi uma consideração essencial para se permitir o cálculo da trajetória.

A utilização do MATLAB para calcular previamente a rota, foi essencial, pois são cálculos complexos que adicionam muito tempo de processamento ao dispositivo embarcado. Dessa forma, sendo o melhor dos mundos inserir uma rota pré-definida em sua memória para fazer o percurso.

Adicionalmente, foi importante também, estabelecer uma velocidade $\boldsymbol{\omega_M}$ de resposta do motor que fosse a mais baixa possível, uma vez que pela ausência de datasheets dos motores da TRAXXAS não foi possível descobrir a velocidade máxima dos servomotores além do fato de que isso foi necessário

para desconsiderar acelerações de percurso do motor real. Assim adotou-se como referência *datashet* de servomotores comuns de porte parecido que possuíam uma capacidade de girar 60º em 0,16 segundos. A velocidade $\boldsymbol{\omega_M}$ escolhida para ser utilizada na simulação foi de 10°/s.

INPUTS CONSIDERADOS NAS SIMULAÇÕES

Para a realização das simulações explicadas neste teste, foram utilizadas as coordenadas dispostas na Tabela 26.

Tabela 26: Coordenadas do de pontos de passagem do teste de trajetória retangular.

Coordenada X [m]	**Coordenada Y [m]**
0	0
2	0
4	1
7	-1
-11	0
13	0

Os valores das constantes referentes às características principais do veículo e outras informações consideradas nas simulações do MATLAB. citados na subseção de estratégia deste teste podem ser visualizados na Tabela 27.

Tabela 27. Valores das constantes referentes às características principais do veículo e outras informações consideradas nas simulações do MATLAB.

Constante	**Descrição**	**Valor**
b	Bitola	0,201m
l	Distância entre eixos	0,324m
$\mathbf{l_r}$	Distância entre o centro de massa e o eixo traseiro em X_v	0,182m
$\mathbf{v_{lon}}$	Velocidade longitudinal constante	1/3,6 m/s
$\boldsymbol{\delta_{max}}$	Ângulo direcional médio máximo	12°
$\boldsymbol{\omega_\delta}$	Velocidade angular do sistema de direção	10°/s
$\mathbf{P_t}$	Passo temporal de cálculos de dados	160Hz
lim	Limitador da correção de ângulo	3
tol	Tolerância de encontro de linhas final	0.01m
(X,Y)	Coordenadas X,Y fixas	(0,0) m

SIMULAÇÕES DE TRAJETÓRIA NO MATLAB

A simulação de trajetória seguindo o modelo da bicicleta cinemático foi um dos problemas mais difíceis de se resolver de todo o estudo, uma vez que o modelo é composto por equações diferenciais de primeiro grau, mostradas na seção 2.1.1 pelas Eq. (7) à (11), que possuem várias soluções possíveis. Assim, foi necessário criar um código que considerasse várias características restritivas ao sistema para chegar em um resultado plausível que pudesse seguir coordenadas plotadas como pontos de passagem.

Na primeira iteração do código, que pode ser verificado no Apêndice L, foi possível realizar uma trajetória sem a consideração de uma velocidade de esterçamento do veículo. A trajetória com os inputs anotados na Tabela 26 e Tabela 27 gerou a simulação de trajetória mostrada na Figura 70. Nela é possível visualizar **linhas vermelhas tracejadas**, correspondentes à ligação dos pontos de passagem das coordenadas **X** e **Y** do sistema de referencial dado, assim como a **linha em azul**, que corresponde à trajetória calculada para o veículo com os arredondamentos calculados para curvatura do veículo.

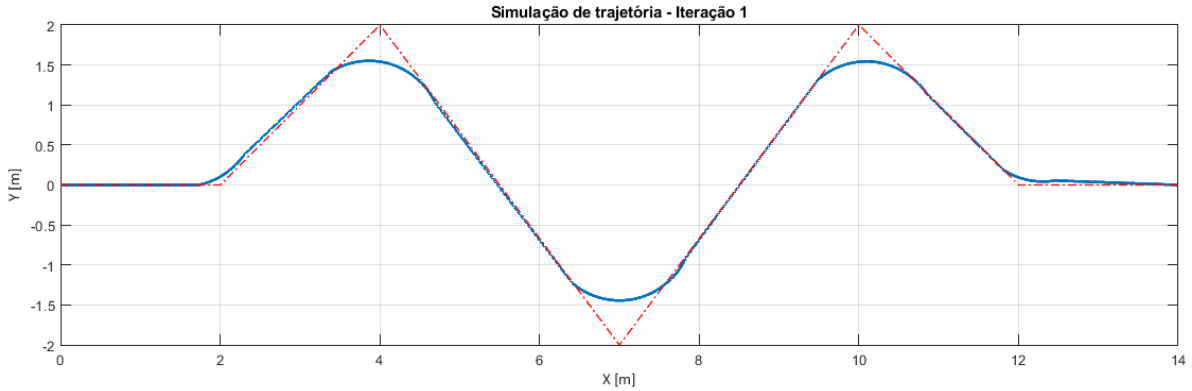


Figura 70. Iteração de simulação de trajetória retangular 1.

Fonte: elaborado pelo autor.

Neste código, o veículo recebe as coordenadas dos pontos de passagem, onde ele leva em conta o ponto de onde ele está vindo P_0, o ponto de passagem mais próximo P_1 e o segundo ponto em sucessão P_2. Assim identifica-se a reta R_1 entre P_0 e P_1 e a reta R_2 entre P_1 e P_2, calculando-se a angulação entre elas e mostrando ao veículo que é necessário realizar uma curva com tal angulação. Após isso o programa formula um raio de curvatura de acordo com δ_{max} e calcula o ponto onde ele deve sair da linha reta e iniciar a curva.

Com estes dados é possível calcular-se o ângulo de esterção médio δ realizado pelo sistema direcional, assim como o ângulo Ψ instantâneo de guinada, de acordo com as equações diferenciais do modelo cinemático da bicicleta.

Como este código não considera uma velocidade de esterção e sim que o sistema direcional se moveu de 0° para 20° instantaneamente, ele só poderia ser simulada fazendo com que o veículo andasse para frente nas linhas retas, parasse, esterçasse totalmente suas rodas, acelerasse realizando a curva, parasse no final novamente, voltasse com as rodas para a posição reta para percorrer um segmento reto

e continuasse nesta estratégia. Como isso não apresenta uma situação ideal, não foi feita nenhuma tentativa de teste físico e foi realizada uma segunda iteração de código para considerar a problemática.

Nesta segunda, disponível para verificação no Apêndice M, é possível verificar que foi adicionado uma função que permitia-se fazer incrementos no sistema de direção considerando um input de velocidade angular, de forma que o programa simulava a realização de curvas esterçadas de forma mais lenta e menos abrupta se aproximando mais do que aconteceria na realidade. O resultado foi plotado na Figura 71.

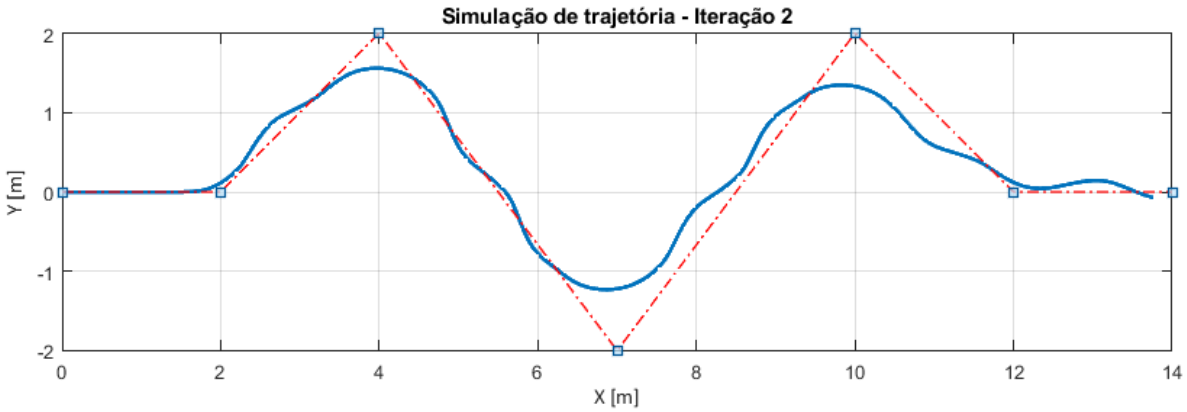


Figura 71. Iteração de simulação de trajetória retangular 2.

Fonte: elaborado pelo autor.

Ao se ter verificado o resultado, supôs-se que o que estava acontecendo é que o incremento do programa estava sempre defasado temporalmente da linha objetivo e toda vez que o programa tentava realizar uma curva, acabava passando de seu objetivo o que fazia-o tentar corrigir-se sucessivas vezes gerando oscilações. Um problema parecido com o enfrentado em sistemas de controle PID.

Assim, para que se pudesse corrigir estas oscilações foi implementado, em uma terceira iteração disposta no Apêndice N, um sistema limitador que verifica a proximidade da trajetória da curva à linha objetivo e diminui sua taxa de incremento de forma que fique mais estável, como pode ser visto no gráfico da trajetória plotado na Figura 72.

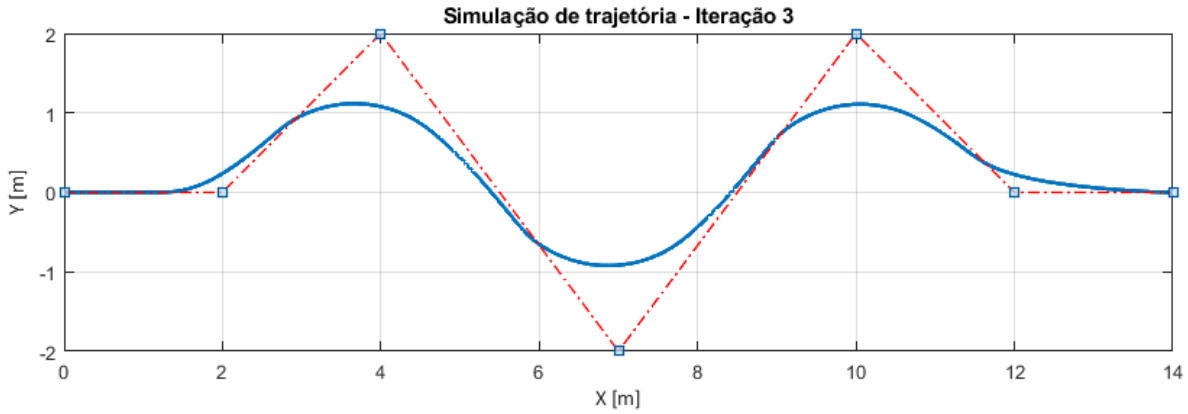


Figura 72. Iteração de simulação de trajetória retangular 3.

Fonte: elaborado pelo autor.

Esta iteração foi a final, obtendo-se um resultado satisfatório para se prosseguir com o teste físico. Após gerar o gráfico, foi gerado um arquivo de texto contendo uma matriz de duas colunas onde se eram geradas informações de distância percorrida **Dp** [m] e o ângulo **Ψ** de guinada [°] em uma frequência de 160Hz.

TESTE DE TRAJETÓRIA REALIZADA PELO VEÍCULO SEGUINDO AS INFORMAÇÕES DA SIMULAÇÃO 3

Para se inserir as informações do ângulo **Ψ** plotadas centímetro a centímetro no código do *Arduino*, copiou-se os dados do arquivo de texto e colou-se na plataforma como um vetor. De modo que, para o programa acessar o ângulo de guinada certo com relação a trajetória com relação à sua distância percorrida, bastaria puxar a linha do vetor de índice igual ao da sua distância instantaneamente percorrida em centímetros. Assim o programa utilizava os ângulos de guinada como *setpoint* instantâneo do sistema PID, que por sua vez, recebia o *feedback* do *process variable* pelo módulo com magnetômetro localizado em seu centro de massa, calculava o erro e tentava sempre corrigir o veículo. Este código pode ser verificado no Apêndice O.

Na prática, percebeu-se que isto deixou o programa ainda mais lento deixando suas respostas de PID ineficientes. Assim, foi necessário calibrar novamente o PID da direção, mas mesmo assim, não foi possível se obter respostas tão ágeis quanto antes. Além disso, o fato de o último teste necessitar de um espaço maior, foi necessário realizar em outro ambiente, com outro piso, consequentemente um novo coeficiente de atrito entre pneu e pavimento o que gerou diferenças também na calibração do sistema. Os resultados dos novos ganhos do sistema de direção podem ser visualizados na Tabela 28.

Tabela 28. Ganhos do sistema PID de direção no teste 4.9.

Ganho:	Valor:
$\mathbf{K_p}$	0,008
$\mathbf{K_i}$	0,00001
$\mathbf{K_d}$	10

É importante relatar também que previamente à calibração, durante curvas mais fechadas, o sistema PID tendia a acumular grandes outputs devido à altos erros que consequentemente não conseguiam retornar ao percurso correto e faziam o veículo ficar girando em círculos.

Após isso, foi demarcado no piso onde o teste seria realizado, várias linhas para facilitar a medição da trajetória realizada pelo veículo. Este esquema pode ser visualizado na Figura 73. Nela é possível ver a linha de referência que liga os pontos de coordenadas de referência, assim como marcações de 1 em 1 metro na direção X e linhas verticais separadas 2 à 2 metros (na direção Y). Assim é possível fazer um paralelo com o gráfico da trajetória programada mostrado na Figura 72.

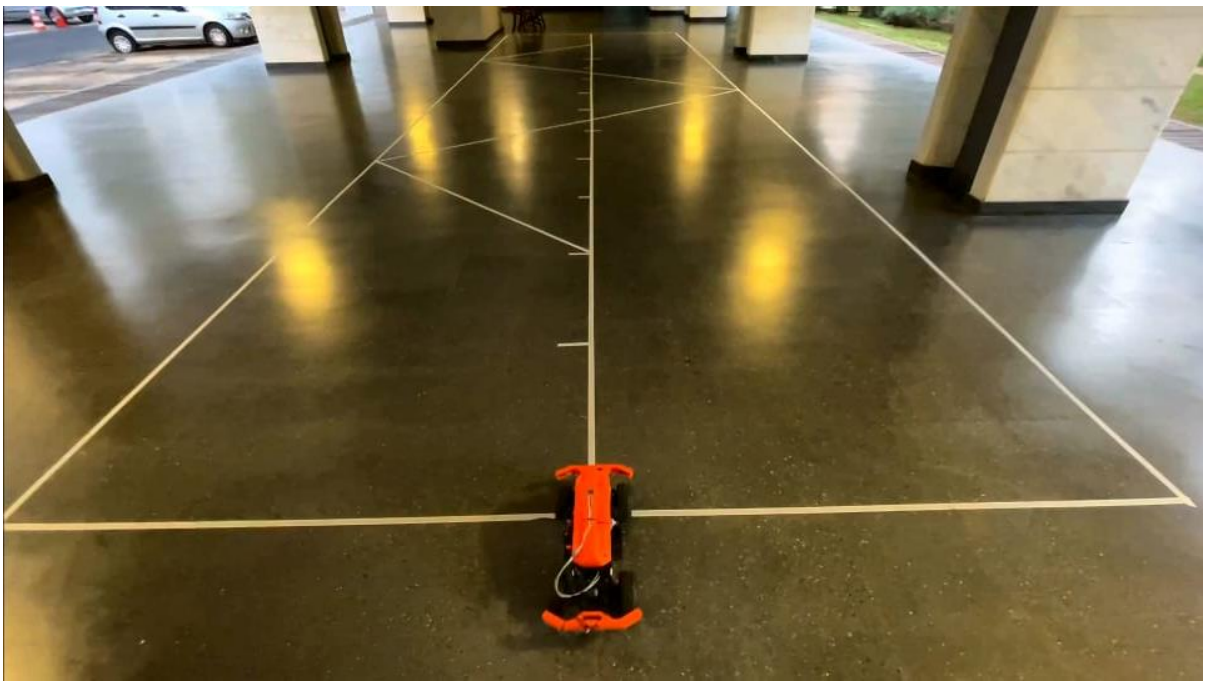


Figura 73. Foto do percurso do teste demarcado.

Fonte: elaborado pelo autor.

Assim o teste foi iniciado e gravado um vídeo com a angulação mais superior possível com o intuito de capturar todo o percurso. Durante sua execução foi possível perceber que o veículo realizou com êxito a trajetória programada, porém, com erros.

As curvas desenhadas pela linha de trajetória foram realizadas no geral antes do momento certo, além de que por algumas vezes pode-se perceber que o veículo apresentou algumas derivações em sua direção principalmente nos momentos seguidos das curvas mais fechadas pelo zigue-zague.

Foi observado também uma leve tendência neste percurso para o lado direito e percebeu-se que o veículo antes do ponto em **X** de objetivo final e com um deslocamento lateral na direção **Y**, como se pode observar na Tabela 29.

Tabela 29. Coordenadas iniciais e finais do teste 4.9.

Situação	Coordenada X [m]	Coordenada Y [m]
Posição inicial da trajetória simulada	0	0
Posição inicial do teste	0	0
Posição final da trajetória simulada	14	0
Posição final do teste	13,21	0,23

Como este teste não possuiu grandes meios de medição, a análise foi majoritariamente qualitativa de acordo com as capturas da cenas do vídeo do teste, que podem ser visualizadas no compilado da Figura 74.

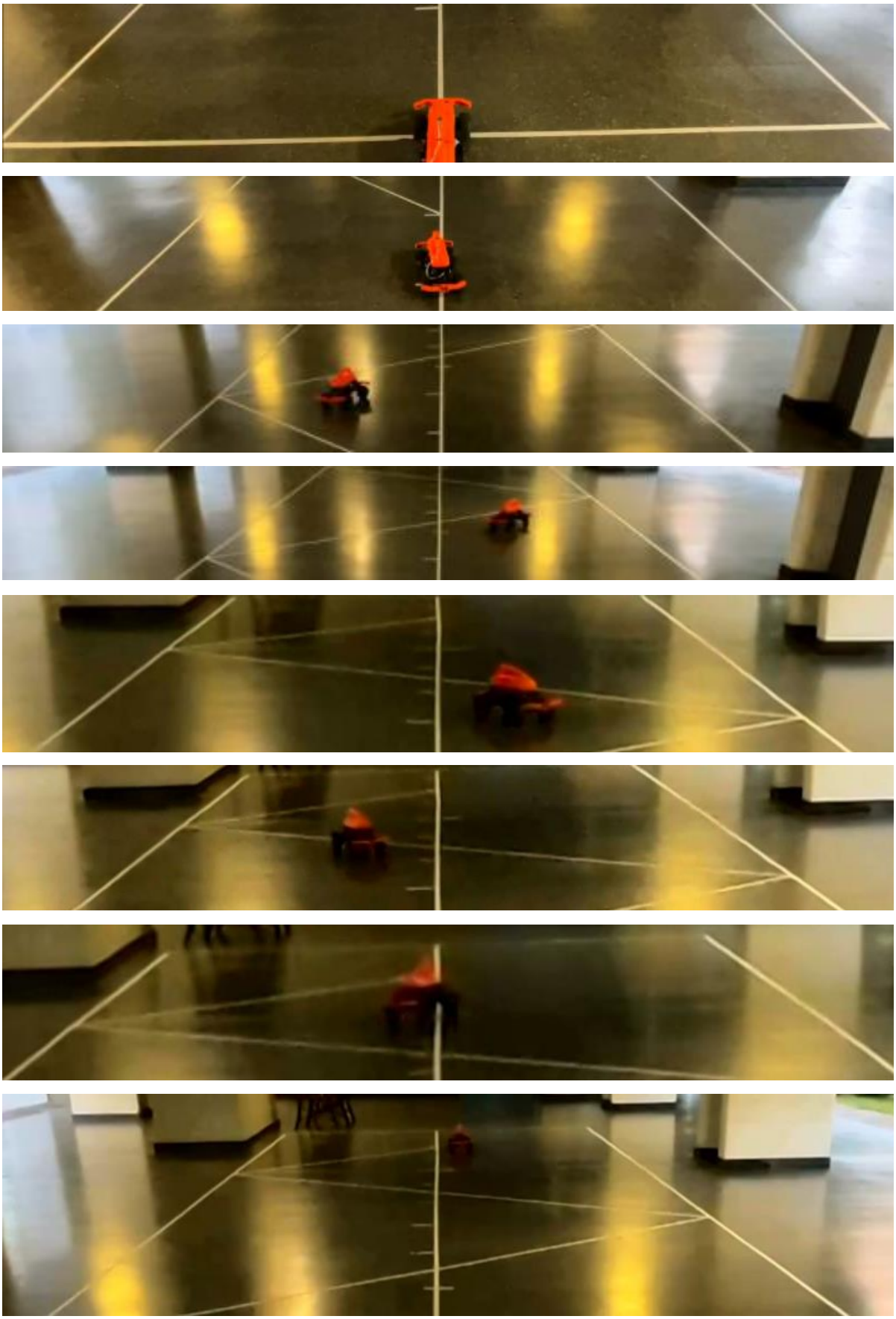


Figura 74. Capturas do vídeo do teste 4.9 - Percurso guiado por trajetória.

Fonte: elaborado pelo autor.

Para que se pudesse ter uma noção um pouco melhor do resultado quantitativo do teste, ele foi repetido mais 4 vezes e todas as coordenadas finais foram anotadas, como se pode ver na Tabela 30.

Tabela 30. Coordenadas finais dos ensaios do teste 4.9.

Teste:	**Coordenada em X [m]**	**Coordenada em Y [m]**
Valor Simulado	14	0
Teste 1	13,21	0,23
Teste 2	12,96	0,67
Teste 3	12,31	-0,11
Teste 4	13,45	1,01
Teste 5	13,56	0,27
Média dos testes	13,098	0,414
Desvio padrão	0,4449	0,3872

Assim, pode-se perceber que apesar da média dos erros em $\boldsymbol{X}$ comporem apenas 6,4% do comprimento máximo nesta direção e 10,1% de erros médios em relação ao range máximo percorrido em $\boldsymbol{Y}$, houveram muitas oscilações e imprecisões nos percursos. É provável que grande parte causadora das distâncias finais mais curtas sejam devido às grandes oscilações que consomem a velocidade longitudinal de forma errônea.

Pôde-se concluir que uma calibração mais fina e meticulosa dos sistemas de PID daria a possibilidade levando em conta um contato pneu-pavimento específico e com um código mais eficiente dariam resultados melhores com menos variações, porém o hardware apresentado também se mostrou um forte limitador.

É provável também, que outros métodos de transmitir as posições da trajetória simulada sejam mais eficientes, como o de estabelecer uma relação precisa entre o ângulo de esterção médio $\boldsymbol{\delta}$ e o valor do sinal de output para controle do servo com um angulímetro. Dessa forma, seria possível receber inputs do próprio ângulo $\boldsymbol{\delta}$. Além de ser interessante a utilização de placas controladoras mais rápidas e sensores com maior precisão.

Possivelmente poderia se fazer um estudo, também, levando-se em conta a realização do teste com e sem PID de velocidade, visto que traz uma constância de 1 Km/h, porém deixa o PID de direção mais lento ao mesmo tempo.

5 CONCLUSÃO

Os resultados encontrados durante os testes, assim como uma análise geral e ponderação com os objetivos iniciais são apresentados neste capítulo.

5.1 CONCLUSÃO

O objetivo deste estudo foi instalar uma plataforma em um automodelo, instrumentando-o com sensores e implementando um sistema eletrônico de arquitetura modular e programável/flexível que fosse capaz de se conectar com todos os motores atuadores pré-existentes do automodelo, assim como os novos sensores instalados. Além de desenvolver um software que seja capaz de receber comandos de trajetórias, adquirir dados do meio ambiente e realizar leituras de dados internos de acordo com sua dinâmica veicular aproximada. Estabelecendo assim uma plataforma de estudo base que sirva para futuros projetos que desenvolvam sistemas de automação e assistividade veicular.

Para a validação da plataforma, foi feito um estudo de caso com uma malha de controle de trajetória utilizando um PID e os sensores implementados. Antes disso, porém, foi necessário estudar o funcionamento do automodelo, projetar sistemas de sensoriamento, suporte e controle, assim como testá-los separadamente de forma a montar a estrutura utilizada na validação final.

5.1.1 CONCLUSÕES OBTIDAS DURANTE A IMPLEMENTAÇÃO, PRIMEIROS TESTES E VALIDAÇÕES

Em primeiro lugar foram feitos alguns testes de viabilidade do projeto e tomadas uma série de medidas características importantes do automodelo. Além disso, foi feita uma revisão teórica básica de todos os componentes e modelos matemáticos necessários para seguir com o estudo.

Após estes testes, foi modelado e impresso em 3D, um suporte/invólucro universal instalado no chassi do automodelo que possibilita acolher não só a plataforma utilizada neste estudo, mas outros sensores, câmeras e futuras placas microcontroladoras diferentes que possam ser implementadas em projetos futuros.

Quanto à instrumentação, pode-se citar que houve uma implementação e testes bem-sucedidos de um odômetro e velocímetro feitos a partir de um sensor infravermelho que detecta o giro do eixo, da implementação de um módulo composto por um IMU juntamente com um magnetômetro e sensores ultrassônicos para medir a proximidade.

O sistema de medição de velocidade e distância longitudinal foi implementado com sucesso entregando valores com mais acurácia do que era esperado além do fato de que foi possível fazer um teste onde o próprio programa lia a distância percorrida previamente estipulada acionando e desligando

seu motor. O que basicamente valida a parte de implementação de controle longitudinal, apesar de que o veículo não conseguiu ficar alinhado com a reta original no teste, pelo fato de ainda não haver um controle do sistema de direção que fosse capaz de impor pequenas correções (causadas por folgas do sistema) necessárias para manter o veículo na em linha reta. Este problema foi resolvido instalando-se o módulo GY-87 composto por um sensor IMU juntamente com um magnetômetro.

Durante os testes deste módulo, foi feito um estudo comparando a aquisição de dados de posição angular instantânea por meio de um magnetômetro e de um giroscópio por meio da integração de sua velocidade angular pelo tempo. Foi concluído que o giroscópio perdia acurácia nas suas medições ao longo do tempo e movimentos, causando erros discrepantes para longos testes, escolhendo assim a melhor opção o magnetômetro. Foi descoberto também, que este último possuía a limitação de necessariamente ser posicionado à uma distância mínima de motores elétricos e outros dispositivos que pudessem gerar campos eletromagnéticos fortes capazes de causar interferências em suas medições.

Foi implementado um estudo e diversos testes sobre a aplicação de controles PID em sistemas de malha fechada para controlar os erros e derivações na direção do veículo durante o percurso de trajetórias, assim como para manter a velocidade longitudinal do veículo constante durante toda a rota. O sistema PID foi aplicado com sucesso nas situações específicas, porém percebeu-se havia a necessidade de calibrá-lo diversas vezes, quando ocorriam mudanças mínimas das condições de contorno, como um pavimento levemente diferente, novos comandos executados no processador, etc.

Uma das conclusões que pôde ser tirada entre os diversos testes feitos, foi que a adição de novos códigos na linha de processamento da placa de prototipagem ESP 32 utilizada neste estudo afetava diretamente o tempo de resposta e calibração dos sistemas de PID, principalmente o PID que é responsável por controlar o sistema direcional.

Em concordância com a conclusão tirada do artigo de Georg Schildbach (2015) [20], este estudo escolheu utilizar o modelo cinemático por sua simplicidade, por exigir menos poder de processamento e a rapidez de resposta quando comparado com o dinâmico. Assim, foi desenvolvido um programa no MATLAB onde se pode calcular as trajetórias baseadas em um modelo cinemático da bicicleta baseado nas características medidas do veículo descritas em detalhes neste projeto.

O programa é capaz de pegar diversos pontos de coordenadas e desenhar uma trajetória considerando um ângulo de curvatura máximo, velocidade de esterção, distância entre eixos e centro de massa, levando-se em conta uma velocidade longitudinal constante pré-definida. Ainda é possível exportar diversos dados da trajetória, como ângulo de esterção, ângulo de guinada, distância percorrida entre outros em forma de matriz por arquivo de texto no final dos cálculos, de forma a facilitar a importação das coordenadas e dados para a leitura do programa pelo chip reprogramável.

5.1.2 ESTUDO DE CASO FINAL EM TESTE DE PERCURSO SEGUINDO UMA TRAJETÓRIA PRÉCALCULADA

No teste final, foi reunido todos os conhecimentos adquiridos, resumidos na subseção 5.1.2, para que fosse possível fazer o automodelo realizar o percurso de uma trajetória programada com base no Modelo Cinemático da Bicicleta. Ao estabelecer a trajetória no programa do *MATLAB*, foi tomada a estratégia de se obter uma matriz de uma coluna onde era disposto o ângulo de guinada $\boldsymbol{\Psi}$ do veículo simulada à cada centímetro do percurso. Assim o veículo lia esse ângulo e utilizava de *setpoint* no controle de PID durante seu funcionamento para esterçar a direção de modo a se mimetizar a trajetória simulada. Paralelamente há um outro sistema PID mantendo uma velocidade constante de 1Km/h.

O resultado obtido dos testes foi positivo no sentido de o veículo ter de fato percorrido a trajetória, porém com erros consideráveis, que podem ser verificados na seção 4.9 deste documento. Considerando o erro dos sensores o resultado obtido foi bom, mas apesar do veículo ter seguido majoritariamente a sua trajetória, o caminho foi repleto de oscilações do sistema direcional, ficou defasado na distância do eixo **X** e não foi capaz de obter uma posição final com boa repetibilidade, mostrando que ainda há muito espaço para a melhora.

Tal melhora, pode acontecer fazendo-se uma calibração mais minuciosa e fina dos ganhos dos sistemas PID, assim como a programação de um código mais eficiente para o controlador, mas é possível que não seja muito grande. Isso se deve ao fato de que o projeto ainda é fortemente limitado pela capacidade de processamento da placa controladora entregar uma resposta rápida suficiente para o sistema PID funcione de forma eficiente, assim como os sensores não tem uma precisão tão grande que permita uma variação menor.

Assim, pode-se sugerir para próximos estudos a implementação de novos testes utilizando uma estratégia diferente da guiagem exclusivamente pelo ângulo de guinada, por exemplo uma que junte o ângulo de esterçamento juntamente com o ângulo de guinada e o estudo de outras tecnologias para ajudar na localização do veículo, como a utilização de LIDARs, câmeras de reconhecimento de imagem, sensoriamento de distância por laser, etc.

Além disso é provável o uso de um microprocessador no lugar de um microcontrolador seja mais adequado para resolver problemas mais complexos e abrangentes de guiagem autônoma, uma vez que o ideal é cruzar várias informações de sensores em tempo hábil de resposta para se obter uma direção mais efetiva, segura e precisa.

5.2 ESTUDOS FUTUROS

Como este trabalho tem o objetivo de servir de base para novos estudos em cima da plataforma, o suporte impresso em 3D foi desenvolvido de forma que coubesse a maioria dos microcontroladores e processadores, havendo um lugar específico para se encaixar um suporte de câmera, sensores

ultrassônicos traseiro e diagonais, entre outros. Dentre os trabalhos possíveis, podem ser destacados alguns como:

A implementação de sistemas de sensores de proximidade nas extremidades do veículo para realização de paradas de emergência que possam deixar o veículo mais seguro contra acidentes pode ser uma boa ideia também. Este estudo pode ser extremamente complexo, uma vez que a qualidade do seu funcionamento ao distinguir obstáculos físicos transponíveis versus obstáculos não transponíveis é extremamente complexa.

Além de obstáculos, pode-se desenvolver um sistema que é chamado pelo mercado de automóveis como *Parking Assitant*, que consistem basicamente na localização de uma vaga por meio de sensores de proximidade e câmera de ré que permitem que o carro estacione sozinho em vagas de baliza e perpendiculares.

Pode-se fazer também, o estudo de modelos dinâmicos que levem em conta os efeitos dos diferenciais travados, considerando o deslizamento entre os pneus e o solo, pode se demonstrar extremamente complexo e interessante. Além disso, pode-se calcular algumas medias experimentais também como o coeficiente de dureza lateral dos pneus em determinado tipo de solo, por exemplo.

Outro sistema interessante é a implementação de um controle que realize a guiagem semi-autônoma do veículo por meio de reconhecimento de imagens adquiridas por uma câmera em tempo real e/ou por um sensor *LIDAR*.

Todos estes estudos mostram uma capacidade de se chegar mais perto do objetivo final, de implementar uma plataforma que deixe o veículo realmente autônomo.

REFERÊNCIAS BIBLIOGRAFICAS

1 Almeida, Daniel Gonçalves. Implementing and Tuning an Autonomous Racing Car Tesbed. 2019. 116 f. Tese (Mestrado em Ciências da Computação) - Curso de Pós-Graduação em Engenharia, Instituto Superior de Engenharia do Porto, Porto, Portugal, 2019.

2 DELCKER, JANOSCH. The man who invented the self-driving car (in 1986). Long before Big Tech got behind the wheel, Ernst Dickmanns unleashed a driverless Mercedes onto the roads of Europe. Politico. Disponível em: < https://www.politico.eu/article/delf-driving-car-born-1986-ernst-dickmanns-mercedes/ >. Acesso em: 13 julho 2020.

3 SIGFUSSON, LAUREN. How Do Autonomous Cars Work?. Discover Magazine. Disponível em: <https://www.discovermagazine.com/technology/autonomous-cars.> Acesso em: 13 julho 2020.

4 Guedes, Nuno Miguel Santos. A Robotic Platooning Testbed for ITS Components. Porto, 2019, 88 f. Dissertação (Mestrado em Ciências da Computação) - Curso de Pós-Graduação em Engenharia, Instituto Superior de Engenharia do Porto, Porto, 2019.

5 Silva, José Miguel Mestre Fernandes da. Enabling Cooperative Vehicle Platooning in a Robotic Testbed. Porto, 2020, 88 f. Relatório (Licenciatura em Engenharia Eletrotécnica e de Computadores) - Curso de Pós-Graduação em Engenharia, Instituto Superior de Engenharia do Porto, Porto, 2020.

6 Amer, N. H. et al. Modelling and Control Strategies in Path Tracking Control for Autonomous Ground Vehicles: A Review of State of the Art and Challenges. Journal of Intelligent & Robotic Systems, n.86, p.225–254, mai. 2017.

7 WORLD HEALTH ORGANIZATION. Road traffic injuries. World Health Organization. Disponível em: <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries#:~:text=Approximately%201.35%20million%20people%20die,road%20traffic%20crashes%20by%202020.>. Acesso em: 15 novembro 2020.

8 INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. Road vehicles – Vehicle dynamics and road-holding ability – Vocabulary, ISO 8855:2011(E).

9 Jazar, Reza N. Vehicle Dynamics: Theory and Applications. RMIT University, Melbourne VIC, Australia, 2009. 1020p.

10 Educação Automotiva. O que são as bitolas de um carro?. 2017. Disponível em: < https://educacaoautomotiva.com/2017/03/03/bitolas-de-um-carro/>. Acesso em: 15 novembro 2020.

11 TAKÁCS D & STÉPÁN G. Contact patch memory of tyres leading to lateral vibrations of four-wheeled vehicles. The Royal Society. Budapest, Hungary, v.371, n.1993, jun. 2013.

12 MILLIKEN & MILLIKEN. Race Car Vehicle Dynamics. SAE, Warrendale, USA, 1995. 890p.

13 SCHILDBACH, GEORG. UNIVERSITY OF LUEBECK. Vehicle Dynamics. Lista de videoaulas. Disponível em: <https://www.youtube.com/playlist?list=PLW3FM5Kyc2_4PGkumkAHNXzWtgHhaYe1d>. Acesso em: 23 setembro 2020.

14 GILLESPIE, D. THOMAS. Fundamentals of vehicle Dynamics. SAE, Warrendale, USA, 1992. 470p.

15 WERLING, MORITZ, et al. Invariant Trajectory Tracking With a Full-Size Autonomous Road Vehicle. IEEE Transactions on Robotics Journal. v.26, n.4, ago. 2010.

16 REIMPELL, JORNSEN et al. The Automotive Chassis: Engineering Principles. 2. ed. SAE, Warrendale, USA, 2002. 454p.

17 PACEJKA, H. Tire and Vehicle Dynamics. Society of Automotive Engineers, Inc. and Butterworth Heinemann, Warrendale, PA, ISBN 978-0-7680-1126-5, 2002.

18 LIU, GANG et al. Development of Effective Bicycle Model for Wide Ranges of Vehicle Operations. SAE Technical Paper 2014-01-0841, 2014, doi:10.4271/2014-01-0841

19 BRAVO, DIEGO MORENO. Projeto Cinemático da Suspensão e Direção de um Veículo de Fórmula SAE Elétrico. Campinas, 2013, 84 f. Dissertação (Graduação em Engenharia Mecânica) – Universidade Estadual de Campi as, Faculdade de Engenharia Mecânica, Campinas, 2013.

20 SCHILDBACH, GEORG. UNIVERSITY OF LUEBECK. Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design. IEEE Intelligent Vehicles Symposium. v.4, p.1094-1099, Seoul, 2015.

21 MECAWEB EDUCATION SITE. PWM - **Modulação Por Largura de Pulso**. Disponível em: <http://www.mecaweb.com.br/eletronica/content/e_pwm.>. Acesso em: 8 novembro 2020

22 GROVE AIRCRAFT LANDING GEAR SYSTEMS Inc. **Important Torque Considerations for Large Diameter Tires**. Disponível em: <https://www.groveaircraft.com/torque.html>. Acesso em: 8 novembro 2020

23 Traxxas, Disponível em: <https://traxxas.com/sites/default/files/82056-4_TQi_explodedviews_200206_82056-4%20Front%20Assembly.jpg.>. Acesso em: 17 outubro. 2020.

24 MILLS, MATT. Site ITIGIC. **Self-driving Car: Everything You Need to Know**. 2020. Disponível em: <https://itigic.com/autonomous-car-everything-you-need-to-know/>. Acesso em: 8 novembro 2020.

25 SVOBODA, ELIZABETH. Site Popular Science. **PopSci's Darpa Grand Challenge Preview: Update #4**.2005 Disponível em: <https://www.popsci.com/scitech/article/2005-10/popscis-darpa-grand-challenge-preview-update-4/>. Acesso em: 8 novembro 2020.

26 ODUNLADE, EMMANUEL. Site Eletronics Lab. Getting started with ESP32. Disponível em: <https://www.electronics-lab.com/project/getting-started-esp32>. Acesso em: 8 novembro 2020.

27 BACKYARD BRAINS - **Experiment: Controlling the Claw**. Disponível em: <https://backyardbrains.com/experiments/musclespikershield_gripperhand>. Acesso em: 25 novembro 2020.

28 GRÖSSINGERR, SATO R. Magnetic Measurements: Quasistatic and ac, generating an electric current in properly placed sensing coils. Encyclopedia of Materials: Science and Technology, 2001. Disponível em: <https://www.sciencedirect.com/topics/engineering/magnetometer#:~:text=According%20to%20its%20physical%20effects,force%20are%20called%20magnetic%20magnetometers%3B.>. Acesso em: 18 abril 2021.

29 CARUSO, MICHAEL J. Applications of Magnetoresistive Sensors in Navigation Systems. Honeywell Inc.

30 HOW TO MECHATRONICS. MEMS Accelerometer Gyroscope Magnetometer & Arduino. Disponível em: < https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/>. Acesso em: 18 abril 2021.

31 ADARSH, S et al. Performance comparison of Infrared and Ultrasonic sensors for obstacles of different materials in vehicle/ robot navigation applications. 2016. IOP Conf. Series: Materials Science and Engineering 149 (2016) 012141 doi:10.1088/1757-899X/149/1/012141

32 GUIMARÃES, FÁBIO - Sensor de obstáculo infravermelho- 2018. Disponível em: <http://mundoprojetado.com.br/sensor-de-obstaculo-infravermelho/>. Acesso em: 21 abril 2021

33 THOMSEN, ADILSON - Como conectar o Sensor Ultrassônico HC-SR04 ao Arduino - 2011. Disponível em: <https://www.filipeflop.com/blog/sensor-ultrassonico-hc-sr04-ao-arduino/>. Acesso em: 21 abril 2021

34 RAFIQ, ARIF A. et al. Development of a Simple and Low-cost Smartphone Gimbal with MPU-6050 Sensor. 2020. Journal of Robotics and Control (JRC) ISSN: 2715-5072, DOI: 10.18196/jrc.1428.

35 FETICK & TOCKN. MPU6050 light library documentation. MIT License. 2020. Disponível em: <https://github.com/rfetick/MPU6050_light>. Acesso em: 21 abril 2021

36 YAN & GU. A review of rapid prototyping technologies and systems. Computer Aided Design v.28, n.4, pp. 307-318, 1996.

37 Correia.,A. P. (2007) Processador Embarcado em Lógica Reconfigurável para o Controle de Movimentação de Veículo de Passeio. Dissertação de Mestardo em Sistemas Mecatrônicos. ENM.DM-14ª/07, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 178p.

38 GILBERTO DE OLIVEIRA COSTA (2019) Identificação de Parâmetros Cinemáticos e Controle Dinâmico de Robôs Móveis com Rodas tipo Skid-Steering Utilizando Múltiplos Sensores Inerciais. Dissertação de Mestrado em Sistemas Mecatrônicos, Publicação xxx/2019, Departamento de Sistemas Mecatrônicos, Universidade de Brasília, Brasília, DF, 111p.

39 MAGALHÃES, Z. R. (2020). VEHICLE STABILITY CONTROLLER BASED ON MODEL PREDICTIVE CONTROL . Master Dissertation, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 92 p.

40 BADUE, C. et al. Instituto Federal do Espírito Santo. Self-driving cars: A survey. Expert Systems With Applications. v.165, 2021.

41 Mattede, H. Motor de corrente contínua, características e aplicações!. Site Mundo da Elétrica. Disponível em: < https://www.mundodaeletrica.com.br/motor-de-corrente-continua-caracteristicas-e-aplicacoes/ >. Acesso em: 05 de Maio de 2021.

42 How Accelerometer works? Interface ADXL335 with Arduino. Site Last Minute Engeneers. Disponível em: < https://lastminuteengineers.com/adxl335-accelerometer-arduino-tutorial/ >. Acesso em: 05 de Maio de 2021.

43 BERTONCELLO, S. D. Artigo – Controle PID: rompendo a barreira do tempo. 9 de Junho de 2020. Site Novus. Disponível em: < https://www.novus.com.br/blog/artigo-controle-pid-rompendo-a-barreira-do-tempo/ >. Acesso em: 06 de Maio de 2021.

44 LUCENA, T. V. L. Análise Dinâmica e Controle de Temperatura para um Recinto Refrigerado/ Nísio de Carvalho Lobo Brum. – Rio de Janeiro: UFRJ/Escola Politécnica, 2017. IX, 54p.: il.; 29,7 cm. Orientador: Nísio de Carvalho Lobo Brum Projeto de graduação –UFRJ/ Escola Politécnica/ Curso de Engenharia Mecânica, 2017.

45 OGATA, KATSUHIKO. Engenharia de controle moderno / Katsuhiko Ogata ; tradutora Heloísa Coimbra de Souza; revisor técnico Eduardo Aoun Tannuri. -- 5. ed. -- São Paulo: Pearson Prentice Hall, 2010.

46 NISE, NORMAN S. Engenharia de sistemas de controle / Norman S. Nise; tradução e revisão técnica Jackson Paul Matsuura. - 6. ed. - [Reimpr.]. - Rio de Janeiro: LTC, 2013. il.; 28 cm.

47 PAULA, A. Integração Numérica em Cálculo Numérico. Departamento de Ciência da Computação – UFMG. 2017. Slides de aula. 30p. Disponível em: < https://homepages.dcc.ufmg.br/~ana.coutosilva/Aulas-CN/Integra%e7%e3oNum%e9rica/Newton-CotesRepetida.pdf >. Acesso em: 08 de Maio de 2021.

APÊNDICES

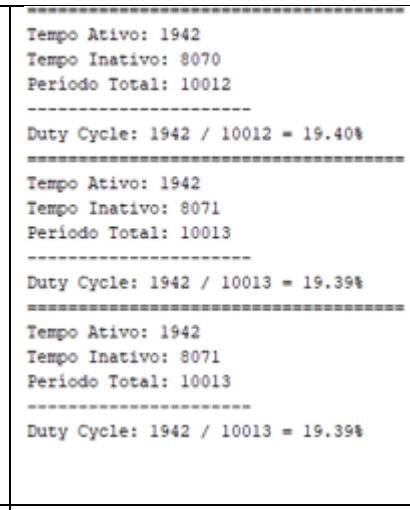
APÊNDICE A: Código para aferição de duty cycle

```
//=====================================================
// Código utilizado para a aferição do duty cycle da
// frequência PWM de saída do RRM
//=====================================================
//                Programa escrito por Lucas Koeler

const int PINO = 27;
unsigned long pwm_valor_ativo;
unsigned long pwm_valor_inativo;
unsigned long pwm_periodo;
float porcentagem_duty_cycle;

void setup() {
  pinMode(PINO, INPUT);
  Serial.begin(115200);
}

void loop() {
  pwm_valor_ativo = pulseIn(PINO, HIGH);
  pwm_valor_inativo = pulseIn(PINO, LOW);
  pwm_periodo = (pwm_valor_ativo + pwm_valor_inativo);
  porcentagem_duty_cycle = (pwm_valor_ativo / (float) pwm_periodo) * 100;

  Serial.println("====================================");
  Serial.print("Tempo Ativo: ");
  Serial.println(pwm_valor_ativo);
  Serial.print("Tempo Inativo: ");
  Serial.println(pwm_valor_inativo);
  Serial.print("Período Total: ");
  Serial.println(pwm_periodo);
  Serial.println("----------------------");
  Serial.print("Duty Cycle: ");
  Serial.print(pwm_valor_ativo);
  Serial.print(" / ");
  Serial.print(pwm_periodo);
```

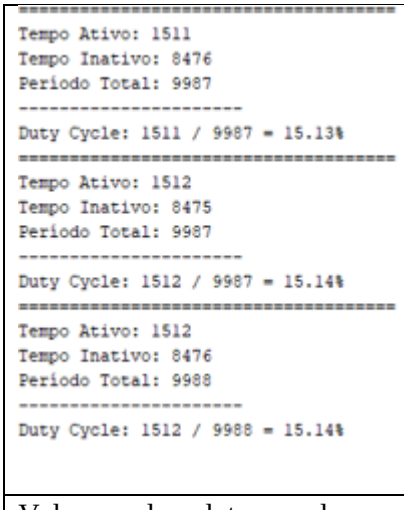
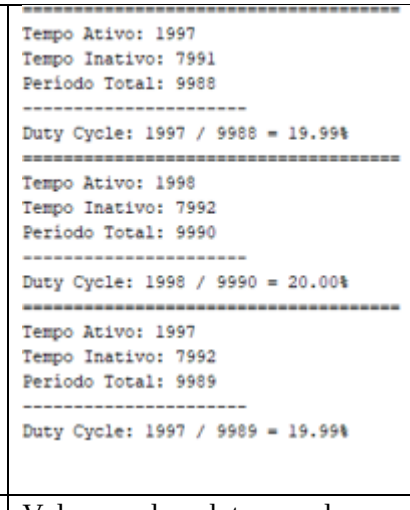
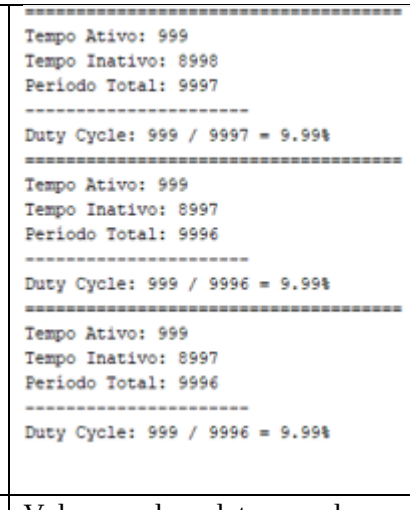
```
    Serial.print(" = ");
    Serial.print(porcentagem_duty_cycle);
    Serial.println("%");
    delay(500);
}
```

APÊNDICE B: Resultados das medições de períodos do sinal de PWM

Canal 1: Servo do subsistema de direção

		
Valores do duty cycle em microssegundos do subsistema de direção com o sinal de direção em posição neutra (“volante” reto).	Valores do duty cycle em microssegundos do subsistema de direção com o sinal de direção esterçado ao máximo para a esquerda.	Valores do duty cycle em microssegundos do subsistema de direção com o sinal de direção esterçado ao máximo para a direita.

Canal 2: Controle eletrônico de velocidade

		
Valores do duty cycle em microssegundos do subsistema do controle eletrônico de velocidade em posição neutra (parado).	Valores do duty cycle em microssegundos do subsistema do controle eletrônico de velocidade em posição de potência máxima para frente (“acelerador” no máximo).	Valores do duty cycle em microssegundos do subsistema do controle eletrônico de velocidade em posição de potência máxima para trás.

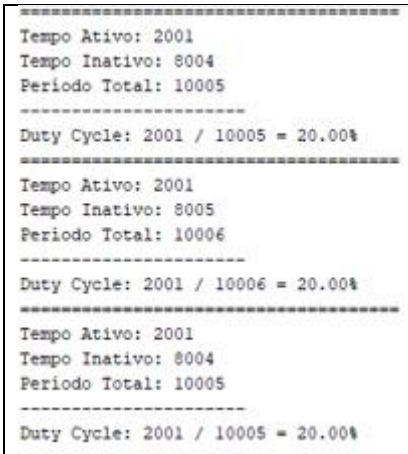
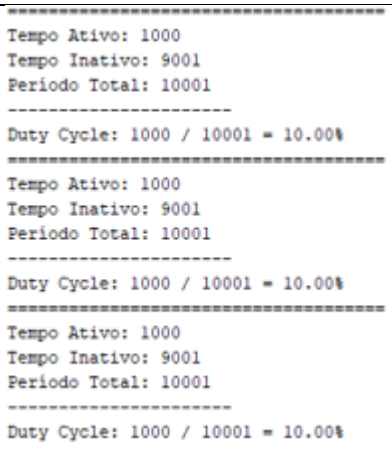
Canal 3: Servo de Transmissão de velocidade

``` ========== Tempo Ativo: 2002 Tempo Inativo: 8009 Período Total: 10011 ---------- Duty Cycle: 2002 / 10011 = 20.00% ========== Tempo Ativo: 2001 Tempo Inativo: 8009 Período Total: 10010 ---------- Duty Cycle: 2001 / 10010 = 19.99% ========== Tempo Ativo: 2001 Tempo Inativo: 8008 Período Total: 10009 ---------- Duty Cycle: 2001 / 10009 = 19.99% ```	``` ========== Tempo Ativo: 1000 Tempo Inativo: 9007 Período Total: 10007 ---------- Duty Cycle: 1000 / 10007 = 9.99% ========== Tempo Ativo: 1000 Tempo Inativo: 9008 Período Total: 10008 ---------- Duty Cycle: 1000 / 10008 = 9.99% ========== Tempo Ativo: 1000 Tempo Inativo: 9008 Período Total: 10008 ---------- Duty Cycle: 1000 / 10008 = 9.99% ```
Valores do duty cycle em microssegundos do servomotor de transmissão de velocidade em posição trativa (marcha de baixa velocidade e alto torque).	Valores do duty cycle em microssegundos do servomotor de transmissão de velocidade em posição de velocidade (marcha de alta velocidade e baixo torque).

Canal 4: Servo de controle do diferencial dianteiro

``` ========== Tempo Ativo: 1000 Tempo Inativo: 9006 Período Total: 10006 ---------- Duty Cycle: 1000 / 10006 = 9.99% ========== Tempo Ativo: 1000 Tempo Inativo: 9008 Período Total: 10008 ---------- Duty Cycle: 1000 / 10008 = 9.99% ========== Tempo Ativo: 1000 Tempo Inativo: 9008 Período Total: 10008 ---------- Duty Cycle: 1000 / 10008 = 9.99% ```	``` ========== Tempo Ativo: 2001 Tempo Inativo: 8005 Período Total: 10006 ---------- Duty Cycle: 2001 / 10006 = 20.00% ========== Tempo Ativo: 2001 Tempo Inativo: 8005 Período Total: 10006 ---------- Duty Cycle: 2001 / 10006 = 20.00% ========== Tempo Ativo: 2000 Tempo Inativo: 8004 Período Total: 10004 ---------- Duty Cycle: 2000 / 10004 = 19.99% ```
Valores do duty cycle em microssegundos do servomotor do diferencial dianteiro em posição travada (as rodas dianteiras giram sempre juntas).	Valores do duty cycle em microssegundos do servomotor do diferencial dianteiro em posição ativa (as rodas dianteiras não necessariamente giram juntas).

Canal 5: Servo de controle do diferencial traseiro

	
Valores do duty cycle em microssegundos do servomotor do diferencial traseiro em posição travada (as rodas traseiras giram sempre juntas).	Valores do duty cycle em microssegundos do servomotor do diferencial traseiro em posição ativa (as rodas dianteiras não necessariamente giram juntas).

APÊNDICE C: Teste de controle do servo do sistema de direção e do motor DC

```
//=====================================================
// Código utilizado para Teste de controle do servo do
// sistema de direção e do motor DC do automodelo
//=====================================================
//                    Programa escrito por Lucas Koeler

#include <Servo.h>
Servo Motor_DC, Servo_direcional; //Declarando o Servo e o ESC do motor DC

void setup(){
    //Serial.begin(115200);

    //Estabelecendo as portas para conexão dos componentes declarados
    Motor_DC.attach(13);
    Servo_direcional.attach(27);

//Iniciando o programa com os valores do Servo e do ESC do motor DC nas posições neutras
    Servo_direcional.writeMicroseconds(1451);
    Motor_DC.writeMicroseconds(1512);

//Acelerando o automodelo até sua velocidade máxima
    for(int duttyCycle = 1512; duttyCycle < 1942; duttyCycle+=10) {
        //Serial.println(duttyCycle);
        Motor_DC.writeMicroseconds(duttyCycle);
        delay(100);
    }
//Parando o automodelo
    Motor_DC.writeMicroseconds(1512);

//Mudando o servo da direção para a esquerda-direita e neutro
    Servo_direcional.writeMicroseconds(1001);
    delay(1000);
    Servo_direcional.writeMicroseconds(1942);
    delay(1000);
    Servo_direcional.writeMicroseconds(1451);
}

void loop() {
}
```

APÊNDICE D: Teste acurácia da medida de distância do automodelo

```
//================================================
// Código utilizado para Teste de
// acurácia da medida de distância
//================================================
//                   Programa escrito por Lucas Koeler

int Dhall = 23;
int leitura;
int A;
int B = 1;
int ContState = 0;

//==== Mapeamento dos pinos ======================================
//const byte sensorIR = 23;

//==== Declarando Variáveis ======================================
unsigned long Cont = 0;
unsigned long ContOld;
unsigned long REV = 0;        //número de revoluções
float RPS;                    //revoluções por segundo
float RPM;                    //revoluções por minuto
float mPs;                    //metros por segundo
float kmPh;                   //Km por hora
float m = 0;                  //Metros percorridos
int tempo;                    //Tempo em segundos
int tempo_antigo = 0;         //Tempo em segundos usado para fazer a função Time Interrupt
float tempo_de_Leitura = 500; //Tempo em que a função ISR está ativa contando as revoluções
unsigned long Time;
unsigned long timeOld;

//= Declarando Constantes de geometria do veículo:  ==================
float Diam_roda = 0.1106;     //Diâmetro da Roda em [m]
float razao_eixo_roda = 7.4181;   //Para cada 7.4181 rotações do eixo a roda gira 1 vez
float pi = 3.14159265;
float Circ_roda = 0.3474;    //Circ_roda = Diam_roda * pi; //Circunferência da roda


TaskHandle_t Task1, Task2;
SemaphoreHandle_t baton;

//=========================================================================
// Tarefa 1
//=========================================================================
```

```
void codeForTask1( void * parameter )
{
  for (;;) {
      leitura = digitalRead(Dhall);

    if(leitura == 1){
      A = 0;
    }
    else if(leitura == 0){
      A = 1;
     }

    if (B==0 && A==1){
      if(ContState==0){
        Cont++;
        ContState = 1;
      }
      else{
        ContState = 0;
        Cont++;
      }
    }
    B = A;
    delay(1);
  }
}

//=============================================================================
// Tarefa 2
//=============================================================================
void codeForTask2( void * parameter )
{
  for (;;) {

      REV = Cont - ContOld;
      RPS = (REV / tempo_de_Leitura)* 1000;
      RPM = RPS*60;
      mPs = RPS / razao_eixo_roda * Circ_roda;
      kmPh = mPs * 3.6;
      m = Cont * Circ_roda / razao_eixo_roda ;
      ContOld = Cont;

      Serial.println("==============");
      Serial.print("VOLTAS: ");
      Serial.println(Cont);
```

```
      Serial.print("REV: ");
      Serial.println(REV);
      Serial.print("RPS: ");
      Serial.println(RPS);
      Serial.print("RPM: ");
      Serial.println(RPM);
      Serial.print("m/s: ");
      Serial.println(mPs);
      Serial.print("km/h: ");
      Serial.println(kmPh);
      Serial.print("Dist: ");
      Serial.print(m);
      Serial.println(" m");

      delay(500);
  }
}

//================================================================================
// Código responsável por separar códigos de multitarefas que ocorrem simultaneamente
//================================================================================
void setup() {
  Serial.begin(115200);
  pinMode(Dhall, INPUT);

  baton = xSemaphoreCreateMutex();

  xTaskCreatePinnedToCore(
    codeForTask1,
    "Main Code",
    1000,
    NULL,
    1,
    &Task1,
    1);

  delay(500);  // para começar a tarefa1

  xTaskCreatePinnedToCore(
    codeForTask2,
    "Speed Readings",
    1000,
    NULL,
    1,
    &Task2,
    0);
```

```
  delay(500);  // para começar a tarefa2
}


void loop() {
}
```

APÊNDICE E: Teste de autoguiagem em linha reta

```
//=================================================
// Código utilizado para Teste de
// autoguiagem em linha reta
//=================================================
//                   Programa escrito por Lucas Koeler

#include <Servo.h>

int Dhall = 23;
int leitura;
int A;
int B = 1;
int ContState = 0;

//====  Mapeamento dos pinos  ========================================
const byte sensorIR = 2;
Servo motorDC, direcao, marcha, difFront, difTras;

//====  Declarando Variáveis  ========================================
unsigned long Cont = 0;
unsigned long ContOld;
unsigned long REV = 0;        //número de revoluções
float RPS;                    //revoluções por segundo
float RPM;                    //revoluções por minuto
float mPs;                    //metros por segundo
float kmPh;                   //Km por hora
float m = 0;                  //Metros percorridos
int tempo;                    //Tempo em segundos
int tempo_antigo = 0;         //Tempo em segundos usado para fazer a função Time Interrupt
float tempo_de_Leitura = 500;   //Tempo em que a função ISR está ativa contando as revoluções
unsigned long Time;
unsigned long timeOld;
int Motor_Ligado = false;


//====  Declarando Constantes de geometria do veículo:  ==============================================
float Diam_roda = 0.10448029;                 //Diâmetro da Roda em [m]
float razao_eixo_roda = 7.4181;          //Para cada 7.4181 rotações do eixo a roda gira 1 vez
float pi = 3.14159265;
float Circ_roda = 0.328234513;                  //Circ_roda = Diam_roda * pi; //Circunferência da roda: aprox. =
0.370697 [m] (pode mudar se mudar o diam.)
```

```
//====  Declarando Funções das tarefas RTOS:  ================================================
TaskHandle_t Task1, Task2, Task3;
SemaphoreHandle_t baton;

//=======================================================================================
// Tarefa 1 - Leitura do sensor de velocidade
//=======================================================================================

void codeForTask1( void * parameter )
{
  for (;;) {
      leitura = digitalRead(Dhall);

    if(leitura == 1){
      A = 0;
    }
    else if(leitura == 0){
      A = 1;
     }

    if (B==0 && A==1){
      if(ContState==0){
        Cont++;
        ContState = 1;
      }
      else{
        ContState = 0;
        Cont++;
      }
    }
    B = A;
    delay(1);
  }
}

//=======================================================================================
// Tarefa 2 - Cálculos de distância e outros sensores
//=======================================================================================
void codeForTask2( void * parameter )
{
  for (;;) {

    //Time = millis();
    //if(Time - timeOld >= tempo_de_Leitura){

      REV = Cont - ContOld;
```

```
      RPS = (REV / tempo_de_Leitura)* 1000;
      RPM = RPS*60;
      mPs = RPS / razao_eixo_roda * Circ_roda;
      kmPh = mPs * 3.6;
      m = Cont * Circ_roda / razao_eixo_roda ;
      ContOld = Cont;

      Serial.println("==============");
      Serial.print("Motor ligado: ");
      Serial.println(Motor_Ligado);
      Serial.print("VOLTAS: ");
      Serial.println(Cont);
      Serial.print("REV: ");
      Serial.println(REV);
      Serial.print("RPS: ");
      Serial.println(RPS);
      Serial.print("RPM: ");
      Serial.println(RPM);
      Serial.print("m/s: ");
      Serial.println(mPs);
      Serial.print("km/h: ");
      Serial.println(kmPh);
      Serial.print("Dist: ");
      Serial.print(m);
      Serial.println(" m");

      //timeOld = Time;
    //}

      delay(500);
  }
}
```

```
//=============================================================================
// Código responsável por separar códigos de multitarefas que ocorrem simultaneamente
// e fazer sutup inicial do carrinho
//=============================================================================
void setup() {
  Serial.begin(115200);
  pinMode(Dhall, INPUT);
```

```
  baton = xSemaphoreCreateMutex();

  xTaskCreatePinnedToCore(
    codeForTask1,
    "Main Code",
    1000,
    NULL,
    1,
    &Task1,
    1);

  delay(500);  // para iniciar a tarefa 1

  xTaskCreatePinnedToCore(
    codeForTask2,
    "Speed Readings",
    1000,
    NULL,
    1,
    &Task2,
    0);
  delay(500);  // para iniciar a tarefa 2


//====  Setando os Pinos  dos Servos ==================================
  motorDC.attach(27);   //fio AMARELO //27
  direcao.attach(33);   //fio LARANJA
  marcha.attach(32);    //fio VERDE
  difFront.attach(21);  //fio MARROM //21
  difTras.attach(22);   //fio ROXO

//====  Servos no neutro/Diferenciais destravados e Marcha Trativa =====
  //motorDC.writeMicroseconds(1451);
  direcao.writeMicroseconds(1512);
  marcha.writeMicroseconds(2001);
  difFront.writeMicroseconds(2001);
  difTras.writeMicroseconds(1000);

//====  Teste de direção, termina reta ================================
    direcao.writeMicroseconds(1451);    //Direção reta
        delay(1000);
        direcao.write(44);
        delay(500);
        direcao.write(139);
        delay(500);
        direcao.write(94);
```

```
        delay(1000);

//====  Teste de direção, termina reta [método com for]=================
//    direcao.writeMicroseconds(1451);    //Direção reta
//        for(int angleD=94; angleD <= 139; angleD++) {
//            //Serial.println(angleD);
//             direcao.write(angleD);
//            delay(100);
//        }
//
//      direcao.write(139);
//      delay(5000);
//
//      for(int angleD=139; angleD <44; angleD--) {
//          //Serial.println(angleD);
//          direcao.write(angleD);
//          delay(100);
//      }
//      direcao.write(44);
//
//      for(int angleD=44; angleD <88; angleD--) {
//          //Serial.println(angleD);
//          direcao.write(angleD);
//          delay(100);
//      }
//      direcao.write(88);


//====  Liga o motor ================================================
    delay(3000);
    Motor_Ligado = true;

//                                      motorDC.writeMicroseconds(1512);    //Motor parado
//                                      for(int angleT=94; angleT <= 106; angleT++) {
//                                          //Serial.println(angleT);
//                                          motorDC.write(angleT);
//                                          delay(100);
//                                      }
//
//                                        motorDC.write(106);
//                                        delay(5000);
//
//                                      for(int angleT=106; angleT <94; angleT--) {
//                                        //Serial.println(angleT);
//                                        motorDC.write(angleT);
//                                        delay(100);
```

```
//                                        }
//                                        motorDC.write(94);
}

//=========================================================================================
// Tarefa 3 - movimentação do motor
//=========================================================================================
void loop() {
      while(Motor_Ligado == true){
          while(m <= 5){
                                        motorDC.attach(27);
                                        motorDC.write(104);
                                        delay(1);
          }
          Motor_Ligado = false;
          motorDC.write(94);
      }
}
```

APÊNDICE F: Teste de funcionamento do magnetômetro

```
//=====================================================
// Código utilizado para Teste de funcionamento
// do magnetômetro
//=====================================================
//                    Programa escrito por Lucas Koeler

#include "Wire.h"
#include "I2Cdev.h"
#include "HMC5883L.h"

HMC5883L mag;

int16_t mx, my, mz;

void setup() {
    Wire.begin();
    Serial.begin(38400);

    // inicializa o magnetômetro
    Serial.println("Inicializando o I2C e magnetômetro...");
    mag.initialize();

    // verifica a conexão
    Serial.println("Testando conexões...");
    Serial.println(mag.testConnection() ? "HMC5883L conexão bem sucedida" : "HMC5883L connecxão falhou!");

    //Código para ativar o magnetônetro junto com o acelerômetro no módulo
    //Bypass Mode
    Wire.beginTransmission(0x68);
    Wire.write(0x37);
    Wire.write(0x02);
    Wire.endTransmission();

    Wire.beginTransmission(0x68);
    Wire.write(0x6A);
    Wire.write(0x00);
    Wire.endTransmission();
    //Disable Sleep Mode
    Wire.beginTransmission(0x68);
    Wire.write(0x6B);
    Wire.write(0x00);
    Wire.endTransmission();
}
```

```
void loop() {
    // leitura da direção crua
    mag.getHeading(&mx, &my, &mz);

    // mostrar dados separados por tab: valores do mag. x/y/z
    //Serial.print("mag:¥t");
    //Serial.print(mx); Serial.print("¥t");
    //Serial.print(my); Serial.print("¥t");
    //Serial.print(mz); Serial.print("¥t");

// Para calcular a direção. Grau zero = Norte Magnético
    float heading = atan2(my, mx);
    if(heading < 0)
      heading += 2 * M_PI;
    //Serial.print("Direção:¥t");
    Serial.println(heading * 180/M_PI);
    delay(500);
}
```

APÊNDICE G: Teste de funcionamento do giroscópio

```
//=====================================================
// Código utilizado para Teste de funcionamento do
// giroscópio com integração de velocidade angular
//=====================================================
//                    Programa escrito por Lucas Koeler
// Adaptado do programa de exemplo da biblioteca <MPU6050_light.h>

#include "Wire.h"
#include <MPU6050_light.h>

MPU6050 mpu(Wire);
unsigned long timer = 0;

void setup() {
  Serial.begin(230400);
  Wire.begin();

  byte status = mpu.begin();
  Serial.print(F("O Status do MPU6050 é: "));
  Serial.println(status);
  while(status!=0){ } // para o programa se não puder conectar ao MPU6050

  Serial.println(F("Calculando os offsets, NÃO MOVER O SENSOR! MPU6050"));
  delay(1000);
  mpu.setGyroConfig(3);
  mpu.calcOffsets(); // gyro e accelero
  Serial.println("Done!¥n");
}

void loop() {
  mpu.update();

  if((millis()-timer)>500){ // serial print a cada meio segundo
        //Serial.print("X : ");
        //Serial.print(mpu.getAngleX());
        //Serial.print("Y : ");
        //Serial.print(mpu.getAngleY());
        //Serial.print("¥tZ : ");
        Serial.println(mpu.getAngleZ());
        timer = millis();
  }
}
```

APÊNDICE H: Teste de performance: magnetômetro Vs. giroscópio

```
//=================================================
// Código utilizado para Teste de Performance
// Giroscópio VS. Magnetômetro
//=================================================
//                    Programa escrito por Lucas Koeler

#include <Servo.h>
#include "Wire.h"
#include "I2Cdev.h"
#include "HMC5883L.h"
#include <math.h>
#include <MPU6050_light.h>

int milliNew;
float t_i;
float t_f;
float t_corrido;

//==== Variáveis para calcular o theta ================================
float theta_ZERO;
float theta_i;
float theta_f;
float dt_Gyro;

//==== INICIANDO O MPU 6050 ===========================================
MPU6050 mpu(Wire);
unsigned long timer = 0;

//====  Sistema do IMU - Magnetômetro  ================================================================
HMC5883L mag;                         //Inicializa o Magnetômetro
int16_t mx, my, mz;
float heading;
float GYRO_Z_R;

//====  Variáveis do Magnetômetro  ====================================================================
int16_t mx1, my1, mz1,  mx2, my2, mz2,  mx3, my3, mz3,  mx4, my4, mz4;
float heading1,heading2,heading3,heading4,headingM ;
float headingR;

//=================================================================================
//  SETUP                                                                        =
//=================================================================================
void setup(){
```

```
delay(3000);
//==== Iniciando serial + I2C ========================================
  Serial.begin(230400);
  Wire.begin();



    //inicializa o magnetômetro
    mag.initialize();

    //===  Ativando o magnetômetro no HMC5883L e deativando o modo sleep  ===
    Wire.beginTransmission(0x68);
    Wire.write(0x37);
    Wire.write(0x02);
    Wire.endTransmission();
    Wire.beginTransmission(0x68);
    Wire.write(0x6A);
    Wire.write(0x00);
    Wire.endTransmission();
    //Disable Sleep Mode
    Wire.beginTransmission(0x68);
    Wire.write(0x6B);
    Wire.write(0x00);
    Wire.endTransmission();

    delay(1000);


//====  Extrai a média da leitura do Magnetômetro e chama de zero =======
//functionRead_MAG_HEADING
     mag.getHeading(&mx1, &my1, &mz1);
     delay(100);
     mag.getHeading(&mx2, &my2, &mz2);
     delay(100);
     mag.getHeading(&mx3, &my3, &mz3);
     delay(100);
     mag.getHeading(&mx4, &my4, &mz4);
     delay(100);

     heading1 = atan2(my1, mx1);
     if(heading1 < 0){
     heading1 += 2 * M_PI;
     }
     heading2 = atan2(my2, mx2);
     if(heading2 < 0){
     heading2 += 2 * M_PI;
     }
```

```
     heading3 = atan2(my3, mx3);
     if(heading3 < 0){
     heading3 += 2 * M_PI;
     }
     heading4 = atan2(my4, mx4);
     if(heading4 < 0){
     heading4 += 2 * M_PI;
     }
     headingM = (heading1+heading2+heading3+heading4)/4;

    delay(3000);



//==== Verificando MPU ================================================
  byte status = mpu.begin();

  while(status!=0){ } // stop everything if could not connect to MPU6050

  delay(1000);
  mpu.setGyroConfig(3);
  delay(1000);
  mpu.calcOffsets(); // gyro and accelero
  //Serial.println("Done!¥n");
  //delay(1000);

    milliNew=millis();
}



//=============================================================================
//  VOID LOOP (dps do setup)
//=============================================================================
void loop() {

//código para pegar o dt em segundos que será utilizado para integrar a velocidade
t_f = t_i;
t_i = millis();
dt_Gyro = t_i-t_f;
t_corrido = t_corrido + dt_Gyro;

// read raw heading measurements from device
    mag.getHeading(&mx, &my, &mz);
// To calculate heading in degrees. 0 degree indicates North
    float heading = atan2(my, mx);
```

```
    if(heading < 0){
       heading += 2 * M_PI;
    }

    //Cálculos do Magnetômetro
    heading = atan2(my, mx);
    if(heading < 0){
       heading += 2 * M_PI;
    }
    headingR = ((heading * 180/M_PI) - (headingM * 180/M_PI))*-1;

    //Direção do Magnetômetro
    //Serial.print("headingR: ");
    //Serial.print(headingR);
    //Serial.print("¥t");
    //Serial.print("¥t");
    //Serial.println("º          ");


//==== Verificando MPU ================================================
  mpu.update();

  if((millis()-timer)>5){ // print data every 5ms
      GYRO_Z_R = mpu.getAngleZ();
      //Serial.print(headingR);
      Serial.print(headingR);
      Serial.print("  ");
      //Serial.print("Z: ");
      Serial.print(GYRO_Z_R - 15);
      Serial.print("  ");
      //Serial.print("tempo: ");
      Serial.println((millis()-milliNew)/1000.0);
      //Serial.println("¥t");

      timer = millis();
  }
}


//=====================================================
// Código utilizado para Teste de Performance
// Giroscópio VS. Magnetômetro [CONTROLE DO SERVO]
//=====================================================
//                   Programa escrito por Lucas Koeler

#include <Servo.h>
```

```
Servo meuServo;
unsigned long tempoDeSetup;
unsigned long tempoCorrido;
unsigned long tempoLimite = 180000; // 1min=60000, 20min=1200000, 60min=3600000
int delayServo = 60; //150(lento) e 75 rápido e 110(media boa)
int incremento = 15;
int angulo;

void setup() {
  meuServo.attach(5);
  meuServo.write(90);
  delay(3000);
  Serial.begin(115200);
  tempoCorrido = millis();
  tempoDeSetup = millis();
}

void loop() {
  Serial.print("================COMEÇOU!=========================");
  while (((tempoCorrido - tempoDeSetup) <= tempoLimite)){
      for(angulo = 90; angulo <=180; angulo=angulo+incremento){
          meuServo.write(angulo);
          //Serial.print("Ângulo: ");
          Serial.print(angulo);
          Serial.print("  ");
          Serial.println((tempoCorrido-tempoDeSetup)/1000);
          delay(delayServo);
      }
      for(angulo = 180; angulo >=90; angulo=angulo-incremento){
          meuServo.write(angulo);
          Serial.print(angulo);
          Serial.print("  ");
          Serial.println((tempoCorrido-tempoDeSetup)/1000);
          delay(delayServo);
      }
      for(angulo = 90; angulo <=180; angulo=angulo+incremento){
          meuServo.write(angulo);
          Serial.print(angulo);
          Serial.print("  ");
          Serial.println((tempoCorrido-tempoDeSetup)/1000);
          delay(delayServo);
      }
      for(angulo = 180; angulo >=0; angulo=angulo-incremento){
          meuServo.write(angulo);
          Serial.print(angulo);
```

```
            Serial.print("  ");
            Serial.println((tempoCorrido-tempoDeSetup)/1000);
            delay(90);
        }
        for(angulo = 0; angulo >=90; angulo=angulo+incremento){
            meuServo.write(angulo);
            Serial.print(angulo);
            Serial.print("  ");
            Serial.println((tempoCorrido-tempoDeSetup)/1000);
            delay(delayServo);
        }
        tempoCorrido= millis();
    }
    meuServo.write(90);
    Serial.print("90");
    Serial.print("  ");
    Serial.println(tempoCorrido-tempoDeSetup);
    Serial.println("=========================ACABOU!==========================");
    Serial.end();
}
```

APÊNDICE I: Teste de PID em plataforma giratória

```
//================================================
// Código utilizado para Teste de funcionamento
// de sistema PID com um Magnetômetro
//================================================
//                   Programa escrito por Lucas Koeler

#include "Wire.h"
#include "I2Cdev.h"
#include "HMC5883L.h"
#include <Servo.h>
#include <math.h>

Servo pitchServo;
int pinServoX = 33;

//====  Sistema do IMU - Magnetômetro  ============================================================
HMC5883L mag;                         //Inicializa o Magnetômetro
int16_t mx, my, mz;
float heading;

//====  Variáveis do Magnetômetro  ================================================================
int16_t mx1, my1, mz1,  mx2, my2, mz2,  mx3, my3, mz3,  mx4, my4, mz4;
float heading1,heading2,heading3,heading4,headingM ;
float headingR;


float k1=1.0;
float k2=50;
float k3=.00001;

int milliOld;
int milliNew;
int dt;

float pitchTarget=0;
float pitchActual;
float pitchError=0;
float pitchErrorOld;
float pitchErrorChange;
float pitchErrorSlope=0;
float pitchErrorArea=0;
float pitchServoVal=90;

//=============================================================================================================
//=============================================================================================================
void setup() {

    Wire.begin();
    Serial.begin(9600);

    // initialize device
    Serial.println("Initializing I2C devices...");
    mag.initialize();

    // verify connection
    Serial.println("Testing device connections...");
    Serial.println(mag.testConnection() ? "HMC5883L connection successful" : "HMC5883L connection failed");

    //===  Ativando o magnetômetro no HMC5883L  ===
    Wire.beginTransmission(0x68);
    Wire.write(0x37);
    Wire.write(0x02);
    Wire.endTransmission();
```

```
    Wire.beginTransmission(0x68);
    Wire.write(0x6A);
    Wire.write(0x00);
    Wire.endTransmission();

    //Disable Sleep Mode
    Wire.beginTransmission(0x68);
    Wire.write(0x6B);
    Wire.write(0x00);
    Wire.endTransmission();

    delay(2000);

    pitchServo.attach(pinServoX);
    pitchServo.write(90);
    delay(20);


//====  Extrai a média da leitura do Magnetômetro e chama de zero =======
//functionRead_MAG_HEADING
     mag.getHeading(&mx1, &my1, &mz1);
     delay(100);
     mag.getHeading(&mx2, &my2, &mz2);
     delay(100);
     mag.getHeading(&mx3, &my3, &mz3);
     delay(100);
     mag.getHeading(&mx4, &my4, &mz4);
     delay(100);

     heading1 = atan2(my1, mx1);
     if(heading1 < 0){
     heading1 += 2 * M_PI;
     }
     heading2 = atan2(my2, mx2);
     if(heading2 < 0){
     heading2 += 2 * M_PI;
     }
     heading3 = atan2(my3, mx3);
     if(heading3 < 0){
     heading3 += 2 * M_PI;
     }
     heading4 = atan2(my4, mx4);
     if(heading4 < 0){
     heading4 += 2 * M_PI;
     }
     headingM = (heading1+heading2+heading3+heading4)/4;

    delay(3000);

    milliNew=millis();
}

//=============================================================================================================
//=============================================================================================================
void loop() {
    // read raw heading measurements from device
    mag.getHeading(&mx, &my, &mz);

//    display tab-separated gyro x/y/z values
//    Serial.print("mag:¥t");
//    Serial.print(mx); Serial.print("¥t");
//    Serial.print(my); Serial.print("¥t");
//    Serial.print(mz); Serial.print("¥t");

// To calculate heading in degrees. 0 degree indicates North
    float heading = atan2(my, mx);
    if(heading < 0){
       heading += 2 * M_PI;
```

```
    }

    //Cálculos do Magnetômetro
    heading = atan2(my, mx);
    if(heading < 0){
       heading += 2 * M_PI;
    }
    headingR = (heading * 180/M_PI) - (headingM * 180/M_PI);

    //Direção do Magnetômetro
    Serial.print("headingR: ");
    Serial.print(headingR);
    Serial.print("º          ");
    //Serial.print("headingM: ");
    //Serial.print(headingM * 180/M_PI);
    //Serial.println("º");


//============================================== MEXER
    //pitchActual=asin(2*(q0*q2-q3*q1));//aqui
    //pitchActual=pitchActual/(2*3.141592654)*360;//aqui
    pitchActual = headingR;


milliOld=milliNew;
milliNew=millis();
dt=milliNew-milliOld;

pitchErrorOld=pitchError;
pitchError=pitchTarget-pitchActual;
pitchErrorChange=pitchError-pitchErrorOld;
pitchErrorSlope=pitchErrorChange/dt;
pitchErrorArea=pitchErrorArea+pitchError*dt;

pitchServoVal=pitchServoVal+k1*pitchError+k2*pitchErrorSlope+k3*pitchErrorArea;
//pitchServo.write(pitchServoVal);

pitchServo.write(180-pitchServoVal);


//Serial.print(pitchTarget);
//Serial.print(",");
//Serial.print(pitchActual);
//Serial.println(",");

//delay(300);//tirar

    Serial.print("pitchServoVal: ");
    Serial.print(180-pitchServoVal);
    Serial.println("º");

}
```

APÊNDICE J: Teste de linha reta no veículo guiado por PID utilizando um magnetômetro

```
//=====================================================
// Código utilizado para Teste de linha reta no veículo guiado
// por PID utilizando um magnetômetro
//=====================================================
//                   Programa escrito por Lucas Koeler

//====  Bibliotecas Utilizadas  ========================================
#include <Servo.h>
#include "Wire.h"
#include "I2Cdev.h"
#include "HMC5883L.h"
#include <math.h>

//====  Sistema de Leitura de Velocidade e Odômetro  ===================
int Dhall = 23; // fio Laranja
int leitura;
int A;
int B = 1;
int ContState = 0;
int pos;

//====  Mapeamento dos pinos  ==========================================
const byte sensorIR = 2;
Servo motorDC, direcao, marcha, difFront, difTras;

//====  Declarando Variáveis  ==========================================
unsigned long Cont = 0;
unsigned long ContOld;
unsigned long REV = 0;          //número de revoluções
float RPS;                      //revoluções por segundo
float RPM;                      //revoluções por minuto
float mPs;                      //metros por segundo
float kmPh;                     //Km por hora
float m = 0;                    //Metros percorridos
int tempo;                      //Tempo em segundos
int tempo_antigo = 0;           //Tempo em segundos usado para fazer a função Time Interrupt
float tempo_de_Leitura = 500;   //Tempo em que a função ISR está ativa contando as revoluções
unsigned long Time;
unsigned long timeOld;
int Motor_Ligado = false;


//====  Declarando Constantes de geometria do veículo:  ================
```

```
float Diam_roda = 0.10448029;         //Diâmetro da Roda em [m]
float razao_eixo_roda = 7.4181;       //Para cada 7.4181 rotações do eixo a roda gira 1 vez
float pi = 3.14159265;
float Circ_roda = 0.328234513;        //Circ_roda = Diam_roda * pi; //Circunferência da roda: aprox. = 0.370697 [m]
(pode mudar se mudar o diam.)


//====  Sistema do IMU - Magnetômetro  ============================================================
HMC5883L mag;                         //Inicializa o Magnetômetro
int16_t mx, my, mz;
float heading;

//====  Variáveis do Magnetômetro  ================================================================
int16_t mx1, my1, mz1,  mx2, my2, mz2,  mx3, my3, mz3,  mx4, my4, mz4;
float heading1,heading2,heading3,heading4,headingM ;
float headingR;

//====  Variáveis do sistema PID da direção  ======================================================
float k1=0.02; //0.01
float k2=50; // 25 ou 20 dá um bom resultado 0.02x50x0.00001
float k3=.00001; //0.00001
int milliOld;
int milliNew;
int dt;
float pitchTarget=0;
float pitchActual;
float pitchError=0;
float pitchErrorOld;
float pitchErrorChange;
float pitchErrorSlope=0;
float pitchErrorArea=0;
float pitchServoVal=90;

//====  Declarando Funções das tarefas RTOS:  =====================================================
TaskHandle_t Task1, Task2, Task3;
SemaphoreHandle_t baton;


//===============================================================================
// LOOP #1 (Leitura de Mag, Conta as medições do sensor IR de velocidadeX, calcula o magnetômetro)
//===============================================================================
void codeForTask1( void * parameter )
{
  for (;;) {

//    //functionRead_MAG_HEADING
```

```
//    mag.getHeading(&mx, &my, &mz);
//
//    //Cálculos do Magnetômetro
//    heading = atan2(my, mx);
//    if(heading < 0){
//      heading += 2 * M_PI;
//    }
//    headingR = (heading * 180/M_PI) - (headingM * 180/M_PI);
//    //Cálculos do Magnetômetro com PID
//    pitchActual = headingR;
//      milliOld=milliNew;
//      milliNew=millis();
//      dt=milliNew-milliOld;
//    pitchErrorOld=pitchError;
//      pitchError=pitchTarget-pitchActual;
//      pitchErrorChange=pitchError-pitchErrorOld;
//      pitchErrorSlope=pitchErrorChange/dt;
//      pitchErrorArea=pitchErrorArea+pitchError*dt;
//    pitchServoVal=pitchServoVal+k1*pitchError+k2*pitchErrorSlope+k3*pitchErrorArea;
//      //pitchServo.write(180-pitchServoVal);


//functionRead_X_Speed()
    leitura = digitalRead(Dhall);
//Contagem das leituras para contar a velocidade
    if(leitura == 1){
      A = 0;
    }
    else if(leitura == 0){
      A = 1;
     }

    if (B==0 && A==1){
      if(ContState==0){
        Cont++;
        ContState = 1;
      }
      else{
        ContState = 0;
        Cont++;
      }
    }
    B = A;
    delay(1);
  }
```

```
}

//================================================================================
// LOOP #2
//================================================================================
void codeForTask2( void * parameter )
{
  for (;;) {

      //Time = millis();
      //if(Time - timeOld >= tempo_de_Leitura){

      //Cálculos do Odômetro
      REV = Cont - ContOld;
      RPS = (REV / tempo_de_Leitura)* 1000;
      RPM = RPS*60;
      mPs = RPS / razao_eixo_roda * Circ_roda;
      kmPh = mPs * 3.6;
      m = Cont * Circ_roda / razao_eixo_roda ;
      ContOld = Cont;

      //Dados do Odômetro
      Serial.println("===============");
      Serial.print("Motor ligado: ");
      Serial.println(Motor_Ligado);
      Serial.print("VOLTAS: ");
      Serial.println(Cont);
      Serial.print("REV: ");
      Serial.println(REV);
      Serial.print("RPS: ");
      Serial.println(RPS);
      Serial.print("RPM: ");
      Serial.println(RPM);
      Serial.print("m/s: ");
      Serial.println(mPs);
      Serial.print("km/h: ");
      Serial.println(kmPh);
      Serial.print("Dist: ");
      Serial.print(m);
      Serial.println(" m");
      //Direção do Magnetômetro
      Serial.print("headingR: ");
      Serial.print(headingR);
      Serial.println("º");
      Serial.print("headingM: ");
```

```
      Serial.print(headingM * 180/M_PI);
      Serial.println("º");
      Serial.print("pitchServoVal: ");
      Serial.print(180-pitchServoVal);
      Serial.println("º");
      Serial.print("                    ");
      Serial.println(millis());


      //timeOld = Time;
    //}
      delay(500);
      //vTaskDelay(500);//Tempo em que a função está inativa_se mexer aqui, a velocidade muda
  }
}



//=============================================================================
//  SETUP
//=============================================================================
void setup() {
//====  Setup do FreeRTOs  ========================================
  baton = xSemaphoreCreateMutex();
  xTaskCreatePinnedToCore(
    codeForTask1,
    "Main Code",
    1000,
    NULL,
    1,
    &Task1,
    1);
  delay(500);  // needed to start-up task1
  //
  xTaskCreatePinnedToCore(
    codeForTask2,
    "Speed Readings",
    1000,
    NULL,
    1,
    &Task2,
    0);
  delay(500);  // needed to start-up task2
  ////////TÁ CANCELANDO A TASK3
//  xTaskCreatePinnedToCore(
//    codeForTask2,
```

```
//    "Magnetometer readings",
//    1000,
//    NULL,
//    0,
//    &Task3,
//    1);
//  delay(500);  // needed to start-up task3

//====  Setup do Magnetômetro  ========================================
  // inicializa a biblioteca para comunicação I2C
  Wire.begin();
  // inicializa o Magnetômetro
  Serial.println("Initializing I2C devices...");
  mag.initialize();
  // verify connection
  Serial.println("Testing device connections...");
  Serial.println(mag.testConnection() ? "HMC5883L connection successful" : "HMC5883L connection failed");
  //Ativando o magnetômetro no HMC5883L
  Wire.beginTransmission(0x68);
  Wire.write(0x37);
  Wire.write(0x02);
  Wire.endTransmission();
  //
  Wire.beginTransmission(0x68);
  Wire.write(0x6A);
  Wire.write(0x00);
  Wire.endTransmission();
  //Desabilitando o Sleep Mode
  Wire.beginTransmission(0x68);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission();


//====Inicia Serial e seta o pino do velocímetro/odometro==============
  Serial.begin(115200);
  pinMode(Dhall, INPUT);

//====  Setando os Pinos  dos Servos ===================================
  motorDC.attach(27);   //fio AMARELO //27
  direcao.attach(33);   //fio LARANJA
  marcha.attach(32);    //fio VERDE
  difFront.attach(19);  //fio ROXO //21
  difTras.attach(18);   //fio MARROM

//====  Servos no neutro/Diferenciais destravados e Marcha Trativa =====
```

```
  motorDC.writeMicroseconds(1451);
  direcao.writeMicroseconds(1512);
  marcha.writeMicroseconds(2001);
  difFront.writeMicroseconds(2001);
  difTras.writeMicroseconds(1000);

//====  Teste de direção, termina reta ================================
    direcao.writeMicroseconds(1451);    //Direção reta
        delay(1000);
        direcao.write(44);
        delay(500);
        direcao.write(139);
        delay(500);
        direcao.write(90);
        delay(1000);

//====  Teste de direção, termina reta [método com for]=================
//    direcao.writeMicroseconds(1451);    //Direção reta
//        for(int angleD=94; angleD <= 139; angleD++) {
//            //Serial.println(angleD);
//             direcao.write(angleD);
//            delay(100);
//        }
//
//      direcao.write(139);
//      delay(5000);
//
//      for(int angleD=139; angleD <44; angleD--) {
//          //Serial.println(angleD);
//          direcao.write(angleD);
//          delay(100);
//      }
//      direcao.write(44);
//
//      for(int angleD=44; angleD <88; angleD--) {
//          //Serial.println(angleD);
//          direcao.write(angleD);
//          delay(100);
//      }
//      direcao.write(88);


//====  Armando o ESC do Motor DC ======================================
   for (int angleT = 90; angleT <= 100; angleT++) {
      motorDC.write(angleT);
      delay(150);
```

```
   }
  motorDC.writeMicroseconds(1512);    //Motor parado

//====  Liga o motor ================================================
    delay(3000);
    Motor_Ligado = true;

//                                    motorDC.attach(27);   //fio AMARELO //27
//                                    motorDC.writeMicroseconds(1512);    //Motor parado
//                                    for(int angleT=94; angleT <= 126; angleT++) {
//                                          //Serial.println(angleT);
//                                          motorDC.write(angleT);
//                                          delay(100);
//                                      }
//
//                                        motorDC.write(126);
//                                        delay(1000);
//
//                                      for(int angleT=126; angleT <94; angleT--) {
//                                        //Serial.println(angleT);
//                                        motorDC.write(angleT);
//                                        delay(100);
//                                      }
//                                      motorDC.write(94);



//====  Reseta o Odômetro ============================================
    ContOld = 0;
    Cont = 0;

//====  Extrai a média da leitura do Magnetômetro e chama de zero =======
//functionRead_MAG_HEADING
     mag.getHeading(&mx1, &my1, &mz1);
     delay(100);
     mag.getHeading(&mx2, &my2, &mz2);
     delay(100);
     mag.getHeading(&mx3, &my3, &mz3);
     delay(100);
     mag.getHeading(&mx4, &my4, &mz4);
     delay(100);

     heading1 = atan2(my1, mx1);
     if(heading1 < 0){
     heading1 += 2 * M_PI;
     }
```

```
      heading2 = atan2(my2, mx2);
      if(heading2 < 0){
      heading2 += 2 * M_PI;
      }
      heading3 = atan2(my3, mx3);
      if(heading3 < 0){
      heading3 += 2 * M_PI;
      }
      heading4 = atan2(my4, mx4);
      if(heading4 < 0){
      heading4 += 2 * M_PI;
      }
      headingM = (heading1+heading2+heading3+heading4)/4;

    delay(3000);

    milliNew=millis();
}

//================================================================================
// VOID LOOP (dps do setup)
//================================================================================
void loop() {
       while(Motor_Ligado == true){
           while(m <= 5){
              motorDC.attach(27);
              motorDC.write(104);

    //functionRead_MAG_HEADING
    mag.getHeading(&mx, &my, &mz);

    //Cálculos do Magnetômetro
    heading = atan2(my, mx);
    if(heading < 0){
      heading += 2 * M_PI;
    }
    headingR = (heading * 180/M_PI) - (headingM * 180/M_PI);


    //Cálculos do Magnetômetro com PID
    pitchActual = headingR;

    milliOld=milliNew;
    milliNew=millis();
    dt=milliNew-milliOld;
```

```
    pitchErrorOld=pitchError;

    pitchError=pitchTarget-pitchActual;
    pitchErrorChange=pitchError-pitchErrorOld;
    pitchErrorSlope=pitchErrorChange/dt;
    pitchErrorArea=pitchErrorArea+pitchError*dt;

    pitchServoVal=pitchServoVal + k1*pitchError + k2*pitchErrorSlope + k3*pitchErrorArea;



    Serial.print("                                                pitchServoVal: ");
    Serial.print(pitchServoVal);
    Serial.println("º");
    direcao.write(pitchServoVal);
          delay(2);


          }
          Motor_Ligado = false;
          motorDC.write(94);
      }
          motorDC.detach();
}
```

APÊNDICE K: Teste de implementação de código PID para velocidade constante.

```
//=====================================================
// Código utilizado para teste trajetória em linha reta
// utilizando PID para controle de velocidade de direção
//=====================================================
//                   Programa escrito por Lucas Koeler

//====  Bibliotecas Utilizadas  ========================================
#include <Servo.h>
#include "Wire.h"
#include "I2Cdev.h"
#include "HMC5883L.h"
#include <math.h>

//====  Sistema de Leitura de Velocidade e Odômetro  ===================
int Dhall = 23; // fio Laranja
int leitura;
int A;
int B = 1;
int ContState = 0;
int pos;

//====  Mapeamento dos pinos  ==========================================
const byte sensorIR = 2;
Servo motorDC, direcao, marcha, difFront, difTras;

//====  Declarando Variáveis  ==========================================
unsigned long Cont = 0;
unsigned long ContOld;
unsigned long REV = 0;          //número de revoluções
float RPS = 0;                      //revoluções por segundo
float RPM = 0;                      //revoluções por minuto
float mPs = 0;                      //metros por segundo
float kmPh = 0;                     //Km por hora
float m = 0;                    //Metros percorridos
int tempo;                      //Tempo em segundos
int tempo_antigo = 0;           //Tempo em segundos usado para fazer a função Time Interrupt
float tempo_de_Leitura;     //Tempo em que a função ISR está ativa contando as revoluções
unsigned long Time;
unsigned long timeOld;
float Vlon = 0;
float REVs = 0;
int Motor_Ligado = false;
```

```
//====  Declarando Constantes de geometria do veículo:  ========================================
float Diam_roda = 0.10448029;        //Diâmetro da Roda em [m]
float razao_eixo_roda = 7.4181;      //Para cada 7.4181 rotações do eixo a roda gira 1 vez
float pi = 3.14159265;
float Circ_roda = 0.328234513;       //Circ_roda = Diam_roda * pi; //Circunferência da roda: aprox. = 0.370697 [m]
(pode mudar se mudar o diam.)


//====  Sistema do IMU - Magnetômetro  ============================================================
HMC5883L mag;                        //Inicializa o Magnetômetro
int16_t mx, my, mz;
float heading;

//====  Variáveis do Magnetômetro  ================================================================
int16_t mx1, my1, mz1,  mx2, my2, mz2,  mx3, my3, mz3,  mx4, my4, mz4;
float heading1,heading2,heading3,heading4,headingM ;
float headingR;

//====   Variáveis do sistema PID da direção  ================================================== antes era
0.02x50x0.00001
float k1=0.05; //0.01        //0.05
float k2=60; // 10           //40
float k3=.00002; //0.00002   //0.00002 ou 4
int milliOld;
int milliNew;
int dt;
float pitchTarget=0;
float pitchActual;
float pitchError=0;
float pitchErrorOld;
float pitchErrorChange;
float pitchErrorSlope=0;
float pitchErrorArea=0;
float pitchServoVal=90;


//====    Variáveis do sistema PID da velocidade  ============================================ antes era
0.02x50x0.00001
float kp=0.02;         //0.01
float kd=10;          //10
float ki=0.0000001;      //0.000001
float rollTarget=2.00;               //2Km/h
float rollActual;
float rollError=0;
float rollErrorOld;
```

```
float rollErrorChange;
float rollErrorSlope=0;
float rollErrorArea=0;
float rollServoVal=90;

//====  Declarando Funções das tarefas RTOS:  ================================================
TaskHandle_t Task1, Task2, Task3;
SemaphoreHandle_t baton;



//=============================================================================
// LOOP #1 (Leitura de Mag, Conta as medições do sensor IR de velocidadeX, calcula o magnetômetro)
//=============================================================================
void codeForTask1( void * parameter )
{
  for (;;) {

//functionRead_X_Speed()
    leitura = digitalRead(Dhall);
//Contagem das leituras para contar a velocidade
    if(leitura == 1){
      A = 0;
    }
    else if(leitura == 0){
      A = 1;
     }

    if (B==0 && A==1){
      if(ContState==0){
        Cont++;
        ContState = 1;
      }
      else{
        ContState = 0;
        Cont++;
      }
    }
    B = A;
    delay(1);
  }
}



//=============================================================================
//  SETUP
//=============================================================================
```

```
void setup() {
//====  Setup do FreeRTOs  ========================================
  baton = xSemaphoreCreateMutex();
  xTaskCreatePinnedToCore(
    codeForTask1,
    "Main Code",
    1000,
    NULL,
    1,
    &Task1,
    1);
  delay(500);  // needed to start-up task1

//====  Setup do Magnetômetro  ====================================
  // inicializa a biblioteca para comunicação I2C
  Wire.begin();
  // inicializa o Magnetômetro
  Serial.println("Initializing I2C devices...");
  mag.initialize();
  // verify connection
  Serial.println("Testing device connections...");
  Serial.println(mag.testConnection() ? "HMC5883L connection successful" : "HMC5883L connection failed");
  //Ativando o magnetômetro no HMC5883L
  Wire.beginTransmission(0x68);
  Wire.write(0x37);
  Wire.write(0x02);
  Wire.endTransmission();
  //
  Wire.beginTransmission(0x68);
  Wire.write(0x6A);
  Wire.write(0x00);
  Wire.endTransmission();
  //Desabilitando o Sleep Mode
  Wire.beginTransmission(0x68);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission();


//====Inicia Serial e seta o pino do velocímetro/odometro==============
  Serial.begin(115200);
  pinMode(Dhall, INPUT);

//====  Setando os Pinos  dos Servos ====================================
  motorDC.attach(27);   //fio AMARELO //27
  direcao.attach(33);   //fio LARANJA
```

```
  marcha.attach(32);    //fio VERDE
  difFront.attach(19);  //fio ROXO //21
  difTras.attach(18);   //fio MARROM

//====  Servos no neutro/Diferenciais destravados e Marcha Trativa =====
  motorDC.writeMicroseconds(1451);
  direcao.writeMicroseconds(1512);
  marcha.writeMicroseconds(2001);
  difFront.writeMicroseconds(2001);
  difTras.writeMicroseconds(1000);

//====  Teste de direção, termina reta ================================
    direcao.writeMicroseconds(1451);    //Direção reta
        delay(1000);
        direcao.write(44);
        delay(500);
        direcao.write(139);
        delay(500);
        direcao.write(90);
        delay(1000);

//====  Armando o ESC do Motor DC =====================================
   for (int angleT = 90; angleT <= 100; angleT++) {
      motorDC.write(angleT);
      delay(150);
   }
  motorDC.writeMicroseconds(1512);    //Motor parado

//====  Liga o motor ==================================================
    delay(3000);
    Motor_Ligado = true;

//====  Reseta o Odômetro =============================================
    ContOld = 0;
    Cont = 0;

//====  Extrai a média da leitura do Magnetômetro e chama de zero =======
//functionRead_MAG_HEADING
     mag.getHeading(&mx1, &my1, &mz1);
     delay(100);
     mag.getHeading(&mx2, &my2, &mz2);
     delay(100);
     mag.getHeading(&mx3, &my3, &mz3);
     delay(100);
     mag.getHeading(&mx4, &my4, &mz4);
     delay(100);
```

```
      heading1 = atan2(my1, mx1);
      if(heading1 < 0){
      heading1 += 2 * M_PI;
      }
      heading2 = atan2(my2, mx2);
      if(heading2 < 0){
      heading2 += 2 * M_PI;
      }
      heading3 = atan2(my3, mx3);
      if(heading3 < 0){
      heading3 += 2 * M_PI;
      }
      heading4 = atan2(my4, mx4);
      if(heading4 < 0){
      heading4 += 2 * M_PI;
      }
      headingM = (heading1+heading2+heading3+heading4)/4;

      //pitchTarget = headingM;
    delay(3000);


    Time = millis();
    milliNew=millis();
}

//================================================================================
// VOID LOOP (dps do setup) (Loop 2)
//================================================================================
void loop() {
       while(Motor_Ligado == true){
           while(m < 5){ //16.11m se for o programa do circulo
              motorDC.attach(27);

              //motorDC.write(104); //Agora o motor DC está controlado pelo PID

    //functionRead_MAG_HEADING
    mag.getHeading(&mx, &my, &mz);

    //Cálculos do Magnetômetro
    heading = atan2(my, mx);
    if(heading < 0){
      heading += 2 * M_PI;
    }
    headingR = (heading * 180/M_PI) - (headingM * 180/M_PI);
```

```
//Delta_t do velocimetro/odometro
if(Cont!=ContOld){
  timeOld = Time;
  Time = millis();
  tempo_de_Leitura = (Time - timeOld);
}
if(tempo_de_Leitura == 0){
  tempo_de_Leitura = 0.001;
}

//Cálculos do Odômetro
REV = Cont - ContOld;
if(REV != 0){
  REVs = REV;
}
RPS = ((REVs / tempo_de_Leitura)* 1000);
RPM = RPS*60;
mPs = RPS / razao_eixo_roda * Circ_roda;
kmPh = mPs * 3.6;
m = Cont * Circ_roda / razao_eixo_roda ;
ContOld = Cont;
Vlon = kmPh;

//Cálculos do Magnetômetro com PID
pitchActual = headingR;

//Cálculos do velocimetro com PID
rollActual = Vlon; //Mostra a velocidade real


//Pega o angulo de acordo com o cm
//input = m * 100;
//pitchTarget = trajetoria[input];

//Cálculo do Tempo de resposta do PID
milliOld=milliNew;
milliNew=millis();
dt=milliNew-milliOld;

//Cálculo de cada termo do PID da direção
pitchErrorOld=pitchError;
pitchError=pitchTarget-pitchActual;
pitchErrorChange=pitchError-pitchErrorOld;
pitchErrorSlope=pitchErrorChange/dt;
pitchErrorArea=pitchErrorArea+pitchError*dt;
```

```
//Cálculo de cada termo do PID da velocidade
rollErrorOld=rollError;
rollError=rollTarget-rollActual;
rollErrorChange=rollError-rollErrorOld;
rollErrorSlope=rollErrorChange/dt;
rollErrorArea=rollErrorArea+rollError*dt;

//Cálculo e limitação do SP de direção
pitchServoVal=pitchServoVal + k1*pitchError + k2*pitchErrorSlope + k3*pitchErrorArea;
if(pitchServoVal >= 180){
  pitchServoVal=180;
}
if(pitchServoVal <= 0){
  pitchServoVal=0;
}
//Serial.print("                                          pitchServoVal: ");
//Serial.print(pitchServoVal);
//Serial.println("º");
direcao.write(pitchServoVal);
//delay(2);

//Cálculo e limitação do SP de velocidade
rollServoVal=rollServoVal + kp*rollError + kd*rollErrorSlope + ki*rollErrorArea;
if(rollServoVal >= 180){
  rollServoVal=180;
}
if(rollServoVal <= 94){
  rollServoVal=94;
}
//Serial.print("                                          rollServoVal: ");
//Serial.print(rollServoVal);
//Serial.println("  ");
motorDC.write(rollServoVal);
//delay(2);



  //Dados do Odômetro
  Serial.println("===============");
  Serial.print("Motor ligado: ");
  Serial.println(Motor_Ligado);
  Serial.print("VOLTAS: ");
  Serial.println(Cont);
  Serial.print("TEMPO DE LEITURA: ");
  Serial.println(tempo_de_Leitura);
```

```
        Serial.print("REVs: ");
        Serial.println(REVs);
        Serial.print("REV: ");
        Serial.println(REV);
        Serial.print("RPS: ");
        Serial.println(RPS);
        Serial.print("RPM: ");
        Serial.println(RPM);
        Serial.print("m/s: ");
        Serial.println(mPs);
        Serial.print("Dist: ");
        Serial.print(m);
        Serial.println(" m");
        Serial.print("TEMPO: ");
        Serial.print(tempo_de_Leitura);
        Serial.println(" ms");
        Serial.print("Vlon: ");
        Serial.print(Vlon);
        //Serial.println("  ");
        Serial.println("  km/h");



            }
            Motor_Ligado = false;

        }
            motorDC.write(94);
            motorDC.detach();

            direcao.write(94);
            delay(1000);
            direcao.detach();
}
```

APÊNDICE L: Iteração de Simulação de trajetória 1

```
clear all
close all
clc

lr_car = 0.182; % Dimensão lr do carro (m)
l_car = 0.324; % Dimensão l do carro (m)
car = [0 0 0.55 0 0]; %Variáveis de estado do carro [X(m) Y(m) V_lon(m/s)
Psi(°) delta(°)]
trajeto = [
    car(1),car(2);
    4,0;
    4,3;
    -2,3;
    -2,0;
    0,0];

dt = 1/160; % Tamanho do passo temporal (s)
delta_curv = 20; % Máximo ângulo delta usado nas curvas (°)
tol = 0.01; % tolerância do objetivo final (m)

x = car(1);
y = car(2);
v_lon = car(3);
psi = car(4);
delta = car(5);
t(1) = 0; %tempo inicial

passos  = numel(trajeto)/2;
done = 0;
passo = 1;
i = 1; % contador
while done==0
    if passo ~= passos-1
        dest0 = trajeto(passo,:);   %Ponto do qual ele está vindo
        dest1 = trajeto(passo+1,:); %Ponto para o qual o veículo está indo
no momento
        dest2 = trajeto(passo+2,:); %Ponto no qual ele vai depois
        ret_a = dest1 - dest0;
        ret_b = dest2 - dest1;

        xa = ret_a(1); ya = ret_a(2);
        xb = ret_b(1); yb = ret_b(2);
        alpha = acosd((xa*xb+ya*yb)/(sqrt(xa^2 +
ya^2)*sqrt(xb^2+yb^2)));   %ângulo entre vetores
        beta_curv =
atand((lr_car/l_car)*tand(delta_curv));                 %angulação do vetor
velocidade resultante em realçaõ ao eixo central do carro
        r_curv = sqrt(lr_car^2 +
(l_car/tand(delta_curv))^2);               %formula do raio de curvatura
        x_curv = r_curv*tand((alpha-
beta_curv)/2);                          %Comprimento onde começa a fazer a
curva

        d_dest = norm(dest1 - [x y]);           %Calcula a distância em
linha reta da posição atual até o destino
        while d_dest>x_curv                     %Calcula quando deve
iniciar a curva
```

```
            i = i+1;                                %Contador começa no 1 para
não apagar o vetor inicial

            beta =
atand((lr_car/l_car)*tand(delta));                  %Calcula beta a
partir de delta
            v =
v_lon*cosd(beta);                                   %
            psi_m = psi +
0.5*dt*(v/l_car)*cosd(beta)*tand(delta)*180/pi;   %Valor medio de psi:
psi_anterior + Psi_próximo dividido por 2 peq variação de tempo (metade da
variação)
            psi = psi + dt*(v/l_car)*cosd(beta)*tand(delta)*180/pi;
            x =
x+dt*v*cosd(psi_m+beta);                            %Novas
coordenadas via regra dos trapésios
            y = y+dt*v*sind(psi_m+beta);

            car(i,:) = [x, y, v_lon, psi, delta];  %Cria novas linhas com
esses dados
            t(i) = t(i-1)+dt;                       %
            d_dest = norm(dest1 - [x y]);          %Calcular a angulação da
roda para o passo seguinte

            %Decisão de angulação da roda
            if d_dest>x_curv                        %
                dir = dest1 - [x y];                %Vetor destino desejado
            else
                dir = dest2 - [x y];
            end
            dir = dir/norm(dir);                    %Deixa o vetor destino
desejado unitátio
            dir_pres = [cosd(psi) sind(psi)];     %Vetor direcional do carro
Xv e Yv
            vec = cross([dir_pres 0], [dir 0]);   %Vetor resultado de
produto vetorial.
            if vec(3)> 0                            %Se a terceira coluna é
maior que zero, o produto é positivo e um está à esquerda do outro
                delta = delta_curv;
            elseif vec(3)< 0                        %Se a terceira coluna é
menor que zero, o produto é negativo e um está à direita do outro
                delta = -delta_curv;
            elseif dir(1) == dir_pres(1)          %Se der zero não vira
                delta = 0;
            else
                delta = delta_curv;                 %Se estiver indo pra trás,
vira à esquerda
            end
        end
        passo = passo+1;                            %Quando termina um passo,
vai pro prox, volta o loop
    else                                            %Else indo ao destino
final
        dest1 = trajeto(passo+1,:);
        d_dest = norm(dest1 - [x y]);
        while d_dest>tol                            % ctrlC ctrlV do loop
acima sem considerar um próximo destino
            i = i+1;

            beta = atand((lr_car/l_car)*tand(delta));
            v = v_lon*cosd(beta);
```

```
            psi_m = psi + 0.5*dt*(v/l_car)*cosd(beta)*tand(delta)*180/pi;
            psi = psi + dt*(v/l_car)*cosd(beta)*tand(delta)*180/pi;
            x = x+dt*v*cosd(psi_m+beta);
            y = y+dt*v*sind(psi_m+beta);

            car(i,:) = [x, y, v_lon, psi, delta];
            t(i) = t(i-1)+dt;
            d_dest = norm(dest1 - [x y]);

            if d_dest<tol
                done=1;   %Para sair do looop grandão
            else
                dir = dest1 - [x y];
            end
            dir = dir/norm(dir);
            dir_pres = [cosd(psi) sind(psi)];
            vec = cross([dir_pres 0], [dir 0]);
            if vec(3)> 0
                delta = delta_curv;
            elseif vec(3)< 0
                delta = -delta_curv;
            elseif dir(1) == dir_pres(1)
                delta = 0;
            else
                delta = delta_curv;
            end
        end
    end
end
dist_roda = v_lon*t; %Contador de dist. percorrida pela roda (m)

plot(car(:,1),car(:,2))
hold on
plot(trajeto(:,1),trajeto(:,2),'-.r')
grid on;
set(gcf, 'color', 'white');
title('Simulação de trajetória - Iteração 1');
xlabel('X [m]', 'FontName', 'Calibri');
ylabel('Y [m]', 'FontName', 'Calibri');
daspect([1 1 1])

Result = [t' car];

output = [dist_roda' car(:,4)];
fileID = fopen('output_ideal.txt','w');
fprintf(fileID,'%.2f %.2f\n',output');
fclose(fileID);
```

APÊNDICE M: Iteração de Simulação de trajetória 2

```
clear all
close all
clc

lr_car = 0.182;          % Dimensão lr do carro (m)
l_car = 0.324;           % Dimensão l do carro (m)
car = [0 0 0.55 0 0];    %Var. do carro [X(m) Y(m) V_lon(m/s) Psi(°)
delta(°)]
trajeto = [car(1),car(2);
    4,0;
    4,3;
    -2,3;
    -2,0;
    0,0];                %Coordenadas x e y dos vértices do trajeto

dt = 1/160;              % Tamanho do passo temporal (s)
delta_curv = 20;         % Máximo ângulo delta usado nas curvas (°)
v_delta = 30;            % Velocidade de variação de delta (°/s) %%NOVIDADE
DO PROGRAMA
tol = 0.25;              % tolerância do objetivo final (m)

x = car(1);              %Coordenada x (atualizada a cada passo de tempo)
y = car(2);              %Coordenada y (atualizada a cada passo de tempo)
v_lon = car(3);          %Velocidade longitudinal
psi = car(4);            % Ângulo Psi (atualizada a cada passo de tempo)
delta = car(5);          % Ângulo delta (atualizada a cada passo de tempo)

t(1) = 0;                % Iniciando contador de tempo
dist(1) = 0;             % Iniciando contador de dist. percorrida pelo cm

passos  = numel(trajeto)/2; % Quantidade de trechos na trajetória
d_delta = v_delta*dt;    %variação ângulo delta a cada dt
done = 0;
passo = 1;
i = 1;                   % contador
while done==0
    if passo ~= passos-1
        dest0 = trajeto(passo,:); % Ponto de partida do passo
        dest1 = trajeto(passo+1,:); % Destino do passo
        dest2 = trajeto(passo+2,:); % Destino seguinte
        ret_a = dest1 - dest0;
        ret_b = dest2 - dest1;

%  x_curv = Antecipação necessária para a curva (toma em conta ângulo da
%           curva, angulo beta máx e velocidade de variação de beta)
        xa = ret_a(1); ya = ret_a(2);
        xb = ret_b(1); yb = ret_b(2);
        alpha = acosd((xa*xb+ya*yb)/(sqrt(xa^2 + ya^2)*sqrt(xb^2+yb^2)));
        beta_curv = atand((lr_car/l_car)*tand(delta_curv));
        a = lr_car/l_car; b = delta_curv;
        int = (atand((sqrt(1-a^2)*tand(b))/sqrt(1+a^2*tand(b)^2)))/sqrt(1-
a^2)/v_delta*v_lon*cos(atand(a*tand(b))); %integral de beta, dá um
angulo(arc tan) divide por v_delta e multiplica pela velocidade (vlon x
cosBeta)
        r_curv = sqrt(lr_car^2 + (l_car/tand(delta_curv))^2); % Raio (m)
        x_curv = r_curv*tand((alpha-beta_curv)/2)+int; % Antecipação (m)
```

```
    d_dest = norm(dest1 - [x y]);      %Distância ao destino atual (m)
    while d_dest>x_curv
        i = i+1;

        beta = atand((lr_car/l_car)*tand(delta));
        v = v_lon*cosd(beta);
        psi_m = psi + 0.5*dt*(v/l_car)*cosd(beta)*tand(delta)*180/pi;
        psi = psi + dt*(v/l_car)*cosd(beta)*tand(delta)*180/pi;
        x = x+dt*v*cosd(psi_m+beta);
        y = y+dt*v*sind(psi_m+beta);

        car(i,:) = [x, y, v_lon, psi, delta];
        t(i) = t(i-1)+dt;
        dist(i) = dist(i-1)+v*dt;
        d_dest = norm(dest1 - [x y]);

        if d_dest>x_curv
            dir = dest1 - [x y];
        else
            dir = dest2 - [x y];
        end
        dir = dir/norm(dir);
        dir_pres = [cosd(psi) sind(psi)];
        vec = cross([dir_pres 0], [dir 0]);
        if vec(3)> 0
            if (delta_curv-delta)>d_delta
                delta = delta+d_delta;
            else
                delta = +delta_curv;
            end
        elseif vec(3)< 0
            if (delta+delta_curv)>d_delta       %
                delta = delta-d_delta;          %alteração
            else
                delta = -delta_curv;            %alteração
            end
        elseif dir(1) == dir_pres(1)
            delta = 0;
        else
            if (delta_curv-delta)>d_delta
                delta = delta+d_delta;
            else
                delta = +delta_curv;
            end
        end
    end
    passo = passo+1;
else
    dest1 = trajeto(passo+1,:);
    d_dest = norm(dest1 - [x y]);
    while d_dest>tol
        i = i+1;

        beta = atand((lr_car/l_car)*tand(delta));
        v = v_lon*cosd(beta);
        psi_m = psi + 0.5*dt*(v/l_car)*cosd(beta)*tand(delta)*180/pi;
        psi = psi + dt*(v/l_car)*cosd(beta)*tand(delta)*180/pi;
        x = x+dt*v*cosd(psi_m+beta);
        y = y+dt*v*sind(psi_m+beta);

        car(i,:) = [x, y, v_lon, psi, delta];
```

```
            t(i) = t(i-1)+dt;
            dist(i) = dist(i-1)+v*dt;
            d_dest = norm(dest1 - [x y]);

            if d_dest<tol
                done=1;
            else
                dir = dest1 - [x y];
            end
            dir = dir/norm(dir);
            dir_pres = [cosd(psi) sind(psi)];
            vec = cross([dir_pres 0], [dir 0]);
            if vec(3)> 0
                if (delta_curv-delta)>d_delta
                    delta = delta+d_delta;
                else
                    delta = +delta_curv;
                end
            elseif vec(3)< 0
                if (delta+delta_curv)>d_delta
                    delta = delta-d_delta;
                else
                    delta = -delta_curv;
                end
            elseif dir(1) == dir_pres(1)
                delta = 0;
            else
                if (delta_curv-delta)>d_delta
                    delta = delta+d_delta;
                else
                    delta = +delta_curv;
                end
            end
        end
    end
end
dist_roda = v_lon*t; %Contador de dist. percorrida pela roda (m)

plot(car(:,1),car(:,2))
hold on
plot(trajeto(:,1),trajeto(:,2),'-.r')
grid on;
set(gcf, 'color', 'white');
title('Simulação de trajetória - Iteração 2');
xlabel('X [m]', 'FontName', 'Calibri');
ylabel('Y [m]', 'FontName', 'Calibri');
daspect([1 1 1])

Result = [t' car];

output = [dist_roda' car(:,4)];
fileID = fopen('output.txt','w');
fprintf(fileID,'%.2f %.2f\n',output');
fclose(fileID);
```

APÊNDICE N: Iteração de Simulação de trajetória 3

```
clear all
close all
clc

lr_car = 0.182;          % Dimensão lr do carro (m)
l_car = 0.324;           % Dimensão l do carro (m)
car = [0 0 1/3.6 0 0];  %Var. do carro [X(m) Y(m) V_lon(m/s) Psi(°)
delta(°)]
trajeto = [car(1),car(2);
                2,0;
                4,1;
                7,-1;
               11,0;
               13,0];
%      2,0;
%      4,3;
%      -2,3;
%      -2,0;
%      0,0];               %Coordenadas x e y dos vértices do trajeto

% trajeto = [car(1),car(2);
%      -2,1
%      -8,1
%      -9,2
%      -11.5,0
%      -9,-2
%      -8,-1
%      -2,-1
%      2,-2.75
%      0,0];               %Coordenadas x e y dos vértices do trajeto

dt = 1/(car(3)*1000);              % Tamanho do passo temporal (s)
delta_curv = 12;         % Máximo ângulo delta usado nas curvas (°)
v_delta = 10;            % Velocidade de variação de delta (°/s)
limitador = 3;           % Limitador da correção de ângulo
tol = 0.002;               % tolerância do objetivo final (m)

x = car(1);              %Coordenada x (atualizada a cada passo de tempo)
y = car(2);              %Coordenada y (atualizada a cada passo de tempo)
v_lon = car(3);          %Velocidade longitudinal
psi = car(4);            % Ângulo Psi (atualizada a cada passo de tempo)
delta = car(5);          % Ângulo delta (atualizada a cada passo de tempo)

t(1) = 0;                % Iniciando contador de tempo
dist(1) = 0;             % Iniciando contador de dist. percorrida pelo cm

passos  = numel(trajeto)/2; % Quantidade de trechos na trajetória
d_delta = v_delta*dt;    %variação ângulo delta a cada dt
done = 0;
passo = 1;
i = 1;                   % contador
while done==0
    if passo ~= passos-1
        dest0 = trajeto(passo,:); % Ponto de partida do passo
        dest1 = trajeto(passo+1,:); % Destino do passo
        dest2 = trajeto(passo+2,:); % Destino seguinte
        ret_a = dest1 - dest0;
```

```
        ret_b = dest2 - dest1;

%  x_curv = Antecipação necessária para a curva (toma em conta ângulo da
%           curva, angulo beta máx e velocidade de variação de beta)
        xa = ret_a(1); ya = ret_a(2);
        xb = ret_b(1); yb = ret_b(2);
        alpha = acosd((xa*xb+ya*yb)/(sqrt(xa^2 + ya^2)*sqrt(xb^2+yb^2)));
        beta_curv = atand((lr_car/l_car)*tand(delta_curv));
        a = lr_car/l_car; b = delta_curv;
        int = (atand((sqrt(1-a^2)*tand(b))/sqrt(1+a^2*tand(b)^2)))/sqrt(1-
a^2)/v_delta*v_lon*cos(atand(a*tand(b)));
        r_curv = sqrt(lr_car^2 + (l_car/tand(delta_curv))^2); % Raio (m)
        x_curv = r_curv*tand((alpha-beta_curv)/2)+int; % Antecipação (m)

        d_dest = norm(dest1 - [x y]);      %Distância ao destino atual (m)
        while d_dest>x_curv
            i = i+1;

            beta = atand((lr_car/l_car)*tand(delta));
            v = v_lon*cosd(beta);
            psi_m = psi + 0.5*dt*(v/l_car)*cosd(beta)*tand(delta)*180/pi;
            psi = psi + dt*(v/l_car)*cosd(beta)*tand(delta)*180/pi;
            x = x+dt*v*cosd(psi_m+beta);
            y = y+dt*v*sind(psi_m+beta);

            car(i,:) = [x, y, v_lon, psi, delta];
            t(i) = t(i-1)+dt;
            dist(i) = dist(i-1)+v*dt;
            d_dest = norm(dest1 - [x y]);

            if d_dest>x_curv
                dir = dest1 - [x y];
            else
                dir = dest2 - [x y];
            end
            dir = dir/norm(dir);
            dir_pres = [cosd(psi) sind(psi)];
            vec = cross([dir_pres 0], [dir 0]);
            esc = dot([dir_pres 0], [dir 0]);

            %limitador
            if esc>=0
                delta_curv_lim = limitador*delta_curv*abs(vec(3));
                delta_curv_lim = min(delta_curv,delta_curv_lim);
            else
                delta_curv_lim = delta_curv;
            end
            if vec(3)> 0
                if (delta_curv_lim-delta)>d_delta
                    delta = delta+d_delta;
                else
                    delta = +delta_curv_lim;
                end
            elseif vec(3)< 0
                if (delta+delta_curv_lim)>d_delta
                    delta = delta-d_delta;
                else
                    delta = -delta_curv_lim;
                end
            elseif dir(1) == dir_pres(1)
                delta = 0;
```

```
        else
            if (delta_curv_lim-delta)>d_delta
                delta = delta+d_delta;
            else
                delta = +delta_curv_lim;
            end
        end
    end
    passo = passo+1;
else
    dest1 = trajeto(passo+1,:);
    d_dest = norm(dest1 - [x y]);
    while d_dest>tol
        i = i+1;

        beta = atand((lr_car/l_car)*tand(delta));
        v = v_lon*cosd(beta);
        psi_m = psi + 0.5*dt*(v/l_car)*cosd(beta)*tand(delta)*180/pi;
        psi = psi + dt*(v/l_car)*cosd(beta)*tand(delta)*180/pi;
        x = x+dt*v*cosd(psi_m+beta);
        y = y+dt*v*sind(psi_m+beta);

        car(i,:) = [x, y, v_lon, psi, delta];
        t(i) = t(i-1)+dt;
        dist(i) = dist(i-1)+v*dt;
        d_dest = norm(dest1 - [x y]);

        if d_dest<tol
            done=1;
        else
            dir = dest1 - [x y];
        end
        dir = dir/norm(dir);
        dir_pres = [cosd(psi) sind(psi)];
        vec = cross([dir_pres 0], [dir 0]);
        esc = dot([dir_pres 0], [dir 0]);
        if esc>=0
            delta_curv_lim = limitador*delta_curv*abs(vec(3));
            delta_curv_lim = min(delta_curv,delta_curv_lim);
        else
            delta_curv_lim = delta_curv;
        end
        if vec(3)> 0
            if (delta_curv_lim-delta)>d_delta
                delta = delta+d_delta;
            else
                delta = +delta_curv_lim;
            end
        elseif vec(3)< 0
            if (delta+delta_curv_lim)>d_delta
                delta = delta-d_delta;
            else
                delta = -delta_curv_lim;
            end
        elseif dir(1) == dir_pres(1)
            delta = 0;
        else
            if (delta_curv_lim-delta)>d_delta
                delta = delta+d_delta;
            else
                delta = +delta_curv_lim;
            end
```

```
            end
        end
    end
end
dist_roda = v_lon*t; %Contador de dist. percorrida pela roda (m)

plot(car(:,1),car(:,2))
hold on
plot(trajeto(:,1),trajeto(:,2),'-.r')
grid on;
set(gcf, 'color', 'white');
title('Simulação de trajetória - Iteração 3');
xlabel('X [m]', 'FontName', 'Calibri');
ylabel('Y [m]', 'FontName', 'Calibri');
daspect([1 1 1])

Result = [t' car];

%output = [dist_roda' car(:,4)];
j=0;
for i = 11:10:(numel(dist_roda))
    j=j+1;
    output(j) = car(i,4);
end
fileID = fopen('output_correcaoCurva_TESTE_FINAL.txt','w');
fprintf(fileID,'[');
fprintf(fileID,'%.2fa ',output(1:j-1)');
fprintf(fileID,'%.2f',output(j)');
fprintf(fileID,']');
fclose(fileID);
```

APÊNDICE O: Teste de percurso seguindo trajetória pré calculada.

```
//=====================================================
// Código utilizado para teste trajetória pre calculada
// por programa de matlab lendo vetor de ângulo psi
//=====================================================
//                   Programa escrito por Lucas Koeler

//====  Bibliotecas Utilizadas  ========================================
#include <Servo.h>
#include "Wire.h"
#include "I2Cdev.h"
#include "HMC5883L.h"
#include <math.h>

//====  Sistema de Leitura de Velocidade e Odômetro  ===================
int Dhall = 23; // fio Laranja
int leitura;
int A;
int B = 1;
int ContState = 0;
int pos;

//====  Mapeamento dos pinos  ==========================================
const byte sensorIR = 2;
Servo motorDC, direcao, marcha, difFront, difTras;

//====  Declarando Variáveis  ==========================================
unsigned long Cont = 0;
unsigned long ContOld;
unsigned long REV = 0;          //número de revoluções
float RPS = 0;                      //revoluções por segundo
float RPM = 0;                      //revoluções por minuto
float mPs = 0;                      //metros por segundo
float kmPh = 0;                     //Km por hora
float m = 0;                    //Metros percorridos
int tempo;                      //Tempo em segundos
int tempo_antigo = 0;           //Tempo em segundos usado para fazer a função Time Interrupt
float tempo_de_Leitura;     //Tempo em que a função ISR está ativa contando as revoluções
unsigned long Time;
unsigned long timeOld;
float Vlon = 0;
float REVs = 0;
int Motor_Ligado = false;
```

```
//====  Declarando Constantes de geometria do veículo:  ========================================
float Diam_roda = 0.10448029;        //Diâmetro da Roda em [m]
float razao_eixo_roda = 7.4181;      //Para cada 7.4181 rotações do eixo a roda gira 1 vez
float pi = 3.14159265;
float Circ_roda = 0.328234513;       //Circ_roda = Diam_roda * pi; //Circunferência da roda: aprox. = 0.370697 [m]
(pode mudar se mudar o diam.)

//====  Sistema do IMU - Magnetômetro  ============================================================
HMC5883L mag;                        //Inicializa o Magnetômetro
int16_t mx, my, mz;
float heading;

//====  Variáveis do Magnetômetro  ================================================================
int16_t mx1, my1, mz1,  mx2, my2, mz2,  mx3, my3, mz3,  mx4, my4, mz4;
float heading1,heading2,heading3,heading4,headingM ;
float headingR;

//====  Variáveis do sistema PID da direção  ================================================== antes era
0.02x50x0.00001
// o melhor da minha vida é sem seriais 0.01, 10, 0.00001 depois da matriz até agr fica bom o 0.008, 10, 0.00001
float k1=0.008; //0.01        //0.05             //0.05
float k2=10; // 10           //40               //60
float k3=.00001; //0.00002   //0.00002 ou 4     //0.00002
int milliOld;
int milliNew;
int dt;
float pitchTarget=0;
float pitchActual;
float pitchError=0;
float pitchErrorOld;
float pitchErrorChange;
float pitchErrorSlope=0;
float pitchErrorArea=0;
float pitchServoVal=90;

//====  Variáveis do sistema PID da velocidade ================================================ antes era
0.02x50x0.00001
float kp=0.01   ;         //0.01
float kd=10;          //10
float ki=0.0000001;      //0.000001
float rollTarget=1.00;               //1Km/h
float rollActual;
float rollError=0;
float rollErrorOld;
float rollErrorChange;
float rollErrorSlope=0;
```

```
float rollErrorArea=0;
float rollServoVal=94;

//====  FUÇÃO SEGUIR ROTA  ========================================================================
int input;
//float trajetoria[] = {0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.01, 0.03, 0.07,
0.11, 0.17, 0.23, 0.31, 0.40, 0.50, 0.61, 0.73, 0.86, 1.01, 1.16, 1.33, 1.50, 1.69, 1.89, 2.10, 2.32, 2.56, 2.80,
3.06, 3.32, 3.60, 3.89, 4.19, 4.50, 4.82, 5.15, 5.50, 5.85, 6.22, 6.60, 6.98, 7.38, 7.79, 8.22, 8.65, 9.09, 9.55,
10.02, 10.49, 10.98, 11.48, 11.99, 12.52, 13.05, 13.59, 14.15, 14.72, 15.29, 15.88, 16.48, 17.09, 17.71, 18.33,
18.95, 19.56, 20.18, 20.80, 21.42, 22.04, 22.65, 23.27, 23.89, 24.51, 25.12, 25.74, 26.36, 26.98, 27.60, 28.21,
28.83, 29.45, 30.07, 30.68, 31.30, 31.92, 32.54, 33.16, 33.77, 34.39, 35.01, 35.63, 36.25, 36.86, 37.48, 38.10,
38.72, 39.33, 39.95, 40.57, 41.19, 41.81, 42.42, 43.04, 43.66, 44.28, 44.89, 45.51, 46.13, 46.75, 47.37, 47.98,
48.60, 49.22, 49.84, 50.45, 51.07, 51.69, 52.31, 52.93, 53.54, 54.16, 54.78, 55.40, 56.02, 56.63, 57.25, 57.87,
58.49, 59.10, 59.72, 60.34, 60.96, 61.58, 62.19, 62.81, 63.43, 64.05, 64.66, 65.28, 65.90, 66.52, 67.14, 67.75,
68.37, 68.99, 69.61, 70.22, 70.81, 71.38, 71.94, 72.48, 73.01, 73.52, 74.01, 74.49, 74.96, 75.42, 75.86, 76.28,
76.70, 77.10, 77.49, 77.87, 78.24, 78.60, 78.95, 79.29, 79.61, 79.93, 80.24, 80.54, 80.83, 81.11, 81.38, 81.65,
81.91, 82.16, 82.40, 82.63, 82.86, 83.08, 83.30, 83.51, 83.71, 83.91, 84.10, 84.28, 84.46, 84.64, 84.81, 84.97,
85.13, 85.28, 85.43, 85.58, 85.72, 85.86, 85.99, 86.12, 86.25, 86.37, 86.49, 86.60, 86.71, 86.82, 86.92, 87.03,
87.12, 87.22, 87.31, 87.40, 87.49, 87.58, 87.66, 87.74, 87.82, 87.89, 87.97, 88.04, 88.11, 88.17, 88.24, 88.30,
88.36, 88.42, 88.48, 88.54, 88.59, 88.64, 88.70, 88.75, 88.79, 88.84, 88.89, 88.93, 88.97, 89.02, 89.06, 89.10,
89.13, 89.17, 89.21, 89.24, 89.28, 89.31, 89.34, 89.37, 89.40, 89.43, 89.46, 89.49, 89.52, 89.54, 89.57, 89.59,
89.62, 89.64, 89.66, 89.69, 89.71, 89.73, 89.75, 89.77, 89.79, 89.81, 89.83, 89.84, 89.87, 89.91, 89.96, 90.02,
90.10, 90.18, 90.27, 90.38, 90.50, 90.63, 90.76, 90.92, 91.08, 91.25, 91.43, 91.63, 91.83, 92.05, 92.28, 92.52,
92.77, 93.03, 93.30, 93.59, 93.88, 94.19, 94.51, 94.83, 95.17, 95.52, 95.89, 96.26, 96.64, 97.04, 97.44, 97.86,
98.29, 98.73, 99.18, 99.64, 100.11, 100.60, 101.09, 101.60, 102.12, 102.65, 103.19, 103.74, 104.30, 104.87, 105.46,
106.05, 106.66, 107.27, 107.89, 108.51, 109.13, 109.75, 110.36, 110.98, 111.60, 112.22, 112.83, 113.45, 114.07,
114.69, 115.31, 115.92, 116.54, 117.16, 117.78, 118.39, 119.01, 119.63, 120.25, 120.87, 121.48, 122.10, 122.72,
123.34, 123.96, 124.57, 125.19, 125.81, 126.43, 127.04, 127.66, 128.28, 128.90, 129.52, 130.13, 130.75, 131.37,
131.99, 132.60, 133.22, 133.84, 134.46, 135.08, 135.69, 136.31, 136.93, 137.55, 138.16, 138.78, 139.40, 140.02,
140.64, 141.25, 141.87, 142.49, 143.11, 143.73, 144.34, 144.96, 145.58, 146.20, 146.81, 147.43, 148.05, 148.67,
149.29, 149.90, 150.52, 151.14, 151.76, 152.37, 152.99, 153.61, 154.23, 154.85, 155.46, 156.08, 156.70, 157.32,
```

157.93, 158.55, 159.17, 159.79, 160.40, 161.00, 161.58, 162.14, 162.69, 163.22, 163.73, 164.23, 164.71, 165.18,
165.63, 166.07, 166.50, 166.92, 167.32, 167.71, 168.09, 168.45, 168.81, 169.15, 169.48, 169.81, 170.12, 170.43,
170.72, 171.01, 171.28, 171.55, 171.81, 172.06, 172.31, 172.54, 172.77, 172.99, 173.21, 173.42, 173.62, 173.82,
174.01, 174.19, 174.37, 174.54, 174.71, 174.87, 175.03, 175.18, 175.33, 175.47, 175.61, 175.75, 175.88, 176.00,
176.13, 176.25, 176.36, 176.47, 176.58, 176.68, 176.79, 176.88, 176.98, 177.07, 177.16, 177.25, 177.33, 177.41,
177.49, 177.57, 177.64, 177.71, 177.78, 177.85, 177.92, 177.98, 178.04, 178.10, 178.16, 178.21, 178.27, 178.32,
178.37, 178.42, 178.47, 178.51, 178.56, 178.60, 178.64, 178.68, 178.72, 178.76, 178.80, 178.83, 178.87, 178.90,
178.94, 178.97, 179.00, 179.03, 179.06, 179.08, 179.11, 179.14, 179.16, 179.19, 179.21, 179.24, 179.26, 179.28,
179.30, 179.32, 179.34, 179.36, 179.38, 179.40, 179.41, 179.43, 179.45, 179.46, 179.48, 179.49, 179.51, 179.52,
179.54, 179.55, 179.56, 179.57, 179.59, 179.60, 179.61, 179.62, 179.63, 179.64, 179.65, 179.66, 179.67, 179.68,
179.69, 179.70, 179.70, 179.71, 179.72, 179.73, 179.73, 179.74, 179.75, 179.75, 179.76, 179.77, 179.77, 179.78,
179.78, 179.79, 179.80, 179.80, 179.81, 179.81, 179.81, 179.82, 179.82, 179.83, 179.83, 179.84, 179.84, 179.84,
179.85, 179.85, 179.85, 179.86, 179.86, 179.86, 179.87, 179.87, 179.87, 179.88, 179.88, 179.88, 179.88, 179.89,
179.89, 179.89, 179.89, 179.89, 179.90, 179.90, 179.90, 179.90, 179.90, 179.91, 179.91, 179.91, 179.91, 179.91,
179.91, 179.92, 179.92, 179.92, 179.92, 179.92, 179.92, 179.92, 179.93, 179.93, 179.93, 179.93, 179.93, 179.93,
179.93, 179.93, 179.93, 179.93, 179.94, 179.94, 179.94, 179.94, 179.94, 179.94, 179.94, 179.94, 179.94, 179.94,
179.94, 179.94, 179.94, 179.94, 179.95, 179.95, 179.95, 179.95, 179.95, 179.95, 179.95, 179.95, 179.95, 179.95,
179.95, 179.95, 179.95, 179.95, 179.95, 179.95, 179.95, 179.95, 179.95, 179.95, 179.95, 179.95, 179.95, 179.95,
179.95, 179.95, 179.95, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96,
179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96,
179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96,
179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96,
179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96,
179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96,
179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96,
179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96,
179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96,
179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96,
179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96,
179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96, 179.96,
179.96, 179.96, 179.96, 179.96, 179.96, 179.97, 179.98, 180.01, 180.05, 180.10, 180.16, 180.23, 180.31, 180.41,
180.51, 180.63, 180.76, 180.89, 181.04, 181.20, 181.38, 181.56, 181.75, 181.96, 182.17, 182.40, 182.64, 182.89,
183.15, 183.42, 183.71, 184.00, 184.30, 184.62, 184.95, 185.29, 185.64, 186.00, 186.37, 186.75, 187.15, 187.55,
187.97, 188.40, 188.83, 189.28, 189.74, 190.22, 190.70, 191.19, 191.70, 192.22, 192.74, 193.28, 193.83, 194.39,
194.97, 195.55, 196.14, 196.75, 197.36, 197.98, 198.60, 199.22, 199.84, 200.45, 201.07, 201.69, 202.31, 202.93,
203.54, 204.16, 204.78, 205.40, 206.01, 206.63, 207.25, 207.87, 208.49, 209.10, 209.72, 210.34, 210.96, 211.57,
212.19, 212.81, 213.43, 214.05, 214.66, 215.28, 215.90, 216.52, 217.13, 217.75, 218.37, 218.99, 219.61, 220.22,
220.84, 221.46, 222.08, 222.70, 223.31, 223.93, 224.55, 225.17, 225.78, 226.40, 227.02, 227.64, 228.26, 228.87,
229.49, 230.11, 230.73, 231.34, 231.96, 232.58, 233.20, 233.82, 234.43, 235.05, 235.67, 236.29, 236.91, 237.52,
238.14, 238.76, 239.38, 239.99, 240.61, 241.23, 241.85, 242.47, 243.08, 243.70, 244.32, 244.94, 245.55, 246.17,
246.79, 247.41, 248.03, 248.64, 249.26, 249.88, 250.48, 251.06, 251.63, 252.18, 252.72, 253.24, 253.74, 254.23,
254.71, 255.17, 255.62, 256.05, 256.48, 256.89, 257.29, 257.67, 258.05, 258.41, 258.77, 259.11, 259.44, 259.77,
260.08, 260.38, 260.68, 260.97, 261.25, 261.52, 261.78, 262.03, 262.28, 262.52, 262.75, 262.98, 263.20, 263.41,
263.61, 263.81, 264.01, 264.20, 264.38, 264.56, 264.73, 264.90, 265.06, 265.22, 265.37, 265.52, 265.66, 265.80,
265.94, 266.07, 266.20, 266.32, 266.44, 266.56, 266.67, 266.78, 266.89, 266.99, 267.09, 267.19, 267.28, 267.38,
267.46, 267.55, 267.64, 267.72, 267.80, 267.87, 267.95, 268.02, 268.09, 268.16, 268.23, 268.29, 268.35, 268.41,

```
268.47, 268.53, 268.58, 268.64, 268.69, 268.74, 268.79, 268.84, 268.89, 268.93, 268.97, 269.02, 269.06, 269.10,
269.14, 269.18, 269.21, 269.25, 269.28, 269.32, 269.35, 269.38, 269.41, 269.44, 269.47, 269.50, 269.53, 269.55,
269.58, 269.61, 269.63, 269.65, 269.68, 269.70, 269.72, 269.74, 269.76, 269.78, 269.80, 269.82, 269.84, 269.86,
269.88, 269.91, 269.96, 270.02, 270.08, 270.16, 270.25, 270.35, 270.46, 270.58, 270.72, 270.86, 271.02, 271.19,
271.36, 271.55, 271.75, 271.97, 272.19, 272.42, 272.67, 272.92, 273.19, 273.47, 273.76, 274.06, 274.37, 274.69,
275.03, 275.37, 275.73, 276.10, 276.47, 276.86, 277.26, 277.68, 278.10, 278.53, 278.98, 279.44, 279.90, 280.38,
280.87, 281.37, 281.88, 282.41, 282.94, 283.49, 284.04, 284.61, 285.19, 285.78, 286.38, 286.99, 287.61, 288.23,
288.85, 289.46, 290.08, 290.70, 291.32, 291.93, 292.55, 293.17, 293.79, 294.41, 295.02, 295.64, 296.26, 296.88,
297.50, 298.11, 298.73, 299.35, 299.97, 300.58, 301.20, 301.82, 302.44, 303.06, 303.67, 304.29, 304.91, 305.53,
306.14, 306.76, 307.38, 308.00, 308.62, 309.23, 309.85, 310.47, 311.09, 311.70, 312.32, 312.94, 313.56, 314.18,
314.79, 315.41, 316.03, 316.65, 317.27, 317.88, 318.50, 319.12, 319.74, 320.35, 320.97, 321.59, 322.21, 322.83,
323.44, 324.06, 324.68, 325.30, 325.91, 326.53, 327.15, 327.77, 328.39, 329.00, 329.62, 330.24, 330.86, 331.48,
332.09, 332.71, 333.33, 333.95, 334.56, 335.18, 335.80, 336.42, 337.04, 337.65, 338.27, 338.87, 339.46, 340.03,
340.58, 341.12, 341.65, 342.16, 342.66, 343.14, 343.61, 344.06, 344.51, 344.94, 345.36, 345.77, 346.16, 346.55,
346.92, 347.29, 347.64, 347.99, 348.32, 348.65, 348.96, 349.27, 349.57, 349.86, 350.15, 350.42, 350.69, 350.95,
351.20, 351.45, 351.69, 351.92, 352.15, 352.37, 352.59, 352.79, 353.00, 353.19, 353.39, 353.57, 353.76, 353.93,
354.10, 354.27, 354.44, 354.59, 354.75, 354.90, 355.05, 355.19, 355.33, 355.46, 355.59, 355.72, 355.85, 355.97,
356.09, 356.20, 356.31, 356.42, 356.53, 356.64, 356.74, 356.84, 356.93, 357.03, 357.12, 357.21, 357.29, 357.38,
357.46, 357.54, 357.62, 357.70, 357.77, 357.85, 357.92, 357.99, 358.06, 358.12, 358.19, 358.25, 358.31, 358.38,
358.44, 358.49, 358.55, 358.61, 358.66, 358.72, 358.77, 358.82, 358.87, 358.92, 358.97, 359.02, 359.06, 359.11,
359.15, 359.20, 359.24, 359.29, 359.33, 359.37, 359.41, 359.45, 359.50, 359.54, 359.58, 359.62, 359.66, 359.70,
359.74, 359.78, 359.82, 359.86, 359.90, 359.94, 359.99, 360.03, 360.09, 360.14, 360.21};
//float trajetoria2[] = {0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.01, 0.03, 0.06, 0.10, 0.15, 0.22, 0.29, 0.38, 0.48, 0.58, 0.70, 0.83, 0.98,
1.13, 1.29, 1.47, 1.65, 1.85, 2.06, 2.28, 2.51, 2.75, 3.00, 3.27, 3.54, 3.83, 4.13, 4.43, 4.75, 5.09, 5.43, 5.78,
6.14, 6.52, 6.91, 7.30, 7.71, 8.13, 8.56, 9.00, 9.46, 9.92, 10.40, 10.87, 11.33, 11.77, 12.20, 12.62, 13.03, 13.42,
13.81, 14.18, 14.54, 14.89, 15.23, 15.56, 15.88, 16.19, 16.49, 16.78, 17.07, 17.34, 17.61, 17.87, 18.12, 18.37,
18.60, 18.83, 19.06, 19.27, 19.48, 19.69, 19.89, 20.08, 20.26, 20.45, 20.62, 20.79, 20.96, 21.12, 21.27, 21.42,
21.57, 21.71, 21.85, 21.99, 22.12, 22.24, 22.36, 22.48, 22.60, 22.71, 22.82, 22.92, 23.03, 23.13, 23.22, 23.32,
23.41, 23.49, 23.58, 23.66, 23.74, 23.82, 23.90, 23.97, 24.04, 24.11, 24.18, 24.25, 24.31, 24.37, 24.43, 24.49,
24.55, 24.60, 24.65, 24.70, 24.75, 24.80, 24.85, 24.90, 24.94, 24.98, 25.03, 25.07, 25.11, 25.15, 25.18, 25.22,
25.25, 25.29, 25.32, 25.35, 25.39, 25.42, 25.45, 25.47, 25.50, 25.53, 25.56, 25.58, 25.61, 25.63, 25.65, 25.68,
25.70, 25.72, 25.74, 25.76, 25.78, 25.80, 25.82, 25.84, 25.85, 25.87, 25.89, 25.90, 25.92, 25.94, 25.95, 25.96,
25.98, 25.99, 26.01, 26.02, 26.03, 26.04, 26.05, 26.07, 26.08, 26.09, 26.10, 26.11, 26.12, 26.13, 26.14, 26.15,
26.16, 26.17, 26.17, 26.18, 26.18, 26.16, 26.14, 26.10, 26.05, 25.99, 25.92, 25.84, 25.75, 25.65, 25.53, 25.41,
25.27, 25.12, 24.96, 24.79, 24.61, 24.42, 24.22, 24.00, 23.78, 23.54, 23.29, 23.03, 22.76, 22.48, 22.19, 21.88,
21.57, 21.24, 20.91, 20.56, 20.20, 19.83, 19.45, 19.06, 18.65, 18.24, 17.81, 17.37, 16.93, 16.47, 16.00, 15.51,
15.02, 14.52, 14.00, 13.48, 12.94, 12.39, 11.83, 11.26, 10.68, 10.09, 9.48, 8.87, 8.25, 7.63, 7.01, 6.40, 5.78,
5.16, 4.54, 3.93, 3.31, 2.69, 2.07, 1.45, 0.84, 0.22, -0.40, -1.02, -1.64, -2.25, -2.87, -3.49, -4.11, -4.72, -
5.34, -5.96, -6.58, -7.20, -7.81, -8.43, -9.05, -9.67, -10.28, -10.90, -11.52, -12.14, -12.76, -13.37, -13.99, -
14.61, -15.23, -15.85, -16.46, -17.08, -17.70, -18.32, -18.93, -19.55, -20.17, -20.79, -21.41, -22.02, -22.64, -
23.26, -23.88, -24.48, -25.07, -25.65, -26.21, -26.75, -27.27, -27.78, -28.28, -28.76, -29.22, -29.68, -30.12, -
30.54, -30.96, -31.36, -31.75, -32.13, -32.50, -32.86, -33.20, -33.54, -33.86, -34.18, -34.49, -34.79, -35.08, -
35.36, -35.63, -35.89, -36.15, -36.40, -36.64, -36.87, -37.10, -37.32, -37.54, -37.74, -37.95, -38.14, -38.33, -
```

```
38.52, -38.69, -38.87, -39.04, -39.20, -39.36, -39.51, -39.66, -39.80, -39.94, -40.08, -40.21, -40.34, -40.47, -
40.59, -40.70, -40.82, -40.93, -41.04, -41.14, -41.24, -41.34, -41.43, -41.53, -41.61, -41.70, -41.79, -41.87, -
41.95, -42.02, -42.10, -42.17, -42.24, -42.31, -42.38, -42.44, -42.50, -42.56, -42.62, -42.68, -42.74, -42.79, -
42.84, -42.89, -42.94, -42.99, -43.04, -43.08, -43.13, -43.17, -43.21, -43.25, -43.29, -43.33, -43.36, -43.40, -
43.43, -43.47, -43.50, -43.53, -43.56, -43.59, -43.62, -43.65, -43.68, -43.70, -43.73, -43.75, -43.78, -43.80, -
43.83, -43.85, -43.87, -43.89, -43.91, -43.93, -43.95, -43.97, -43.99, -44.00, -44.02, -44.04, -44.05, -44.07, -
44.09, -44.10, -44.12, -44.13, -44.14, -44.16, -44.17, -44.18, -44.19, -44.21, -44.22, -44.23, -44.24, -44.25, -
44.26, -44.27, -44.28, -44.29, -44.30, -44.31, -44.32, -44.32, -44.33, -44.34, -44.35, -44.36, -44.36, -44.37, -
44.38, -44.38, -44.39, -44.40, -44.40, -44.41, -44.42, -44.42, -44.43, -44.43, -44.44, -44.44, -44.45, -44.45, -
44.46, -44.46, -44.47, -44.47, -44.48, -44.48, -44.48, -44.49, -44.49, -44.50, -44.50, -44.50, -44.51, -44.51, -
44.51, -44.52, -44.52, -44.52, -44.53, -44.53, -44.53, -44.54, -44.54, -44.54, -44.54, -44.55, -44.55, -44.55, -
44.55, -44.56, -44.56, -44.56, -44.56, -44.56, -44.57, -44.57, -44.57, -44.57, -44.57, -44.58, -44.58, -44.58, -
44.58, -44.58, -44.59, -44.59, -44.59, -44.59, -44.59, -44.59, -44.60, -44.60, -44.60, -44.60, -44.60, -44.60, -
44.60, -44.61, -44.61, -44.61, -44.61, -44.61, -44.61, -44.62, -44.62, -44.62, -44.62, -44.62};
float trajetoria[] = {0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.01, 0.02, 0.05, 0.09, 0.14, 0.20, 0.27, 0.35, 0.45, 0.55, 0.67,
0.79, 0.93, 1.08, 1.24, 1.41, 1.60, 1.79, 2.00, 2.21, 2.44, 2.68, 2.93, 3.19, 3.46, 3.74, 4.04, 4.34, 4.66, 4.98,
5.32, 5.67, 6.03, 6.40, 6.77, 7.14, 7.52, 7.89, 8.26, 8.63, 9.00, 9.37, 9.74, 10.11, 10.48, 10.85, 11.22, 11.59,
11.96, 12.33, 12.70, 13.07, 13.44, 13.82, 14.19, 14.56, 14.93, 15.30, 15.67, 16.04, 16.41, 16.78, 17.15, 17.52,
17.89, 18.26, 18.63, 19.00, 19.37, 19.75, 20.12, 20.49, 20.86, 21.23, 21.60, 21.96, 22.32, 22.68, 23.02, 23.36,
23.70, 24.03, 24.36, 24.68, 24.99, 25.31, 25.61, 25.91, 26.21, 26.50, 26.78, 27.06, 27.34, 27.61, 27.88, 28.14,
28.40, 28.66, 28.91, 29.16, 29.40, 29.64, 29.87, 30.10, 30.33, 30.55, 30.77, 30.99, 31.20, 31.41, 31.62, 31.82,
32.02, 32.21, 32.41, 32.60, 32.78, 32.97, 33.15, 33.32, 33.50, 33.67, 33.84, 34.01, 34.17, 34.33, 34.49, 34.64,
34.80, 34.95, 35.10, 35.24, 35.38, 35.53, 35.66, 35.80, 35.94, 36.07, 36.20, 36.33, 36.45, 36.57, 36.70, 36.82,
36.93, 37.05, 37.16, 37.28, 37.39, 37.50, 37.60, 37.71, 37.81, 37.91, 38.01, 38.11, 38.21, 38.31, 38.40, 38.49,
38.58, 38.67, 38.76, 38.85, 38.93, 39.02, 39.10, 39.18, 39.26, 39.33, 39.38, 39.43, 39.46, 39.49, 39.50, 39.50,
39.49, 39.47, 39.44, 39.39, 39.34, 39.27, 39.19, 39.11, 39.01, 38.90, 38.78, 38.64, 38.50, 38.35, 38.18, 38.00,
37.81, 37.62, 37.41, 37.18, 36.95, 36.71, 36.45, 36.19, 35.91, 35.62, 35.32, 35.01, 34.69, 34.36, 34.02, 33.66,
33.29, 32.92, 32.55, 32.18, 31.81, 31.44, 31.07, 30.70, 30.33, 29.96, 29.59, 29.22, 28.85, 28.48, 28.11, 27.74,
27.36, 26.99, 26.62, 26.25, 25.88, 25.51, 25.14, 24.77, 24.40, 24.03, 23.66, 23.29, 22.92, 22.55, 22.18, 21.81,
21.44, 21.06, 20.69, 20.32, 19.95, 19.58, 19.21, 18.84, 18.47, 18.10, 17.73, 17.36, 16.99, 16.62, 16.25, 15.88,
15.51, 15.14, 14.76, 14.39, 14.02, 13.65, 13.28, 12.91, 12.54, 12.17, 11.80, 11.43, 11.06, 10.69, 10.32, 9.95,
9.58, 9.21, 8.83, 8.46, 8.09, 7.72, 7.35, 6.98, 6.61, 6.24, 5.87, 5.50, 5.13, 4.76, 4.39, 4.02, 3.65, 3.28, 2.91,
2.53, 2.16, 1.79, 1.42, 1.05, 0.68, 0.31, -0.06, -0.43, -0.80, -1.17, -1.54, -1.91, -2.28, -2.65, -3.02, -3.39, -
3.77, -4.14, -4.51, -4.88, -5.25, -5.62, -5.99, -6.36, -6.73, -7.10, -7.47, -7.84, -8.21, -8.58, -8.95, -9.32, -
9.70, -10.07, -10.44, -10.81, -11.18, -11.55, -11.92, -12.29, -12.66, -13.03, -13.40, -13.77, -14.14, -14.51, -
14.88, -15.25, -15.62, -16.00, -16.37, -16.74, -17.11, -17.48, -17.85, -18.22, -18.59, -18.96, -19.33, -19.70, -
20.07, -20.44, -20.81, -21.18, -21.55, -21.92, -22.30, -22.67, -23.04, -23.41, -23.78, -24.15, -24.52, -24.89, -
25.26, -25.63, -26.00, -26.37, -26.74, -27.11, -27.48, -27.85, -28.23, -28.60, -28.97, -29.34, -29.71, -30.08, -
30.44, -30.80, -31.15, -31.50, -31.84, -32.17, -32.50, -32.82, -33.14, -33.45, -33.76, -34.06, -34.36, -34.65, -
34.94, -35.22, -35.50, -35.77, -36.04, -36.31, -36.57, -36.82, -37.07, -37.32, -37.56, -37.80, -38.03, -38.26, -
```

38.49, -38.71, -38.93, -39.15, -39.36, -39.56, -39.77, -39.97, -40.17, -40.36, -40.55, -40.74, -40.92, -41.10, -41.28, -41.45, -41.63, -41.79, -41.96, -42.12, -42.28, -42.44, -42.60, -42.75, -42.90, -43.05, -43.19, -43.33, -43.47, -43.61, -43.74, -43.88, -44.01, -44.14, -44.26, -44.39, -44.51, -44.63, -44.74, -44.86, -44.97, -45.09, -45.20, -45.30, -45.41, -45.51, -45.62, -45.72, -45.82, -45.91, -46.01, -46.10, -46.20, -46.29, -46.38, -46.46, -46.55, -46.63, -46.72, -46.80, -46.88, -46.96, -47.04, -47.11, -47.19, -47.26, -47.34, -47.41, -47.48, -47.55, -47.62, -47.68, -47.75, -47.81, -47.88, -47.94, -48.00, -48.06, -48.12, -48.18, -48.24, -48.29, -48.35, -48.40, -48.46, -48.51, -48.56, -48.61, -48.66, -48.71, -48.76, -48.81, -48.85, -48.90, -48.95, -48.99, -49.03, -49.08, -49.12, -49.16, -49.20, -49.24, -49.28, -49.32, -49.36, -49.40, -49.43, -49.47, -49.51, -49.54, -49.58, -49.61, -49.64, -49.68, -49.71, -49.74, -49.77, -49.80, -49.83, -49.86, -49.89, -49.92, -49.95, -49.98, -50.01, -50.03, -50.06, -50.09, -50.11, -50.14, -50.16, -50.19, -50.21, -50.23, -50.25, -50.25, -50.24, -50.22, -50.18, -50.14, -50.09, -50.02, -49.94, -49.85, -49.76, -49.65, -49.52, -49.39, -49.25, -49.09, -48.93, -48.75, -48.56, -48.36, -48.15, -47.93, -47.70, -47.46, -47.20, -46.94, -46.66, -46.37, -46.07, -45.76, -45.44, -45.11, -44.76, -44.41, -44.04, -43.67, -43.30, -42.93, -42.56, -42.19, -41.82, -41.45, -41.08, -40.71, -40.34, -39.97, -39.60, -39.23, -38.86, -38.49, -38.11, -37.74, -37.37, -37.00, -36.63, -36.26, -35.89, -35.52, -35.15, -34.78, -34.41, -34.04, -33.67, -33.30, -32.93, -32.56, -32.18, -31.81, -31.44, -31.07, -30.70, -30.33, -29.96, -29.59, -29.22, -28.85, -28.48, -28.11, -27.74, -27.37, -27.00, -26.63, -26.26, -25.88, -25.51, -25.14, -24.77, -24.40, -24.03, -23.66, -23.29, -22.92, -22.55, -22.18, -21.81, -21.44, -21.07, -20.70, -20.33, -19.96, -19.58, -19.21, -18.84, -18.47, -18.10, -17.73, -17.36, -16.99, -16.62, -16.25, -15.88, -15.51, -15.14, -14.77, -14.40, -14.03, -13.65, -13.28, -12.91, -12.54, -12.17, -11.80, -11.43, -11.06, -10.69, -10.32, -9.95, -9.58, -9.21, -8.84, -8.47, -8.10, -7.73, -7.35, -6.98, -6.61, -6.24, -5.87, -5.50, -5.13, -4.76, -4.39, -4.02, -3.65, -3.28, -2.91, -2.54, -2.17, -1.80, -1.43, -1.05, -0.68, -0.31, 0.06, 0.43, 0.80, 1.17, 1.54, 1.91, 2.28, 2.65, 3.02, 3.39, 3.76, 4.13, 4.50, 4.87, 5.25, 5.62, 5.99, 6.36, 6.73, 7.10, 7.47, 7.84, 8.21, 8.58, 8.95, 9.32, 9.69, 10.06, 10.43, 10.80, 11.18, 11.55, 11.92, 12.29, 12.66, 13.03, 13.40, 13.77, 14.14, 14.51, 14.88, 15.25, 15.62, 15.99, 16.36, 16.73, 17.10, 17.48, 17.85, 18.22, 18.59, 18.96, 19.33, 19.70, 20.07, 20.44, 20.81, 21.18, 21.55, 21.92, 22.29, 22.66, 23.03, 23.40, 23.78, 24.15, 24.52, 24.89, 25.26, 25.63, 26.00, 26.37, 26.74, 27.11, 27.48, 27.85, 28.22, 28.59, 28.96, 29.33, 29.71, 30.08, 30.45, 30.81, 31.17, 31.53, 31.88, 32.22, 32.56, 32.89, 33.22, 33.54, 33.85, 34.16, 34.47, 34.77, 35.06, 35.35, 35.64, 35.92, 36.19, 36.47, 36.73, 36.99, 37.25, 37.51, 37.75, 38.00, 38.24, 38.48, 38.71, 38.94, 39.16, 39.39, 39.60, 39.82, 40.03, 40.23, 40.44, 40.64, 40.83, 41.03, 41.22, 41.40, 41.59, 41.77, 41.94, 42.12, 42.29, 42.46, 42.62, 42.79, 42.95, 43.11, 43.26, 43.41, 43.56, 43.71, 43.85, 44.00, 44.14, 44.27, 44.41, 44.54, 44.67, 44.80, 44.93, 45.05, 45.17, 45.29, 45.41, 45.53, 45.64, 45.75, 45.86, 45.97, 46.08, 46.18, 46.28, 46.39, 46.49, 46.58, 46.68, 46.77, 46.87, 46.96, 47.05, 47.14, 47.22, 47.31, 47.39, 47.48, 47.56, 47.64, 47.72, 47.79, 47.87, 47.95, 48.02, 48.09, 48.16, 48.23, 48.30, 48.37, 48.44, 48.50, 48.56, 48.63, 48.69, 48.75, 48.81, 48.87, 48.93, 48.99, 49.04, 49.10, 49.15, 49.21, 49.26, 49.31, 49.36, 49.41, 49.46, 49.51, 49.56, 49.60, 49.65, 49.70, 49.74, 49.79, 49.83, 49.87, 49.91, 49.95, 49.99, 50.03, 50.07, 50.11, 50.15, 50.19, 50.22, 50.26, 50.30, 50.33, 50.37, 50.40, 50.43, 50.47, 50.50, 50.53, 50.56, 50.59, 50.62, 50.65, 50.68, 50.71, 50.74, 50.77, 50.79, 50.82, 50.83, 50.84, 50.83, 50.81, 50.78, 50.74, 50.69, 50.62, 50.55, 50.46, 50.37, 50.26, 50.14, 50.01, 49.87, 49.72, 49.55, 49.38, 49.19, 48.99, 48.79, 48.57, 48.34, 48.10, 47.84, 47.58, 47.31, 47.02, 46.72, 46.42, 46.10, 45.77, 45.42, 45.07, 44.71, 44.34, 43.97, 43.60, 43.23, 42.86, 42.48, 42.11, 41.74, 41.37, 41.00, 40.63, 40.26, 39.89, 39.52, 39.15, 38.78, 38.41, 38.04, 37.67, 37.30, 36.93, 36.56, 36.18, 35.81, 35.44, 35.07, 34.70, 34.33, 33.96, 33.59, 33.22, 32.85, 32.48, 32.11, 31.74, 31.37, 31.00, 30.63, 30.25, 29.88, 29.51, 29.14, 28.77, 28.40, 28.03, 27.66, 27.29, 26.92, 26.55, 26.18, 25.81, 25.44, 25.07, 24.70, 24.33, 23.95, 23.58, 23.21, 22.84, 22.47, 22.10, 21.73, 21.36, 20.99, 20.62, 20.25, 19.88, 19.51, 19.14, 18.77, 18.40, 18.03, 17.65, 17.28, 16.91, 16.54, 16.17, 15.80, 15.43, 15.06, 14.69, 14.32, 13.95, 13.58, 13.21, 12.84, 12.47, 12.10, 11.72, 11.35, 10.98, 10.61, 10.24, 9.87, 9.50, 9.13, 8.76, 8.39, 8.02, 7.65, 7.28, 6.91, 6.54, 6.17, 5.80, 5.42, 5.05, 4.68, 4.31, 3.94, 3.57, 3.20, 2.83, 2.46, 2.09, 1.72, 1.35, 0.98, 0.61, 0.24, -0.13, -0.50, -0.88, -1.25, -1.62, -1.99, -2.36, -2.73, -3.10, -3.47, -3.84, -4.21, -4.58, -4.95, -5.32, -5.69, -6.06, -6.43, -6.80, -7.18, -7.55, -7.92, -8.29, -8.66, -9.03, -

9.40, -9.77, -10.14, -10.51, -10.88, -11.25, -11.62, -11.99, -12.36, -12.73, -13.11, -13.48, -13.85, -14.22, -14.59, -14.96, -15.33, -15.70, -16.07, -16.44, -16.81, -17.18, -17.54, -17.90, -18.26, -18.60, -18.95, -19.29, -19.62, -19.95, -20.27, -20.59, -20.91, -21.22, -21.52, -21.82, -22.12, -22.41, -22.70, -22.98, -23.26, -23.54, -23.81, -24.08, -24.34, -24.60, -24.86, -25.11, -25.36, -25.61, -25.85, -26.09, -26.33, -26.56, -26.79, -27.01, -27.24, -27.46, -27.67, -27.88, -28.10, -28.30, -28.51, -28.71, -28.91, -29.10, -29.30, -29.49, -29.68, -29.86, -30.04, -30.23, -30.40, -30.58, -30.75, -30.92, -31.09, -31.26, -31.42, -31.59, -31.75, -31.90, -32.06, -32.21, -32.36, -32.51, -32.66, -32.81, -32.95, -33.10, -33.24, -33.37, -33.51, -33.65, -33.78, -33.91, -34.04, -34.17, -34.30, -34.43, -34.55, -34.67, -34.79, -34.91, -35.03, -35.15, -35.26, -35.35, -35.44, -35.51, -35.58, -35.63, -35.67, -35.70, -35.72, -35.73, -35.72, -35.71, -35.68, -35.65, -35.60, -35.54, -35.47, -35.39, -35.29, -35.19, -35.08, -34.95, -34.81, -34.67, -34.51, -34.34, -34.16, -33.96, -33.76, -33.55, -33.32, -33.08, -32.83, -32.58, -32.31, -32.02, -31.73, -31.43, -31.11, -30.79, -30.45, -30.10, -29.74, -29.37, -29.00, -28.63, -28.26, -27.89, -27.52, -27.15, -26.78, -26.41, -26.05, -25.70, -25.35, -25.00, -24.66, -24.33, -24.00, -23.68, -23.36, -23.04, -22.73, -22.43, -22.13, -21.84, -21.55, -21.26, -20.98, -20.70, -20.43, -20.16, -19.90, -19.64, -19.38, -19.13, -18.88, -18.64, -18.40, -18.16, -17.93, -17.70, -17.47, -17.25, -17.03, -16.82, -16.60, -16.39, -16.19, -15.99, -15.79, -15.59, -15.40, -15.21, -15.02, -14.84, -14.65, -14.48, -14.30, -14.13, -13.96, -13.79, -13.62, -13.46, -13.30, -13.14, -12.99, -12.83, -12.68, -12.53, -12.39, -12.24, -12.10, -11.96, -11.82, -11.69, -11.55, -11.42, -11.29, -11.16, -11.04, -10.92, -10.79, -10.67, -10.55, -10.44, -10.32, -10.21, -10.10, -9.99, -9.88, -9.77, -9.67, -9.56, -9.46, -9.36, -9.26, -9.16, -9.07, -8.97, -8.88, -8.79, -8.69, -8.61, -8.52, -8.43, -8.34, -8.26, -8.18, -8.09, -8.01, -7.93, -7.85, -7.77, -7.70, -7.62, -7.55, -7.47, -7.40, -7.33, -7.26, -7.19, -7.12, -7.05, -6.98, -6.92, -6.85, -6.79, -6.72, -6.66, -6.60, -6.54, -6.48, -6.42, -6.36, -6.30, -6.24, -6.19, -6.13, -6.08, -6.02, -5.97, -5.91, -5.86, -5.81, -5.76, -5.71, -5.66, -5.61, -5.56, -5.51, -5.46, -5.41, -5.37, -5.32, -5.27, -5.23, -5.18, -5.14, -5.10, -5.05, -5.01, -4.97, -4.92, -4.88, -4.84, -4.80, -4.76, -4.72, -4.68, -4.64, -4.60, -4.56, -4.52, -4.49, -4.45, -4.41, -4.37, -4.34, -4.30, -4.26, -4.23, -4.19, -4.15, -4.12, -4.08, -4.05, -4.01, -3.98, -3.94, -3.91, -3.87, -3.84, -3.80, -3.77, -3.74, -3.70, -3.67, -3.63, -3.60, -3.56, -3.53, -3.50, -3.46, -3.43, -

```
3.39, -3.36, -3.32, -3.29, -3.25, -3.21, -3.18, -3.14, -3.10, -3.06, -3.02, -2.98, -2.94, -2.90, -2.86, -2.81, -
2.76, -2.71, -2.66, -2.60, -2.54, -2.47, -2.40, -2.31, -2.21, -2.10};

//====  Declarando Funções das tarefas RTOS:  ================================================
TaskHandle_t Task1, Task2, Task3;
SemaphoreHandle_t baton;



//============================================================================
// LOOP #1 (Leitura de Mag, Conta as medições do sensor IR de velocidadeX, calcula o magnetômetro)
//============================================================================
void codeForTask1( void * parameter )
{
  for (;;) {

//functionRead_X_Speed()
    leitura = digitalRead(Dhall);
//Contagem das leituras para contar a velocidade
    if(leitura == 1){
      A = 0;
    }
    else if(leitura == 0){
      A = 1;
     }

    if (B==0 && A==1){
      if(ContState==0){
        Cont++;
        ContState = 1;
      }
      else{
        ContState = 0;
        Cont++;
      }
    }
    B = A;
    delay(1);
  }
}



//============================================================================
//  SETUP
//============================================================================
void setup() {
//====  Setup do FreeRTOs  ===========================================
```

```
  baton = xSemaphoreCreateMutex();
  xTaskCreatePinnedToCore(
    codeForTask1,
    "Main Code",
    1000,
    NULL,
    1,
    &Task1,
    1);
  delay(500);  // needed to start-up task1

//====  Setup do Magnetômetro  ======================================
  // inicializa a biblioteca para comunicação I2C
  Wire.begin();
  // inicializa o Magnetômetro
  Serial.println("Initializing I2C devices...");
  mag.initialize();
  // verify connection
  Serial.println("Testing device connections...");
  Serial.println(mag.testConnection() ? "HMC5883L connection successful" : "HMC5883L connection failed");
  //Ativando o magnetômetro no HMC5883L
  Wire.beginTransmission(0x68);
  Wire.write(0x37);
  Wire.write(0x02);
  Wire.endTransmission();
  //
  Wire.beginTransmission(0x68);
  Wire.write(0x6A);
  Wire.write(0x00);
  Wire.endTransmission();
  //Desabilitando o Sleep Mode
  Wire.beginTransmission(0x68);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission();


//====Inicia Serial e seta o pino do velocímetro/odometro==============
  Serial.begin(115200);
  pinMode(Dhall, INPUT);

//====  Setando os Pinos  dos Servos ====================================
  motorDC.attach(27);   //fio AMARELO //27
  direcao.attach(33);   //fio LARANJA
  marcha.attach(32);    //fio VERDE
  difFront.attach(19);  //fio ROXO //21
```

```
  difTras.attach(18);   //fio MARROM

//====  Servos no neutro/Diferenciais destravados e Marcha Trativa =====
  motorDC.writeMicroseconds(1451);
  direcao.writeMicroseconds(1512);
  marcha.writeMicroseconds(2001);
  difFront.writeMicroseconds(2001);
  difTras.writeMicroseconds(1000);

//====  Teste de direção, termina reta ================================
    direcao.writeMicroseconds(1451);    //Direção reta
        delay(1000);
        direcao.write(44);
        delay(500);
        direcao.write(139);
        delay(500);
        direcao.write(90);
        delay(1000);

//====  Armando o ESC do Motor DC =====================================
   for (int angleT = 90; angleT <= 100; angleT++) {
      motorDC.write(angleT);
      delay(150);
   }
  motorDC.writeMicroseconds(1512);    //Motor parado

//====  Liga o motor ==================================================
    delay(3000);
    Motor_Ligado = true;

//====  Reseta o Odômetro =============================================
    ContOld = 0;
    Cont = 0;

//====  Extrai a média da leitura do Magnetômetro e chama de zero =======
//functionRead_MAG_HEADING
     mag.getHeading(&mx1, &my1, &mz1);
     delay(100);
     mag.getHeading(&mx2, &my2, &mz2);
     delay(100);
     mag.getHeading(&mx3, &my3, &mz3);
     delay(100);
     mag.getHeading(&mx4, &my4, &mz4);
     delay(100);

     heading1 = atan2(my1, mx1);
```

```
      if(heading1 < 0){
      heading1 += 2 * M_PI;
      }
      heading2 = atan2(my2, mx2);
      if(heading2 < 0){
      heading2 += 2 * M_PI;
      }
      heading3 = atan2(my3, mx3);
      if(heading3 < 0){
      heading3 += 2 * M_PI;
      }
      heading4 = atan2(my4, mx4);
      if(heading4 < 0){
      heading4 += 2 * M_PI;
      }
      headingM = (heading1+heading2+heading3+heading4)/4;

      //pitchTarget = headingM;
     delay(3000);


     Time = millis();
     milliNew=millis();
}

//================================================================================
// VOID LOOP (dps do setup) (Loop 2)
//================================================================================
void loop() {
       while(Motor_Ligado == true){
            while(m < 16.1  ){ //16.11m se for o programa do circulo, 5.84, 16.10
               motorDC.attach(27);

               //motorDC.write(104); //Agora o motor DC está controlado pelo PID

     //functionRead_MAG_HEADING
     mag.getHeading(&mx, &my, &mz);

     //Cálculos do Magnetômetro
     heading = atan2(my, mx);
     if(heading < 0){
       heading += 2 * M_PI;
     }
     headingR = (heading * 180/M_PI) - (headingM * 180/M_PI);

     //Delta_t do velocimetro/odometro
```

```
if(Cont!=ContOld){
  timeOld = Time;
  Time = millis();
  tempo_de_Leitura = (Time - timeOld);
}
if(tempo_de_Leitura == 0){
  tempo_de_Leitura = 0.001;
}

//Cálculos do Odômetro
REV = Cont - ContOld;
if(REV != 0){
  REVs = REV;
}
RPS = ((REVs / tempo_de_Leitura)* 1000);
RPM = RPS*60;
mPs = RPS / razao_eixo_roda * Circ_roda;
kmPh = mPs * 3.6;
m = Cont * Circ_roda / razao_eixo_roda ;
ContOld = Cont;
Vlon = kmPh;

//Cálculos do Magnetômetro com PID
pitchActual = headingR;

//Cálculos do velocimetro com PID
rollActual = Vlon; //Mostra a velocidade real


//Pega o angulo de acordo com o cm
input = m * 100;
//pitchTarget = 0;
 pitchTarget = - (trajetoria[input]);

//Cálculo do Tempo de resposta do PID
milliOld=milliNew;
milliNew=millis();
dt=milliNew-milliOld;

//Cálculo de cada termo do PID da direção
pitchErrorOld=pitchError;
pitchError=pitchTarget-pitchActual;
pitchErrorChange=pitchError-pitchErrorOld;
pitchErrorSlope=pitchErrorChange/dt;
pitchErrorArea=pitchErrorArea+pitchError*dt;
```

```
    //Cálculo de cada termo do PID da velocidade
    rollErrorOld=rollError;
    rollError=rollTarget-rollActual;
    rollErrorChange=rollError-rollErrorOld;
    rollErrorSlope=rollErrorChange/dt;
    rollErrorArea=rollErrorArea+rollError*dt;

    //Cálculo e limitação do SP de direção
    //pitchServoVal=94 + k1*pitchError + k2*pitchErrorSlope + k3*pitchErrorArea;
    pitchServoVal=pitchServoVal + k1*pitchError + k2*pitchErrorSlope + k3*pitchErrorArea;
    if(pitchServoVal >= 180){
      pitchServoVal=180;
    }
    if(pitchServoVal <= 0){
      pitchServoVal=0;
    }
    //Serial.print("                                                pitchServoVal: ");
    //Serial.print(pitchServoVal);
    //Serial.println("º");
    direcao.write(pitchServoVal);
    //delay(2);

    //Cálculo e limitação do SP de velocidade
    rollServoVal=rollServoVal + kp*rollError + kd*rollErrorSlope + ki*rollErrorArea;
    if(rollServoVal >= 180){
      rollServoVal=180;
    }
    if(rollServoVal <= 94){
      rollServoVal=94;
    }
    //Serial.print("                                                rollServoVal: ");
    //Serial.print(rollServoVal);
    //Serial.println("  ");
    motorDC.write(rollServoVal);
    //delay(2);


      //Dados do Odômetro
//      Serial.println("===============");
//      Serial.print("Motor ligado: ");
//      Serial.println(Motor_Ligado);
//      Serial.print("VOLTAS: ");
//      Serial.println(Cont);
//      Serial.print("TEMPO DE LEITURA: ");
//      Serial.println(tempo_de_Leitura);
//      Serial.print("REVs: ");
```

```
//      Serial.println(REVs);
//      Serial.print("REV: ");
//      Serial.println(REV);
//      Serial.print("RPS: ");
//      Serial.println(RPS);
//      Serial.print("RPM: ");
//      Serial.println(RPM);
//      Serial.print("m/s: ");
//      Serial.println(mPs);
//      Serial.print("Dist: ");
//      Serial.print(m);
//      Serial.println(" m");
//      Serial.print("TEMPO: ");
//      Serial.print(tempo_de_Leitura);
//      Serial.println(" ms");
//      Serial.print("Vlon: ");
//      Serial.print(Vlon);
//      //Serial.println("  ");
//      Serial.println("  km/h");
//      Serial.print("               TG: ");
//      Serial.print(pitchTarget);
//      Serial.print("                       ACT: ");
//      Serial.print(pitchActual);
//      Serial.print("                           pSVal: ");
//      Serial.print(pitchServoVal);
//      Serial.print("                                    E: ");
//      Serial.println(pitchError);

        }
        Motor_Ligado = false;

    }
        motorDC.write(94);
        motorDC.detach();

        direcao.write(94);
        delay(1000);
        direcao.detach();
}
```