



Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia Automotiva

DESENVOLVIMENTO DE ARQUITETURA ELETROELETRÔNICA PARA VEÍCULO DE PEQUENA ESCALA

Autor: João Vítor Cavalcanti Duarte e Pedro Henrique
Lourenço Figueiredo

Orientador: Me. Rafael Rodrigues da Silva

Brasília, DF

2022



João Vítor Cavalcanti Duarte e Pedro Henrique Lourenço Figueiredo

DESENVOLVIMENTO DE ARQUITETURA ELETROELETRÔNICA PARA VEÍCULO DE PEQUENA ESCALA

Monografia submetida ao curso de graduação em Engenharia Automotiva da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Automotiva.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Me. Rafael Rodrigues da Silva

Brasília, DF

2022

João Vítor Cavalcanti Duarte e Pedro Henrique Lourenço Figueiredo
DESENVOLVIMENTO DE ARQUITETURA ELETROELETRÔNICA
PARA VEÍCULO DE PEQUENA ESCALA/ João Vítor Cavalcanti Duarte e Pe-
dro Henrique Lourenço Figueiredo. – Brasília, DF, 2022-
128 p. : il. (algumas color.) ; 30 cm.

Orientador: Me. Rafael Rodrigues da Silva

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2022.

1. Palavra-chave01. 2. Palavra-chave02. I. Me. Rafael Rodrigues da Silva.
II. Universidade de Brasília. III. Faculdade UnB Gama. IV. DESENVOLVI-
MENTO DE ARQUITETURA ELETROELETRÔNICA PARA VEÍCULO DE
PEQUENA ESCALA

CDU 02:141:005.6

João Vítor Cavalcanti Duarte e Pedro Henrique Lourenço Figueiredo

DESENVOLVIMENTO DE ARQUITETURA ELETROELETRÔNICA PARA VEÍCULO DE PEQUENA ESCALA

Monografia submetida ao curso de graduação em Engenharia Automotiva da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Automotiva.

Brasília, DF, 24 de setembro de 2022:

Me. Rafael Rodrigues da Silva
Orientador

Dr. André Murilo de Almeida Pinto
Convidado 1

Dr. Evandro Leonardo Silva Teixeira
Convidado 2

Brasília, DF
2022

Agradecimentos

João Vítor: *A Deus, que agraciou-me com as dádivas da saúde e da persistência. À minha família, que desde sempre me apoiou nos meus sonhos e escolhas, fazendo o que era possível e impossível para que eu me sentisse realizado. Aos meus amigos, que em momentos de desânimo estiveram aqui para incentivar e apoiar. Aos meus colegas e também amigos de laboratório, sem os quais muitas etapas não seriam cumpridas e sem os quais o ânimo de trabalhar não seria o mesmo. Aos meus professores, especialmente meu orientador, que esteve ao meu lado desde o começo, oferecendo ajuda, apoio, conhecimento, companheirismo e, principalmente, oportunidades.*

Pedro Figueiredo: *Agradeço primeiramente a Deus por tudo o que me deu e tudo o que me permite conquistar. Agradeço aos meus pais Antônio e Gizeuda por todo o amparo e ensinamentos, pois sem eles eu não me tornaria quem sou hoje e nem conseguiria ter chegado até este ponto. Agradeço aos meus irmãos que sempre me ajudam e me apoiam, obrigado por acreditam em mim. Agradeço à minha namorada Ana Flávia por sempre estar comigo nos momentos difíceis, pelo apoio e companheirismo ao longo desses cinco anos. Agradeço aos meus amigos e colegas, companheiros de laboratório pela ajuda e por nos agregar ânimo e também conhecimento. Agradeço aos meus professores por partilharem seu conhecimento, especialmente ao Prof. Rafael Silva que nos acolheu, partilha seu conhecimento sem egoísmo e sempre busca ajudar.*

*“Meus irmãos, considerem motivo de grande alegria o fato de passarem por diversas
provações, pois vocês sabem que a prova da sua fé produz perseverança. E a perseverança
deve ter ação completa, a fim de que vocês sejam maduros e íntegros, sem que falte a
vocês coisa alguma.*

Resumo

Com a crescente tendência de eletrificação e de automação dos veículos, surge a grande demanda por sistemas de controle automotivo em seus diversos subsistemas. Neste estudo, são abordados os sistemas de direção e de aceleração e frenagem, que estão presentes em todos os carros comercializados no mundo atualmente e que são de indispensável aplicação para a segurança e para o conforto. Nesse sentido, são desenvolvidos dois sistemas de controle baseados em controladores PID para um veículo de pequena escala e uma rede de comunicação entre essas unidades de controle, por meio do protocolo CAN. Desse modo, este trabalho explora ferramentas virtuais voltadas para a modelagem e simulação dos sistemas de controle, bem como para a comunicação entre as unidades de controle, baseando-se na teoria de controle de sistemas. Assim, foi possível obter resultados demonstrativos do comportamento dos subsistemas quando submetidos a diferentes tipos de entrada, bem como submeter o veículo de pequena escala, com os sistemas embarcados, a testes de validação.

Palavras-chaves: Automação, Direção, Aceleração e Frenagem, Controle, CAN.

Abstract

With the growing trend of electrification and automation of vehicles, there is a great demand for automotive control systems in their various subsystems. In this study, the steering, acceleration and braking systems are present, which are present in practically all cars produced in the world today and which are of indispensable application for safety and comfort. In the meantime, for a small-scale vehicle, two control systems based on PID controllers are developed, one for each subsystem, and a communication network between these control units, through the CAN protocol. In this sense, this work uses virtual tools to modeling and simulating of control systems, as well as to communication between the control units, based on the theory of control systems. Thus, it was possible to obtain results demonstrating the behavior of the subsystems when submitted to different types of inputs, as well as submitting the small-scale vehicle, with embedded systems, to validation tests.

Key-words: Automation, Steering, Acceleration and Braking, Control, CAN.

Lista de ilustrações

Figura 1 – LKAS e AEBS	18
Figura 2 – Veículo de pequena escala	19
Figura 3 – Geometria de Ackerman	24
Figura 4 – Geometria das barras do veículo de pequena escala	25
Figura 5 – Modelo caixa de redução	26
Figura 6 – Forças significativas sobre um veículo	28
Figura 7 – Esquemático de um motor de corrente contínua acoplado a uma carga	29
Figura 8 – Exemplo de sistema de controle de malha fechada	31
Figura 9 – Regiões de estabilidade e instabilidade no plano complexo	34
Figura 10 – Influência dos ganhos na resposta do sistema	36
Figura 11 – Funcionamento do método de agendamento de ganhos	38
Figura 12 – Arquitetura Centralizada e Distribuída	39
Figura 13 – Relação entre o modelo OSI e os padrões ISO 11898 e ISO 11519	41
Figura 14 – Níveis lógicos obtidos através da tensão diferencial no barramento	41
Figura 15 – Relação entre a taxa de transmissão de dados e o comprimento do barramento CAN	42
Figura 16 – Método de arbitragem utilizado no barramento CAN para concessão do acesso ao barramento	43
Figura 17 – Campos de um <i>frame</i> de dados da CAN 2.0A	44
Figura 18 – Sistema de medição para verificação de grandezas	46
Figura 19 – Sistema de medição para aplicações em malha fechada	46
Figura 20 – Impressora 3D Tevo Tornado	49
Figura 21 – Componentes	50
Figura 22 – Sistema mecânico da direção montado	50
Figura 23 – Esquemático do experimento	51
Figura 24 – Procedimento real	51
Figura 25 – Média das medições para os ângulos de esterçamento das rodas	52
Figura 26 – Fluxograma da modelagem mecânica da direção	52
Figura 27 – Componentes do sistema de aceleração e frenagem	53
Figura 28 – Conjunto resultante da união dos componentes	53
Figura 29 – Posicionamento Sensor-Disco	54
Figura 30 – Componentes principais da arquitetura eletroeletrônica	55
Figura 31 – ECUs construídas	55
Figura 32 – Montagem do sistema de direção	56
Figura 33 – Montagem do sistema de aceleração e frenagem	57
Figura 34 – Disposição de dados na mensagem	60

Figura 35 – Fluxograma representativo da transformação da escala do ângulo de direção	61
Figura 36 – Fluxograma representativo da aplicação da ação de controle no motor CC	61
Figura 37 – Fluxograma da lógica de controle para a direção	62
Figura 38 – Fluxograma do código da ABM	62
Figura 39 – Fluxograma da lógica de agendamento de ganhos	64
Figura 40 – Fluxograma de Aquisição dos Ganhos	66
Figura 41 – Gráfico PWM e ângulo x tempo	67
Figura 42 – Funções de Transferência	68
Figura 43 – Dados coletados em malha aberta - Ensaio frente	69
Figura 44 – Dados coletados em malha aberta - Ensaio trás	69
Figura 45 – Resposta do sistema para uma entrada degrau de 15° para a esquerda .	72
Figura 46 – Resposta do sistema para uma entrada degrau de 10° para a esquerda .	73
Figura 47 – Resposta do sistema para uma entrada degrau de 0° a 20° para a direita	74
Figura 48 – Resposta do sistema para uma entrada degrau de 15° a 0°	75
Figura 49 – Entrada degrau utilizada para geração dos resultados	77
Figura 50 – Estrutura base utilizada para plotagem dos resultados	77
Figura 51 – Resultado primeiro ensaio	78
Figura 52 – Resultado segundo ensaio	78
Figura 53 – Resultado terceiro ensaio	79
Figura 54 – Resultado quarto ensaio	79
Figura 55 – Resultado quinto ensaio	80
Figura 56 – Resultado sexto ensaio	80

Lista de tabelas

Tabela 1 – Funcionalidade de cada campo de dados de um <i>frame</i> padrão	44
Tabela 2 – Componentes Arquitetura eletroeletrônica	59
Tabela 3 – Dicionário de dados	63
Tabela 4 – Matriz de comunicação	63

Lista de abreviaturas e siglas

ACK	Acknowledge
ADAS	Advanced Driver Assistance Systems
AEBS	Advanced Emergency Braking System
CAN	Controller Area Network
CRC	Cyclic Redundance Check
CSLL	Contribuição Social sobre o Lucro Líquido
CSMA/CD	Carrier Sense Multiple Access with Collision Detect
DLC	Data Length Code
ECU	Eletronic Control Unit
EOF	End of Frame
ICTs	Instituições Científicas, Tecnológicas e de Inovação
IDE	Identifier Extended Bit
IPI	Imposto sobre Produtos Industrializados
IRPJ	Imposto de Renda - Pessoa Jurídica
ISO	International Organization for Standardization
LDWS	Lane Departure Warning System
LKAS	Lane Keeping Assist System
M_p	Sobressinal
ONSV	Observatório Nacional de Segurança Viária
OSI	Open Systems Interconnection
RPM	Rotações por minuto
RTR	Remote Transmission Request
SOF	Start of Frame

T_d	Tempo de atraso
T_p	Tempo de pico
T_r	Tempo de subida
T_s	Tempo de assentamento

Lista de símbolos

θ_c	Ângulo de esterçamento da coluna de direção
θ_r	Ângulo de esterçamento das rodas
V	Volt
L	Entre eixos
t	Bitola
R	Raio de giro
δ_o	Ângulo da roda externa
δ_i	Ângulo da roda interna
R_{cr}	Relação entre coluna de direção e rodas
W	Força peso
F_{xf}	Força trativa no eixo dianteiro
F_{xr}	Força trativa no eixo traseiro
R_{xf}	Resistência ao rolamento no eixo dianteiro
R_{xr}	Resistência ao rolamento no eixo traseiro
W_f	Força normal no eixo dianteiro
W_r	Força normal no eixo traseiro
D_a	Força de arrasto aerodinâmico
θ	Ângulo de inclinação da rampa
m	Massa do veículo
a	Aceleração do veículo
f	Coefficiente de resistência ao rolamento
ρ	Densidade do ar
C_D	Coefficiente de arrasto aerodinâmico

A	Área de contato do ar com o veículo
u	Velocidade do veículo
u_w	Velocidade do vento
V_b	Tensão na bobina do motor CC
$\dot{\theta}_m$	Velocidade angular de saída do motor CC
T_m	Torque de saída do motor
i	Corrente elétrica
k_b	Constante de proporcionalidade
k_t	Constante de proporcionalidade
L	Indutância
R	Resistência
J_m	Momento de Inércia do motor
$\ddot{\theta}_m$	Aceleração angular de saída do motor CC
c_m	Coefficiente de atrito viscoso
θ_m	Posição angular de saída do motor CC
$V(s)$	Tensão de alimentação
ω_n	Frequência natural
ξ	Fator de amortecimento
ω_d	Frequência natural amortecida
e_{ss}	Erro em regime permanente
k_p	Constante proporcional
k_i	Constante integrativa
k_d	Constante derivativa
K_{cr}	Valor crítico de constante
P_{cr}	Período crítico

Sumário

1	INTRODUÇÃO	17
1.1	Objetivos	19
1.2	Organização do trabalho	20
2	FUNDAMENTAÇÃO TEÓRICA	21
2.1	Estado da Arte	21
2.1.1	Modelo de pequena escala autônomo	21
2.1.2	Modelo de pequena escala com comando via <i>Bluetooth</i>	21
2.1.3	Modelo de pequena escala autônomo com uso de controle LQR	22
2.1.4	Modelo de pequena escala com uso de um <i>datalogger</i>	22
2.1.5	Veículo de pequena escala autônomo capaz de seguir uma trajetória poligonal sem repetição	22
2.1.6	Veículo inteligente de pequena escala semi-autônomo	23
2.2	Sistemas de direção	23
2.2.1	Modelo da Caixa de Redução	26
2.3	Sistema de Aceleração e Frenagem	27
2.3.1	Modelo da Dinâmica Longitudinal	27
2.3.2	Modelo do Trem de Força	29
2.4	Sistemas de Controle e Controladores	30
2.4.1	Sistemas de Controle	30
2.4.2	Modelagem matemática	31
2.4.2.1	Funções de Transferência e Espaço de Estados	31
2.4.3	Estabilidade e Desempenho	33
2.4.4	Tipos de Controladores	34
2.4.5	Agendamento de Ganhos	37
2.5	Rede CAN	38
2.5.1	Conceito	38
2.5.2	Funcionamento	40
2.5.2.1	Camada Física	40
2.5.2.2	Camada de Enlace	42
2.6	Instrumentação	45
2.7	Identificação de Sistemas	47
3	METODOLOGIA	49
3.1	Modelagem Mecânica	49
3.2	Arquitetura Eletroeletrônica	54

3.3	Desenvolvimento da Lógica de Controle	60
3.3.1	Sistema de Direção	60
3.3.2	Sistema de Aceleração e frenagem	62
3.3.2.1	Ciclo de Desenvolvimento do Controlador	64
3.4	Testes e Validação	65
3.4.1	Sistema de Direção	67
3.4.2	Sistema de Aceleração e Frenagem	68
4	RESULTADOS	71
4.1	Sistema de Direção	71
4.2	Sistema de Aceleração e Frenagem	76
5	CONCLUSÃO	82
	REFERÊNCIAS	84
	APÊNDICES	86
	APÊNDICE A – CÓDIGOS DE CONTROLE DOS SISTEMAS DE DIREÇÃO E DE ACELERAÇÃO E FRENAGEM	87
	APÊNDICE B – DESENHOS TÉCNICOS DOS COMPONENTES DO SISTEMA DE DIREÇÃO	108
	APÊNDICE C – DESENHOS TÉCNICOS DOS COMPONENTES DO SISTEMA DE ACELERAÇÃO E FRENAGEM	115
	APÊNDICE D – CÓDIGOS DE CARACTERIZAÇÃO DOS MOTORES	120
	ANEXOS	127
	ANEXO A – TABELA DAS PRINCIPAIS TRANSFORMADAS DE LAPLACE	128

1 Introdução

O programa ROTA 2030 foi concebido a partir da consolidação da lei nº 13.755/18. O principal objetivo desse programa é elaborar uma política industrial de longo prazo para o setor automotivo e de autopeças, estimulando o investimento e o fortalecimento das empresas brasileiras do setor. A maior vantagem para as empresas do ramo automotivo em adotar o programa está relacionada à atenuação da carga tributária relacionada aos produtos (IPI), ao imposto de renda (IRPJ) e à contribuição social sobre lucro líquido (CSLL) (FIGROUP, 2021). Dentre os vários desafios existentes na mobilidade e logística brasileira, o programa visa atuar em algumas frentes, tais como:

- Na baixa competitividade da indústria automotiva nacional;
- Na defasagem tecnológica, especialmente em eficiência energética, desempenho estrutural e tecnologias assistivas à direção, do produto nacional frente às novas tecnologias, já implementadas nos grandes mercados dos países desenvolvidos;
- Na transferência das atividades de P&D (Pesquisa e Desenvolvimento) para o território nacional.

A frente de pesquisa e desenvolvimento do setor automotivo do ROTA 2030 ficou estabelecida por meio do consórcio entre Institutos de Ciência e Tecnologia (ICTs) e empresas privadas do setor automobilístico, com o objetivo de desenvolver tecnologias de rotulagem veicular, de eficiência energética veicular e de segurança veicular associada às tecnologias assistivas à direção. Dentre os diversos grupos formados para o desenvolvimento de projetos, existe o SegurAuto (Projeto e Desenvolvimento Integrado de Funções de Segurança Assistida ao Condutor e Ambiente para Veículos Autônomos), formado pela parceria entre empresas do setor e universidades, cuja principal finalidade é desenvolver dispositivos e sistemas de segurança e de assistência ativa para veículos, por meio da utilização de sistemas ADAS (*Advanced Driver Assistance System*), permitindo, assim, a implementação de funções de auxílio ao condutor.

De acordo com Rajamani (2006), há uma variedade de sistemas de assistência ao condutor para proporcionar conforto e, sobretudo, evitar acidentes que, segundo dados do ONSV (Observatório Nacional de Segurança Viária), têm o fator humano como determinante em mais de 90% dos casos (ONSV, 2015). Nesse ínterim, dois dos sistemas que surgiram foram os de controle do sistema de direção e do de controle longitudinal do veículo, que fazem parte do sistema ADAS, cujo objetivo é também efetivar o sistema de estabilidade do carro, tanto em relação à derrapagem e ao deslizamento quanto ao

capotamento. O desenvolvimento e a aplicação do sistema de direção e de frenagem para o veículo de pequena escala são mandatórios, pois representarão os comandos do veículo, sendo, assim, o foco deste trabalho.

As diversas frentes de atividade deste projeto foram divididas entre as ICTs envolvidas, sendo elas: UnB, USP, UFPE e UTFPR. A Universidade de Brasília tem como objetivo desenvolver, em ambiente acadêmico, duas das principais funções que auxiliam o condutor, que são: LKAS (*Lane Keeping Assist System*) e AEBS (*Advanced Emergency Braking System*). A primeira consiste na assistência ao condutor em manter o veículo em uma trajetória retilínea, de forma a impedir deslocamentos transversais para fora da faixa. Segundo [Ye \(2019\)](#), o LKAS, feito por meio da atuação no sistema de direção, é projetado para uma situação de saída de pista causada por desatenção do condutor, de maneira que o veículo se mantenha na sua trajetória original. A assistência dada por esse sistema ao condutor consiste na correção automática em caso de desvio do curso original, o que pode vir acompanhado de um outro sistema, o LDWS (*Lane Departure Warning System*), que pode se utilizar de mecanismos audiovisuais e/ou vibratórios para alertar o condutor da saída de sua rota.

O sistema AEBS opera a partir da detecção de obstáculos no trajeto e a utiliza como entrada para um controlador principal de decisão ([TAEYOUNG et al., 2011](#)). Com isso, o sistema definirá a ação de controle mais adequada, bem como as condições adequadas para a frenagem, em tempo oportuno, para evitar a colisão. De acordo com [Grover et al. \(2008\)](#), esse sistema tem dois principais níveis de atuação: o primeiro deles atua de forma preventiva, alertando o motorista de que existe um risco de colisão iminente, a partir da atuação de sensores. A outra forma de atuação desse sistema é intervencionista, isto é, automaticamente a velocidade será reduzida, ativando a frenagem do veículo. De forma geral, o AEBS aplica algoritmos de controle para frenagem autônoma, para reduzir a velocidade ou evitar um maior impacto, caso o mesmo venha a ocorrer.



Figura 1 – LKAS e AEBS

Fonte: *Google Imagens*

Neste contexto, como parte das atividades do projeto, os sistemas LKAS e AEBS serão desenvolvidos em um veículo de pequena escala, um Jeep Wrangler (Figura 2), para que, posteriormente, seja possível aplicar essas funcionalidades em um veículo real. Esses

sistemas serão consolidados no veículo de pequena escala por meio de fusão sensorial, isto é, um processo que lida com a combinação de dados e informações de diferentes fontes para obter informações mais precisas. Dessa forma, no SegurAuto, haverá a integração de uma câmera e de um radar para que o veículo seja capaz de atuar de maneira autônoma.



Figura 2 – Veículo de pequena escala

Para isso, um sistema de direção e um sistema de aceleração e frenagem serão desenvolvidos por meio da utilização de duas centrais de controle, as ECUs (*Electronic Control Unit*), uma para cada sistema de controle, além da elaboração de uma rede CAN (*Controller Area Network*), para que haja uma comunicação efetiva entre as duas ECUs.

1.1 Objetivos

O objetivo geral deste trabalho foi desenvolver uma arquitetura eletroeletrônica para o veículo de pequena escala, embasando-se em conceitos de controle de velocidade de cruzeiro, do sistema de direção e de frenagem.

Os objetivos específicos do trabalho são:

- Desenvolver um algoritmo de controle para o sistema de direção;
- Desenvolver um algoritmo de controle para o sistema de aceleração e frenagem;
- Desenvolver ECUs para viabilizar a aplicação dos sistemas de controle de cada subsistema de forma embarcada no veículo de pequena escala;
- Integrar módulos de comunicação CAN e desenvolver o *software* para essa comunicação;

- Desenvolver o barramento CAN para a comunicação entre as ECUs de direção, aceleração e frenagem e comunicação.

1.2 Organização do trabalho

O trabalho será dividido em capítulos, cada um contendo um tema principal, com objetivos específicos, que serão lembrados no começo de cada seção. Espera-se a seguinte disposição de capítulos para isso:

- Capítulo 2: Neste capítulo será apresentada a fundamentação teórica acerca dos temas deste trabalho. Os temas estudados aqui serão: sistema de aceleração e frenagem, sistemas de direção, sistemas de controle e teoria de controladores, modelagem matemática de motores de corrente contínua e introdução à rede CAN.
- Capítulo 3: Neste capítulo, será apresentada a metodologia utilizada, evidenciando os métodos e as ferramentas.
- Capítulo 4: Neste capítulo, serão apresentados e discutidos os resultados obtidos.
- Capítulo 5: Neste capítulo, será apresentada a conclusão sobre este trabalho.

2 Fundamentação Teórica

Este capítulo apresentará os principais estudos e as principais teorias consolidadas sobre os temas relacionados aos sistemas de direção, ao sistema de aceleração e frenagem, aos sistemas de controle e ao desenvolvimento de rede CAN. A utilização de veículos em pequena escala para o desenvolvimento de novos projetos funcionais proporciona benefícios relevantes, porque a utilização de modelos reduzidos permite a realização de diferentes testes, em sistemas embarcados por exemplo, e validações.

Existem diversos trabalhos no âmbito de veículos autônomos e desenvolvimento de controladores PID para veículos de pequena escala e protótipos. E no tópico seguinte são apresentados alguns deles.

2.1 Estado da Arte

2.1.1 Modelo de pequena escala autônomo

O estudo de [Robila et al. \(2021\)](#), cujo objetivo foi desenvolver um modelo autônomo, baseou-se na modelagem e na implementação de um controlador PID (Proporcional Integral Derivativo) para o controle da direção de um veículo de escala reduzida, bem como de um sistema de detecção de obstáculos. O trabalho teve o seu desenvolvimento a partir do uso de um protótipo de pequena escala e de componentes de *hardware*, como o motor Titan12-Turn550, o controlador eletrônico de velocidade XL5, Arduino MEGA 2560 e um sensor ultrassônico para identificar distâncias. Para a incorporação do *software*, foi utilizada uma placa de processamento gráfico NVIDIA Jetson Xavier. Como resultado, os autores conseguiram estabelecer duas capacidades principais para o protótipo de pequena escala: a detecção de obstáculos, analisando se o veículo diminuiu sua velocidade quando necessário, e o rastreamento de pista, o que permitiu que o protótipo mudasse a sua direção de acordo com uma trajetória predefinida.

2.1.2 Modelo de pequena escala com comando via *Bluetooth*

De forma semelhante, [Ciuciu et al. \(2019\)](#) focaram em desenvolver um veículo de pequena escala autônomo, capaz de navegar sem a intervenção humana, de forma que o modelo fosse capaz de seguir por rotas predefinidas, com os dados coletados sendo exibidos via aplicativo utilizando *Bluetooth*. Para tanto, os autores utilizaram um protótipo, um servo motor, um motor de CC sem escovas, um controlador eletrônico de velocidade, sensores ultrassônicos e um Arduino Nano. Como resultado, os autores estabeleceram no veículo a capacidade de seguir rotas pré-estabelecidas, bem como de identificar situações

de risco para o sistema, como a perda de contato com o solo e situações de subida de rampa, o que, naturalmente, causaria esforços maiores no motor.

2.1.3 Modelo de pequena escala autônomo com uso de controle LQR

Ainda no âmbito de veículos autônomos, em [Simone e Guida \(2018\)](#) foi feita a identificação do sistema e a aplicação do controle em um pequeno veículo, utilizando componentes de baixo custo. O modelo possui, além do chassi, um Arduino Mega 2560, dois motores de corrente contínua com *encoders* incrementais para medir a velocidade, um módulo driver, sensores ultrassônicos e acelerômetros para a detecção de obstáculos e para a obtenção do *feedback* do sinal de controle. Utilizou-se também o *N4SID (Numerical algorithms for Subspace State Space System Identification)* para a identificação do modelo dinâmico do veículo e em malha aberta e fechada, o algoritmo de controle LQR (*Linear Quadratic Regulator*) foi implementado. Como conclusão, a metodologia aplicada funcionou e demonstrou a eficácia na identificação e no controle.

2.1.4 Modelo de pequena escala com uso de um *datalogger*

Tratando-se do barramento CAN, em [Sreevatsan et al. \(2021\)](#) foi realizada a construção de um *datalogger* para o modelo em miniatura de um carro com o barramento. O carro foi equipado com sensores como acelerômetro e fins de curso, dois microcontroladores e um dispositivo de armazenamento. Um dos microcontroladores adquire diretamente dados advindos dos sensores, valores como a aceleração, frenagem, indicação de possível colisão, coordenadas GPS e a velocidade do veículo, e o outro microcontrolador recebe esses dados via barramento CAN e os armazena no formato *.txt* para poderem ser utilizados posteriormente.

2.1.5 Veículo de pequena escala autônomo capaz de seguir uma trajetória poligonal sem repetição

Sobre a definição de trajetórias predefinidas para um modelo autônomo, surge o estudo de [Petterman \(2015\)](#). Em seu trabalho, ele propõe um projeto de algoritmos de geração de seguimento de trajetórias, de modo que o modelo de pequena escala percorra integralmente áreas predefinidas. A velocidade do protótipo é constante, sendo controlada a partir do desenvolvimento de controladores P (Proporcional) e PD (Proporcional Derivativo). Esses controladores também são aplicados na manutenção do veículo na trajetória definida. A instrumentação utilizada nesse trabalho usou um Arduino Uno, uma ponte H L298N, um encoder de velocidade e um módulo *Bluetooth* para o envio de comandos de forma remota. A trajetória era definida a partir de coordenadas, que geravam uma forma poligonal à qual o protótipo se encontrava restrito. Com os testes executados, foi consta-

tado um erro médio de 3.9 milímetros, isto é, o robô passava 3.9 mm da área delimitada e voltava à trajetória correta.

2.1.6 Veículo inteligente de pequena escala semi-autônomo

No projeto de [Rosier, Riccoboni e Rothhammer-Ruiz \(2018\)](#), foi instrumentado um veículo de pequena escala, de forma a criar uma plataforma semi-autônoma, possibilitando que ele se comporte de forma previsível em um ambiente experimental e que colete e processe dados de tal forma que possam servir de ferramenta de instrução para o veículo. O objetivo geral do projeto foi desenvolver, no veículo, uma plataforma de estudos, a partir da documentação dos procedimentos feitos. Estudos na área da mecânica, da mecatrônica, da eletrônica e de software foram feitos para a implementação de sistemas de controle para o LKAS, para o ACC (Adaptative Cruise Control) e para o ESC (Electronic Stability Control), por meio da utilização de sensores, de LiDAR (Light Detection and Ranging) e de GPS (Global Positioning System). Para o desenvolvimento das funções de controle, foi utilizado o *software* Matlab, por meio do qual os resultados foram obtidos e discutidos.

Nesse sentido, é de grande importância para este trabalho o desenvolvimento de um sistema de direção por meio de um controlador, uma vez que o veículo de pequena escala, objeto de estudo, deverá ser capaz de seguir trajetórias, sejam elas predefinidas ou não. Assim, o tópico abaixo apresenta os principais aspectos de um sistema de direção, bem como as equações correlatas a ele.

2.2 Sistemas de direção

Segundo [Gillespie \(1992\)](#), a principal função de um sistema de direção é permitir o esterçamento das rodas frontais a partir de um comando dado pelo condutor. Os modelos variam entre os mais diversos veículos, especialmente quanto à complexidade da montagem e da adequação no sistema. Os modelos mais comuns para veículos de passeio são: modelo pinhão-cremalheira e o modelo da caixa de redução. Contudo, antes de entrar na análise de cada modelo de direção, é importante entender o conceito de geometria de Ackerman (Figura 3).

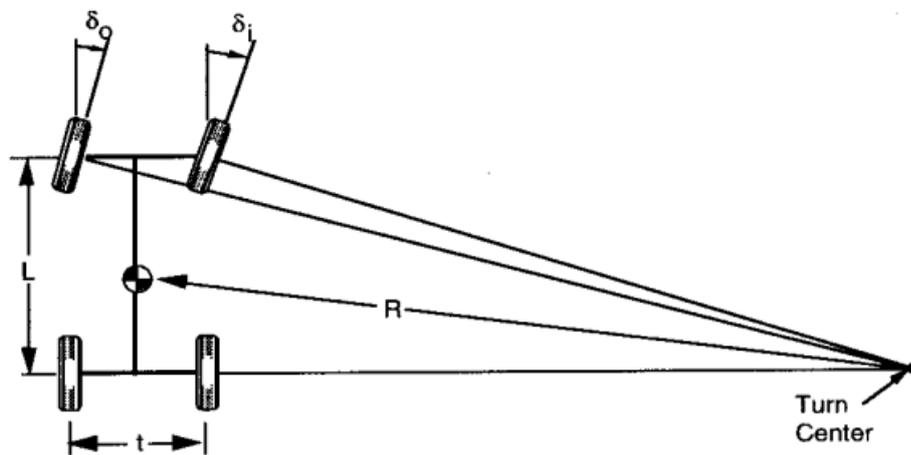


Figura 3 – Geometria de Ackerman

Fonte: (GILLESPIE, 1992, p. 278)

Esse termo se refere ao comportamento das rodas em um processo de esterçamento. Comumente, uma roda gira mais do que a outra devido a uma geometria trapezoidal formada entre a barra de direção e os braços de direção, o que faz com que o ângulo da roda externa seja menor do que o ângulo da roda interna. Na figura 3, pode-se notar que o cálculo desses ângulos pode ser feito a partir das informações da bitola (t), do entre eixos (L) e do raio de giro (R). Os ângulos interno e externo são obtidos da seguinte forma:

$$\delta_0 = \arctan\left(\frac{L}{R + \frac{t}{2}}\right) \quad (2.1)$$

$$\delta_i = \arctan\left(\frac{L}{R - \frac{t}{2}}\right) \quad (2.2)$$

Todavia, como os ângulos de esterçamento em curvas são geralmente pequenos, as equações 2.1 e 2.2 podem ser simplificadas para:

$$\delta_0 = \left(\frac{L}{R + \frac{t}{2}}\right) \quad (2.3)$$

$$\delta_i = \left(\frac{L}{R - \frac{t}{2}}\right) \quad (2.4)$$

De maneira geral, os veículos dificilmente apresentam o Ackerman perfeito, ou seja, não seguirão pontualmente as equações supracitadas. A composição e o posicionamento do sistema de barras e estruturas do sistema de direção influenciam definitivamente no comportamento dos ângulos de esterçamento. A influência mais impactante da geometria de Ackerman se dá em baixas velocidades, quando o torque necessário para o esterço é

maior. Em contrapartida, os veículos, de forma geral, não apresentam o Ackerman zero, isto é, uma geometria com braços de direção paralelos, o que traria como consequência ângulos iguais para a roda externa e para a roda interna. Na prática, o comportamento de esterçamento das rodas pode ocorrer de forma diferente da prevista na teoria, o que gera uma diferença entre o ângulo da roda de fato esterçada e o ângulo calculado pela geometria ideal de Ackerman, essa diferença é conhecida como erro de Ackerman (MARTINS, 2010).



Figura 4 – Geometria das barras do veículo de pequena escala

O veículo de pequena escala, objeto de estudo deste trabalho, possui as geometrias das barras do sistema de direção aproximadamente semelhantes à situação de Ackerman zero, isto é, apresentam uma composição em que os braços de direção estão praticamente paralelos, como se nota na figura 4. Essa disposição traz como consequência ângulos de esterçamento praticamente iguais tanto para a direita quanto para a esquerda, o que implica em maiores dificuldades em executar curvas com raios menores, isto é, curvas mais fechadas, fazendo com que o veículo tenha que percorrer uma maior distância para realizar a curva de forma completa.

Os principais modelos de direção são o modelo pinhão-cremalheira e o modelo da caixa de redução. No veículo de pequena escala, esse sistema é formado por uma coluna que conecta o volante diretamente na barra de direção, a qual está acoplada às rodas. O sistema é simples, puramente mecânico e satisfatório para essa aplicação, mas apresenta muitas folgas e fatores causadores de atrito entre os elementos. O modelo tradicional que mais se assemelha ao modelo do veículo é o da caixa de redução, que está apresentado abaixo.

2.2.1 Modelo da Caixa de Redução

Nesse modelo, a coluna de direção passa por uma caixa que contém uma associação de engrenagens que rotaciona o braço de direção, por meio de uma associação de barras de ligação (GILLESPIE, 1992). A caixa de redução, vista na figura 5, é o primeiro mecanismo de redução efetiva a partir da entrada de rotação colocada no sistema por meio do giro do volante. Entre o ângulo de giro da coluna e o ângulo de giro das rodas, existe uma relação que, mais comumente, faz com que a coluna de direção gire um ângulo maior do que o das rodas. Em sistemas de direção mais complexos e modernos, essa assistência ao condutor é feita por meio de motores elétricos, que fazem com que o torque dado pelo condutor seja menor, garantindo menos esforço e mais conforto.

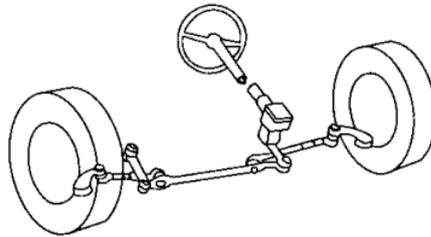


Figura 5 – Modelo caixa de redução

Fonte: (GILLESPIE, 1992, p. 276)

A garantia de maior conforto e menor esforço se dá pela relação entre o ângulo do volante e o ângulo da roda. Dessa forma, a relação emergente da conexão entre o ângulo da coluna de direção e o ângulo de giro das rodas, pode ser definida como a razão entre esses ângulos, considerando que θ_c é o ângulo de esterçamento da coluna de direção e que θ_r é o ângulo de esterçamento das rodas:

$$R_{cr} = \frac{\theta_c}{\theta_r} \quad (2.5)$$

De forma geral, os veículos de passeio possuem essa relação R_{cr} na ordem de 15:1, matematicamente, essa relação revela que a cada 15° de esterçamento da coluna de direção, é girado 1° na roda. Para caminhões leves e caminhonetes, essa relação é de 36:1. A grande vantagem de se ter uma alta relação de ângulo de volante é a de se reduzir o esforço do motorista, tornando o sistema de direção mais confortável para o condutor.

Outro sistema de fundamental importância para a análise da dinâmica de um veículo é o sistema de aceleração e frenagem, e para o veículo de pequena escala em questão, são importantes o modelo da dinâmica longitudinal e o modelo do trem de força,

os quais têm a sua modelagem matemática e principais características apresentadas no tópico seguinte.

2.3 Sistema de Aceleração e Frenagem

Um sistema de aceleração e frenagem funcional, precisa ser capaz de acelerar e desacelerar um determinado veículo, fazendo com que a velocidade do mesmo varie ao longo do tempo, de acordo com as referências desejadas. Diferentes veículos possuem capacidades distintas de frenagem e de aceleração também, fatores regidos pelos componentes desse sistema e isso inclui a lógica de controle quando o sistema é baseado em uma arquitetura eletroeletrônica. Para melhor compreensão das principais variáveis envolvidas na dinâmica do veículo quando se trata do sistema de aceleração e frenagem, o desenvolvimento de um modelo é de grande utilidade, no tópico seguinte, é construído o modelo da dinâmica longitudinal.

2.3.1 Modelo da Dinâmica Longitudinal

A dinâmica lateral está relacionada ao sistema de direção, já o sistema de aceleração e frenagem influencia na dinâmica longitudinal de um veículo. Pode-se construir um modelo matemático da dinâmica longitudinal de um veículo, algo que auxilia na compreensão de quais variáveis influenciam no desempenho do automóvel durante seu funcionamento.

Para a construção do modelo matemático que viabiliza a análise do comportamento da aceleração e da frenagem, o primeiro passo é estimar o carregamento sobre os eixos, dianteiro e traseiro, determinando a força trativa obtida em cada eixo (GILLESPIE, 1992).

Na Figura 6 pode-se observar as forças significativas sobre um veículo e, com base nesse figura, o modelo pode ser obtido aplicando a Segunda Lei de Newton.

Para esclarecer de melhor forma as forças que estão apresentadas na Figura 6, as mais relevantes para o caso deste trabalho estão descritas são: W é a força peso, decomposta em duas devido ao ângulo de inclinação θ ; F_{xf} e F_{xr} são as forças trativas no eixo dianteiro e traseiro respectivamente; R_{xf} e R_{xr} são as forças de resistência ao rolamento no eixo dianteiro e traseiro respectivamente; W_f e W_r são as forças normais no eixo dianteiro e traseiro respectivamente; D_A é a força de arrasto aerodinâmico e θ ângulo de inclinação da rampa.

Utilizando a Segunda Lei de Newton e acrescentando algumas definições, obtêm-se as seguintes relações presentes nas equações 2.6, 2.7, 2.8, 2.9 e 2.10 nas quais m é a massa do veículo; a é a aceleração do veículo; f é o coeficiente de resistência ao rolamento; ρ é a densidade do ar; C_D é o coeficiente de arrasto; A é a área de contato do ar com o veículo;

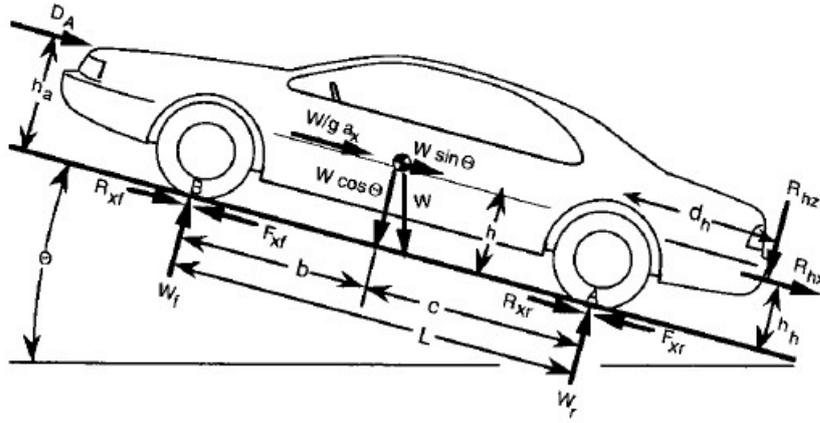


Figura 6 – Forças significativas sobre um veículo

Fonte: (GILLESPIE, 1992, p. 11)

u é a velocidade do veículo e u_w é a velocidade do vento.

$$m \cdot a = F_{xf} + F_{xr} - R_{xf} - R_{xr} - D_A - W \cdot \sin \theta \quad (2.6)$$

$$0 = W_f + W_r - W \cdot \cos \theta \quad (2.7)$$

Considerando $F_x = F_{xf} + F_{xr}$, $R_x = R_{xf} + R_{xr}$ e $R_x = f \cdot W_f + f \cdot W_r$

$$R_x = f \cdot W \cdot \cos \theta \quad (2.8)$$

$$D_A = \frac{1}{2} \cdot \rho \cdot C_D \cdot A \cdot (u + u_w)^2 \quad (2.9)$$

$$a = \frac{F_x}{m} - g \cdot f \cdot \cos \theta - g \cdot \sin \theta - \frac{1}{2} \cdot \frac{\rho}{m} \cdot C_D \cdot A \cdot (u + u_w)^2 \quad (2.10)$$

Portanto, a Equação 2.10 mostra que a força trativa F_x é a responsável pela movimentação do veículo e as demais forças são dissipativas. A força trativa possui relação direta com os pneus utilizados e com o trem de força.

O desempenho da aceleração pode ser limitado por basicamente dois fatores: a potência e a tração, dessa forma, a força F_x citada anteriormente também depende deles. Essa limitação é analisada entendendo-se as características da motorização e sua interação através do trem de força. O trem de força, também conhecido como Power Train, consiste em um conjunto de elementos que juntos transmitem a potência do motor para o solo através das rodas. Esses elementos podem variar de acordo com o projeto e o tipo de veículo, mas no geral são: motor, embreagem ou conversor de torque, transmissão, eixo de

transmissão, diferencial e semi-eixo (GILLESPIE, 1992). A fundamentação matemática e a montagem característica desse modelo estão representadas no tópico a seguir.

2.3.2 Modelo do Trem de Força

Para o caso deste trabalho, o sistema de aceleração e frenagem envolve dois motores de corrente contínua, esses motores podem ser modelados matematicamente a fim de se obter a função de transferência associada, que relaciona a entrada com a saída, além de que esse procedimento contribui para a compreensão de quais fatores implicam na resposta do sistema.

A modelagem citada está desenvolvida logo a seguir. Na Figura 7, o esquemático de um motor de corrente contínua acoplado a uma carga é apresentado.

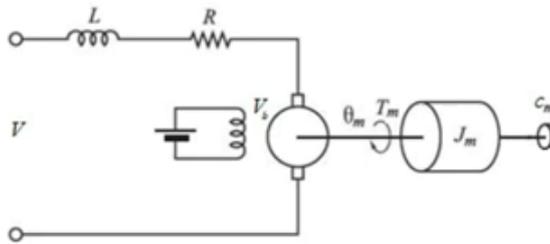


Figura 7 – Esquemático de um motor de corrente contínua acoplado a uma carga

Fonte: (PINTO, 2022)

Portanto, combinando as Equações 2.11 e 2.12, é possível obter a função de transferência para o motor de corrente contínua apresentado, considerando que V_b é a tensão na bobina do motor CC; k_b é a constante de proporcionalidade; $\dot{\theta}_m$ é a velocidade angular de saída do motor CC; T_m é o torque de saída do motor; k_t é a constante de proporcionalidade e $i(t)$ é a corrente elétrica.

$$V_b(t) = k_b \cdot \dot{\theta}_m(t) \quad (2.11)$$

$$T_m(t) = k_t \cdot i(t) \quad (2.12)$$

A Equação 2.11 mostra que a velocidade angular de saída do motor é diretamente proporcional à tensão na bobina e a Equação 2.12 demonstra a relação direta entre o torque de saída do motor e a corrente de alimentação.

$$V(t) = L \frac{di(t)}{dt} + Ri(t) + V_b(t) \quad (2.13)$$

Utilizando a Lei das Malhas (Segunda Lei de Kirchhoff), é possível chegar na Equação 2.13, na qual $V(t)$ é a tensão de alimentação; L é a indutância e R é a resistência.

Fazendo o somatório de torque, chega-se a Equação 2.14, em que J_m é o momento de inércia do eixo; $\ddot{\theta}_m$ é a aceleração angular de saída do motor e c_m é o coeficiente de atrito viscoso.

$$T_m = J_m \ddot{\theta}_m + c_m \dot{\theta}_m \quad (2.14)$$

Considerando a indutância L muito menor que a resistência R , aplicando a Transformada de Laplace nas quatro últimas equações citadas e rearranjando-as, obteve-se a Equação 2.15, função de transferência do motor CC.

$$\frac{\Theta_m(s)}{V(s)} = \frac{k_t}{J_m R s^2 + (c_m R + k_t k_b) s} \quad (2.15)$$

Dessa forma, a modelagem matemática do motor é dada pela Equação 2.15. Esse modelo será utilizado em ambiente virtual para desenvolvimento e teste de ganhos dos controladores, os quais posteriormente serão utilizados para controle do sistema real.

Atualmente, com o aumento das funções eletroeletrônicas surge a necessidade de testar de forma mais rápida os sistemas presentes em um veículo, os subsistemas estão cada vez mais sendo modelados e validados em ambiente virtual por meio de simulações. Assim, estão apresentados abaixo os principais conceitos, modelos e fundamentações matemáticas a respeito de controladores, modelagem de sistemas, redes de comunicação CAN e instrumentação, que para a construção de uma arquitetura eletroeletrônica de um veículo são indispensáveis.

2.4 Sistemas de Controle e Controladores

2.4.1 Sistemas de Controle

Segundo Bars, Banyasz e Hetthessi (2019), um sistema de controle pode ser definido como um conjunto de equipamentos e dispositivos que gerenciam o comportamento de máquinas ou outros sistemas físicos. Controlar significa, ainda, medir o valor da variável analisada e aplicar, se necessário, um sinal de controle ao sistema para corrigir ou limitar os desvios no valor medido a partir de um valor desejado (OGATA, 2010).

Dentre os diversos tipos de sistemas de controle, tem-se o controle em malha fechada, isto é, o sinal de saída retorna para o começo da malha de controle para ser analisado, pelo controlador, se houve diferença entre a entrada requerida e a saída lida, por meio de comparação. Neste tipo de sistema, que pode ser visto na figura 8, o valor de saída retorna para o começo da malha, o que gera uma diferença entre a referência e o que de fato foi feito, diferença esta chamada de erro. O termo controle de malha fechada

sempre implica a utilização do controle com realimentação para reduzir o erro do sistema (OGATA, 2010).

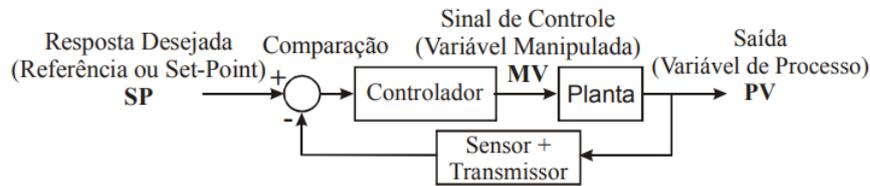


Figura 8 – Exemplo de sistema de controle de malha fechada

Fonte: (ARAUJO, 2007)

Toda essa base de projeto deve ser amparada por um desenvolvimento matemático que permita não somente o entendimento dos modelos, mas os pontos nos quais eles podem ser previstos e, assim, melhorados.

2.4.2 Modelagem matemática

Para o desenvolvimento de controle, geralmente é modelada a planta, que é o modelo matemático representativo do sistema físico, a fim de testar o controle antes da implementação em um sistema real, sempre que possível, equilibrando a relação simplicidade/precisão (OGATA, 2010). A modelagem parte da análise do sistema dinâmico envolvido no estudo, o que culmina em equações diferenciais. Uma vez com as equações estabelecidas para o sistema, pode-se aplicar a transformada de Laplace no sinal de saída e no de entrada e, assim, correlacioná-los, o que permite encontrar uma solução que descreva a operação do sistema. A sequência ideal, segundo Dorf e Bishop (2018), para estabelecer a modelagem de um sistema dinâmico segue o seguinte conjunto de etapas:

1. Definir o sistema e seus componentes;
2. Formular o modelo matemático e as hipóteses necessárias para torná-lo aplicável;
3. Obter as equações diferenciais;
4. Resolver as equações para as variáveis de saída desejadas;
5. Examinar as soluções e as hipóteses

2.4.2.1 Funções de Transferência e Espaço de Estados

Em sistemas lineares, a transformada de Laplace permite a visualização do sistema na forma de uma função de transferência. Matematicamente, a sua relação é dada a partir de uma função dependente do tempo, definido em um intervalo de 0 a ∞ :

$$F(s) = \int_0^{\infty} f(t)e^{-st} dt \quad (2.16)$$

Para facilitar as análises, existe a tabela de transformadas de Laplace para diferentes funções $f(t)$, as quais podem ser observadas no anexo A. De acordo com Dorf e Bishop (2018), uma função de transferência pode ser representada pela relação entre a transformada de Laplace da saída sobre a da entrada, com todas as condições iniciais iguais a zero. Cabe ressaltar também que a aplicação desse método para sistemas lineares só se adequa em sistemas invariantes no tempo, isto é, aqueles nos quais os parâmetros não se alteram ao longo do funcionamento do sistema.

Outra forma bastante comum de representação da planta de um sistema, é a forma de espaço de estados. "O estado de um sistema é um conjunto de variáveis cujos valores, em conjunto com os sinais de entrada e as equações descrevendo a dinâmica, irão fornecer o estado e as saídas futuras do sistema" (DORF; BISHOP, 2018, pp. 121). De acordo com Ogata (2010) e com Araujo (2007), as vantagens de se utilizar essa metodologia para a análise de sistemas de controle residem na expansão de possibilidades, pois esse método permite trabalhar com sistemas de múltiplas entradas e saídas, lineares ou não, invariantes no tempo ou não e com condições iniciais nulas ou não. São chamadas de variáveis de estado o número mínimo de variáveis que são capazes de descrever o estado de um sistema dinâmico.

Outra vantagem da multiplicidade de variáveis é que ela permite uma filtragem do que se quer analisar dentro de um sistema com inúmeros parâmetros. Quando essas variáveis são escritas como elementos de um vetor, este vetor é conhecido como vetor de estado que, segundo Ogata (2010), é aquele capaz de descrever de maneira unívoca o estado de um sistema $x(t)$ para qualquer instante $t \geq t_0$. Matematicamente, o espaço de estados é representado pelas seguintes expressões:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \quad (2.17)$$

$$y(t) = C(t)x(t) + D(t)u(t) \quad (2.18)$$

A equação 2.17 representa a equação de estados e a equação 2.18 representa a equação de saídas do sistema, onde $A(t)$ é a matriz de estado, $B(t)$ é a matriz de entrada, $C(t)$ é a matriz de saída, $D(t)$ é a matriz de transmissão direta e $u(t)$ são as entradas do sistema.

Com as principais equações assimiladas, é importante dar uma visão mais crítica a respeito dos sistemas de controle, analisando sua estabilidade, desempenho e robustez.

Esses tópicos são fundamentais para o desenvolvimento de um sistema confiável e pouco suscetível à falhas.

2.4.3 Estabilidade e Desempenho

Prever o comportamento de um sistema de controle é fundamental para determinar o que pode torná-lo instável e quais as suas faixas de estabilidade, ao mesmo tempo em que é possível determinar quais os melhores parâmetros para o seu bom desempenho. Diz-se que um sistema é estável se a sua saída sempre retorna ao estado de equilíbrio quando o sistema é submetido a uma condição inicial ou a uma perturbação externa. Quando não há essa convergência do sinal de saída, tem-se um sistema instável. Em consonância com [Franklin, Powell e Naeini \(2009\)](#), um sistema de controle é estável se todas as raízes do polinômio do denominador da função de transferência do sistema, isto é, os polos têm sua parte real negativa. Considerando σ a parte real e $j\omega$ a parte imaginária do polo de um sistema, então ele pode ser representado pela equação 2.19.

$$s = (\sigma \pm j\omega) \quad (2.19)$$

Se $\sigma > 0$, então o sistema é instável e se $\sigma < 0$, o sistema é estável. Graficamente, pode-se observar esse comportamento no plano Argand-Gauss (figura 9), que possui como eixo das abscissas os valores reais e como eixo das ordenadas os valores imaginários. Se a parte real dos polos se encontrar no semiplano esquerdo, o sistema está em região de estabilidade, do contrário estará em situação de instabilidade. Quando o polo do sistema estiver na fronteira das duas regiões, diz-se que ele está criticamente estável, isto é, na iminência de entrar em condição de instabilidade.

Uma técnica utilizada para a determinação da estabilidade de um sistema é a do Critério de Routh. Esse critério permite a determinação da existência de instabilidade no sistema sem a necessidade de se resolver a equação polinomial do denominador da função de transferência representativa do sistema. Esse método consiste na montagem de uma tabela com valores de coeficientes, que são calculados a partir de operações de determinantes. Se houver mudança de sinal na primeira coluna de coeficientes, então haverá ao menos um polo em região de instabilidade. Quanto mais mudanças de sinal houver nessa primeira coluna, mais polos se encontram no semiplano direito do plano Argand-Gauss.

Quando um sistema é submetido a um sinal de entrada qualquer, ele fornece uma resposta a essa entrada que, na teoria de controle, se divide em duas partes principais: a resposta em regime transitório e a resposta em regime permanente ou estacionária. Por resposta transitória, entende-se aquela que vai do estado inicial para o estado final. Já

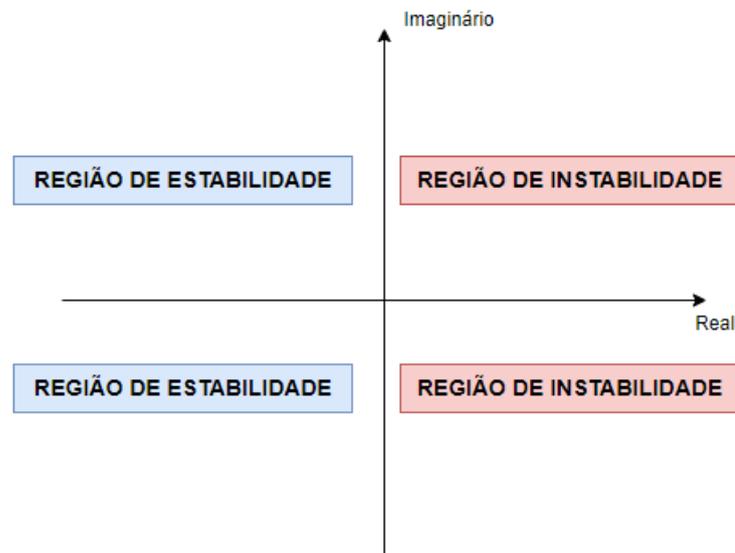


Figura 9 – Regiões de estabilidade e instabilidade no plano complexo

por resposta estacionária, entende-se como o comportamento do sinal de saída na medida em que o tempo tende ao infinito (OGATA, 2010).

O desempenho de um sistema pode ser definido como sendo o seu comportamento quando submetido a uma entrada qualquer. De forma geral, para se analisar o desempenho de um sistema, são colocadas nele entradas conhecidas e padronizadas, com comportamentos esperados, como é o caso de um sinal degrau. Quando o sistema é submetido a um sinal degrau unitário, a representação mais comum traz alguns outros parâmetros de relevância para o desempenho, como o sobressinal, o tempo de subida e o tempo de assentamento.

Com os principais assuntos a respeito de sistemas de controle estabelecidos, surge a necessidade de entender os meios pelos quais eles serão colocados em um modelo qualquer. Esses meios são os controladores, cujos principais tipos e características estão apresentados no tópico abaixo.

2.4.4 Tipos de Controladores

Os controladores são criados por meio de algoritmos e têm por objetivo aplicar, se necessário, ações corretivas em um sistema físico, para que ele esteja em um valor ou com um comportamento esperado. Os controladores são desenvolvidos a partir de ganhos (K), que serão determinantes para o comportamento do sistema no regime transitório e no regime permanente. Geralmente, ao se aumentar um tipo de ganho, melhora-se um regime e piora-se o outro, sendo, portanto, um procedimento também empírico e de testes.

Dois tipos de controladores, segundo [Araujo \(2007\)](#), são:

- Controlador Proporcional (P)
- Controlador Proporcional-Integral (PI)
- Controlador Proporcional-Derivativo (PD)
- Controlador Proporcional Integral Derivativo (PID)
- Controlador PID discreto

O controlador proporcional (P) relaciona a saída com a entrada a partir de um ganho K . Ao aumentar o valor de K , o sistema tende a tornar-se mais oscilatório e instável, porém reduz o erro em regime permanente. Dessa forma, é visto como um controlador extremamente limitado, já que dificilmente será encontrado um bom meio termo entre desempenho e estabilidade, especialmente, em aplicações mais complexas. Seu equacionamento está representado pela equação [2.20](#):

$$U(s) = K_p E(s) \quad (2.20)$$

O controlador do tipo proporcional-integral (PI) possui dois ganhos que irão influenciar no erro: k_p e k_i . Guardando as mesmas características do tipo P, esse controlador atua com uma ação integral, isto é, com um acumulador de erros a partir de uma taxa de variação do sinal de entrada em relação à saída. A adição do integrador no sistema traz como consequência a alocação de mais um polo no plano complexo, o que, embora traga uma melhora no regime permanente, faz o sistema tender para a instabilidade no regime transitório, aumentando seu tempo de acomodação ([ARAUJO, 2007](#)). O equacionamento relacionado a este tipo de controlador elucida essas características, e pode ser encontrado na equação [2.21](#):

$$U(s) = E(s) \left(K_p + \frac{K_i}{s} \right) \quad (2.21)$$

Em suma, é um tipo de controlador mais abrangente e que permite solucionar problemas relacionados ao regime permanente, sendo aplicado quando já se tem uma boa resposta no regime transiente.

O controlador do tipo proporcional-derivativo (PD) também possui dois ganhos que irão atuar nas respostas do sistema: k_p e k_d . A adição do termo derivativo traz como consequência o surgimento de um zero no sistema, o que tende a estabilizar o sistema no regime transitório e a diminuir o seu tempo de acomodação. Contudo, a sua implementação causa um acréscimo no tempo de subida do sistema, tornando-o mais lento, além de em

nada contribuir para o regime permanente. Seu equacionamento está representado na equação 2.22:

$$U(s) = E(s)(K_p + K_d s) \quad (2.22)$$

De acordo com Araujo (2007), esse controlador introduz um efeito de antecipação no sistema, fazendo com que o mesmo reaja não somente à magnitude do sinal de erro, como também à sua tendência para o instante futuro, iniciando, assim, uma ação corretiva mais cedo. Todavia, a inserção do termo derivativo pode amplificar os sinais de ruído no sistema, causando um efeito de saturação nos seus atuadores.

O controlador do tipo proporcional integral derivativo (PID) une as três ações de controle citadas acima, trabalhando, portanto, com três constantes: K_p , K_i e K_d . É amplamente utilizado na indústria e nas aplicações de sistemas de controle, porque atua tanto no regime transitório como no permanente. Dessa forma, se mostra uma ação de controle efetiva e satisfatória para diferentes situações. Esse tipo de controlador, segundo Ogata (2010), é muito utilizado quando não se conhece ou não se pode obter o modelo matemático da planta, o que não permite o processo analítico padrão. O equacionamento para este controlador é:

$$U(s) = E(s)\left(K_p + \frac{K_i}{s} + K_d s\right) \quad (2.23)$$

A influência dos parâmetros k_p , k_i e k_d na resposta do sistema é, de forma direta e simplificada, o que se observa na figura 10.

Parâmetros	tr	Mp	ts	ess
Kp	Diminui	Aumenta	Pouca influência	Diminui
Ki	Diminui	Aumenta	Diminui	Diminui ou elimina
Kd	Pouca influência	Diminui	Diminui	Pouca influência

Figura 10 – Influência dos ganhos na resposta do sistema

Sendo tr o tempo de subida, Mp o sobressinal, ts o tempo de assentamento e ess o erro em regime permanente.

O controlador PID discreto é uma versão alternativa do controlador PID, com os mesmos fundamentos, mas comumente aplicado a sistemas de controle digital, operando

em um tempo discreto ao invés de contínuo. Para a aplicação deste tipo de controlador, os valores são amostrados em intervalos regulares de tempo, possibilitando que seja calculada uma saída do controlador a cada intervalo de amostragem. Semelhante ao controlador PID em tempo analógico, o parâmetro K_p é baseado no valor do erro gerado entre o valor medido e o valor desejado, o parâmetro K_i é baseado na integral do erro ao longo do tempo, e o parâmetro K_d é baseado na taxa de variação do erro. Dessa forma, pode-se escrever a equação geral desse controlador como apresentado na equação 2.24:

$$u[k] = u[k - 1] + K_p(e[k] - e[k - 1]) + K_i.T_s.e[k] + \frac{K_d}{T_s}.(e[k] - 2e[k - 1] + e[k - 2]) \quad (2.24)$$

Onde $u[k]$ é a saída do controlador, $e[k]$ é o erro no instante de tempo k , T_s é o intervalo de tempo entre as amostras e $u[k - 1]$, $e[k - 1]$ e $e[k - 2]$ são valores anteriores da saída do controlador e do erro, respectivamente.

Para complementar os estudos a respeito de sistemas de controle, o tópico abaixo apresentará uma estratégia de extrema relevância a aplicação em controles de sistemas reais: o agendamento de ganhos.

2.4.5 Agendamento de Ganhos

O agendamento de ganhos, do inglês *gain scheduling*, é um método amplamente utilizado em sistemas de características não lineares. De forma geral, o controlador é formado pela interpolação entre um conjunto de controladores lineares para um conjunto correspondente de linearizações da planta associada, agindo em diferentes pontos de operação. (BETT, 2005). Entretanto, a sua aplicação não é restrita a sistemas não lineares, podendo ser aplicado a qualquer planta que apresente um comportamento inconstante ao longo de sua operação.

A grande vantagem desse método é a capacidade de atender de forma estável e linear diferentes faixas de operação do sistema. Isso só se torna possível porque, comumente, há uma alternância no conjunto dos valores de parâmetros do controlador, a depender da situação a que a planta está sujeita. A figura 11 apresenta, de forma simplificada, a implementação desse método.

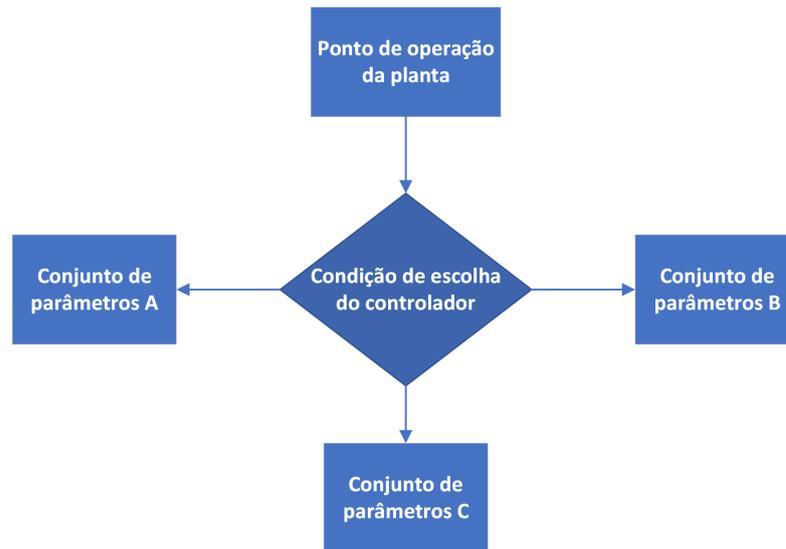


Figura 11 – Funcionamento do método de agendamento de ganhos

Na prática, o agendamento de ganhos pode ser aplicado em um sistema com a utilização de *softwares* como Matlab/Simulink ou por meio da implementação da sua lógica em um código de controle, utilizando comandos condicionais que segmentem a sua atuação na planta.

De posse das informações apresentadas, é fundamental entender que os diversos sistemas e métodos de controle são embarcados nas centrais dos veículos, que se comunicam continuamente por meio de um barramento, que utiliza o protocolo CAN. Esse protocolo é utilizado largamente no mercado automotivo desde a sua invenção, pois otimizou o processo de comunicação entre as centrais do veículo. Dessa forma, visando estabelecer no veículo de pequena escala o mesmo padrão da indústria, foi aplicada a rede CAN. No tópico abaixo, são apresentados os principais aspectos relacionados a ela.

2.5 Rede CAN

2.5.1 Conceito

Na indústria automotiva, o surgimento da eletrônica embarcada provocou mudanças significativas na arquitetura de um veículo. Os sistemas automotivos passaram a ser constituídos por basicamente três tipos de componentes: os componentes mecânicos, componentes eletroeletrônicos e componentes de software. Os componentes eletroeletrônicos são classificados como:

- Sensores automotivos

- Atuadores automotivos
- Unidade Eletrônica de Controle

O desenvolvimento da eletroeletrônica embarcada se deu devido à necessidade de melhorias, veículos com menor consumo de combustível, menor emissão de poluentes, que atendam à demanda de segurança veicular, que forneçam entretenimento, entre outras (SILVA, 2015).

No princípio, a arquitetura eletroeletrônica dos veículos era centralizada, posteriormente, se tornou distribuída devido aos benefícios encontrados. As arquiteturas citadas podem ser visualizadas na Figura 12.

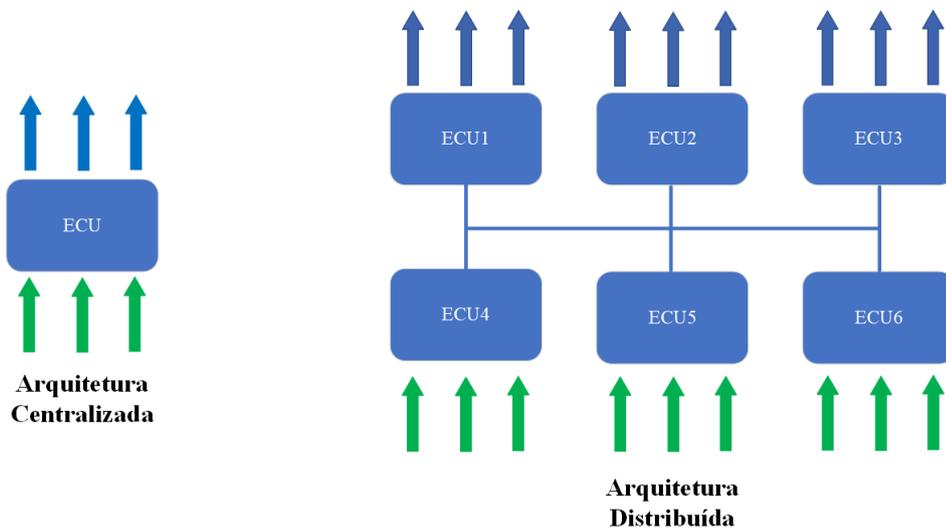


Figura 12 – Arquitetura Centralizada e Distribuída

Devido ao aumento e complexidade da eletroeletrônica, a arquitetura distribuída se tornou predominante, uma vez que apresenta benefícios tais como: o cabeamento utilizado é menor, a possibilidade de ocorrência de falhas também possui uma redução, a fabricação do veículo é mais simplificada e a ampliação modular torna-se possível. E é devido a este formato que o nome rede de comunicação é utilizado, isto porque as ECUs estão interconectadas transmitindo e recebendo dados.

Para a troca de informações entre as ECUs, protocolos de comunicação são utilizados, estes visam estabelecer padrões de desenvolvimento e implementação das redes automotivas. Existem vários protocolos de comunicação, FlexRay, LIN, Ethernet e CAN, eles são um meio utilizado para a transmissão e recepção de dados, cada um deles possuem características técnicas específicas que definem para qual aplicação podem ser utilizados.

A abreviação CAN vem do inglês *Controller Area Network*, criado em meados dos anos 1980, pela empresa alemã Robert Bosch, o barramento CAN foi desenvolvido inicialmente para a aplicação em ônibus e caminhões, cujo objetivo era, e ainda é, possibilitar a comunicação entre dispositivos de controle e atualmente é utilizado também em máquinas agrícolas, na robótica, em navios, entre outros. (SILVA, 2015).

O protocolo CAN consiste em um padrão de comunicação serial síncrono, em que a sincronia é feita entre os módulos no início da transmissão de cada mensagem enviada ao barramento, o que ocorre em um período de tempo conhecido e regular. O barramento CAN, ou do inglês, *CAN bus*, consiste basicamente em um par trançado de fios (trançado para reduzir interferência eletromagnética), um denominado de *CAN HIGH* e outro de *CAN LOW* e em cada uma das extremidades desse barramento, por padrão, é colocada uma resistência de 120 Ohms para evitar a reflexão do sinal. No protocolo CAN, todas as ECUs verificam o estado do barramento para que caso algum módulo esteja mandando uma mensagem de maior prioridade, o módulo com mensagem de menor prioridade pare de enviar e ele continue (GUIMARAES, 2007).

Com o objetivo de padronizar o compartilhamento de dados entre dispositivos em uma rede de comunicação geral, a ISO (*International Organization for Standardization*) criou o modelo de transmissão OSI (*Open Systems Interconnection*), o qual possui sete camadas, em que cada uma deverá executar tarefas ou estabelecer características e uma interface com as demais camadas. A Figura 13 mostra as sete camadas definidas pelo modelo OSI, por norma, apenas a camada física e a de enlace de dados estão definidas para a rede CAN, as demais camadas irão ser construídas de acordo com a aplicação e definição dos projetistas (SOUSA, 2002).

2.5.2 Funcionamento

As mensagens que transitam pelo barramento CAN são chamadas de *frame* e podem ser de dois tipos de acordo com a versão do protocolo CAN, a versão *Standard CAN* (CAN 2.0A), utiliza o *frame* com identificador constituído por 11 *bits*, já a *Extended CAN* (CAN 2.0B), contém identificador composto por 29 *bits* no total e ambos formatos possuem um campo de dados com tamanho de 64 *bits* (8 *bytes*) (SOUSA, 2002).

2.5.2.1 Camada Física

A camada física é o meio pelo qual as informações são transmitidas entre os módulos eletrônicos, no formato de *bits*, 0 (dominante) e 1 (recessivo). Então, o par de fios elétricos citados anteriormente, *CAN HIGH* e *CAN LOW*, fazem parte dessa camada e além disso, nessa camada são definidos parâmetros como: níveis de tensão, sincronização das ECUs, codificação e decodificação dos *bits* e os cabos e conectores utilizados. Os níveis lógicos 0 e 1 são formados a partir dos níveis de tensão nos condutores do barramento,

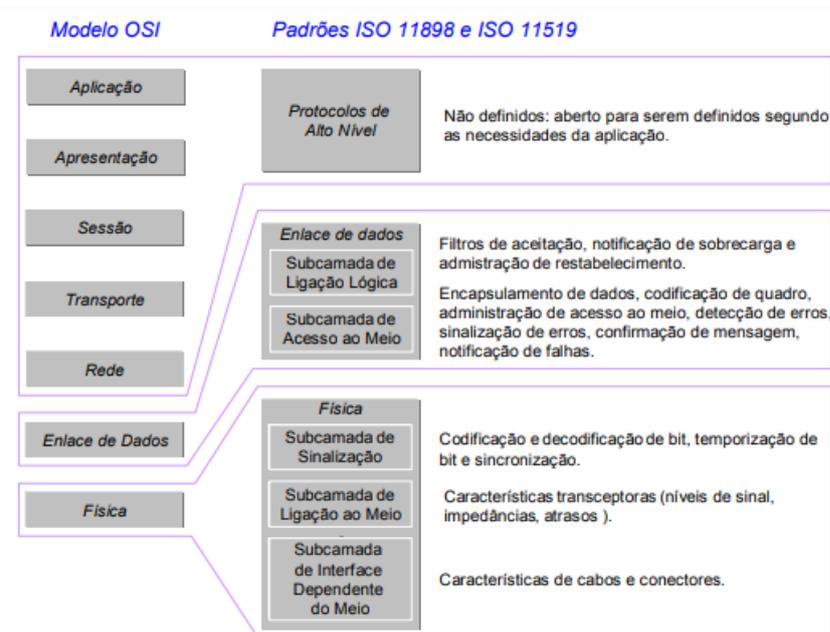


Figura 13 – Relação entre o modelo OSI e os padrões ISO 11898 e ISO 11519

Fonte: (SOUSA, 2002, p. 6)

quando a tensão diferencial entre eles é igual a 2 Volts o bit transmitido é dominante e quando essa tensão for igual a 0 Volt o bit é recessivo, como pode ser visto na Figura 14.

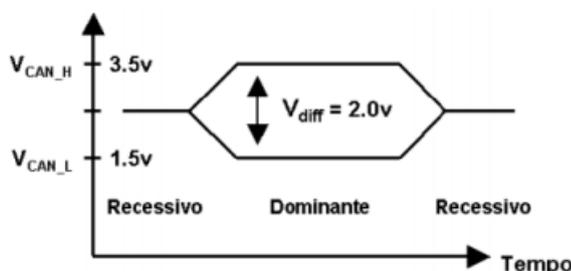


Figura 14 – Níveis lógicos obtidos através da tensão diferencial no barramento

Fonte: (SILVA, 2015, p. 22)

O protocolo CAN utiliza a técnica de *bit stuffing*, que impede a transmissão de mais de cinco bits iguais consecutivos, no envio de uma mensagem, quando o módulo identifica a presença de cinco *bits* consecutivos de mesmo nível lógico, ele próprio adiciona um *bit* de nível lógico diferente (O *bit* adicionado é removido pelas ECUs receptoras, conservando o formato padrão da mensagem), dessa maneira, são formados blocos de 6 em 6 *bits* com propriedades previsíveis e é isso que as ECUs utilizam para a resincronização e para a verificação de erros de transmissão (SOUSA, 2002).

No barramento CAN, a velocidade de transmissão de dados é delimitada pelo comprimento do barramento, quanto maior o comprimento, menor é a velocidade. A

maior velocidade de transmissão especificada para o CAN é de 1Mbps, considerando um comprimento de 40 metros para o barramento. Essa relação é apresentada na Figura 15.

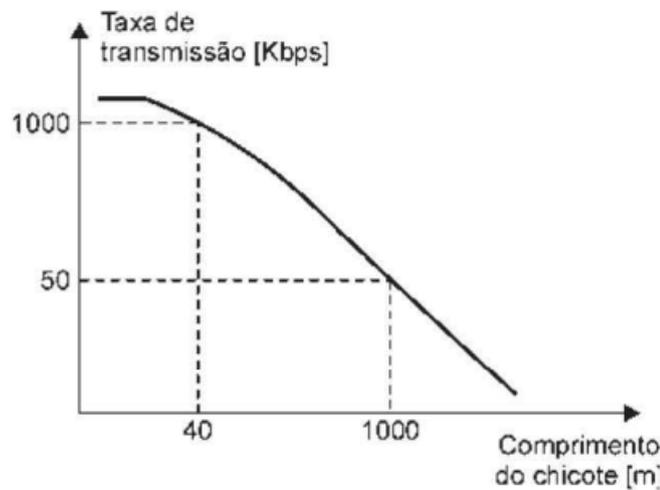


Figura 15 – Relação entre a taxa de transmissão de dados e o comprimento do barramento CAN

Fonte: (GUIMARAES, 2007, p. 217)

2.5.2.2 Camada de Enlace

Na camada de enlace é feito o encapsulamento e desencapsulamento de mensagens, detecção e indicação de erros, e a arbitragem das mensagens é feita. Para a arbitragem das mensagens, caso mais de um módulo esteja mandando alguma mensagem no barramento, o protocolo CAN utiliza o método baseado no CSMA/CD (*Carrier Sense Multiple Access with Collision Detect*), em que o bit dominante se sobrepõe ao bit recessivo no caso de colisão de mensagens. Esse método de arbitragem pode ser visto na Figura 16, nessa figura é ilustrado um caso de colisão de mensagens enviadas pelo módulo 1 e o módulo 2. A partir do SOF (*Start of Frame*), cada *bit* das mensagens possuem o mesmo nível lógico, até que o módulo 1 envia um *bit* dominante e o módulo 2 envia um *bit* recessivo, dessa forma, o segundo módulo entra em modo de espera e o primeiro módulo prevalece continuando a mensagem. Essa arbitragem continua até que haja apenas uma ECU transmitindo no barramento (SILVA, 2015).

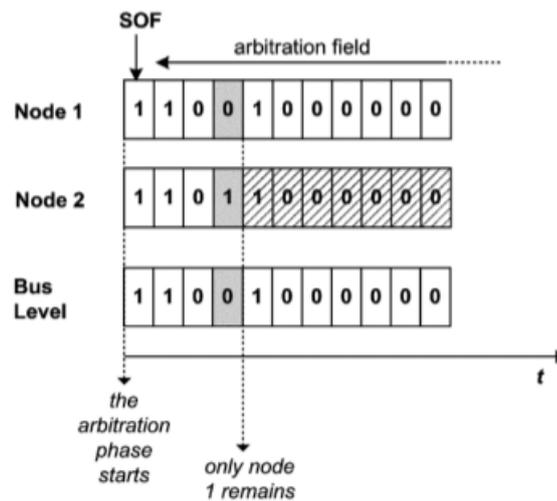


Figura 16 – Método de arbitragem utilizado no barramento CAN para concessão do acesso ao barramento

Fonte: (SILVA, 2015, p. 23)

As mensagens que transitam pelo barramento CAN são denominadas *frames*, eles possuem um formato conhecido, constituído por diferentes campos, cada um deles possui uma funcionalidade. Os *frames* podem ser dos tipos:

- *Frame* de dados: é o tipo de *frame* que carrega informações (encapsuladas) enviadas por uma ECU;
- *Frame* de requisição: *frame* enviado por uma ECU para a requisição de dados que são fornecidos por outra;
- *Frame* de erro: *frame* utilizado para a indicação de erros detectados;
- *Frame* de sobrecarga: utilizado para adicionar tempo entre um *frame* e outro para garantir a prontidão das ECUs, tanto as que estão enviando quando as que estão recebendo.

Na Figura 17 a seguir, os campos de um *frame* de dados são mostrados para o caso da CAN 2.0A. São basicamente sete campos na estrutura desse *frame* e na Tabela 1 cada campo é explicado.

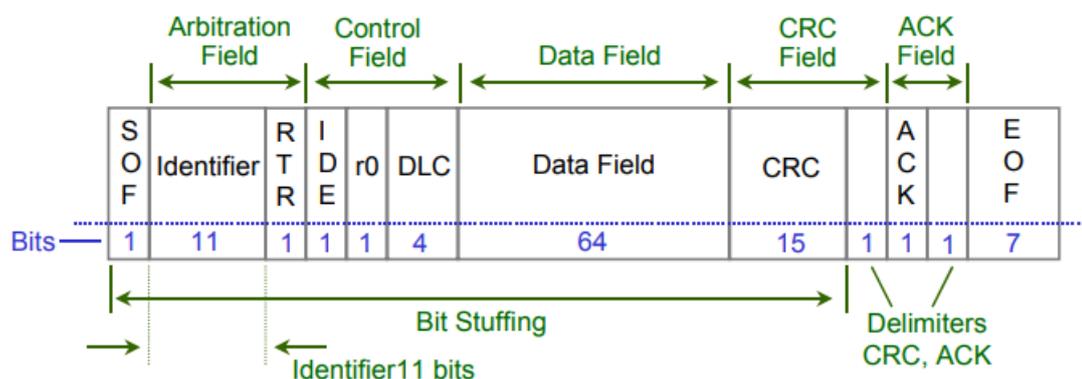


Figura 17 – Campos de um *frame* de dados da CAN 2.0A

Fonte: (SOUSA, 2002, p. 11)

Campo	Subcampo	Funcionalidade
SOF	-	Indicar o início de um <i>frame</i> através de um <i>bit</i> dominante
<i>Arbitration Field</i> Campo de Arbitragem	<i>Identifier</i>	Indicar o <i>ID</i> da mensagem, definindo a prioridade e contribuindo para o endereçamento de mensagens
	RTR <i>Remote Transmission Request</i>	Indicar se o <i>frame</i> é de dados ou de requisição (0 - Dados; 1 - Requisição)
<i>Control Field</i> Campo de Controle	IDE <i>Identifier Extended Bit</i>	Indicar se o <i>frame</i> é do formato padrão ou estendido (0 - Padrão; 1 - Estendido)
	r0	<i>Bit</i> reservado para aplicações futuras não definidas
	DLC <i>Data Length Code</i>	Indicar o número de <i>Bytes</i> presentes no campo de dados
<i>Data Field</i> Campo de Dados	-	Trazer em sua estrutura as informações encapsuladas que se deseja transmitir. Organizando os <i>Bytes</i> do mais significativo para o menos significativo
<i>CRC Field</i> Campo CRC	CRC <i>Cyclic Redundancy Check</i>	Detectar erros utilizando a verificação de redundância cíclica
	<i>Bit</i> recessivo	Indicar o fim do CRC
<i>ACK Field</i> Campo ACK	ACK <i>Acknowledge</i>	<i>Bit</i> recessivo utilizado para a confirmação de recebimento do <i>frame</i> por outro módulo
	<i>Bit</i> recessivo	Indicar o fim do ACK
EOF <i>End of Frame</i>	-	Fim do <i>frame</i> - conjunto de 7 <i>Bits</i> recessivos que informam o fim do <i>frame</i>

Tabela 1 – Funcionalidade de cada campo de dados de um *frame* padrão

Detecção de erros

No total o protocolo CAN utiliza cinco métodos de detecção de erros, a nível de *bit* utiliza: o monitoramento de *bit* e o *bit stuffing*, e a nível de mensagens são utilizados

o *Cyclic Redundancy Check*, a checagem de *frame* e o *Acknowledge*. Caso aconteça a identificação de algum erro, as centrais envolvidas podem tomar as ações de diagnose específicas para cada tipo detectado (BRUDNA, 2000).

O *bit stuffing* já foi explicado anteriormente de forma completa, a seguir estão dispostas as explicações sobre os demais métodos.

- Monitoramento de *bit*: quando uma ECU está enviando uma mensagem, ela também está verificando o estado do barramento, pois, caso esteja enviando um *bit* dominante e do barramento seja lido um *bit* recessivo, isso indica um problema, logo, um *frame* de erro é enviado pela ECU.
- *Cyclic Redundancy Check*: utilizando um polinômio definido, uma ECU emissora calcula um valor utilizando os *bits* enviados na mensagem, a ECU receptora também calcula usando os *bits* lidos, utilizando o mesmo polinômio, se o valor calculado pelas duas for diferente, uma mensagem de erro é enviada, pois houve algum problema na transmissão
- Checagem de *frame*: campos específicos do *frame*, devem possuir valores fixos de *bit*, como o *bit* (dominante) de início de mensagem por exemplo, então as ECUs receptoras verificam esses campos, se não estiverem de acordo, uma mensagem de erro é transmitida
- *Acknowledge*: quando uma ECU envia uma mensagem no barramento, no campo ACK é enviado um *bit* recessivo para verificar se a mensagem foi recebida, dessa forma, quando as ECUs receptoras recebem o *frame* sem erros, de forma síncrona, enviam um *bit* dominante para confirmar o recebimento, assim, o módulo que transmitiu a informação também tem conhecimento da chegada da mensagem sem erros.

Dentre os assuntos já citados anteriormente, a instrumentação tem relação direta com o tema do trabalho em questão, já que a base dos sistemas construídos envolve a utilização de sensores, cruciais para o pleno funcionamento do todo. Diante disso, é importante a existência da conceituação dessa área do conhecimento, como está a seguir.

2.6 Instrumentação

A Instrumentação é uma área de estudo que desenvolve e aplica instrumentos de medição, assim como técnicas para a adequação dos instrumentos a cada aplicação. A finalidade de um sistema de medição é obter informações sobre o valor da grandeza física que se deseja medir. Os sistemas de medição podem ser classificados de acordo com seus objetivos: sistemas de medição para verificação de grandezas e sistemas de medição para aplicações em malha fechada.

Os sistemas de medição para verificação de grandezas tem como objetivo principal, fornecer o valor da grandeza física do *mesurando*, normalmente para análise e/ou monitoramento, ou seja, são aplicados em malha aberta, como pode ser visto na Figura 18. Já os sistemas de medição para aplicações em malha fechada, são utilizados para realimentar o valor da grandeza física medida na malha fechada, para que dessa maneira, o sistema possa tomar decisões de maneira independente, Figura 19 (AGUIRRE, 2013).

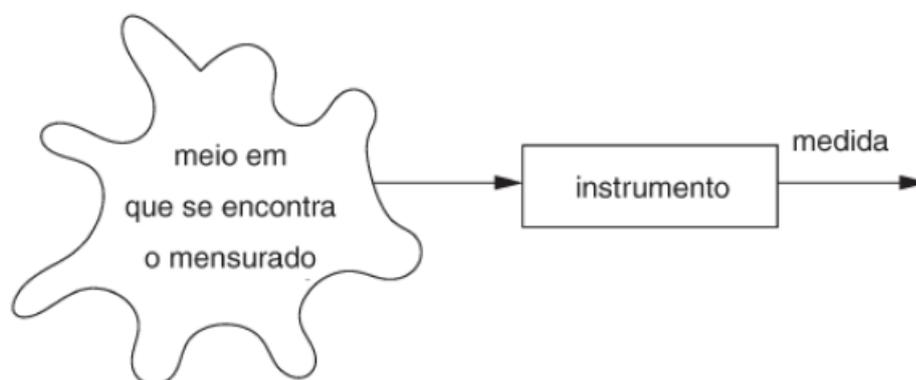


Figura 18 – Sistema de medição para verificação de grandezas

Fonte: (AGUIRRE, 2013, p. 4)

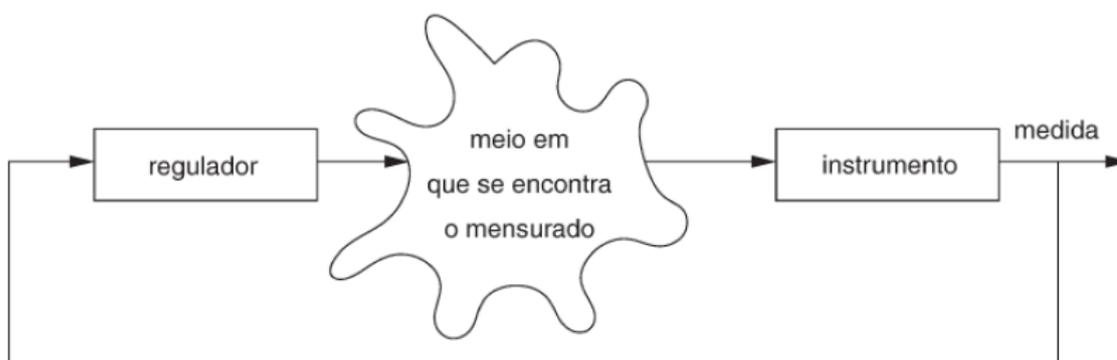


Figura 19 – Sistema de medição para aplicações em malha fechada

Fonte: (AGUIRRE, 2013, p. 4)

Um sistema de medição é normalmente constituído por quatro elementos: sensor, condicionador de sinal, processador de sinal e apresentador de dados (visor). Transdutores são dispositivos que transformam uma forma de energia em outra, diante disso, os sensores pertencem a uma classe dos transdutores. A particularidade dos sensores é a transmissão de informações, a conversão de um grandeza física em um sinal elétrico é principal propriedade dos sensores. Os sensores podem ser divididos em passivos e ativos, os passivos não necessitam de energia adicional para gerarem um sinal elétrico diante um estímulo

externo vindo do *mesurando*, já os ativos precisam de uma fonte de energia externa para seu funcionamento (BALBINOT; BRUSAMARELLO, 2010).

2.7 Identificação de Sistemas

Identificação de sistemas é o processo de obter um modelo matemático de um sistema a partir de dados experimentais. Um sistema é qualquer processo que transforma um conjunto de entradas em um conjunto de saídas, e pode ser encontrado em muitas áreas, como engenharia, física, biologia, economia, entre outras.

O objetivo da identificação de sistemas é obter um modelo matemático que represente o comportamento dinâmico do sistema, de forma que possa ser utilizado para prever o comportamento do sistema em diferentes condições, projetar controladores ou otimizar o desempenho do sistema, de forma a entender como as variáveis envolvidas se relacionam entre si. (LJUNG, 1999). O modelo matemático pode ser de vários tipos, como modelos lineares ou não lineares, de tempo contínuo ou discreto, e de diferentes graus de complexidade. A identificação de sistemas pode ser feita por diferentes métodos, incluindo técnicas estatísticas, técnicas baseadas em modelos físicos, técnicas baseadas em redes neurais, entre outras.

De acordo com Ljung (1999), é adequado caracterizar todos os subcomponentes de um sistema de forma separada, sempre que possível, para determinar suas características matemáticas e, posteriormente, uni-las em uma só expressão que descreva o sistema como um todo. Entretanto, ainda que esse método de identificação se apresente como eficaz na maioria dos casos, o sistema nunca será descrito de forma perfeita, uma vez que o seu funcionamento na prática está sujeito a uma série de fatores que escapam ao rigoroso matemático, como atritos, folgas em sistemas mecânicos, fatores ambientais, dentre outros.

O modelo matemático mais simples que correlaciona uma entrada com uma saída é dado na forma de uma equação diferencial linear, como pode-se observar na equação 2.25:

$$y(t) + a_1y(t - 1) + \dots + a_ny(t - n) = b_1u(t - 1) + \dots + b_mu(t - m) \quad (2.25)$$

Sendo a e b constantes do sistema, y as saídas e u as entradas.

Existem três etapas gerais a serem seguidas para identificar um sistema, são elas: a obtenção de dados, que se dá comumente de forma experimental, a escolha de modelos existentes que possam vir a se tornar o que melhor descreve o sistema, e a comparação dos dados obtidos com cada modelo selecionado, para proceder à escolha mais ajustada para o sistema.

Nesta etapa, foram apresentados trabalhos com objetivos semelhantes e revisadas as principais bibliografias relacionadas aos temas deste trabalho. Primeiramente, abordou-se os conceitos dos sistemas que foram objeto de controle: o de direção e o de aceleração e frenagem, de forma a trazer suas principais características, os seus funcionamentos e os tipos mais usuais. Posteriormente, para estabelecer um sistema de controle efetivo, foi abordada a teoria relacionada aos sistemas de controle, com seus principais conceitos básicos, seus equacionamentos relacionados ao desempenho e à estabilidade e aos modelos de controladores mais comuns, sendo o que foi aplicado ao projeto o tipo Proporcional Integral Derivativo (PID). Por fim, foi explanada a teoria a respeito do protocolo de comunicação CAN, de forma a explicar seu conceito, suas características, seu funcionamento e sua aplicação.

3 Metodologia

Este capítulo apresentará os métodos e as ferramentas utilizadas para desenvolver o trabalho nas seguintes etapas: modelagem mecânica, arquitetura eletroeletrônica e simulação e testes. São apresentadas as montagens, o desenvolvimento dos sistemas e os componentes utilizados.

3.1 Modelagem Mecânica

Esta etapa consistiu no desenvolvimento e na modelagem dos sistemas mecânicos necessários para a instalação do sensor de ângulo do sistema de direção e da roda dentada para medição de velocidade do veículo. Para o desenvolvimento dos modelos, foi utilizado o *software* CATIAV5 e posteriormente as partes foram impressas utilizando uma impressora 3D (Figura 20).



Figura 20 – Impressora 3D Tevo Tornado

Fonte: Google Imagens

O sistema mecânico associado à direção foi composto por 5 elementos principais: uma caixa para englobar o sistema; uma engrenagem de 24 dentes; uma engrenagem de 18 dentes; dois pés de fixação para a caixa; uma plataforma para o sensor e um potenciômetro linear B10K. Esses componentes podem ser observados na figura 21. No apêndice B, são apresentados os desenhos técnicos de cada componente referenciado, bem como a vista explodida do conjunto.

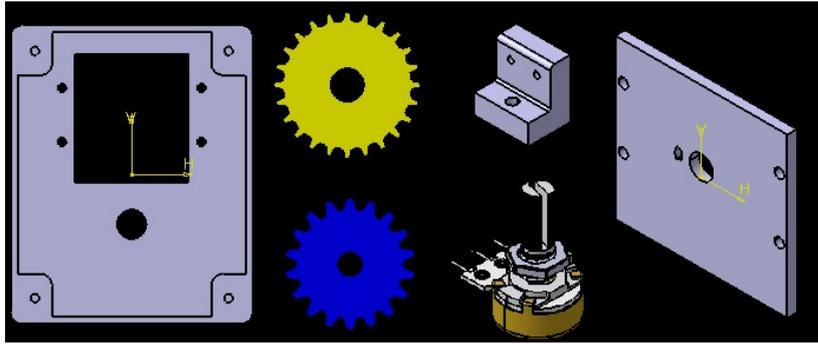


Figura 21 – Componentes

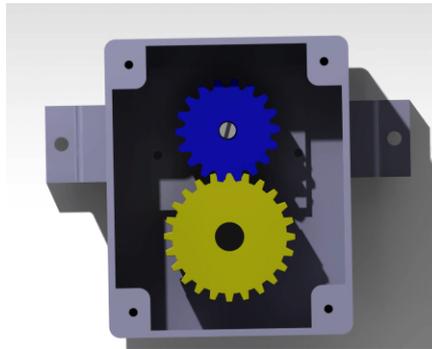


Figura 22 – Sistema mecânico da direção montado

A engrenagem de 24 dentes está acoplada na coluna de direção e a engrenagem de 18 dentes está associada à engrenagem da coluna e acoplada no sensor potenciômetro, como se nota na figura 22. Essa relação entre as engrenagens foi necessária para a transmissão do movimento do giro do volante para o potenciômetro. O potenciômetro é um sensor que mede deslocamento angular e que conta com uma trilha de material resistivo, com um cursor e com três terminais (VCC, GND, Sinal). Com o giro do da coluna de direção, a engrenagem amarela se movimenta, provocando a movimentação da engrenagem azul, na qual o sensor está acoplado. Utilizando a função *map* do Arduino para transformar da escala de bits para de ângulo e com a relação de giro entre a roda e o potenciômetro, é possível obter os valores de ângulo na faixa de ângulo adequada.

Dessa forma, foi possível desenvolver um procedimento experimental para obter a relação entre o giro do potenciômetro e o esterçamento das rodas. Para tanto, o veículo de pequena escala foi colocado no chão e, sob suas rodas dianteiras, foram colocadas folhas em branco. A partir disso, foi traçada uma reta na posição de origem da roda, com o volante no centro, estabelecendo uma posição de referência. O volante foi esterçado totalmente para a direita e uma nova reta foi traçada a partir da nova posição da roda. Esse procedimento foi repetido para o outro lado, e resultou em um ângulo de esterçamento próximo de 25°, após uma média de cinco medições, para ambas as rodas. A figura 23

mostra, de forma esquematizada, como foi executado esse experimento. A figura 24 mostra como o procedimento foi executado na prática. A figura 25 mostra a média das medições de ângulo para cada roda.

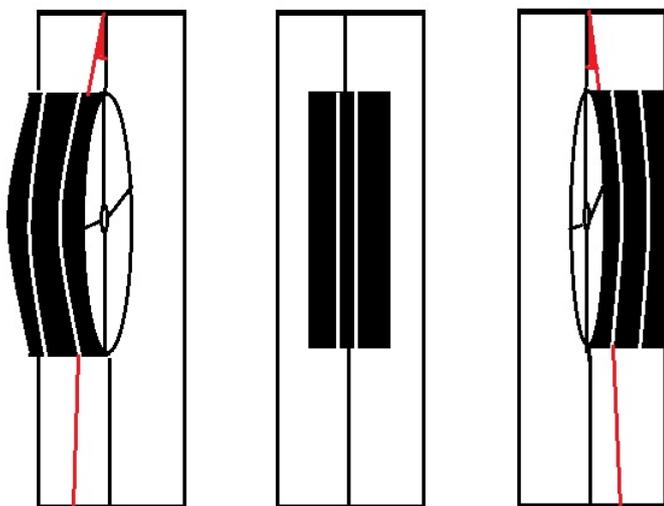


Figura 23 – Esquemático do experimento



(a)



(b)

Figura 24 – Procedimento real

Roda	Medição 1	Medição 2	Medição 3	Medição 4	Medição 5	Média
Esquerda	27°	24°	25°	26°	24°	25,2°
Direita	25°	25°	25°	25°	25°	25°

Figura 25 – Média das medições para os ângulos de esterçamento das rodas

Com isso, o sistema mostrado na figura 22 foi montado no veículo e o volante foi esterçado totalmente para um lado. Assim, foi possível obter quantos graus o potenciômetro girava. Foi constatada uma leitura do sensor em um valor de aproximadamente 37°, enquanto o esterçamento da roda era de, em média, 25°. A relação entre giro do sensor e giro da roda encontrada foi de 1.48°, isto é, para cada 1° grau de roda, o potenciômetro girava 1.48°. A figura 26 apresenta um fluxograma explanatório de todo processo descrito.

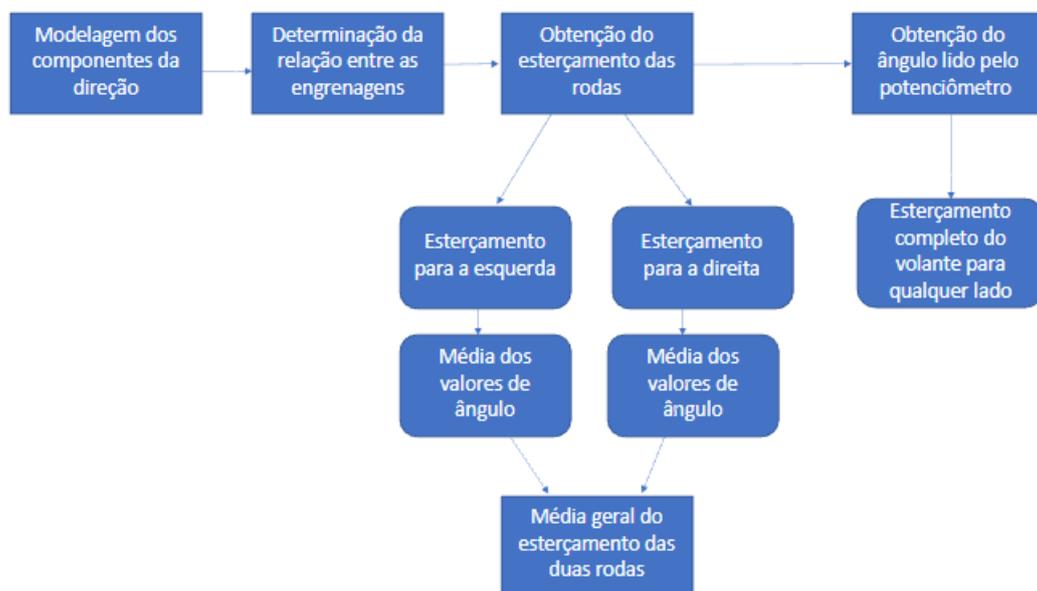


Figura 26 – Fluxograma da modelagem mecânica da direção

O sistema mecânico correspondente ao sistema de aceleração e frenagem foi composto por quatro elementos principais: duas caixas de proteção do sensor; dois suportes da caixa de proteção; dois sensores encoder de velocidade e dois discos perfurados. Esses elementos podem ser observados na Figura 27.

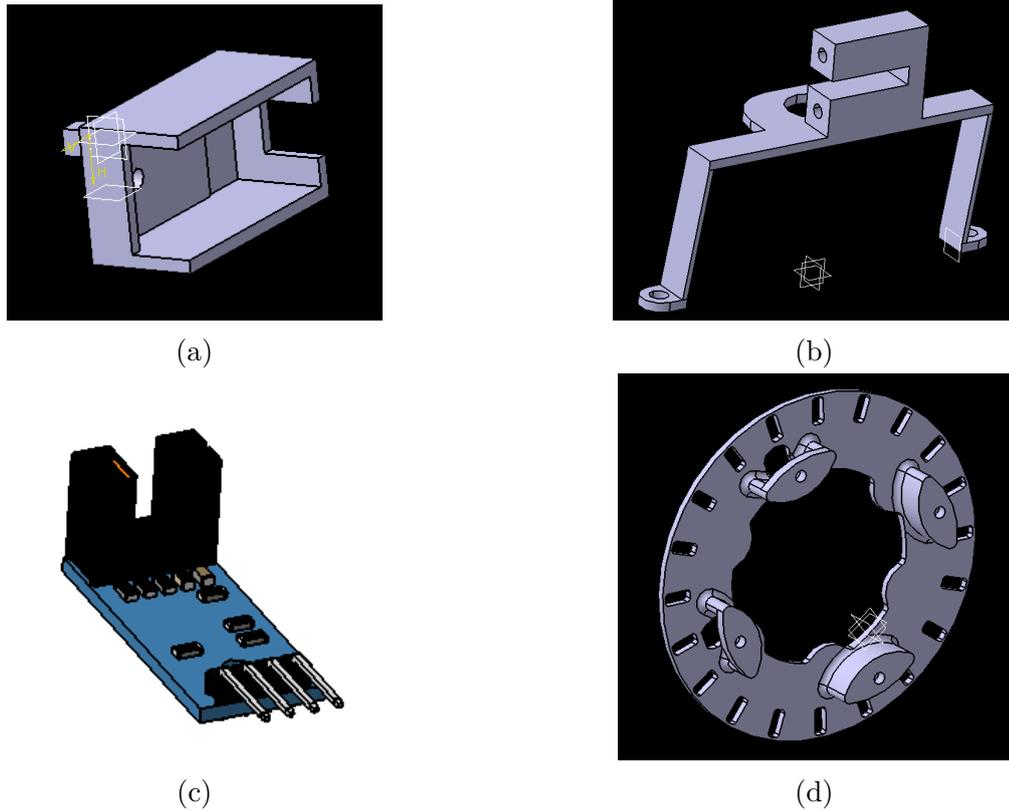
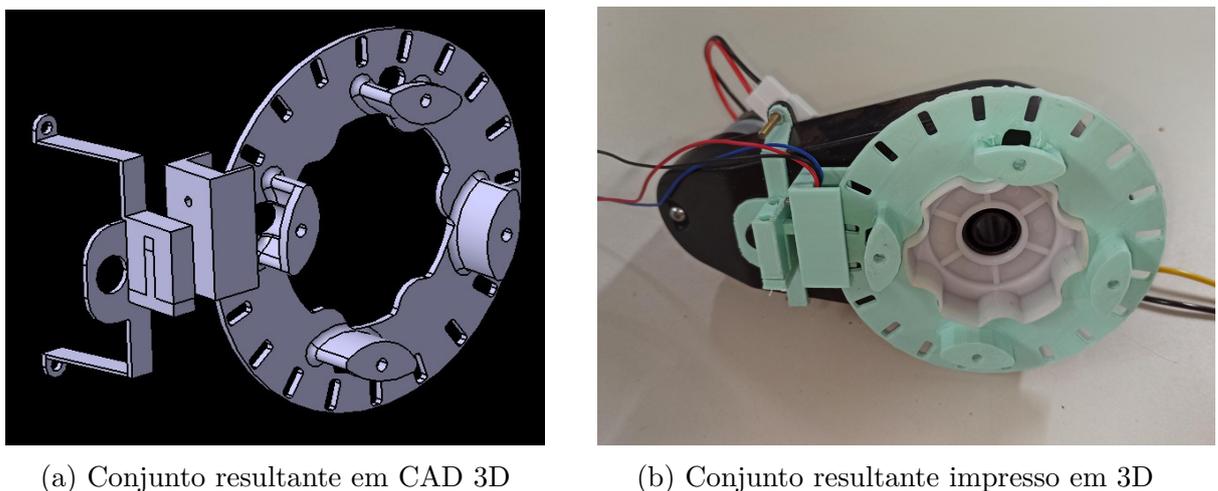


Figura 27 – Componentes do sistema de aceleração e frenagem

O conjunto resultante da união dos componentes, em CAD 3D e já impressos utilizando a impressora 3D, estão apresentados na Figura 28.



(a) Conjunto resultante em CAD 3D

(b) Conjunto resultante impresso em 3D

Figura 28 – Conjunto resultante da união dos componentes

O primeiro passo na construção desse sistema foi desenvolver uma estrutura que posicionasse o sensor encoder na posição correta para que os furos do disco encoder passassem justamente onde o sensor possui um emissor de luz infravermelho e um sensor infravermelho, Figura 29. Então, com o disco perfurado preso à roda do veículo, quando a

roda gira, o disco também gira, dessa maneira, o sensor consegue contar pulsos ao longo do tempo.

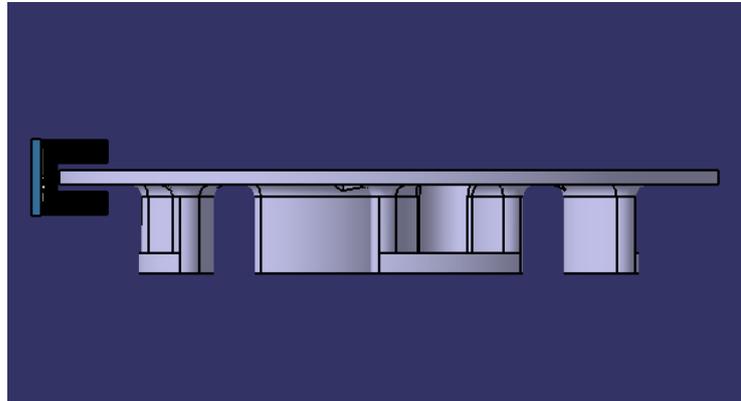


Figura 29 – Posicionamento Sensor-Disco

Através disso, calcula-se a velocidade em RPM, verificando o tempo entre um pulso e outro, sabendo que o disco possui 20 furos distribuídos igualmente ao longo de 360 graus. E essa montagem pode ser melhor vista na Figura 28.

Com o código feito para o Arduino, é possível então, medir a velocidade de rotação de cada roda do veículo de pequena escala, com isso, também implementar o controlador PID e criar uma rede CAN simples utilizando outro Arduino e os demais componentes necessários. Em bancada, foi possível descobrir que a velocidade máxima alcançada pelos motores desse sistema é de 115 RPM e em testes fora de bancada a velocidade máxima é de 110 RPM. Para a implementação da rede CAN simplificada utilizou-se apenas um byte do campo de dados de um *frame* CAN e esse *byte* assume um valor mínimo de 0 e máximo de 220, isso para que sejam usados apenas valores positivos de velocidade requerida, pois o motor pode girar no sentido horário e antihorário com a velocidade de 110 RPM.

3.2 Arquitetura Eletroeletrônica

Esta etapa consistiu no desenvolvimento de toda a estrutura eletrônica necessária para exercer o controle do veículo de pequena escala e estabelecer a comunicação entre os sistemas. Para tanto, foram utilizados equipamentos, como Arduino, ponte-H, *jumpers*, módulos de comunicação CAN e fonte de energia 12V, além de sensores para adquirir os dados, como um potenciômetro para a direção e um encoder de velocidade para a aceleração e frenagem. Os principais componentes podem ser visualizados na figura 30.

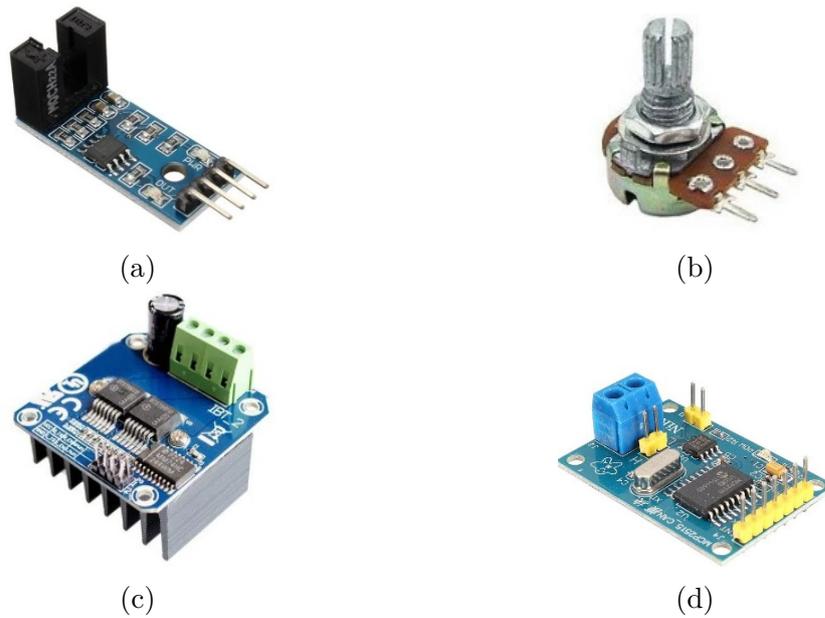
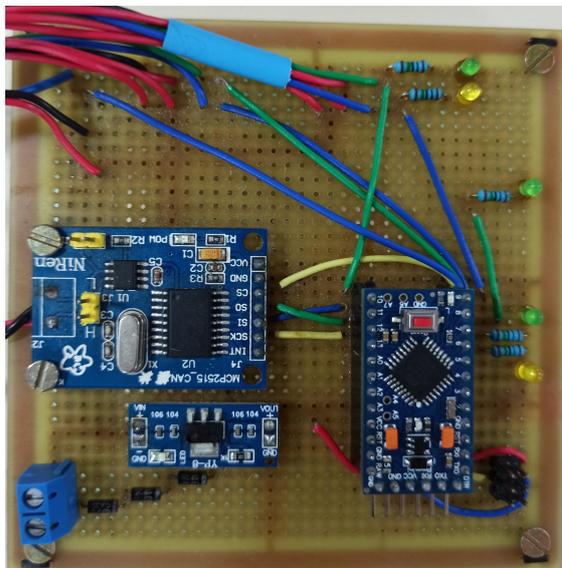
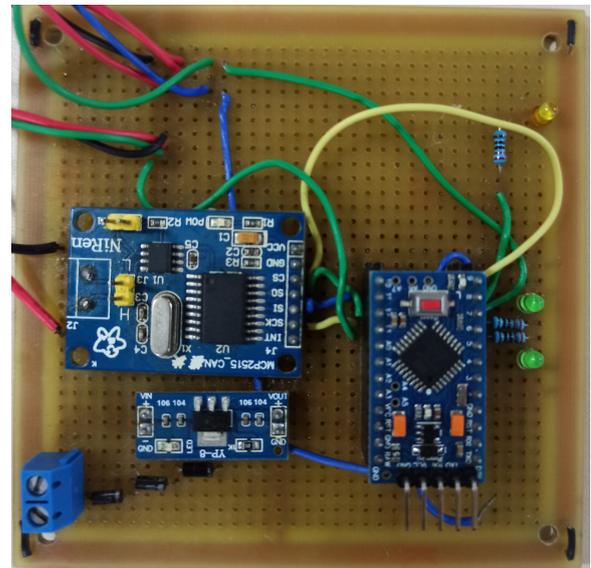


Figura 30 – Componentes principais da arquitetura eletroeletrônica

Outro procedimento eletroeletrônico obtido foi a montagem das ECUs de controle de cada sistema, utilizando uma placa perfurada e componentes como Arduinos Pro Mini, reguladores de tensão, módulos de comunicação CAN e diodos. Essa montagem pode ser verificada na Figura 31 e foi feita para que os sistemas de controle fossem embarcados no veículo e estivessem aptos a atuarem sob comandos via controle de radiofrequência.



(a) ECU do sistema de aceleração e frenagem



(b) ECU do sistema de direção

Figura 31 – ECUs construídas

O circuito montado para executar os testes no sistema de direção foi o esquematizado na figura 32.

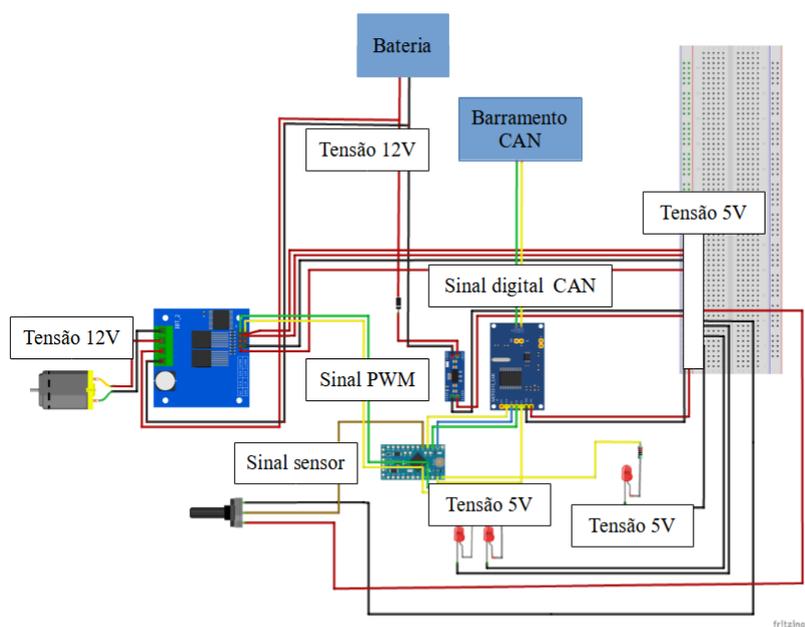


Figura 32 – Montagem do sistema de direção

O circuito montado para executar os testes no sistema de aceleração e frenagem foi o esquematizado na figura 33.

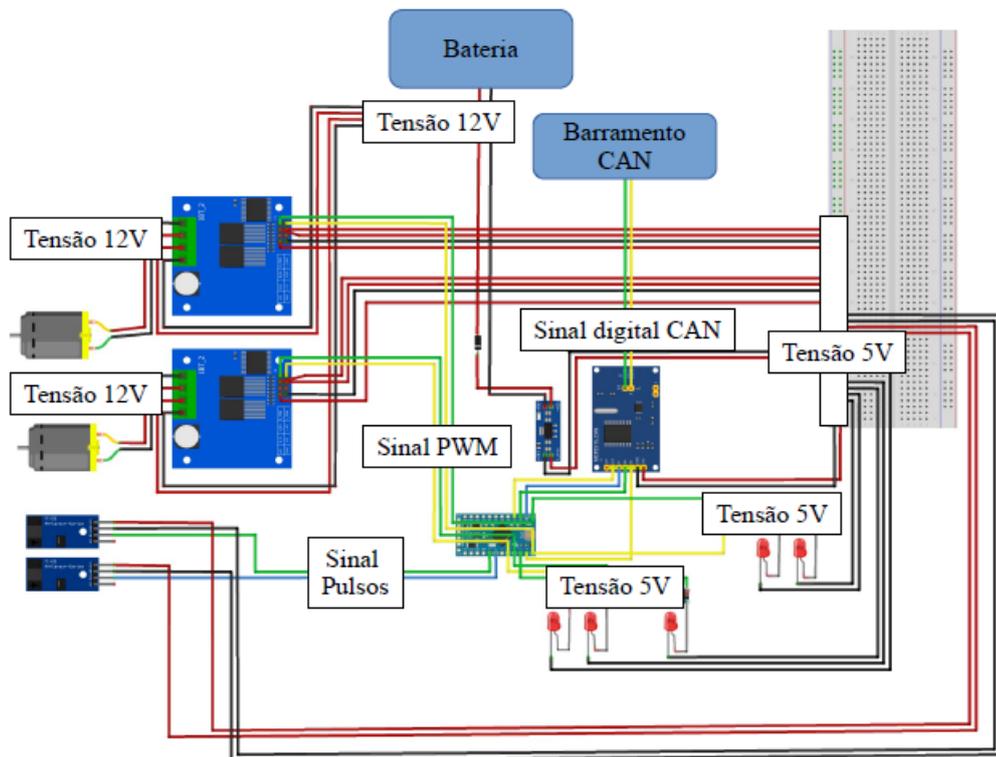
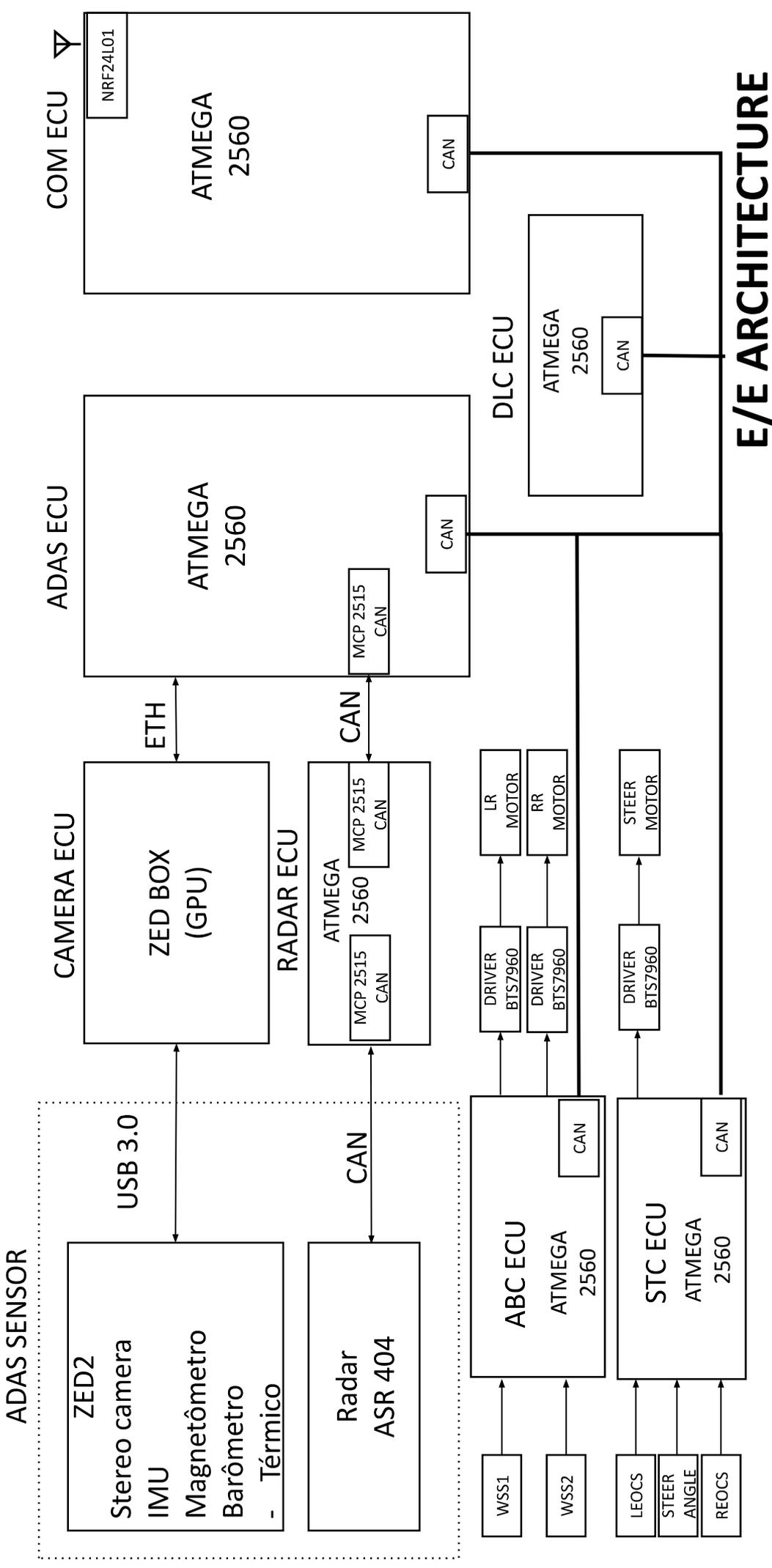


Figura 33 – Montagem do sistema de aceleração e frenagem

A seguir, é apresentado o diagrama referente à Arquitetura Eletroeletrônica final para o veículo de pequena escala do Projeto SecurAuto da Faculdade UnB-Gama. A arquitetura desenvolvida até o momento é semelhante à apresentada, a ECU do Sistema de direção é a STC e a ECU do Sistema de aceleração e frenagem é a ABC, o que difere entre elas são os microcontroladores utilizados até o momento, ATmega328P. Os componentes utilizados estão apresentados na tabela 2.



Componente	Componente Interno	Modelo/ Especificação
ECU DIREÇÃO	Arduino	Pro mini 5 V
	Módulo CAN	MCP2515
	Regulador de tensão	YP-8 5 V
Módulo driver - Direção	-	IBT_2 BTS7960
Motor CC - Direção	-	8000 RPM
Sensor de ângulo	-	Potenciômetro linear B10K
ECU ACEL/FRENAGEM	Arduino	Pro mini 5 V
	Módulo CAN	MCP2515
	Regulador de tensão	YP-8 5 V
2 Módulos driver - ACEL/FRENAGEM	-	IBT_2 BTS7960
2 Motores CC - ACEL/FRENAGEM	-	12000 RPM
2 Sensores de velocidade	-	Sensor de velocidade Encoder - LM393
ECU COMUNICAÇÃO	Arduino	Pro mini 5 V
	Módulo CAN	MCP2515
	Regulador de tensão	YP-8 5 V
	Regulador de tensão	3.3 V AMS1117
	Módulo transceptor de radiofrequência	NRF24L01
RÁDIO CONTROLE	Arduino	Nano 5 V
	Regulador de tensão	YP-8 5 V
	Regulador de tensão	3.3 V AMS1117
	2 Módulos joystick analógico	KY - 023
	4 Pushbuttons	Botão push button com capa
	2 Chaves seletoras	2 Posições
	2 Potenciômetros	Potenciômetro linear B1K
	Conversor de nível	5 V - 3.3 V 8 canais
Chave gangorra ON/OFF	-	

Tabela 2 – Componentes Arquitetura eletroeletrônica

Dessa forma, com o *hardware* devidamente construído, foi possível iniciar o desenvolvimento de software para cada um dos subsistemas abordados neste trabalho. O tópico seguinte apresenta esse processo, por meio de fluxogramas e da exposição detalhada a respeito dos procedimentos.

3.3 Desenvolvimento da Lógica de Controle

Com os sistemas modelados e montados em bancada e com a programação desenvolvida na interface do Arduino (Apêndice A), foi possível estabelecer uma série de testes, necessários para projetar o controlador, de forma a atender aos requisitos do projeto. Assim, este capítulo apresenta a explanação das lógicas de controle e o ciclo de desenvolvimento dos controladores.

3.3.1 Sistema de Direção

Para o sistema de direção, o funcionamento do código é baseado no recebimento de um comando via rede CAN, comando que, dentre os 8 *bytes* da mensagem, se encontrava no *byte* 6 e que variava dentro de uma escala de 0 a 50, de forma a não se trabalhar com valores negativos no barramento, o que pode ser observado na figura 34.

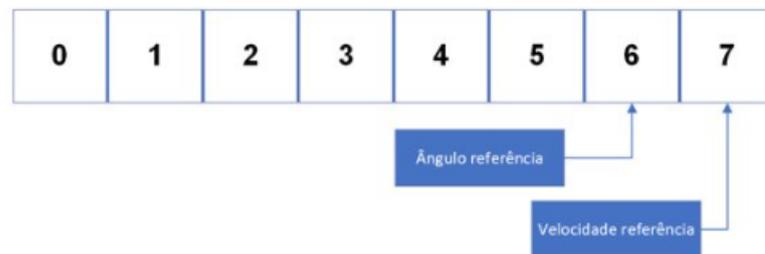


Figura 34 – Disposição de dados na mensagem

Como as rodas esterçam de -25° até 25° , para que o controlador atuasse nessa escala, a variável identificada como *setpoint* usava o valor recebido no *byte* 6 subtraído de 25. Após receber esse comando, o programa calcula o ângulo da roda, fazendo uma média de valores, por meio da função 'calculaangulo', para conferir maior estabilidade na leitura do potenciômetro e para obter a posição atual das rodas. O procedimento esquematizado pode ser entendido na figura 35.

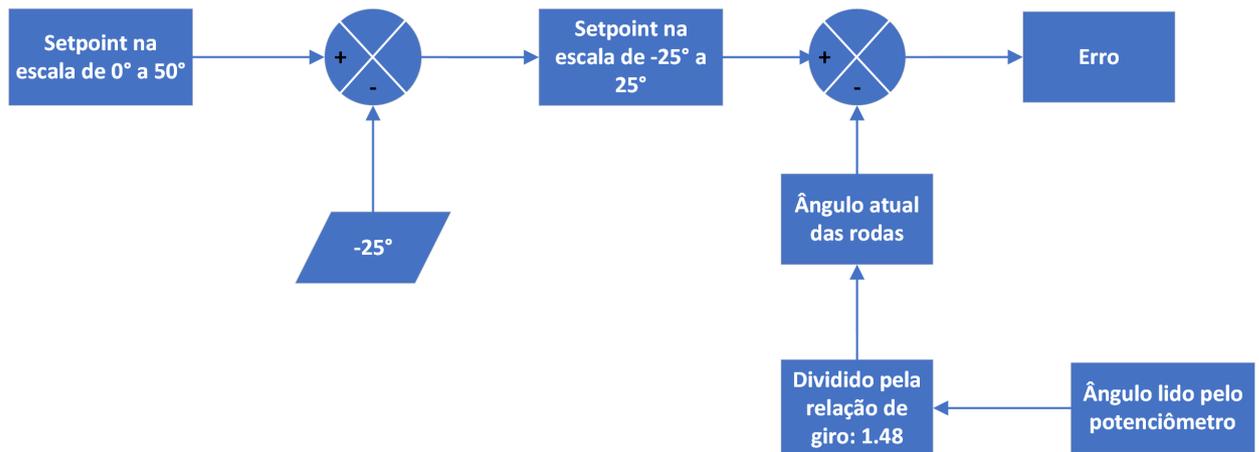


Figura 35 – Fluxograma representativo da transformação da escala do ângulo de direção

Analisando a teoria apresentada a respeito dos controladores e ponderando o que melhor se aplicaria ao sistema de direção, por meio de testes realizados, o controlador escolhido foi o PID, devido à presença dos três ganhos, que contribuem para o melhor desempenho do sistema. No programa, o erro foi considerado a diferença entre o *setpoint*, isto é, o ângulo desejado, e o ângulo atual da roda. Após a ação de controle ser calculada de acordo com os parâmetros k_p , k_i , k_d , ela é aplicada em uma função que retorna um valor de PWM a ser aplicado no motor de corrente contínua (Figura 36).

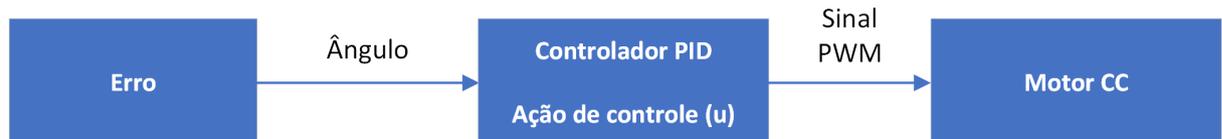


Figura 36 – Fluxograma representativo da aplicação da ação de controle no motor CC

A lógica de controle do sistema de direção está representada no fluxograma da figura 37.

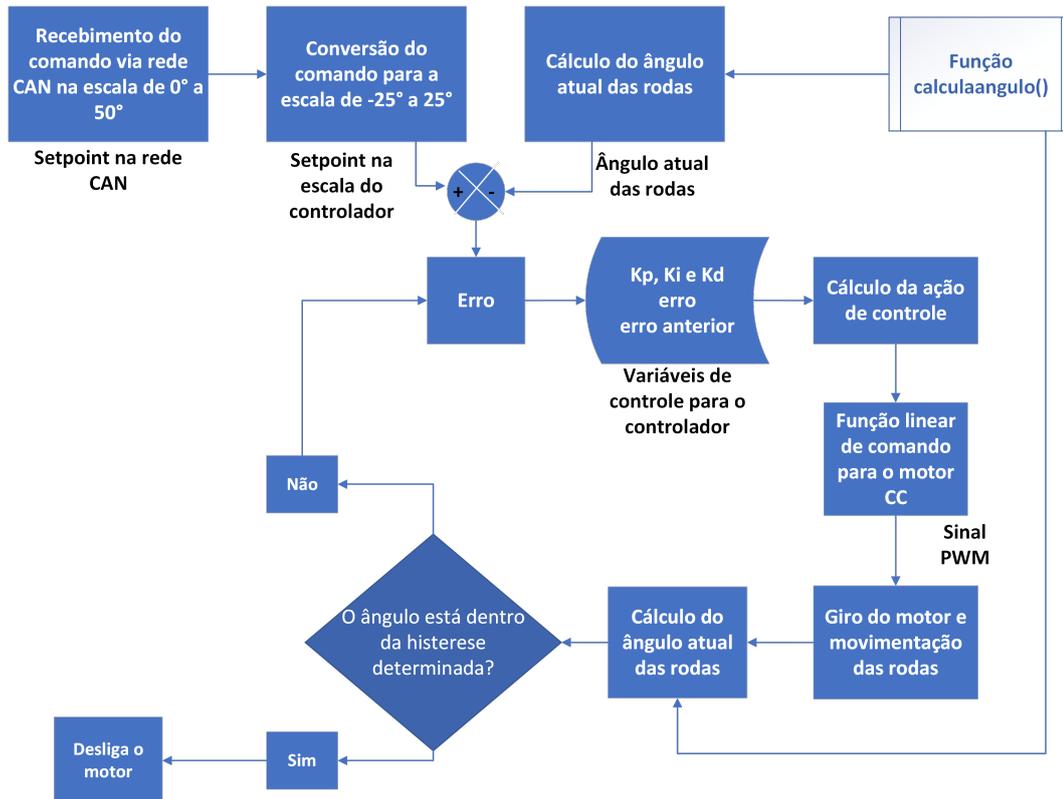


Figura 37 – Fluxograma da lógica de controle para a direção

3.3.2 Sistema de Aceleração e frenagem

A Figura 38 apresenta um fluxograma da lógica do código principal embarcado na ECU do Sistema de Aceleração e Frenagem.

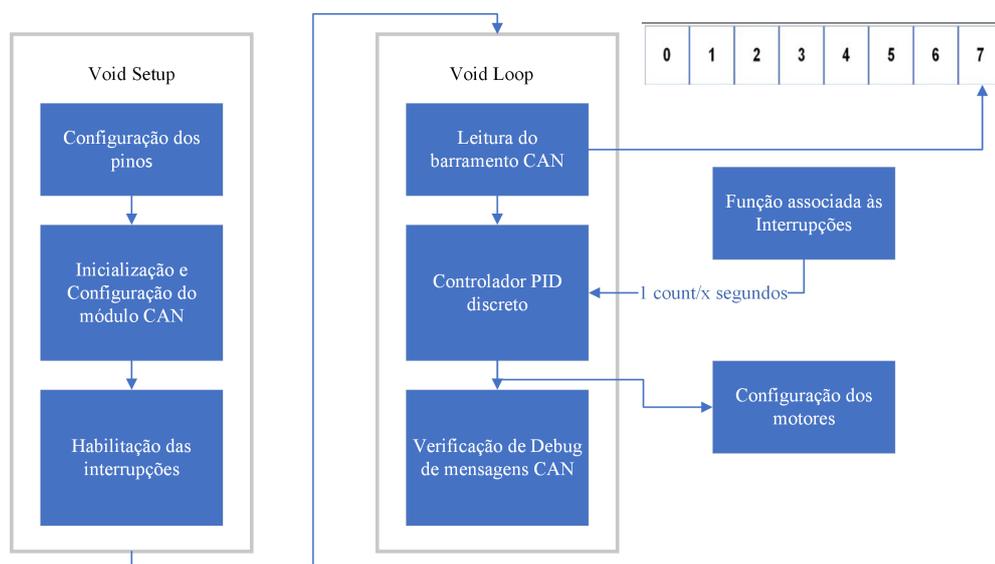


Figura 38 – Fluxograma do código da ABM

No início do código são configurados os pinos utilizados, o módulo MCP2515 é inicializado e configurado e as interrupções são habilitadas para cada sensor, chamando uma função que calcula o intervalo de tempo entre pulsos na subida, para assim a velocidade poder ser calculada através da divisão por 1200, que corresponde ao número de furos no disco perfurado (20) multiplicado por (60) para converter de segundos para minutos, Equação 3.1.

$$v = \frac{\frac{1}{t}}{20 \times 60} \quad (3.1)$$

Onde:

v - velocidade [RPM];

t - intervalo de tempo entre um pulso e outro [segundos].

Após isso, o Void Loop se inicia com a leitura do barramento CAN, com o objetivo de armazenar a mensagem com ID 0x100, cuja velocidade referência está presente no byte 7 do campo de dados, bastando subtrair o valor inteiro por 110 para adequar a escala -110 a 110.

O dicionário de dados e a matriz de comunicação da arquitetura eletroeletrônica podem ser vistos nas Tabelas 3 e 4 seguintes.

Dicionário de dados							
COM	M1: Identificador (0x100)						
ABM							
STM							
Mensagem	Posição (Byte)	Início (Bit)	Comprimento (Bits)	Nome do parâmetro	Faixa (Alcance)	Resolução	Offset
M1	6	1	1 Byte	Ângulo referência	-25 a 25	1 deg/bit	-25
M1	7	1	1 Byte	Velocidade referência	-110 a 110	1 rpm/bit	-110

Tabela 3 – Dicionário de dados

ECU	Mensagem	Sinal	COM ECU	ABM ECU	STM ECU
COM	M1	Direção e Velocidade	T	R	R
ABM					
STM					

Tabela 4 – Matriz de comunicação

Até esse ponto, já foi obtida a velocidade referência e a velocidade medida também já foi calculada através dos *interrupts*, portanto, o controlador PID discreto já possui *Setpoint* e *Input*, possibilitando o cálculo do *Output*.

Por fim, os motores são alimentados através do *Output* calculado e passado, na escala de um sinal PWM, aos *Drivers* Ponte H. E também, é feita uma lógica de *debug* para acender um LED quando não é recebida nenhuma mensagem do barramento após 1 segundo.

Com as lógicas dos código de controle apresentadas, abaixo estão explicados os ciclos de desenvolvimento dos controladores.

3.3.2.1 Ciclo de Desenvolvimento do Controlador

O controlador escolhido para o controle do sistema de direção foi o do tipo PID. A escolha desse tipo de controlador se baseou no fato de que ele oferece uma maior gama de parâmetros de controle, utilizando ações proporcionais, integrais e derivativas. Como o sistema de direção do veículo de pequena escala se mostrou muito sensível à distúrbios externos, especialmente ao atrito estático da superfície em que ele se encontrava, o PID foi o controlador que mais amenizou esses efeitos e tornou a atuação do sistema aceitável.

Na tentativa de se utilizar apenas um conjunto de parâmetros para toda a faixa de operação de ângulos do sistema, a resposta se mostrava satisfatória para pequenos valores de ângulo, mas para altos valores apresentava-se imprecisa e, em muitos testes, não robusta. Então, uma sintonização do PID foi feita a partir do método de Agendamento de Ganhos (*Gain Scheduling*), que consiste em definir diferentes valores de parâmetros k_p , k_i e k_d para diferentes faixas de operação do sistema. A lógica do chaveamento implementada no sistema de direção, a partir de dois conjuntos de parâmetros, pode ser observada na figura 39.

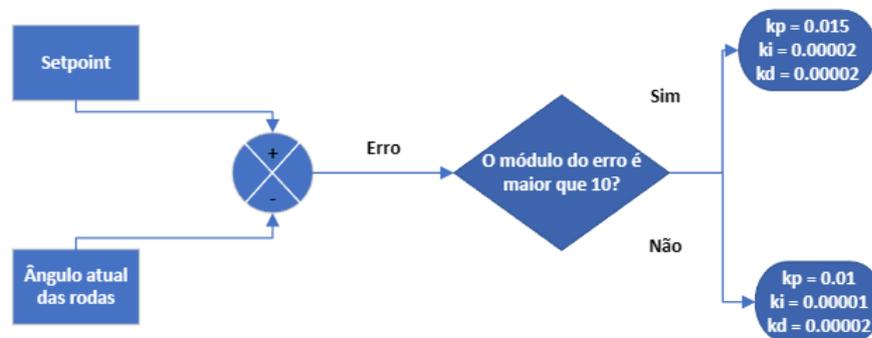


Figura 39 – Fluxograma da lógica de agendamento de ganhos

Os valores expostos no fluxograma da figura 39 foram encontrados após uma série de testes e suposições iniciais. Essas simulações e seus respectivos resultados são apresentados abaixo, na seção de Resultados.

Como já foi dito, o controlador utilizado no sistema de aceleração e frenagem foi um controlado PID discreto, devido sua simplicidade e eficiência. O controlador utilizado possui a forma expressa na Equação 3.2.

$$Output = Kp \cdot erro + (Ki \cdot T \cdot erro + parcelaintegralanterior) - \frac{Kd \cdot dInput}{T} \quad (3.2)$$

Onde o erro é calculado por $Setpoint - Input$, o T corresponde ao intervalo de tempo entre cada cálculo do $Output$ e o $dInput$ representa a operação $Input - Input anterior$.

Como pode ser visto, a única diferença do controlador utilizado para o controlador convencional é que o ganho derivativo é aplicado à variação do $input$, para reduzir possíveis efeitos drásticos devido o Kd.

3.4 Testes e Validação

Os procedimentos executados para estabelecer os parâmetros para os controladores PID, tanto do sistema de direção quanto do sistema de aceleração e frenagem, foram inicialmente baseados no método de agendamento de ganhos e de tentativa e erro, de forma a encontrar um compromisso, isto é, valores de parâmetros que satisfizessem a atuação do veículo em diferentes cenários. Entretanto, para um embasamento mais sólido dos parâmetros escolhidos, foi desenvolvido um modelo de caracterização dos motores de corrente contínua de cada sistema, por meio do uso de ferramentas de desenvolvimento e simulação, como o *Matlab* e o *Simulink*.

O uso dessas ferramentas permite a análise simulada do comportamento dos motores diante de diversos tipos de entradas. Para os dois sistemas, o mesmo método foi aplicado, a fim de garantir uniformidade no processo:

1. Código em Arduino aplicando entradas conhecidas de PWM no motor durante um tempo conhecido.
2. Utilização do software *TeraTerm* para obter o registro dos dados de saída, na forma da variável manipulada (ângulo ou velocidade), relacionando-a com o tempo de execução do programa.
3. A fim de obter uma função de transferência correspondente ao veículo real e assim propor ganhos dos controladores em ambiente simulado, foi utilizado, no sistema de direção, o *System Identification* e no sistema aceleração e frenagem, a obtenção da constante de tempo. No sistema de direção, buscou-se a estimativa de uma função de transferência que descreve de maneira mais realista e fiel o sistema analisado e para o sistema de aceleração e frenagem foi assumido que o sistema tratava-se de um sistema de primeira ordem e foi encontrada uma função de transferência por meio da constante de tempo identificada.

4. (Sistema de Direção) Utilização do Simulink para criar um sistema retroalimentado sujeito a uma entrada conhecida, utilizando como planta a função de transferência encontrada anteriormente, e utilizando o bloco de controlador PID para extrair os ganhos. Para o melhor ajuste possível de ganhos, foi utilizada a função *Tune PID*, a qual retornava valores de ganhos ajustados para os requisitos do sistema colocados pelo usuário. No caso deste estudo, buscou-se estabelecer um controlador o mais rápido e robusto possível.
5. (Sistema de Aceleração e Frenagem) Utilização de um script de otimização para definir os ganhos necessários para a resposta se igualar a uma função de transferência referência.
6. Repetição do processo para os cenários de teste analisados. Dessa forma, serão gerados valores de ganhos para todos e, assim, será possível determinar o melhor para todos os casos.

O fluxograma da figura 40 mostra as etapas do método:

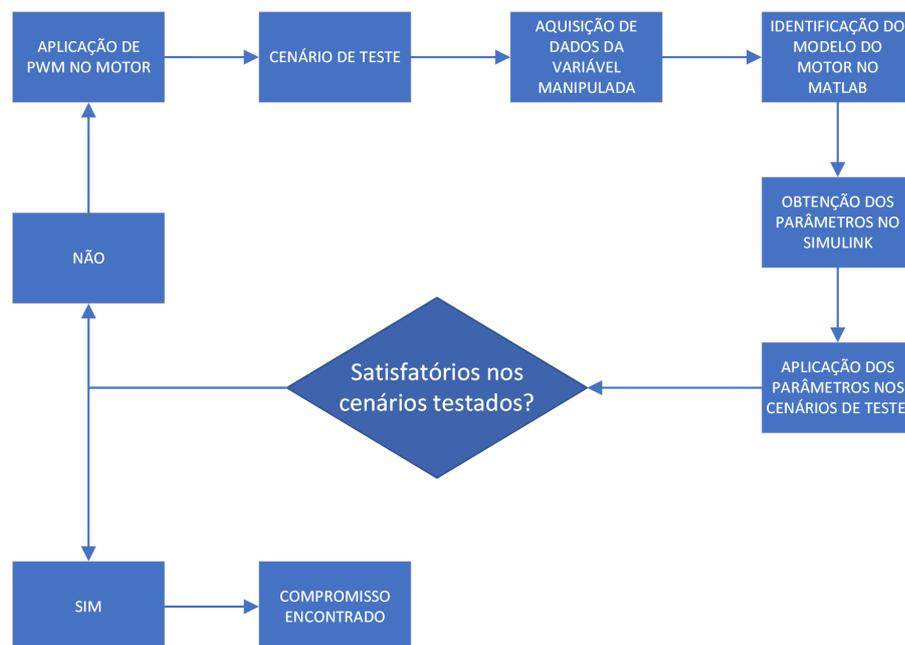


Figura 40 – Fluxograma de Aquisição dos Ganhos

As funções *System Identification* e *Tune PID* são funcionalidades encontradas na interface de desenvolvimento do Matlab/Simulink. A primeira função tem como objetivo permitir a estimação de funções de transferência representativas de um sistema físico a partir da correlação de entradas e saídas. A segunda função tem como principal objetivo

permitir o ajuste dos parâmetros de um controlador PID de acordo com as características desejadas para o sistema: rápido ou lento, robusto ou agressivo.

Embora as etapas acima tenham sido semelhantes para os dois sistemas, cada um possui suas particularidades e, portanto, estarão segmentados abaixo os processos, com maior nível de detalhamento, para cada sistema.

3.4.1 Sistema de Direção

O experimento foi feito com o veículo em um piso de madeira e sem massas consideráveis sobre ele. O primeiro procedimento executado foi correlacionar a aplicação de PWM no motor com o valor do ângulo, como pode-se visualizar na figura 41, na qual PWM está em vermelho e ângulo em azul. Esse procedimento foi embasado no sinal de excitação conhecido como PRBS (*Pseudorandom binary sequence*), que consiste na aplicação de diferentes valores de PWM durante tempos conhecidos. Como a direção de giro pode variar para a direita ou para esquerda, tem-se valores de PWM para os dois sentidos. O objetivo de aplicar essa técnica é o de varrer o máximo de regiões de atuação do motor, de forma a gerar um modelo fiel com a realidade.

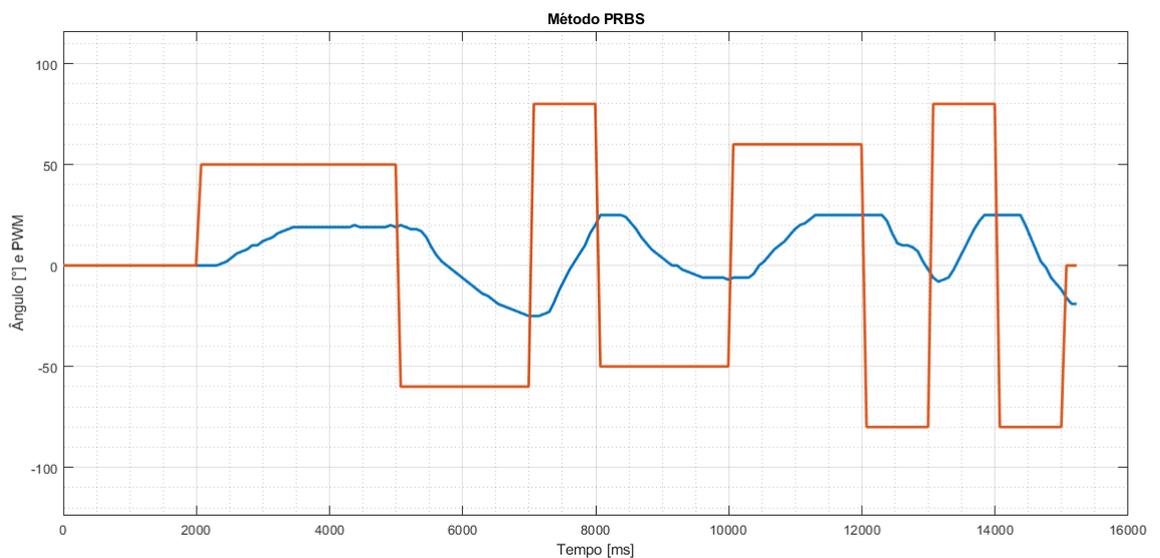


Figura 41 – Gráfico PWM e ângulo x tempo

É importante ressaltar que foram aquisitados os dados acima 5 vezes, e o padrão obtido foi semelhante ao apresentado. Usando a função *systemIdentification* e colocando os dados da figura 41 como referências, o *input* sendo o PWM e o *output* como o ângulo medido, foi possível gerar quatro funções de transferência, cujas características estão apresentadas na figura 42.

Função de Transferência	Número de zeros	Número de polos	Fit da curva
tf_1	1	2	52,38%
tf_2	2	3	63,25%
tf_3	3	3	68,99%
tf_4	3	4	76,55%

Figura 42 – Funções de Transferência

a função de transferência encontrada que melhor se encaixou com a curva de ângulo do motor foi a representada na equação 3.3:

$$H(s) = \frac{-0.02848s^3 + 0.02272s^2 - 0.0002465s + 0.0000092}{s^4 + 0.571s^3 + 0.05198s^2 + 0.0001268s + 0.000011} \quad (3.3)$$

Desta forma, utilizando o *System Identification* não se obteve um *fit* satisfatório. Outros testes podem ser efetuados para tentar garantir uma maior adequação das curvas e, assim, buscar obter melhores ganhos por meio desse método. Portanto, a função de transferência obtida não é representativa do modelo real e desta forma não será utilizada para cálculo dos ganhos do controlador. Deste modo, o método utilizado foi o de tentativa e erro associado ao do *Gain Scheduling*, que serão explicados na seção de Resultados.

3.4.2 Sistema de Aceleração e Frenagem

Para a obtenção da função de transferência de primeira ordem que representa de forma mais fidedigna o sistema em questão, foram necessários dois ensaios, um movimentando o VPE para frente e outro movimentando o VPE para trás. Para ambos os ensaios em malha aberta, a entrada utilizada foi um degrau durante um intervalo de tempo definido, esse degrau, juntamente com a resposta do sistema podem ser vistos nas Figuras 43 e Figura 44. A entrada pode ser considerada bastante simples, mas como os resultados obtidos foram consideravelmente satisfatórios e a utilização mais comum do VPE é feita através da aplicação de entradas degrau, esse procedimento se mostrou eficiente.

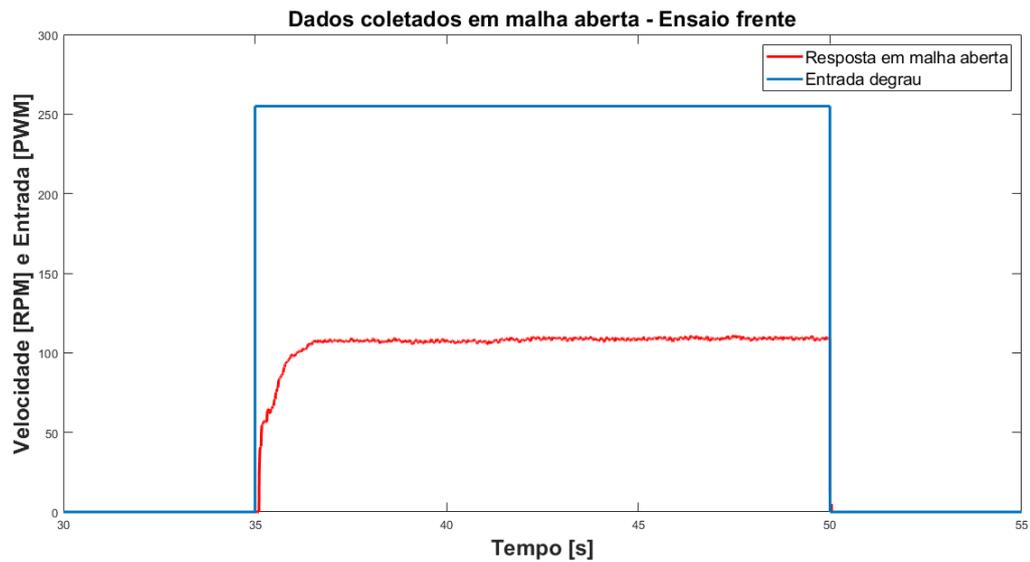


Figura 43 – Dados coletados em malha aberta - Ensaio frente

Como pode ser visto na Figura 43, o sistema tem grande semelhança com um sistema de primeira ordem. Com base na resposta do sistema em malha aberta no ensaio para frente, obteve-se uma constante de tempo $t = 0,499$, valor encontrado quando a velocidade medida é igual a 63,21% da velocidade máxima obtida durante o ensaio. Portanto, a função de transferência obtida tem a forma apresentada na Equação 3.4;

$$G_1 = \frac{0,864}{s + 2004} \quad (3.4)$$

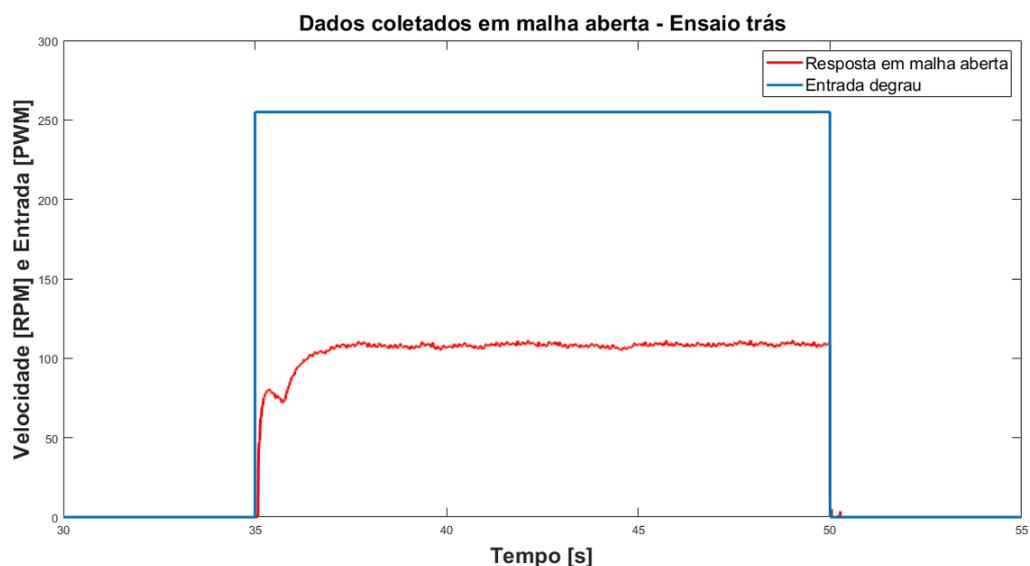


Figura 44 – Dados coletados em malha aberta - Ensaio trás

A constante de tempo obtida para o ensaio trás foi $t = 0,185$, valor consideravelmente diferente em comparação ao ensaio frente, possivelmente devido a uma não line-

aridade no início da subida da resposta. Dessa maneira, a função de transferência obtida pode ser vista na Equação 3.5.

$$G_2 = \frac{2,332}{s + 5.405} \quad (3.5)$$

Foi utilizado o *script* de otimização de ganhos PID em anexo, para obter os ganhos necessários do controlador para que a resposta do sistema seja o mais semelhante possível a uma função de transferência referência pré-definida. A referência consistiu em uma função de transferência de primeira ordem semelhante à presente na Equação 3.6.

$$G_{ref} = \frac{1}{tr \cdot s + 1} \quad (3.6)$$

Sendo tr a constante de tempo requerida.

Os valores testados para tr foram 1 e 2, dessa maneira, foram obtidos diferentes conjuntos de ganhos dependendo da direção do movimento e do valor de tr .

Com $tr = 1$ e movimento para frente, $K_p = 1,16$, $K_i = 2,32$ e $K_d = 2,05e-06$.

Com $tr = 2$ e movimento para frente, $K_p = 0,579$, $K_i = 1,16$ e $K_d = 8,94e-06$.

Com $tr = 1$ e movimento para trás, $K_p = 0,429$, $K_i = 2,32$ e $K_d = 1,27e-06$.

Com $tr = 2$ e movimento para trás, $K_p = 0,214$, $K_i = 1,16$ e $K_d = 4,76e-06$.

4 Resultados

4.1 Sistema de Direção

Algumas razões pelas quais o método PRBS se mostrou inadequado para o sistema de direção são: as limitações do próprio sistema, que apresenta muitas folgas mecânicas e se submete diretamente ao atrito estático presente entre componentes e entre roda e chão, e a aplicação de PWM no motor, que só era possível durante um intervalo de tempo muito curto, pois existe uma limitação mecânica de giro, já que o limite da roda é de 25° para ambos os lados. Esse último fator é o que mais interfere na caracterização correta do motor, porque impossibilita a verificação efetiva de diferentes faixas de PWM, já que a janela de tempo de análise é muito curta. Dessa forma, a correlação obtida entre PWM aplicado e ângulo da roda não corresponde ao modelo real de forma fiel, o que gera um modelo aparentemente satisfatório, mas que quando aplicado no sistema real, se mostra impreciso e inadequado.

Para sanar esse problema, diversos testes foram executados para diferentes valores de parâmetros, por meio do método de tentativa e erro e do método do agendamento de ganhos. Esse método foi adequado para o sistema de direção, pois foi possível criar dois conjuntos de parâmetros PID para diferentes faixas de operação do giro da roda. Isso porque apenas um conjunto de parâmetros não atendia com a mesma qualidade diferentes valores de ângulo, o que tornava o controlador incompleto, em termos de atuação. A análise consistiu em receber sinais na forma de degrau e observar o comportamento do sistema a essa entrada para, assim, ir ajustando o controlador para um desempenho ótimo.

Com a utilização do método do agendamento de ganhos (*Gain Scheduling Method*), foi possível obter o seguinte conjunto de parâmetros, que se mostrou o mais adequado para o sistema de direção: $k_p = 0.01$, $k_i = 0.00001$ e $k_d = 0.00002$ para o intervalo entre 0° e 10° e $k_p = 0.015$, $k_i = 0.00002$ e $k_d = 0.00002$ para o intervalo entre 11° e 25° . Aplicaram-se diferentes entradas no sistema para a validação dos ganhos, e as respostas podem ser observadas nas figuras 45, 46, 47 e 48. As figuras apresentam pontos de 1 a 6, que são coordenadas estratégicas para a análise do controlador desenvolvido, sendo a coordenada (1) o início da resposta, a (2) 10% da resposta final, a (3) 50% da resposta final, a (4) 90% da resposta final, a (5) o tempo de pico e a (6) o tempo levado para atingir o regime permanente. A componente x da coordenada representa o tempo, em segundos, e a componente y representa o valor do ângulo da roda, em graus. Na aquisição de dados, devido a limitações do próprio software, não foi possível começar a amostragem no tempo 0.

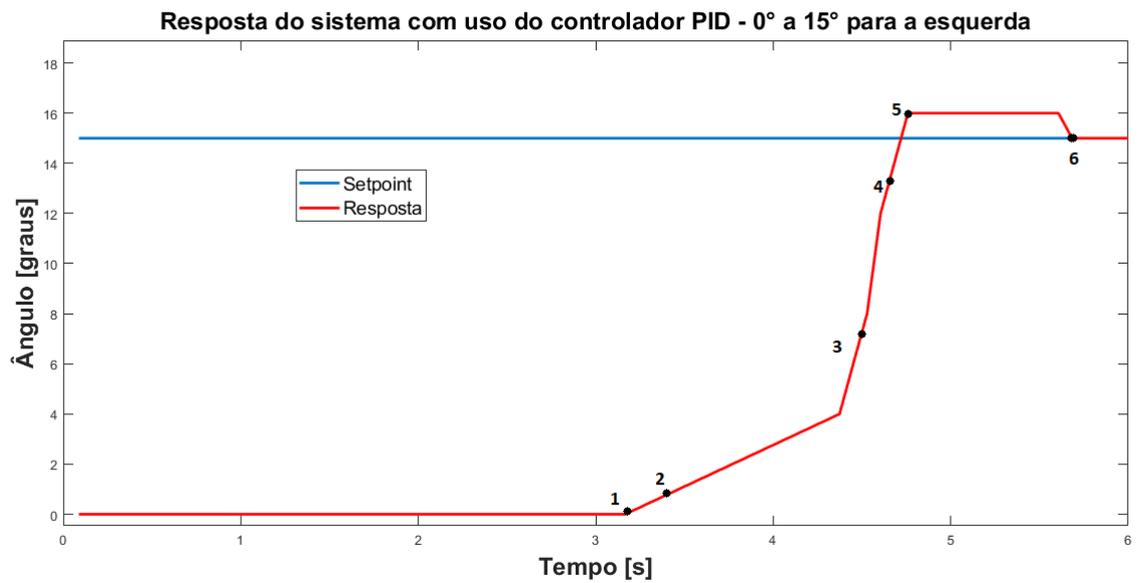


Figura 45 – Resposta do sistema para uma entrada degrau de 15° para a esquerda

As coordenadas desses pontos são, de 1 a 6, (3.1, 0), (3.3, 1.5), (4.5, 7.5), (4.65, 13.5), (4.76, 16) e (5.7, 15). A resposta observada na figura 45 mostra que o tempo total até o sistema atingir o *setpoint* desejado foi de aproximadamente 2.6 segundos, representado pela diferença entre os pontos 6 e 1. O tempo de subida de uma resposta é o tempo que o sistema leva para sair de 10% (1.5°) para 90% (13.5°) da resposta, isto é, do ponto 2 para o ponto 4. Analisando o gráfico, é possível observar que esse tempo foi de cerca de 1.35 segundo. Considerando que o comando foi aplicado em 3.1 segundos, o tempo de atraso, que é o tempo necessário para a resposta atingir 50% de seu valor final (7.5°), foi muito baixo, cerca de 1.4 segundo, dado pela diferença entre os pontos 3 e 1. Outro parâmetro que pode ser identificado na figura é o tempo de pico, que é o tempo decorrido até que se atinja o primeiro pico da resposta. Nesse caso, esse valor ficou próximo de 1.7 segundo. O sobressinal pode ser calculado como a relação entre o máximo valor atingido pela resposta e o valor de *setpoint* desejado, em termos percentuais. Dessa forma, o sobressinal para essa resposta se apresentou em um valor próximo de 7%.

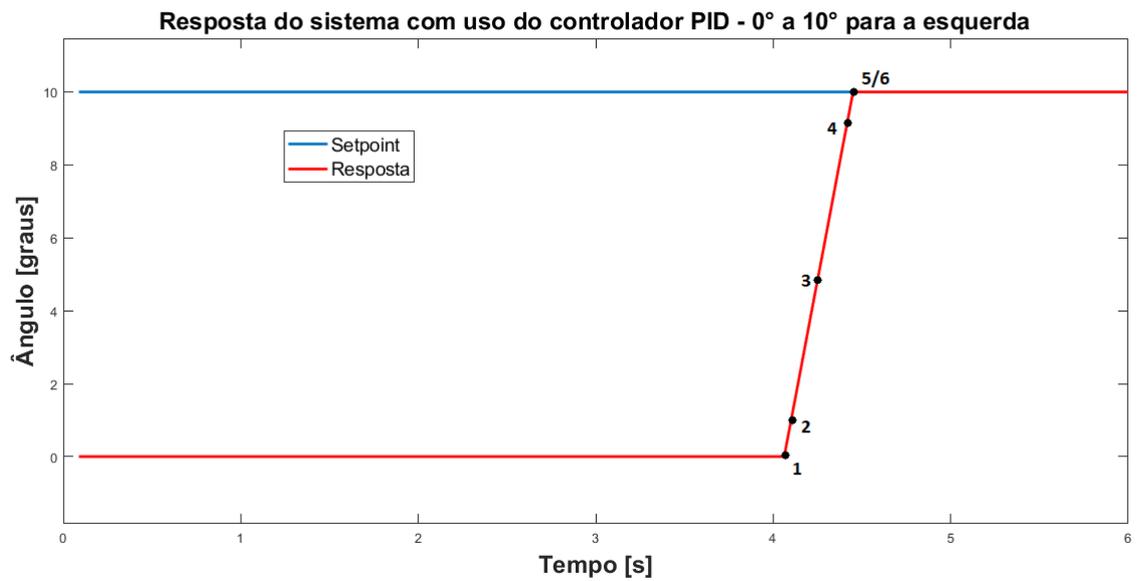


Figura 46 – Resposta do sistema para uma entrada degrau de 10° para a esquerda

As coordenadas desses pontos são, de 1 a 6, (4, 0), (4.1, 1), (4.25, 5), (4.35, 9), (4.4, 10) e (4.4, 10). A resposta observada na figura 46 mostra que o tempo total até o sistema atingir o *setpoint* desejado foi de aproximadamente 0.4 segundos. O tempo de subida, analisando o gráfico, foi cerca de 0.25 segundos. Considerando que o comando foi aplicado em 4 segundos, o tempo de atraso se apresentou muito baixo, cerca de 0.25 segundos. Nesse caso, não houve sobressinal, de forma que a coordenada 5 coincidiu com a coordenada 6.

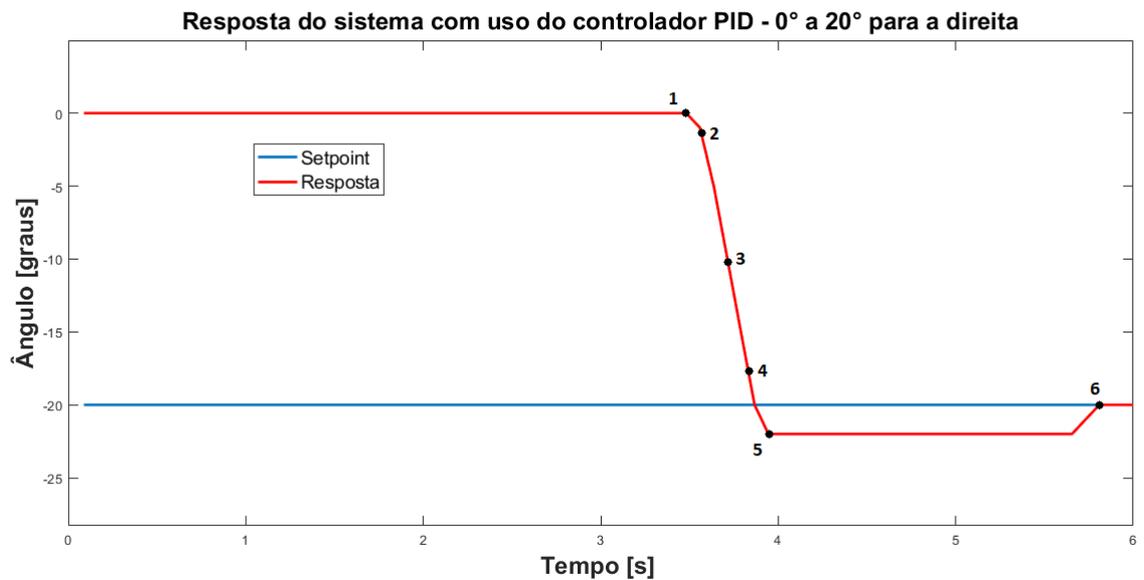


Figura 47 – Resposta do sistema para uma entrada degrau de 0° a 20° para a direita

As coordenadas desses pontos são, de 1 a 6, (3.5, 0), (3.6, -2), (3.7, -10), (3.85, -18), (3.95, -22) e (5.8, -20). A resposta observada na figura 47 mostra que o tempo total até o sistema atingir o *setpoint* desejado foi de aproximadamente 2.3 segundos. O tempo de subida foi de cerca de 0.25 segundos. Considerando que o comando foi aplicado em 3.5 segundos, o tempo de atraso foi muito baixo, cerca de 0.2 segundos. O tempo de pico foi 0.45 segundos. Nesse caso, o ângulo atingido no pico foi 22°, logo, o sobressinal foi de 10%.

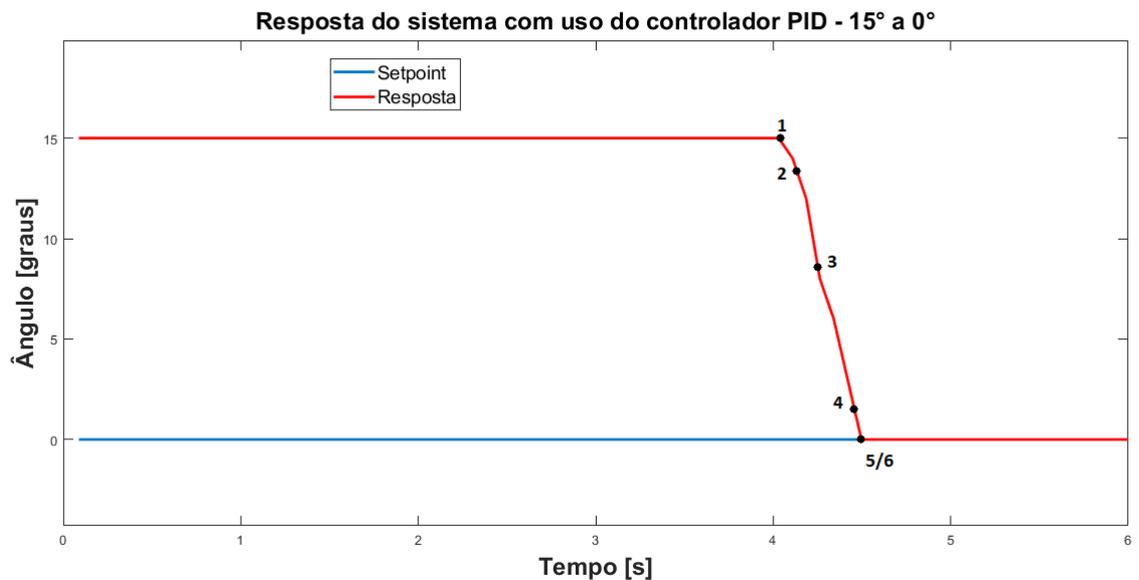


Figura 48 – Resposta do sistema para uma entrada degrau de 15° a 0°

As coordenadas desses pontos são, de 1 a 6, (4, 15), (4.15, 13.5), (4.3, 7.5), (4.45, 1.5), (4.5, 0) e (4.5, 0). A resposta observada na figura 48 mostra que o tempo total até o sistema atingir o *setpoint* desejado foi de aproximadamente 0.5 segundos. O tempo de subida se apresentou próximo de 0.45 segundos. Considerando que o comando foi aplicado em 4 segundos, o tempo de atraso foi muito baixo, cerca de 0.3 segundos. Não houve pico, portanto, a coordenada 5 coincide com a coordenada 6. Nesse caso, como não houve sobressinal na resposta, ele foi nulo.

A expectativa para esse controlador era de uma melhor sintonia entre estabilidade e desempenho, com uma resposta em um tempo razoável para esse tipo de sistema. O objetivo de sua aplicação passa pelos seguintes requisitos: um tempo de assentamento baixo, o que confere ao sistema uma confiabilidade, no sentido de que ele deve se apresentar como ágil em relação às entradas requeridas, um sobressinal muito baixo ou inexistente e um erro em regime permanente nulo.

Cabe ressaltar que antes de chegar aos ganhos utilizados, muitos testes foram executados para diferentes combinações de valores. A metodologia utilizada foi a de começar testando apenas o parâmetro k_p até encontrar um valor razoável para ele. Após isso, o mesmo foi feito para os outros parâmetros, até que culminasse em resultados satisfatórios, como apresentados na figura 45.

Na teoria de controle, o atrito estático se mostra como um grande desafio para a elaboração de sistemas de controle, porque é uma perturbação inerente à maioria dos sistemas. De forma geral, o atrito estático é posto no desenvolvimento das malhas de

controle como uma perturbação de ordem linear. O sobressinal, persistente nos diferentes tipos de controladores testados, é causado também em decorrência da existência desse fator.

A histerese também foi considerada na estratégia de controle, isto é, uma faixa de valores que torna a atuação do sistema aceitável. Embora o *setpoint* desejado fosse um ângulo específico, para que o sistema não entrasse em instabilidade e guardasse suas características positivas, foi adotada uma faixa de aceitação de ângulo de -2° a 2° , isto é, o ângulo poderia variar 2 graus para mais ou para menos e, ainda assim, o controlador PID o interpretaria como correto, retirando a ação de controle no motor.

4.2 Sistema de Aceleração e Frenagem

Para o sistema de aceleração e frenagem o procedimento seguiu o mesmo padrão: foi colocada uma entrada degrau para cada teste feito e, assim, observado o comportamento do sistema. Esses testes podem ser encontrados abaixo.

Com a obtenção dos ganhos para o controlador PID, foram realizados 6 testes para validar o modelo e o controlador, e estão descritos os cenários testados logo a baixo. Os ensaios descritos a seguir foram feitos com a tensão da bateria igual a 12,7 V, informação relevante já que ela interfere diretamente na velocidade final do veículo.

Então, após encontrar, para cada direção, uma função de transferência que possa representar o sistema, foi utilizado um *script* de otimização para obter os ganhos necessários que façam a resposta do sistema controlado ter o comportamento de uma função de transferência referência (Equação 3.6). Foram testados dois valores de constante de tempo para a Gref, em cada direção, para verificar se o sistema real responde de acordo com o esperado, validando o modelo e além disso, pôde-se definir qual o melhor valor de t_r para o sistema.

1. Ensaio para frente, $t_r = 1$;
2. Ensaio para frente, $t_r = 2$
3. Ensaio para trás, $t_r = 1$
4. Ensaio para trás, $t_r = 2$
5. Ensaio para trás, $t_r = 1$ e utilizando os ganhos do ensaio para frente
6. Ensaio para trás, $t_r = 2$ e utilizando os ganhos do ensaio para frente

Para a coleta de dados da resposta do sistema com o controlador PID discreto, também foi utilizada uma entrada degrau durante um intervalo de tempo determinado, Figura 49.

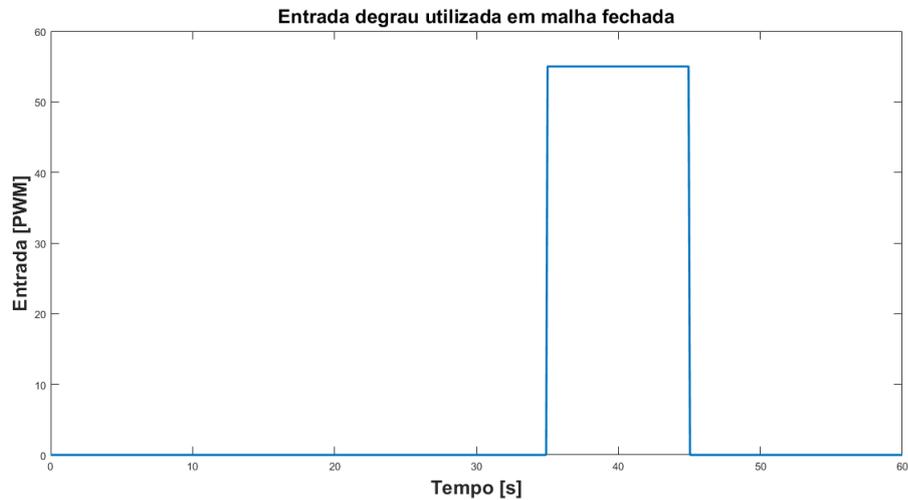


Figura 49 – Entrada degrau utilizada para geração dos resultados

Utilizando o Simulink, foram plotados a função de transferência referência e o resultado coletado da resposta do sistema com o controlador. A estrutura base utilizada no Simulink pode ser vista na Figura 50.

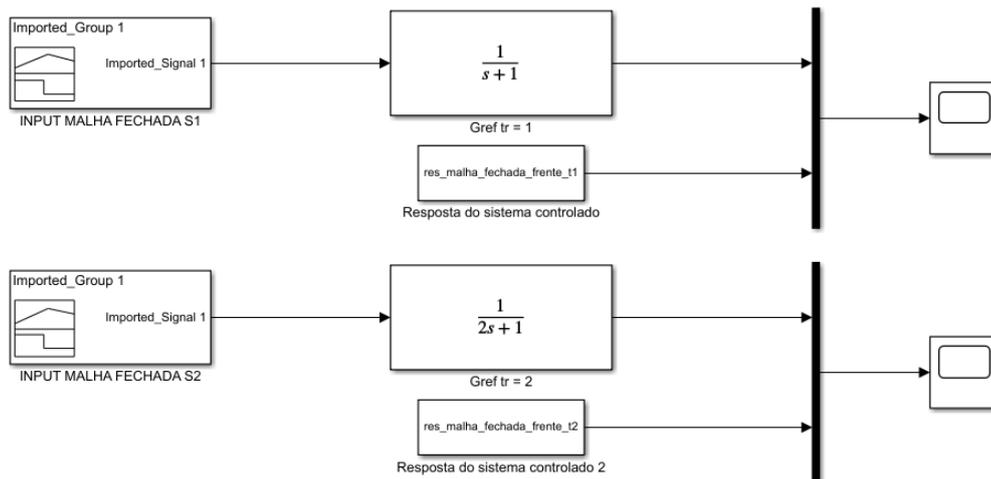


Figura 50 – Estrutura base utilizada para plotagem dos resultados

Os resultados obtidos estão apresentados logo a seguir:

1. Ensaio para frente, $tr = 1$ e tensão da bateria igual a 12,7 V - Figura 51

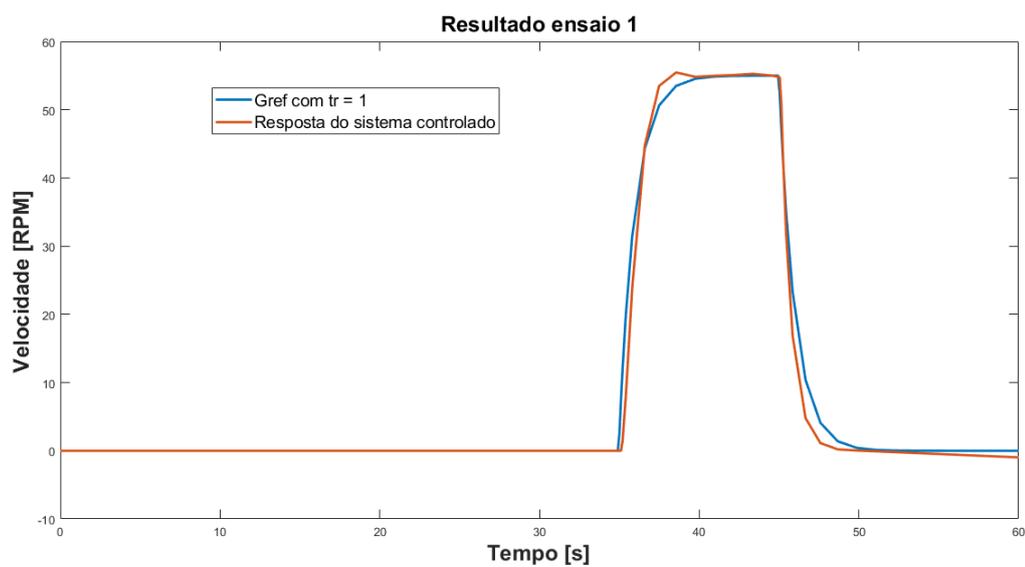


Figura 51 – Resultado primeiro ensaio

A constante de tempo obtida para o resultado do ensaio 1 é $t = 1,115$ segundos.

2. Ensaio para frente, $tr = 2$ e tensão da bateria igual a 12,7 V - Figura 52

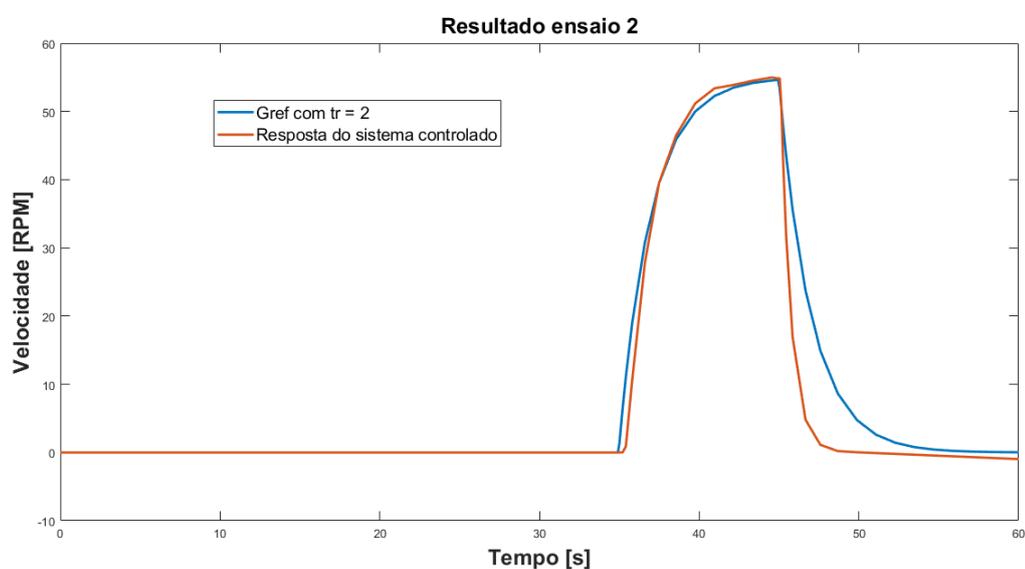


Figura 52 – Resultado segundo ensaio

A constante de tempo obtida para o resultado do ensaio 2 é $t = 2,026$ segundos.

3. Ensaio para trás, $tr = 1$ e tensão da bateria igual a 12,7 V - Figura 53

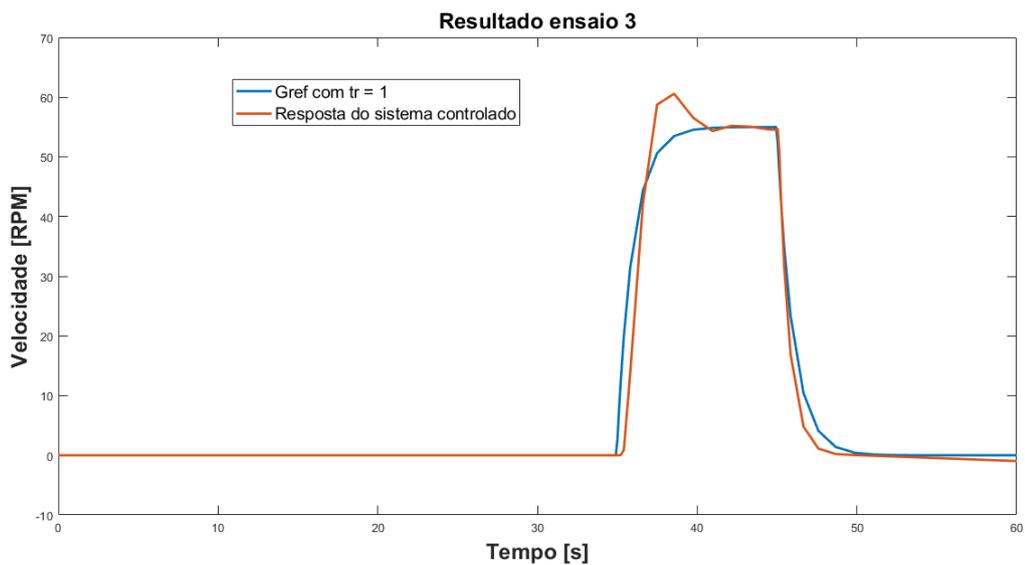


Figura 53 – Resultado terceiro ensaio

A constante de tempo obtida para o resultado do ensaio 3 é $t = 1,359$ segundos.

4. Ensaio para trás, $tr = 2$ e tensão da bateria igual a 12,7 V - Figura 54

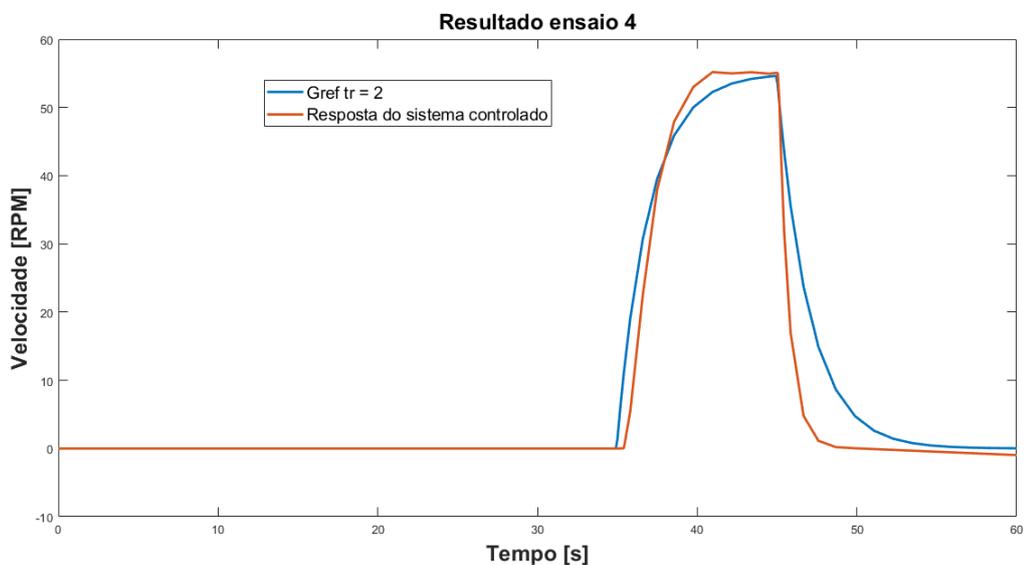


Figura 54 – Resultado quarto ensaio

A constante de tempo obtida para o resultado do ensaio 4 é $t = 2,261$ segundos.

Verificou-se que os resultados obtidos para os ensaios 3 e 4 não foram satisfatórios, pois os valores de t encontrados possuem divergências consideráveis com o esperado, isso pode ser justificado pela constante de tempo encontrada em malha aberta, isso porque ela apresentou diferença significativa quando comparada com o resultado do ensaio frente em malha aberta. Quando é verificada a Figura 44, percebe-se que no início há um pos-

sível problema na medição, devido ao formato inesperado do gráfico, justificado por erros aleatórios e/ou instrumentais.

Diante disso, foram feitos mais dois ensaios com movimento para trás, utilizando os ganhos obtidos para o ensaio frente.

5. Ensaio para trás, $t_r = 1$, tensão da bateria igual a 12,7 V e utilizando os ganhos do ensaio para frente - Figura 55

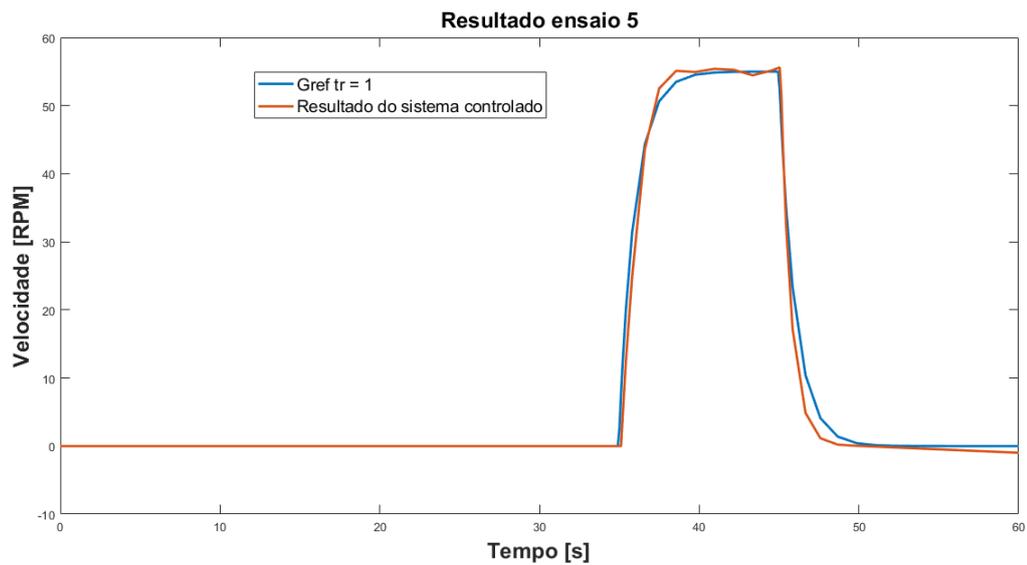


Figura 55 – Resultado quinto ensaio

A constante de tempo obtida para o resultado do ensaio 5 é $t = 1.145$ segundos.

5. Ensaio para trás, $t_r = 2$, tensão da bateria igual a 12,7 V e utilizando os ganhos do ensaio para frente - Figura 56

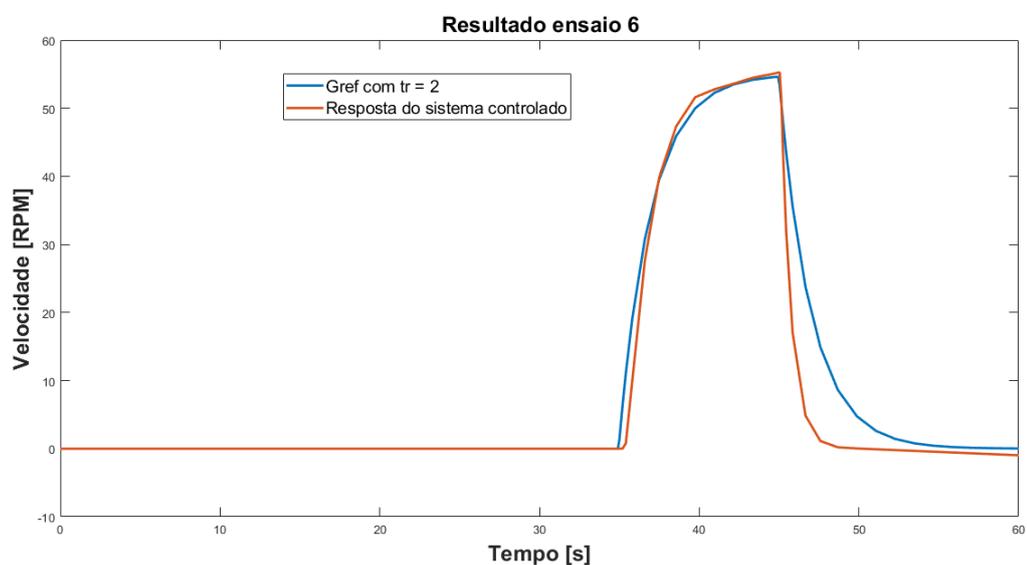


Figura 56 – Resultado sexto ensaio

A constante de tempo obtida para o resultado do ensaio 6 é $t = 2.057$ segundos.

Os ensaios 5 e 6 mostraram-se satisfatórios, portanto, os ganhos obtidos no ensaio trás mostraram-se inválidos e notou-se que os ganhos obtidos para o ensaio frente podem ser utilizados para as duas direções.

A constante de tempo igual a 1 segundo é a mais adequada ao sistema, já que sua resposta é mais rápida e suficiente para o sistema de aceleração e frenagem, portanto, os ganhos selecionados, para ambas as direções, são os ganhos do ensaio para frente, $t_r = 1$.

Outros fatores que provavelmente interferiram nos resultados são: a saturação de alimentação da bateria (Não possui capacidade infinita), limitações do sistema de medição e não linearidades presentes no sistema como um todo.

Então, a metodologia adotada para otimização dos ganhos se mostrou eficiente e o modelo encontrado representa suficientemente o sistema de aceleração e frenagem, mesmo trabalhando com entradas degrau.

5 Conclusão

O desenvolvimento da arquitetura eletroeletrônica para o veículo de pequena escala é de imensa relevância para a consolidação dos estudos direcionados aos veículos autônomos, porque permite, em uma proporção mais simples, a avaliação do comportamento de sistemas de controle em diferentes subsistemas de um veículo. Os objetivos deste trabalho foram atingidos satisfatoriamente, uma vez que os sistemas de controle para a direção e para a aceleração e frenagem foram elaborados baseados em controladores PID, apresentando respostas condizentes com as teorias relacionadas. Além disso, a construção das ECUs foi um meio adequado para embarcar os sistemas desenvolvidos no veículo, o que permitiu que testes fossem executados em campo, em diversas situações. Os resultados obtidos são a comprovação de que os controladores desenvolvidos são eficazes para esta aplicação. Nota-se que, tanto no subsistema de direção quanto no de aceleração e frenagem, os testes executados apresentaram uma evolução da escolha do tipo de controlador.

No sistema de direção, a escolha de um controlador apenas proporcional apresentou alto sobressinal. Com a adição de um termo integral, isto é, um controlador PI, o sobressinal ficou mantido, ou seja, não apresentou melhoria no desempenho de resposta. Com a utilização do controlador PD, o sobressinal foi diminuído, porém ainda persistia em um alto valor, não ideal para um sistema de direção. Já com a utilização do controlador PID, os impasses presentes em outros controladores foram sanados, apresentando um sobressinal bastante baixo e um tempo de assentamento adequado para essa aplicação. Também foi possível concluir que este sistema é inserido em um contexto de consideráveis não linearidades, como o atrito estático, as folgas mecânicas e as limitações do motor CC, o que foram fatores determinantes para o insucesso do método de identificação da planta por meio do Matlab/Simulink.

No sistema de aceleração e frenagem, a consideração de que o sistema trata-se de um sistema de primeira ordem mostrou-se válida. A construção de um controlador PID discreto associado a uma planta representada por uma função de transferência obtida através da constante de tempo identificada, utilizando um *script* de otimização, forneceu resultados que comprovam a validade e, portanto, os resultados obtidos mostram que o sistema funciona de forma de maneira eficiente. E se necessário, a resposta do sistema pode ser alterada de acordo com uma nova referência. Diante disso, a teoria de sistemas de controle aplicada tornou a modelagem e simulação fidedignas ao real comportamento do sistema.

Dessa forma, este trabalho demonstrou que o desenvolvimento de uma arquite-

tura eletroeletrônica em um veículo de pequena escala se apresenta como uma alternativa viável, econômica e menos complexa para a implementação de tecnologia autônoma em veículos reais. Além disso, é um trabalho passível de receber melhorias e aprimoramentos, como a adição de outros sensores, como radares e câmeras, o que garante mais confiabilidade na aquisição de dados do ambiente.

Referências

- AGUIRRE, L. A. *Fundamentos de INSTRUMENTAÇÃO*. São Paulo: PEARSON, 2013. Citado na página 46.
- ARAÚJO, F. M. U. de. Sistemas de controle. 2007. Citado 4 vezes nas páginas 31, 32, 35 e 36.
- BALBINOT, A.; BRUSAMARELLO, V. J. *Instrumentação e Fundamentos de Medidas*. 2. ed. São Paulo: PEARSON, 2010. v. 1. Citado na página 47.
- BARS, R.; BANYASZ, C.; HETTHESSI, J. *Control Engineering*. [S.l.]: Springer, 2019. Citado na página 30.
- BETT, C. J. *The Electrical Engineering Handbook*. [S.l.]: Prentice Hall, 2005. v. 2. Citado na página 37.
- BRUDNA, C. *Desenvolvimento de Sistemas de Automação Industrial Baseados em Objetos Distribuídos e no Barramento CAN*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, Departamento de Engenharia Elétrica, Curso de Pós-Graduação em Engenharia Elétrica, Porto Alegre, 2000. Citado na página 45.
- CIUCIU, C. et al. Autonomous scale model car with ultrasonic sensors and arduino board. 12 2019. Citado na página 21.
- DORF, R. C.; BISHOP, R. H. *Sistemas de controle modernos*. [S.l.]: LTC, 2018. v. 13. Citado 2 vezes nas páginas 31 e 32.
- FIGROUP. *Benefícios fiscais do ROTA 2030*. 2021. <https://www.rota2030.com.br/beneficios-fiscais-rota-2030/>. Citado na página 17.
- FRANKLIN, G. F.; POWELL, J. D.; NAEINI, A. E. *Feedback Control of Dynamic Systems*. [S.l.]: Pearson, 2009. v. 6. Citado na página 33.
- GILLESPIE, T. D. *Fundamentals of Vehicle Dynamics*. [S.l.]: SAE, 1992. Citado 6 vezes nas páginas 23, 24, 26, 27, 28 e 29.
- GROVER, C. et al. Automated emergency braking systems: Technical requirements, costs and benefits. 2008. Citado na página 18.
- GUIMARAES, A. de A. *Eletrônica Embarcada Automotiva*. [S.l.]: Érica, 2007. Citado 2 vezes nas páginas 40 e 42.
- LJUNG, L. *System Identification: Theory for the user*. [S.l.]: Prentice Hall, 1999. v. 2. Citado na página 47.
- MARTINS, H. L. M. Simulação dinâmica de um veículo sob diferentes geometrias de direção. 2010. Citado na página 25.
- OGATA, K. *Engenharia de Controle Moderno*. [S.l.]: Pearson, 2010. v. 5. Citado 6 vezes nas páginas 30, 31, 32, 34, 36 e 128.

- ONSV. *90% DOS ACIDENTES SÃO CAUSADOS POR FALHAS HUMANAS, ALERTA OBSERVATÓRIO*. 2015. <https://www.onsv.org.br/90-dos-acidentes-sao-causados-por-falhas-humanas-alerta-observatorio/>. Citado na página 17.
- PETTERMAN, M. P. Projeto de um veículo autônomo capaz de cobrir uma área poligonal sem passar mais de uma vez pela mesma região. 2015. Citado na página 22.
- PINTO, A. M. de A. *Modelagem no Domínio da Frequência - Parte 2*. 2022. Youtube. Disponível em: Prof. André Murilo. Acesso em Setembro de 2022. Citado na página 29.
- RAJAMANI, R. *Vehicle Dynamics and Control*. [S.l.]: Springer, 2006. v. 1. Citado na página 17.
- ROBILA, V. et al. Design and implementation of pid-based steering control for 1/10-scale autonomous vehicle. In: *2021 IEEE 12th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*. [S.l.: s.n.], 2021. p. 0758–0762. Citado na página 21.
- ROSIER, S. de; RICCOBONI, D.; ROTHHAMMER-RUIZ, P. Small scale intelligent vehicle final design repor. 2018. Citado na página 23.
- SILVA, R. R. da. *Modelagem e Análise de Redes Automotivas em Ambiente Virtual*. 2015. Orientador: Evandro Teixeira. 78 páginas. Trabalho de Conclusão de Curso (Graduação) - Curso de Engenharia Automotiva, Faculdade Gama, Universidade de Brasília, Brasília, Brasil. Citado 5 vezes nas páginas 39, 40, 41, 42 e 43.
- SIMONE, M. D.; GUIDA, D. Identification and control of a unmanned ground vehicle by using arduino. *UPB Scientific Bulletin, Series D: Mechanical Engineering*, v. 80, p. 141–154, 01 2018. Citado na página 22.
- SOUSA, R. V. de. *CAN (Controller Area Network): uma abordagem para automação e controle na área agrícola*. Dissertação (Mestrado) — Universidade de São Paulo, Departamento de Engenharia Mecânica, Curso de Pós-Graduação em Engenharia Mecânica, São Carlos, 2002. Citado 3 vezes nas páginas 40, 41 e 44.
- SREEVATSAN, S. et al. Data logger for a miniature model car using can bus. *Pakistan Journal of Biotechnology*, v. 13, n. special issue 1, p. 52–54, Mar. 2021. Disponível em: <<https://pjbt.org/index.php/pjbt/article/view/212>>. Citado na página 22.
- TAEYOUNG, L. et al. Development and evaluations of advanced emergency braking system algorithm for the commercial vehicle. 2011. Citado na página 18.
- YE, X. Adaptive lane keeping assistance system design based on driver's behavior. Politecnico di Torino, 2019. Citado na página 18.

Apêndices

APÊNDICE A – Códigos de controle dos sistemas de direção e de aceleração e frenagem

1. Sistema de direção

```

1 // DEFINICAO DE BIBLIOTECAS
2
3 #include <mcp_can.h>
4 #include <SPI.h>
5 #include "calculaangulo.h"
6
7 // DECLARACAO DE VARIAVEIS
8
9 #define CANO_INT 2 // Pino INT em 2
10 MCP_CAN CANO(8); // Pino CS em 8
11
12 // DECLARACAO DAS VARIAVEIS DE CONTROLE:
13
14 int leitura = 25; // Valor de comando inicial recebido na
    rede CAN
15 int anguloroda = 0;
16 float erro = 0;
17 float erroAnterior = 0;
18 int setpoint = 0;
19
20 unsigned long loopTimerEnd = 5, loopTimerStart = 0, loopTime
    = 1;
21 float ac_e = 0; // Acumulador de erro para gerar o efeito
    integrativo
22
23 // FUNCAO DE VERIFICACAO DA CAN
24
25 void setupCAN()
26 {
27     // Inicializa o MCP2515 rodando a 16MHz com uma taxa de 500
    kb/s e the mascaras e filtros desabilitados.
28     if (CANO.begin(MCP_ANY, CAN_500KBPS, MCP_16MHZ) == CAN_OK)

```

```

29  {
30      Serial.println("MCP2515 Initialized Successfully!");
31  }
32  else
33  {
34      Serial.println("Error Initializing MCP2515...");
35  }
36
37  // Determina o modo de operacao para normal para que o
      MCP2515 envie acks para o dado recebido.
38  CAN0.setMode(MCP_NORMAL);
39
40  // Configurando o pino para /INT input
41  pinMode(CAN0_INT, INPUT);
42
43  Serial.println("MCP2515 Library Receive Example...");
44  }
45
46 void setup()
47 {
48     // Determinacao dos pinos usados para aplicar o PWM
49     pinMode(5, OUTPUT);
50     pinMode(6, OUTPUT);
51
52     // Taxa de atualizacao
53     Serial.begin(115200);
54
55     // Chamada da funcao de inicializacao da CAN
56     setupCAN();
57 }
58
59 // FUNCAO DE IDENTIFICACAO DA MENSAGEM
60
61 void handleCAN()
62 {
63     long unsigned int rxId;
64     unsigned char len = 0;
65     unsigned char rxBuf[8];
66     char msgString[128]; // Vetor para armazenar serial string
67
68     CAN0.readMsgBuf(&rxId, &len, rxBuf); // Le dados: len =
      tamanho do dado, buf = data byte(s)

```

```

69
70 // Mensagem de validacao
71 if ((rxId & 0x80000000) == 0x80000000)
72 {
73     // Determina se o ID e padrao (11 bits) ou estendido (29
74     // bits)
75     sprintf(msgString, "Extended ID: 0x%.8lX  DLC: %1d  Data:
76     ", (rxId & 0x1FFFFFFF), len);
77 }
78 else
79 {
80     sprintf(msgString, "Standard ID: 0x%.3lX  DLC: %1d  Data:
81     ", rxId, len);
82 }
83
84 if ((rxId & 0x40000000) == 0x40000000)
85 {
86     // Determina se a mensagem e um remote request frame.
87     sprintf(msgString, " REMOTE REQUEST FRAME");
88 }
89 else
90 {
91     for (byte i = 0; i < len; i++)
92     {
93         sprintf(msgString, " 0x%.2X", rxBuf[i]);
94     }
95 }
96
97 // Byte 6 traz a informacao de comando do angulo desejado
98 if (rxId == 0x100)
99 {
100     leitura = rxBuf[6]; // Valor recebido no byte 6 (escala 0
101     // a 50)
102 }
103 }
104
105 // FUNCAO QUE CALCULA O PWM
106
107 int calculatePWM(float erro, float erroAnterior, unsigned
108     long deltaT, float KP, float KI, float KD)
109 {

```

```
106 // Calculo do acumulo de erros
107 ac_e = ac_e + (erro * deltaT);
108
109 // Limitacao da acumulacao de erros ao valor maximo de 1/KI
    para que a acao de controle nao cresca indefinidamente
110 if (KI * ac_e > 1 && KI != 0)
111 {
112     ac_e = 1 / KI;
113 }
114 else if (-1 <= erro && erro <= 1)
115 {
116     ac_e = 0;
117 }
118
119 // Calculo da acao de controle
120 float u_proporcional = KP * erro;
121 float u_derivativo = (KD / deltaT) * (erro - erroAnterior);
122 float u_integral = KI * ac_e;
123
124 float u = u_proporcional + u_derivativo + u_integral;
125
126 Serial.print(" u: ");
127 Serial.print(u);
128
129 Serial.print(" = P:");
130 Serial.print(u_proporcional);
131
132 Serial.print(" + I:");
133 Serial.print(u_integral);
134
135 Serial.print(" + D: ");
136 Serial.print(u_derivativo);
137
138
139 // PWM necessario a partir da acao de controle adequada
140 int PWM = round(255 * abs(u));
141
142 if (PWM > 255)
143 {
144     PWM = 255;
145 }
146
```

```
147     return PWM;
148 }
149
150 // FUNCAO QUE APLICA O PWM NOS PINOS
151
152 void applyPWM(float erro, int PWM)
153 {
154
155     // Define condicoes de histerese para o sistema de controle
156     if (erro >= 2)
157     {
158         analogWrite(5, PWM);
159         analogWrite(6, 0);
160     }
161     else if (erro <= -2)
162     {
163         analogWrite(5, 0);
164         analogWrite(6, PWM);
165     }
166     else
167     {
168         analogWrite(5, 0);
169         analogWrite(6, 0);
170     }
171 }
172
173 // FUNCAO DE CONTROLE
174
175 void controlFunction()
176 {
177
178     // Finaliza o timer
179     loopTimerEnd = millis();
180
181     // Atribui valores iniciais para o erro e seus dois
182     // antecedentes, que serao sempre atualizados
183     erroAnterior = erro;
184     erro = setpoint - anguloroda;
185
186     Serial.print(" ERRO: ");
187     Serial.print(erro);
```

```

188 // Agendamento de ganhos (Gain Scheduling Method)
189 // Define quais valores usar de acordo com a situação
190 // Histórico de valores:
191 // CASO 1: KP = 0.1; KI = 0.1; KD = 0.1
192 // CASO 2: KP = 0.05; KI = 0.00000000001; KD =
    0.00000000001
193 // CASO 3: KP = 0.04; KI = 0.00002; KD = 0.00000000001
194 // CASO 4: KP = 0.3; KI = 0.00000000001; KD = 0.00002
195 // CASO 5: KP = 0.03; KI = 0.00002; KD = 0.00002
196
197 float KP, KI, KD;
198
199 if (abs(erro) <= 10)
200 {
201     KP = 0.01;
202     KI = 0.00002;
203     KD = 0.75;
204 }
205 else if (abs(erro) > 10)
206 {
207     KP = 0.025;
208     KI = 0.00002;
209     KD = 0.00002;
210 }
211
212 int PWM = calculatePWM(erro, erroAnterior, loopTime, KP, KI
    , KD);
213
214 Serial.print(" PWM: ");
215 Serial.print(PWM);
216
217 applyPWM(erro, PWM);
218
219 // Calcula o tempo decorrido até esse ponto
220 loopTime = abs(loopTimerEnd - loopTimerStart);
221 // Inicia o timer
222 loopTimerStart = millis();
223
224 // Tempo de amostragem
225 Serial.print(" t0: ");
226 Serial.print(loopTime);
227

```

```

228 }
229
230 void loop()
231 {
232     if (digitalRead(CANO_INT) == LOW) // Se o pino CANO_INT
        esta em LOW, leia o buffer recebido
233     {
234         handleCAN(); // Chama a funcao
235     }
236
237     setpoint = leitura - 25;          // Conversao da leitura para
        a escala -25 a 25
238     anguloroda = calculaangulo(); // Calcula o valor do angulo
        da roda
239
240     Serial.print(" ang: ");
241     Serial.print(anguloroda);
242     Serial.print(" set: ");
243     Serial.print(setpoint);
244
245     controlFunction(); // Chama a funcao de controle
246
247     Serial.println();
248 }
249
250 // FUNCAO DE CALCULAR O ANGULO
251
252 #include <Arduino.h>
253
254 int calculaangulo()
255 {
256
257     // CALCULA A MEDIA DE 500 VALORES LIDOS:
258     int anguloatual = 0;
259     for (int i = 0; i < 500; i++)
260     {
261         int valor = analogRead(A0);
262         int angulocoluna = map(valor, 100, 300, -40, 40);
263
264         anguloatual += angulocoluna;
265     }
266

```

```

267     anguloatual = anguloatual / 500;
268
269     // CONVERSAO DO VALOR DA COLUNA PARA O DAS RODAS:
270     int anguloroda = 0.676 * anguloatual;
271
272     // LIMITACAO DA LEITURA DE ANGULO MAX
273     if (anguloroda > 25)
274     {
275         anguloroda = 25;
276     }
277     else if (anguloroda < -25)
278     {
279         anguloroda = -25;
280     }
281
282     return anguloroda;
283 }

```

2. Sistema de Aceleração e Frenagem:

```

1
2     // Bibliotecas para comunica o CAN
3 #include <SPI.h>
4 #include <mcp_can.h>
5
6 // Biblioteca para Interrupt
7 #include <util/atomic.h>
8
9 // Vari veis CAN
10 long unsigned int rxId;
11 unsigned char len = 0;
12 unsigned char rxBuf[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
13     0x00, 0x73}; // Byte[7] = 6E => 110 RPM
14
15 // Vari veis do controle PID
16 double previousPWM;
17 double referenceVelocity = 0;
18 int controlSignal = 0;
19
20 // Vari veis para filtragem da velocidade medida
21 double velocityRPMFiltered = 0;
22 double velocityRPMPrevious = 0;

```

```

23
24 double velocityRPMFiltered2 = 0;
25 double velocityRPMPrevious2 = 0;
26
27 // Variáveis para controlador PID discreto
28 double PWM_Setpoint, PWM_Input, PWM_Output;
29 double lastInput, outputSum;
30 unsigned long tempo_anterior;
31 double Kp=1.16, Ki=2.32, Kd=0.00000205; //Melhores ganhos
    encontrados
32
33 // Variáveis voláteis utilizadas no cálculo da velocidade
34 volatile double velocity_i = 0;
35 volatile long prevT_i = 0;
36 volatile double velocity_i_2 = 0;
37 volatile long prevT_i_2 = 0;
38
39 // Variável para controle do tempo de recebimento de
    mensagens CAN
40 unsigned long lastMessageReceivedTime = 0;
41
42 void readEncoder() {
43     // Esta função captura o intervalo de tempo entre os
    pulsos do sensor de velocidade
44     // e transforma o tempo em velocidade a partir de pulso
    por segundos
45
46     if (previousPWM == 0)
47         return;
48
49     long currT = micros();
50     double deltaT = ((double)(currT - prevT_i)) / 1.0e6;
51
52     // Caso o tempo entre medições seja muito pequeno, o
    valor da velocidade 1
53     // tende a infinito, o que não é desejável, por isso
    as medições
54     // que ocorram em menos de 0.015 segundos
55
56     if (deltaT >= 0.015) {
57         velocity_i = 1 / deltaT;
58

```

```

59         // Salva o momento da ultima medição
60         prevT_i = currT;
61     }
62 }
63
64 void readEncoder2() {
65     // Esta função captura o intervalo de tempo entre os
66     // pulsos do sensor de velocidade 2
67     // e transforma o tempo em velocidade a partir de pulso
68     // por segundos
69
70     if (previousPWM == 0)
71         return;
72
73     long currT2 = micros();
74     double deltaT2 = ((double)(currT2 - prevT_i_2)) / 1.0e6;
75
76     // Caso o tempo entre medições seja muito pequeno, o
77     // valor da velocidade
78     // tende a infinito, o que não é desejável, por isso
79     // as medições são ignoradas
80     // que ocorram em menos de 0.015 segundos
81
82     if (deltaT2 >= 0.015) {
83         velocity_i_2 = 1 / deltaT2;
84
85         // Salva o momento da ultima medição
86         prevT_i_2 = currT2;
87     }
88 }
89
90 void setMotor(int dir, int pwmVal) {
91     int oldPMW = pwmVal;
92
93     if (dir == 1) {
94         // Gira num sentido
95         analogWrite(L_PWM_2, 0);
96         analogWrite(R_PWM_2, pwmVal);
97         analogWrite(L_PWM_1, 0);
98         analogWrite(R_PWM_1, pwmVal);
99         previousPWM = oldPMW;
100     } else if (dir == 2) {

```

```

97         // Gira em outro sentido
98         analogWrite(R_PWM_2, 0);
99         analogWrite(L_PWM_2, pwmVal);
100        analogWrite(R_PWM_1, 0);
101        analogWrite(L_PWM_1, pwmVal);
102        previousPWM = -oldPMW;
103    } else if (dir == 0) {
104        // Desliga os motores
105        analogWrite(R_PWM_2, 0);
106        analogWrite(L_PWM_2, 0);
107        analogWrite(R_PWM_1, 0);
108        analogWrite(L_PWM_1, 0);
109
110        previousPWM = 0;
111        velocity_i = 0;
112        velocity_i_2 = 0;
113        PWM_Input = 0;
114    }
115
116    delay(100);
117 }
118
119 void setupPins() {
120     // Pinos
121     pinMode(CAN_ST, OUTPUT); // Pino de Status CAN
122     pinMode(D_Sensor, INPUT); // Pino Sensor 1
123     pinMode(D_Sensor_2, INPUT); // Pino Sensor 2
124
125     // Pinos PWM
126     pinMode(R_PWM_1, OUTPUT);
127     pinMode(L_PWM_1, OUTPUT);
128     pinMode(R_PWM_2, OUTPUT);
129     pinMode(L_PWM_2, OUTPUT);
130 }
131
132 void setupCAN() {
133     // Inicializando m dulo CAN - A 8 MHz com taxa de
134     // transmiss o de 500 kb/s - Filtros e m scaras
135     // desabilitados
136     if (CAN0.begin(MCP_ANY, CAN_500KBPS, MCP_8MHZ) == CAN_OK)
137         Serial.println("MCP2515 Initialized Successfully!");
138     else

```

```

137     Serial.println("Error Initializing MCP2515...");
138
139     CAN0.setMode(MCP_NORMAL); // Configurando o modo de
        opera o como normal, ent o o MCP2515 envia acks
        para receber dados.
140 }
141
142 void setup() {
143
144     Serial.begin(115200);
145
146     setupPins();
147     setupCAN();
148
149     // Habilitando as interrup es
150     attachInterrupt(digitalPinToInterrupt(D_Sensor),
        readEncoder, RISING);
151     attachInterrupt(digitalPinToInterrupt(D_Sensor_2),
        readEncoder2, RISING);
152 }
153
154 void handleCAN() {
155     CAN0.readMsgBuf(&rxId, &len, rxBuf); // Lendo dados: len
        = comprimento do campo de dados, buf = byte[s] de
        dados
156
157     if ((rxId & 0x80000000) == 0x80000000) // Determina se o
        frame standard (11 bits) ou extended (29 bits)
158     {
159         sprintf(msgString, "Extended ID: 0x%.8lX DLC: %1d
            Data:", (rxId & 0x1FFFFFFF), len);
160     } else {
161         sprintf(msgString, "Standard ID: 0x%.3lX DLC:
            %1d Data:", rxId, len);
162     }
163     Serial.print(msgString);
164
165     if ((rxId & 0x40000000) == 0x40000000) { // Determina se
        a mensagem um remote request frame.
166         sprintf(msgString, " REMOTE REQUEST FRAME");
167         Serial.print(msgString);
168     } else {

```

```

169     for (byte i = 0; i < len; i++) {
170         sprintf(msgString, " 0x%.2X", rxBuf[i]);
171         Serial.print(msgString);
172     }
173 }
174
175 Serial.println();
176
177 //Recebendo a mensagem de interesse
178 if (rxId == 0x100) {
179     referenceVelocity = rxBuf[7] - 110; // Aplicando o
180     // offset para a velocidade referencia
181     lastMessageReceivedTime = millis();
182 }
183
184 void controlFunction() {
185
186     // A variavel ser lida no bloco atomico pois o valor
187     // alterado num interrupt
188     double velocityCountPerSecond = 0;
189     double velocityCountPerSecond2 = 0;
190
191     ATOMIC_BLOCK(ATOMIC_RESTORESTATE) {
192         velocityCountPerSecond = velocity_i;
193         velocityCountPerSecond2 = velocity_i_2;
194     }
195
196     // Converte count/s para RPM
197     double velocityRPM = velocityCountPerSecond / 20.0 *
198         60.0;
199     double velocityRPM2 = velocityCountPerSecond2 / 20.0 *
200         60.0;
201
202     // Filtrando
203     velocityRPMFiltered = 0.854 * velocityRPMFiltered +
204         0.0728 * velocityRPM + 0.0728 * velocityRPMPrevious;
205     velocityRPMPrevious = velocityRPM;
206
207     velocityRPMFiltered2 = 0.854 * velocityRPMFiltered2 +
208         0.0728 * velocityRPM2 + 0.0728 * velocityRPMPrevious2;
209     velocityRPMPrevious2 = velocityRPM2;

```

```

206
207     if (referenceVelocity == 0) {
208         controlSignal = 0;
209     } else if (referenceVelocity > 0) {
210         controlSignal = 1;
211     } else {
212         controlSignal = 2;
213     }
214
215     ////////// C digo PID discreto
216     //////////////////////////////////////
217     PWM_Setpoint = abs(referenceVelocity);
218     PWM_Input = (velocityRPMFiltered + velocityRPMFiltered2)
219                /2; // M dia dos valores de velocidade medidos
220
221     unsigned long tempo_atual = millis();
222     unsigned long mud_tempo = (tempoatual - lastTime);
223
224     if(mud_tempo >= 100) {
225
226         double erro = PWM_Setpoint - PWM_Input;
227         double dInput = PWM_Input - lastInput;
228
229         outputSum += (Ki*mud_tempo*erro);
230
231         if(outputSum > 255) {
232             outputSum = 255;
233         }else if(outputSum < 0) {
234             outputSum = 0;
235         }
236
237         PWM_Output = Kp*erro;
238
239         PWM_Output += outputSum - ((Kd*dInput)/mud_tempo);
240
241         if(PWM_Output > 255) {
242             PWM_Output = 255;
243         }else if(PWM_Output < 0) {
244             PWM_Output = 0;
245         }
246
247         lastInput = PWM_Input;

```

```

246     tempo_anterior = tempo_atual;
247
248     }
249
250     //
251     ////////////////////////////////////////////////////////////////////
252     // Limitando sa da do controlador
253     if (PWM_Output > 255) {
254         PWM_Output = 255;
255     } else if (PWM_Output < 0) {
256         PWM_Output = 0;
257     }
258     setMotor(controlSignal, PWM_Output);
259 }
260 void loop() {
261     if (CAN0.checkReceive() == CAN_MSGAVAIL) {
262         handleCAN();
263     }
264
265     controlFunction();
266
267     // Verifica o do tempo
268     unsigned long diftempo = millis() -
269         lastMessageReceivedTime;
270
271     // Verifica o de recebimento de mensagens CAN
272     if (diftempo > 1000) {
273         digitalWrite(CAN_ST, HIGH);
274     } else {
275         digitalWrite(CAN_ST, LOW);
276     }
277 }

```

```

1
2     // Arquivo de definicao de pinos ABM
3
4     #define CAN_ST 7 // Pino Status CAN
5
6     // Pinos de sinal dos Sensores Encoder Velocidade
7     #define D_Sensor 2

```

```
8 #define D_Sensor_2 3
9
10 // Pinos portas PWM
11 #define R_PWM_1 5
12 #define L_PWM_1 6
13 #define R_PWM_2 9
14 #define L_PWM_2 10
15
16 // Pino CS CAN
17 #define CAN_CS 8
```

3. Códigos em Matlab

```
%% Código utilizado para a plotagem dos dados coletados

clear all; close all; clc;

% Carregando dados em malha aberta - Para frente - Tensão = 12.7 V
load Ensaio_malha_aberta_frente_12_7.txt
% Carregando dados em malha aberta - Para trás - Tensão = 12.7 V
load Ensaio_malha_aberta_tras_12_7.txt

% Atribuindo os dados à variáveis para a plotagem do ensaio frente
time1=Ensaio_malha_aberta_frente_12_7(:,1);
measuredVel1=Ensaio_malha_aberta_frente_12_7(:,2);
pwmValue1=Ensaio_malha_aberta_frente_12_7(:,3);

% Plotagem dos dados ensaio frente
figure
plot(time1,measuredVel1, time1, pwmValue1); legend('Resposta em malha aberta','Entrada de grau');
title('Dados coletados em malha aberta - Ensaio frente');
xlabel('Tempo [s]');
ylabel('Velocidade [RPM] e Entrada [PWM]');
grid minor;

%% Atribuindo os dados à variáveis para a plotagem do ensaio trás
time2=Ensaio_malha_aberta_tras_12_7(:,1);
measuredVel2=Ensaio_malha_aberta_tras_12_7(:,2);
pwmValue2=Ensaio_malha_aberta_tras_12_7(:,3);

% Plotagem dos dados ensaio trás
figure
plot(time2,measuredVel2, time2, pwmValue2); legend('Resposta em malha aberta','Entrada de grau');
title('Dados coletados em malha aberta - Ensaio trás');
xlabel('Tempo [s]');
ylabel('Velocidade [RPM] e Entrada [PWM]');
grid minor;

%% Resultados do controlador PID
load Resultado_Malha_Fechada_Frente_t1.txt
load Resultado_Malha_Fechada_Frente_t2.txt
load Resultado_Malha_Fechada_Tras_t1.txt
load Resultado_Malha_Fechada_Tras_t2.txt
load Resultado_Malha_Fechada_Tras_t1_Parametros_frente.txt
load Resultado_Malha_Fechada_Tras_t2_Parametros_frente.txt

time3=Resultado_Malha_Fechada_Frente_t1(:,1);
measuredVel3=Resultado_Malha_Fechada_Frente_t1(:,2);
velreferencia_ensaiol=Resultado_Malha_Fechada_Frente_t1(:,3);

res_malha_fechada_frente_t1 = [time3 measuredVel3];

time4=Resultado_Malha_Fechada_Frente_t2(:,1);
measuredVel4=Resultado_Malha_Fechada_Frente_t2(:,2);
velreferencia_ensaiol2=Resultado_Malha_Fechada_Frente_t2(:,3);
```

```
res_malha_fechada_frente_t2 = [time4 measuredVel4];

time5=Resultado_Malha_Fechada_Tras_t1(:,1);
measuredVel5=Resultado_Malha_Fechada_Tras_t1(:,2);
velreferencia_ensaio3=Resultado_Malha_Fechada_Tras_t1(:,3);

res_malha_fechada_tras_t1 = [time5 measuredVel5];

time6=Resultado_Malha_Fechada_Tras_t2(:,1);
measuredVel6=Resultado_Malha_Fechada_Tras_t2(:,2);
velreferencia_ensaio4=Resultado_Malha_Fechada_Tras_t2(:,3);

res_malha_fechada_tras_t2 = [time6 measuredVel6];

time7=Resultado_Malha_Fechada_Tras_t1_Parametros_frente(:,1);
measuredVel7=Resultado_Malha_Fechada_Tras_t1_Parametros_frente(:,2);
velreferencia_ensaio5=Resultado_Malha_Fechada_Tras_t1_Parametros_frente(:,3);

res_malha_fechada_tras_t1_parametros_frente = [time7 measuredVel7];

time8=Resultado_Malha_Fechada_Tras_t2_Parametros_frente(:,1);
measuredVel8=Resultado_Malha_Fechada_Tras_t2_Parametros_frente(:,2);
velreferencia_ensaio6=Resultado_Malha_Fechada_Tras_t2_Parametros_frente(:,3);

res_malha_fechada_tras_t2_parametros_frente = [time8 measuredVel8];

%% Plotagem dos Resultados
figure
plot(time3,measuredVel3); hold on
plot(time3,velreferencia_ensaio1); hold off
title('Resultados Malha Fechada - Ensaio frente t1');
xlabel('Tempo [s]');
ylabel('Velocidade [RPM]');
grid minor;

figure
plot(time4,measuredVel4); hold on
plot(time4,velreferencia_ensaio2); hold off
title('Resultados Malha Fechada - Ensaio frente t2');
xlabel('Tempo [s]');
ylabel('Velocidade [RPM]');
grid minor;

figure
plot(time5,measuredVel5); hold on
plot(time5,velreferencia_ensaio3); hold off
title('Resultados Malha Fechada - Ensaio tras t1');
xlabel('Tempo [s]');
ylabel('Velocidade [RPM]');
grid minor;

figure
plot(time6,measuredVel6); hold on
plot(time6,velreferencia_ensaio4); hold off
title('Resultados Malha Fechada - Ensaio tras t2');
```

```
xlabel('Tempo [s]');  
ylabel('Velocidade [RPM]');  
grid minor;
```

```
figure  
plot(time7,measuredVel7); hold on  
plot(time7,velreferencia_ensaio5); hold off  
title('Resultados Malha Fechada - Ensaio tras t1 - ganhos frente');  
xlabel('Tempo [s]');  
ylabel('Velocidade [RPM]');  
grid minor;
```

```
figure  
plot(time8,measuredVel8); hold on  
plot(time8,velreferencia_ensaio6); hold off  
title('Resultados Malha Fechada - Ensaio tras t2 - ganhos frente');  
xlabel('Tempo [s]');  
ylabel('Velocidade [RPM]');  
grid minor;
```

```
%% Código de otimização para obtenção de ganhos de um Controlador PID
% Universidade de Brasília - Faculdade Gama
% Autor: Professor André Murilo Pinto
% Código utilizado no curso - Sistemas de Controle Automotivo
% Alterações feitas por: Pedro H. L. Figueiredo - Graduando em Engenharia
% Automotiva

clc; close all; clear all

% Ponto Inicial dos Parâmetros do Controlador
kp_ini=1; ki_ini=1; kd_ini=1;
x0 = [kp_ini ki_ini kd_ini];

% Limites Máximos dos Parâmetros
kp_max=30; ki_max=30; kd_max=30;
max=[kp_max ki_max kd_max];

% Especificação Desejada
tresposta_des=1;

% Modelo do Sistema
G = zpk([], [-2,004], 0,864);
% Ensaio1 - G = zpk([], [-4.451], 4.433);
% Ensaio2 - G = zpk([], [-3.931], 3.934);
% Demais testes - G = zpk([-27.77], [-3.91 -229.6], 32.209);
% Deu certo G = zpk([], [-2.146], 0.909);

%Tempo de Simulação
ts=0.01; tfinal=20; t=[0:ts:tfinal];

% Definição do perfil de entrada
yfinal=1; u=ones(length(t),1);

% Definição do Problema de Otimização
options = optimset('Display','iter','MaxFunEvals',500);
[x,fval] = fmincon(@funcao_objetiva_PID_v2,x0,[],[],[],[],[0 0 0],max,[],options,↵
tresposta_des,G,t,u)

% Controlador com Parâmetros Ótimos obtidos
Gc=pid(x(1),x(2),x(3),0)

% Simulação em Malha Fechada
MF1=feedback(G,1); MF2=feedback(Gc*G,1);Gref=tf([0 1],[tresposta_des 1]);
u=ones(length(t),1);

y1=lsim(MF1,u,t); y2=lsim(MF2,u,t); y3=lsim(Gref,u,t);
plot(t,y1,t,y2,t,y3);legend('Sem PID','Com PID','Trajetória referência');title↵
('Resposta ao Degrau - PID');
```

```
function f = funcao_objetiva_PID_v2(x0,tr,G,t,u)

% Parâmetros do controlador à serem otimizados
kp=x0(1);
ki=x0(2);
kd=x0(3);
Gc=pid(kp,ki,kd,0);

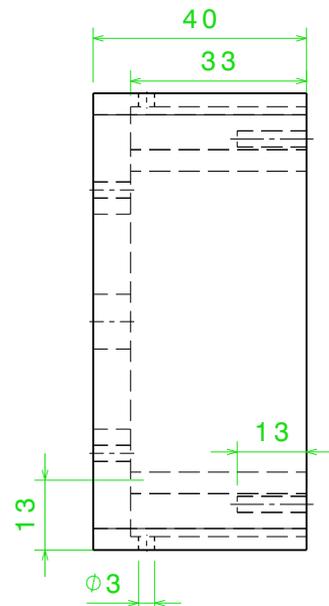
% Malha Fechada
MF=feedback(Gc*G,1);

% Simulação do Sistema
y=lsim(MF,u,t);

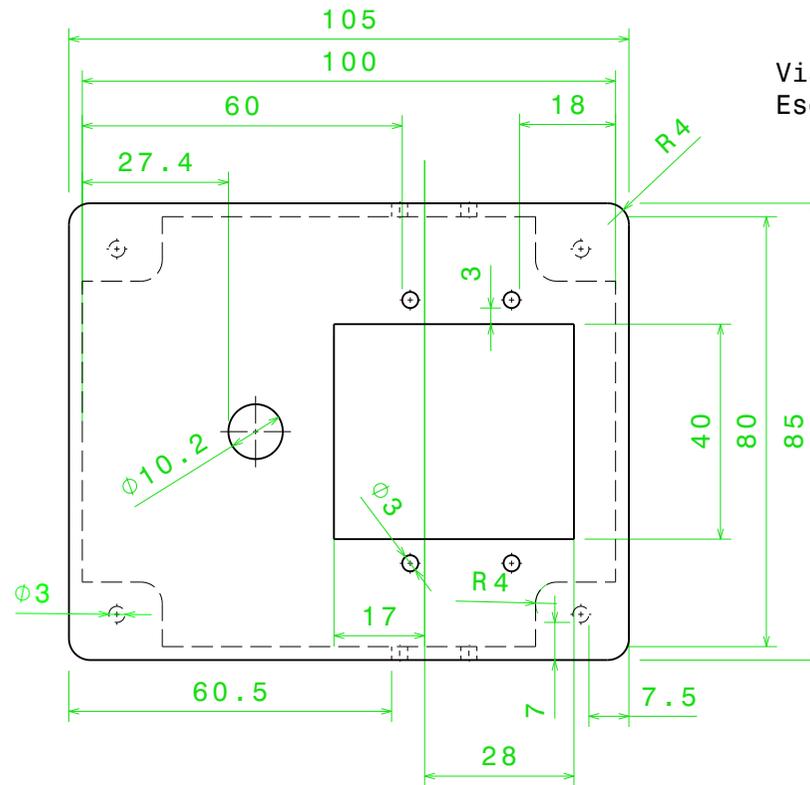
% Saída Referência
Gref=tf([0 1],[tr 1]);
yref=lsim(Gref,u,t);

% Função Custo Obtida pelos Objetivos do Controle
f = sqrt(sum((y-yref).^2));
end
```

APÊNDICE B – Desenhos técnicos dos componentes do Sistema de Direção

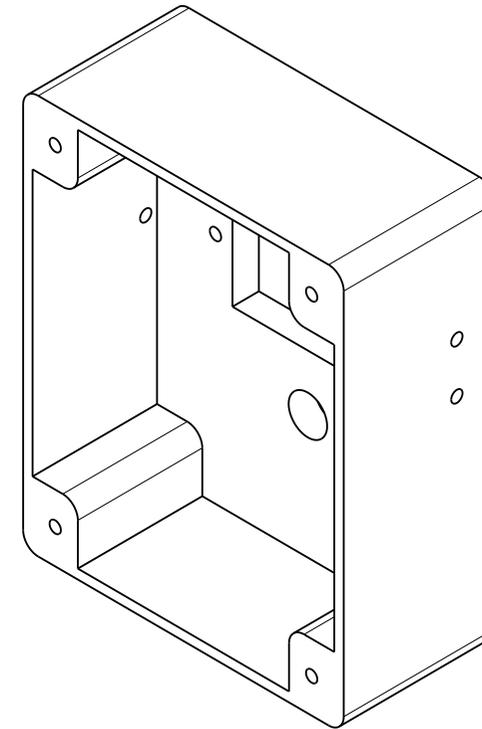


Vista Frontal
Escala 1:1

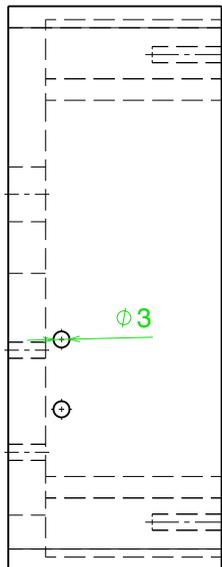


Visão Lateral Esquerda
Escala 1:1

OBS: TODAS AS MEDIDAS
ESTÃO EM MILÍMETROS

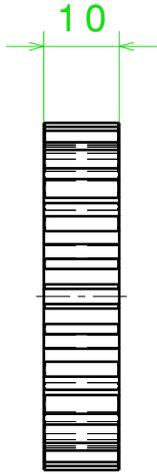


Vista Isométrica
Escala 1:1

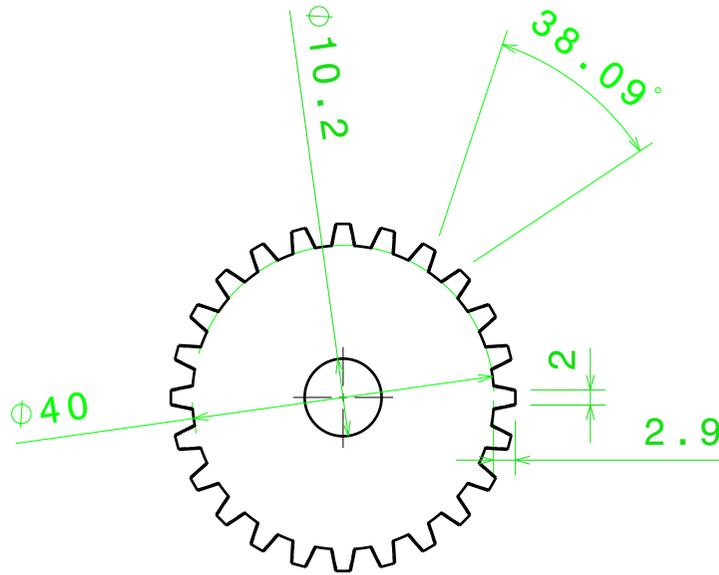


Vista Superior
Escala 1:1

Desenhado por: João Vítor			I	-
DATE: 03/10/2022			H	-
		G	-	
		F	-	
		E	-	
		D	-	
		C	-	
		B	-	
		A	-	
TAMANHO A3		Projeto SegurAuto		
SCALE 1:1	DRAWING NUMBER Caixa	SHEET 1/1		

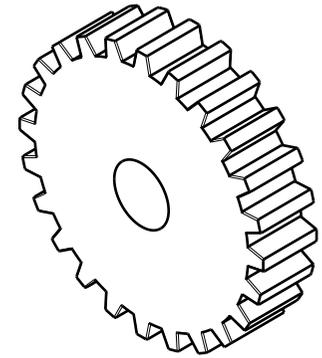


Vista Frontal
Escala 1:1



Vista Lateral Esquerda
Escala 1:1

OBS: TODAS AS
MEDIDAS ESTÃO
EM MILÍMETROS



Vista Isométrica
Escala 1:1



Vista Superior
Escala 1:1

DESENHADO POR: João Vítor				I	-
DATA: 03/10/2022				H	-
				G	-
				F	-
				E	-
TAMANHO A4				D	-
ESCALA 1:1				Projeto SecurAuto	
		DRAWING NUMBER		B	-
		Engrenagem de 24 dentes		A	-
		SHEET			
		1 / 1			

This drawing is our property; it can't be reproduced or communicated without our written agreement.

D

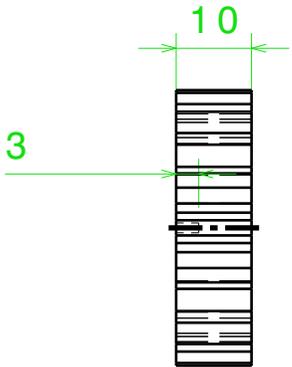
C

B

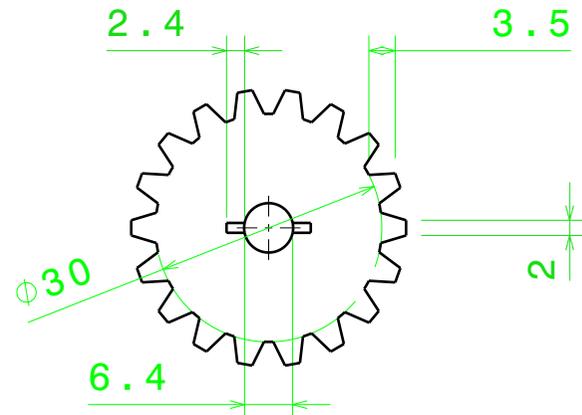
A

4

4

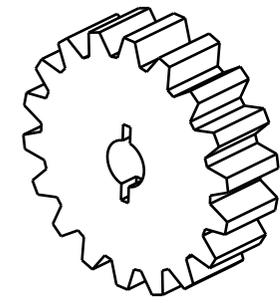


Vista Frontal
Escala 1:1



Vista Lateral Esquerda
Escala 1:1

OBS: TODAS AS MEDIDAS ESTÃO EM MILÍMETROS



Vista Isométrica
Escala 1:1

3

3

2

2



Vista Superior
Escala 1:1

1

1

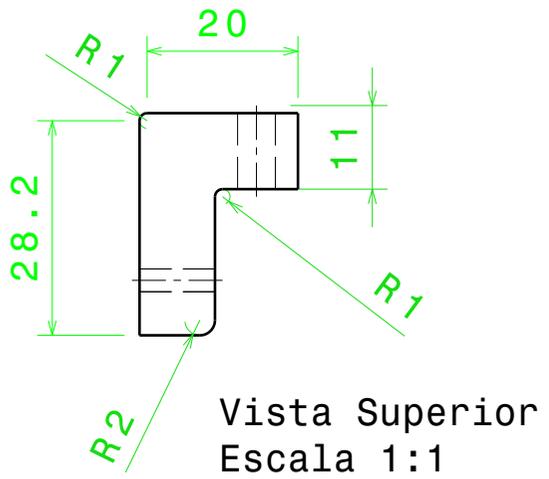
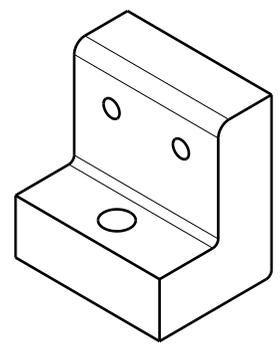
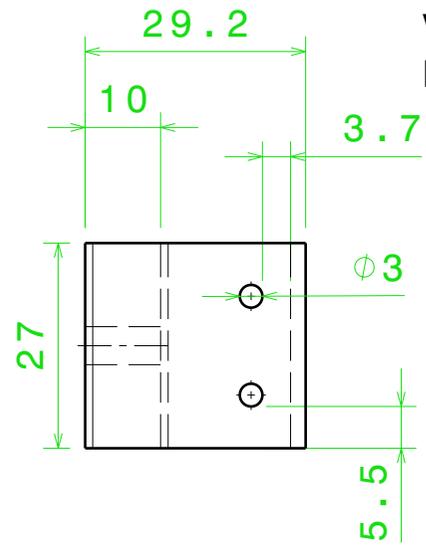
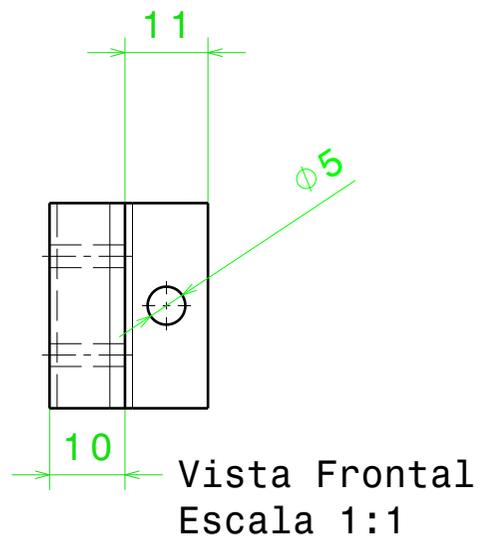
DESENHADO POR: João Vítor				I	-
DATA: 03/10/2022				H	-
TAMANHO: A4				G	-
				Projeto SecurAuto	
ESCALA: 1:1	DRAWING NUMBER: Engrenagem de 18 dentes	SHEET: 1/1	E		
This drawing is our property; it can't be reproduced or communicated without our written agreement.				D	-
				C	-
				B	-
				A	-

D

A

D C B A

4
3
2
1



DESENHADO POR: João Vítor				I	-	
DATA: 03/10/2022				H	-	
				G	-	
				F	-	
				E	-	
TAMANHO: A4			Projeto SecurAuto		D	-
ESCALA: 1:1					C	-
		DRAWING NUMBER		SHEET		
		Pé de fixação		1/1	B	-
This drawing is our property; it can't be reproduced or communicated without our written agreement.					A	-

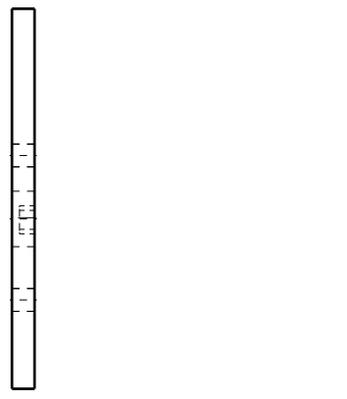
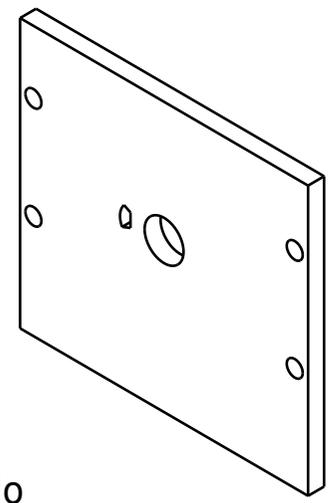
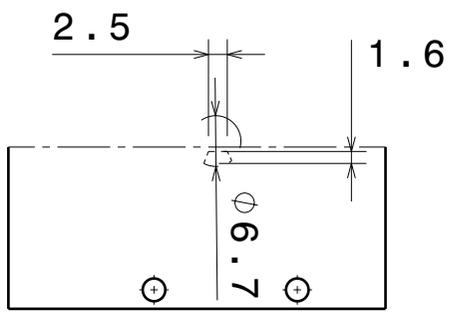
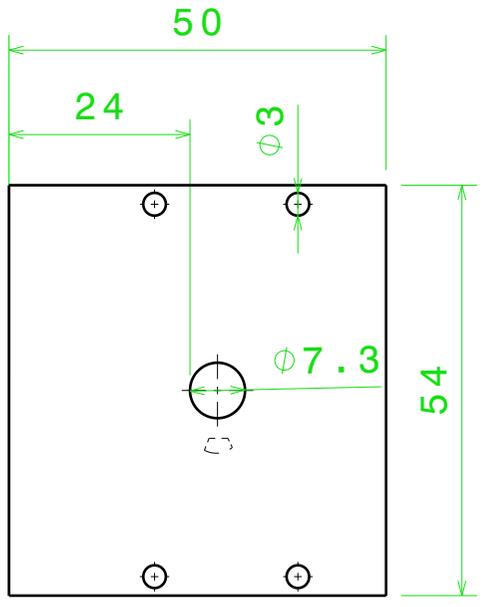
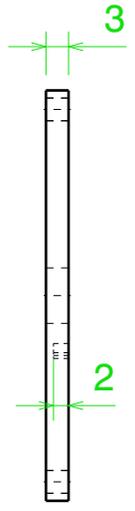
D A

D

C

B

A



DESENHADO POR:
João Vítor
DATA:
03/10/2022

TAMANHO
A4

ESCALA
1:1

Projeto SecurAuto

DRAWING NUMBER

Plataforma

I	-
H	-
G	-
F	-
E	-
D	-
C	-
B	-
A	-

SHEET
1/1

This drawing is our property; it can't be reproduced or communicated without our written agreement.

D

A

4

3

2

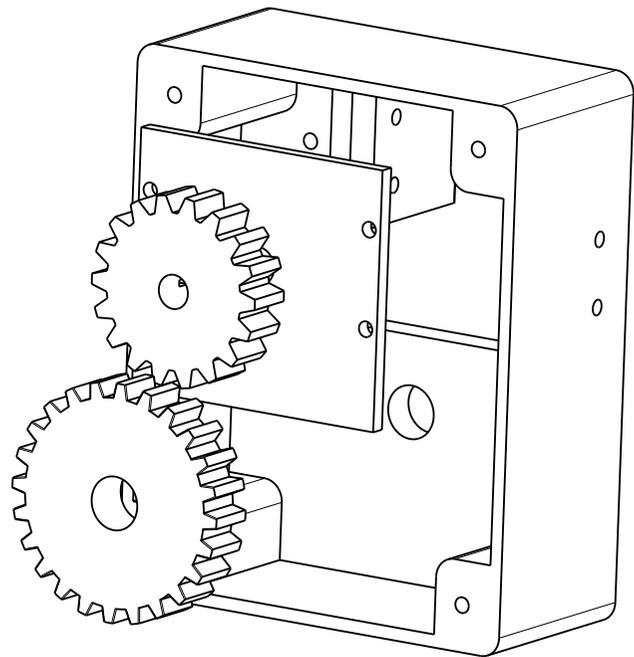
1

4

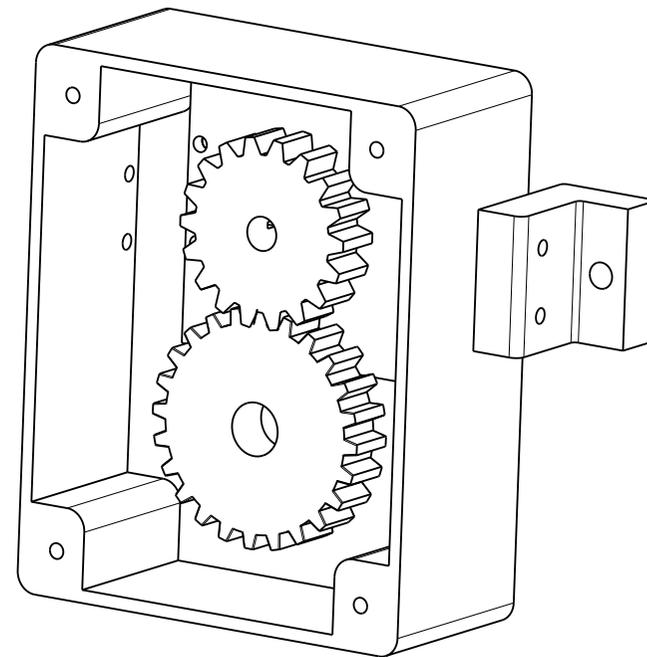
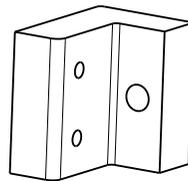
3

2

1



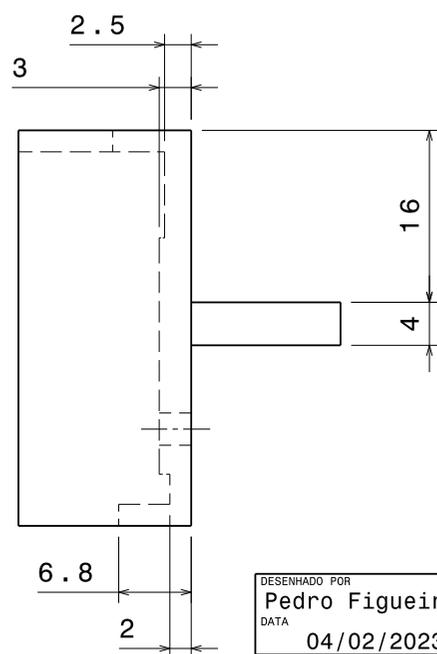
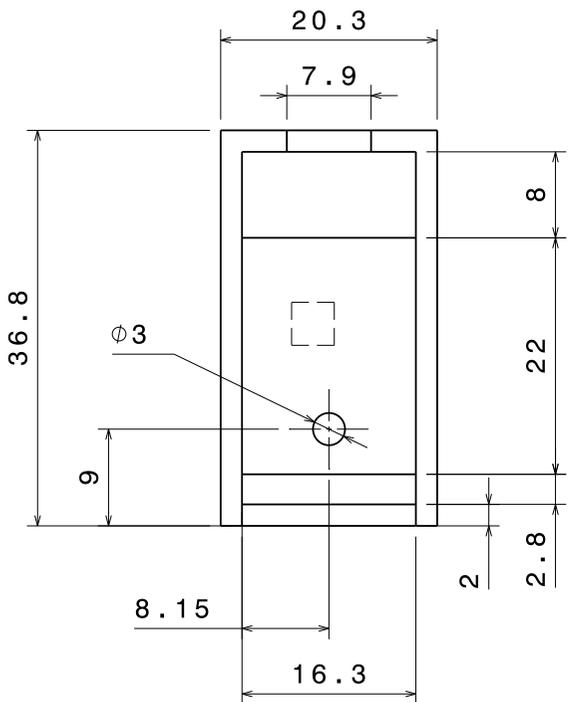
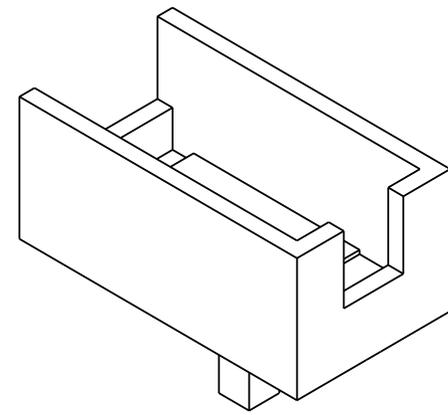
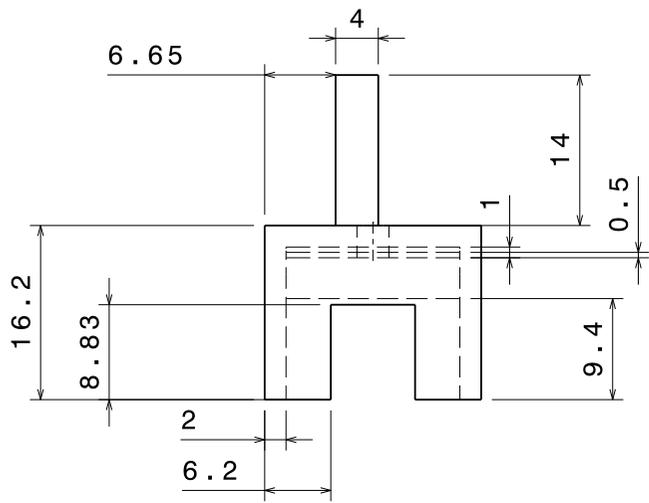
Vista Explodida
Escala 1:1



Vista Isométrica
Escala 1:1

DESENHADO POR João Vítor				I	-
DATA 03/10/2022				H	-
				G	-
				F	-
				E	-
				D	-
				C	-
				B	-
				A	-
TAMANHO A3		Projeto SegurAuto			
ESCALA 1:1		DRAWING NUMBER Vista Explodida	SHEET 1/1		
This drawing is our property; it can't be reproduced or communicated without our written agreement.					

APÊNDICE C – Desenhos técnicos dos componentes do Sistema de Aceleração e Frenagem



DESENHADO POR Pedro Figueiredo				I	-
DATA 04/02/2023				H	-
				G	-
				F	-
				E	-
				D	-
TAMANHO A3		Projeto SecurAuto		C	-
ESCALA 2:1				NOME DO DESENHO Caixa do sensor encoder	FOLHA 1/1
This drawing is our property; it can't be reproduced or communicated without our written agreement.				A	-

4

4

3

3

2

2

1

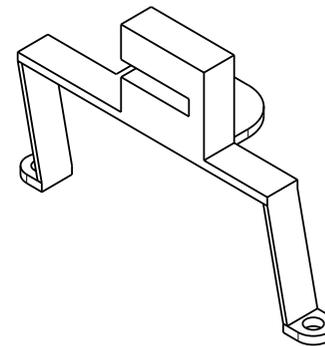
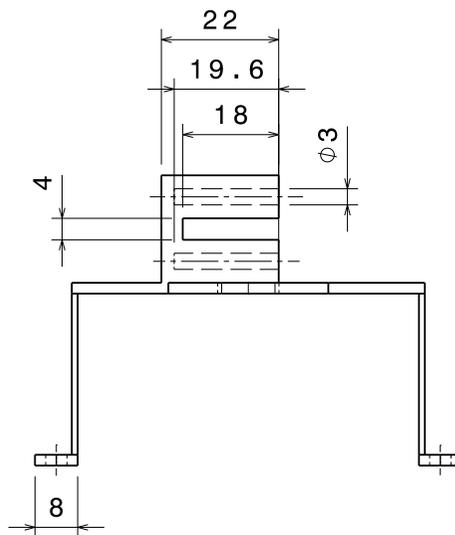
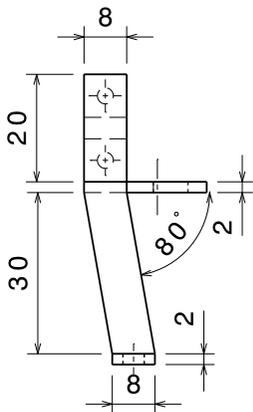
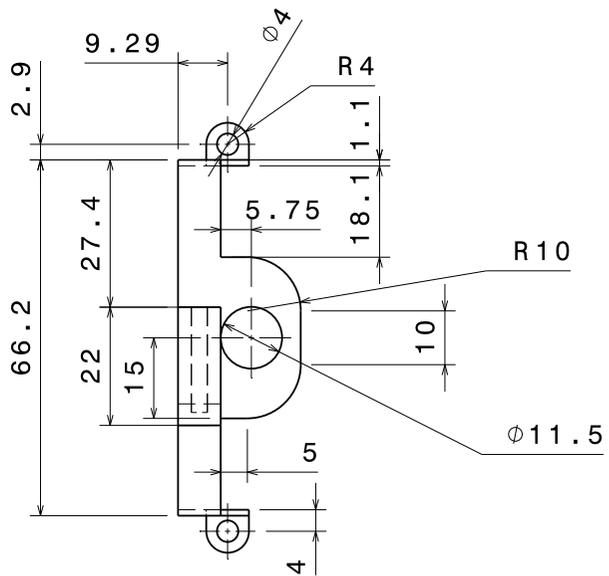
1

H

G

B

A



DESENHADO POR Pedro Figueiredo				I	-
DATA 04/02/2023				H	-
				G	-
				F	-
				E	-
				D	-
				C	-
				B	-
				A	-
TAMANHO A3		Projeto SegurAuto			
ESCALA 1:1		NOME DO DESENHO Suporte do sensor encoder	FOLHA 1/1		
This drawing is our property; it can't be reproduced or communicated without our written agreement.					

H G F E D C B A

4

4

3

3

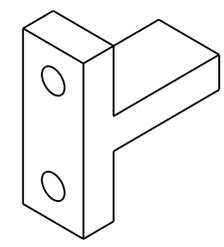
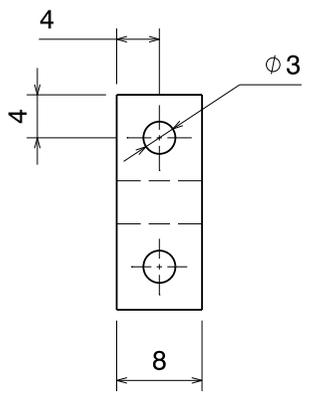
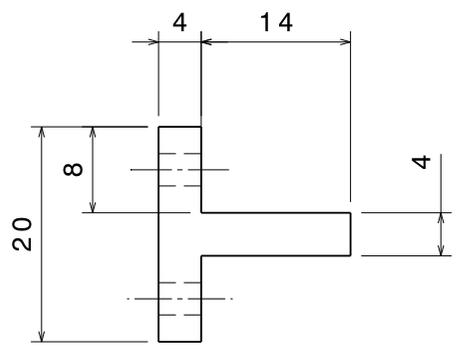
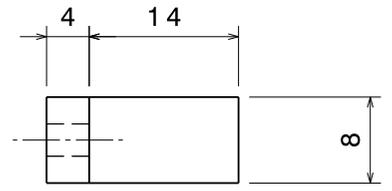
2

2

1

1

H G F E D C B A

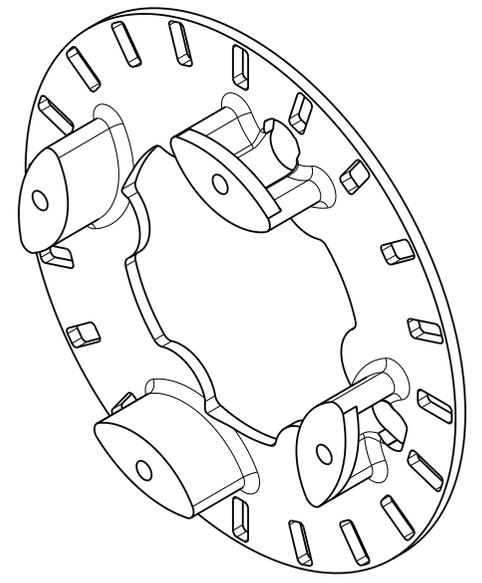
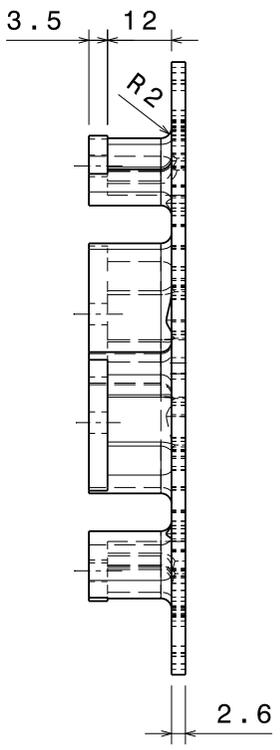
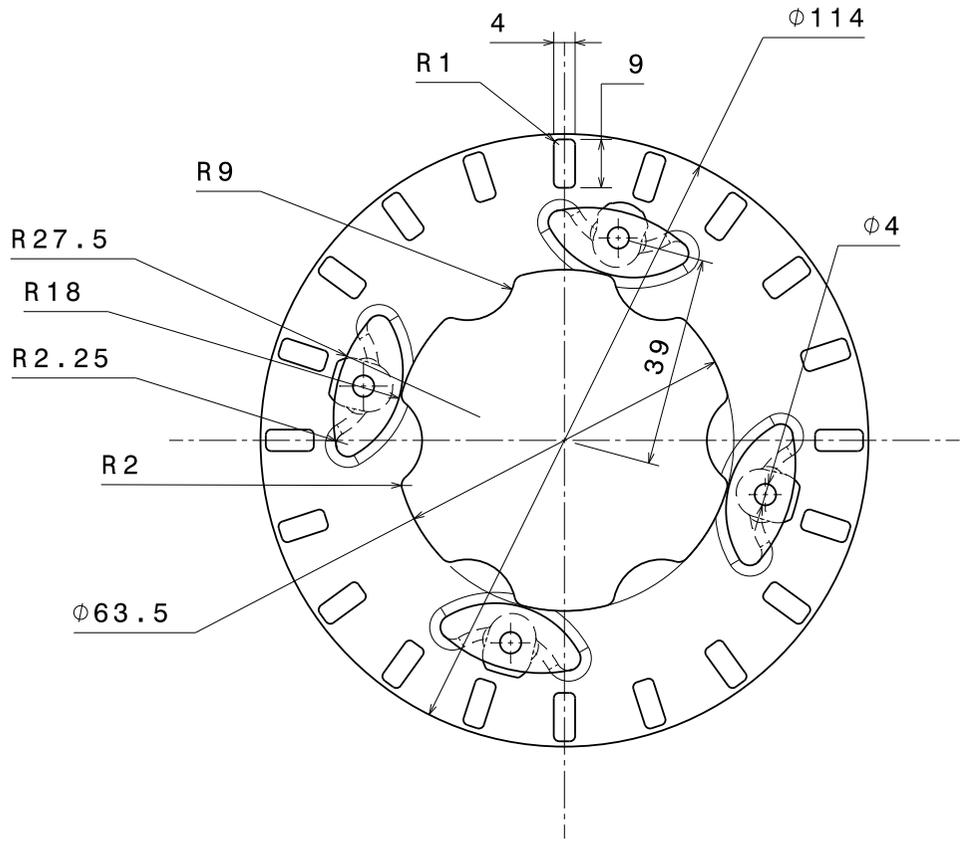


DESENHADO POR Pedro Figueiredo				I	-
DATA 04/02/2023				H	-
				G	-
				F	-
				E	-
				D	-
TAMANHO A3		Projeto SegurAuto		C	-
ESCALA 2:1		NOME DO DESENHO Complemento do suporte do sensor	FOLHA 1/1	B	-
This drawing is our property; it can't be reproduced or communicated without our written agreement.				A	-

H G F E D C B A

4

4



3

3

2

2

1

1

H G F E D C B A

DESENHADO POR: Pedro Figueiredo			I	-
DATA: 25/01/2023			H	-
		G	-	
		F	-	
		E	-	
		D	-	
		C	-	
		B	-	
		A	-	
TAMANHO: A3		Projeto SecurAuto		
ESCALA: 1:1	NOME DO DESENHO: Disco perfurado	SHEET: 1/1		
This drawing is our property; it can't be reproduced or communicated without our written agreement.				

APÊNDICE D – Códigos de Caracterização dos Motores

1. Sistema de Direção

```
1 float t;
2 float dt = 20;
3 float t0 = millis();
4 float t1 = t0;
5 int PWM;
6 int flag_s = 1; //direita
7 int anguloroda;
8
9 void setup() {
10 pinMode(5, OUTPUT);
11 pinMode(6, OUTPUT);
12 Serial.begin(9600);
13 }
14
15 void loop() {
16
17 t = millis();
18 anguloroda = calculaangulo();
19 if((t - t0) <= 5000){
20     PWM = 0;
21     Serial.print((t-t0));
22     Serial.print(",");
23     Serial.println(anguloroda);
24 }else if((t - t0) > 5000 && (t - t0) <= 10000){
25     PWM = 50;
26     flag_s = 1;
27     Serial.print((t-t0));
28     Serial.print(",");
29     Serial.println(anguloroda);
30 }else if((t - t0) > 10000 && (t - t0) <= 15000){
31     PWM = 60;
32     flag_s = -1;
33     Serial.print((t-t0));
34     Serial.print(",");
```

```

35     Serial.println(anguloroda);
36 }else if((t - t0) > 15000 && (t - t0) <= 20000){
37     PWM = 80;
38     flag_s = 1;
39     Serial.print((t-t0));
40     Serial.print(",");
41     Serial.println(anguloroda);
42 }else if((t - t0) > 20000 && (t - t0) <= 25000){
43     PWM = 100;
44     flag_s = -1;
45     Serial.print((t-t0));
46     Serial.print(",");
47     Serial.println(anguloroda);
48 }else if((t - t0) > 25000 && (t - t0) <= 30000){
49     PWM = 120;
50     flag_s = 1;
51     Serial.print((t-t0));
52     Serial.print(",");
53     Serial.println(anguloroda);
54 }else if((t - t0) > 30000){
55     PWM = 0;
56     Serial.print((t-t0));
57     Serial.print(",");
58     Serial.println(anguloroda);
59 }
60
61     if(flag_s == 1){
62         analogWrite(5,PWM);
63         analogWrite(6,0);
64     }else{
65         analogWrite(5,0);
66         analogWrite(6,PWM);
67     }
68 }
69
70 int calculaangulo(){
71
72     // CALCULA A M DIA DE 500 VALORES LIDOS:
73     int anguloatual = 0;
74     for (int i = 0; i < 500; i++)
75     {
76         int valor = analogRead(A0);

```

```

77     int angulocoluna = map(valor, 100, 300, -40, 40);
78     anguloatual += angulocoluna;
79 }
80
81     anguloatual = anguloatual / 500;
82
83     // CONVERS O DO VALOR DA COLUNA PARA O DAS RODAS:
84     int anguloroda = 0.676 * anguloatual;
85
86     // LIMITA O DA LEITURA DE ANGULO M X
87     if (anguloroda > 25)
88     {
89         anguloroda = 25;
90     }
91     else if (anguloroda < -25)
92     {
93         anguloroda = -25;
94     }
95     return anguloroda;
96 }

```

2. Sistema de Aceleração e Frenagem

```

1
2     // Biblioteca para Interrupt
3 #include <util/atomic.h>
4 #include "ABM.h"
5
6 //Vari veis
7 double velocityRPM_GLOBAL = 0; // Velocidade Medida sem
   Filtro
8 double velocityRPMFiltered = 0; // Velocidade Medida Filtrada
9 double velocityRPMPrevious = 0; // Velocidade Medida Anterior
   - Utilizada no filtro
10 double old_velocityRPM = 0; // Velocidade Medida Anterior
11 double previousPWM = 0; // Valor de PWM anterior
12
13 // Vari veis vol teis utilizadas para c lculo da
   Velocidade medida em RPM
14 volatile double velocity_i = 0;
15 volatile long prevT_i = 0;
16
17 // Velocidade requerida

```

```

18 unsigned int rpm = 0;
19
20 unsigned long lastPrint = 0;
21
22 // Variavel de controle da direcao
23 unsigned int controlSignal = 0;
24
25 void readEncoder() {
26     // Esta funcao captura o intervalo de tempo entre um
27     // pulso e outro, vindos do sensor de velocidade
28     // e transforma o tempo em velocidade a partir de pulso
29     // por segundo
30
31     long currT = micros();
32     double deltaT = ((double)(currT - prevT_i)) / 1.0e6;
33
34     // Caso o tempo entre medições seja muito pequeno, o
35     // valor da velocidade
36     // tende a infinito, o que não é desejável, por isso
37     // as medições
38     // que ocorram em menos de 0.015 segundos
39
40     if(deltaT >= 0.015) {
41         velocity_i = 1 / deltaT;
42     }
43
44     prevT_i = currT; //Salva o momento da última medição
45 }
46
47 void setMotor(int directionCtrl, int output_PWM) {
48     int oldPMW = output_PWM;
49
50     if (directionCtrl == 1) {
51         // Gira num sentido
52         analogWrite(L_PWM_2, 0);
53         analogWrite(R_PWM_2, output_PWM);
54         analogWrite(L_PWM_1, 0);
55         analogWrite(R_PWM_1, output_PWM);
56         previousPWM = oldPMW;
57     } else if (directionCtrl == 2) {
58         // Gira em outro sentido
59         analogWrite(R_PWM_2, 0);

```

```

56     analogWrite(L_PWM_2, output_PWM);
57     analogWrite(R_PWM_1, 0);
58     analogWrite(L_PWM_1, output_PWM);
59     previousPWM = -oldPMW;
60 } else if (directionCtrl == 0) {
61     // Para os motores
62     analogWrite(R_PWM_2, 0);
63     analogWrite(L_PWM_2, 0);
64     analogWrite(R_PWM_1, 0);
65     analogWrite(L_PWM_1, 0);
66     previousPWM = 0;
67
68 }
69 }
70
71 void setupPins() {
72
73     // Configura o dos Pinos
74     // Pino Sensor
75     pinMode(D_Sensor, INPUT);
76     // Pinos PWM
77     pinMode(R_PWM_1, OUTPUT);
78     pinMode(L_PWM_1, OUTPUT);
79     pinMode(R_PWM_2, OUTPUT);
80     pinMode(L_PWM_2, OUTPUT);
81 }
82
83 void setup() {
84
85     // Inicializa o da Serial
86     Serial.begin(115200);
87
88     // Configura o dos pinos
89     setupPins();
90
91     // Habilitando as interrupções para medição da
92     // velocidade
93     attachInterrupt(digitalPinToInterrupt(D_Sensor),
94                     readEncoder, RISING); // Rising - Subida
95 }
96
97 void testFunction() {

```

```

96
97 unsigned long currentTime = millis(); //Tempo atual
98
99 if(currentTime >= 30000 && currentTime <= 35000) {
100     controlSignal = 0;
101     rpm = 0;
102 }else if(currentTime > 35000 && currentTime <= 50000) {
103     controlSignal = 2;
104     rpm = 255;
105 }else if(currentTime > 50000 && currentTime <= 55000) {
106     controlSignal = 0;
107     rpm = 0;
108 }else {
109     controlSignal = 0;
110     rpm = 0;
111 }
112
113 unsigned long sampleTime = millis() - lastPrint;
114 if(currentTime >= 30000 && currentTime <= 55000) {
115     Serial.print(currentTime); // Tempo
116     Serial.print(",");
117     Serial.print(velocityRPMFiltered); // Velocidade com
        filtro
118     Serial.print(",");
119     Serial.println(rpm); // PWM fornecido
120     lastPrint = millis();
121 }
122
123 }
124
125 void controlFunction() {
126     // A variavel ser lida no bloco at mico pois o valor
127     // alterado num interrupt
128
129     double velocityCountPerSecond = 0; // Velocidade [1 Count
        / X seconds]
130
131     ATOMIC_BLOCK(ATOMIC_RESTORESTATE) {
132         velocityCountPerSecond = velocity_i;
133     }
134
135     // Converte count/s para RPM

```

```
136     double velocityRPM = velocityCountPerSecond / 20.0 *
137         60.0;
138     velocityRPM_GLOBAL = velocityRPM;
139     // Filtro passa baixa
140     velocityRPMFiltered = 0.854 * velocityRPMFiltered +
141         0.0728 * velocityRPM + 0.0728 * velocityRPMPrevious;
142     velocityRPMPrevious = velocityRPM;
143     // Condição que exclui a possibilidade do armazenamento
144     // de um valor de velocidade anterior
145     if(rpm == 0 && velocityRPM == old_velocityRPM){
146         velocityRPMFiltered = 0;
147         velocityRPMPrevious = 0;
148         velocityRPM = 0;
149         velocity_i = 0;
150     }
151     old_velocityRPM = velocityRPM; // Armazenando valor de
152     // velocidade anterior
153     setMotor(controlSignal, rpm);
154 }
155
156 void loop() {
157
158
159     testFunction();
160     controlFunction();
161
162 }
```

Anexos

ANEXO A – Tabela das principais transformadas de Laplace

	$f(t)$	$F(s)$
1	Impulso unitário $\delta(t)$	1
2	Degrau unitário $1(t)$	$\frac{1}{s}$
3	t	$\frac{1}{s^2}$
4	$\frac{t^{n-1}}{(n-1)!}$ ($n = 1, 2, 3, \dots$)	$\frac{1}{s^n}$
5	t^n ($n = 1, 2, 3, \dots$)	$\frac{n!}{s^{n+1}}$
6	e^{-at}	$\frac{1}{(s+a)}$
7	te^{-at}	$\frac{1}{(s+a)^2}$
8	$\frac{1}{(n-1)!} t^{n-1} e^{-at}$ ($n = 1, 2, 3, \dots$)	$\frac{1}{(s+a)^n}$
9	$t^n e^{-at}$ ($n = 1, 2, 3, \dots$)	$\frac{n!}{(s+a)^{n+1}}$
10	$\text{sen } \omega t$	$\frac{\omega}{s^2 + \omega^2}$
11	$\text{cos } \omega t$	$\frac{s}{s^2 + \omega^2}$

Fonte: (OGATA, 2010)