



UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
MESTRADO EM ENGENHARIA ELÉTRICA

FELLIPE FONSECA BASTOS

ESTUDO E IMPLEMENTAÇÃO DE CONTROLADORES FUZZY E PID PARA
CONTROLE DE DIREÇÃO E VELOCIDADE DE UM AGV COM VISÃO
COMPUTACIONAL

MOSSORÓ

2019

FELLIPE FONSECA BASTOS

ESTUDO E IMPLEMENTAÇÃO DE CONTROLADORES FUZZY E PID PARA
CONTROLE DE DIREÇÃO E VELOCIDADE DE UM AGV COM VISÃO
COMPUTACIONAL

Dissertação apresentada ao Mestrado em Engenharia Elétrica do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Mestre em Engenharia Elétrica.

Linha de Pesquisa: Sistemas de Controle e Automação.

Orientador: Prof. Dr. Marcelo Roberto Bastos Guerra Vale.

MOSSORÓ

2019

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

©Todos os direitos estão reservados à Universidade Federal Rural do Semi-Árido. O conteúdo desta obra é de inteira responsabilidade do (a) autor (a), sendo o mesmo, passível de sanções administrativas ou penais, caso sejam infringidas as leis que regulamentam a Propriedade Intelectual, respectivamente, Patentes: Lei nº 9.279/1996, e Direitos Autorais: Lei nº 9.610/1998. O conteúdo desta obra tornar-se-á de domínio público após a data de defesa e homologação da sua respectiva ata, exceto as pesquisas que estejam vinculadas ao processo de patenteamento. Esta investigação será base literária para novas pesquisas, desde que a obra e seu (a) respectivo (a) autor (a) seja devidamente citado e mencionado os seus créditos bibliográficos.

B327e Bastos, Fellipe Fonseca.
Estudo e implementação de controladores Fuzzy e
PID para controle de direção e velocidade de um
AGV com visão computacional / Fellipe Fonseca
Bastos. - 2019.
109 f. : il.

Orientador: Marcelo Roberto Bastos Guerra
Vale.
Dissertação (Mestrado) - Universidade Federal
Rural do Semi-árido, Programa de Pós-graduação em
Engenharia Elétrica, 2019.

1. Robótica móvel. 2. Visão computacional. 3.
Controle PID. 4. Controle Fuzzy. I. Vale, Marcelo
Roberto Bastos Guerra, orient. II. Título.

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC's) foi desenvolvido pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP) e gentilmente cedido para o Sistema de Bibliotecas da Universidade Federal Rural do Semi-Árido (SISBI-UFERSA), sendo customizado pela Superintendência de Tecnologia da Informação e Comunicação (SUTIC) sob orientação dos bibliotecários da instituição para ser adaptado às necessidades dos alunos dos Cursos de Graduação e Programas de Pós-Graduação da Universidade.

FELLIPE FONSECA BASTOS


ESTUDO E IMPLEMENTAÇÃO DE CONTROLADORES FUZZY E PID PARA
CONTROLE DE DIREÇÃO E VELOCIDADE DE UM AGV COM VISÃO
COMPUTACIONAL

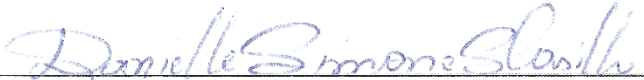
Dissertação apresentada ao Mestrado em Engenharia Elétrica do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Mestre em Engenharia Elétrica.


Linha de Pesquisa: Sistemas de Controle e Automação.

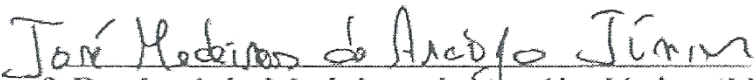
Defendida em: 27 / 06 / 2019.

BANCA EXAMINADORA


Prof. Dr. Marcelo Roberto Bastos Guerra Vale (UFERSA)
Presidente


Profa. Dra. Danielle Simone da Silva Casillo (UFERSA)
Membro Examinador


Prof. Dr. Leonardo Augusto Casillo (UFERSA)
Membro Examinador


Prof. Dr. José de Medeiros de Araújo Júnior (UFPI)
Membro Examinador

*Aos meus pais, por todo o esforço,
dedicação e suporte.*

AGRADECIMENTOS

Aos amigos e colegas do Curso de Engenharia Elétrica e do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal Rural do Semi-Árido, pelo apoio e momentos de descontração.

A todo o corpo docente da graduação e do Mestrado em Engenharia Elétrica, em especial, ao meu orientador Marcelo Roberto Bastos Guerra Vale pela paciência e suporte desde a graduação até o desenvolvimento dessa pesquisa.

A todos os amigos e familiares que ajudaram, direta ou indiretamente, na realização deste trabalho.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro.

"Não importa o quão ruim a vida possa ser, há sempre alguma coisa que você pode fazer e ter sucesso. Enquanto há vida, há esperança."

Stephen Hawking

RESUMO

A diminuição dos custos de fabricação de componentes eletrônicos, o surgimento de novas tecnologias e áreas de pesquisa, e a necessidade de mão de obra de baixo custo vêm impulsionando a construção de robôs móveis e favorecendo sua disseminação em vários setores, principalmente no setor industrial. Entre os robôs móveis, os Veículos Guiados Automaticamente (AGV - Automated Guided Vehicle em inglês) estão ganhando espaço no setor industrial por diminuírem custos e aumentarem a produtividade. Os AGVs são robôs utilizados para locomover equipamentos e peças no chão de fábrica e não necessitam de supervisão humana. O desenvolvimento de um controlador *Fuzzy* em conjunto com controladores Proporcional Integral Derivativo (PID), a utilização de câmera digital e a aplicação de técnicas de visão computacional podem ser utilizadas para auxiliar AGVs nos deslocamentos, evitando colisões e divergências das marcações de trajeto. Com base nos fatos apresentados, o objetivo deste trabalho é estudar e implementar um sistema de controle de direção e velocidade baseado em controladores *Fuzzy* e PID para um robô móvel experimental. Neste robô serão utilizados uma câmera digital e técnicas de visão computacional para detectar percursos a serem seguidos. A metodologia empregada consiste em desenvolver um algoritmo que capture as imagens e, a partir de técnicas de processamento de imagem, entregue uma imagem segmentada por binarização que permita encontrar o centro da trilha a ser seguida. Estas informações são tratadas em um controlador *Fuzzy* que determina a direção do robô ao variar a velocidade de cada roda. Controladores PID são utilizados para garantir a velocidade individual dos motores e consequentemente das rodas. Por fim, são utilizadas noções básicas de odometria para recriar o trajeto percorrido pelo robô e assim comparar com o trajeto real. Dessa forma, o controlador *Fuzzy* em conjunto com os controladores PID foram capazes de controlar o robô durante toda trajetória, favorecendo seu alinhamento com o centro da trilha.

Palavras-chave: Robótica móvel. Visão computacional. Controle PID. Controle *Fuzzy*.

ABSTRACT

The decrease in manufacturing costs of electronics components, the emergence of new technologies and research areas, and the need for cheaper labor have been driving the construction of mobile robots and favoring their spread in several sectors, particularly in the industrial sector. Among mobile robots, Automated Guided Vehicles (AGV) are gaining ground in the industrial sector by lowering costs and increasing productivity. AGVs are robots used to move equipment and parts to the factory floor and do not require human supervision. The development of a Fuzzy controller in conjunction with Proportional-Integral-Derivative (PID) controllers, the use of a digital camera and the application of computer vision techniques can be used to assist AGVs in the displacements, avoiding collisions and divergences in the waypoints. Based on the facts presented, the objective of this work is to study and implement a control and speed control system based on Fuzzy and PID controllers for an experimental mobile robot. In this robot will be used a digital camera and computer vision techniques to detect paths to be followed. The methodology consists in developing an algorithm that captures the images and, based on image processing techniques, delivers an image segmented by binarization that allows finding the center of the track to be followed. This information is provided in a Fuzzy controller that determines the direction of the robot by varying the speed of each wheel. PID controllers are used to ensure the individual speed of the motors and thus the wheels. Finally, basic notions of odometry are used to recreate the path traveled by the robot and thus compare with the real path. In this way, the Fuzzy controller in conjunction with the PID controllers were able to control the robot during the entire trajectory, favoring its alignment with the center of the track.

Keywords: Mobile robots. Computer Vision, PID Control. Fuzzy Control.

LISTA DE FIGURAS

Figura 1 – Algumas estruturas de AGVs.	25
Figura 2 - Sensor utilizado na captura de imagens.	27
Figura 3 - Processo de captura de imagem digital.	28
Figura 4 - Representação da função Gaussiana bidimensional com $\sigma = 1$	30
Figura 5 - Sistema <i>Fuzzy</i>	34
Figura 6 - Variável linguística “Temperatura”.	36
Figura 7 - Tipos de funções de pertinência.	36
Figura 8 - Diagrama de blocos do controlador PID.	40
Figura 9 - Curva de resposta ao degrau, parâmetros K,L e T.	42
Figura 10 - Modelo cinemático AGV.....	47
Figura 11 - Diagrama de blocos da metodologia.	51
Figura 12 - Estrutura do robô.	52
Figura 13 - Motores A e B.	53
Figura 14 - <i>Encoder</i> de Quadratura.....	54
Figura 15 - Modelo de roda utilizada.	55
Figura 16 - Roda castor.....	56
Figura 17 - Ponte H dupla.	57
Figura 18 - Pinagem ponte H.	58
Figura 19 - Raspberry Pi	59
Figura 20 - Câmera CMOS.....	61
Figura 21 - Baterias LiPo.....	62
Figura 22 - Conversor Buck.....	63
Figura 23 - Fluxograma da fase de visão computacional.	64
Figura 24 - Objetivo de toda etapa de visão computacional.....	65
Figura 25 - Diagrama de blocos do controlador <i>Fuzzy</i>	66
Figura 26 - Variável linguística “Erro”	67
Figura 27 - Variável linguística de entrada “DesvioErro”	69
Figura 28 - Variáveis de saída do controlador <i>Fuzzy</i>	71
Figura 29 - Conjunto de regras.....	72
Figura 30 - Versão final do robô móvel.	77
Figura 31 - Parte inferior do robô.	78
Figura 32 - AGV sobre a trilha a ser seguida.	78
Figura 33 - Tacômetro medindo o RPM do Motor B.....	79

Figura 34 - PPR por segundo Motor B	79
Figura 35 - PPR a 6V - Motor B.....	80
Figura 36 - RPM do Motor B a 9V.	81
Figura 37 - Resultados obtidos com as etapas de visão.	82
Figura 38 - Imagem binarizada pelo método de Otsu e invertida.	83
Figura 39 - Matriz de zeros e uns.....	84
Figura 40 - Resultado do algoritmo de visão.....	85
Figura 41 - Trilha circular feita com pincel de quadro.....	86
Figura 42 - Imagem binarizada e complementada/invertida.....	87
Figura 43 - Imagem capturada pelo robô.	87
Figura 44 - Ponto central da linha branca.	88
Figura 45 - Saídas do <i>Fuzzy</i> para entradas 0 e 0.	89
Figura 46 - Entradas diferente, saídas diferentes.....	90
Figura 47 - Resposta ao degrau do Motor A.	92
Figura 48 - Resposta ao degrau do Motor B.	94
Figura 49 – Pontos para sintonia do PID do Motor A	95
Figura 50 - PID via AMIGO - Motor A.....	96
Figura 51 - Pontos para sintonia do PID do Motor B	98
Figura 52 - PID via AMIGO - Motor B.....	99
Figura 53 - Controlador <i>Fuzzy</i> e Controladores PID	100
Figura 54 - Resultado da odometria para o círculo.	102

LISTA DE TABELAS

Tabela 1 - Regras de sintonia AMIGO.	42
Tabela 2 - Parâmetros do motor.....	53
Tabela 3 - Especificações da câmera.	60
Tabela 4 - Conjunto de regras do <i>Fuzzy</i>	89
Tabela 5 - Parâmetros de desempenho Motor A.....	92
Tabela 6 - Parâmetros de desempenho motor b.	94
Tabela 7 - Resultados da sintonia AMIGO – Motor A.....	96
Tabela 8 - Parâmetros do PID motor A.	97
Tabela 9 - Resultados da sintonia AMIGO – Motor B.....	98
Tabela 10 - Parâmetros de desempenho PID motor B.....	99

LISTA DE ABREVIATURAS E SIGLAS

AGVs	<i>Automated Guided Vehicle</i> (Veículo Guiado Automaticamente)
AMIGO	<i>Approximate MIGO</i> (MIGO Aproximado)
CC	Corrente Continua
CCD	<i>Charge-coupled device</i> (Dispositivo de Carca Acoplada)
CLPs	Controladores Lógicos Programáveis
CMOS	<i>Complementary metal-oxide-semiconductor</i> (Semicondutor de Metal-Óxido Complementar)
CSI	<i>Camera Serial Interface</i> (Interface Serial para Câmeras)
FET	<i>Field Effect Transistor</i> (Transistor de Efeito de Campo)
FPS	<i>Frames Per Second</i> (Quadros Por Segundo)
GPIO	<i>General Purpose Input Output</i> (Entra e Saída de uso geral)
Li-Po	Lithium Polymer (Polímero de Lítio)
MIGO	<i>M-constrained integral gain optimization</i> (Otimização do ganho integral por limitação da máxima sensibilidade)
Mneg	Muito negativa
Mpos	Muito positivo
PD	Proporcional Derivativo
PI	Proporcional Integral
PID	Proporcional Integral Derivativo
Pneg	Pouco negativo
Ppos	Pouco positivo
PPR	Pulso por revolução
PWM	<i>Pulse-Width Modulation</i> (Modulação por Largura de Pulso)
RAM	<i>Random Access Memory</i> (Memória de Acesso Aleatório)
RPM	Revoluções por minuto

LISTA DE SÍMBOLOS

A	Acelerado
C	Comprimento da circunferência da roda do robô
$C1, C2$	Classes de pixels
D	Devagar
dr	Diâmetro da roda do robô
$e(t)$	Erro do sistema
epa	Erro de pixel anterior
ep	Erro de pixel
E	Força contra eletromotriz
$f(x, y)$	Função que representa uma determinada imagem
$f_a(x)$	Função característica conjunto clássico
$Gauss(x, y)$	Função Gaussiana
$gaussmf$	Função de pertinência gaussiana
i	Pixels
K_p	Ganho proporcional ou constante proporcional
K_i	Ganho integral ou constante integrativa
K_d	Ganho derivativo ou constante derivativa
K_{cr}	Ganho crítico
k_t	Constante de torque
K	Ganho estático
l	Ponto de Corte (Limiar)
MD	Muito devagar
MA	Muito acelerado

M_p	Média ponderada
μ_1, μ_2	Média das classes
μ_t	Média total da imagem
$\mu_A(x)$	Grau de pertinência de x ao conjunto <i>Fuzzy A</i>
P_{cr}	Período crítico
$P(i)$	Quantidade de pixels i dividida pelo total de pixels.
P_m	Pixel médio
P_c	Pixel central
R	Raio
$r(t)$	Referência (<i>setpoint</i>)
$sgmf$	Função de pertinência sino generalizada
$trmf$	Função de pertinência triangular
$trapmf$	Função de pertinência trapezoidal
τ_i	Tempo integrativo
τ_d	Tempo derivativo
τ	Constante de tempo
T	Torque
$u(t)$	Saída do controlador
v	Velocidade linear
VM	Velocidade média
V_{RE}	Velocidade linear roda esquerda
V_{RD}	Velocidade linear roda direita
$V_{saída}$	Tensão de saída PWM
$y(t)$	Saída do sistema

ω	Velocidade angular do motor
σ	Desvio Padrão
σ^2	Variância das classes de pixels
θ	Tempo morto do sistema
ϕ	Fluxo magnético
Δe	Desvio do erro de pixel

SUMÁRIO

1	INTRODUÇÃO	20
1.1	<i>Objetivos gerais</i>	22
1.2	<i>Objetivos específicos</i>	22
1.3	<i>Organização do texto</i>	22
2	REFERÊNCIAL TEÓRICO	24
2.1	<i>Robótica móvel</i>	24
2.2	<i>Visão Computacional</i>	26
2.2.1	Captura da imagem	27
2.2.2	Processamento da imagem	29
2.2.2.1	Filtragem Gaussiana	29
2.2.2.2	Segmentação	30
2.2.2.3	Binarização	31
2.2.2.4	Método de Otsu	32
2.3	<i>Lógica Fuzzy</i>	33
2.3.1	Conjunto Fuzzy	34
2.3.2	Variável linguística e conjunto de termos	35
2.3.3	Função de Pertinência	35
2.3.4	Fuzzificação	37
2.3.5	Conjunto de regras	37
2.3.6	Base de dados	37
2.3.7	Inferência	38
2.3.8	Defuzzificação	38
2.4	<i>Controle PID</i>	39
2.4.1	Controlador Proporcional, Integral e Derivativo (PID)	39

2.4.2	Sintonia de controladores PID	40
2.4.2.1	Método de sintonia AMIGO	41
2.4.3	PID Digital.....	43
2.5	<i>Sensores e atuadores</i>	44
2.5.1	Encoders.....	44
2.5.2	Motores de Corrente Contínua	45
2.6	<i>Modulação por Largura de Pulso</i>	46
2.7	<i>Modelo cinemático de um robô com tração diferencial</i>	46
3	MATERIAIS E MÉTODOS	50
3.1	<i>Metodologia</i>	50
3.2	<i>Construção do AGV</i>	51
3.2.1	Estrutura	51
3.2.2	Motores e caixa de redução.....	52
3.2.3	Encoders de quadratura	54
3.2.4	Rodas	55
3.2.5	Ponte H.....	56
3.2.6	Raspberry Pi	58
3.2.7	Câmera	60
3.2.8	Bateria	61
3.2.9	Conversor CC abaixador (Buck):	62
3.3	<i>Visão computacional</i>	63
3.4	<i>Controlador Fuzzy</i>	66
3.5	<i>Controlador PID</i>	73
3.6	<i>Cálculos da odometria</i>	74

3.7	<i>Softwares e bibliotecas</i>	75
4	RESULTADOS E DISCUSSÕES	77
4.1	<i>Estrutura e montagem do robô</i>	77
4.2	<i>Conferência dos valores nominais dos Motores.</i>	79
4.3	<i>Visão computacional e processamento de imagem</i>	81
4.4	<i>Controlador Fuzzy</i>	88
4.5	<i>Controladores PID</i>	91
4.6	<i>Controlador Fuzzy e controladores PID em conjunto</i>	100
4.7	<i>Odometria.</i>	101
5	CONSIDERAÇÕES FINAIS	103
	REFERÊNCIAS	106

1 INTRODUÇÃO

A utilização de robôs na indústria e no dia a dia vem crescendo nos últimos anos devido à necessidade de diminuir o tempo de produção, ao elevado custo da mão de obra em certos setores e à diminuição dos custos de produção dos robôs. A diminuição dos custos dos dispositivos utilizados na construção de robôs e o surgimento de novos dispositivos também vêm possibilitando a criação de robôs cada vez mais sofisticados, precisos e eficientes, capazes de atuar em terrenos de diferentes complexidades (Romero et al, 2017).

Em especial, a robótica móvel vem ganhando destaque nos dias atuais com o surgimento de carros autônomos, robôs domésticos, que conseguem mapear e locomover em casas e escritórios fazendo a limpeza do ambiente, AGVs (*Automated Guided Vehicle* – Veículo Guiado Automaticamente), usados no setor industrial, entre outros.

Os AGVs são robôs aplicados em diversos setores em que se faz necessário o transporte de cargas dentro de uma linha de produção, podendo ser utilizados para buscar peças em estoque ou guardar um determinado produto depois de pronto.

Os AGVs utilizam linhas ou marcações no chão das industriais para se guiarem. Por esse motivo, estes robôs também são conhecidos como robôs industriais seguidores de linha. Para detectar as marcações no chão os AGVs comuns utilizam, geralmente, um sistema composto por sensores ópticos ou magnéticos.

Para Romero et al. (2017), um AGV comum é incapaz de detectar objetos presentes em seu caminho, sendo necessário a instalação de outros sensores para que possam detectar objetos a sua frente.

Em relação ao sistema utilizado para detecção de linhas, a aplicação de visão computacional permite o desenvolvimento de estratégias que visam detectar curvas, obstáculos e trechos apagados no trajeto do robô e assim diminuir ou aumentar a velocidade no percurso.

Além dos sistemas para identificar as marcações no chão de fábrica, para que os AGVs possam ser utilizado de forma produtiva nas indústrias, se faz necessário a adoção de controladores que garantam que os robôs se orientem de forma correta e

mantenham o controle de velocidade e direção por todo o trajeto previamente estipulado.

Baseado nestes fatos, este trabalho tem como objetivo estudar, desenvolver e implementar controladores de direção e velocidade para um robô móvel seguidor de linha, tendo como entrada imagens da trajetória.

Para a captura da trajetória a ser seguida pelo robô foi desenvolvido um algoritmo de visão computacional para captura das imagens, processamento e detecção do centro da trilha a ser seguida.

A saída do algoritmo de visão computacional é utilizada como entrada de um controlador *Fuzzy* desenvolvido para receber os dados de visão e informar a direção que o robô deve tomar para se manter alinhado com a trajetória. O controlador *Fuzzy* retorna a velocidade que as rodas do motor devem girar para que o mesmo siga em linha reta ou faça curvas. A saída do controlador *Fuzzy* é fornecida para controladores PID.

Os controladores PID recebem as informações do controlador *Fuzzy* como valores de referência e são os responsáveis por manter a velocidade das rodas em um valor desejável, para que o robô mantenha a trajetória com a menor oscilação possível.

Para a validação dos controladores e do sistema de visão foi construído um pequeno robô móvel com tração diferencial. O robô utiliza o *Raspberry Pi* como sistema de controle principal e uma ponte H dupla para o controle individual dos motores, assim como uma câmera para aquisição de imagens.

O percurso de validação utilizado pelo robô móvel foi um círculo com raio de 70cm feito com auxílio de um pincel para quadro branco.

Por fim, os pontos coletados foram utilizados por um sistema de odometria para recriar o percurso realizado pelo robô e comparar com o trajeto desenhado para ter uma estimativa do desempenho do controlador.

1.1 Objetivos gerais

O objetivo geral deste trabalho é estudar, desenvolver e implementar controladores de direção e velocidade para um robô móvel seguidor de linha, tendo como entrada um sistema de visão computacional que capture as imagens da trajetória e informe o centro da trilha. Para a realização dos objetivos gerais se faz necessário o cumprimento dos seguintes objetivos específicos.

1.2 Objetivos específicos

Os objetivos específicos deste trabalho são listados nos tópicos abaixo:

- Construção da estrutura física, mecânica e eletrônica do robô móvel (AGV).
- Configuração do *Raspberry PI*, câmeras e outros periféricos.
- Aplicação de técnicas de visão computacional e processamento de imagem para detectar linhas e extrair informações das imagens processadas.
- Acionamento dos motores via Ponte H pelo *Raspberry PI*.
- Desenvolvimento do controlador *Fuzzy* para receber as saídas da etapa de visão e controlar a trajetória do robô.
- Desenvolver e sintonizar controladores PID para receber as saídas do *Fuzzy* e controlar a velocidade dos motores.
- Desenvolvimento de um sistema de odometria para capturar os pontos percorridos pelo robô como forma de validação de percurso.

1.3 Organização do texto

Este trabalho é organizado do seguinte modo: No capítulo 2, é apresentado o referencial teórico que norteia o desenvolvimento da dissertação. Neste capítulo são apresentadas as noções básicas de um robô móvel com tração diferencial; as etapas de processamento de imagem e visão computacional; os controladores *Fuzzy* e PID; as técnicas de sintonia de controladores PID; o modelo cinemático de um robô com tração diferencial.

No capítulo 3, são apresentados os materiais utilizados para a construção do robô móvel e a metodologia utilizada nas etapas de visão computacional e dos controladores *Fuzzy* e PID.

No capítulo 4, são apresentados os resultados obtidos nas principais etapas de controle do robô móvel.

Por fim, o capítulo 5 é reservado para a apresentação das conclusões obtidas em cada etapa e a indicação de trabalhos futuros que podem ser originados a partir desta dissertação.

2 REFERÊNCIAL TEÓRICO

Neste capítulo são apresentadas as informações coletadas durante o levantamento bibliográfico realizado para este trabalho. Inicialmente são apresentadas algumas noções básicas sobre os robôs móveis incluindo os AGVs. Posteriormente, as etapas da visão computacional e processamento de imagem são apresentadas de acordo com as necessidades desta dissertação. Em seguida, a teoria básica em volta dos controladores *Fuzzy* e PID são detalhadas. Por fim, são apresentadas noções sobre encoders, motores de corrente contínua e modulação por largura de pulso.

2.1 Robótica móvel

As aplicações da robótica nos dias atuais são vastas. Os robôs podem ser utilizados em vários setores, desde ambientes de manufatura, diversão, aplicações subaquáticas e espaciais, ou para ajudar deficientes (Niku, 2017).

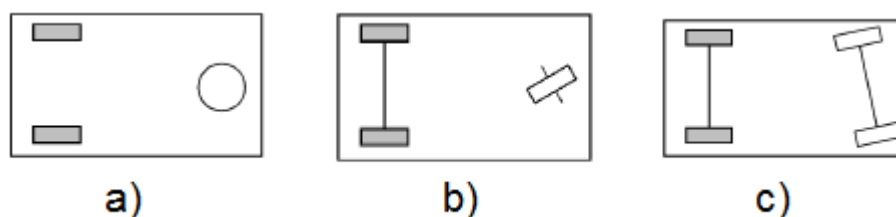
Segundo Niku (2017), a robótica é interdisciplinar, envolve as engenharias, a mecânica, elétrica, eletrônica, ciência da computação e muitas outras áreas. Dessa forma, é possível encontrar diversos tipos de robôs para diversas aplicações, como por exemplo, a utilização de robôs móveis na indústria.

Neste setor, os robôs são ferramentas poderosas, sendo capazes de realizar tarefas e operações com precisão, sem a necessidade de usar Equipamentos de Proteção Individual (EPIs) e equipamentos de conforto que um ser humano requer.

Segundo Romero et al. (2017), os robôs móveis são caracterizados por sua capacidade de deslocamento, podendo ser de modo guiado, semiautônomo ou totalmente autônomo.

Na literatura existem diversas topologias de estruturas para robôs moveis terrestres, muitas utilizadas por AGVs, que se aplicam a diversas situações. Silva (2010), em seu trabalho, ilustra várias estruturas possíveis de AGVs, sendo as mais alinhadas com a proposta deste trabalho destacadas na Figura 1.

Figura 1 – Algumas estruturas de AGVs.



Fonte: Adaptado de Silva (2010).

Segundo Silva (2010), a Figura 1 a) representa um robô móvel com dois motores traseiros independentes, responsáveis pela tração e direção do robô e um apoio omnidirecional (todas as direções) na frente. Este modelo utiliza tração diferencial para se locomover. Já a Figura 1 b), representa um AGV com tração traseira com um único motor e uma roda livre na frente responsável pela direção do robô. Por último, tem-se a Figura 1 c), representando um robô com tração no eixo traseiro e um eixo dianteiro com duas rodas responsável pela direção.

Apesar de existirem outros modelos, os apresentados na Figura 1 são os mais referenciados na literatura, por exemplo, Kodagoda et al (2002), desenvolveu e implementou em seu trabalho, controladores de direção e velocidade *Fuzzy PD-PI* para um AGV baseado em um carro de golfe seguindo a topologia da Figura 1 c). Já Butdee et al (2008), propôs um modelo baseado no descrito na Figura 1 b), no entanto, seu trabalho é composto por um AGV de três rodas, sendo a roda dianteira responsável pela tração e direção do robô, e as rodas traseiras são livres e atuam como apoio.

Segundo Silva (2010), o modelo com tração diferencial é, entre os modelos de AGVs apresentados, o com maior facilidade de controle de direção, não sendo necessário o controle de direção pela inclinação do eixo da roda dianteira. No entanto, o modelo da Figura 1 a), pode consumir mais energia que os outros dois modelos, devido a utilização de dois motores responsáveis pela tração-direção.

Neste trabalho o AGV experimental é baseado em tração diferencial. O robô também utiliza técnicas de visão computacional. Algumas etapas da etapas do sistema de visão computacional são apresentadas na seção 2.2.

2.2 Visão Computacional

Segundo Gonzalez e Woods (2010), pode se definir como imagem uma função bidimensional, $f(x,y)$, em que x e y são coordenadas espaciais e a intensidade da imagem é a amplitude de f em qualquer par de coordenadas. Uma imagem é considerada digital se os valores de intensidade de f são finitos e discretos.

Para Gonzalez e Woods (2010), não existe um consenso geral entre os autores em relação até que ponto vai o processamento de imagens e em que ponto começa outras áreas como a visão computacional. Já para Backes e Sá Junior (2016), a visão computacional pode ser definida como a área de estudo que tenta repassar para máquinas a capacidade da visão.

Gonzalez e Woods (2010) atribui como processamento digital de imagens, processos que tenham como entrada e saída imagens digitais, processos de extração de atributos de imagens e o reconhecimento de objetos. Já a etapa de dar sentido ao conteúdo de uma imagem pode ser considerado como visão computacional, dependendo da complexidade da tarefa.

De acordo com Gonzalez e Woods (2010), a visão computacional é uma área da inteligência artificial e tem como objetivo emular a visão humana utilizando computadores. O campo da visão computacional inclui a capacidade de máquinas realizarem inferências e agirem com base em informações visuais.

De acordo com Backes e Sá Junior (2016), um sistema de visão computacional pode ser resumida nos seguinte conjuntos de fases: captura da imagem; pré-processamento da imagem; filtragem; segmentação; extração de características e o reconhecimento de padrões.

Muito embora, como afirma Backes e Sá Junior (2016), essa sequência de fases não é um consenso entre os autores da área, e algumas dessas fases podem ser suprimidas de acordo com a tarefa a ser realizada.

A primeira fase do sistema de visão computacional é a captura da imagem. Essa fase é apresentada na seção 2.2.1.

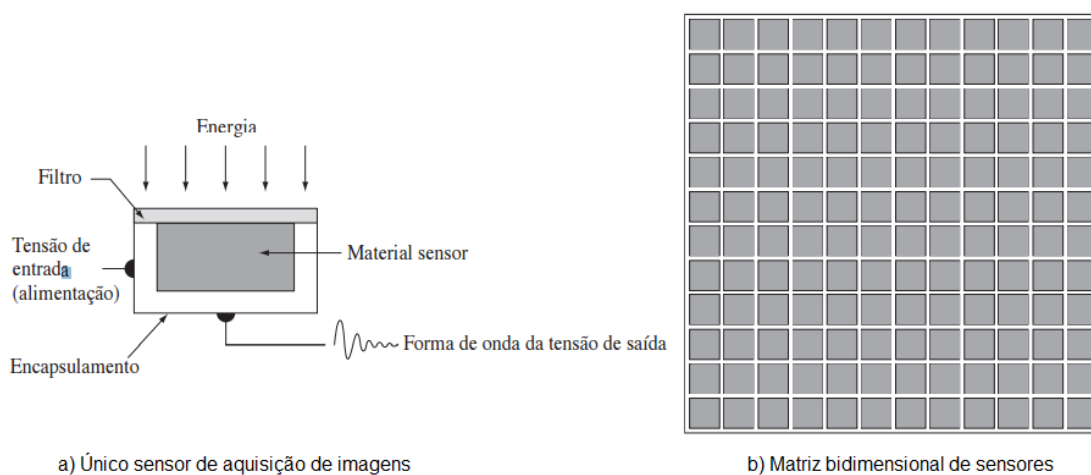
2.2.1 Captura da imagem

A aquisição ou captura de imagens digitais é realizada via dispositivos eletrônicos como câmeras fotográficas e filmadoras. Estes dispositivos são equipados com sensores capazes de discretizar a imagem em pixels. Cada pixel recebe um valor numérico com o objetivo de representar as gradações de tonalidade da imagem (Backes e Sá Junior, 2016).

As imagens são geradas pela combinação de uma fonte de energia e a reflexão ou absorção de parte dessa energia pelos elementos de uma cena ou objeto de interesse, cuja imagem está sendo gerada (Gonzalez e Woods, 2010).

A Figura 2 a) mostra o funcionamento de um único sensor utilizado na captura de imagens. O sensor mais conhecido nessa aplicação é o fotodiodo, material semicondutor que permite a passagem de corrente elétrica de forma proporcional à intensidade luminosa que o atinge.

Figura 2 - Sensor utilizado na captura de imagens.



Fonte: Gonzalez e Woods (2010).

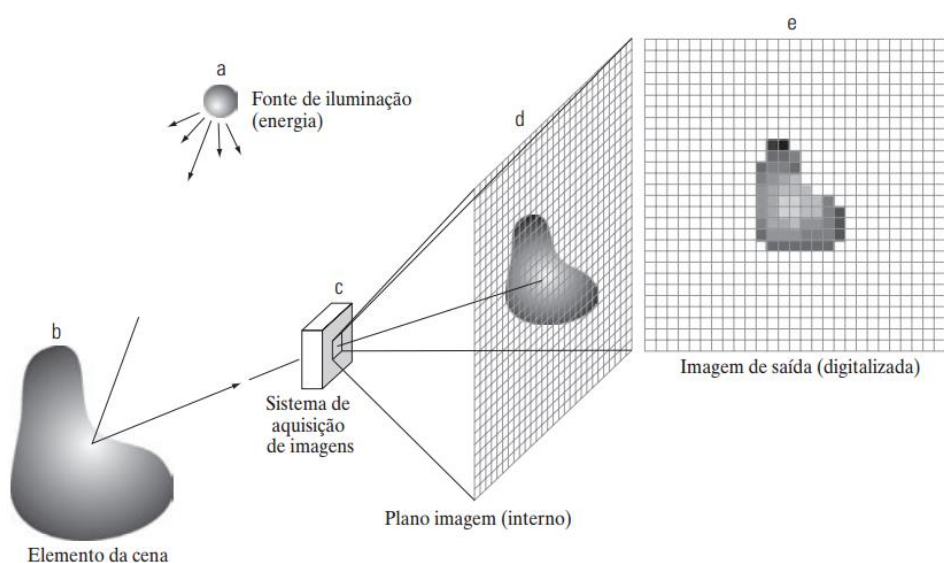
A Figura 2 b) mostra uma matriz bidimensional com vários sensores individuais. O agrupamento em forma matricial é o mais utilizado em dispositivos para aquisição de imagens, como filmadoras e câmeras digitais.

A matriz CCD (*charge-coupled device*) é um dos sensores típicos utilizados em câmeras digitais, estes sensores são compostos por arranjos matriciais de 4000x4000

ou mais células fotoelétricas. A resposta de cada célula é proporcional a integral da energia luminosa projetada sobre as mesmas (Gonzalez e Woods, 2010).

A Figura 3 mostra como os sensores de aquisição de imagem digitais são utilizados.

Figura 3 - Processo de captura de imagem digital.



Fonte: Gonzalez e Woods (2010).

A energia luminosa a) é refletida de um objetivo de interesse b) e captada pelo sistema de aquisição c) composto por uma lente ótica que projeta a cena sobre um plano focal d). O sensor matricial coincide com o plano focal, ou seja, a energia é captada por cada sensor da matriz produzindo saídas proporcionais à integral da luz incidida. Por fim, um conjunto de circuitos digitais é responsável por amostrar e quantizar o sinal analógico da matriz e transformar a saída em uma imagem digital e) (Gonzalez e Woods, 2010).

Após a captura da imagem é necessário a aplicação de algumas técnicas de processamento digital para corrigir e extrair as informações necessárias. A seção 2.2.2 apresenta algumas destas técnicas.

2.2.2 Processamento da imagem

Após a captura da imagem é necessário efetuar o seu processamento para adequar a imagem e facilitar sua manipulação nas etapas seguintes que compõem a visão computacional.

De acordo com Niku (2017), o processamento de imagem diz respeito às técnicas de preparação para a posterior análise e utilização da mesma, pois as imagens após serem capturadas, geralmente, não estão com as características adequadas para análise.

A fase de processamento da imagem é responsável por definir a resolução, rotação, escala de cor da imagem, como também a aplicação de filtros para diminuir ruídos, salientar bordas, entre outros.

Como o universo do processamento de imagem é extenso, a seguir serão detalhadas somente as técnicas de interesse deste trabalho.

2.2.2.1 Filtragem Gaussiana

Um dos principais problemas de uma imagem digital são os ruídos, geralmente causados na etapa de digitalização, mas também podem ser causados durante a aquisição da imagem, na etapa de transmissão da imagem, na variação do brilho, pouca iluminação e até por altas temperaturas. Para diminuir ou eliminar os ruídos foram criadas algumas técnicas ou filtros de suavização, como o filtro Gaussiano. Um dos objetivos dos filtros é reduzir o ruído e preparar as imagens para outras etapas de processamento, como por exemplo, a segmentação (Sanches et al, 2015).

De acordo com Gonzalez e Woods (2010), filtros de suavização além de serem utilizados para a remoção de pequenos detalhes de uma imagem, também podem ser úteis para fazer a correção de pequenas discontinuidades em linhas ou curvas.

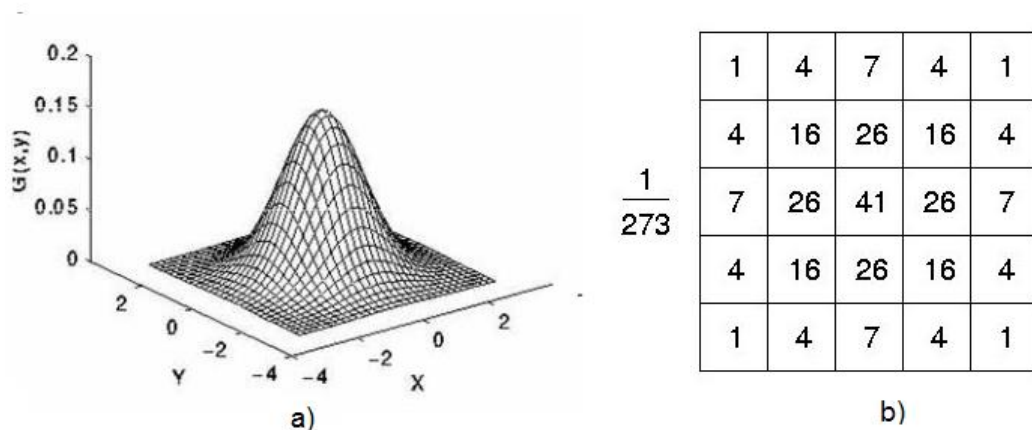
O filtro Gaussiano é útil em diferentes áreas do processamento de imagens. Este filtro é geralmente utilizado como um filtro de suavização, ou seja, um filtro passa-baixa, deixando passar as baixas frequências e eliminando as frequências mais altas. O filtro Gaussiano leva esse nome devido os valores de suas máscaras serem determinados a partir de uma função bidimensional Gaussiana discreta, de média zero

e desvio padrão (σ) geralmente igual a 1. A Equação 1 representa a função Gaussiana bidimensional.

$$Gauss(x, y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{(x^2+y^2)}{2\sigma^2}} \quad (1)$$

Na Equação 1, quanto maior for o desvio padrão, maior será a largura do filtro Gaussiano e também o grau de suavização (Sanches et al, 2015). A Figura 4 a) mostra a representação gráfica em forma de sino da função Gaussiana bidimensional com média em (0,0) e desvio padrão igual a 1 ($\sigma = 1$).

Figura 4 - Representação da função Gaussiana bidimensional com $\sigma = 1$.



Fonte: a) Jesus e Costa (2015); b) Sanches et al (2015).

A Figura 4 b) representa a forma discreta aproximada da função Gaussiana bidimensional com desvio padrão igual a 1. A Figura 4 b) é também conhecida como máscara de suavização. O grau de suavização depende do tamanho da máscara, sendo maior a suavização em máscaras maiores (Sanches et al, 2015).

Outra técnica de processamento digital muito utilizada é a segmentação, apresentada na seção 2.2.2.2.

2.2.2.2 Segmentação

Segundo Niku (2017), segmentação é um nome genérico para descrever um conjunto de técnicas diferentes que visam dividir uma determinada imagem em segmentos. O principal objetivo da segmentação é separar a informação contida em uma única imagem em parcelas menores que possam ser utilizadas para outros fins.

Para Gonzalez e Woods (2010), a precisão da segmentação determina o sucesso ou o fracasso nas etapas futuras de análise computadorizada.

De acordo com Niku (2017), técnicas de detecção de bordas, crescimento de regiões e análise de textura estão inclusas na segmentação, mas não a limitam.

2.2.2.3 Binarização

Para Monteiro (2018), a binarização é o método mais simples de segmentação de imagens, pois consiste em dividir uma imagem em regiões de interesse ao escolher um ponto de corte ou limiar (*threshold*).

Devido sua simplicidade de implementação, propriedades intuitivas e a velocidade computacional, a binarização de imagens tem uma posição de destaque nas aplicações de segmentação de imagens (Gonzalez e Woods, 2010).

A binarização é um processo de segmentação de imagens baseado na diferença dos níveis de cinza que constituem os objetos de uma imagem. A imagem pode ser segmentada em dois grupos a partir de um limiar escolhido com base nas características dos objetos que se deseja isolar. Dessa forma, é obtido um grupo de pixels com níveis de cinza abaixo do limiar e outro com pixels com níveis acima do limiar.

Segundo Backes e Sá Junior (2016), a binarização consiste em estabelecer um limiar e converter os pixels da imagem maiores que o limiar para 1 e os demais para 0, ou seja, seria obtido uma matriz composta por 1's e 0's, daí o nome binarização.

A binarização pode ser representada matematicamente pela Equação 2:

$$f(x, y) = \begin{cases} 1, & f(x, y) > l \\ 0, & f(x, y) \leq l \end{cases} \quad (2)$$

em que $f(x, y)$ é a função que representa uma determinada imagem e l é o ponto de corte estabelecido (Backes e Sá Junior, 2016).

Para Gonzalez e Woods (2010), a binarização é global quando l é aplicável a uma imagem inteira, ou a binarização é variável quando l muda ao longo da imagem.

A determinação incorreta do ponto de corte pode dificultar a obtenção de bons resultados com a técnica de binarização, ou seja, para que a segmentação por

binarização consiga separar corretamente os elementos desejados é necessário uma escolha adequada do ponto de corte pelo usuário.

Existem métodos que permitem que a escolha deste ponto de corte não dependa exclusivamente do usuário, uma delas é o Método de Otsu. Neste método o limiar é determinado pelas características intrínsecas da imagem.

2.2.2.4 Método de Otsu

Segundo Backes e Sá Junior (2016), o sucesso da binarização está na escolha correta de um limiar. No entanto, quando a escolha do limiar fica limitada à escolha do usuário, as chances de um limiar incorreto aumentam. Em situações onde as imagens ou o ambiente mudam com o tempo, a escolha manual de um limiar se torna impraticável. Uma forma de resolver este problema é a utilização de métodos que permitam escolher o limiar de forma automática. O método de Otsu (1979) é um destes métodos que visam encontrar um limiar adequado de forma automática.

O método de Otsu visa encontrar um limiar que separe a imagem em duas classes, C_1 e C_2 , de pixels minimizando e maximizando a variância entre os grupos simultaneamente. As classes são o objeto e o fundo da imagem (Artero, 1999).

A primeira classe possui valores de pixels entre $[0, 1, 2, 3, \dots, l]$ e a outra classe valores entre $[l + 1, l + 2, \dots, G]$, onde G é a maior intensidade de pixel e l é o limiar (Backes e Sá Junior, 2016).

Para cada classe é calculada a porcentagem de pixels utilizando as Equações 3 e 4.

$$C_1 = \sum_{i=0}^l P(i) \quad (3)$$

$$C_2 = \sum_{i=l+1}^G P(i) \quad (4)$$

$P(i)$ representa a quantidade de pixels i dividida pelo total de pixels (Backes e Sá Junior, 2016). Já a média das classes são representadas pelas Equações 5 e 6

$$\mu_1 = \sum_{i=0}^l \frac{i \cdot P(i)}{C_1} \quad (5)$$

$$\mu_2 = \sum_{i=l+1}^G \frac{i \cdot P(i)}{C_2} \quad (6)$$

A média total da imagem é dada pela Equação 7:

$$\mu_t = \frac{C1}{\mu_1} + \frac{C2}{\mu_2} \quad (7)$$

Já a variância entre as classes é dada pela Equação 8:

$$\sigma^2 = C1.(\mu_1 - \mu_t)^2 + C2.(\mu_2 - \mu_t)^2 \quad (8)$$

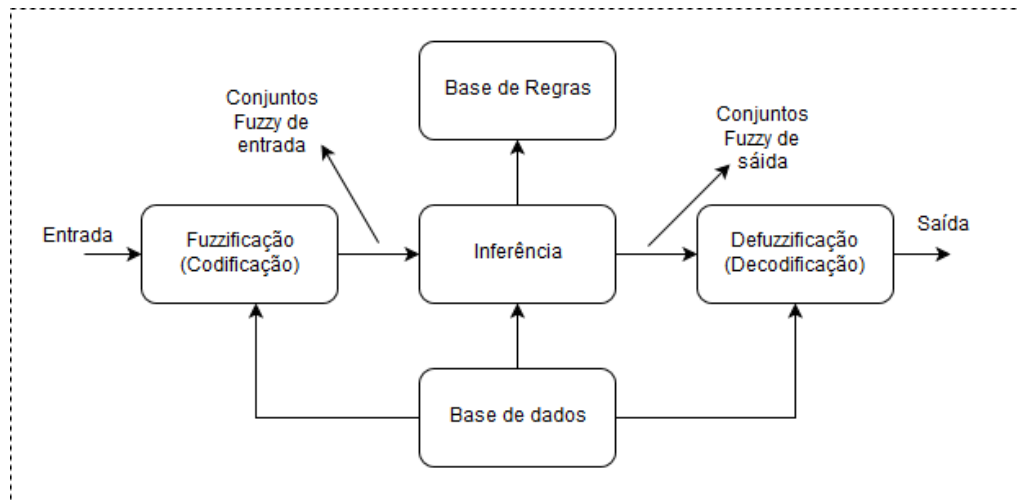
Com base nestas equações o método de Otsu selecionará o limiar ideal l que maximiza a equação 8 (Backes e Sá Junior, 2016).

Neste trabalho, as informações obtidas com a visão computacional são utilizadas por um controlador *Fuzzy*. Um resumo da teoria envolvendo Lógica *Fuzzy* é apresentado na seção 2.3.

2.3 Lógica *Fuzzy*

Segundo Niku (2017), Gonzalez e Woods (2010), a ideia de controle utilizando lógica *Fuzzy* começou com um artigo publicado por Latfi Zadeh em 1965. Para Gomide e Gudwin (1994), a lógica *Fuzzy* é a base para o desenvolvimento de métodos de modelagem e controle que permitem a redução da complexidade e a solução de problemas de controle até então não solucionados por técnicas clássicas.

Um controlador baseado em Lógica *Fuzzy* é composto basicamente dos seguintes elementos: Base de Dados; Fuzzificação; Inferência; Conjunto de Regras; e Defuzzificação. A Figura 5 ilustra a sequência de passos de um controlador baseado em Lógica *Fuzzy*.

Figura 5 - Sistema *Fuzzy*

Fonte: Autoria própria.

A seção 2.3.1 apresenta a diferença entre a teoria clássica e a lógica *Fuzzy*, etapa necessária para o entendimento da Figura 5.

2.3.1 Conjunto *Fuzzy*

Na teoria clássica de conjuntos, um elemento x pertence ou não a um conjunto A em um universo de interesse U . Dessa forma o grau de pertinência do elemento pode ser representado pela função característica dada pela Equação 9 (Gomide e Gudwin, 1994).

$$f_a(x) = \begin{cases} 1 & \text{Se e somente se } x \in A \\ 0 & \text{Se e somente se } x \notin A \end{cases} \quad (9)$$

Já na lógica *Fuzzy*, segundo Gomide e Gudwin (1994), Zadeh propôs uma generalização da função característica de forma que o grau de pertinência possa assumir qualquer valor entre 0 e 1, aumentando seu poder de expressão.

$$A = \{\mu_A(x)/x\} \quad x \in U \quad (10)$$

onde: $\mu_A(x)$ é o grau de pertinência de x com o conjunto A , x é o elemento ou variável de interesse, A é o conjunto *Fuzzy* e U é o universo de discurso.

Uma das principais características da Lógica *Fuzzy* é a utilização de variáveis e termos linguísticos, detalhados na seção 2.3.2.

2.3.2 Variável linguística e conjunto de termos

Uma variável linguística pode ser entendida como uma variável que assume como valores um conjunto de termos linguísticos (Gomide e Gudwin, 1994). Por exemplo, a variável linguística “Altura” pode assumir valores do tipo “Baixa”, “Média” e “Alta”. Os termos linguísticos “Baixa”, “Média” e “Alta” são representados por funções de pertinência. Cada termo linguístico é considerado um conjunto *Fuzzy*.

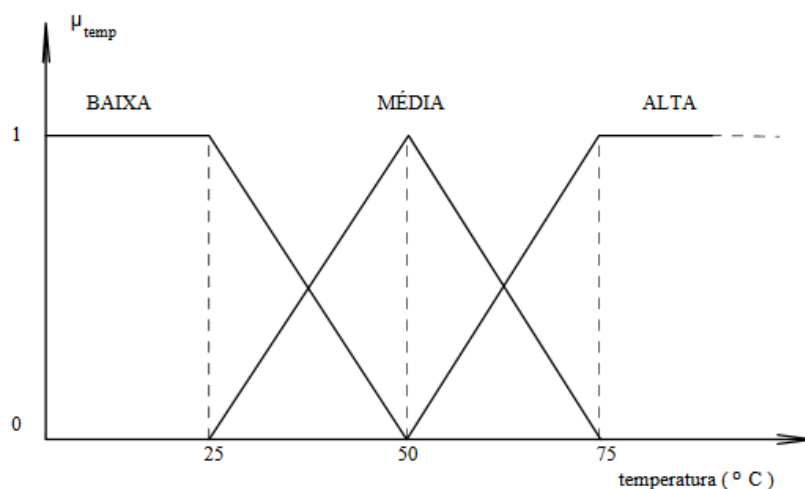
Segundo Tanscheit (2018), a principal característica das variáveis linguísticas é a possibilidade de caracterização aproximada de sistemas complexos, que não podem ser analisados por termos matemáticos convencionais.

2.3.3 Função de Pertinência

As funções de pertinência podem assumir valores entre o intervalo $[0,1]$ para indicar o grau de pertencimento de um elemento a um conjunto. O grau de pertinência na lógica *Fuzzy* pode variar de 0 a 1, ou seja, um elemento pode pertencer de 0 a 100% a um conjunto, diferente da lógica clássica em que o elemento pertence ou não ao conjunto. Um elemento também pode pertencer a mais de um conjunto *Fuzzy* com grau de pertinência diferente.

A Figura 6 mostra o exemplo da variável linguística “Temperatura” e seu conjunto de termos “Baixa”, “Média” e “Alta” representados por funções de pertinência que indicam o grau de pertencimento de um elemento a cada um destes conjuntos, levando em consideração o universo de discurso comum que vai de 0 a 100°C.

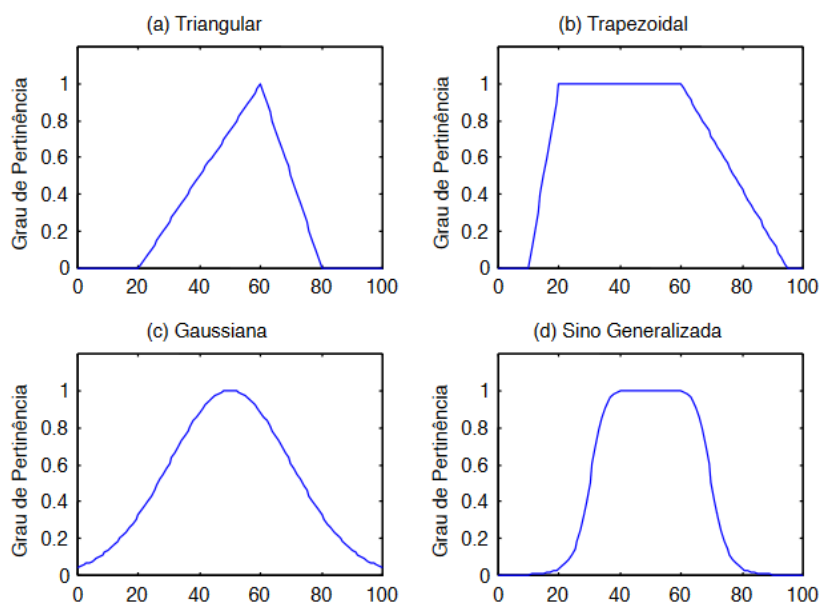
Figura 6 - Variável linguística “Temperatura”.



Fonte: Gomide e Gudwin (1994).

Como mostra a Figura 7, as funções de pertinência podem assumir formas diferentes. Existem várias funções de pertinência, porém as mais comuns são: Triangular; Trapezoidal, Gaussiana e Sino Generalizada. O tipo de função de pertinência é escolhido por um especialista com base em seu conhecimento sobre o comportamento do sistema. A Figura 7 mostra o formato de cada uma destas funções.

Figura 7 - Tipos de funções de pertinência.



Fonte: Neto (2018).

Após a apresentação das principais características da Lógica *Fuzzy* é possível detalhar o primeiro bloco da Figura 5, referente à Fuzzificação.

2.3.4 Fuzzificação

A Fuzzificação ou codificação é a etapa onde os valores reais de entrada são codificados ou convertidos para o conjunto *Fuzzy*. Após a Fuzzificação os valores de entrada são convertidos para um conjunto de equações que descrevem o grau de pertinência destes valores em relação aos termos de uma variável linguística (Niku, 2017).

A Fuzzificação escalona os valores reais de entrada para condicioná-los a universos de discursos normalizados, e os converte de modo que virem instâncias de variáveis linguísticas (Gomide e Gudwin, 1994).

A etapa de Fuzzificação engloba, além da definição das variáveis e funções de pertinência, a análise do problema e a criação das regiões.

2.3.5 Conjunto de regras

O conjunto de regras são sentenças com elementos lógicos que relacionam as entradas e saídas já fuzzificadas. O conjunto de regras assim como as funções de pertinência são escolhidas pelo especialista. As regras são construídas utilizando sentenças linguísticas baseadas nos conjuntos de termos de cada variável linguística de entrada e saída, como por exemplo:

Se “Erro” é “Zero” e “Desvio do Erro” é “Pequeno” então “Sistema” é “Estável”

Neste exemplo “Erro” e “Desvio do Erro” são variáveis linguísticas de entrada, “Sistema” é uma variável linguística de saída e “Zero”, “Pequeno” e “Estável” são os termos destas variáveis, os quais são representados por funções de pertinência.

2.3.6 Base de dados

A base de dados tem a função de armazenar todas as configurações e informações necessárias nas etapas de Fuzzificação, Inferência e Defuzzificação, assim como as definições das funções de pertinência e normalizações dos universos de discurso (Gomide e Gudwin, 1994).

2.3.7 Inferência

Na etapa de inferência é onde as regras são processadas de acordo com a estratégia adotada para as entradas e saídas. Na inferência é gerado o conjunto de saída, com base nas operações lógicas definidas nas regras.

A etapa de inferência engloba, além da definição das proposições, a análise das regras e a criação da região resultante. A inferência também tem a função de calcular o quanto uma regra é importante para o sistema.

Existem técnicas diferentes para encontrar a conclusão mais apropriada. Uma das mais utilizadas é a Mamdani (Max-Min). Neste método, ao ser utilizado o conectivo “E” no conjunto de regras, as entradas são combinadas utilizando o operador mínimo (Min), obtendo o menor valor como saída. Já se for utilizado o conectivo “OU” no conjunto de regras, as entradas são analisadas com o operador máximo (Max), onde o maior valor é obtido para a saída.

2.3.8 Defuzzificação

De forma geral, a defuzzificação ou decodificação é a etapa onde os valores de saída do conjunto *Fuzzy* são convertidos em valores reais para serem utilizados em diversas aplicações.

Na etapa de inferência, para cada entrada *Fuzzy* é criada uma saída *Fuzzy*, que tem como objetivo, indicar a ação a ser tomada pelo controle. No entanto, como a saída é *Fuzzy*, e não real, é necessário a adoção de algum método que permita converter a saída *Fuzzy* para um número real. Os métodos de defuzzificação mais conhecidos são: Centroide, Critério Máximo, Média do Máximo e Maior do Máximo.

Entre os métodos, o da centroide é o mais utilizado. Este método de defuzzificação, também conhecido como centro de gravidade, retorna a média das áreas das figuras que representam o grau de pertinência de um elemento no conjunto *Fuzzy* de saída. O método da centroide é representado pela Equação 11:

$$Y = \frac{\sum y_i \mu_{y_i}}{\mu_{y_i}} \quad (11)$$

onde Y é a saída com valores reais, μ_{y_i} é o grau de pertinência e y_i é a área das figuras que representam cada função de pertinência.

Neste trabalho, além do controlador *Fuzzy*, é necessário o emprego de controladores PID para o controle de velocidade das rodas do robô. A seção 2.4 apresenta um resumo da teoria envolvendo controle PID.

2.4 Controle PID

Entre as técnicas de controle, o algoritmo PID, abreviação de Proporcional-Integral-Derivativo, está entre os mais populares, tanto no meio acadêmico como industrial. Os controladores PID são encontrados nas mais variadas plantas industriais, sendo utilizados, muitas vezes, em várias etapas em um mesmo processo de fabricação. Para Åström e Hägglund (1995), mais de 95% das malhas de controle são do tipo PID ou variante.

De acordo com Åström e Hägglund (1995), o controlador PID possui várias funcionalidades importantes, como a capacidade de fornecer *feedback*, a habilidade de eliminar o erro de estado estacionário e antecipar ações futuras.

2.4.1 Controlador Proporcional, Integral e Derivativo (PID)

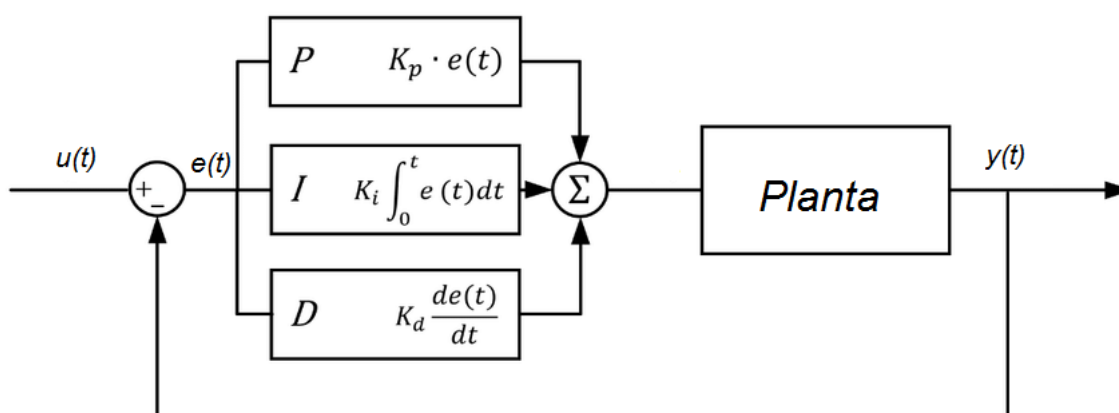
Como o nome sugere, o controlador PID é a soma de três coeficientes, sendo eles: proporcional, integral e derivativo, que podem ser variados para a obtenção de uma melhor resposta. A equação ideal do PID no domínio do tempo é dada pela Equação 12:

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{d}{dt} e(t) \quad (12)$$

onde $u(t)$ é a saída fornecida a uma planta, $e(t)$ é o erro, K_p é a constante proporcional, K_i é a constante integrativa e K_d é a constante derivativa.

Um diagrama de blocos representando as componentes de um controlador PID pode ser visto na Figura 8.

Figura 8 - Diagrama de blocos do controlador PID.



Fonte: Autoria própria.

De uma forma geral, a componente proporcional do controlador PID tende a acelerar o processo de forma controlada enquanto a componente integrativa corrige as oscilações geradas pela componente proporcional e a componente derivativa estabiliza o processo.

Além da combinação dos controladores citados, a grande popularidade deste controlador também se dá devido à sua robustez e à sua simplicidade funcional. De acordo com Åström e Hägglund (1995), nas diferentes etapas da geração e transmissão de energia o controle PID é frequentemente utilizado em conjunto com circuitos lógicos e outras tecnologias para a construção de estratégias mais complexas.

Para que os controladores PID funcionem corretamente, é necessário sua sintonia. Na seção 2.4.2 são apresentadas algumas técnicas de sintonia de controladores PID.

2.4.2 Sintonia de controladores PID

O processo de escolha dos parâmetros, ou ajuste dos ganhos dos controladores PID, é conhecido como sintonia ou *tuning* do controlador. Estes parâmetros podem ser encontrados em uma análise do modelo matemático do sistema. No entanto, nem sempre é possível, ou é viável, encontrar um modelo matemático que represente a dinâmica do sistema a ser controlado para que se possa encontrar os parâmetros de controle. Para estes casos, é necessário o uso de técnicas

que não dependem do modelo para sintonizar o controlador. Os métodos de Ziegler-Nichols são alguns dos modelos mais conhecidos e muito uteis quando não se conhece o modelo matemático da planta (Ogata, 2011).

Ziegler e Nichols (1942) propuseram a primeira metodologia simples e direta para sintonizar controladores PID, propondo dois métodos para sintonia de controladores PID.

O primeiro método de Ziegler-Nichols consiste em analisar a resposta em malha aberta do sistema ao ser aplicado uma entrada do tipo degrau unitário. Já o segundo método de Ziegler-Nichols ou método da sensibilidade limiar, consiste em analisar a resposta em malha fechada do sistema ao elevar o ganho proporcional até que o sistema mantenha a oscilação.

Os métodos de Ziegler e Nichols também serviram como base para o surgimento de vários outros métodos experimentais, como por exemplo: O método de Chien et al (1952), conhecido como método CHR, sendo aplicável para sistemas com razão de incontrolabilidade baixa ; o método de Cohen e Coon (1953), destinado para sintonia de sistema com atrasos de tempo elevado. No entanto, estes métodos apresentam algumas limitações que não facilitam ou permitem sintonizar alguns controladores PID.

Uma alternativa aos métodos de Ziegler e Nichols e derivados é o método AMIGO, utilizado neste trabalho e explicado na seção 2.4.5.1.

2.4.2.1 Método de sintonia AMIGO

O método MIGO (*M-constraint Integral Gain Optimization*) é um método computacional que tem como critério de otimização a maximização do ganho integral (K_i) do controlador, limitado pela máxima sensibilidade. Åström e Hägglund (2004), realizaram um serie de ensaios em vários processos industriais utilizando o método MIGO, a partir das correlações dos resultados destes ensaios os autores propuseram um conjunto de regras de sintonia aproximadas, conhecidas como AMIGO (*Approximate M-constraint Integral Gain Optimization*). Segundo Levine (2010), os parâmetros do controlador PID podem ser encontrados usando o método de sintonia AMIGO a partir da Tabela 1.

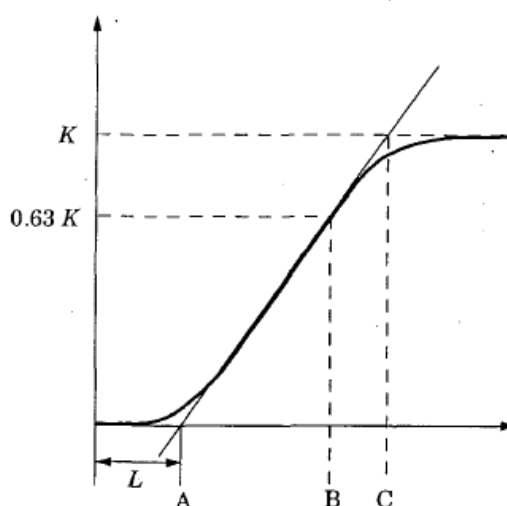
Tabela 1 - Regras de sintonia AMIGO.

Controlador	K_p	τ_i	τ_d
PID	$\frac{1}{K} \left(0,2 + 0,45 \cdot \frac{T}{L} \right)$	$L \cdot \frac{0,4L + 0,8T}{L + 0,1T}$	$\frac{0,5LT}{T + 0,3L}$

Fonte: Levine, (2010).

Os parâmetros no modelo podem ser determinados graficamente utilizando a Figura 9. T pode ser obtido de duas formas graficamente, a primeira levando em conta a distância entre AB e a segunda levando em conta a distância entre AC. Onde B é o ponto entre o valor de 63% de K e o eixo horizontal, e C é o ponto onde a tangente intercepta a linha referente ao valor de K.

Figura 9 - Curva de resposta ao degrau, parâmetros K,L e T.



Fonte: Åström e Hägglund (1995).

Segundo Åström e Hägglund (1995), os dois métodos para encontrar o valor de T fornecem resultados parecidos. No entanto, o método usando B, geralmente, entrega uma aproximação melhor, enquanto o outro tende a entregar um valor maior de T.

O controlador PID pode ser implementado de várias formas, sendo as principais: em circuitos eletrônicos à base de amplificadores operacionais, e circuitos microcontrolados ou microprocessados como os CLPs. Mas para isso é necessário a adaptação do controlador PID para os meios digitais.

2.4.3 PID Digital

O surgimento dos microprocessadores e microcontroladores tiveram uma influência muito importante no aprimoramento dos controladores PID. Praticamente todos os controladores PID feitos atualmente são baseados em microprocessadores e microcontroladores. Esse fato permitiu o desenvolvimento e a adição de recursos adicionais aos controladores, como por exemplo, a sintonia automática (Åström e Hägglund, 1995).

No entanto, para a implementação dos controladores em sistemas microcontrolados ou microprocessados é necessário que se faça a discretização do sistema em tempo contínuo.

O projeto de controladores PID discretos pode ser realizado por emulação ou diretamente no plano Z. A primeira opção permite projetar um controlador usando os resultados e equações para controladores analógicos (contínuo) e depois discretizá-los. Esse método tem a vantagem de dispensar um estudo mais detalhado de técnicas discretas. Já o projeto de um controlador diretamente no plano Z é mais preciso do que o método por emulação, uma vez que considera a adição de seguradores de ordem zero no sistema. No entanto, a segunda opção necessita de um estudo de maior complexidade.

Em Åström e Hägglund (1995) e Isermann (2013) são apresentadas algumas técnicas de aproximação para a discretização de sistemas PID contínuos e representá-los na forma discreta.

A equação de diferenças do controlador PID discreto pode ser encontrada utilizando a aproximação *backward* na parte derivativa e *tustin* na integrativa (Bertachi et al, 2013).

$$u[n] = u[n - 1] + kp * (e[n] - e[n - 1]) + k_i \frac{t_s}{2} * (e[n] - e[n - 1]) + \frac{k_d}{t_s} * (e[n] - 2 * e[n - 1] + e[n - 2]) \quad (13)$$

onde t_s é o período de amostragem.

A Equação 13 também é conhecida como PID discreto de velocidade. Dessa forma, o PID discreto fica mais próximo do PID contínuo quanto menor for o tempo de amostragem (Pinto, 2014).

Com base na Equação 13 é possível implementar os controladores PID em algoritmos escritos em linguagens de programação como: C, C++, entre outras. O que também facilita a implementação em microcontroladores, CLPs e outros dispositivos digitais.

2.5 Sensores e atuadores

Nesta seção é apresentado um resumo da teoria referente aos principais sensores e atuadores utilizados neste trabalho, iniciando pelos *encoders*.

2.5.1 Encoders

Os *encoders* ou odômetros – como também são conhecidos – são sensores bastante utilizados na indústria, em automóveis, robôs e na engenharia de forma geral. Estes sensores são utilizados para diversos fins como medição de rotação, posicionamento, mensurar distância, inclinação, entre outros. Na indústria são comumente empregados para o controle de velocidade de motores. Nos automóveis são utilizados para indicar a distância percorrida e a velocidade do veículo. Segundo Romero et al. (2017), estes sensores servem para controlar a posição e a velocidade das rodas de um robô móvel, permitindo estimar a distância relativa navegada.

Existem alguns tipos de *encoders*, os mais conhecidos são os *encoders* óticos e magnéticos. No entanto, o princípio de funcionamento destes sensores são bastante parecidos.

Os *encoders* óticos são compostos por componentes eletrônicos do tipo fotoemissor e fotorreceptor. Eles também possuem um disco plástico ou metálico com furações. Neste tipo de sensor a luz do fotoemissor é direcionada para o fotorreceptor. O disco é fixado em um eixo girante do dispositivo a ser analisado, como por exemplo, o eixo de um motor. O disco tem a função de interromper a passagem de luz do fotoemissor para o fotorreceptor. Essa sequência de interrupções tem o objetivo de gerar um sinal retangular para o sistema. A frequência deste sinal é diretamente proporcional à velocidade de giro do eixo. Segundo Romero et al. (2017), os *encoders* óticos utilizados na robótica móvel medem cerca de 2000 pulsos por revolução.

O princípio de funcionamento de um *encoder* magnético é o mesmo, só que no lugar do par fotoemissor-fotorreceptor é inserido um sensor de efeito hall e o disco

plástico ou metálico é substituído por um disco com vários ímãs, geralmente do tipo ímã de neodímio. Sempre que um dos ímãs passa pelo sensor de efeito hall um pulso é gerado para o sistema, dessa forma também é possível gerar um sinal retangular para o sistema. A vantagem deste tipo de *encoder* é que ele é menos suscetível às interferências causadas por sujeira que pode acabar bloqueando a passagem de luz no caso dos sensores óticos.

Dentro destes dois grupos de *encoders* existem os *encoder* de quadratura. Estes *encoders* são compostos por dois pares de fotoemissor-fotorreceptor ou dois sensores de efeito hall. O objetivo é obter dois sinais retangulares defasados em 90° um do outro. Estes sinais são chamados de canais, por exemplo, Canal A e Canal B. Dessa forma é possível saber o sentido da direção de giro. Por exemplo, é possível detectar se um robô está indo para frente ou para trás com este tipo de sensor.

Quando o *encoder* possui dois elementos óticos ou magnéticos, eles são chamados de odômetros de quadratura ou *encoders* de quadratura. Estes são os mais utilizados, pois também permitem saber o sentido de giro das rodas do robô.

2.5.2 Motores de Corrente Contínua

Os motores de corrente contínua (motores CC) são máquinas versáteis e são utilizados na indústria, em carros, brinquedos e em muitas outras aplicações. De acordo com Niku (2017), os motores CC por já serem utilizados no setor industrial há um longo tempo, são considerados confiáveis e resistentes.

Devido à facilidade com que podem ser controlados, os motores de corrente contínua são usados em diversas aplicações, principalmente em aplicações onde é exigido uma ampla faixa de velocidade e controle preciso da saída do motor (Fitzgerald et al., 2006).

Nos motores CC o estator pode ser feito de uma estrutura ferromagnética com polos salientes, onde as bobinas são enroladas ou pode ser feito a partir de ímãs permanentes fixados na carcaça do motor. No motor CC o rotor é um eletroímã composto por um núcleo de ferro com enrolamentos em sua parte superior.

Os ímãs são utilizados com o objetivo de criar um fluxo magnético fixo enquanto o rotor é percorrido por uma corrente elétrica. Com o auxílio de escovas e

comutadores, a direção da corrente é alterada, fazendo com que o rotor gire continuamente (Niku, 2017).

2.6 Modulação por Largura de Pulso

A Modulação por Largura de Pulso (*Pulse Width Modulation – PWM*) é um método bastante utilizado para controle de robôs, motores e diversos outros dispositivos. Como os microcontroladores e microprocessadores possuem portas de saída com PWM, o controle de mecanismos com essa modulação tornou-se ainda mais popular.

De acordo com Niku (2017), na modulação por largura de pulso a tensão sobre uma determinada porta de um microprocessador é ativada e desativada muitas vezes por segundo, dessa forma a tensão média eficaz varia de acordo com o tempo que a porta está com tensão máxima ou mínima.

A tensão de saída média de PWM é dada pela Equação 14, em que t_1 é a largura de pulso e t é o período:

$$V_{saída} = V \cdot \frac{t_1}{t} \quad (14)$$

A frequência de PWM pode ser muito maior que a frequência natural de vários dispositivos e essa frequência de comutação pode causar ruídos e interferência em alguns destes. No entanto, a comutação (liga/desliga) causada pela PWM tem pouco efeito sobre o desempenho do motor, pois o mesmo atua como um filtro passa-baixa (Niku, 2017). Entretanto, o motor responde satisfatoriamente ao valor médio da tensão de entrada do sinal PWM.

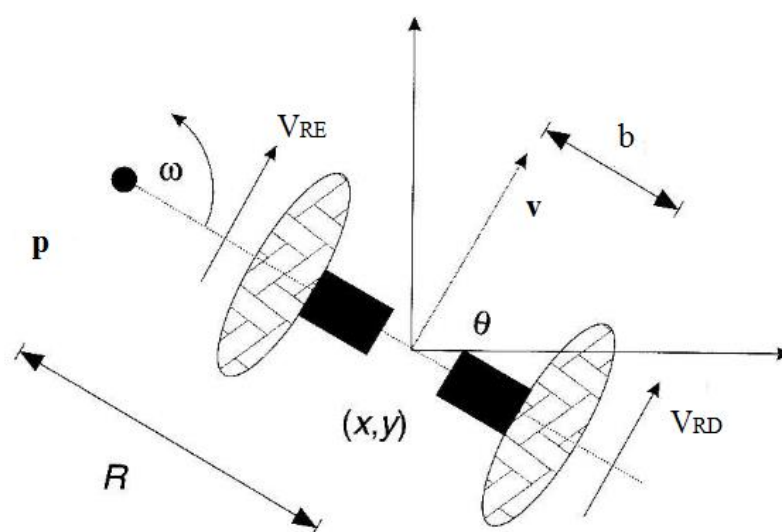
Além do entendimento dos sensores e dispositivos necessários para o funcionamento de um robô móvel com tração diferencial, é necessário compreender seu modelo matemático. A seção 2.7 detalha o modelo cinemático de um robô com tração diferencial.

2.7 Modelo cinemático de um robô com tração diferencial

Nesta seção é descrito o modelo cinemático de um robô móvel com tração diferencial.

A localização e a posição do robô no sistema de coordenadas cartesiano podem ser determinadas através das variáveis x , y , e θ , que representam a posição e o ângulo de orientação do robô respectivamente (HUANG e HUANG, 2010). A Figura 10 mostra o modelo cinemático do robô.

Figura 10 - Modelo cinemático AGV.



Fonte: Adaptado de Dudek e Jenkin (2010).

Dessa forma e de acordo como a Figura 10, o modelo do robô pode ser expresso no espaço de estado selecionando as derivadas de x , y e θ como mostra a Equação 15.

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (15)$$

onde v é a velocidade linear (escalar) do robô ao longo de seu eixo longitudinal e ω é a velocidade angular do robô ao longo de seu eixo de rotação.

A relação matemática que associa a velocidade linear com a velocidade angular é descrita pela Equação 16.

$$v = \omega \cdot R \quad (16)$$

em que R é o raio da trajetória do robô levando em conta o centro de massa do robô e um determinado ponto de giro p .

De acordo com a Figura 10 e com a Equação 16, as velocidades das rodas direita e esquerda, representadas por V_{RD} e V_{RE} respectivamente podem ser expressas pela Equação 17.

$$\begin{cases} V_{RD} = \omega \cdot (R + b) \\ V_{RE} = \omega \cdot (R - b) \end{cases} \quad (17)$$

onde b é a metade do comprimento do eixo que separa as rodas direita e esquerda, como ilustra a Figura 10.

A partir da Equação 17 é possível encontrar uma relação para a velocidade angular ω em função das velocidades independentes das rodas direita e esquerda do robô e também uma relação para o raio, representadas pelas Equações 18 e 19 respectivamente.

$$\omega = \frac{V_{RD} - V_{RE}}{2b} \quad (18)$$

$$R = \frac{b \cdot (V_{RE} + V_{RD})}{V_{RE} - V_{RD}} \quad (19)$$

Substituindo as Equações 18 e 19 na Equação 16, é possível encontrar uma relação para a velocidade linear do robô em função das velocidades de cada roda como mostra a Equação 20.

$$v = \frac{V_{RE} + V_{RD}}{2} \quad (20)$$

Analisando a Equação 20, é possível concluir que a velocidade linear do robô é a média das velocidades de cada roda.

Substituindo as Equações 18 e 20 na Equação 15, obtemos o modelo cinemático do robô móvel com tração diferencial em função das velocidades das rodas esquerda e direita do robô.

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \frac{\cos \theta}{2} & \frac{\cos \theta}{2} \\ \frac{\sin \theta}{2} & \frac{\sin \theta}{2} \\ \frac{-1}{2b} & \frac{1}{2b} \end{bmatrix} \cdot \begin{bmatrix} V_{RE} \\ V_{RD} \end{bmatrix} \quad (21)$$

De acordo com Huang e Huang (2010), aplicando a transformada de Laplace na Equação 21 é possível encontrar a função de transferência do robô em função das velocidades de cada roda, como mostra a Equação 22.

$$\begin{cases} \theta(s) = \frac{V_{RE}-V_{RD}}{2b} \\ x(s) = \frac{(V_{RE}+V_{RD})\cdot\cos\theta}{2s} \\ y(s) = \frac{(V_{RE}+V_{RD})\cdot\sen\theta}{2s} \end{cases} \quad (22)$$

Os pontos que marcam a posição e direção do robô podem ser encontrados ao integrar a Equação 15 como, afirma Yacine, Fatima e Aissa (2015). Dessa forma obtêm-se as coordenadas x , y e a inclinação θ representadas pelas Equações 23, 24 e 25.

$$x(t) = x_0 + \int_0^t v(t) \cdot \cos(\theta(t)) \cdot dt \quad (23)$$

$$y(t) = y_0 + \int_0^t v(t) \cdot \sen(\theta(t)) \cdot dt \quad (24)$$

$$\theta(t) = \theta_0 + \int_0^t \omega(t) \cdot dt \quad (25)$$

Neste capítulo foi realizado um levantamento das teorias envolvendo as técnicas de visão computacional, controle *Fuzzy*, controle PID, robô de tração diferencial e dos dispositivos necessários para a construção de um robô móvel. Com base nessas informações, o capítulo 3 apresenta a metodologia utilizada para a construção do robô móvel e seu controle.

3 MATERIAIS E MÉTODOS

Este capítulo descreve a metodologia aplicada na construção do veículo guiado automaticamente, as etapas de captura das imagens, o processamento das imagens e a estratégia de visão computacional, assim como as estratégias de controle baseadas em lógica *Fuzzy* e nos controladores PID. O capítulo também descreve os materiais utilizados para construção do robô, a aquisição de dados e os softwares utilizados.

3.1 Metodologia

A metodologia aplicada neste trabalho consiste inicialmente na sequência de técnicas de captura e processamento de imagens. As imagens são capturadas em uma superfície plana com luz interna controlada como um chão de fábrica ou laboratório. A imagem resultante após as etapas de processamento é uma imagem binarizada, ou seja, preta e branca. A parte branca representa o percurso do robô, ou seja, o caminho a ser seguido. Nesta fase o objetivo é capturar e fornecer um conjunto de imagens adequadas para a fase de visão computacional.

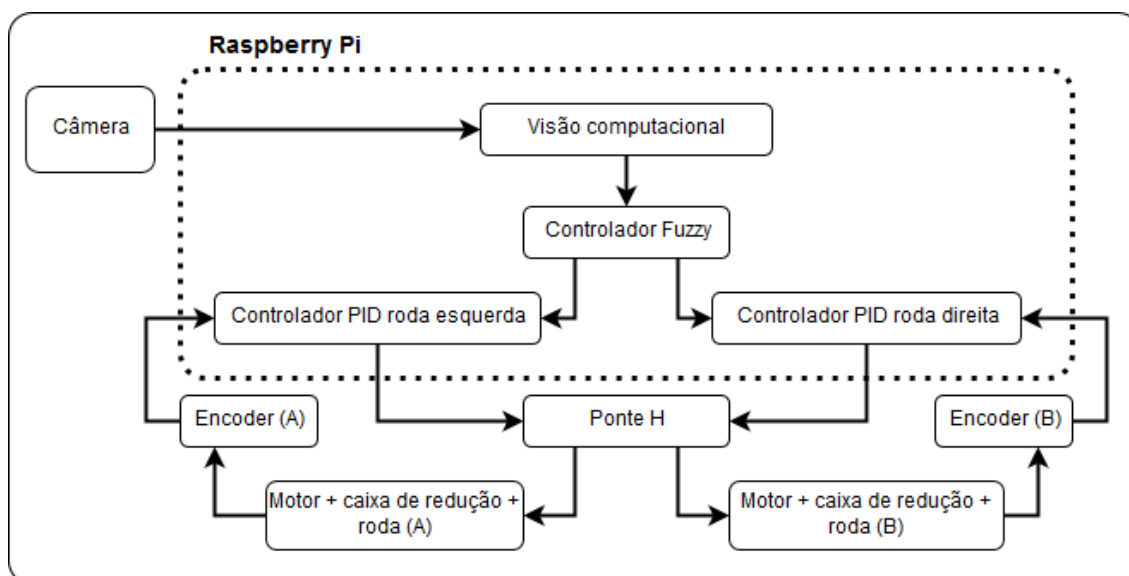
A fase de visão computacional tem a finalidade de analisar e interpretar as imagens, com o objetivo de encontrar a faixa (linha) branca e descobrir a distância entre o centro da faixa e o centro do robô. O centro do robô é representado por uma linha guia fixa no centro da imagem. A distância entre a linha detectada e a linha guia serve como dado de entrada para o controlador baseado em lógica *Fuzzy*.

Nesta fase, o controlador *Fuzzy* é o responsável por informar o quanto a velocidade de cada roda deve aumentar ou diminuir para que a direção do robô seja controlada e o robô permaneça sobre a linha detectada. Dessa forma, o controlador *Fuzzy* pode ser entendido como um controlador de posição para o robô, uma vez que este trabalho trata-se de um robô com tração diferencial.

Por fim, a informação fornecida pelo controlador *Fuzzy* serve como referência para os controladores PID de cada motor. Estes controladores PID são os responsáveis por assegurar a velocidade correta dos motores e conseqüentemente das rodas. Eles garantem que ambos os motores girem na mesma velocidade quando necessário. Portanto, os controladores PID podem ser entendidos como os controladores de velocidade do robô.

A Figura 11 ilustra o diagrama de blocos da metodologia aplicada neste trabalho.

Figura 11 - Diagrama de blocos da metodologia.



Fonte: Autoria própria.

Cada etapa da metodologia, representada na Figura 11, são compostas por uma sequência definida de passos. Nos tópicos a seguir todas etapas são detalhadas.

3.2 Construção do AGV

Neste tópico são apresentados os materiais utilizados na construção do robô móvel, sua interligação e configuração.

3.2.1 Estrutura

Para a construção da estrutura do robô móvel foi utilizado duas placas plásticas de 21 cm de comprimento por 15 cm de largura e 4mm de espessura. As placas foram fixadas paralelamente com um espaçamento de 6 cm entre elas. Para a fixação das placas foram utilizados 4 hastes metálicas rosqueáveis.

O espaçamento entre placas permite que os motores sejam fixados na parte inferior de uma das placas e os demais componentes de controle dos motores sejam fixados dentro do espaçamento entre elas. A placa superior permite a fixação da câmera e controladores. Essa configuração também permite a separação dos componentes mais sensíveis como a bateria.

A Figura 12 ilustra a estrutura plástica com as hastes metálicas construída (parte superior esquerda da Figura 12) e os demais componentes utilizados para a construção do robô.

Figura 12 - Estrutura do robô.



Fonte: Autoria própria.

Os sensores, motores, câmera e demais componentes usados na construção do robô proposto neste trabalho são apresentados nas seções a seguir. O robô montado com todos os componentes é apresentado no capítulo 4.

3.2.2 Motores e caixa de redução

Para este trabalho foram escolhidos dois motores de corrente contínua com escovas (*brushed*) e estator em ímã permanente. A escolha foi motivada pela grande versatilidade deste tipo de motor, sua simplicidade no controle de velocidade e o baixo custo.

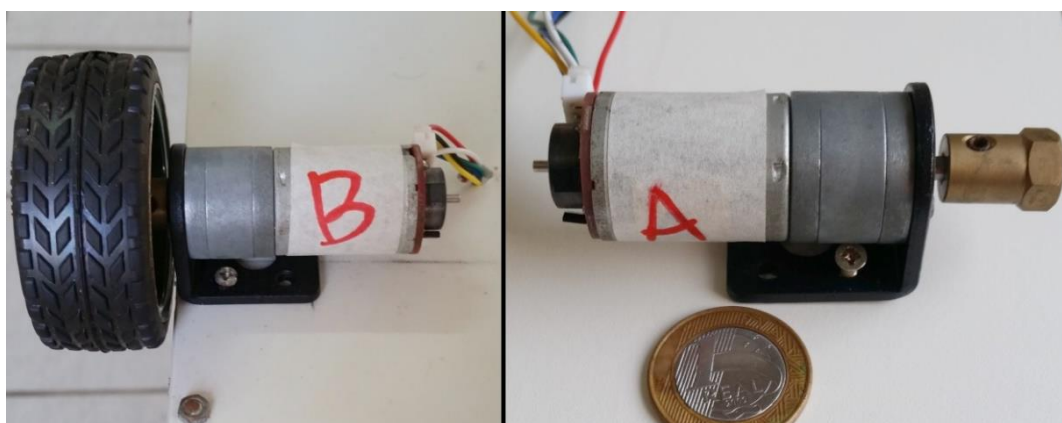
O motor CC utilizado para o deslocamento do robô foi um modelo com tensão e velocidade nominais de 6 a 9V e 210 e 310 RPM respectivamente. Neste motor já vem embutido uma caixa de redução que aumenta o torque no eixo ao ponto que diminui a velocidade. A relação de redução é de 1:34, ou seja, para cada 34 voltas

completas no eixo do motor, o eixo final da redução realiza 1 volta completa. O torque, no entanto, aumenta em 34 vezes, desconsiderando as perdas.

Além da caixa de redução o motor também é comercializado com um encoder de quadratura já fixo no motor.

A Figura 13 ilustra os motores utilizado e as marcações efetuadas.

Figura 13 - Motores A e B.



Fonte: Autoria própria.

O fabricante dos motores fornece algumas informações uteis, como por exemplo, a relação de Pulsos Por Revolução (PPR) do encoder, a relação da caixa de redução, o torque máximo, entre outros. A Tabela 2 informa todas as especificações do motor fornecidas pelo fabricante.

Tabela 2 - Parâmetros do motor.

Parâmetro	Valor	Unidade
Tensão nominal	6 - 9	V
Corrente sem carga	0.13	A
Velocidade sem carga	210 - 310	RPM
Eficiência máxima	170 / 0.6	RPM / A
Potência máxima	3.1	W
Torque máximo	10	kg.cm
Relação caixa redução	1:34	
PPR com a caixa de redução	341	PPR

Fonte: Autoria própria.

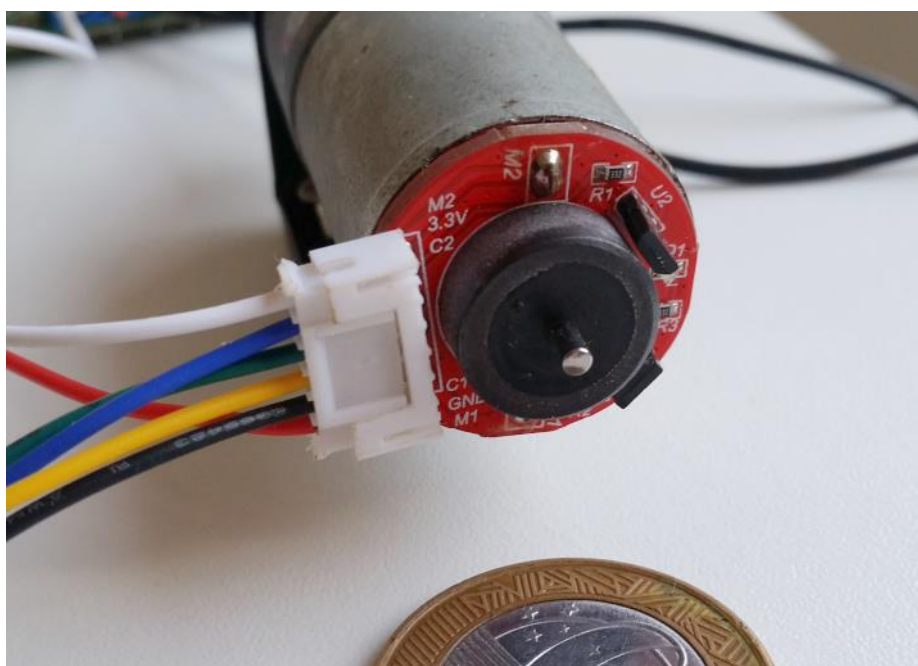
No entanto, foram efetuados alguns testes usando um osciloscópio portátil e um tacômetro digital, para conferir os valores de RPM e PPR fornecidos pelo fabricante.

O teste foi feito aplicando um valor de tensão que permitisse que o eixo onde a roda é fixada girasse a 60 RPM, ou seja, a 1 RPS. Então foi medido a quantidade de pulsos na saída do *encoder*. Os resultados são apresentados no Capítulo 4.

3.2.3 Encoders de quadratura

Os *encoders* de quadratura utilizados são baseados em sensores de efeito hall. Cada *encoder* possui dois sensores defasados entre eles em um ângulo de 90° para efeito de quadratura como foi apresentado no Capítulo 2. Na Figura 14 é possível visualizar os *encoders* utilizados e o defasamento dos sensores.

Figura 14 - *Encoder* de Quadratura.



Fonte: Autoria própria.

A relação de Pulsos Por Revolução (PPR) do *encoder* fornecida pelo fabricante, levando em conta a caixa de redução de 1:34, é de 341 PPR. Dessa forma, para cada volta completa da roda do robô o *encoder* fornecerá 341 pulsos. Este valor é importante para calcular a velocidade em que o robô está se locomovendo.

3.2.4 Rodas

Foram utilizadas duas rodas de plástico com diâmetro de 65mm. A cada volta completa realizada pela roda, o robô efetua um deslocamento de aproximadamente 20.41cm. Essa relação foi obtida com a aplicação da Equação 26, referente ao comprimento da circunferência.

$$C = dr.\pi \quad (26)$$

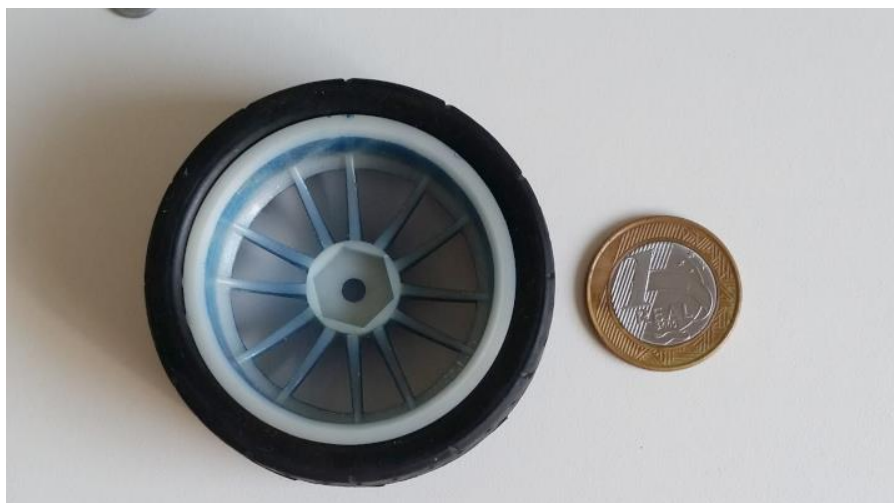
Na Equação 26, C representa o comprimento da circunferência, dr o diâmetro da roda e π o número Pi.

Dessa forma, podemos obter o valor informado substituindo o valor do diâmetro da roda na Equação 27.

$$C = 65mm \times 3.14 = 20.41cm \quad (27)$$

As rodas utilizadas são ilustradas na Figura 15.

Figura 15 - Modelo de roda utilizada.



Fonte: Autoria própria.

Já a roda castor utilizada pode ser vista na Figura 16.

Figura 16 - Roda castor.



Fonte: Autoria própria.

A roda castor permite que a parte traseira do robô rode livremente e com menor requerimento de força.

A direção do robô é ditada pelas rodas da frente, acopladas aos motores que são controlados por uma ponte H.

3.2.5 Ponte H

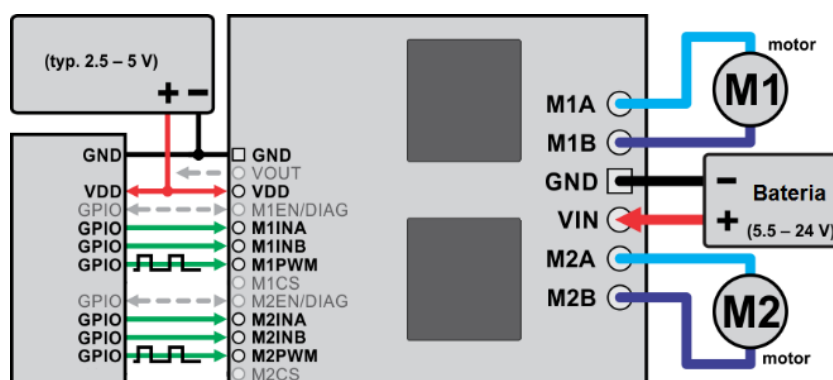
Como o sinal de controle proveniente do controlador não tem potência suficiente para acionar corretamente os motores, se faz necessário a utilização de uma ponte H. A ponte H permite o controle de motores de maior potência através de sinais PWM de baixa potência.

A ponte H escolhida para o controle dos motores foi a placa modelo *Pololu Dual VNH5019*. A placa possui dois circuitos integrados VNH5019, cada um permite o controle bidirecional de um motor CC até 24V, com corrente máxima de 12A e pico de até 30A. A placa também permite o monitoramento da corrente solicitada por cada motor. Dessa forma, cada motor pode ser controlado independentemente.

A escolha desta ponte H foi motivada pela sua robustez e facilidade de implementação, tanto na programação quanto na conexão com os demais componentes do robô móvel.

A conexão da ponte H foi feita seguindo o esquemático ilustrado na Figura 17. Os pinos necessários para controle foram ligados diretamente no *Raspberry Pi*, a alimentação do circuito lógico da ponte H foi feita em paralelo com a placa *Raspberry Pi* e a entrada de potência foi feita diretamente na bateria.

Figura 17 - Ponte H dupla.



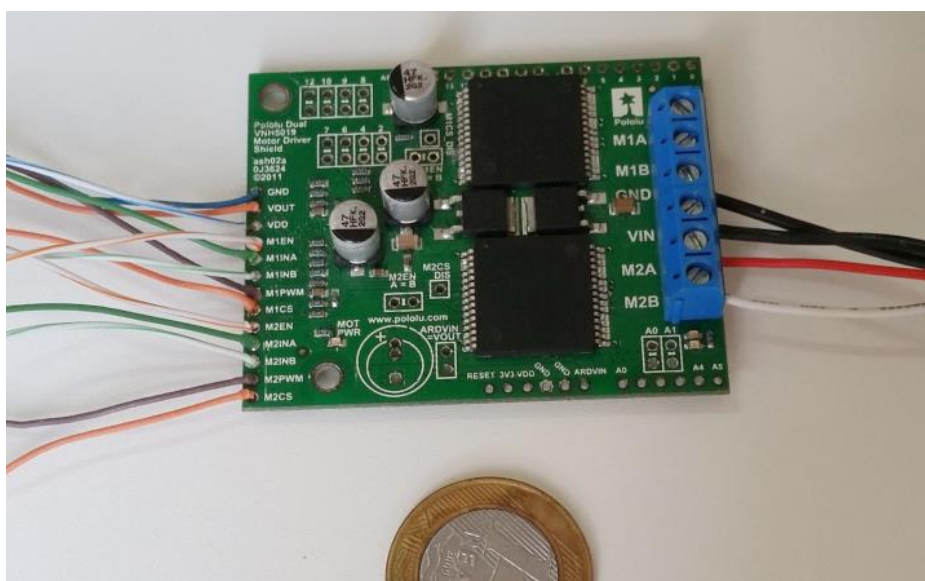
Fonte: Polulu (2019).

Como indicado na Figura 17, essa ponte H facilita o controle de dois motores separadamente utilizando a mesma fonte de alimentação. Isso ocorre pois a placa utiliza dois circuitos integrados VNH5019, uma para cada motor.

Segundo a Polulu (2019), o VNH5019 é um circuito integrado fabricado pela *ST Eletronics* amplamente utilizado em aplicações onde é necessário o controle de velocidade e sentido de rotação de motores de corrente contínua, devido seu circuito de controle PWM embutido e também por sua estabilidade e segurança, garantida pelos resistores de elevação, limitadores de corrente e um módulo FET para proteção de alimentação reversa, que protegem todo o circuito de sobrecargas geralmente causadas nas partidas dos motores.

A ponte H possui no seu lado direito dois canais, M1 e M2, para a conexão dos motores e controle independente. Cada canal possui dois terminais, A e B, para a conexão dos polos positivo e negativo do motor. Ainda no lado direito da placa se encontra as entradas, GND e VIN, que correspondem respectivamente aos terminais negativo e positivo da fonte de alimentação que supre a demanda dos motores. A Figura 18 detalha a pinagem referida acima.

Figura 18 - Pinagem ponte H.



Fonte: Autoria própria.

Já os pinos de controle dos motores, parte lógica da placa, fica localizado no lado esquerdo, como mostra as Figuras 17 e 18. Essa parte da placa é alimentada com 3.3 - 5V.

Os pinos presentes no lado esquerdo são divididos da seguinte forma:

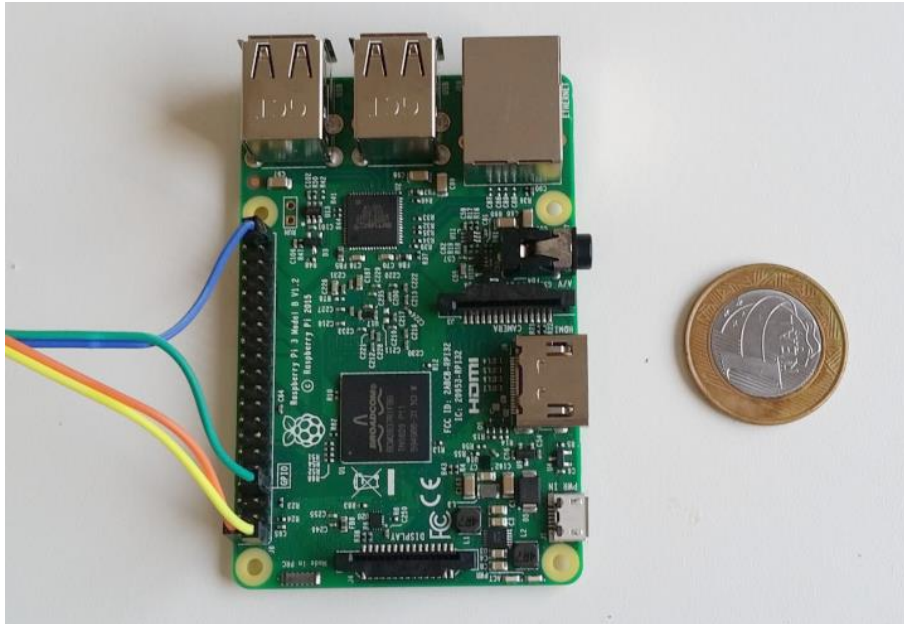
- Alimentação da parte lógica da ponte-H.
- Acionamento e direção dos motores.
- Velocidade dos motores (PWM).

O sinal PWM de entrada da ponte H é enviado pelo Raspberry Pi.

3.2.6 *Raspberry Pi*

O *Raspberry Pi* é um computador portátil em placa única compatível com sistemas operacionais Linux. Neste pequeno computador é possível executar diversos programas que facilitam o controle de dispositivos e a aquisição de dados. O *Raspberry Pi* é utilizado como controlador geral neste trabalho. Todos os algoritmos das fases de processamento de imagens, visão computacional, controle *Fuzzy*, controle PID e controle da ponte H foram desenvolvidos para serem executados no *Raspberry Pi*. A Figura 19 mostra o *Raspberry Pi* utilizado.

Figura 19 - Raspberry Pi



Fonte: Autoria própria.

O *Raspberry Pi* utilizado é o modelo 3 B com processador *Quad Core* de 1.2GHz e 1GB de memória RAM. As principais especificações deste modelo são:

- Processador *Quad Core* BCM2837 de 64bit e 1.2GHz fabricado pela Broadcom.
- Memória RAM de 1GB.
- 40 Pinos de entrada e saída de uso geral.
- Porta CSI para conexão de câmera.
- Porta DSI para conexão de display.
- Portas USB x 4.
- Saída HDMI.
- Saída de vídeo *stereo*.
- Porta Micro SD.
- Porta Micro USB de até 2.5A para alimentação.

Além do processamento capaz de suportar sistemas operacionais Linux, o *Raspberry Pi* possui 40 pinos de entrada e saída de uso geral. Esse barramento permite controlar todos os demais dispositivos necessários para o funcionamento do robô,

como o controle da ponte H ligada aos motores e o recebimento das informações dos *encoders* para medição de velocidade.

3.2.7 Câmera

A câmera é um dos dispositivos principais deste trabalho. É por meio dela que são capturadas todas as imagens necessárias na etapa de processamento de imagens e visão computacional. Dessa forma, é possível afirmar que a câmera em conjunto com as etapas de processamento de visão atuam como um sensor, que fornece a informação de entrada de todo o sistema.

A câmera utilizada neste trabalho é baseada no sensor OV5647 de 5 megapixel com tamanho de $\frac{1}{4}$ de polegada. O OV5647 é um sensor com tecnologia CMOS (*complementary metal-oxide semiconductor*) de baixa tensão fabricado pela *OmniVision* e amplamente utilizado em celulares e dispositivos móveis. O sensor permite a saída de vídeo com tamanho de imagem real máxima de 2592 x 1944 pixels a uma taxa de 15 FPS (frames por segundo). No entanto, segundo a *OmniVision* (2009), o sensor OV5647 permite a seleção de modos com diferentes resoluções e taxas de transferência, como mostra a Tabela 3.

Tabela 3 - Especificações da câmera.

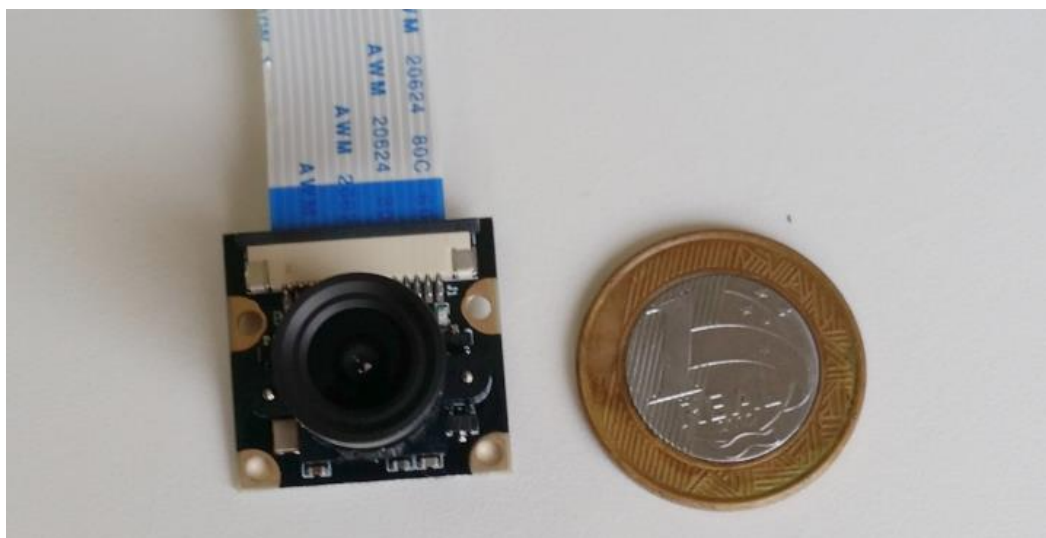
Resolução	FPS
QSXGA (2592 x 1944)	15 FPS
1080p	30 FPS
960p	45 FPS
720p	60 FPS
VGA (640 x 480)	90 FPS
QVGA (320 x 240)	120 FPS

Fonte: *OmniVision* (2009).

A escolha desta câmera foi motivada pelo seu baixo custo e pelo fato do sensor OV5647 ter sido utilizado na primeira versão das câmeras oficiais destinadas ao *Raspberry Pi*, fabricadas pela *Raspberry Pi Foundation*. No entanto, o modelo utilizado não é o oficial, mas possui o mesmo sensor e é compatível com a versão do *Raspberry Pi* utilizada, com a documentação e códigos de apoio da versão original.

Diferentemente da versão original, a câmera utilizada acompanha uma lente com ajuste de foco manual que facilita na implementação da tarefa ao qual a câmera é destinada. Na Figura 20 é possível ver a câmera utilizada.

Figura 20 - Câmera CMOS



Fonte: Autoria própria.

Como mostra a Tabela 3, é possível escolher entre altas e baixas resoluções de imagens. No entanto, a tarefa a ser executada não necessita de altas resoluções. Por este motivo, e para facilitar o processamento das imagens a resolução em VGA foi a escolhida para este trabalho.

3.2.8 Bateria

Para que o robô consiga uma boa autonomia é necessário uma bateria, ou conjunto de baterias, que consigam suprir a demanda dos motores e do circuito de controle. Neste projeto as baterias escolhidas foram do tipo Li-Po (polímero de lítio), pois são baterias de alta descarga e que podem ser recarregadas completamente no intervalo de 1 hora. O robô foi projetado para funcionar com somente uma bateria, mas neste projeto foram utilizadas duas baterias, uma por vez, para diminuir o ciclo de carga e descarga das baterias.

Foram utilizadas dois modelos de baterias Li-Po, ambas de três células e tensão nominal de 11.1V. Estas baterias são amplamente utilizadas em aeromodelos, drones, entre outros.

A diferença entre as baterias se limita à capacidade delas e ao fabricante. Uma possui 4200 mAh e a segunda 3800 mAh.

Na Figura 21 é possível ver as baterias utilizadas e algumas de suas características.

Figura 21 - Baterias LiPo.



Fonte: Autoria própria.

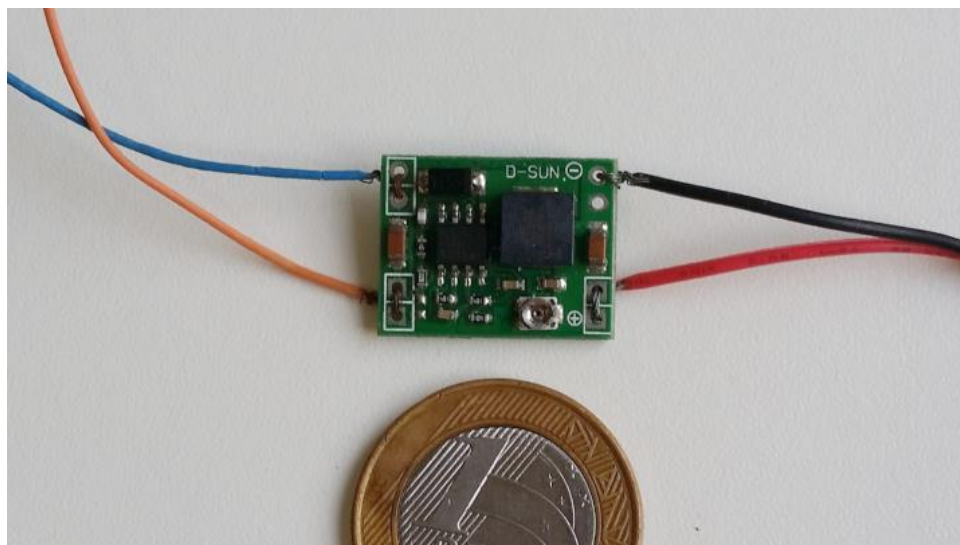
Essas baterias são de alta descarga, isso quer dizer que elas conseguem fornecer energia suficiente para a partida de motores. A corrente máxima que cada uma destas baterias pode fornecer é calculada multiplicando sua capacidade pela constante C, informada no rotulo de cada bateria. Por exemplo, uma bateria de 4200 mAh e 25C pode suportar a demanda de um pico de corrente até 105A.

Como as baterias Li-Po conseguem fornecer elevada corrente, é importante impedir que ocorram curtos circuitos ao utilizar tais baterias.

3.2.9 Conversor CC abaixador (Buck):

Como as baterias utilizadas para alimentar todo o sistema do robô possuem tensão nominal de 11.1V, é necessário a utilização de um conversor que forneça uma tensão de 5V para o *Raspberry Pi*. Para isso foi utilizado um conversor abaixador *buck* que mantém a tensão de saída fixa em aproximadamente 5V. A Figura 22 mostra o conversor utilizado.

Figura 22 - Conversor Buck.



Fonte: Autoria própria.

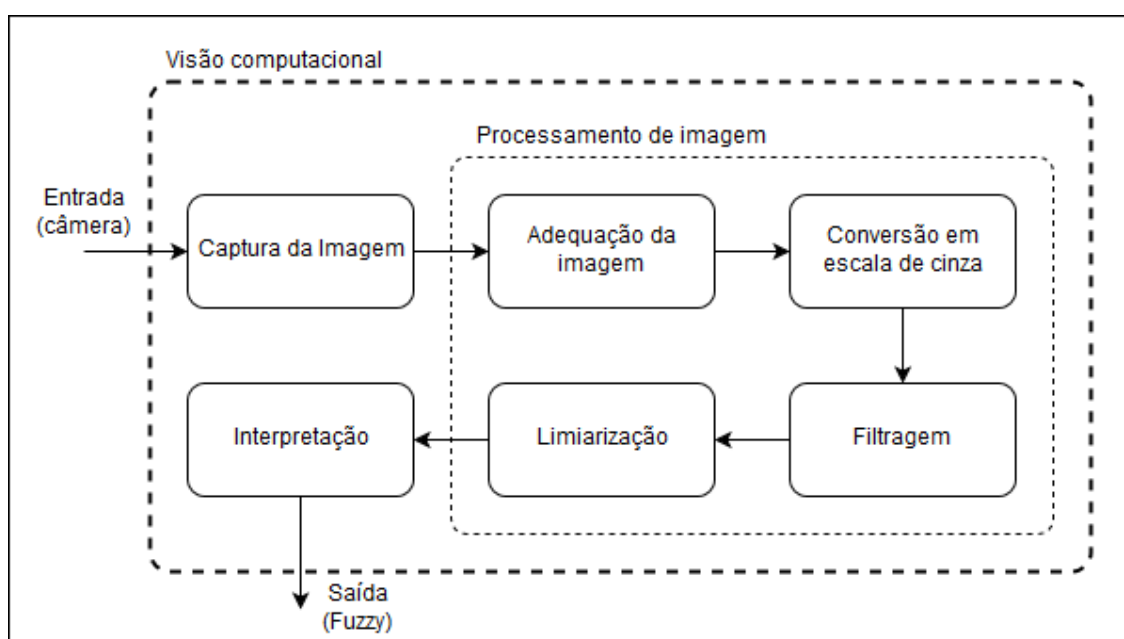
Este conversor é de uso geral e é uma melhor alternativa aos reguladores de tensão lineares, como o 7805 e semelhantes, pois não ocorre a dissipação da energia na forma de calor para diminuir a tensão.

Após o detalhamento dos materiais utilizados na construção do robô, a seção 3.3 detalha as etapas da metodologia envolvendo a visão computacional.

3.3 Visão computacional

A metodologia aplicada durante a fase de visão computacional é dividida em 6 etapas, sendo elas: captura da imagem, adequação da imagem, conversão da imagem em escala de cinza, filtragem, binarização (limiarização), e interpretação das informações. Destas etapas, 4 estão relacionadas ao processamento de imagem diretamente. Já as outras 2 etapas são referentes a captura da imagem e a interpretação das informações. A Figura 23 mostra o fluxograma da fase de visão computacional.

Figura 23 - Fluxograma da fase de visão computacional.



Fonte: Autoria própria.

O bloco de Captura da Imagem constitui o conjunto de configurações feitas no *Raspberry Pi* para que seja possível o acionamento da câmera e a captura das imagens. Já a adequação da imagem visa obter uma imagem em 640x480 pixels.

A conversão em escala de cinza, tem como objetivo, converter uma imagem RGB, composta por 3 matrizes, em uma única matriz de intensidades, onde cada pixel varia de 0 a 255. Essa etapa facilita o processamento e o sucesso das etapas seguintes.

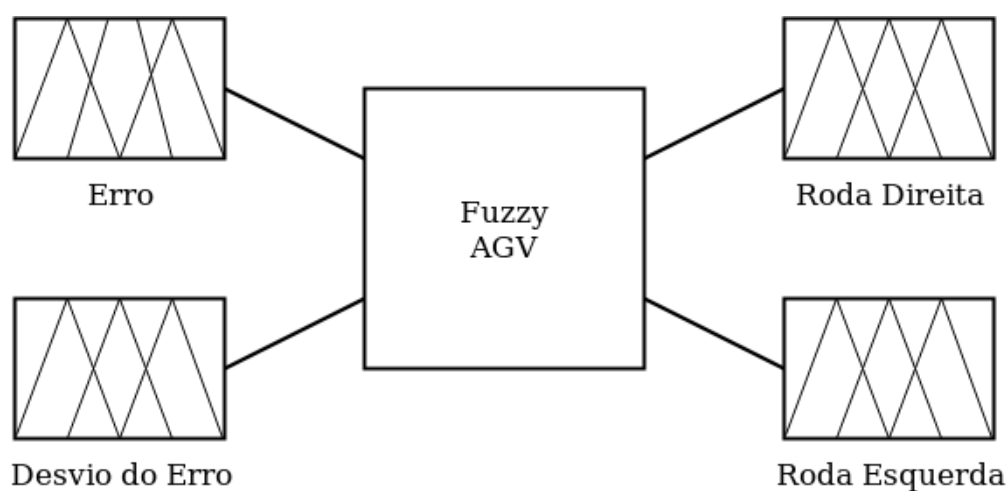
A etapa seguinte utiliza um Filtro Gaussiano de suavização. O objetivo da aplicação desse filtro é eliminar ruídos presentes nas imagens digitais, além de suavizar a trilha a ser seguida.

A binarização ou limiarização é a fase de segmentação que objetiva destacar as partes escuras das brancas, convertendo tudo que estiver acima de um limiar para a cor branca e o restante para preto. O método de binarização empregado é o método de Otsu. Ainda nesta etapa é necessário complementar (inverter) a matriz gerada pelo método de Otsu, caso a trilha seja na cor preta. Dessa forma, a trilha será representada por uns e o restante por zeros.

3.4 Controlador *Fuzzy*

O controlador *Fuzzy* desenvolvido é composto por duas variáveis linguísticas de entrada e duas de saída. As variáveis de entrada referem-se ao erro e ao desvio do erro respectivamente. Já as variáveis de saída referem-se à velocidade das rodas direita e esquerda respectivamente. A Figura 25 ilustra o diagrama de blocos do controlador *Fuzzy*.

Figura 25 - Diagrama de blocos do controlador *Fuzzy*.



Fonte: Autoria Própria.

A variável linguística de entrada “Erro”, fornece ao controlador *Fuzzy*, a diferença entre o valor em pixel do centro da imagem e o valor em pixel onde se encontra o centro da linha branca na imagem fornecida pelo sistema de processamento de imagem e visão computacional.

Como as dimensões da imagem são 640 pixels por 480 pixels, o centro da imagem ou o valor de referência é 320 pixels. Já o valor em pixel onde se encontra o centro da linha branca pode variar entre 1 e 640 pixels.

Dessa forma, o valor do erro pode assumir valores entre 319 pixels e -320 pixels de acordo com a Equação 28:

$$ep = Pc - Pm \quad (28)$$

onde Pc é o pixel central, Pm é o pixel medido e ep é o erro.

No entanto, na prática os valores do “Erro” não assumem tais valores extremos, devido o sistema ser realimentado, ou seja, o trajeto do robô é corrigido a cada iteração.

Como o valor de P_c é fixo. A equação 28 pode ser reescrita como:

$$ep = 320 - P_m \quad (29)$$

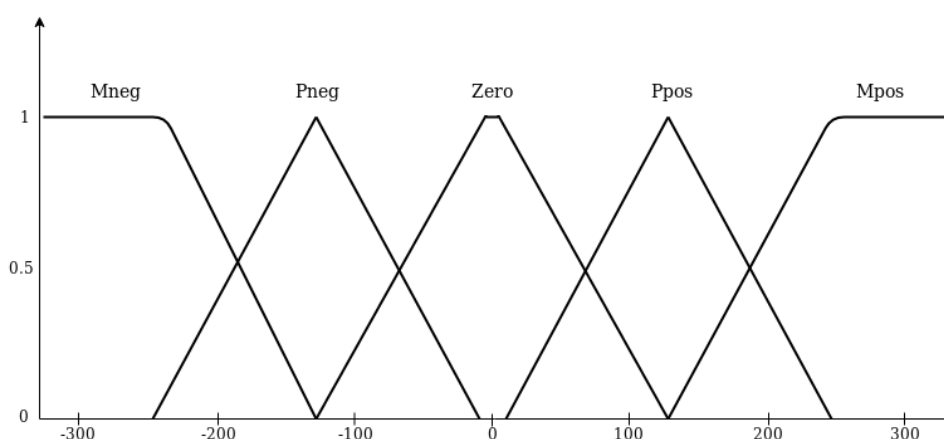
Portanto, em um cenário em que P_m assuma valores iguais a 1 ou 640 pixel, o valor de ep será 319 ou -320 respectivamente.

Quando nenhuma linha branca for detectada na imagem o valor do erro será igual a 320 pixel, ou seja, o valor o pixel medido será zero.

O erro representa o quanto o robô está aproximando-se da linha a ser seguida. Quando o erro é próximo de zero o robô está o mais alinhado possível com a linha. Já quando o erro assume valores negativos o robô está se afastando da linha central para a direita, ou seja, o valor do pixel medido é maior que 320. Quando o erro assume valores positivos o robô está se afastando da linha central para a esquerda, ou seja, o valor do pixel medido é menor que 320.

A Figura 26 ilustra graficamente o alcance da variável linguística “Erro” e a distribuição de seu conjunto de termos.

Figura 26 - Variável linguística “Erro”



Fonte: Autoria Própria.

Para a variável linguística “Erro” foi criado o conjunto de termos: Zero; Pneg; Ppos; Mneg e Mpos. Os termos “Pneg” e “Pos” representam valores de erro pouco

negativos e pouco positivos respectivamente. “Mneg” e “Mpos” representam valores muito negativos e muito positivos. Já o termo “Zero” representa valores próximos do erro zero. O alcance “range” da variável “Erro” foi definido entre -320 e 319. Foram utilizadas funções de pertinência do tipo triangular para os termos Pneg e Ppos da variável “Erro”, e funções trapezoidais para o termo Zero, Mneg e Mpos.

O intervalo de alcance para cada um dos cinco termos da variável de entrada “Erro” é listado a seguir:

- Muito negativo (Mpeg): [-320 -320 -250 -125];
- Pouco negativo (Ppos): [-255 -130 -5];
- Zero (Zero): [-125 -5 5 125];
- Pouco positivo (Ppos): [5 130 255];
- Muito positivo (Mpos): [125 250 319 319].

Note que o termo “Zero” desta variável de entrada utiliza uma função de pertinência trapezoidal com um intervalo central de -5 a 5. Isso foi pensado para permitir uma maior liberdade do robô quando estiver posicionado sobre a linha. Na pratica, isso aumenta o intervalo de controle desejado, ou seja, para qualquer valor entre -5 e 5 o controlador vai entender que o objetivo de centralizar o robô com a linha foi alcançado, ou seja, o erro é “Zero”. Isso é necessário pelo fato de ser praticamente impossível que o robô fique centralizado sobre um determinado ponto por pouco mais de alguns segundos.

Já a variável de entrada “Desvio do Erro” fornece ao controlador *Fuzzy* o valor da diferença entre o erro atual e o erro anterior. O desvio do erro ou tendência do erro, como também é conhecido, indica o quão rápido o robô está se aproximando ou afastando do seu objetivo, ou seja, o quão rápido o robô está se distanciando ou aproximando da trilha a ser seguida.

Nesta metodologia, quando o erro assume valores negativos e o desvio do erro assume valores mais negativos o robô tenderá a distanciar-se da linha para a direita. Já se o desvio do erro assumir valores mais positivos o robô tenderá a aproximar-se da linha pela direita.

No caso de valores positivos para o erro, quando o desvio do erro for negativo o robô tenderá a aproximar-se da linha pela esquerda. Já para valores positivos para o desvio, o robô tenderá a distanciar-se da linha pela esquerda.

Quanto maior o modulo do valor do desvio, mais rápido será a aproximação ou afastamento da linha a ser seguida. Já quando o valor do desvio do erro for zero ou próximo de zero, o robô estará seguindo um trajeto de forma estável, ou seja, sem variações bruscas na direção, pois neste cenário o erro não está variando.

O desvio do erro é representado pela Equação 30:

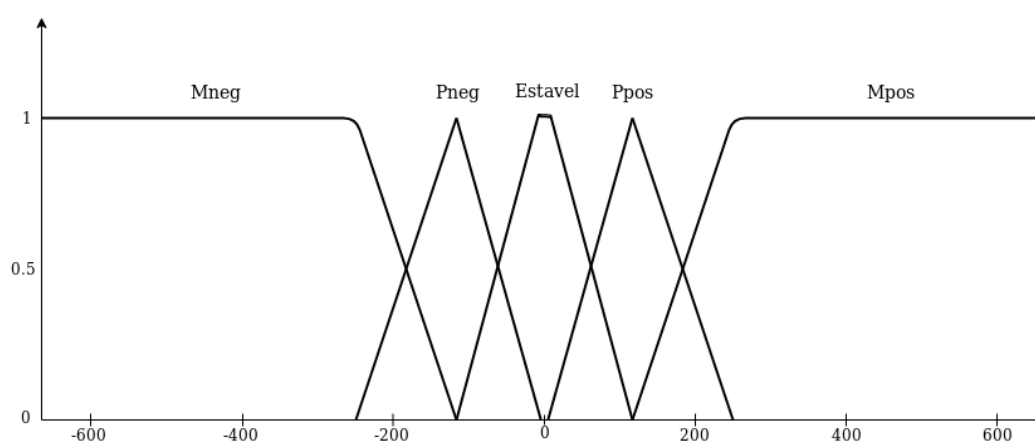
$$\Delta e = ep - epa \quad (30)$$

em que Δe é o desvio do erro, ep é o erro atual e epa é o erro da interação anterior.

Dessa forma, o desvio do erro pode assumir valores entre -639 e 639, levando em conta variações bruscas na velocidade do robô. No entanto, na prática os valores do desvio do erro não assumem tais valores extremos devido o sistema ser realimentado.

A Figura 27 ilustra o alcance da variável linguística de entrada “DesvioErro” e a distribuição de seu conjunto de termos.

Figura 27 - Variável linguística de entrada “DesvioErro”



Fonte: Autoria Própria.

Como ilustra a Figura 27, para a variável linguística “DesvioErro” foi criado o conjunto de termos: Mneg, Pneg, Estável, Ppos e Mpos. Os termos “Pneg” e “Pos” representam valores de desvio do erro pouco negativos e pouco positivos

respectivamente. “Mneg” e “Mpos” representam valores muito negativos e muito positivos para o desvio. Já o termo “Estável” representa valores próximo de zero. O alcance (range da variável “DesvioErro”) foi definido entre -639 e 639. Foram utilizadas funções de pertinência do tipo triangular para os termos Pneg e Ppos da variável “DesvioErro”, e funções trapezoidais para o termo Estável, Mneg e Mpos.

O intervalo de alcance para cada um dos cinco termos da variável de entrada “DesvioErro” é listado a seguir:

- Muito negativo (Mneg): [-639 -639 -250 -125];
- Pouco negativo (Pneg): [-255 -130 -5];
- Estável (Estavel): [-125 -5 5 125];
- Pouco positivo (Ppos): [5 130 255];
- Muito positivo (Mpos): [125 250 639 639].

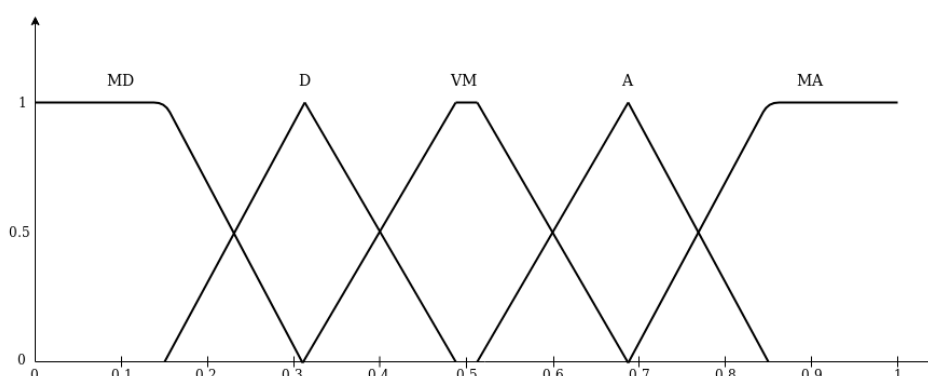
Note que o termo “estável” desta variável de entrada também é baseado em uma função de pertinência trapezoidal. Assim como na entrada “Erro”, este intervalo de -5 a 5 foi inserido para fornecer maior liberdade para o robô, pois entenderá que o objetivo de estabilizar o desvio do erro foi alcançado e o controlador não modificará a velocidade das rodas.

Para a saída do controlador *Fuzzy* foram criadas duas variáveis linguísticas, sendo elas: “RodaDireita” e “RodaEsquerda”. As funções de pertinência e o alcance (*range*) das saídas são iguais. No entanto, na construção das regras, as saídas são combinadas de forma inversa para que o robô possa efetuar curvas, exceto quando estiver em linha reta.

O alcance representa o intervalo de velocidade que as rodas direita e esquerda podem alcançar. Neste caso, a velocidade máxima é 1 m/s e a velocidade média pretendida é de 0.5m/s. Estes valores foram determinados de acordo com o desempenho dos motores utilizados.

A Figura 28 representam graficamente as variáveis de saída do controlador *Fuzzy* e seus conjuntos de termos.

Figura 28 - Variáveis de saída do controlador *Fuzzy*.



Fonte: Autoria Própria.

O conjunto de termos das variáveis “RodaDireita” e “RodaEsquerda” são; MD, D, VM, A e MA, representando valores muito devagar, devagar, velocidade moderada, acelerado, muito acelerado.

Como ilustra a Figura 28, foram utilizados funções de pertinência triangulares e retangular para o conjunto de termos das variáveis. O intervalo de alcance de cada termo das variáveis de saída “RodaDireita” e “RodaEsquerda” são listados a seguir:

- Muito devagar (MD): [0 0 0.1517 0.318];
- Devagar (D): [0.1517 0.3183 0.485];
- Velocidade moderada (VM): [0.3183 0.485 0.515 0.682];
- Acelerado (A): [0.515 0.682 0.848];
- Muito acelerado (MA): [[0.682 0.8487 1 1].

No termo velocidade moderada “VM” também foi utilizado uma função de pertinência trapezoidal para dar mais liberdade para o robô.

Como as duas entradas do controlador *Fuzzy* são compostas por 5 funções de pertinência cada e as saídas do sistema também são compostas por 5 termos cada, mas são acionadas simultaneamente, o conjunto de regras para o controlador resultou em um conjunto de 25 regras.

Como o robô é movido por direção diferencial, ou seja, a direção do mesmo é controlada pela diferença na rotação das rodas, quando a velocidade de uma das rodas aumentar, a velocidade da outra deve diminuir ou permanecer inalterada para

que o robô possa efetuar curvas. Para que o robô se desloque em linha reta é necessário que ambas as rodas girem em mesma velocidade.

Quando a velocidade de ambas as rodas forem a mesma, mas com sentido oposto, o robô girará em círculos. Também é possível fazer curvas com o robô mudando a direção de giro de uma das rodas com velocidade diferente da outra. No entanto, para facilitar o controle neste trabalho, essas duas possibilidades foram descartadas. Dessa forma, as rodas giram apenas em uma direção, mas com velocidades diferentes quanto for necessário.

Na Figura 29 é possível ver as 25 regras criadas para o controlador *Fuzzy* levando em conta as premissas de um robô com direção diferencial e as velocidades necessárias para que o mesmo possa andar em linha reta e fazer curvas dentro do limite de velocidade suportado pelos motores e conjuntos de caixas de redução.

Figura 29 - Conjunto de regras.

1:	Se (Erro é Zero) e (DesvioErro é Estavel)	então (RodaDireita é VM) e (RodaEsquerda é VM)
2:	Se (Erro é Zero) e (DesvioErro é Pneg)	então (RodaDireita é VM) e (RodaEsquerda é VM)
3:	Se (Erro é Zero) e (DesvioErro é Ppos)	então (RodaDireita é VM) e (RodaEsquerda é VM)
4:	Se (Erro é Zero) e (DesvioErro é Mneg)	então (RodaDireita é VM) e (RodaEsquerda é D)
5:	Se (Erro é Zero) e (DesvioErro é Mpos)	então (RodaDireita é D) e (RodaEsquerda é VM)
6:	Se (Erro é Pneg) e (DesvioErro é Estavel)	então (RodaDireita é VM) e (RodaEsquerda é D)
7:	Se (Erro é Pneg) e (DesvioErro é Pneg)	então (RodaDireita é A) e (RodaEsquerda é D)
8:	Se (Erro é Pneg) e (DesvioErro é Mneg)	então (RodaDireita é A) e (RodaEsquerda é MD)
9:	Se (Erro é Pneg) e (DesvioErro é Ppos)	então (RodaDireita é A) e (RodaEsquerda é VM)
10:	Se (Erro é Pneg) e (DesvioErro é Mpos)	então (RodaDireita é VM) e (RodaEsquerda é D)
11:	Se (Erro é Ppos) e (DesvioErro é Estavel)	então (RodaDireita é D) e (RodaEsquerda é VM)
12:	Se (Erro é Ppos) e (DesvioErro é Pneg)	então (RodaDireita é D) e (RodaEsquerda é A)
13:	Se (Erro é Ppos) e (DesvioErro é Mneg)	então (RodaDireita é MD) e (RodaEsquerda é A)
14:	Se (Erro é Ppos) e (DesvioErro é Ppos)	então (RodaDireita é VM) e (RodaEsquerda é A)
15:	Se (Erro é Ppos) e (DesvioErro é Mpos)	então (RodaDireita é D) e (RodaEsquerda é VM)
16:	Se (Erro é Mneg) e (DesvioErro é Estavel)	então (RodaDireita é VM) e (RodaEsquerda é MD)
17:	Se (Erro é Mneg) e (DesvioErro é Pneg)	então (RodaDireita é A) e (RodaEsquerda é MD)
18:	Se (Erro é Mneg) e (DesvioErro é Mneg)	então (RodaDireita é MA) e (RodaEsquerda é MD)
19:	Se (Erro é Mneg) e (DesvioErro é Ppos)	então (RodaDireita é MA) e (RodaEsquerda é D)
20:	Se (Erro é Mneg) e (DesvioErro é Mpos)	então (RodaDireita é MA) e (RodaEsquerda é VM)
21:	Se (Erro é Mpos) e (DesvioErro é Estavel)	então (RodaDireita é MD) e (RodaEsquerda é VM)
22:	Se (Erro é Mpos) e (DesvioErro é Pneg)	então (RodaDireita é MD) e (RodaEsquerda é A)
23:	Se (Erro é Mpos) e (DesvioErro é Mneg)	então (RodaDireita é VM) e (RodaEsquerda é MA)
24:	Se (Erro é Mpos) e (DesvioErro é Ppos)	então (RodaDireita é D) e (RodaEsquerda é MA)
25:	Se (Erro é Mpos) e (DesvioErro é Mpos)	então (RodaDireita é MD) e (RodaEsquerda é MA)

Fonte: Autoria Própria.

O método de inferência utilizado neste trabalho para a aplicação do conjunto de regras foi o método Mamdani. Esse método tem como base a regra de composição de inferência max-min como foi apresentado no Capítulo 2.

Após as informações serem processadas na etapa de inferência, a ação recomendada pelo controlador *Fuzzy* precisa ser decodificada do universo *Fuzzy* para

um valor real a ser utilizado pela etapa do controlador PID. Essa decodificação ocorre na etapa de Defuzzificação.

O método de Defuzzificação empregado foi o centro de gravidade (centroide). O centro de gravidade retorna a média das áreas de todas as figuras que representam os graus de pertinência de um subconjunto *Fuzzy*.

A saída do controlador *Fuzzy* após a decodificação é um valor de 0 a 1 que representa a velocidade em metros por segundo (m/s) que cada roda do AGV pode desempenhar. No entanto, para garantir um melhor controle do AGV, essa informação não é aplicada diretamente nas rodas do robô. Os valores de saída do controlador *Fuzzy* são enviados para um controlador PID. Neste controlador o valor de 0 a 1 fornecidos pelo *Fuzzy* servem como referência para controlador PID.

A necessidade de um controlador PID, sua implementação e sintonia são apresentadas na seção 3.5.

3.5 Controlador PID

Mesmo os motores e as caixas de redução terem sido comprados do mesmo fabricante e com mesmas especificações, na prática, elas apresentam pequenas diferenças que impedem que o robô siga, por exemplo, uma linha reta, com valores de tensão iguais sendo aplicados em cada motor.

Para corrigir este problema, foram desenvolvidos dois controladores PID, um para cada motor, que recebem como referência (*setpoint*), os valores fornecidos pelo controlador *Fuzzy*. Dessa forma é possível garantir a velocidade requerida para que o robô siga em linha reta ou efetue curvas.

Os controladores PID são sintonizados utilizando o método AMIGO, apresentado no capítulo 2, para isso é necessário a captura da curva de resposta ao degrau dos motores. Isso é feito de forma individual usando os motores com as caixas de redução, rodas e ponte H.

Para obter as curvas de resposta ao degrau dos motores, os mesmos são submetidos a um valor de tensão de 5V. O sinal é enviado diretamente do *Raspberry PI* para a ponte H, assim como as curvas também são armazenadas no cartão de memória do *Raspberry PI*.

3.6 Cálculos da odometria

Os cálculos de odometria para encontrar a posição do robô em um plano cartesiano são baseados nas equações apresentadas no Capítulo 2. De acordo com essas equações é possível reconstruir graficamente o percurso realizado pelo robô. No entanto, para que essas equações permitam recriar o trajeto do robô, é necessário adicionar as características do robô e adaptá-las para funcionarem em um algoritmo. Dessa forma, as equações citadas podem ser reescritas da seguinte forma.

O deslocamento linear realizado pela roda direita e pela roda esquerda são calculados pelas Equações 31 e 32 respectivamente:

$$Deslrd = Cd \cdot \frac{Npulsos}{341} \quad (31)$$

$$Deslre = Ce \cdot \frac{Npulsos}{341} \quad (32)$$

em que Cd e Ce são os comprimentos das circunferências de cada roda, $Npulsos$ é o número de pulsos fornecidos pelo *encoder* e 341 é a quantidade de pulsos que o *encoder* fornece para cada volta completa.

O deslocamento angular ou ângulo de giro do robô é encontrado pela Equação 33:

$$\theta_{rob\hat{o}} = \frac{Deslrd - Deslre}{2b} \quad (33)$$

em que $2b$ é a distância entre as rodas do robô e neste caso é 19cm.

Já o deslocamento linear do robô é encontrado de acordo com a Equação 34:

$$V_{rob\hat{o}} = \frac{Deslrd + Deslre}{2} \quad (34)$$

Com base nas Equações 33 e 34, é possível encontrar a posição do robô utilizando as seguintes aproximações para o deslocamento angular acumulado e para as componentes x e y:

$$\theta(n+1) = \theta(n) + \theta_{robo} \quad (35)$$

$$x(n+1) = x(n) + V_{rob\hat{o}} \cdot \cos(\theta(n)) \quad (36)$$

$$y(n+1) = y(n) + V_{rob\hat{o}} \cdot \sin(\theta(n)) \quad (37)$$

onde n é o número de iterações e começa em zero.

Os valores iniciais de $x(0)$, $y(0)$ e $\theta(0)$ são zero, mas podem ser modificados para outro valor de referência.

A odometria é um método bastante utilizado para encontrar a posição de um robô móvel, entretanto ela não é um método exato e é passível de erros sistemáticos e erros não sistemáticos (Prestes, 2019).

Os principais erros sistemáticos que acometem a odometria são: rodas com diâmetros diferentes ou desalinhadas, erros no *encoder*, entre outros erros de parâmetros. Já os principais erros não sistemático são: percursos irregulares, deslizamento das rodas e sujeira.

Os erros sistemáticos são os mais críticos, pois estão relacionados aos componentes do robô e nem sempre podem ser corrigidos, gerando uma fonte incremental de erro.

3.7 Softwares e bibliotecas

Neste trabalho foi utilizado OpenCV em todas as etapas de visão computacional e processamento de imagens. O OpenCV é um conjunto de bibliotecas multiplataforma gratuito e destinado a aplicações envolvendo visão computacional. Foi utilizada a versão em linguagem C++ do OpenCV.

A escolha deste pacote de bibliotecas foi motivada pela quantidade de conteúdo disponível na internet e sua frequente utilização em cursos acadêmicos de visão computacional ou processamento digital de imagens, como por exemplo, em Marengoni e Stringhini (2009).

Para implementar o controlador *Fuzzy* no robô é necessário transformar todas as etapas de desenvolvimento do controlador em um algoritmo em C/C++. Para isso foi utilizada a biblioteca eFLL (*Embedded Fuzzy Logic Library*), uma opção nacional desenvolvida pelo *Robotic Research Group* na Universidade Estadual do Piauí. A eFLL é escrita em C/C++ e necessita apenas da biblioteca *stdlib.c* padrão. A eFLL utiliza o processo (MAX-MIN) e (Mínimo de Mamdani) para a inferência. O trabalho de Krid et al (2013), apresenta a biblioteca eFLL.

No Raspberry Pi foi utilizado o sistema operacional Linux Ubuntu, dessa forma foi possível desenvolver os controladores e demais algoritmos em C/C++ para funcionarem internamente.

É necessária a instalação do sistema operacional e outras configurações para que o *Raspberry Pi* possa funcionar corretamente. Tais configurações podem ser encontradas no site da *Fundação Raspberry Pi* e não serão abordadas neste trabalho.

No *Raspberry Pi* foi utilizado a biblioteca *WiringPi*, para controle dos pinos de entrada e saída de uso geral (GPIO) e também a construção das rotinas de interrupção necessárias para capturar, de forma eficiente, os pulsos do *encoder*. Esta biblioteca é nativa do *Raspbian*, sistema operacional padrão do *Raspberry Pi* baseado na distribuição Linux Debian, mas não do Ubuntu. As etapas de criação destas rotinas e configurações do *WiringPi* também não serão detalhadas neste trabalho, devido sua grande popularidade.

Neste capítulo foram apresentadas todas as etapas da metodologia, os materiais, *softwares* e bibliotecas necessários para o desenvolvimento do trabalho. No capítulo 4 são apresentados os resultados obtidos.

4 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os resultados e discussões provenientes deste trabalho. Os resultados serão apresentados de acordo com a ordem apresentada na metodologia.

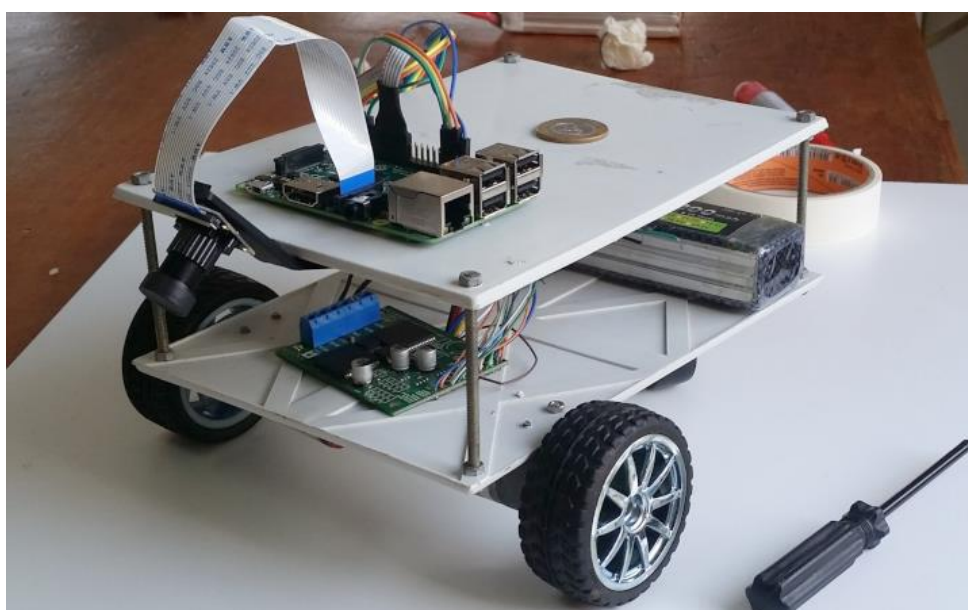
4.1 Estrutura e montagem do robô

Como foi explicado no capítulo 3, o robô móvel utilizado é movido por tração diferencial, ou seja, a direção e velocidade do robô são controladas pela diferença de velocidade das rodas principais. Na Figura 30 é possível ver a versão do robô montado, com a câmera inclinada em 45° para uma melhor captura de imagem do percurso.

A bateria foi fixada na parte de trás do robô, sobre a roda castor. Já a bateria foi colocada nesta posição para balancear o robô e facilitar a remoção.

A ponte H ficou fixada na parte inferior junto com a bateria, conversor *buck* que alimenta o *Raspberry Pi* e chave de acionamento. Já na parte superior ficou somente o *Raspberry Pi* e a câmera.

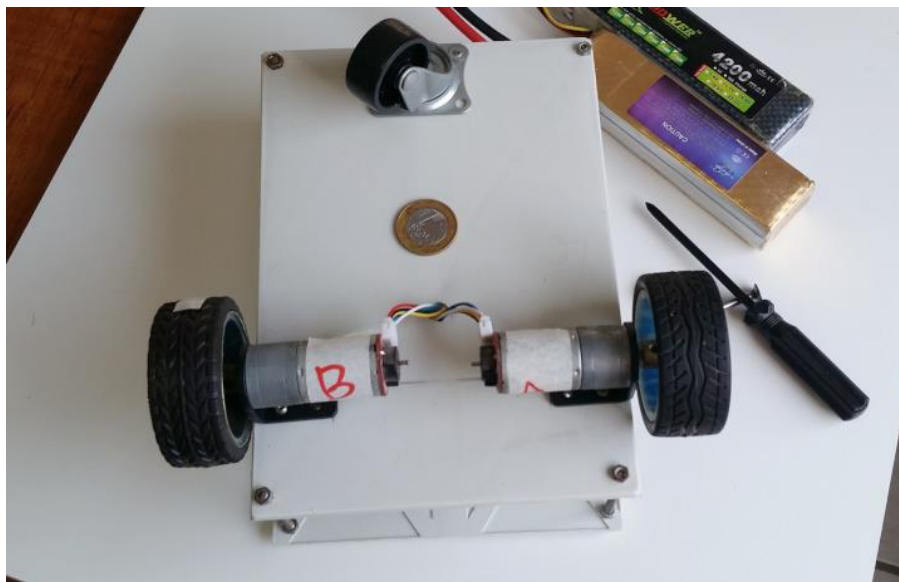
Figura 30 - Versão final do robô móvel.



Fonte: Autoria própria.

A disposição final dos motores, caixas de redução e das rodas do robô móvel podem ser vistas na Figura 31.

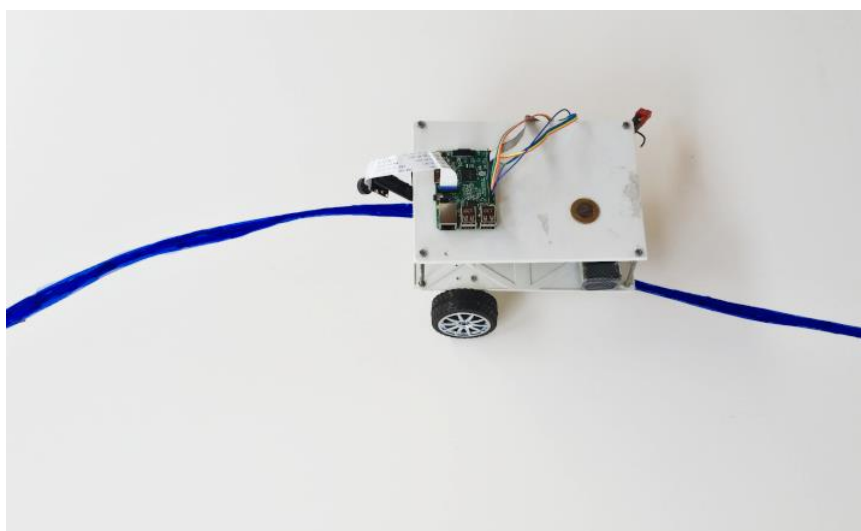
Figura 31 - Parte inferior do robô.



Fonte: Autoria própria.

Os motores e conjunto de caixas de redução foram fixados na parte inferior da estrutura de forma alinhada. A parte com os motores é a frente do robô, já a roda castor ficou centralizada na parte de trás. A Figura 32 mostra como o robô fica disposto sobre a trilha a ser seguida.

Figura 32 - AGV sobre a trilha a ser seguida.



Fonte: Autoria própria.

4.2 Conferência dos valores nominais dos Motores.

Os motores foram checados para conferir se os valores referentes ao PPR e RPM fornecidos pelo fabricante estão de acordo. Inicialmente os motores foram configurados para girarem a 60 RPM, ou seja, 1 RPS, para checar a quantidade de PPR por segundo. Para ter certeza que os motores estavam girando a aproximadamente 60 RPM foi utilizado um tacômetro digital, como ilustra a Figura 33.

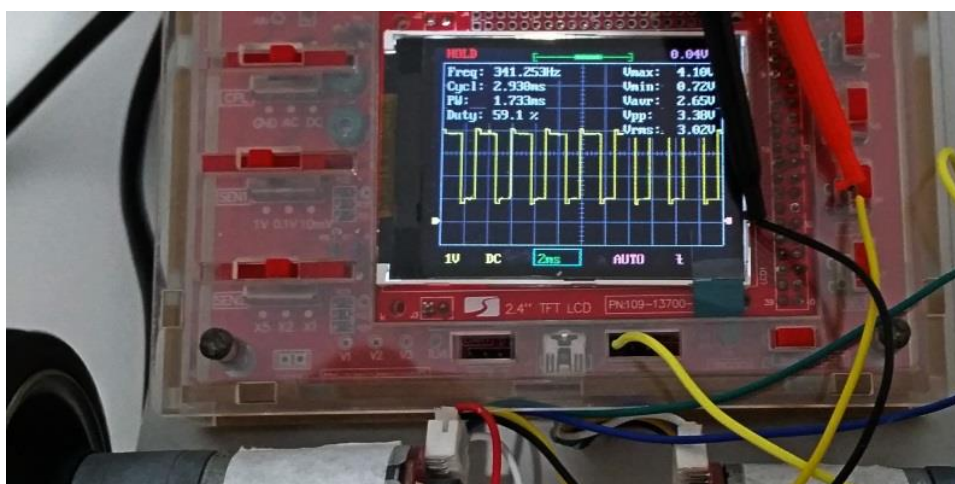
Figura 33 - Tacômetro medindo o RPM do Motor B



Fonte: Autoria própria.

Foi utilizado um osciloscópio portátil para obter o valor de PPR por segundo para cada volta completa realizada pela roda. A Figura 34 mostra o resultado obtido.

Figura 34 - PPR por segundo Motor B

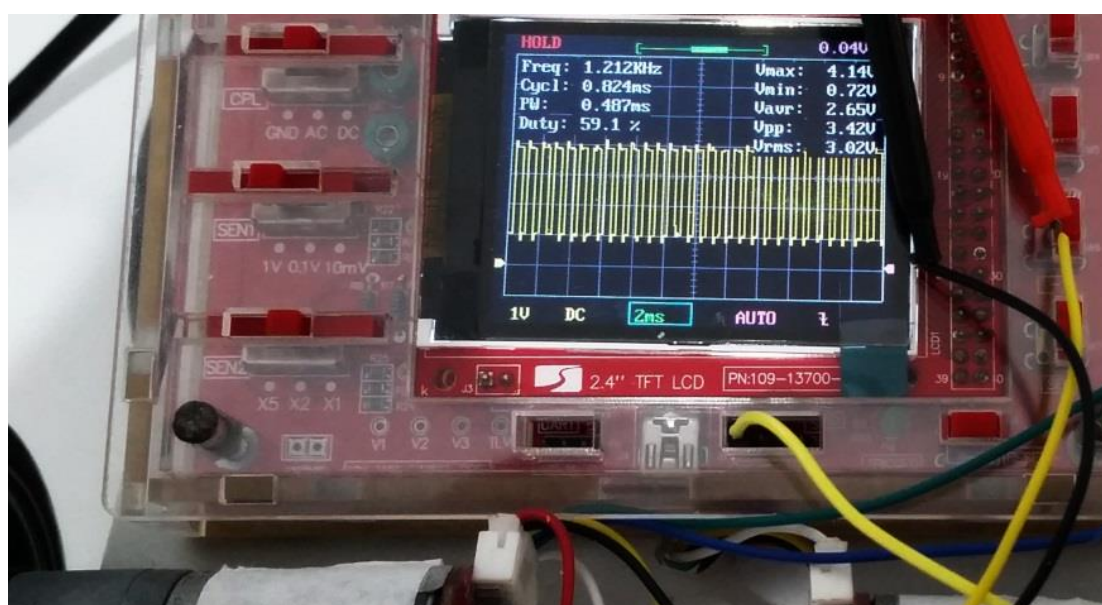


Fonte: Autoria própria.

O valor encontrado é informado na forma de frequência no osciloscópio. Como a roda está girando a 1 RPS, é possível concluir que para cada volta o encoder realiza 341.253 pulsos por segundo. Valor muito próximo do fornecido pelo fabricante, que é de 341 PPR.

A Figura 35 mostra a quantidade de pulsos que o *encoder* fornece ao ser aplicado 6V no Motor B.

Figura 35 - PPR a 6V - Motor B.



Fonte: Autoria própria.

De acordo com a Figura 35, ao ser aplicado 6V no motor, o encoder forneceu aproximadamente 1212 pulsos por segundo.

Note que os valores de tensão mostrados no osciloscópio são referentes aos pulsos do encoder e não ao valor de tensão aplicado no motor. O encoder utilizado opera em tensão nominal de 3.3V como ilustra a Figura 14 no Capítulo 3.

Como o valor de PPR encontrado foi bastante próximo do fornecido pelo fabricante, o valor de 341 PPR foi utilizado como base neste trabalho.

Ao dividir 1 por 341 temos que, para cada pulso, a roda gira 0,0029325 vezes. Então, se o encoder pulsar 1.200 vezes por segundo a roda vai girar aproximadamente 3,5 RPS, ou seja, 210 RPM, que é o valor nominal de RPM fornecido pelo fabricante na tensão de 6V.

O motor consegue funcionar até 9V sem esquentar ou ter um aumento significativo de corrente. Para o valor de 9V o motor conseguiu girar a aproximadamente 317 RPM, como mostra a Figura 36.

Figura 36 - RPM do Motor B a 9V.



Fonte: Autoria própria.

Dessa forma, o valor de 9V foi configurado como o limite de tensão que a ponte H pode fornecer ao motor, uma vez que a bateria utilizada é de 12V. O limite de tensão foi inserido nos controladores PID.

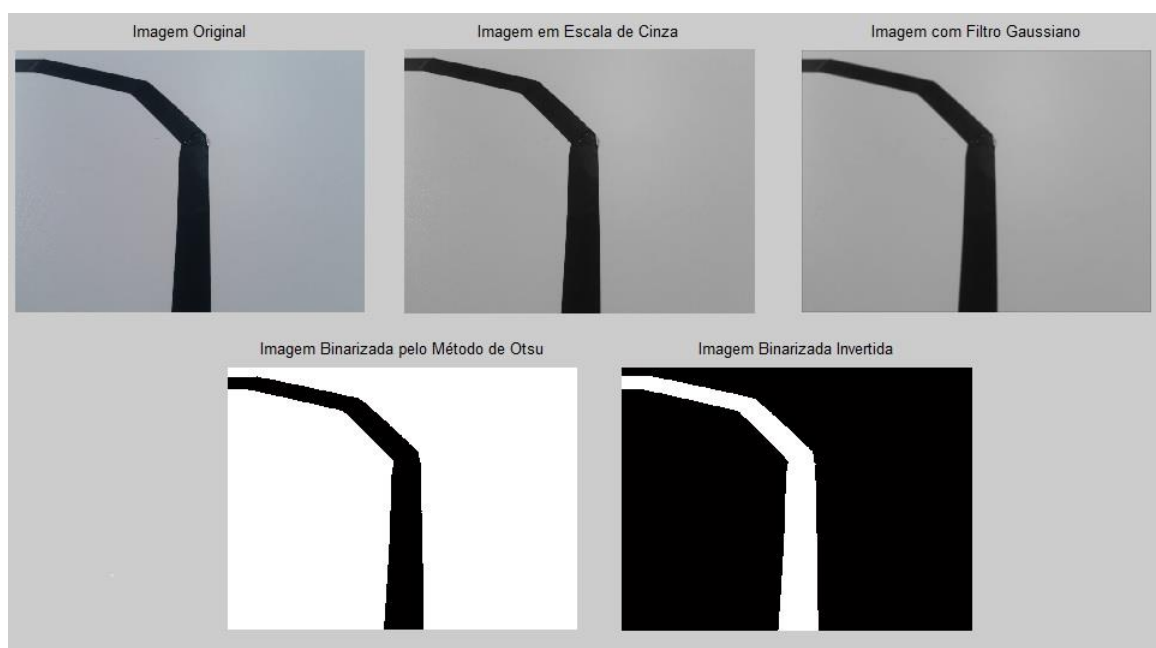
O mesmo processo foi efetuado para o Motor A e foram encontrados valores aproximados do Motor B para o valor de PPR a 1 RPS, porém, o Motor A é mais veloz que o Motor B a um mesmo valor de tensão, pois a 6V foram contados 1232 pulsos.

4.3 Visão computacional e processamento de imagem

A etapa de visão computacional é a etapa responsável pela entrada principal do sistema de controle apresentado no Capítulo 3, ou seja, é uma das etapas fundamentais deste trabalho.

A Figura 37 mostra os resultados encontrados nas fases de processamento de imagem tomando como base a imagem original.

Figura 37 - Resultados obtidos com as etapas de visão.

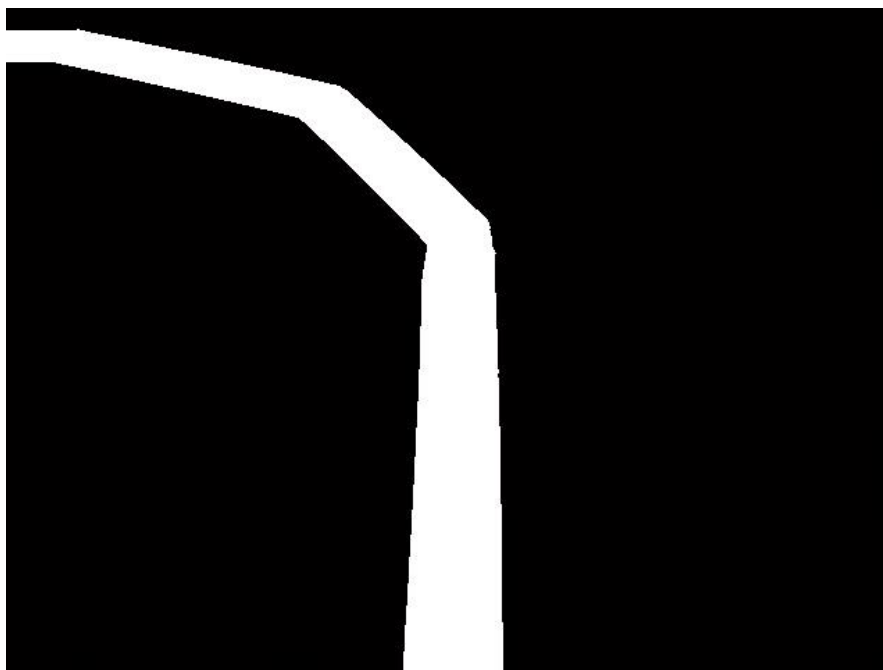


Fonte: Autoria própria.

Na Figura 37 é possível constatar a diferença entre a imagem original e a imagem final binarizada pelo Método de Otsu. Note que para obter a imagem binarizada foi necessário a conversão da imagem para a escala de cinza e a aplicação de um filtro gaussiano. O filtro foi fundamental para eliminar pequenos ruídos presentes na imagem, pois como o resultado é uma matriz de zeros e uns, qualquer ruído pode ser transformado em um 0 ou 1 durante a binarização.

Após a imagem ser binarizada, a mesma foi submetida a uma função de complemento. Essa função teve o objetivo de inverter a imagem, onde era preto passou a ser branco e onde era branco passou a ser preto, ou seja, os zeros e uns foram trocados por seus complementos. A Figura 38 ressalta a imagem binarizada invertida.

Figura 38 - Imagem binarizada pelo método de Otsu e invertida.

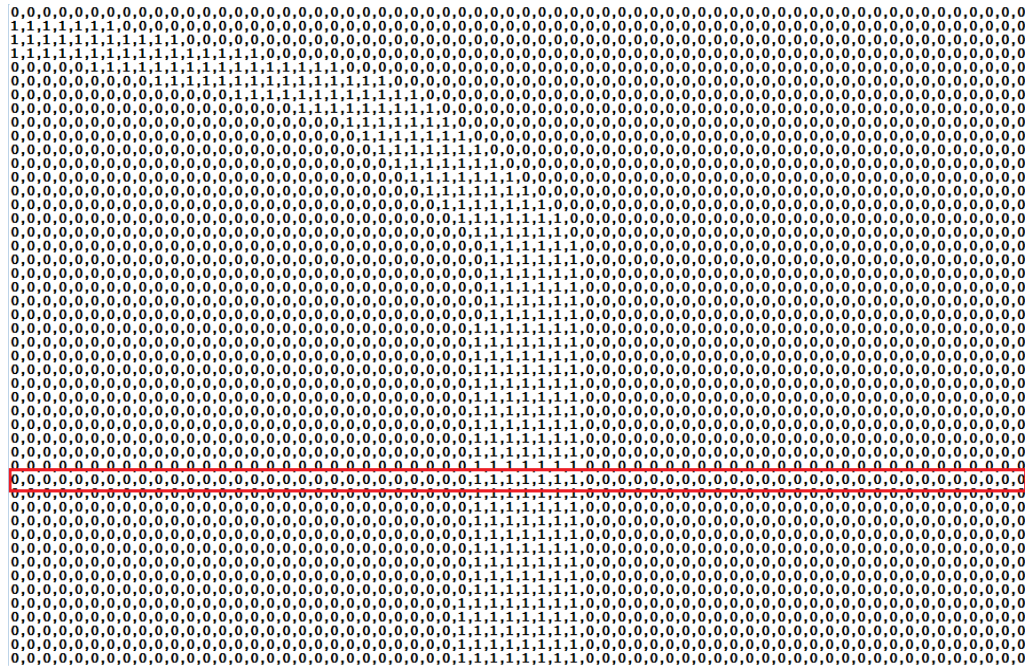


Fonte: Autoria própria.

Essa etapa foi necessária, pois a cor branca é representada por 1 após a binarização pelo método de Otsu, mas como na imagem original a trilha possui uma cor mais escura que o restante da imagem, após a aplicação do método, a trilha ficou representada por zeros e o restante por uns. Por este motivo, a imagem foi complementada (invertida) para que a trilha represente os uns na matriz de pixels binarizada e o restante seja representado pelos zeros. Caso a trilha na imagem original já fosse branca, essa etapa não seria necessária. A cor branca no centro da imagem facilita a aplicação do algoritmo para achar o centro da imagem.

O resultado final é uma matriz de zeros e uns como mostra a Figura 39, sendo a parte central composta por uns e o restante por zeros. Para fins didáticos e para a apresentação dos resultados, as dimensões da imagem binarizada invertida foi reduzida para 64x48 antes de gerar a matriz de zeros e uns. A redução foi feita devido a imagem original possuir 640x480 pixels, ou seja, uma figura com uma matriz de 640 colunas e 480 linhas, que seria impossível inserir neste trabalho sem perda de detalhes. No entanto, na prática, as imagens utilizadas no processamento são sempre em 640x480 pixels.

Figura 39 - Matriz de zeros e uns.



Fonte: Autoria própria.

Como explicado no Capítulo 3, a estratégia utilizada foi escolher uma linha fixa de cada imagem e, através deste vetor, aplicar a equação da média ponderada para achar o centro do vetor e automaticamente achar o centro da linha branca, parte central da trilha a ser seguida pelo robô.

Pela equação 38, a média ponderada aplicada ao vetor formado pela linha em destaque na Figura 39 é igual a 33. Na Equação 38 n é igual a 64, pois a matriz formada pela Figura 39 tem 64 colunas, x_i é o número dos pixels, que vai de 1 a 64 e p_i é o peso de cada pixel, que neste caso só pode ser 0 ou 1.

$$Mp = \frac{\sum_{i=1}^n x_i \cdot p_i}{\sum_{i=1}^n p_i} = \frac{\sum_{i=1}^{64} x_i \cdot p_i}{\sum_{i=1}^{64} p_i} = \frac{231}{7} = 33 \tag{38}$$

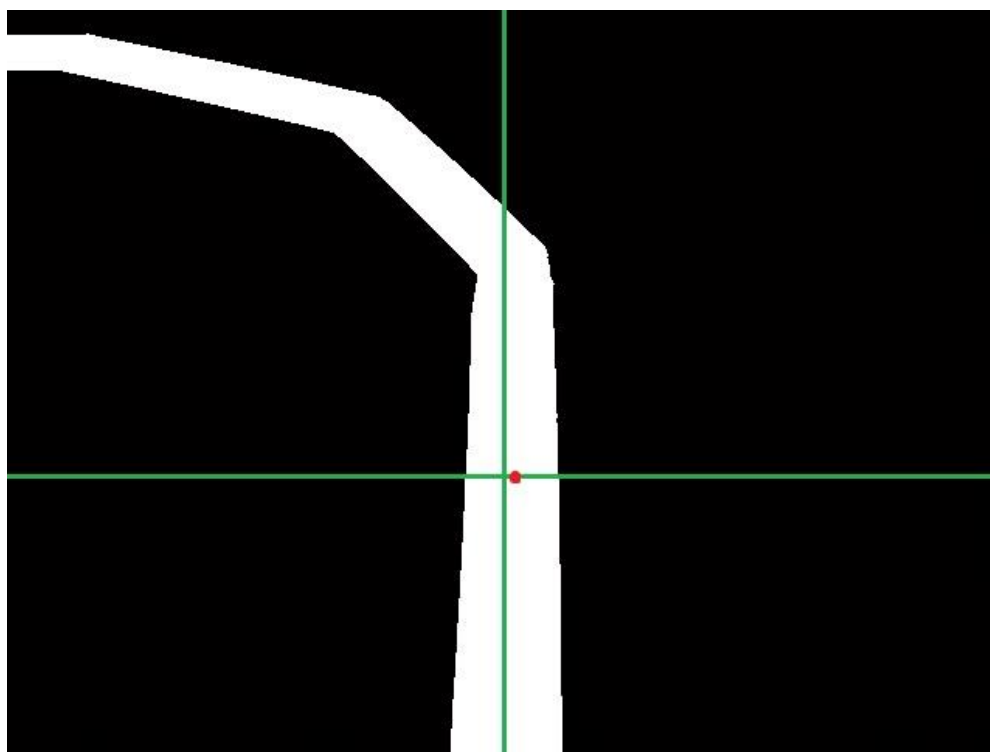
Na linha 14, em destaque, tem somente 7 uns, por este motivo a somatória dos pesos é 7. Já a soma da multiplicação dos números de pixels pelo seus respectivos pesos (0 ou 1) é igual a 231. Dessa forma, o centro da linha para essa imagem é o pixel 33.

O valor encontrado é utilizado para descobrir o erro e o desvio do erro necessários para o controlador *Fuzzy*. O erro é encontrado subtraindo o valor do pixel

central da imagem, que é fixo, pelo valor 33 encontrado para este exemplo. Como o valor fixo para o pixel central dessa imagem reduzida é 32, o erro para este exemplo é igual a -1.

Na imagem em resolução completa, o algoritmo de visão computacional encontraria o seguinte resultado ilustrado pela Figura 40.

Figura 40 - Resultado do algoritmo de visão.



Fonte: Autoria própria.

Na Figura 40, o ponto vermelho representa o resultado da média ponderada encontrada para a linha horizontal. Já o erro é a diferença entre a linha central e o ponto vermelho, ou ainda, o erro é a diferença entre o ponto formado pelo encontro das linhas, vertical e horizontal verdes, e a média ponderada da linha horizontal.

A trilha ou percurso a ser seguido pelo robô não precisa ser muito larga para que o sistema de visão consiga atuar. A trilha da Figura 40 possui aproximadamente 2cm. No entanto, trilhas com espessura mais finas também podem ser utilizadas. No entanto, quanto maior a espessura da trilha mais fácil é para o algoritmo de visão detectar o centro da linha.

A Figura 41 mostra um percurso circular de trilha com espessura entre 0.6 e 0.8 cm aproximadamente. A trilha foi feita em um piso de azulejo branco com um pincel de quadro branco. Esse tipo de trajeto é mais fácil de ser desenhado em comparação com o da Figura 40, que é feita com fita isolante

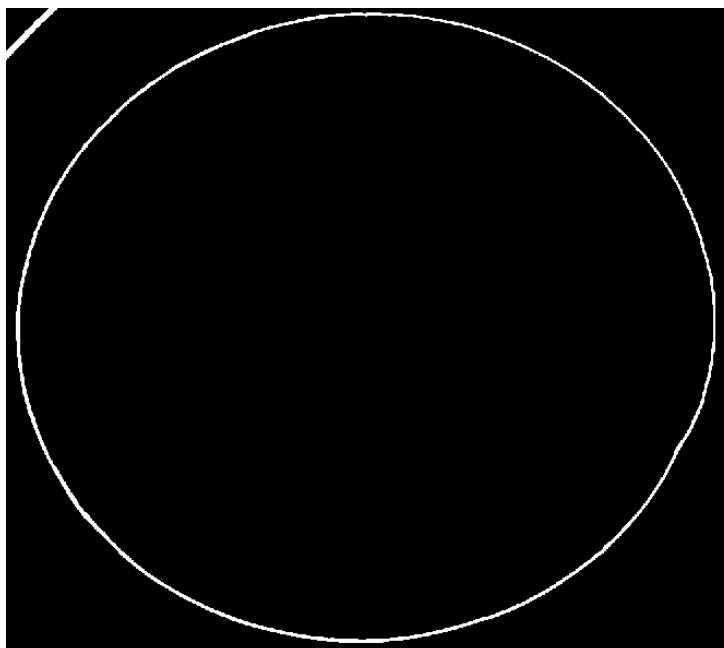
Figura 41 - Trilha circular feita com pincel de quadro.



Fonte: Autoria própria.

O resultado das etapas de processamento de imagem feita na Figura 41 pode ser vista na Figura 42. Note que a imagem binarizada não apresenta nenhum ruído na trajetória, mesmo as separações dos azulejos desaparecem na imagem binarizada, o que comprova a eficiência do filtro Gaussiano e da binarização pelo método de Otsu.

Figura 42 - Imagem binarizada e complementada/invertida.



Fonte: Autoria própria.

A Figura 42 representa um vista superior do trajeto a ser percorrido pelo robô, mas não é utilizada durante o processo. Esta figura serve apenas para mostrar o percurso final e comprovar a efetividade do método em segmentar trilhas com espessuras mais finas.

Do percurso mostrado pela Figura 42, o que é capturado e utilizado pelo robô são as imagens a) e b) da Figura 43.

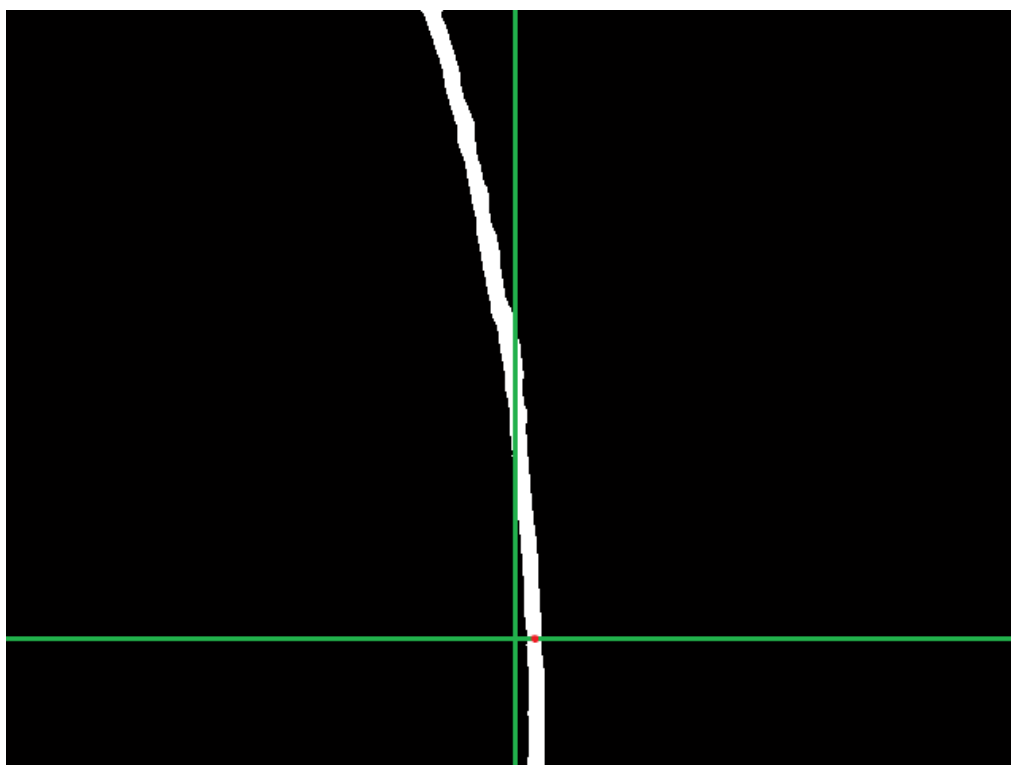
Figura 43 - Imagem capturada pelo robô.



Fonte: Autoria própria.

A Figura 43 a) mostra uma pequena parte do trajeto representado pela Figura 42 e a Figura 43 b) mostra a imagem binarizada e invertida. Já a Figura 44 mostra o ponto central da linha branca (ponto vermelho), fornecida pela média ponderada da linha guia horizontal, além do ponto de intercessão entre as guias horizontal e vertical (verdes). A diferença entre estes pontos é o erro a ser corrigido pelos controladores para que os pontos se coincidam.

Figura 44 - Ponto central da linha branca.



O algoritmo repete a estratégia para cada imagem (frame do vídeo) capturada pela câmera. A câmera foi configurada para uma taxa de 30 FPS. O erro e o desvio do erro são fornecidos para a entrada do controlador *Fuzzy*.

4.4 Controlador *Fuzzy*

Conforme demonstrado durante a metodologia, o controlador *Fuzzy* desenvolvido nesta dissertação é composto de duas variáveis linguísticas de entrada e duas de saída, para cada variável foi criado 5 conjuntos de termos representados por funções de pertinência do tipo triangular e trapezoidal. Para o controlador foram obtidas 25 regras. A Tabela 4 mostra todas as regras obtidas para o controlador.

Tabela 4 - Conjunto de regras do *Fuzzy*.

De	E	Mneg	Pneg	Zero	Ppos	Mneg
Mneg		MD MD	MD A	D VM	A MD	MA VM
Pneg		MD A	D A	VM VM	A D	A MD
Estável		MD VM	D VM	VM VM	VM D	VM MD
Ppos		D MA	VM A	VM VM	A VM	MA D
Mpos		VM MA	D VM	VM D	VM D	MA MD

Fonte: Autoria Própria.

Note que o controlador possui duas variáveis de saídas idênticas, ou seja, com mesmos conjuntos de termos, universo de discurso e graus de pertinência. No entanto, durante a criação das regras as saídas foram escolhidas de formas opostas, exceto para os casos em que é desejado que o robô se desloque em linha reta.

A Figura 45 mostra as saídas do controlador *Fuzzy* para um cenário onde as entradas, Erro e Desvio do Erro, são iguais a zero.

Figura 45 - Saídas do *Fuzzy* para entradas 0 e 0.

```

FuzzyAGV - Teste
-----
Insira um valor para o Erro:
0
Insira um valor para o Desvio do Erro:
0
-----
Saídas:
-----
Roda Direita: 0.5 | Roda Esquerda: 0.5
Press <RETURN> to close this window...
  
```

Fonte: Autoria própria.

O cenário em que o erro e o desvio do erro são zero representa a condição ideal do robô em linha reta, ou seja, ele está completamente alinhado com a trilha a

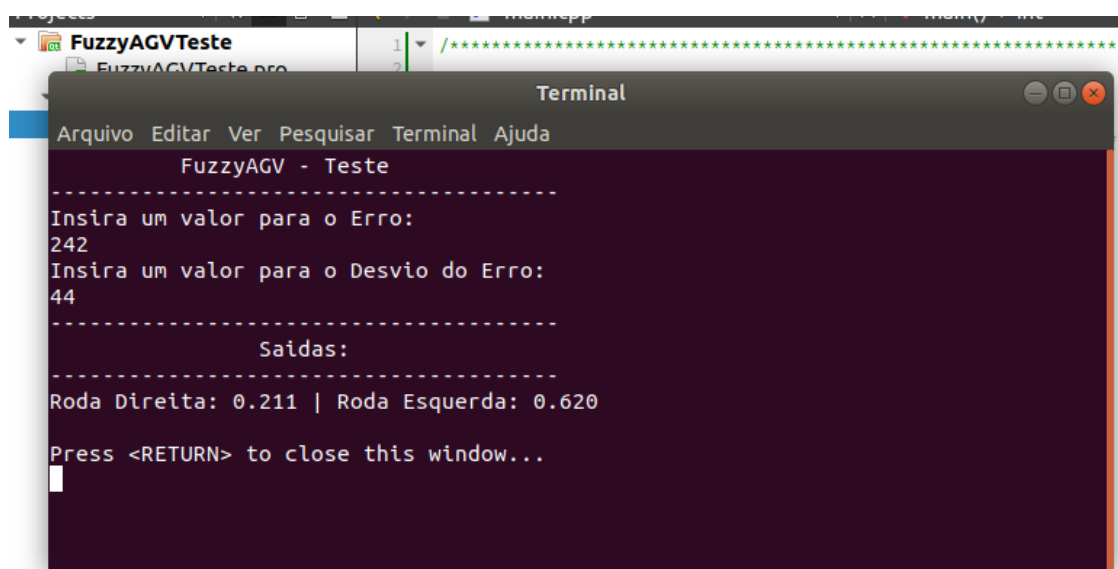
ser seguida, para esta situação é esperado que as rodas girem na mesma velocidade. No caso, a Figura 45 ilustra as saídas iguais.

Como foram obtidas funções de pertinência trapezoidais para os termos “Zero” e “Estável” das variáveis de entrada “Erro” e “Desvio do Erro”, o controlador tem uma certa liberdade para manter o robô alinhado. Dessa forma, essa configuração se repete rotineiramente durante retas perfeitas ou com pouca deformidade. Esse fato não seria possível se na metodologia as funções de pertinência escolhidas fossem, por exemplo, triangulares.

O valor 0.5 é referente à velocidade média de 0.5 m/s, ou seja, neste cenário o robô percorre 50 cm em um segundo. Essa velocidade é relativamente rápida e por isso é destinada para retas.

Um cenário mais extremo para ilustrar as saídas com velocidades diferentes é mostrado na Figura 46.

Figura 46 - Entradas diferente, saídas diferentes.



```
FuzzyAGV - Teste
-----
Insira um valor para o Erro:
242
Insira um valor para o Desvio do Erro:
44
-----
Saídas:
-----
Roda Direita: 0.211 | Roda Esquerda: 0.620
Press <RETURN> to close this window...
```

Fonte: Autoria própria.

Nessa situação o controlador tem como entrada um valor de erro elevado e um desvio do erro pequeno, este cenário é extremo e praticamente só ocorre quando o robô, por algum motivo, está bastante distante da linha central. Neste cenário é então necessário que uma das rodas diminua a velocidade de rotação e a outra aumente significativamente.

Como na Figura 46, o erro indica que o robô está bastante afastado da linha central para a esquerda, é necessário aumentar significativamente a velocidade da roda esquerda e diminuir a da direita, para que o robô gire para a direita e assim volte para o centro. Lembrando que o robô possui tração diferencial e a única forma de mudar sua direção é variando a velocidade das rodas.

Como o robô tem sua trajetória corrigida constantemente, a situação retratada pela Figura 46 só irá ocorrer caso o robô seja ligado nessa posição ou por alguma falha em algum dos sistemas. Do contrário, o controlador não permitirá que o erro chegue a um valor tão elevado. No entanto, a Figura 46 serve como exemplo possível para ilustrar uma situação em que as rodas do robô estão com velocidades de rotação completamente diferentes.

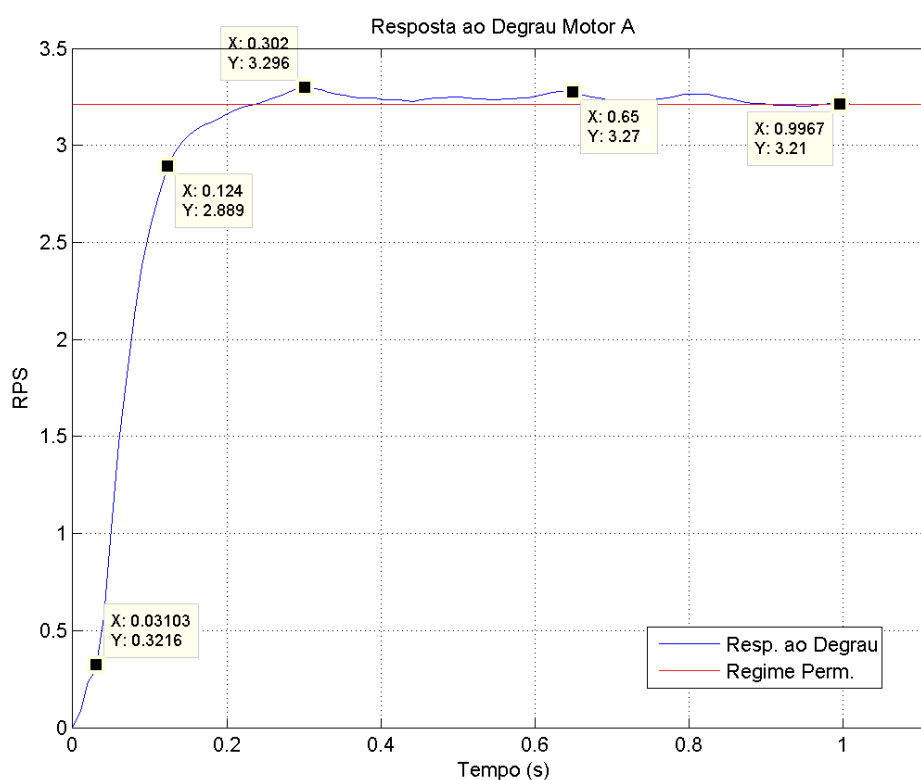
Todas as saídas possíveis do controlador *Fuzzy* não são enviadas diretamente para os motores, como foi explicado durante o Capítulo 3. As saídas servem como valores de referência dos controladores PID, ou seja, é o controlador *Fuzzy* quem gerencia os controladores PID individuais, para que funcionem em conjunto.

4.5 Controladores PID

Como explicado no Capítulo 3, o robô além do controlador *Fuzzy* possui dois controladores PID, um para cada motor. Para a sintonia dos controladores PID foram obtidas as curvas de resposta ao degrau de cada motor. Todas as informações obtidas nessa etapa foram feitas experimentalmente.

As curvas de resposta ao degrau dos motores foram obtidas submetendo os motores, de forma individual, a uma tensão de 5V. Posteriormente, os pulsos gerados pelo *encoder* foram coletados para encontrar o valor de RPS de cada roda. Os motores foram submetidos ao teste com as caixas de redução já acopladas e com as rodas fixadas no eixo das caixas de redução. O teste foi realizado de forma idêntica para ambos os motores. A Figura 47 mostra o resultado obtido para o motor que obteve a marcação A na metodologia.

Figura 47 - Resposta ao degrau do Motor A.



Fonte: Autoria própria.

Com base na Figura 47 foi possível encontrar os parâmetros de desempenho da curva em resposta ao degrau do Motor A. Os resultados são apresentados na Tabela 5.

Tabela 5 - Parâmetros de desempenho Motor A.

Parâmetro	Valor
Tempo de subida (RiseTime):	0,09297 s
Tempo de acomodação (SettlingTime):	0,65 s
Valor em regime (referência)	3,21 RPS
Acomodação +- 2%	0,0642 RPS
Sobressinal (Overshoot):	2,6853 %
Pico (Peak):	3.2962 RPS
Tempo de pico (PeakTime):	0.302 s

Fonte: Autoria própria.

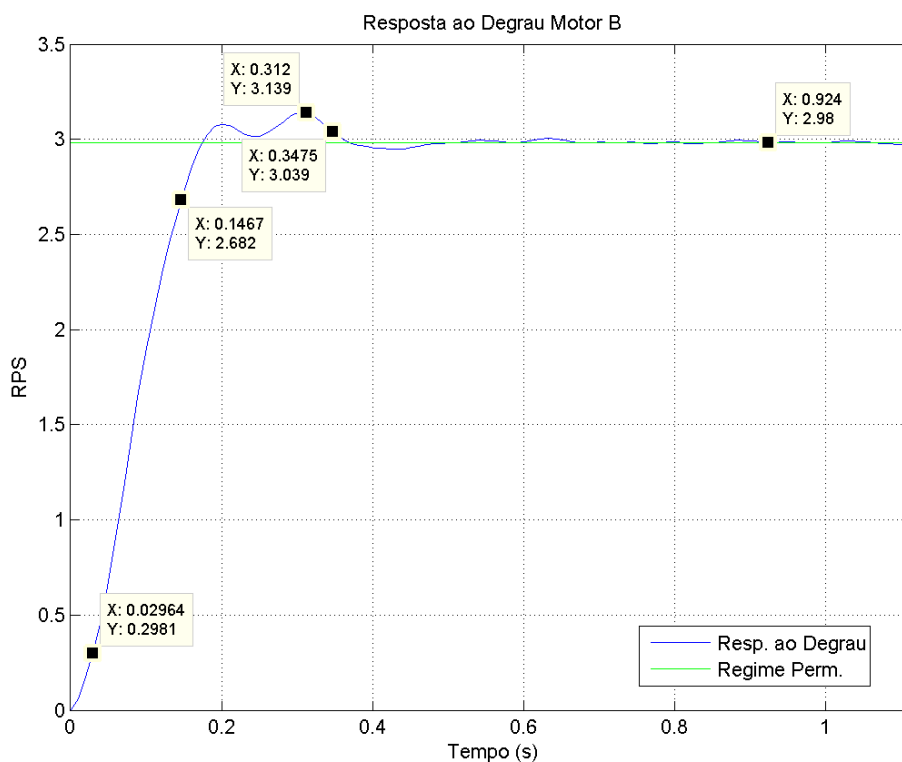
De acordo com a Figura 47 e a Tabela 5, o tempo de subida, ou seja, o tempo para o sinal variar de 10% a 90% do valor em regime foi de 0,09297 segundos. Essa diferença é referente aos dois primeiros pontos da Figura 47. O valor em regime, caracterizado pela linha de referência e pelo último ponto da Figura 47, foi de 3.21 RPS.

Já o tempo de acomodação, ou seja, o tempo que a resposta do degrau leva para estabilizar em um valor de +/- 2% do valor em regime e permanecer no neste intervalo, foi de 0,65 segundos. O valor é referente ao quarto ponto da Figura 47. O tempo que a resposta leva para atingir seu valor máximo, terceiro ponto da Figura 47, foi de 0.302 segundos.

Os dados apresentados para a resposta ao degrau do Motor A mostram que, mesmo sem auxílio de um controlador, o Motor A possui uma boa resposta ao ser submetido a um valor de tensão de 5V, conseguindo atingir o estado estacionário, ou seja, um sinal de saída estável em aproximadamente 0,65 segundos. Este resultado, em parte, é devido a utilização da caixa de redução que ajuda na estabilização do sistema por si só.

Já para a curva de resposta ao degrau do Motor B, representada pela Figura 48, mostra que, por mais que os motores sejam semelhantes e com mesmos valores nominais, os motores apresentam comportamentos diferentes ao serem submetidos ao mesmo sinal de entrada.

Figura 48 - Resposta ao degrau do Motor B.



Fonte: Autoria própria.

Assim como no Motor A, os valores dos principais parâmetros que caracterizam o comportamento da resposta ao degrau também foram coletados para o Motor B. Os resultados obtidos para os parâmetros do Motor B estão listados na Tabela 6.

Tabela 6 - Parâmetros de desempenho motor b.

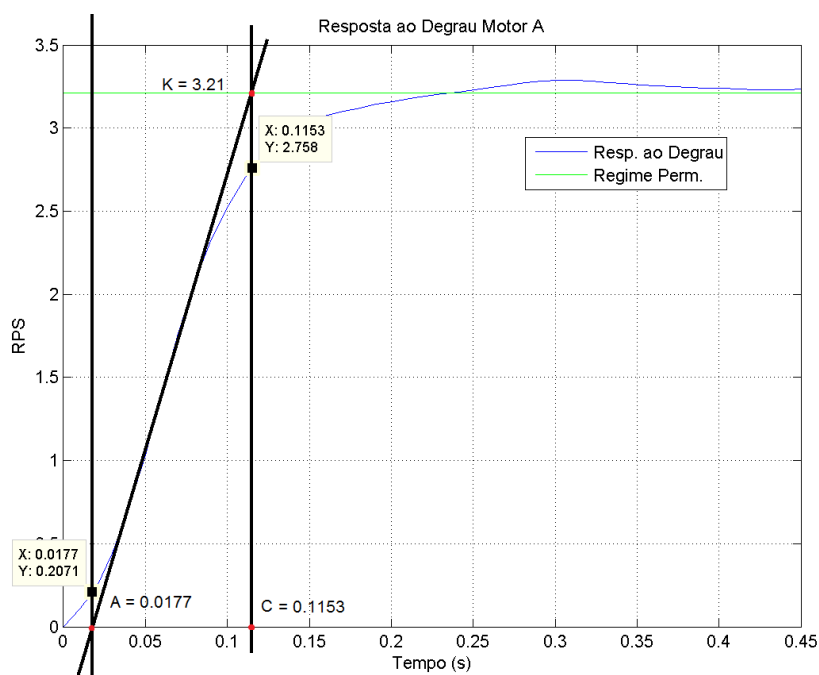
Parâmetro	Valor
Tempo de subida (RiseTime):	0.11706 s
Tempo de acomodação (SettlingTime):	0.3475 s
Valor em regime (referência)	2,97 RPS
Acomodação +- 2%	0.0594 RPS
Sobressinal (Overshoot):	4.7138 %
Pico (Peak):	3.139 RPS
Tempo de pico (PeakTime):	0.302 s

Fonte: Autoria própria.

De acordo com a Tabela 6 e com a Figura 48, o Motor B apresenta tempo de subida de 0.11706 segundos, valor maior que o Motor A. O tempo de subida é referente a diferença entre o segundo e o primeiro ponto da Figura 48. O tempo de acomodação do Motor B é 0.3475 segundos, significativamente menor em relação ao Motor A. A velocidade máxima atingida pelo Motor B é menor que a do Motor A para o mesmo sinal de entrada. Isso é comprovado ao comparar os valores de pico presentes nas Tabelas 5 e 6. A velocidade atingida em estado estacionário no Motor B também é menor, cerca de 2,98 RPS para o Motor B. No entanto, o Motor B possui exatamente o mesmo tempo de pico que o Motor A, levando em consideração as curvas analisadas.

Devido as diferenças entre os motores e a necessidade de controle de velocidade por parte do robô, foi necessário a criação de controladores PID e a sintonia dos mesmos. Para a sintonia dos controladores PID, foram extraídos das curvas de resposta ao degrau os pontos necessários para encontrar as constantes referente ao ganho estático, tempo morto e a constante de tempo, representados pelas letras K, L e T respectivamente. A Figura 49 mostra os resultados com os pontos necessários para a sintonia do PID para o Motor A via o método AMIGO.

Figura 49 – Pontos para sintonia do PID do Motor A



Fonte: Autoria própria.

O ponto K corresponde ao próprio ganho estático e seu valor é 3.21 RPS para o Motor A. Já o ponto A corresponde ao tempo morto L e é igual a 0.0177 segundos. O ponto C é utilizado para encontrar a constante de tempo T, que é a diferença entre C e A. O ponto C é igual a 0.1153 segundos, T é igual a 0.0976 segundos. A Tabela 7 mostra os resultados da sintonia via as regras do método AMIGO.

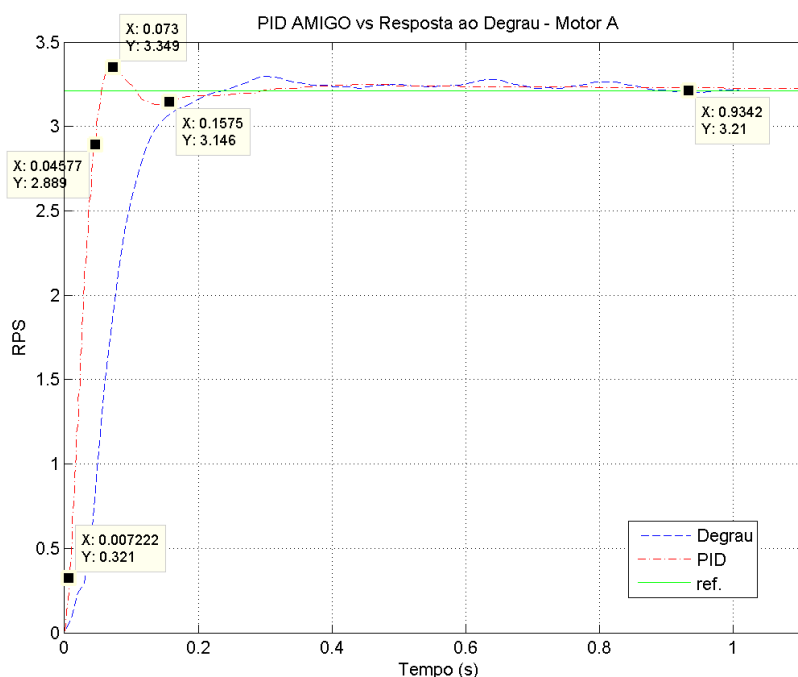
Tabela 7 - Resultados da sintonia AMIGO – Motor A.

Controlador	K_p	τ_i	τ_d
PID	$\frac{1}{K} \left(0,2 + 0,45 \cdot \frac{T}{L} \right)$ = 0,8353	$L \cdot \frac{0,4L + 0,8T}{L + 0,1T}$ = 0,05489 s	$\frac{0,5LT}{T + 0,3L}$ = 0,008393s

Fonte: Autoria própria.

Com base nos valores da Tabela 7, foi possível fazer o comparativo entre a resposta ao degrau do Motor A e o desempenho do controlador PID com sintonia AMIGO, ao informar um valor de referência de 3.21 RPS, valor igual ao ganho estático da curva ao degrau do Motor A, como mostra a Figura 50.

Figura 50 - PID via AMIGO - Motor A.



Fonte: Autoria própria.

A Tabela 8 mostra os resultados de desempenho da curva de resposta do PID para o Motor A sintonizado via método AMIGO e ilustrada na Figura 50.

Tabela 8 - Parâmetros do PID motor A.

Parâmetro	Valor
Tempo de subida (RiseTime):	0.0385 s
Tempo de acomodação (SettlingTime):	0.1575 s
Referência	3.21 RPS
Acomodação +- 2%	0,0642 RPS
Sobressinal (Overshoot):	4,33 %
Pico (Peak):	3,349 RPS
Tempo de pico (PeakTime):	0.073 s

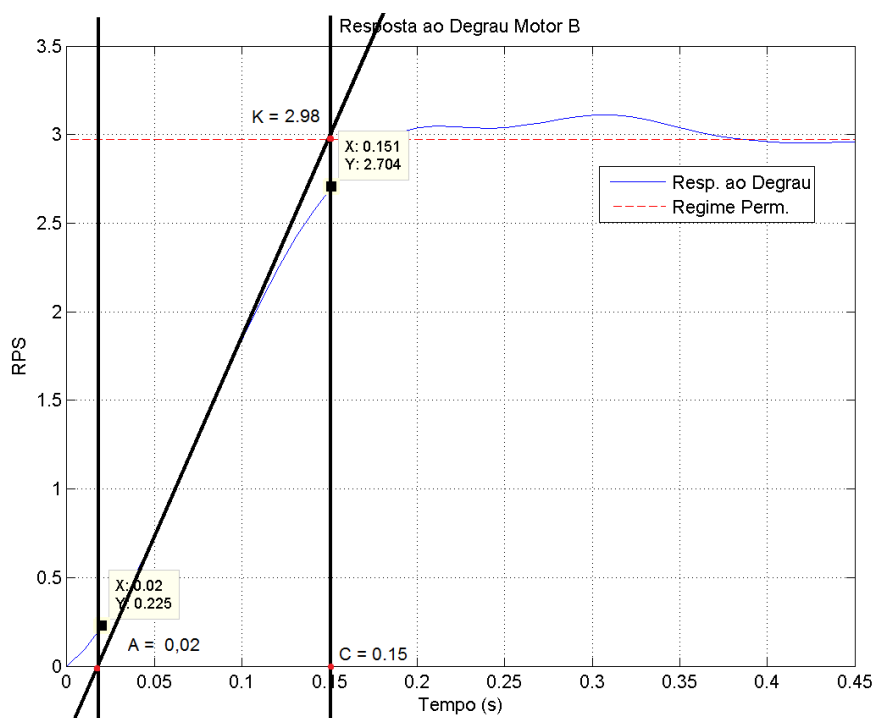
Fonte: Autoria própria.

De acordo com os dados da Tabela 8, os valores de tempo de subida e de assentamento do Motor A diminuíram significativamente, ou seja, o desempenho do motor melhorou com a aplicação do controlador PID sintonizado pelo método AMIGO.

Devido ao pequeno sobressinal, o valor de pico aumentou para 3.349 RPS em relação a curva de degrau. No entanto, o tempo de pico diminuiu para 0.073 segundos devido a ação proporcional.

A curva apresentou um valor de 4.33% de sobressinal, valor considerado aceitável levando em conta o desempenho de outros métodos, como o primeiro método de Ziegler-Nichols. O mesmo processo foi adotado para o Motor B. As constantes K, L e T foram extraídas da Figura 51.

Figura 51 - Pontos para sintonia do PID do Motor B



Fonte: Autoria própria.

De acordo com a Figura 51, o valor de $A = L = 0.02$ segundos, já o valor de T é igual ao valor de $C = 0.15$ segundos, menos o valor de A , ou seja, T é igual a 0.13 segundos. O valor de K é o próprio ganho estático do Motor B, sendo $K = 2,98$ RPS. A Tabela 9 mostra os resultados encontrados usando o método AMIGO para o Motor B.

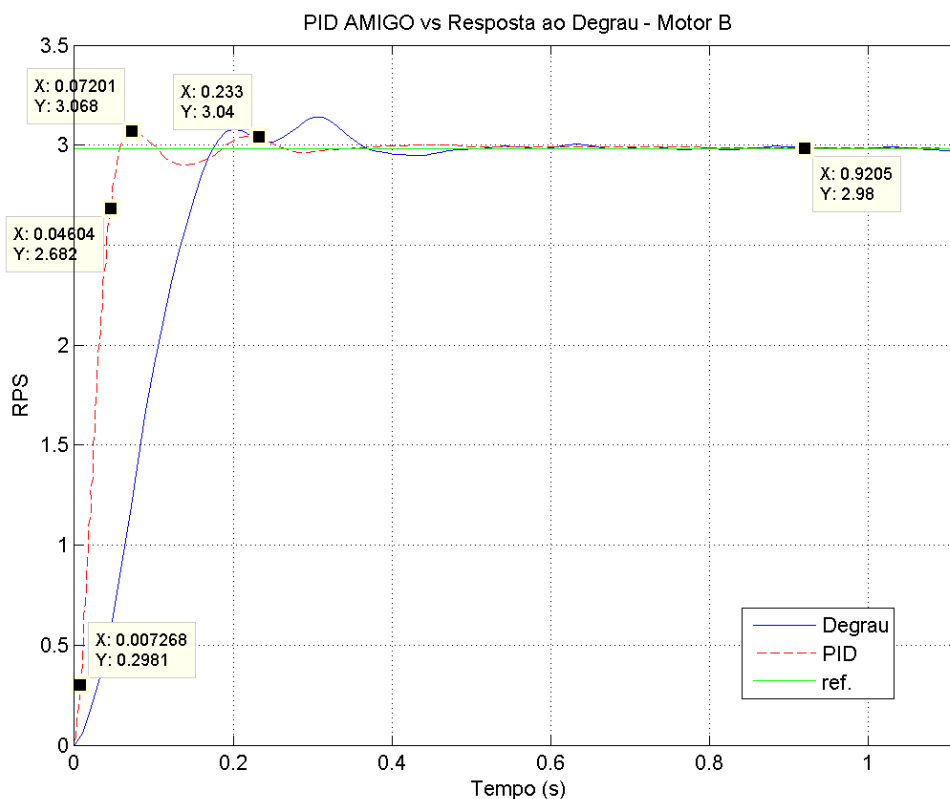
Tabela 9 - Resultados da sintonia AMIGO – Motor B.

Controlador	K_p	τ_i	τ_d
PID	$\frac{1}{K} \left(0,2 + 0,45 \cdot \frac{T}{L} \right)$ $= 1,04865$	$L \cdot \frac{0,4L + 0,8T}{L + 0,1T}$ $= 0,067878 \text{ s}$	$\frac{0,5LT}{T + 0,3L}$ $= 0,009558s$

Fonte: Autoria própria.

Com os resultados das regras do método AMIGO, mostrados na Tabela 9, foi possível sintonizar o controlador PID e obter a curva de resposta do controlador, como mostra a Figura 52.

Figura 52 - PID via AMIGO - Motor B.



Fonte: Autoria própria.

Com base na resposta do PID foi possível encontrar os parâmetros de desempenho da curva de resposta do PID do Motor B, listados na Tabela 10.

Tabela 10 - Parâmetros de desempenho PID motor B.

Parâmetro	Valor
Tempo de subida (RiseTime):	0.0387 s
Tempo de acomodação (SettlingTime):	0.2330 s
Referência	2,98 RPS
Acomodação +-2%	0.0596 RPS
Sobressinal (Overshoot):	2.95 %
Pico (Peak):	3,0679 RPS
Tempo de pico (PeakTime):	0.07201 s

Fonte: Autoria própria.

Como mostra a Tabela 10, os valores para o tempo de subida e tempo de acomodação do moto B também diminuíram significativamente em relação à resposta ao degrau do motor B. Dessa forma, o motor consegue atingir a estabilidade de forma mais rápida.

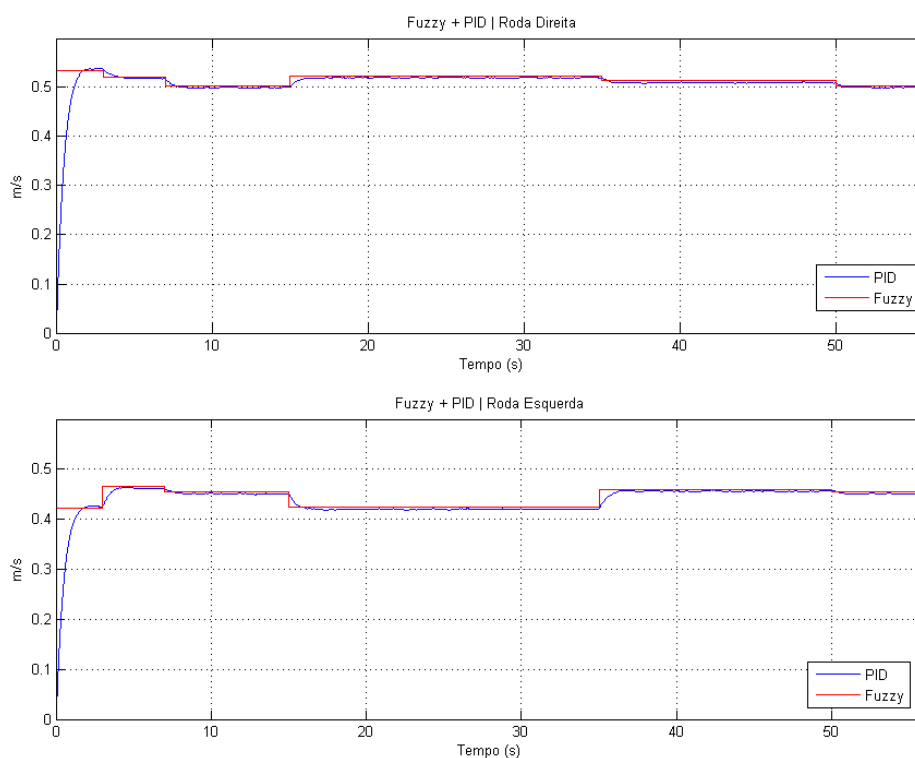
Devido a ação proporcional o controlador obteve um sobressinal de 2,95 % e o valor de pico chegou a 3.0679 RPS. Porém, o tempo de pico e de subida diminuíram significativamente o que garante uma resposta mais rápida do sistema.

De forma geral, mesmo os motores já vindo de fábrica bastante estáveis e com pouca oscilação, o desempenho dos controladores foram superiores à resposta em malha aberta. Os controladores conseguiram manter a estabilidade dos motores de forma rápida e com pouca oscilação.

4.6 Controlador *Fuzzy* e controladores PID em conjunto.

Na Figura 53 é possível conferir o resultado do controlador *Fuzzy* atuando como referência para os controladores PID das rodas direita e esquerda.

Figura 53 - Controlador *Fuzzy* e Controladores PID



Fonte: Autoria própria.

Os gráficos apresentados na Figura 53 foram obtidos ao submeter o robô ao percurso circular ilustrado na Figura 41. A figura mostra o controlador *Fuzzy* fornecendo os valores de referência para os controladores PID para que o robô permaneça alinhado durante o trajeto.

A Figura 53 permite visualizar a diferença de velocidade de cada roda durante o trajeto circular. Essa diferença de velocidade entre as rodas é que garante a realização de curvas em um robô com tração diferencial. No entanto, quando a diferença de velocidade é próxima de zero, o robô está se deslocando em linha reta.

Como mostra a Figura 53, os controladores PID dos motores de ambas as rodas desempenharam uma resposta rápida e bastante estável para as variações de valores de referência fornecidas pelo controlador *Fuzzy*.

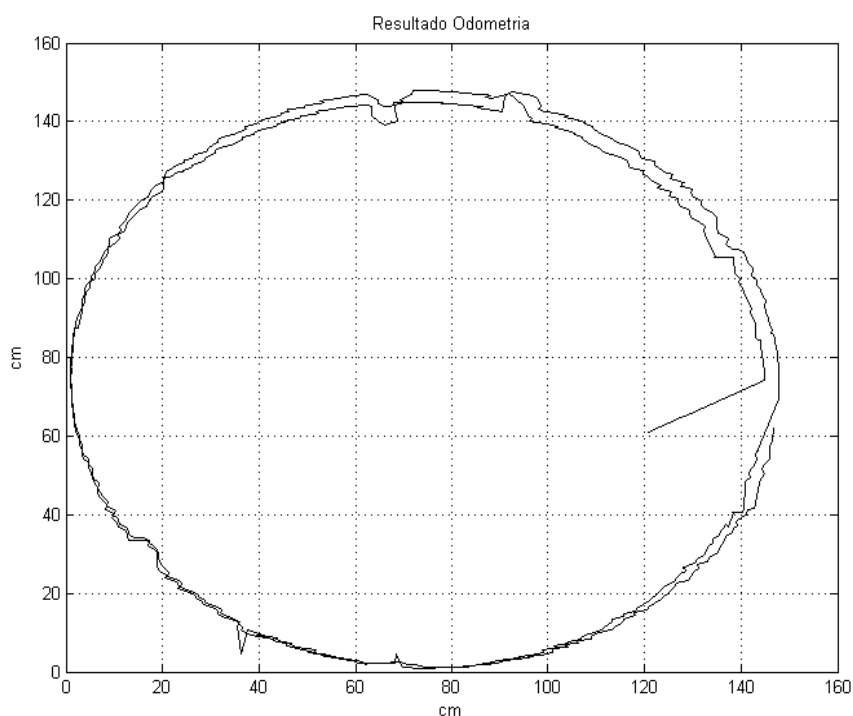
4.7 Odometria.

Os dados obtidos com a etapa de odometria não foram utilizados como entrada em nenhum dos controladores desenvolvidos neste trabalho. As entradas dos controladores foram apenas as informações fornecidas pela etapa de visão computacional e os pulsos do encoder, necessário no PID para controle dos motores. Na literatura, alguns trabalhos utilizam os pontos da odometria como entrada para os controladores como os controladores diferenciais. No entanto, neste trabalho as informações da odometria foram utilizadas apenas para efeito de comparação entre o percurso real do robô e o percurso estimado pela odometria.

Em teoria, quanto menor a diferença entre o percurso real do robô e o percurso gerado pela odometria, melhor o desempenho dos controladores em manter o robô seguindo a trajetória. Entretanto, na prática, para isso ser verdade é necessário que o sistema de odometria esteja perfeitamente ajustado, com a geração mínima de ruídos e calibrado corretamente. Isso é necessário, pois do contrário, os resultados da odometria podem induzir a uma interpretação errada dos dados.

A Figura 53 mostra o trajeto percorrido pelo robô com base nos dados coletados pela odometria. O percurso real traçado para o robô perseguir foi um círculo com raio de 70cm ilustrado na Figura 41.

Figura 54 - Resultado da odometria para o círculo.



Fonte: Autoria própria.

Como pode ser constatado na Figura 54, a forma da resposta da odometria é semelhante ao círculo traçado e mostrado na Figura 41. O que comprova que o robô seguiu um trajeto circular dando exatamente duas voltas no percurso.

Porém, como os dados da odometria encontrados neste trabalho foram bastante ruidosos, a utilização do resultado da odometria como método de qualificar o desempenho dos controladores é insatisfatório, pois os ruídos induzem ao erro. Mesmo em uma situação em que o desempenho do controlador seja excelente, um gráfico ruidoso qualificaria o desempenho do controle de forma errônea.

Com base na Figura 54, a resposta da odometria aparenta se distanciar do centro a cada interação. Isso pode ocorrer pelo acúmulo de pequenos erros ou ruídos.

Este capítulo apresentou os resultados e discussões obtidos com a realização deste trabalho. As considerações finais para esta dissertação são apresentadas no Capítulo 5.

5 CONSIDERAÇÕES FINAIS

A partir dos resultados apresentados no Capítulo 4, podemos concluir que o sistema de visão computacional e processamento de imagem conseguiu capturar, processar e informar corretamente para o controle *Fuzzy* os pontos correspondentes aos centros da trajetória a ser seguida. Estes pontos são referentes à média ponderada do vetor composto pelos pixels da linha guia horizontal que é fixa em cada imagem processada.

A captura do ponto central da trajetória é importante, pois é destas informações que é gerada a saída do sistema de visão, sendo elas o Erro e o Desvio do Erro. O Erro, como foi exposto, é a diferença entre o centro da imagem, que neste trabalho é fixo em 320, e o valor do centro da trajetória que pode variar entre 1 e 640. Já o Desvio do Erro é a diferença entre o Erro atual e o Erro anterior. Portanto, a captura correta do ponto central da trajetória pode ser entendida com uma tarefa de fundamental importância para o sucesso das etapas seguintes e do trabalho como um todo.

Os resultados satisfatórios da etapa de visão computacional se justifica, em parte, pela eficiência do método de Otsu e também pela aplicação do filtro de suavização Gaussiano que permitiu a eliminação de pequenos ruídos causados nas etapas de captura, processamento de imagem, e das pequenas falhas da trilha, principalmente da trilha desenhada com pincel de quadro branco, apresentada na Figura 41 no Capítulo 4. O sistema de visão também foi eficiente em conseguir binarizar diferentes tipos de trilhas com espessuras variadas, como foi ilustrado nas Figuras 40 e 44.

Com base nas respostas do sistema de visão computacional, foi possível encontrar o Erro e o Desvio do Erro, pontos necessários para que o controlador *Fuzzy* calcule a velocidade de cada roda, e conseqüentemente a trajetória do robô corretamente, já que o robô é um veículo de tração diferencial.

O controlador *Fuzzy*, com base nas saídas do sistema de visão, conseguiu calcular e fornecer saídas satisfatórias para as velocidades das rodas direita e esquerda como foi demonstrado nas Figuras 45 e 46 do Capítulo 4, ou seja, a resposta do controlador *Fuzzy* é condizente com o esperado para cada combinação de entrada.

As saídas do Fuzzy representam quão rápidas as rodas devem girar para que o robô consiga corrigir o erro, ou seja, para que o erro seja o mais próximo de zero possível.

Mesmo os motores A e B sendo bastante estáveis, a sintonia para os controladores PID utilizando o método AMIGO, mostraram-se eficientes em controlarem a velocidade dos motores, como mostra a Figura 50 e 52. Os valores de tempo de acomodação e tempo de subida dos controladores diminuíram significativamente, fatos que também comprovam uma significativa melhoria se comparados com os mesmos parâmetros fornecidos pela resposta ao degrau dos motores.

A sintonia correta dos controladores PID visa garantir uma resposta mais rápida, precisa e com pouca oscilação para cada valor de referência fornecida pelo controlador *Fuzzy*. De forma geral, as etapas de visão computacional, controle *Fuzzy* e controle PID, mesmo sendo trabalhosas, conseguiram entregar resultados que satisfazem a demanda do robô móvel, permitindo que o mesmo percorra a trajetória fornecida de forma alinhada e com pouca oscilação, como mostra a Figura 53.

No entanto, a etapa de odometria foi a mais crítica deste trabalho. Como mostra o resultado obtido na Figura 54, as informações provenientes da odometria são bastante ruidosas, mesmo com os *encoders* calibrados e utilizando rodas com diâmetro iguais.

No entanto, os dados provenientes da odometria não interferem no controle do robô, pois o mesmo não os usa como entrada para seu sistema. Dessa forma, a odometria foi utilizada somente como forma de validação. Os dados da odometria foram utilizados para comparar se o percurso percorrido pelo robô condiz com o círculo de raio de 70cm traçado para validação.

Com os resultados da odometria é possível concluir que o robô percorreu realmente um círculo com raio aproximado de 70cm, pois mesmo com os ruídos os círculos formados na Figura 54 são bastante próximos do círculo real. No entanto, como metodologia de análise de desempenho dos controladores, essa estratégia não é satisfatória, pois em uma eventual comparação entre o erro resultante entre os círculos, levaria a uma conclusão errônea do desempenho dos controladores, uma

vez que os ruídos, neste trabalho, impedem que o resultado da odometria represente exatamente o percurso real percorrido pelo robô.

Dessa forma, acredita-se que este trabalho contribuiu para elucidar que a visão computacional é uma importante ferramenta e pode ser utilizada na área de controle, principalmente como forma de obtenção de características repetitivas de um processo e como forma de entrada para outros controladores já consagrados. O trabalho também contribui como ponto de partida para a criação de outros trabalhos onde se faz necessário a identificação de trilhas.

Para trabalhos futuros, sugere-se a inclusão, na etapa de visão computacional, de um sistema de identificação de obstáculos e sinalizações, para que o robô tenha maior autonomia em percorrer um determinado trajeto.

A substituição do controlador *Fuzzy* por controladores baseado em redes neurais também se mostram promissores nessa linha de pesquisa, assim como a implementação dos controladores em um dispositivo separado do robô, para que o controle seja feito a distância e possa ser utilizado por outros robôs em um mesmo local.

REFERÊNCIAS

ARTERO, Almir Olivette. **Técnicas para extração automática de feições retas em imagens digitais**. Presidente Prudente, 1999.

ÅSTRÖM, Karl J.; HÄGGLUND, Tore. **PID controllers: theory, design, and tuning**. 2ª ed. Research Triangle Park, NC: Instrument society of America, 1995.

ÅSTRÖM, Karl J.; HÄGGLUND, Tore. **Revisiting the Ziegler-Nichols Step Response Method for PID Control**. Journal of Process Control, 14, p. 635-650, 2004.

BACKES, André R.; SÁ JUNIOR, Jarbas J. D. M. **Introdução à Visão Computacional Usando Matlab**. Rio de Janeiro: Alta Books, 2016.

BERTACHI, Arthur Hirata et al. **Implementação de um PID Digital em ambiente computacional aplicado a uma planta didática para ensino de controle para engenharia** In: XLI Congresso Brasileiro de Educação em Engenharia-COBENGE. 2013.

BUTDEE, S; SUEBSOMRAN, A; VIGNOOT, F; YARLAGADDA, P. K. **Control and path prediction of an Automate Guided Vehicle**. 2008.

CHIEN, K.L.; HRONES, J.A.; RESWICK, J.B.; **On the Automatic Control of Generalized Passive Systems**. Trans. ASME, v. 74, p. 175-185, 1952.

COHEN, G.H.; COON, G.A.. **Theoretical Consideration of Retarded Control**. Trans. ASME, v.75, p. 827-834, 1953.

DUDEK, Gregory; JENKIN, Michael. **Computational principles of mobile robotics**. Cambridge university press, 2010.

FITZGERALD, A.E.; KINGSLEY, Charles Jr.; UMANS, Stephen D. **Máquinas elétricas: com introdução à eletrônica de potência**. 6ª ed, 2006.

GONZALEZ, Rafael C.; WOODS, Richard C. **Processamento Digital de Imagens**, 3ª edição. Editora Pearson do Brasil, 2010.

GOMIDE, Fernando A. Campos; GUDWIN, Ricardo Ribeiro. **Modelagem, controle, sistemas e lógica fuzzy**. SBA controle & Automação, v. 4, n. 3, p. 97-115, 1994.

HUANG, Min; HUANG, Dagui. **Embedded fuzzy logic control of AGV in path tracking**. In: 2010 IEEE International Conference on Mechatronics and Automation. IEEE, 2010. p. 682-686.

ISERMANN, Rolf. **Digital control systems**. Vol. 1, 2^o ed, Springer Science & Business Media, 2013.

JESUS, Edison O.; COSTA JR, Roberto. **A utilização de filtros gaussianos na análise de imagens digitais**. Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 3, n. 1, 2015.

KODAGODA, K. R. S.; WIJESOMA, W. Sardha; TEOH, Eam Khwang. **Fuzzy speed and steering control of an AGV**. IEEE Transactions on control systems technology, v. 10, n. 1, p. 112-120, 2002.

KRIDI, Douglas Santiago et al. **Desenvolvimento de uma biblioteca fuzzy para o controle autônomo de um robô móvel em ambiente desconhecido**. Mostra Nacional de Robótica, v. 1, 2013.

LEVINE, William S. **The control systems handbook: Control system advanced methods**. CRC Press, 2010.

MARENGONI, Maurício; STRINGHINI, Stringhini. **Tutorial: Introdução à visão computacional usando opencv**. Revista de Informática Teórica e Aplicada, v. 16, n. 1, p. 125-160, 2009.

MONTEIRO, Leonardo Hiss. **Binarização por otsu e outras técnicas usadas na detecção de placas**. Disponível em: <<http://www2.ic.uff.br/~aconci/OTSUeOutras.pdf>> Acessado em: 15 de Março de 2018.

NETO, Adão de Melo. **Lógica Fuzzy**. Notas de Aula. Disponível em: <<https://www.ime.usp.br/~adao/LOGICAFUZZY2017F.pdf>>. Acessado em: 13 de Dezembro de 2018.

NIKU, Saeed B. **Introdução À Robótica: análise, controle, aplicações**. Reimpressão da 2^a edição de 2013. Rio de Janeiro: LTC, 2017.

OGATA, Katsuhiko. **Engenharia de Controle Moderno**. 5. ed., Editora Pearson Brasil, 2011.

OMNIVISION. **OV5647 datasheet**. Preliminary specification, 2009. Disponível em: <https://cdn.sparkfun.com/datasheets/Dev/RaspberryPi/ov5647_full.pdf>. Acessado em: 13 de Março de 2018.

OTSU, Nobuyuki. **A threshold selection method from gray-level histograms**. IEEE Transactions On Systems, Man, And Cybernetics, Vol. SMC-9, n. 1, p. 62–66, Jan, 1979.

PINTO, Jan Erik M., G. **Aplicação prática do método de sintonia de controladores PID utilizando o método do relé com histerese**. 2014. Dissertação de Mestrado. Universidade Federal do Rio Grande do Norte.

POLOLU. **Pololu Dual VNH5019 Motor Driver Shield User's Guide**. Disponível em: <https://www.pololu.com/docs/pdf/0J49/dual_vnh5019_motor_driver_shield.pdf> Acessado em: 18 de Junho de 2019.

PRESTES, Edson. Notas de Aula. **Introdução à Robótica Móvel**. Disponível em: <<http://www.inf.ufrgs.br/~prestes/Courses/Robotics/Slides/RAula25.pdf>>. Acessado em: 28 de Janeiro de 2019.

ROMERO, Roseli A. F.; PRESTES, Edson; OSÓRIO, Fernando; WOLF, Denis. **Robótica Móvel**. 1ª reimpressão da 1ª edição de 2014. Rio de Janeiro: LTC, 2017.

SANCHES, Carlos, H.; FONTOURA, Paulo, J.; VIERA, Phillypi, F.; Batista, Marcos, A.; **Técnicas de Suavização de Imagens e Eliminação de Ruídos**. Anais do EATI - Encontro Anual de Tecnologia da Informação e Semana Acadêmica de Tecnologia da Informação, Ano 5, n. 1, p. 21-30, 2015.

SILVA, Pedro Miguel de Sá Pereira et al. **Movimentação autónoma de robôs móveis de baixo custo, com base no sistema NXT da Lego**. 2010.

TANSCHHEIT, Ricardo. **Sistemas fuzzy**. Departamento de Engenharia Elétrica, PUC-Rio. 2004.

YACINE, Ahmine; FATIMA, Chouireb; AISSA, Bencherif. **Trajectory tracking control of a wheeled mobile robot using an ADALINE neural network.** In: 2015 4th International Conference on Electrical Engineering (ICEE). IEEE, 2015. p. 1-5.

ZIEGLER, J. G.; NICHOLS, N. B.; **Optimum settings for automatic controllers.** Trans ASME, v. 64, p. 759-768, 1942.