Harrison Neves Marciano

Adequação de uma plataforma de um robô móvel articulado e desenvolvimento de soluções para modelagem e controle fuzzy, nas manobras à ré, segundo uma estratégia de transição entre giros

Brasil

Dezembro,2017

Harrison Neves Marciano

Adequação de uma plataforma de um robô móvel articulado e desenvolvimento de soluções para modelagem e controle fuzzy, nas manobras à ré, segundo uma estratégia de transição entre giros

Parte manuscrita da Proposta de Projeto de Graduação do Harrison Neves Marciano, apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para conclusão do curso de engenharia elétrica apresentado na disciplina "ELE08553 - Projeto de Graduação II".

Universidade Federal do Espirito Santo – UFES

Faculdade de Engenharia Elétrica

Graduação

Orientador: Dr. Edson de Paula Ferreira

Coorientador: MSc. Diego Nunes Bertolani

Brasil

Dezembro,2017

Harrison Neves Marciano
Adequação de uma plataforma de um robô móvel articulado e desenvolvimento de soluções para modelagem e controle fuzzy, nas manobras à ré, segundo uma estratégia de transição entre giros/ Harrison Neves Marciano. – Brasil, Dezembro,2017-
113 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Edson de Paula Ferreira

Acadêmico – Universidade Federal do Espirito Santo – UFES
Faculdade de Engenharia Elétrica
Graduação, Dezembro,2017.

1. Palavra-chave1. 2. Palavra-chave2. 2. Palavra-chave3. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

Harrison Neves Marciano

Adequação de uma plataforma de um robô móvel articulado e desenvolvimento de soluções para modelagem e controle fuzzy, nas manobras à ré, segundo uma estratégia de transição entre giros

Parte manuscrita da Proposta de Projeto de Graduação do Harrison Neves Marciano, apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para conclusão do curso de engenharia elétrica apresentado na disciplina "ELE08553 - Projeto de Graduação II".

Trabalho aprovado. Brasil, 15 de Dezembro de 2017:

Dr. Edson de Paula Ferreira
Orientador

MSc. Diego Nunes Bertolani
Coorientador

Professor Dr.
Alessandro Mattedi

Engenheiro Eletricista
Ricardo Carminati de Mello

Brasil

Dezembro,2017

Este trabalho é dedicado primeiramente a Deus,
sem o qual eu não teria chegado tão longe,
e aos meus familiares, sobretudo ao meu irmão,
pelos conselhos sempre pertinentes.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por ter me dado saúde e condições de permanecer na graduação pelo tempo necessário. Agradeço ao meu orientador Edson de Paula Ferreira e ao meu co-orientador Diego Nunes Bertolani, pela paciência e orientação. Agradeço à minha família pela compreensão, sobretudo ao meu irmão, pela paciência aparentemente infinita para lidar com problemas alheios. Por fim, agradeço à Universidade Federal do Espírito Santo (UFES), que tem sido parte da minha vida a alguns anos e contribuiu enormemente para minha formação profissional e pessoal.

"O mundo está como está
por causa das certezas.
A guerra e a vaidade
comem na mesma mesa".
(Jorge Drexler)

RESUMO

Neste projeto, foram implementadas alterações sobre uma plataforma experimental de um Robô Móvel Multi-Articulado (RMMA), disponível para pesquisa no LCI (Laboratório de Controle Inteligente) do DEL/CT/UFES, visando adequá-lo em termos de *hardware* e *software*, para o uso de estruturas avançadas de controle. Estas estruturas visam principalmente a execução de manobras de RMMAs que envolvam movimentos a ré. O principal problema deste tipo de movimento está na complexidade de seu controle quando demanda-se a realização de manobras com variações amplas dos ângulos de configuração das articulações ou a manutenção destes ângulos enquanto o movimento a ré é realizado. Em particular, criou-se um sistema de armazenamento de dados mais prática do que o então existente, o que facilitou a implementação de técnicas de modelagem, tais como *fuzzy* e neural, que necessitam de grandes volumes de dados.

Através dos dados armazenados, um modelo *fuzzy* do RMMA que o representa em diversos pontos de funcionamento foi elaborado e, em seguida, também através destes dados, implementou-se um controlador *fuzzy* o qual exercia um controle sobre os ângulos de configuração enquanto o protótipo movia-se em ré, fazendo-o transitar entre estes ângulos de forma prática, recebendo sinais de comandos provenientes de um dispositivo transmissor de ondas de rádio AM.

Após este controlador ser implementado, realizaram-se testes nos quais era exigido do protótipo a realização de manobras nas quais este precisou alcançar referências distantes de sua configuração inicial e manter-se nelas enquanto se movia a ré. Também foi requerido a realização de manobras em que foi necessário desviar-se de obstáculos, como uma forma de demonstrar que o sistema implementado consegue auxiliar os operadores na execução de movimentos complexos em ré.

Palavras-chave: RMMA, *Fuzzy*, Movimento a Ré e Armazenamento de Dados.

ABSTRACT

In this project, it was implemented alterations upon an experimental platform of a Multi-Articulated Mobile Robot (MAMR), available for research at ICL (Intelligent Control Laboratory) from DEL/CT/UFES, aiming to update it in terms of hardware and software, for purpose of its usage at advanced controlling structures. These structures aime mainly the execution of moviments with the MAMRs which involve backward maneuvers. The main problem concerning this sort of moviment is its control complexity when it is required the execution of maneuvers which involve larges variations of the configuration angles from the articulated elements or even the stillness of these angles while the moviment is made. Particularly, it was made a more practical data storage system than the current one, what made the implementation of modeling techniques, such as fuzzy and neural, easier, once those techniques need a great volumes of data.

Through these storaged data, a fuzzy model of the MAMR which represents it in several operatinal points was made and, after, also through those same storaged data, it was implemented a fuzzy controller which executed a control upon the configuration angles while the prototype was moved in backward, making it vary between these angles in a more practical manner, receiving AM radio waves signals from a transmitter device.

After this controller has been implemented, it was made tests in which it was required that the prototype executed backward maneuvers aiming far appart references from its initial positioning and also that it remained at those references, once it reached them, moving backwardly. Also, it was required the execution of maneuvers in which it was needed to dodge obstacles, in order to demonstrate that this new implemented system can actually help operators to execute complex backward moviments.

Keywords: MAMR, *Fuzzy*, Backward Moviments e Data Storage System.

SUMÁRIO

Sumário 10

Lista de Quadros 12

1 INTRODUÇÃO 17
1.1 Motivação e Escopo do Trabalho 17
1.2 Caracterização do Protótipo RMMA 20
1.3 Revisão Bibliográfica 22
1.4 Caracterização do Problema de Controle e Modelagem 26
1.5 Objetivos 30
1.5.1 Objetivos Gerais: 30
1.5.2 Objetivos Específicos: 30
1.5.3 Desdobramentos: 31
1.6 Metodologia 31
1.6.1 Estudo Teórico 32
1.6.2 Avaliação e Reestruturação Física do Protótipo 32
1.6.3 Criação de um Modelo e Implementação do Sistema de Controle 32
1.6.4 Testes e Análise de Resultados 33

2 EMBASAMENTO TEÓRICO 34
2.1 Modelagem e Controle *Fuzzy* 34
2.2 Geração de Regras e Inferência de Saídas 40
2.2.1 Metodologia de Geração de Regras e Inferência de Wang-Mendel 41
2.2.2 Metodologia de Geração de Regras e Inferência de Takagi-Sugeno 44

3 REESTRUTURAÇÃO DO HARDWARE 48
3.1 *Hardware* Antigo 48
3.2 Placa Arduino 51
3.3 Módulo Gravador/Leitor SD e *Level Converter* 52
3.4 *Shield* e *Octal Buffer Driver* 74HC244N 53
3.5 *Encoder* e Estimação de Velocidade 54
3.6 Rádio e Receptor AM 58

4 ELABORAÇÃO DO MODELO E CONTROLADOR *FUZZY* 60
4.1 Considerações e Simplificações 60
4.2 Levantamento do Modelo *Fuzzy* 60
4.2.1 Obtenção das regras para Pequenas Variações nos Ângulos de Giro 60

4.2.2 Obtenção de um Modelo *Fuzzy* para Validação do Método 64
4.3 Elaboração do Controlador *Fuzzy* **68**

5 RESULTADOS **76**
5.1 Manobras de *Tracking* **76**
5.2 Manobras Desviando-se de Obstáculos **81**
5.3 Esforço de Controle **82**

6 CONCLUSÃO E TRABALHOS FUTUROS **86**

7 ALOCAÇÃO DE RECURSOS **88**

REFERÊNCIAS **89**

A CONTROLE DIRETO SOBRE AS RODAS DIANTEIRAS **91**

B IMPLEMENTAÇÃO DO CONTROLADOR FUZZY ATUANTE SOBRE O ÂNGULO DE CONFIGURAÇÃO **100**

LISTA DE ILUSTRAÇÕES

Figura 1 – Protótipo do Robô Móvel Multi-Articulado (RMMA) 18
Figura 2 – Tipos Principais de Posicionamento de Engates 21
Figura 3 – Cadeia Cinemática genérica de um RMMA 22
Figura 4 – Representação Geral do Elemento *Truck* e *Trailers* 23
Figura 5 – Malha de Controle *feedback* não linear 24
Figura 6 – Particionamento Definido Proporcionalmente 25
Figura 7 – Controle *Fuzzy* Centralizado de Três Entradas 26
Figura 8 – Representação do Modelo *Fuzzy* Utilizado Para Validação 29
Figura 9 – Representação do Controlador *Fuzzy* Proposto 29
Figura 10 – Representação da Variável Lingüística "altura" 36
Figura 11 – Arquitetura Básica de um Sistema de Controle *Fuzzy* 37
Figura 12 – Raciocínio para Geração de Saída *Fuzzy* da Máquina de Inferência . . 39
Figura 13 – *Defuzzyficação* para a Média dos Máximos e Centros de Gravidade . . 40
Figura 14 – Divisão dos Espaços de Entrada e Saída em Regiões *Fuzzy* 42
Figura 15 – Formato da tabela de regras *fuzzy* para duas entradas e uma saída . . 43
Figura 16 – Diagrama de Inferência Takagi-Sugeno 47
Figura 17 – Principais Componentes do Antigo Sistema Embarcado 49
Figura 18 – Potenciômetro e Acoplamento Mecânico dele à Articulação 49
Figura 19 – Sistema de Comunicação RF com Entrada USB para Conexão com Computador 50
Figura 20 – Pinos de Entrada e Saída da Placa de Desenvolvimento Arduino UNO 51
Figura 21 – Módulo para cartão SD 53
Figura 22 – Conversor de Nível Bidirecional 53
Figura 23 – Soldas Embaixo do *Shield* 54
Figura 24 – Pinos para Conexões no *Shield* 55
Figura 25 – *Encoder* Simples 55
Figura 26 – *Encoder de Quadratura* 56
Figura 27 – *Encoder* Absoluto 56
Figura 28 – *Encoder* Simples Existente no Protótipo 57
Figura 29 – Rádio AM de Dois Canais 58
Figura 30 – Receptor AM de Dois Canais 58
Figura 31 – *Freqüência da Ocorrência de Amostras para as Medidas de Velocidade* . 62
Figura 32 – *Freqüência da Ocorrência de Amostras para as Medidas de Direção* . . 62
Figura 33 – *Freqüência da Ocorrência de Amostras para as Medidas do Ângulo de Formação* 63

Figura 34 – *Freqüência da Ocorrência de Amostras para as Medidas de Variação do Ângulo* 63
Figura 35 – *Divisão da Variável Lingüística Direção* 65
Figura 36 – *Divisão da Variável Lingüística Ângulo* 66
Figura 37 – Comparação entre as Respostas Obtidas Através do Modelo *Fuzzy* e da Equação Analítica com os Valores Medidos 67
Figura 38 – Diferença entre Referência e Ângulo Atual 69
Figura 39 – Ângulo Atual 70
Figura 40 – Malha de Controle Proposta 71
Figura 41 – Simulações Utilizando o Controlador e o Modelo *Fuzzy* 71
Figura 42 – Retas Obtidas das Saídas Fazendo-se $\Delta\theta = -6^o$ e $\Delta\theta = 6^o$ 73
Figura 43 – Repetição das Simulações Comparando o Controlador com e sem o Conhecimento Especialista 75
Figura 44 – Respostas Partindo-se Próximo de Zero com Referências em -30º, -20º, -10º, 10º, 20º e 30 77
Figura 45 – Respostas Partindo-se Próximo de Zero com Referências em -25º, -15º, -5º, 5º, 15º e 25º 77
Figura 46 – Respostas Partindo-se do Sentido de Giro Desejado com Referências em -30º, -20º, -10º, 10º, 20º e 30 78
Figura 47 – Respostas Partindo-se do Sentido de Giro Desejado com Referências em -25º, -15º, -5º, 5º, 15º e 25º 78
Figura 48 – Respostas Partindo-se do Sentido de Giro Oposto com Referências em -30º, -20º, -10º, 10º, 20º e 30º 79
Figura 49 – Respostas Partindo-se do Sentido de Giro Oposto com Referências em -25º, -15º, -5º, 5º, 15º e 25º 79
Figura 50 – Resposta para uma Sequência de Mudanças de Referências 80
Figura 51 – Resposta para Manobra Desviando-se de Obstáculos 81
Figura 52 – Esforço de Controle em Manobras Iniciadas com Diversas Configurações e Referências em -30º, -20º e -10º 83
Figura 53 – Esforço de Controle em Manobras Iniciadas com Diversas Configurações e Referências em 30º, 20º e 10º 83
Figura 54 – Esforço de Controle em Manobras Iniciadas com Diversas Configurações e Referências em -25º,-15º e -5º 84
Figura 55 – Esforço de Controle em Manobras Iniciadas com Diversas Configurações e Referências em 25º,15º e 5º 84

LISTA DE QUADROS

Quadro 1 – Características do Arduino UNO 51
Quadro 2 – Recursos e Gastos 88

LISTA DE ABREVIATURAS E SIGLAS

ABNT	*Associação Brasileira de Normas Técnicas*
RMMA	*Robô Móvel Multi Articulado*
ESC	*Eletronic Speed Controler*

LISTA DE SÍMBOLOS

γ	Ângulo da direção das rodas dianteiras
θ	Ângulo entre elementos *trailers* ou entre o elemento *truck* e o próximo *trailer*
μ	Função de Pertinência
α	Cortes de Nível

1 INTRODUÇÃO

1.1 Motivação e Escopo do Trabalho

Em problemas que requeiram otimizar e aperfeiçoar processos, realizar atividades repetitivas mantendo um padrão de qualidade desejável, executar procedimentos os quais necessitam de grande precisão, ou ainda, que envolvam riscos consideráveis para a intervenção humana, a utilização de robôs e outros dispositivos eletromecânicos em geral, já há muito tempo, tem sido uma solução recorrente, ainda mais em problemas de engenharia.

Não apenas na indústria, mas em outros setores como os relacionados à prestação de serviços e à infra-estrutura, percebe-se um processo de difusão cada vez maior da utilização de sistemas automatizados, visando obter maior eficiência, produtividade, qualidade e segurança em todo o processo, além de maior valor agregado ao produto final. Como resultado deste investimento, as empresas obtêm um maior nível de competitividade no mercado.

Pode-se dar como exemplo desta tendência de mercado o porto de Roterdã nos Países Baixos que, em 2015, inaugurou seu primeiro terminal de contêineres totalmente automatizado contando com 8 guindastes controlados remotamente por operadores em uma sala de controle separada da área por onde as cargas passam, além de 62 caminhões-robôs utilizados para transportar contêineres pelo pátio de armazenamento até os trens e caminhos que levarão as cargas ao porto. Embora o sistema de guindastes na época ainda não fosse tão rápido quanto outros sistemas não-automatizados, havia uma forte perspectiva de crescimento da capacidade do terminal uma vez que o mesmo estivesse totalmente operacional, justamente por diminuir a ocorrência de falhas humanas tão costumeiras neste tipo de operação (LOBO, 2016). Outros exemplos poderiam ser dados, demonstrando o potencial existente na robótica para transformar a forma como as pessoas vivem e interagem com o mundo.

Seguindo esta tendência, este trabalho busca somar conhecimento no campo da robótica, especificamente nos problemas de modelagem e controle de robôs móveis articulados. Estes problemas vêm sendo objeto de pesquisas no Laboratório de Controle Inteligente (LCI) do Centro Tecnológico (CT) da Universidade Federal do Espírito Santo (UFES), onde diversos trabalhos já foram desenvolvidos utilizando-se da plataforma experimental que se deseja atualizar.

Esta plataforma experimental é composta por um elemento trator ou *truck* ao qual são acoplados sucessivos *trailers* conectados entre si. Esta é uma versão em escala 1:14 de veículos utilizados para transporte de cargas e produtos tanto em grandes distâncias quanto dentro de áreas delimitadas. A imagem do protótipo pode ser vista na figura 1.

Figura 1 – Protótipo do Robô Móvel Multi-Articulado (RMMA)

Fonte: Bertolani, 2011, p. 20.

A escolha de veículos multiarticulados como no caso de estudo possui relevância quando se pensa no cenário de ambientes onde grandes quantidades de mercadorias precisam ser reposicionadas, armazenadas ou levadas até outros veículos maiores. Tais robôs podem ser utilizados para prover um uso mais racional da área de estocagem em portos e aeroportos, como foi exemplificado em (LOBO, 2016), onde mercadorias e bagagens precisam ser armazenadas e/ou deslocadas para atender um fluxo adequado para as demandas dos usuários finais.

O mesmo se aplica a grandes armazéns automatizados, onde é imprescindível para o processo global das empresas que o tempo de armazenagem e retirada das mercadorias contidas nestes locais seja o menor possível. O uso racional e o barateamento do espaço físico podem exigir robôs móveis com vários *trailers* trafegando em corredores, necessitando da capacidade de manobras automáticas, principalmente nos movimentos a ré para entrada e saída de corredores.

Ainda como exemplo, pode-se enfatizar as vantagens da agregação desta tecnologia ao transporte rodoviário por meio de caminhões articulados, que podem transportar um grande volume de mercadorias de uma só vez. Estes caminhões têm baixa manobrabilidade, mas possuem estruturas semelhantes ao de robôs móveis articulados, sendo possível implementar o mesmo tipo de automação em ambos os casos, principalmente nos casos de manobras em ré. Caminhões dotados de sistemas de manobras automáticas deixariam de ter a desvantagem da utilização em pátios de manobras em espaços restritos.

O maior desafio no controle desses veículos está nas situações em que eles devem mover-se em marcha ré, pois, neste caso, a referência desejada é definida pela direção do último *trailer* no espaço de tarefas, mas o controle do operador é efetuado sobre a direção das

rodas dianteiras, sendo o comportamento dos *trailers* passivos uma consequência disso. O sistema automático converteria a referência do espaço de tarefas em referências no espaço articular ou de configuração do robô.

Conclui-se então que seria extremamente vantajoso haver um sistema automatizado que permita controlar diretamente as partes articuladas passivas de veículos multi-articulados, através da direção do elemento *truck* principal, a fim de auxiliar os operadores a movimentá-los, mesmo em ambientes restritos os quais apresentam obstáculos consideráveis à realização de manobras.

Portanto, a fim de se implementar um sistema de controle com estas características na plataforma de desenvolvimento presente no LCI escolheu-se da metodologia *fuzzy* como base para a elaboração do controlador. Esta escolha foi feita, pois dentre as técnicas de controle existentes, a abordagem *fuzzy* é comumente utilizada em problemas de difícil modelagem matemática ou em que há diversas não linearidades inerentes a planta. Isto, deve-se ao fato da abordagem *fuzzy* ser uma técnica *model free*, ou seja, que pode gerar um controlador independente de modelagens matemáticas e das simplificações e considerações que estes modelos necessitam. Não obstante, ela necessita de um conjunto de dados que não apenas esteja em grande quantidade, mas também abranja majoritariamente toda a dinâmica possível para o protótipo.

Outro aspecto que este projeto precisou abordar foi a atualização não apenas do microcontrolador, mas de todos os sensores do RMMA existente no LCI. Este possuía um sistema embarcado defasado com sensores e atuadores já no final de suas vidas úteis, os quais precisam passar por manutenção ou mesmo serem trocados.

Atuadores como o ESC (*Eletronic Speed Controler*), comumente utilizado em automodelos, precisaram ser trocados devido a defeitos apresentados decorrentes da idade dos equipamentos. Entretanto, como esses atuadores são bastante difundidos em auto-modelos não foi preciso investir em um outro ESC mais moderno que o anterior, pois, desde que ele atenda às especificações de tensão e corrente, não há empecilho em se utilizar uma versão mais simples.

O sistema de odometria composto pelo *encoder* foi mantido, primeiramente porque na literatura considera-se a velocidade constante para este tipo de problema, mesmo em abordagens *fuzzy*, assim ela não representa uma variável relevante no controle. Em segundo lugar, há dificuldades físicas de se implantar outro *encoder* que não o já presente no protótipo, uma vez que o eixo da caixa de redução é de difícil acesso, tornando pouco prático a instalação de algum outro *encoder* que não o já existente.

Os potenciômetros, entretanto, precisaram ser trocados e melhor afixados em seus eixos, pois estes estavam frouxos causando escorregamentos e, consequentemente erros de leitura. Além disso, foi adicionado um conjunto emissor/receptor AM, a fim de enviar os sinais de comando e referência para o protótipo de forma mais prática em relação a forma como isto era feito do *hardware* anterior. Também adicionou-se um módulo de gravação e leitura SD para armazenamento de dados lidos pelos sensores. Devido aos diferentes níveis de tensão requeridos por este gravador, também foi necessária a introdução de um conversor bidirecional de tensão, o qual conecta o *board* de desenvolvimento *Arduino*, onde o microcontrolador está inserido, e o gravador/leitor SD.

Todas essas modificações buscaram não apenas aperfeiçoar o sistema de forma a se obter um melhor controle, mas também deixá-lo mais robusto e adaptável para futuros projetos que o utilizem e abordem o problema sobre outra perspectiva.

1.2 Caracterização do Protótipo RMMA

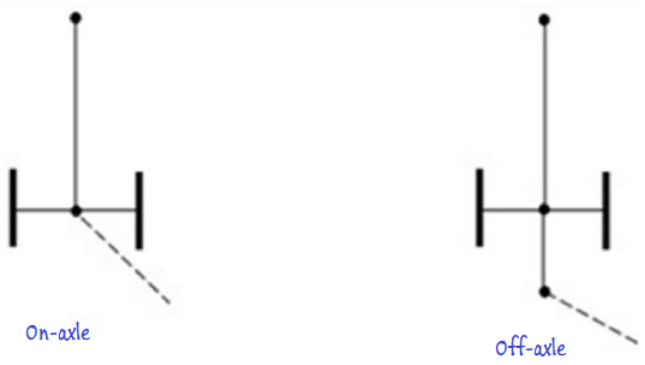
Conforme dito anteriormente, o automodelo constitui-se basicamente de um elemento trator e dois semi-reboques, medidos no total de 1,73m quando alinhados, de fabricação da $Tamiya^{©}$ [TAM,] Co, Ltd.

O elemento trator é dotado de um sistema de transferência de tração do tipo diferencial traseira e de suspensão independente com feixe de molas e amortecedores, o que facilita o transporte de peso sem prejudicar sua movimentação. A tração do veículo é do tipo 6X4, ou seja, quatro das seis rodas são tracionadas por um motor elétrico C.C. da série 540. Acoplado ao motor, existe uma caixa de redução, ou câmbio, que possibilita a escolha da relação de engrenagens entre os valores de 1:32,4, 1:17,7 e 1:10,6, representando a primeira, segunda e terceira marchas, respectivamente (AMERICA, 2017). Para os propósitos deste trabalho, utilizou-se apenas a primeira marcha, pois todos os movimentos os quais desejava-se observar eram em ré, o que demanda torque e não velocidade. Existe um servo-motor que recebe um sinal em PWM, o qual posiciona as engrenagens na marcha desejada. Através do Código, utilizou-se de um dos pinos de entradas e saídas digitais da placa *Arduino* para manter o servo-motor na posição de maior tração.

Em geral, um RMMA é caracterizado por sua cadeia mecânica articulada composta por um *truck* ao qual são acoplados, seqüencialmente, *trailers* por meio de ligações (*hitching*) *on-axle* ou *off-axle*. Ligações *on-axle* são aquelas em que o engate entre dois elementos subseqüentes encontra-se no eixo das rodas, enquanto que ligações *off-axle* são feitas a uma certa distância deste eixo. a figura 2 ilustra estas situações.

O *truck* é o elemento motorizado, responsável pela tração, a qual é imposta pelas rodas

Figura 2 – Tipos Principais de Posicionamento de Engates



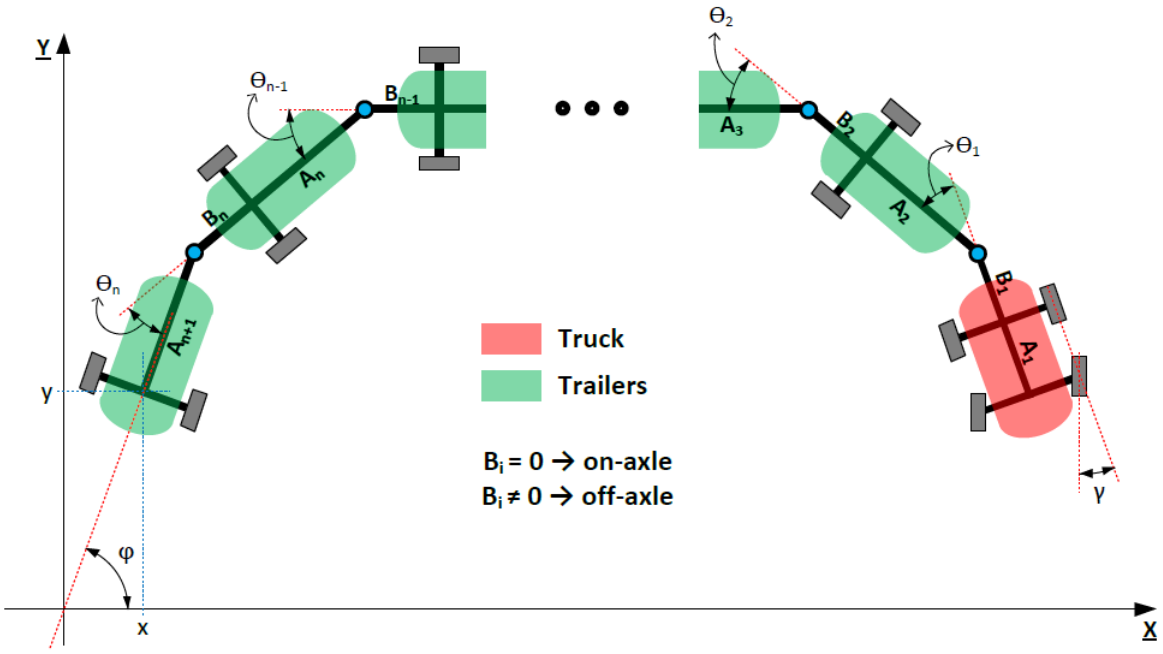
Fonte: Miranda, 2011, p. 39.

traseiras ou dianteiras, e pelo controle da direção de toda a composição, o qual é executado pelas rodas dianteiras. No caso específico deste projeto utilizou-se um *truck* de tração traseira. Os *trailers*, ou ainda semi-reboques, são elementos passivos que não detêm nenhum controle sobre a direção ou tração. A figura 3 mostra uma cadeia articulada genérica, ilustrando os ângulos da direção e de configuração (θ_1, θ_2, ... e θ_n), bem como os parâmetros geométricos da cadeia mecânica articulada, A_i e B_i e a diferença entre ligações *on-axle* e *off-axle* (PANDOLFI, 2012).

No caso específico deste trabalho, usa-se ligações *on-axle*, implicando em elementos B_i iguais a 0. Já o elemento *truck* também possui servos motores que movimentam a direção das rodas dianteiras, movendo-se com um grau de liberdade de $\pm 31^{o}$, aproximadamente, onde o sinal de + convencionou-se como indicando a esquerda em relação ao *truck*, ou seja, um sentido de giro anti-horário, e o sinal de − como indicando a direita em relação ao *truck* (sentido de giro horário). Para simbolizar o ângulo formado pelas rodas dianteiras, usar-se-á o símbolo γ, como na figura 3.

Em se tratando de manobras com RMMAs, a principal dificuldade está relacionada com movimentos em marcha ré. Nestas situações a composição apresenta um comportamento similar a um pêndulo invertido múltiplo horizontal, cuja modelagem e controle são não lineares e de elevada complexidade. Logo, o problema de controlar-se o protótipo para trás representa um desafio considerável do ponto de vista do controle, caso queira-se realizar alguma manobra específica.

Figura 3 – Cadeia Cinemática genérica de um RMMA



Fonte: Pandolfi, 2012, p. 20.

1.3 Revisão Bibliográfica

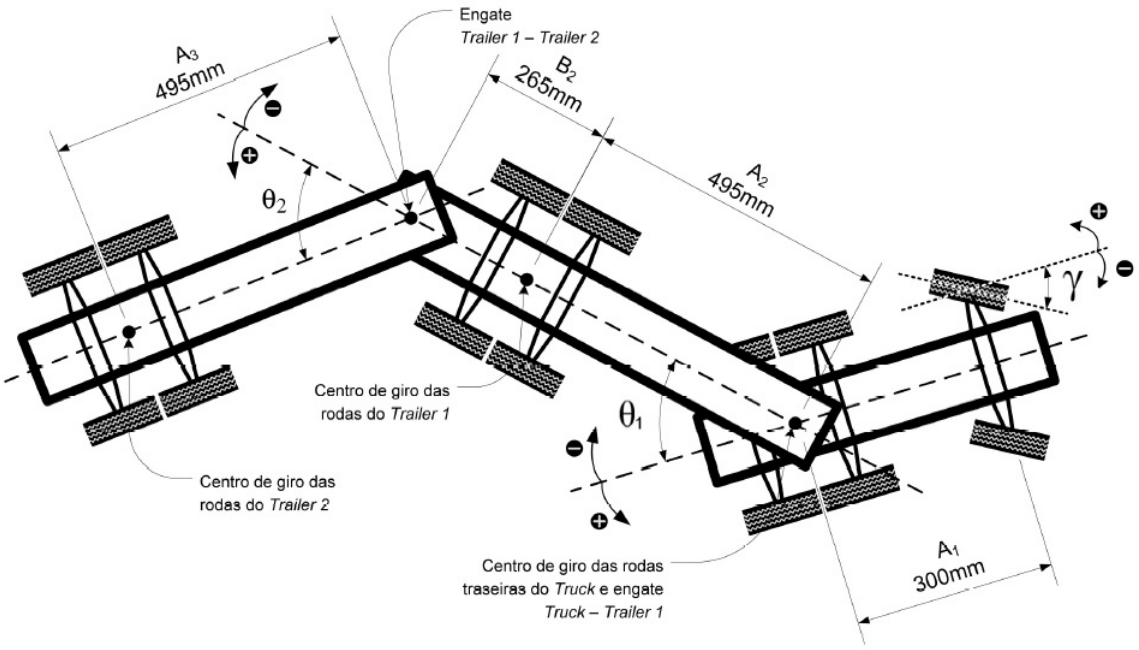
Vários trabalhos já foram desenvolvidos no LCI (Laboratório de Controle Inteligente) da UFES buscando resolver o problema de controle do sistema composto de um RMMA de três graus de liberdade, sendo o primeiro representativo do elemento *truck* e os outros dois dos elementos *trailers* ligados sucessivamente. A figura 4 representa o sistema com esta configuração.

Há diversas abordagens possíveis para o problema de controle deste tipo de RMMA, sendo as principais as que consideram modelos e as que baseiam-se em dados coletados da planta, como é o caso da abordagem *fuzzy*.

Em (BERTOLANI, 2011), usou-se do controlador desenvolvido em (KULITZ, 2003) a fim de fazer com que o RMMA realizasse manobras à ré, partindo de uma configuração qualquer e tendo como referência configurações tidas como desejadas. Esta abordagem utiliza-se de um controlador *feedback* não linear, o qual se baseava em equações de giro do protótipo.

Estas equações são recursivas, no sentido de que a referência para o ângulo θ_{n-1} é obtida através da referência desejada para o ângulo θ_n e do valor real medido desta variável. Seguindo este processo, a referência encontrada para θ_{n-1} serve para se encontrar a referência de θ_{n-2}, e assim sucessivamente, até o ângulo da primeira articulação θ_1. As

Figura 4 – Representação Geral do Elemento *Truck* e *Trailers*



Fonte: Miranda 2011, p. 43.

equações de 1.1 a 1.3 são a base para esta abordagem.

$$\theta_{1giro} = \arctan\left(\frac{A_2 \times \sin(\theta_2)}{A_3 + B_2 \times cos(\theta_2)}\right) \times sign(\theta_2) \tag{1.1}$$

$$\gamma_{giro} = \frac{1}{2} \times \arcsin\left(\frac{2 \times A_1 \times \sin(\theta_1)}{A_2 + B_1 \times cos(\theta_1)}\right) \times (-sign(\theta_2)) \tag{1.2}$$

$$\Delta\theta_1 = VT \left[\left(1 + \frac{\cos\theta_1 B_1}{A_2}\right)\frac{\cos\gamma \sin\gamma}{A_1} + \frac{\sin\theta_1}{A_2}\right] \tag{1.3}$$

Onde VT é a velocidade do protótipo e A_1, A_2 e B_1 são dados na figura 4

No caso de (BERTOLANI, 2011), o problema de controle foi tratado para duas articulações passivas, logo só foi necessário uma equação recursiva para se determinar o ângulo da última articulação com o *truck*.

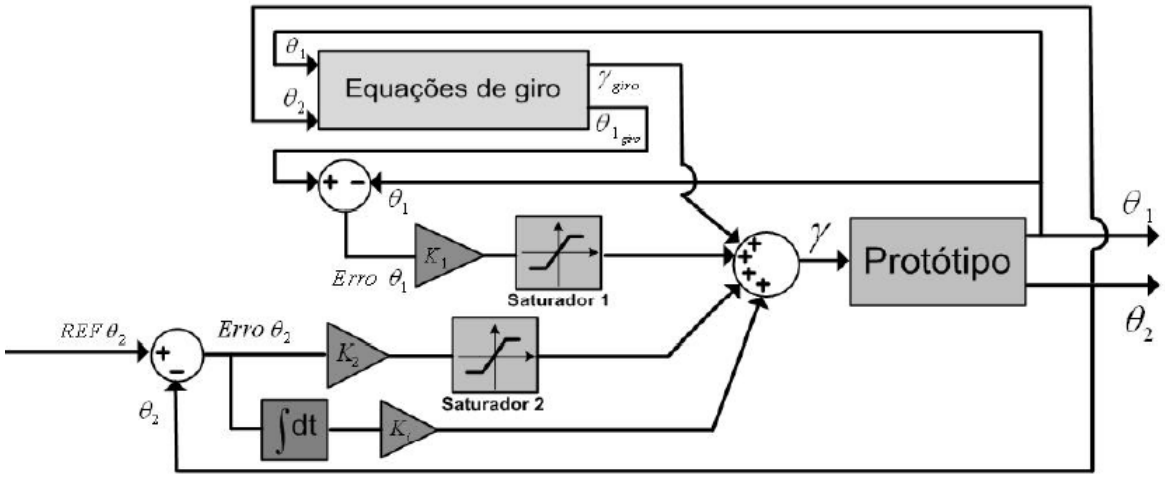
Os ângulos θ_n são formados pelos arcos entre o n-ésimo *trailer* e o elemento que o antecede. De forma arbitrária, definiu-se que as aberturas no sentido horário seriam negativas e no sentido anti-horário positivas. A partir dessas definições, chamou-se de configuração côncava aquelas nas quais os ângulos θ_1 e θ_2 possuem sinais opostos, e configuração convexa aquelas nas quais os ângulos θ_1 e θ_2 possuem sinais iguais.

As situações de configuração côncava são indesejáveis, pois elas tendem a fazer a configuração entrar em *jackknife*, ou seja, causar um engavetamento das articulações em forma de L.

Um dos principais desafios de (BERTOLANI, 2011) foi, justamente tratar da tarefa de tirar o conjunto de uma configuração côncava e pô-la em uma convexa. Configurações convexas são desejáveis, pois delas é mais fácil manter a estabilidade do conjunto, além de permitir "ângulos de giro" definidos, ou seja, fazer com que o conjunto *truck-traillers* gire segundo um arco com uma abertura estabelecida.

De posse das equações de giro, uma malha de controle foi feita de modo a controlar o sistema. Esta malha pode ser vista na figura 5. O modelo de giro é utilizado para se gerar a referência de θ_1, enquanto que, para θ_2, a referência já é pré-fixada.

Figura 5 – Malha de Controle *feedback* não linear



Fonte: Bertolani 2011, p. 42.

O sinal de controle para a direção γ é composto pela referência da direção, também obtida pelas equações de giro, pelo sinal de erro de θ_1 passando por um controlador proporcional e uma saturação, além do erro de θ_2 após passar por um controlador PI e outro saturador. Os valores dos ganhos dos controladores foram estipulados de forma empírica.

No entanto, deve-se ressaltar que a técnica de controle por meio das equações de giro possui limitações. As simplificações matemáticas feitas no modelo teórico limitam a validade dos intervalos de variação dos ângulos de formação. Por isso, manobras que requeiram grandes amplitudes, ou seja, variações angulares maiores do que 45º, tendem a não ser bem feitas através desta abordagem de controle.

Ainda em (BERTOLANI, 2011), a solução do controlador *feedback não-linear* foi comparada

à solução proposta por (PANDOLFI, 2012) de um controlador *fuzzy* centralizado de três entradas.

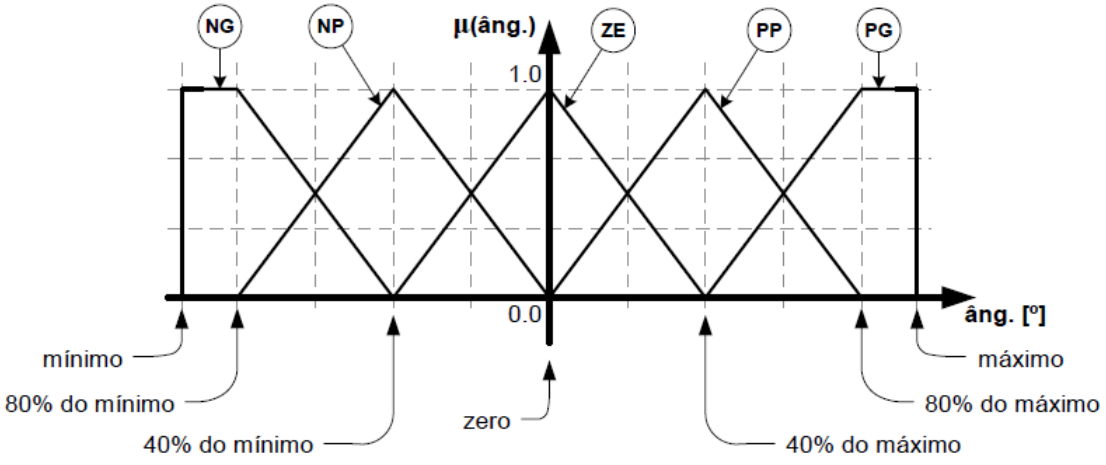
Na abordagem de (PANDOLFI, 2012), utilizou-se do método de Wang-Mendel para gerar regras a partir do banco de dados obtido através da movimentação do protótipo. A grande dificuldade na obtenção destes dados está no fato de que não era prático controlar o RMMA enquanto este se movia, pois as referências eram dadas via *keyboard* em um computador externo, variando em $\pm 1^{o}$ por vez. Assim, não era possível passar referências às rodas dianteiras de forma rápida o suficiente para evitar as situações de *jackknife*. Logo, nesta coleta de dados, utilizou-se da estratégia de pré-determinar a direção γ e os ângulos de configuração θ_1 e θ_2 antes de se colocar o protótipo em movimento. A movimentação em ré era então iniciada e só era interrompida após a configuração entrar em *jackknife* ou depois de um determinado tempo passar.

Essa limitação impossibilitava que fossem gerados dados em quantidade e variedade suficiente para que todas as regras fossem definidas sem a interferência de um especialista, além de diminuir a confiabilidade das que eram geradas.

Das 125 regras necessárias para relacionar as entradas a uma das saídas, 108, ou seja, 86% das regras precisaram da interferência de um especialista. Isso demonstra como a obtenção de dados ficou limitada por não haver como controlar o auto-modelo enquanto este se movia.

O modelo *fuzzy* direto possuía três entradas (θ_{1k}, θ_{2k} e $\Delta\theta_{2k}$) e duas saídas (θ_{1k+1} e θ_{2k+1}). Todas as variáveis foram particionadas de forma semelhante, como pode ser observado na figura 6.

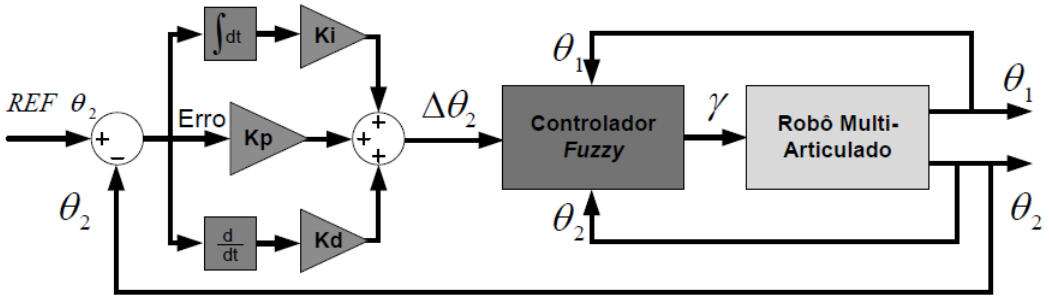
Figura 6 – Particionamento Definido Proporcionalmente



Fonte: Pandolfi 2012, p. 67.

Juntamente ao controlador *fuzzy*, foi utilizado um controlador PID a fim de diminuir as oscilações e erros presentes na planta. A configuração final do sistema de controle neste caso pode ser observada na figura 7. Os ganhos do controlador também foram determinados de forma empírica.

Figura 7 – Controle *Fuzzy* Centralizado de Três Entradas



Fonte: Bertolani 2011, p. 51.

Nos testes realizados em (BERTOLANI, 2011), percebeu-se um desempenho melhor para o método de controle *feedback* não-linear, quando comparado ao método de controle *fuzzy* de três entradas, principalmente em casos em que se desejava fazer uma mudança abrupta de referência para o ângulo da última articulação θ_2. A dificuldade em se obter regras em quantidade e representatividade suficiente pode explicar essa diferença, tendo em vista a necessidade que metodologias como a *fuzzy* têm por dados com essas características.

1.4 Caracterização do Problema de Controle e Modelagem

Sistemas automatizados em veículos móveis multi-articulados que possam auxiliar operadores a executar manobras em ré demonstram-se vantajosos, pois, ao serem movidos para trás, estes veículos tendem a sofrer um engavetamento ou *jackknife*, situação em que uma das articulações dobra-se formando um L, atrapalhando em sua movimentação de forma contínua para trás. Esta situação caracteriza a perda da estabilidade do sistema, pois, a partir de um determinado ângulo, chamado de ângulo crítico, só é possível reverter o arco movendo o veículo para frente e, justamente o que se deseja destes caminhões é que eles possam executar a manobra de posicionamento em ré apenas movendo-se para trás.

As situações de *jackknife* devem ser evitadas, não apenas pela perda de controle sobre as articulações passivas do veículo, impossibilitando a realização de manobras em ré, mas também porque elas podem ocasionar a colisão entre as partes móveis do robô, danificando-as (BERTOLANI, 2011). Em veículos reais carregados com toneladas em cargas isto seria ainda mais desastroso, pois haveria o risco de queda da carga e do próprio veículo, trazendo

não apenas um prejuízo material, mas podendo ocasionar acidentes e riscos para quaisquer indivíduos próximos.

Sob esta perspectiva, percebe-se que um sistema que consiga auxiliar os operadores a executar manobras em ré de uma forma mais fácil e segura seria extremamente relevante, por trazer mais efetividade e segurança para todo o processo. Pela abordagem clássica de controle, este problema se demostra extremamente complexo devido à não-linearidade do sistema e à complexa dinâmica física do mesmo. Isto dificulta o equacionamento do comportamento das articulações quando o protótipo move-se em ré, além de limitar a validade da solução encontrada, pois as simplificações assumidas diminuem as condições onde essas equações representam o sistema real de forma aceitável.

Nos trabalhos realizados por (PANDOLFI, 2012) e por (OLIVEIRA, 2010) foram utilizadas abordagens de controle *fuzzy* sobre o sistema. Estes acabaram tendo desempenhos inferiores aos obtidos por (BERTOLANI, 2011), pois, enquanto este esbarrava no problema de falta de generalidade do modelo analítico, aqueles enfrentaram problemas para se obter um controlador *fuzzy* que cobrisse todas as possibilidades de entrada e saída devido à falta de dados expressivos em quantidade suficiente. Essa dificuldade é explicada pelo fato de que, no processo de obter-se dados sobre o comportamento do sistema, após o protótipo ser posto em movimento, não modificava-se sua direção, sendo observado apenas seu comportamento até este entrar em *jackknife*. A única distinção entre os experimentos era o fato de que as configurações iniciais eram alteradas com a intensão de coletar dados do maior número de configurações possíveis. Entretanto, como não alterava-se a direção durante os experimentos, rapidamente a configuração entrava em *jackknife*, o que limitava o número de dados coletados e as configurações observadas. Ainda mais crítico nesta abordagem é observar que, da forma como ela foi feita, não coletava-se dados de transições, ou seja, de situações em que a direção γ é alterada ocasionando uma mudança na configuração das articulações.

Desta forma, com o intuito de superar esta dificuldade e melhorar os dados coletados durante os experimentos, substituiu-se o antigo sistema de comunicação por um comunicador com sinal AM, o qual permite que os sinais de referência sejam passados mais rapidamente à planta. Isto retira a limitação de ser necessário manter um computador externo próximo ao RMMA e facilita ao usuário controlar o movimento em ré, impedindo que ocorram engavetamentos com frequência, o que permite estender os tempos de cada teste. Assim, consegue-se fazer com que mais dados sejam coletados, mais configurações diferentes sejam representadas e que a influência da direção sobre as transições entre ângulos de formação seja observada. Embarcar o sistema de controle no microcontrolador utilizado também traz benefícios à etapa de coleta de dados, pois possibilita uma resposta mais rápida dos atuadores e sensores embarcados, por haver menos atrasos decorrentes da comunicação

com o computador externo.

Ao observar-se o trabalho realizado por (MIRANDA, 2011), percebe-se como esta atualização faz-se necessária a aquisição de dados, tendo um reflexo direto nos modelos e controladores gerados. Neste projeto, utilizou-se a técnica de redes neurais para levantar-se um preditor para o protótipo. A fim de obter dados para o treinamento da rede, repetiu-se o procedimento de posicionar o RMMA em certas configurações iniciais pré-estabelecidas e, posteriormente, fazê-lo mover-se em ré a uma velocidade constante até o ponto de *jackknife*. Essas posições eram formadas pelos ângulos de formação θ_1 e θ_2, os quais foram particionados em cinco valores: 0^o, $\pm 1^o$, $\pm 3^o$, $\pm 7^o$, $\pm 11^o$ (Valores positivos e negativos indicam o sentido de giro), e pelo ângulo da direção, o qual foi particionado em 0^o, $\pm 1^o$, $\pm 2^o$, $\pm 3^o$, $\pm 5^o$, $\pm 7^o$, $\pm 10^o$, $\pm 13^o$, $\pm 16^o$ e $\pm 20^o$. Através destas partições obteve-se 475 condições iniciais.

Todo o processo de posicionar o protótipo em uma das 475 posições e movê-lo para trás coletando os dados, demonstra-se extremamente cansativo e pouco prático, não permitindo que fossem coletadas informações representativas da maioria das configurações possíveis para o RMMA, sem o custo de um trabalho extremamente repetitivo. Ao observar os resultados obtidos por (MIRANDA, 2011), vê-se que durante os experimentos não foi exigido do sistema mudanças nos ângulos da direção, sendo desejado observar apenas o comportamento do protótipo enquanto este movia-se em ré a partir de certa configuração. Logo, nos dados coletados desta forma, não há informações sobre como ações sobre a direção mudariam a configuração do conjunto.

Assim, percebe-se que com a introdução do rádio AM, consegue-se tornar o trabalho de se obter dados muito mais rápido, ao permitir que o envio de comandos para os servos-motores da direção seja feito de forma direta. Isto viabilizou que, durante as manobras em ré, mudanças acentuadas na direção fossem efetuadas, possibilitando a obtenção de dados sobre como a direção influencia no comportamento do protótipo em situações de transição de giro.

Portanto, caso o trabalho de (MIRANDA, 2011) fosse repetido, utilizando-se do novo *hardware* implementado neste projeto, poder-se-ia obter resultados ainda melhores do que os apresentados por ele em uma faixa ainda mais ampla de ângulos. Devido a essa percepção, uma etapa fundamental deste presente trabalho foi a obtenção de dados do protótipo movendo-se em ré, de forma a coletar informações que abranjam a maior parte das configurações possíveis para o robô e a influência que mudanças na direção tinham sobre a configuração das articulações.

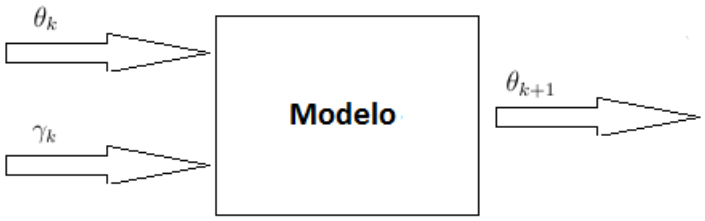
De posse destes dados, um dos desafios deste trabalho foi levantar um modelo do RMMA

e validá-lo ao compará-lo com os dados reais obtidos. Com isso, buscou-se demonstrar que a forma como estes dados foram coletados realmente melhorou a representatividade dos mesmos em relação à planta em estudo.

Já na elaboração do controlador *fuzzy*, deve-se ressaltar que o presente trabalho ateu-se em grande parte à reestruturação física do protótipo, logo trata-se de um novo sistema, para o qual as análises feitas anteriormente precisariam ser revistas ou refeitas. Tendo isso em vista, a estratégia de controle aqui considerada ainda foi bem inicial, envolvendo um controlador *fuzzy* isolado e a remoção de um dos elementos *trailers* passivos. Devido a esta simplificação, não há mais "configurações côncavas" ou "configurações convexas", além de diminuir o número de entradas do controlador *fuzzy* de três (θ_{1k}, θ_{2k} e ${\theta_{k+2}}^*$) para duas (${\theta_{k+1}}^*$ e θ_k) e de saídas de duas ($\Delta\theta_{2k}$ e α_k) para uma (α_{k+1}).

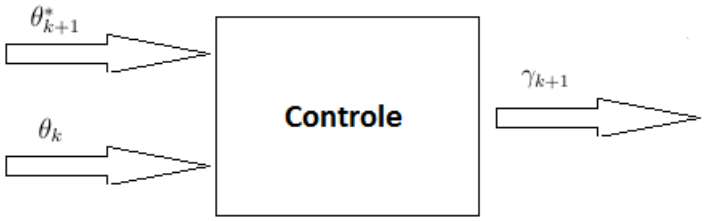
As figuras 8 e 9 esquematizam os problemas de modelagem e controle de forma simplificada, passando a haver duas entradas ao invés de três e apenas uma saída em ambos os casos.

Figura 8 – Representação do Modelo *Fuzzy* Utilizado Para Validação



Fonte: Próprio Autor, 2017.

Figura 9 – Representação do Controlador *Fuzzy* Proposto



Fonte: Próprio Autor, 2017.

Esta simplificação reduz a complexidade do controle, por diminuir exponencialmente o número de regras e torná-las mais intuitivas, mesmo para usuários pouco experientes com o protótipo. Apesar disto, a abordagem aqui feita pode ser reproduzida para casos com múltiplos *trailers*, sem que haja diferenças na metodologia.

Na etapa de elaboração do sistema de controle, inverteu-se o modelo de tal forma que θ_{k+1} passou a ser uma entrada, representando o valor futuro desejado para o ângulo θ da articulação. Outra entrada continuou sendo a posição atual do ângulo, ou seja, θ_k e a saída passou a ser o próximo comando para a direção γ_{k+1}, o qual deve levar a configuração à posição desejada.

A ideia por trás dessa inversão está no fato de que, se é possível presumir o valor de θ_{k+1} a partir dos valores atuais de θ_k e γ, então também pode-se inferir γ, caso saiba-se θ_k e seu próximo valor θ_{k+1}. Desta forma os dados utilizados para levantar-se um modelo *fuzzy* também poderiam ser usados na composição de um controlador *fuzzy*, porém utilizando θ_{k+1} como entrada ao invés de saída e γ como saída ao invés de entrada. Esta inversão pode precisar de algumas adaptações, mas a forma de obter regras a partir dos dados se mantém. No caso específico deste trabalho θ_{k+1} passaria a ser θ^*_{k+1}, ou seja, o ângulo em que se deseja chegar, podendo ser modificável através de um dos canais do rádio AM.

Outra opção seria tratar esta entrada como $\Delta\theta^*_{k+1}$, ou seja, a diferença entre o ângulo desejado e o ângulo atual. Esta mudança busca generalizar mais a tomada de decisão para os casos em que a referência está distante do ângulo atual, mas descrever melhor o comportamento necessário quando ambos estão próximos. Isto porque, desta forma, o comportamento no ponto de equilíbrio fica mais particionado, tornando o controle no entorno destas regiões mais preciso. Estas diferentes abordagens podem ser feitas com os mesmos dados coletados mudando apenas como eles são usados no processo de geração de regras.

1.5 Objetivos

1.5.1 Objetivos Gerais:

Este projeto objetiva a atualização de um auto-modelo RMMA já existente e o embarcamento de um sistema de controle baseado na teoria *fuzzy* sobre o protótipo com dois graus de liberdade.

1.5.2 Objetivos Específicos:

- Trocar o controlador embarcado atual por um mais moderno e mais flexível , no caso, um microcontrolador *Atmega328* incorporado a um *board* de desenvolvimento da família *Arduino*;
- Implementar o sistema de controle no microcontrolador embarcado, retirando a necessidade de realizar o processamento em um computador externo;

- Modernização de atuadores e sensores;
- Reestruturação física do protótipo, tanto no que se refere a *hardware*, quanto às partes mecânicas;
- Melhorar o dinamismo dos testes e coletas de dados, com a introdução de um rádio AM para enviar comandos ao protótipo e um módulo de gravação e leitura de dados SD embarcado ao sistema;
- Implementar um modelo *fuzzy* que inclua não-linearidades, ruídos, atrasos, perturbações e outros aspectos inerentes à planta real;
- Implementar um controlador *fuzzy* que permita ao protótipo a realização de manobras tanto com pequenas quanto com grandes variações de ângulos de formação.

1.5.3 Desdobramentos:

Com a obtenção destes objetivos, espera-se que sejam facilitados alguns aspectos que se demonstraram problemáticos e mesmo limitantes em trabalhos passados. Dentre as principais melhorias destacam-se:

- Criação de uma estrutura que permita novos testes e experimentos de forma mais simples devido à introdução do rádio AM;
- Facilitar a coleta de dados do sistema incorporando um gravador SD à placa embarcada;
- Possibilidade de explorar outras técnicas de controle além da *fuzzy*, tornando o protótipo uma planta eficiente para a realização de pesquisas.

1.6 Metodologia

A elaboração deste trabalho dividiu-se basicamente em quatro etapas: o estudo teórico das técnicas de controle *fuzzy* que poderiam ser utilizadas para modelar e controlar o protótipo; a análise da estrutura encontrada no RMMA e quais seriam as modificações necessárias para atingir-se os objetivos propostos; a obtenção de um modelo *fuzzy*, a partir dos dados coletados, que represente o sistema real e a implementação de um sistema de controle, também a partir de dados, o qual possa ser aplicado à planta real e; finalmente, a realização de testes que permitam analisar se o controle proposto de fato melhora a capacidade do RMMA de realizar manobras em que deve-se executar grandes mudanças em seu ângulo de formação, mantendo-se nestas referências após tê-las alcançado, além de auxiliar os operadores a se desviar de obstáculos quando estes estiverem presentes.

1.6.1 Estudo Teórico

Nesta etapa, tomou-se a teoria de conjuntos *fuzzy* como ponto base na elaboração de um modelo que pudesse representar a planta. Também utilizou-se desta teoria para se construir um controlador que permitisse a realização de manobras em ré. Aqui, discutiu-se os principais aspectos da teoria *fuzzy* e as formas de inferência de saída mais comuns, explicando-as a fim de tornar claro como este processo será feito pelo controlador embarcado.

Técnicas de controle baseadas nos conjuntos *fuzzy* dependem geralmente do conhecimento de um especialista, ou de dados que possam demonstrar o funcionamento de um dado sistema. Devido à introdução do rádio AM, bem como do gravador SD, abriu-se a possibilidade de armazenar dados de forma mais eficiente e rápida do que vinha sendo feito até então. Assim, pôde-se obter informações mais expressivas sobre o sistema, permitindo a elaboração de um modelo bem representativo do RMMA e de um controlador que saiba atuar sobre a planta de forma mais precisa.

1.6.2 Avaliação e Reestruturação Física do Protótipo

Nesta etapa, avaliou-se o estado do sistema embarcado presente no RMMA e quais partes ainda estavam funcionais, tendo em vista o tempo em que o mesmo esteve inativo. Devido ao fim da vida útil do microcontrolador, bem como de alguns sensores, optou-se por mudá-los por outros mais novos, porém acessíveis, de forma a permitir que o projeto possa ser reproduzido ou mesmo consertado, caso hajam tais necessidades.

Na escolha de um novo microcontrolador decidiu-se por um da família *Arduino*, os quais possuem varias faixas de preços, são amplamente comercializados, permitindo fáceis reposições, possuem materiais didáticos disponíveis gratuitamente para usuários e podem ser adaptados facilmente entre suas versões.

A introdução do rádio AM requereu adaptações na placa, sendo necessária a introdução de um circuito integrado que elevasse o sinal enviado por ele, além da conexão entre a placa do *Arduino* com o circuito receptor do rádio AM. Para a introdução do gravador SD também foi necessária a adequação de níveis de tensão entre o microcontrolador e o gravador, além de conectar ambas as placas, utilizando-se de pinos específicos do *Arduino* para isso.

1.6.3 Criação de um Modelo e Implementação do Sistema de Controle

Nesta etapa são discutidos os testes realizados para obter-se as massas de dados que foram usadas na elaboração do modelo e do controlador. Uma vez que tenha-se obtido uma quantidade de dados satisfatória sobre o comportamento do novo sistema, passou-se para

a etapa de criação do modelo *fuzzy* e, posteriormente, do controlador, também *fuzzy*, para a planta. Neste ponto é tratada a forma como o modelo e o controle foram elaborados, ressaltando pontos sobre o processo de inferência de saídas *fuzzy*.

Além disso, simulações serão utilizadas a fim de verificar se o controlador obtido pode ser implementado na planta, passando a atuar no protótipo real do problema em estudo.

1.6.4 Testes e Análise de Resultados

Por fim, serão realizados testes os quais intencionam explorar as capacidades e limites da nova planta do sistema construído.

Nestes testes busca-se observar alguns comportamentos básicos das respostas: a capacidade do sistema de controle de manter o RMMA movendo-se em ré segundo certo ângulo θ^* (entre o elemento *truck* e o *trailer*) dado como referência e sua capacidade de chegar a essa referência, independentemente da posição inicial do protótipo. Também analisou-se o esforço de controle, ou seja, como a direção das rodas dianteiras comportou-se durante estas manobras e se este comportamento foi como o esperado segundo o valor teórico dado nas equações que descrevem estes movimentos.

Outro teste realizado foi a capacidade do sistema de controle de auxiliar na execução de manobras complexas, nas quais era requerido do auto-modelo que este contornasse obstáculos executando um movimento de "oito" em volta deles. Estes testes buscam validar tanto as modificações de *hardware* quanto o controlador em si, indicando como o controle *fuzzy* pode ser melhorado quando consegue-se dispor de dados que realmente representem o sistema analisado.

2 EMBASAMENTO TEÓRICO

2.1 Modelagem e Controle *Fuzzy*

Enquanto que para muitos processos consolidados existam soluções que, se não ótimas, funcionam de forma eficaz, pois se tratam de tecnologias das quais o conhecimento sobre elas sustenta-se em fundações bem estabelecidas na literatura, há outros casos em que ainda enfrenta-se grande dificuldade em estabelecer sistemas de controle eficientes para os problemas apresentados. Uma dificuldade recorrente é o fato de diversas plantas serem fortemente não-lineares, de forma a ser necessário a utilização de técnicas sofisticadas de controle as quais nem sempre possuem uma metodologia simples de sintonia, tornando o ajuste de controladores extremamente complicado para não especialistas, os quais acabam recorrendo a métodos empíricos, o que resulta em sistemas funcionando aquém do que poderiam, em termos de eficiência.

De fato, mesmo sistemas de controle construídos através de técnicas clássicas de linearização e levantamento de modelos matemáticos apresentam imprecisões e discordâncias em relação aos resultados obtidos pelo sistema físico, tendo em vista o número de simplificações e aproximações feitas para que o modelo analítico pudesse ser descrito. Isso levanta o questionamento de até que ponto abordagens clássicas de controle realmente representam soluções factíveis para problemas reais de automação.

Uma das abordagens que busca oferecer alternativas aos métodos clássicos é a modelagem e controle *fuzzy*. Esta técnica busca justamente superar as dificuldades existentes na modelagem clássica através de uma lógica subjetiva, buscando traduzir a complexidade matemática em um conhecimento de certa forma intuitivo.

Esta abordagem permite uma solução de controle *model-free*, ou seja, sem a necessidade de um modelo matemático que represente o sistema que deseja-se controlar. Entretanto, ele precisa da existência de uma quantidade de dados considerável e representativa de todos os possíveis conjuntos de entradas e saídas para que seja possível levantar regras que repliquem a dinâmica de controle desejada. Outra opção complementar a essa é a utilização de um especialista que possa estabelecer quais relações de entradas e saídas devem ser feitas para o funcionamento do sistema de controle.

Este tipo de modelagem se baseia na forma humana de pensar, incorporando o fator da subjetividade com a qual a mente quantifica conceitos como alto, caro, rápido, muito

etc. Além do mais, utilizando esses conceitos, é possível criar conjuntos de regras que estabelecem relações de causa e conseqüência as quais serão responsáveis por produzir saídas *fuzzy*, sendo estas *defuzzyficadas* posteriormente. Simplificadamente, a relação causal de variáveis *fuzzy* funciona se atribuindo pesos a comparações de situações para se chegar à conseqüência destas. Em outras palavras:

Se A e B aconteceram, então C acontecerá

Por exemplo: *"se o entregador de pizza demorar e eu estiver com muita fome, certamente eu me irritarei."*

Termos como *demorar* e *muita* são subjetivos, entretanto eles têm a função de ponderar ideias como *tempo de entrega* e *fome*, o que influenciará no resultado *irritabilidade*.

A principal característica da modelagem *fuzzy* que permite expressar subjetividade advém da diferenciação dada ao conceito de pertinência existente nesta modelagem quando comparada à forma como isso é definido para conjuntos clássicos. Nos conjuntos clássicos a pertinência de certo valor a um dado conjunto é definida de forma brusca: ou o elemento pertence ou não ao conjunto (fronteira abrupta). A conjuntos com este conceito de pertinência, dá-se o nome de conjuntos *crisp*. Já em conjuntos *fuzzy*, a pertinência de certo elemento em algum conjunto é um número real variando dentro do intervalo [0, 1]. Quanto mais perto a pertinência do elemento no conjunto estiver de 1, maior será o grau de pertencimento deste elemento ao conjunto (ASKAR-ZADEH, 1965).

A pertinência de determinado elemento é dada pela equação 2.1:

$$\mu(x),\ onde\ 0 \leq \mu(x) \leq 1 \tag{2.1}$$

Sendo μ(x) a função de pertinência de x no conjunto A.

A definição formal de um conjunto *fuzzy*, segundo (FERREIRA, 2009a) Ferreira é:

> Considere o conjunto U, onde os elementos de interesse estão definidos. U é denominado universo de discurso. Um conjunto *fuzzy* A no universo de discurso U, pode ser definido como o conjunto de pares ordenados:

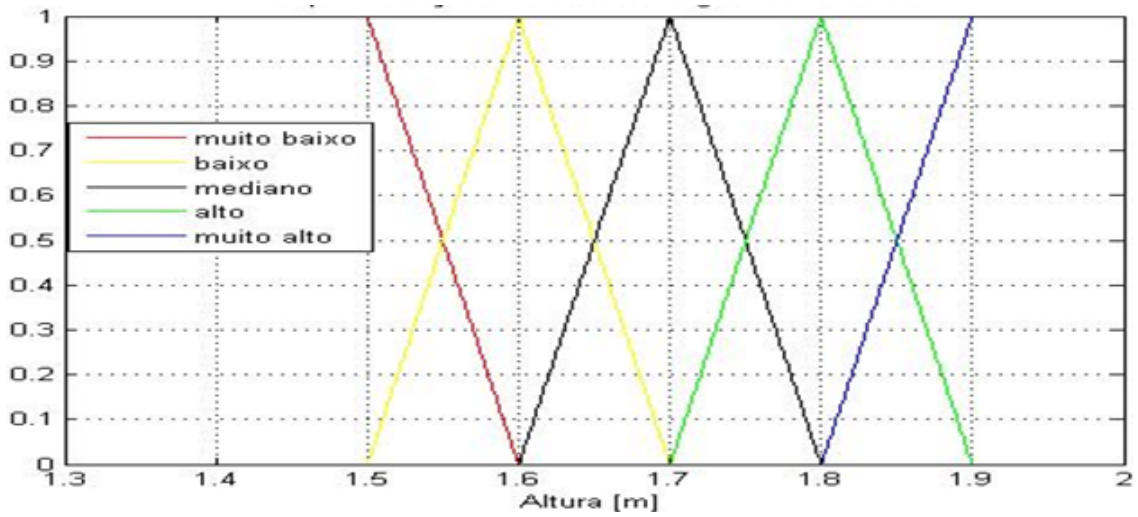
$$A = \{(x, \mu_A(x))/x \in U\}$$

$\mu_A(.)$ é a função de pertinência ou função característica de A. $\mu_A(x)$ é o grau de pertinência de X em A (quantifica a pertinência).

A ideia de dar pesos a certos conceitos (pertinência de elementos dentro de conjuntos) permite incorporar a subjetividade inerente na visão de mundo e experiência dos seres humanos, pois certas definições dependem do indivíduo que a faz. Por exemplo: uma pessoa de 1.75 (m) é alta ou baixa? Em países como Timor Leste certamente essa pessoa seria considerada alta, tendo em vista que a média de altura neste país é inferior a 1.70 (m) para homens, enquanto o mesmo não poderia ser dito em países escandinavos onde a média de altura da população masculina tende a ser maior do que 1.80 (m), por exemplo.

Dentro da modelagem de conjuntos *fuzzy* existe o conceito de variável lingüística. A variável lingüística teria a função de descrever certo conhecimento e as características possíveis deste conhecimento. Por exemplo, tomando novamente o conceito de alto, seja a variável lingüística altura. Uma pessoa pode ser classificada como: muito alta, alta, mediana, baixa ou muito baixa. Cada uma dessas classificações representa valores assumidos pela variável lingüística altura dentro do universo *fuzzy* de discurso. A figura 10 seguinte ilustra o exemplo dado:

Figura 10 – Representação da Variável Lingüística "altura"



Fonte: Próprio Autor, 2017.

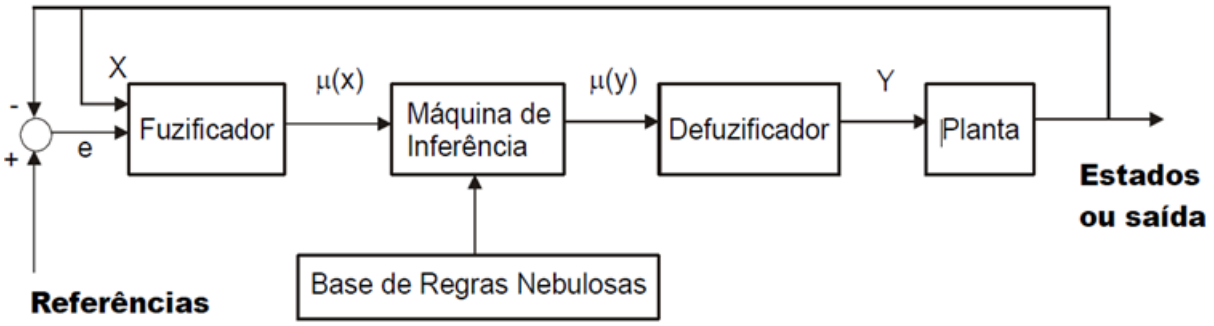
Não necessariamente as representações dos possíveis valores das variáveis lingüísticas precisam ser triangulares. Especificamente neste trabalho, adotou-se curvas normais para representar cada valor. Geralmente, busca-se colocar o *crossing point*, ou seja, o ponto em que dois valores subseqüentes se encontram em 0,5, por igualizar as chances de todas as regras, entretanto isto também não é obrigatório e depende do entendimento e percepção do projetista sobre qual é a melhor configuração para cada variável lingüística.

Como discutido anteriormente, os conceitos de *alto* e *baixo* são subjetivos e podem variar dependendo do contexto de quem os define. A grande vantagem desta modelagem está no fato de que, dependendo da interpretação de quem está definindo a variável lingüística *altura*, modificar o que é considerado *muito alto* ou *muito baixo*, por exemplo, pode ser facilmente feito para um problema semelhante mais em um contexto distinto.

Além disso, como fora dito anteriormente, podem-se criar regras causais entre as variáveis *fuzzy* de uma modelagem e, a partir destas regras, gerar variáveis lingüísticas de saída que incorporem os resultados (conseqüências) das regras existentes. Essa saída *fuzzy* pode ser traduzida para valores *crisp* e ser usada como sinal de controle para a planta.

O estágio responsável por gerar estas saídas a partir das variáveis lingüísticas *fuzzy* é a chamada máquina de inferência. No controlador *fuzzy* este componente é responsável por utilizar a base de regras para gerar a saída *fuzzy* correspondente às entradas. Esta saída será posteriormente traduzida para uma variável *crisp* através do componente *defuzzyficador*. A figura 11 a seguir ilustra uma estrutura simples de um controlador *fuzzy*.

Figura 11 – Arquitetura Básica de um Sistema de Controle *Fuzzy*



Fonte: Ferreira, 2009, p. 63.

A base de regras é responsável por conter as informações sobre as relações causais entre as possíveis combinações das entradas *fuzzy*. Este conhecimento pode provir tanto da experiência de um especialista em dado processo quanto a partir de um grande banco de dados com medições das correspondentes saídas para certos valores de entrada. Como foi dito anteriormente, a máquina de inferência é responsável pelas tomadas de decisão através do "raciocínio aproximado", resultando nas variáveis lingüísticas de saída. Por fim, estas saídas são *defuzzyficadas*, tornando-se sinais de controle (FERREIRA, 2009b).

Pela estrutura apresentada, vê-se que, antes e depois de utilizar a máquina de inferência para simular o raciocínio humano, é preciso *fuzzyficar* os sinais medidos pelos sensores. Esse processo é realizado pelo *fuzzyficador*. Segundo (FERREIRA, 2009c) o *fuzzyficador* converte variáveis medidas em conjuntos *fuzzy* ou em valores do universo de discurso, associados às pertinências de cada termo *fuzzy*.

Neste tipo de modelagem, pode-se inclusive incluir a precisão da medida através da incerteza associada ao valor medido. Caso se tenha confiança no sensor e no valor medido por ele, pode-se associar ao valor medido um *singleton*, o que é definido como:

$$\mu_A(x) = 1, \ onde \ x = x_0 \tag{2.2}$$

$$\mu_A(x) = 0, \ caso \ contrário \tag{2.3}$$

Desta forma o sistema entenderá que o valor medido possui um erro desprezível, de forma a não afetar a ação de controle. Em contrapartida, pode-se associar certa imprecisão em torno do valor medido, adotando-se uma função de pertinência em torno do valor x_0. A forma da função de pertinência que descreve esta incerteza pode ser definida de diversas formas, desde gaussianas até funções triangulares simples. Cabe à experiência do projetista determinar a forma e o *range* adequado para a medida.

Após esta etapa, as variáveis *fuzzyficadas* precisam passar pela máquina de inferência. Esta utiliza os bancos de regras para gerar as saídas. As regras no geral possuem o formato descrito na equação 2.4. De forma mais direta, a equação 10 descreve a relação causal das regras em controladores *fuzzy*:

$$Se \ A_1 \ e \ B_1, \ então \ C_1 \tag{2.4}$$

De uma forma mais geral:

$$Se \ A_n \ e \ B_n, \ então \ C_n \tag{2.5}$$

Para gerar a saída *fuzzy* a máquina de inferência utiliza as operações de máximo e mínimo representadas por max ($\vee$) e min ($\wedge$), respectivamente. Basicamente, o operador max realiza a união entre dois conjuntos enquanto que o operador min realiza a interseção. Para se obter o conjunto C, o qual seria a saída da máquina de inferência, primeiro, deve-se determinar os níveis $\alpha_n's$ para cada conjunto de regras pela operação:

$$\alpha_{cn} = \mu_{An}(x_0) \wedge \mu_{An}(y_0) \tag{2.6}$$

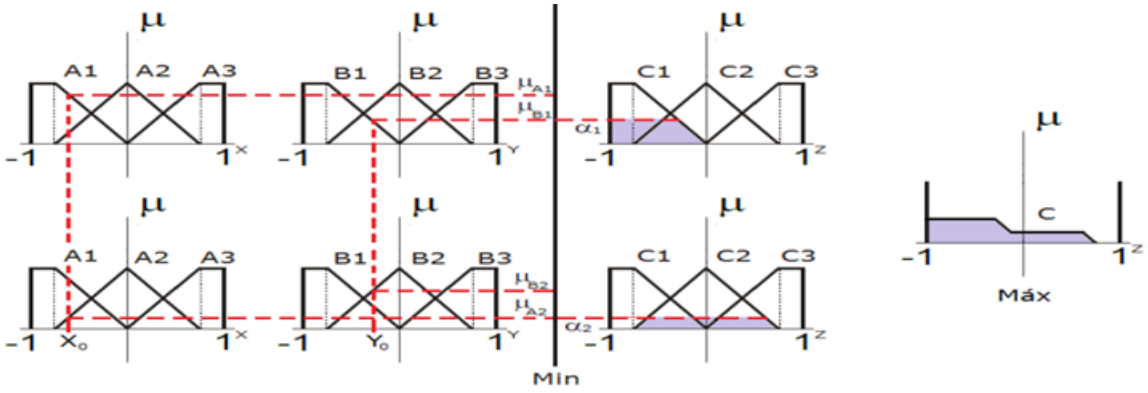
Onde x_0 e y_0 são entradas *singleton*.

O conjunto μ_c é obtido pela união das interseções entre os níveis α_{cn} e os conjuntos $C'_n s$. Formalmente essa operação é expressa por:

$$\mu_c = (\alpha_{c1} \wedge C_1) \vee (\alpha_{c2} \wedge C_2) \vee ... \vee (\alpha_{cn} \wedge C_n) \tag{2.7}$$

Esse raciocínio é exemplificado pela Figura 12:

Figura 12 – Raciocínio para Geração de Saída *Fuzzy* da Máquina de Inferência



Fonte: Silva, 2014, p. 25.

A função de pertinência μ_c do conjunto C é a saída *fuzzy* da máquina de inferência, sendo a representação das conseqüências das entradas *fuzzyficadas* depois de serem processadas pela máquina de inferência segundo as regras contidas no banco de regras.

Para que esta saída possa atuar sobre a planta, ela precisa assumir um valor *crisp*. Isto é feito através do processo de *defuzzyficação*. Este processo não possui uma sistemática definida, sendo passível da escolha do projetista sobre qual seria a forma mais coerente de realizar esta conversão. Entretanto existem duas metodologias que normalmente são adotadas: a metodologia de centros de massas ou centro de gravidade (CG) e a metodologia da média dos máximos (MM) (FERREIRA, 2009d).

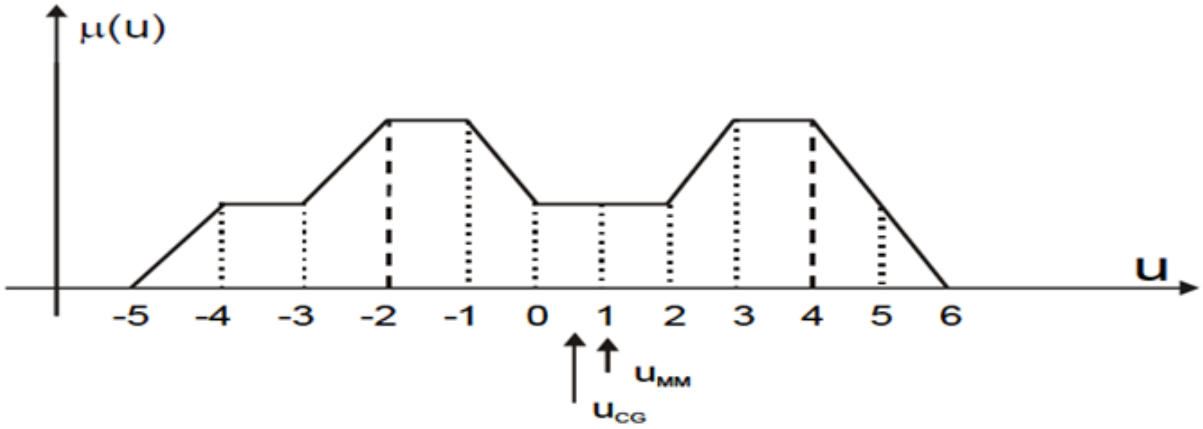
Segundo (FERREIRA, 2009d) a metodologia de médias máximas é feita calculando-se as " ...médias dos dois elementos do universo de discurso que tem a maior pertinência". Esse conceito poderia ser estendido e, ao invés de se escolherem apenas dois elementos, poder-se-ia selecionar três ou quatros valores de máxima pertinência de elementos no universo de discurso, entretanto esta metodologia ainda ignoraria o efeito de outros elementos no universo de discurso que também compõem o conjunto de saída, mesmo que em uma menor pertinência.

A metodologia de centro de massas, por outro lado, considera em certo grau o efeito de todos os elementos contidos dentro do conjunto *fuzzy* de saída. Nesta metodologia,

calcula-se o centro de massa da curva de pertinência do conjunto *fuzzy* de forma a dividir a área do conjunto em duas partes de áreas numericamente iguais.

A Figura 13 ilustra estas metodologias:

Figura 13 – *Defuzzyficação* para a Média dos Máximos e Centros de Gravidade



Fonte: Ferreira, 2009, p. 76.

Onde:

- U_{MM}: Saída *crisp* para a metodologia de média dos máximos.
- U_{CG}: Saída *crisp* para a metodologia do centro de gravidade.

Deve-se notar que na abordagem *fuzzy* o controlador obtido é único para o sistema sobre o qual ele atua, uma vez que este é ajustado justamente para considerar todos os aspectos individuais da planta. Isto pode tornar o trabalho de ajuste do controlador extremamente complicado pela falta de metodologias diretas para isto.

2.2 Geração de Regras e Inferência de Saídas

A geração de regras e a forma de obter saídas numéricas das inferências lingüísticas feitas pelos sistemas *fuzzy* compõem etapas essenciais no funcionamento do sistema de controle que se utiliza desta técnica. Portanto, deve-se conhecer as principais abordagens possíveis para que se escolha aquela que melhor atenderá às necessidades do projeto em questão.

As principais metodologias na literatura para tais tarefas são (WANG; MENDEL, 1992) e (TAKAGI; SUGENO, 1985). Ambas metodologias não diferem em termos da obtenção de regras, as quais podem ser, tanto via dados existentes sobre o sistema que deseja-se controlar, quanto pelo conhecimento especialista do mesmo. As diferenças entre elas, na

verdade, apresentam-se quanto a forma de inferir respostas e em como as saídas são entendidas.

Em (WANG; MENDEL, 1992) a saída é uma variável lingüística, assim como as entradas e precisa que seu valor *fuzzy* seja convertido em um valor *crisp* para servir de sinal para o sistema. Já em (TAKAGI; SUGENO, 1985) as saídas são equações (geralmente lineares), já em valores *crisps* que retornam o valor da saída do controlador diretamente ao sistema, sem precisarem ser *defuzzyficadas*. A maior dificuldade encontrada em (TAKAGI; SUGENO, 1985) é justamente estabelecer quais equações representaram da melhor maneira possível o comportamento desejado para a saída do controlador. Ambas metodologias serão explicadas adiante neste trabalho, a fim de justificar-se a escolha de uma ao invés da outra.

2.2.1 Metodologia de Geração de Regras e Inferência de Wang-Mendel

Em (WANG; MENDEL, 1992) é apresentada uma abordagem que permite sintonizar um controlador *fuzzy* através da aprendizagem via exemplos. Por meio deste método, consegue-se gerar regras *fuzzy* a partir de pares de dados numéricos e, utilizando-se das regras geradas, fazer um mapeamento entre o espaço de entradas e o espaço de saídas. Todo este procedimento é dividido em 5 etapas:

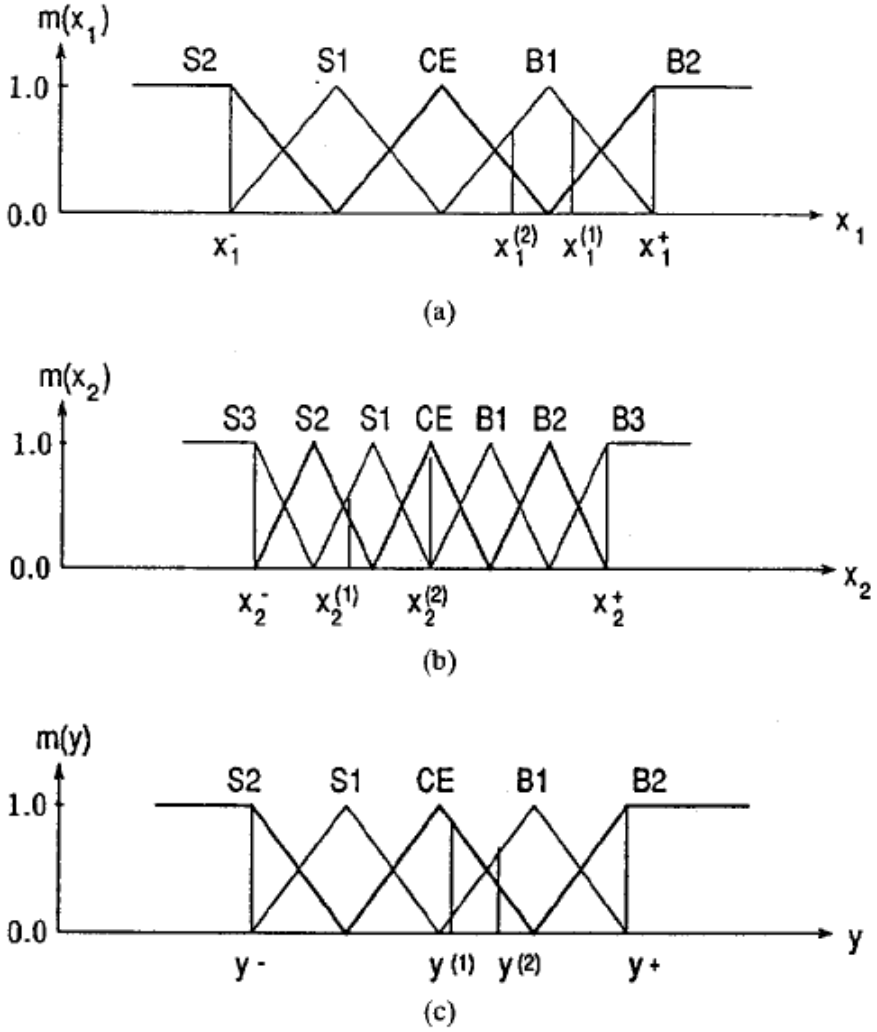
- 1ª Etapa: dividir os espaços de entrada e saída em regiões *fuzzy*. Nesta etapa, delimita-se o intervalo em que as variáveis independente e dependentes podem variar, ou seja, seu domínio. Cada domínio é dividido em 2N + 1 regiões, sendo que N pode variar dependendo da variável e as regiões não precisam ser necessariamente iguais. A figura 14 exemplifica esse processo. Neste exemplo, x_1 e x_2 são as variáveis independentes e y a variável dependente. A forma das regiões não precisam ser triangular nem iguais as demais, como dito anteriormente, ficando a cargo do projetista decidir qual formato melhor desempenhará a função desejada.

- 2ª Etapa: gerar regras *fuzzy* a partir de pares de dados disponíveis. Nesta etapa, basicamente se utiliza dos conjuntos de dados relacionando quais foram as entradas para determinada saída, mapeando estes valores dentro das regiões *fuzzy*. Toma-se como regra a região *fuzzy* com maior pertinência para uma dada variável. Por exemplo, ainda na figura 14, tem-se que:

$$({x_1}^{(1)}, {x_2}^{(1)}; y^{(1)}) \Rightarrow [{x_1}^{(1)}\ (0.8\ em\ B1,\ max),\ {x_2}^{(1)}\ (0.7\ em\ S1,\ max), \\ y^{(1)}(0.9\ em\ CE,\ max)] \Rightarrow Regra1 \quad (2.8)$$

  Logo se x_1 = B1 e x_2 = S1, então y = CE

Figura 14 – Divisão dos Espaços de Entrada e Saída em Regiões *Fuzzy*



Fonte: Wang-Mendel, 1992, p. 1416.

$$(x_1{}^{(2)}, x_2{}^{(2)}; y^{(2)}) \Rightarrow [x_1{}^{(2)}(0.6\ em\ B1,\ max), x_2{}^{(2)}(1.0\ em\ CE,\ max), \\ y^{(2)}(0.7\ em\ B1,\ max)] \Rightarrow Regra2 \quad (2.9)$$

Logo se x_1 = B1 e x_2 = CE, então y = B1

E assim por diante.

- 3ª Etapa: assinalar níveis (ou degraus D) para as regras. Obviamente, podem haver regras redundantes ou contraditórias. Redundâncias apenas indiciam a validade de uma regra, enquanto que contrariedades podem indicar uma divisão inadequada das regiões, muito embora, na prática, seja quase impossível não haver regras conflitantes. Nestes casos, decide-se qual regra será tomada como verdadeira multiplicando as pertinências de cada variável. O que apresentar a maior pertinência global será adotada como regra.

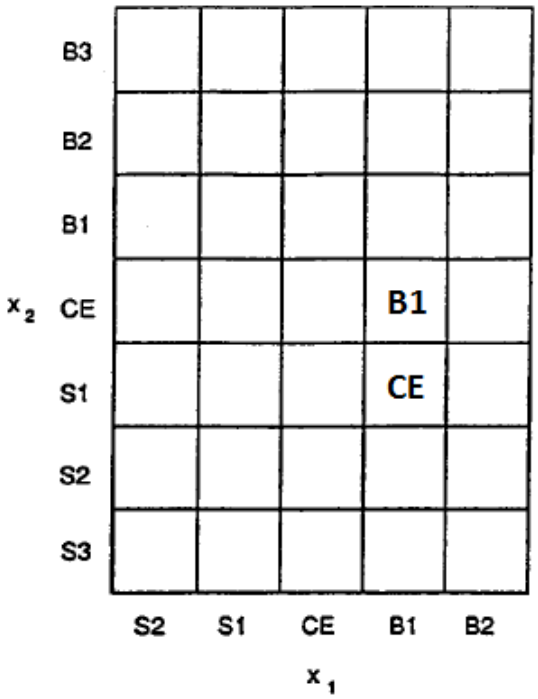
Um fator humano pode ser adicionado para indicar confiabilidade ou não no conjunto de entradas e saídas utilizado. Caso não haja confiança na acurácia e/ou na precisão de algum conjunto de dados pode-se adicionar um fator multiplicativo às pertinências em ordem de favorecer ou não conjuntos de dados mais ou menos confiáveis. Ainda há a possibilidade de um especialista saber de antemão que certa regra acontece ou precisa acontecer para determinado processo. Logo, pode-se atribuir um fator que favoreça tal regra em detrimento das outras possibilidades. A equação 2.10 representa esse procedimento utilizando como exemplo as regras definidas no item 2:

$$D(Regra1) \ = \ \mu_{B1}(x_1)\mu_{S1}(x_2)\mu_{CE}(y)m \tag{2.10}$$

Onde m é um fator multiplicativo adotado arbitrariamente pelo projetista pelas razões dadas anteriormente. Caso se deseje imunizar a obtenção de regras da ação humana basta fazer $m = 1$ em todos os casos.

- 4ª Etapa: criação de uma base de regras *fuzzy*. Esta etapa apenas explicita uma representação do conjunto de regras obtidas através deste método. A figura 15 ilustra o caso de duas entradas e uma saída, como no caso do exemplo dado nos itens anteriores. Vale notar que múltiplas saídas significariam múltiplas tabelas e mais entradas aumentariam a dimensão das tabelas: 3 entradas formariam um cubo, 4 um tesserato (análogo ao cubo em 4 dimensões espaciais), etc.

Figura 15 – Formato da tabela de regras *fuzzy* para duas entradas e uma saída



Fonte: Wang-Mendel, 1992, p. 1416.

A representação da figura 15 indica que a saída é encontrada dando as entradas como coordenadas. No caso da regra 1, quando x_1 = B1 e x_2 = CE, y = B1 e no caso da regra 2, quando x_1 = B1 e x_2 = S1, y = CE.

Com o restante dos dados disponíveis, deve-se completar todas as lacunas da tabela. Como já discutido, muitos dados darão informações repetidas ou contraditórias, além do mais alguns casos podem não ser contemplados. Tudo isso pode deixar a tabela incompleta. Por isso, deve-se ter extremo cuidado na aquisição dos dados para que eles completem a maior e mais significativa quantidade de regras. Mesmo tendo-se este cuidado, no caso de haver regras não contempladas, um especialista pode preenchê-las segundo a sua experiência e conhecimento sobre o processo.

- 5ª Etapa: *defuzzyficar* as saídas segundo a base de regras criada. Nesta etapa deve-se utilizar algum método de *defuzzyficação* para desconverter a saída *fuzzy* em valores *crisp*. Algumas técnicas como a das "médias máximas" e o do "centro de massas" já foram explanadas anteriormente e podem ser utilizados nesta última etapa.

2.2.2 Metodologia de Geração de Regras e Inferência de Takagi-Sugeno

A metodologia descrita em (TAKAGI; SUGENO, 1985) atem-se basicamente a 3ª, 4ª e 5ª etapas do item anterior, sendo que as duas anteriores podem ser repetidas neste caso.

O primeiro a se fazer, após se ter estabelecido quais serão as regras de implicação é decidir quais equações descreverão as relações de causa-conseqüência determinadas. Isso pode ser feito através do conhecimento de um especialista que tenha noção numérica dos valores que as variáveis de saída devem assumir em determinados casos, mas na presença de vários dados de entradas e saídas do sistema, abre-se a possibilidade de se utilizar de métodos numéricos para se determinar quais são as equações mais apropriadas. Um desses métodos possíveis é o de Mínimos Quadrados. Nele, pode-se recorrer a diversos tipos de funções (exponenciais, senoidais, lineares, etc) para representar algum conjunto de pontos com o melhor *fit*, ou seja, com a melhor aproximação possível. Considere a equação 2.11.

$$E = \sum_{i=1}^{N} [y_i - f(x_i)]^2 \tag{2.11}$$

Onde: $y = f(x)$ e N é um dado conjunto de pontos.

Por simplicidade, considere que $f(x)$ é uma equação linear, então:

$$E = \sum_{i=1}^{N} [y_i - (a_n x_{ni} + ... + a_1 x_{1i} + a_0)]^2 \tag{2.12}$$

Os valores de $x_n...x_1$ são conhecidos, bem como os valores de y, logo a equação 2.12 é uma função não linear cujas variáveis são os coeficientes $a_n...a_1$ *e* a_0. Esta função possui um mínimo para cada coeficiente os quais podem ser encontrados por meio das derivadas parciais de cada um deles (GILAT; SUBRAMANIAM, 2008). Logo, tem-se que:

$$\frac{\partial E}{\partial a_0} = -2\sum_{i=1}^{N}(y_i - a_n x_{ni} - ... - a_1 x_{1i} - a_0) \tag{2.13}$$

$$\frac{\partial E}{\partial a_1} = -2\sum_{i=1}^{N}(y_i - a_n x_{ni} - ... - a_1 x_{1i} - a_0)x_{1i} \tag{2.14}$$

$$\frac{\partial E}{\partial a_n} = -2\sum_{i=1}^{N}(y_i - a_n x_{ni} - ... - a_1 x_{1i} - a_0)x_{ni} \tag{2.15}$$

Assim, há $n+1$ equações com $n+1$ variáveis, o que permite encontrar os valores dos coeficientes $a_n, ..., a_1 e a_0$. Portanto:

$$na_0 + \left(\sum_{i=1}^{N} x_{ni}\right) a_n + ... + \left(\sum_{i=1}^{N} x_{1i}\right) a_1 = \left(\sum_{i=1}^{N} y_i\right) \tag{2.16}$$

$$\left(\sum_{i=1}^{N} x_{ni}\right) a_0 + \left(\sum_{i=1}^{N} x_{ni}^2\right) a_n + ... + \left(\sum_{i=1}^{N} x_{1i}x_{ni}\right) a_1 = \left(\sum_{i=1}^{N} y_i x_{ni}\right) \tag{2.17}$$

$$\left(\sum_{i=1}^{N} x_{1i}\right) a_0 + \left(\sum_{i=1}^{N} x_{ni}x_{1i}\right) a_n + ... + \left(\sum_{i=1}^{N} x_{1i}^2\right) a_1 = \left(\sum_{i=1}^{N} y_i x_{1i}\right) \tag{2.18}$$

Pelas equações 2.16, 2.17 e 2.18 consegue-se resolver este sistema de forma a obter a função que melhor aproxima os pontos obtidos

Os tipos de equações de saída ficam a cargo do projetista, o qual deve conciliar a precisão e complexidade necessárias. Para muitos propósitos, equações lineares são suficientes, mas, caso seja necessário, equações exponenciais ou polinomiais podem ser adotadas a fim de representar comportamentos mais gerais para as saídas.

Uma vez que tenha-se encontrado equações que conectam as saídas com as entradas, estas devem ser divididas entre as regras criadas, ou seja, cada equação será a "conseqüência" de determinada regra gerada nos processos anteriores.

Juntamente com a "conseqüência" de cada regra, deve-se estabelecer o "peso" para cada uma delas, dependendo das entradas. Portanto, o próximo passo é determinar o método de inferência que envolva as variáveis lingüísticas e estas saídas.

Seja ω o peso de cada conjunto de entrada para cada regra existente. Isso pode ser feito de várias formas. Uma das mais convencionais é através do produto das pertinências de cada entrada por uma regra AND, exemplificada na equação 2.19.

$$\omega_i = \mu_1(X_{1i})\mu_2(X_{2i})...\mu_n(X_{ni}) \tag{2.19}$$

Onde ω_i representa o peso da i-ésima regra e $\mu_n(X_{ni})$ a pertinência da n-ésima entrada da regra i.

Também seja a regra de saída dada por:

$$Z_i = a_1X_1 + a_2X_2 + ... + a_nX_n + b_0 \tag{2.20}$$

Onde:

- Z_i: Saída, já em um valor *crisp*, da i-ésima regra;
- a_n: Coeficiente multiplicativo;
- X_n: Entradas;
- b_0: Coeficiente linear.

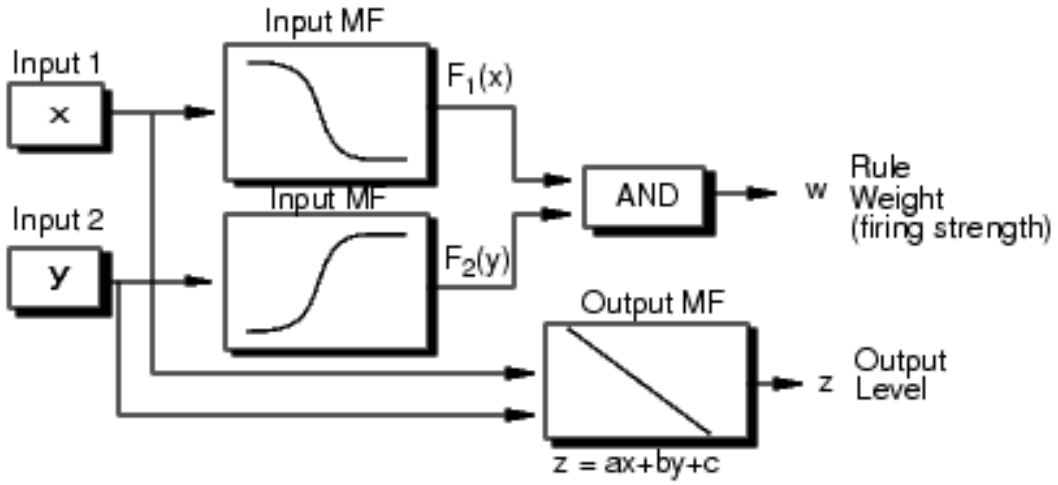
Assim, cada saída Z_i terá associada a si um peso ω_i. A média ponderada, mostrada na equação 2.21, de todas as saídas com seus respectivos pesos dará o valor de saída na metodologia de inferência proposta por Takagi-Sugeno (MATHWORKS, 2015). A figura 16 exemplifica todo o processo.

$$\text{Final Output} = \frac{\sum_{i=1}^{N} \omega_i Z_i}{\sum_{i=1}^{N} \omega_i} \tag{2.21}$$

Embora haja bibliotecas do Arduino para implementação de sistemas *fuzzy* através da inferência Wang-Mendel, estas ocupam uma grande parte de memória e sobrecarregam o processamento, atrapalhando a execução de outras funções do código, como leitura de dados e carregamento destes no cartão de memória. Além do mais, essas bibliotecas possuem muito mais funções do que serão realmente utilizadas neste projeto, logo recorrer a elas não caracteriza uma solução próxima da ótima. Assim, decidiu-se por implementar as regras e o sistema de inferência *fuzzy* diretamente no código.

A função *gen2fis* do *software* $Matlab^{©}$ cria regras e saídas, a partir de um conjunto de dados, utilizando-se da abordagem proposta por Takagi-Sugeno. Pela facilidade de se transpor um código em $Matlab^{©}$ para o C++ do Arduino, optou-se por utilizar o *software* para projetar o sistema de controle e depois replicá-lo no controlador. Como a função *gen2fis* já cria as regras e otimiza as equações de saída através da técnica dos mínimos quadrados, também optou-se por utilizar a abordagem Takagi-Sugeno para realizar a inferência das saídas do sistema.

Figura 16 – Diagrama de Inferência Takagi-Sugeno



Fonte: MATHWORKS, 2015.

3
REESTRUTURAÇÃO DO HARDWARE

Para apresentar quais mudanças foram feitas no *hardware* do protótipo, primeiramente deve-se descrever come ele se encontrava anteriormente, bem como suas principais características.

3.1 *Hardware* Antigo

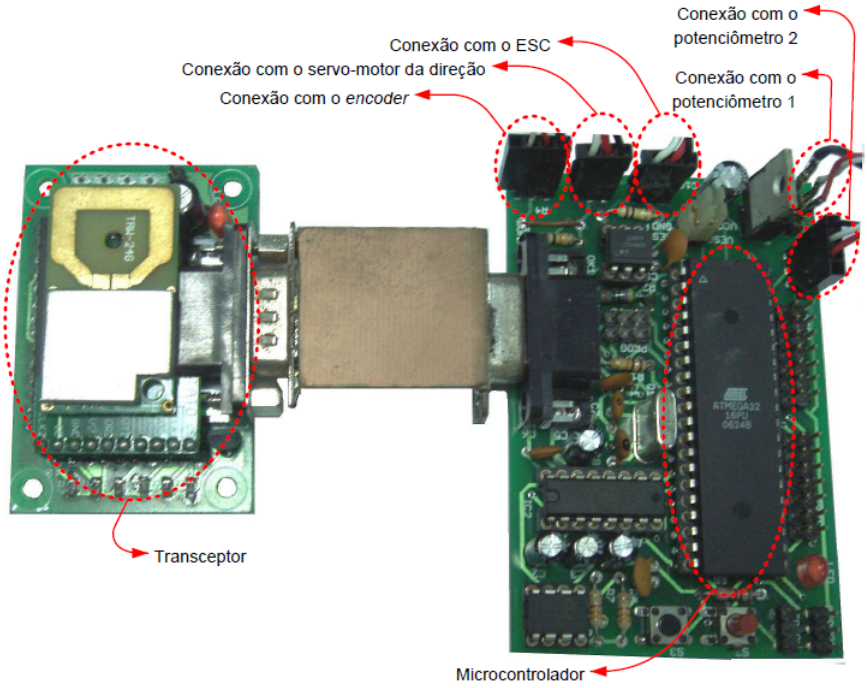
O sistema embarcado do protótipo possuía um microcontrolador *Atmega32* da *Atmel Products* com 32kB de memória *flash*, 2kB de memória RAM, 40 pinos e até 16MHz de freqüência de processamento. Este controlador tinha por função acionar o mecanismo de direção, leitura dos sensores, enviar o sinal de controle para o *driver* de corrente que aciona o motor C.C., acionar os servos motores que selecionam a marcha e comunicação com um computador externo. O controle de alto nível era realizado de modo *off-line*, logo todo o processamento mais complexo envolvendo o controle de manobras e definição das referências teve de ser executado por outro computador de maior desempenho, o qual transmitia a ação de controle ao controlador embarcado. A figura 17 mostra o sistema antigo.

O sistema de comunicação foi implementado com transceptores, modelo TRW-2.4 fabricado pela *Laipac Tech Inc.*, operando na freqüência de 2.45 GHz e permitindo uma transferência de dados entre uma faixa de 250 $kbps$ a 1 $Mbps$.

Tanto o computador quanto o microcontrolador utilizam um sistema de comunicação em rádio freqüência (RF) para transmitir e receber dados entre si. Assim, foram utilizadas duas placas semelhantes. A figura 17 mostra o transceptor conectado à placa embarcada, enquanto que a figura 19 mostra a placa conectada ao computador externo (ATMEL, 2017).

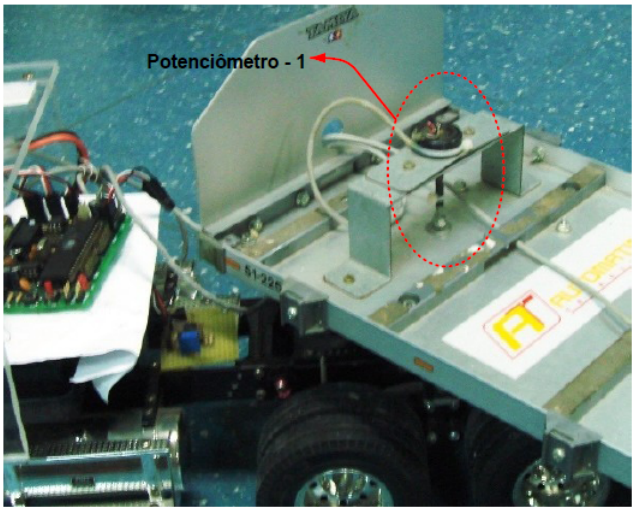
Já o sistema de medição Angular é composto por dois potenciômetros acoplados mecanicamente às articulações. Um dos potenciômetros pode ser visto na figura 18. A tensão nos terminais dos potenciômetros varia de acordo com a variação angular da articulação a qual ele está acoplado. O subsistema em questão capta apenas variações angulares das articulações, obrigando assim, que o usuário assuma algum referencial para que se conheça o valor do ângulo em qualquer outro instante. A equação 3.1 mostra o cálculo realizado pelo sistema antigo para encontrar o ângulo medido.

Figura 17 – Principais Componentes do Antigo Sistema Embarcado



Fonte: Pandolfi, 2012, p. 49.

Figura 18 – Potenciômetro e Acoplamento Mecânico dele à Articulação



Fonte: Pandolfi, 2012, p. 51.

$$Valor_{angulo} = \frac{< valor_{lido} > - < valor_{lidopotalinhado} >}{2.844} \tag{3.1}$$

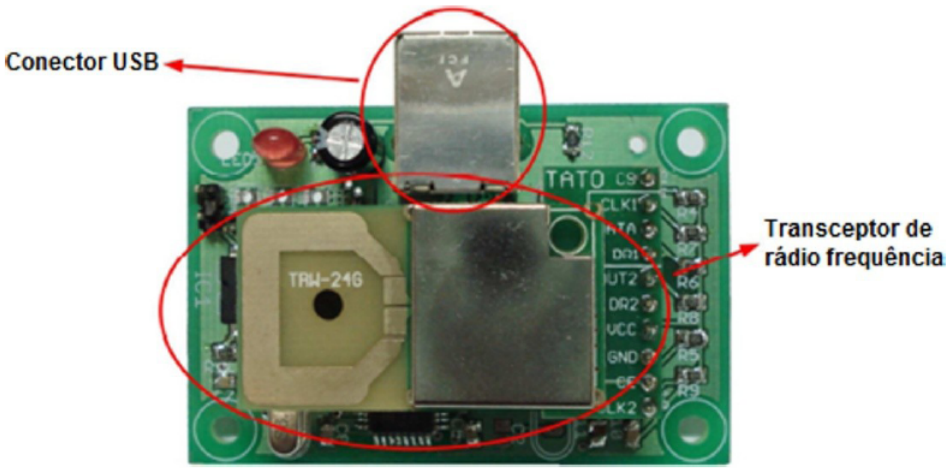
Deve-se esclarecer que este cálculo não é realizado pelo sistema embarcado, devendo o usuário calibrar a composição e, usando a equação 3.1, calcular os valores dos ângulos em seu programa.

Quando analisada, a fiação dos potenciômetros não estava funcional, tendo se rompido. Além disso, havia folga entre o potenciômetro e o eixo ao qual ele estava conectado. Estes problemas foram resolvidos com uma manutenção e reconexão da fiação, mas o potenciômetro linear foi mantido, por ser uma forma simples, porém efetiva, de se medir os ângulos das articulações.

Assim, fora os potenciômetros que passaram por manutenção e o *encoder*, os dispositivos que foram substituídos são: ESC, microcontrolador, receptor e rádio AM. O gravador SD foi adicionado, e não substituído, pois a gravação de dados era feita antes no computador externo. Além disso, uma bateria de polímero de lítio de 5200 mAh e 7,4V foi acrescida para aumentar a autonomia do protótipo. A antiga bateria de níquel-hidreto metálico de 2500 mAh e 7,2V passou a ser usada para alimentar o microcontrolador, já que baterias simples de 9V não apresentavam carga suficiente para manter novo controlador funcional por um tempo satisfatório.

Por fim, uma das engrenagens da caixa de redução apresentou-se quebrada e precisou ser consertada, uma vez que este defeito fazia com que o motor não conseguisse tracionar as rodas. A escolha pelo conserto ao invés da troca foi feita porque esta engrenagem em específico era muito pequena e de difícil manufatura. Além disso, esta não é uma peça facilmente encontrada no mercado e, as que foram encontradas, possuíam uma relação de dentes diferente da peça defeituosa, o que mudaria a relação de torque e velocidade do RMMA. Logo, optou-se por tentar consertá-la antes de se buscar pela alternativa da compra ou manufatura de outra. Esta escolha demonstrou-se suficiente, pois o reparo realizado foi efetivo em fazer a caixa de redução funcionar apropriadamente.

Figura 19 – Sistema de Comunicação RF com Entrada USB para Conexão com Computador

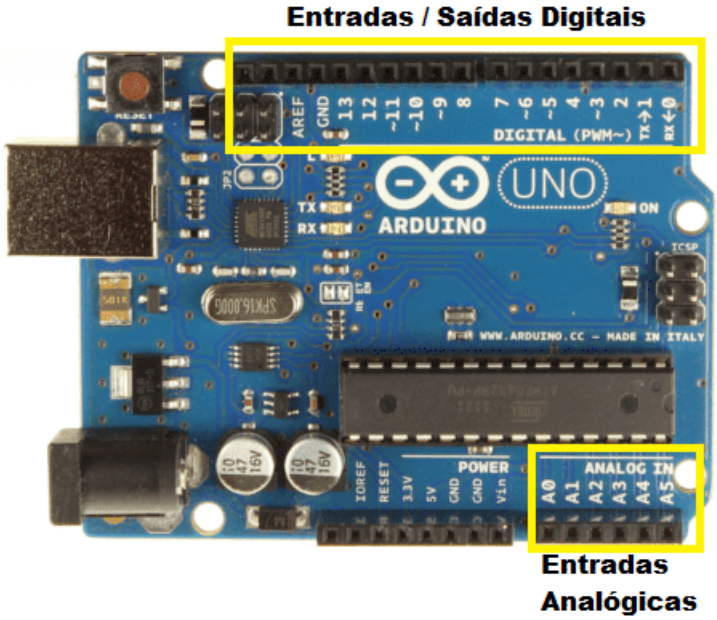


Fonte: Miranda, 2011, p. 48.

3.2 Placa Arduino

O microcontrolador antigo utilizado para fazer o controle sobre o protótipo foi substituído pela placa de desenvolvimento *Arduino UNO*. Esta é a placa mais simples de sua família. Ela possui 6 entradas analógicas, além de 14 pinos digitais que podem ser utilizados como entrada ou saída. Deste 14 pinos, 6 podem funcionar como saídas PWM de 8 bits, uma vez que a placa não possui uma saída genuinamente analógica.

Figura 20 – Pinos de Entrada e Saída da Placa de Desenvolvimento Arduino UNO



Fonte: Souza, 2013.

Quadro 1 – Características do Arduino UNO

Característica	
Microcontrolador	ATmega328
Tensão Operacional	5V
Voltagem de entrada (recomendada)	7-12V
Tensão de entrada (limites)	6-20V
Pinos E/S digitais	14 (dos quais 6 podem ser saídas PWM)
Pinos de entrada analógica	6
Corrente CC por pino E/S	40 mA
Corrente CC para pino 3,3V	50 mA
Flash Memory	32 kB (ATmega328). 0,5 kB utilizados pelo *bootloader*
SRAM	2 kB (ATmega328)
EEPROM	1 kB (ATmega328)
Velocidade Máxima de Operação	20 MHz

Fonte: Cisco, 2015

Esta placa, portanto utiliza um microcontrolador parecido com o anterior, o ATmega328, porém com uma velocidade máxima de operação um pouco superior (de

16MHz para 20 MHz). Os 14 pinos digitais operam até 5V e podem ser utilizados tanto como entradas quanto saídas. Cada um recebe ou fornece no máximo 40mA e possuem internamente resistores de *pull-up* (desconectados por padrão) de 20 a 50 kΩ. Deve-se salientar que alguns pinos podem exercer funções específicas:

- **Serial: 0 (RX) e 1 (TX).** Recebe e transmite dados seriais TTL através dos pinos RX e TX, respectivamente.
- **Interruptores Externos: 2 e 3.** Estes pinos podem ser configurados para disparar interrupções externas.
- **SPI: 10(SS), 11(MOSI), 12(MISO) e 13(SCK).** Estes pinos dão suporte à comuicação SPI utilizando a biblioteca SPI.
- **PWM: 3, 5, 6, 9, 10 e 11.** Estes pinos podem ser utilizados como saídas PWM de 8 bits através da função **analogWrite()**.
- **LED de teste: 13.** Este pino está conectado a um LED. Quando o pino recebe um sinal HIGH o LED acendo, já quando o sinal recebido é LOW ele permanecerá apagado.
- **I2C: 4(SDA) e 5(SCL).** Estes pinos fornecem suporte à comunicação I2C (TWI) utilizando a biblioteca *Wire*.
- **AREF.** Este pino pode fornecer uma tensão de referência diferente dos 5V padrão através da função **analogReference()**.
- **Reset.** Este pino apaga o programa gravado no microcontrolador quando recebe um valor LOW.

O arduino ainda possui 6 entradas analógicas de 10 bits, o que representa 1024 valores possíveis de representação. Como já foi mencionado, por padrão estes pinos funcionam entre 0 e 5V, mas podem ter seu limite máximo modificado através do pino **AREF** e da função **analogReference()**.

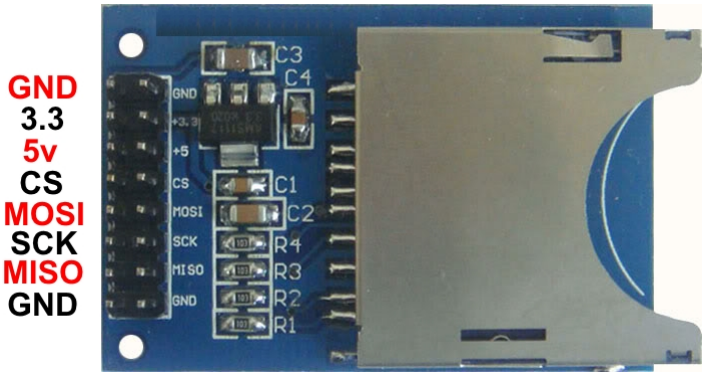
Deve-se notar que alguns pinos possuem mais de uma função específica, logo o projetista que for utilizar esta placa deve se atentar a quais funções ele precisará utilizar, a fim de não ocupar algum pino que seria necessário em outra função.

3.3 Módulo Gravador/Leitor SD e *Level Converter*

Para se obter os dados de velocidade, direção e orientação relativa do *trailer* foi utilizado um gravador SD, o qual gravava os dados enviados pelo microcontrolador em um cartão SD de 4GB.

Como pode ser percebido pela figura 21, o módulo se comunica com o Arduino através da interface SPI, entretanto os pinos CS (ou SS), MOSI e SCK funcionam

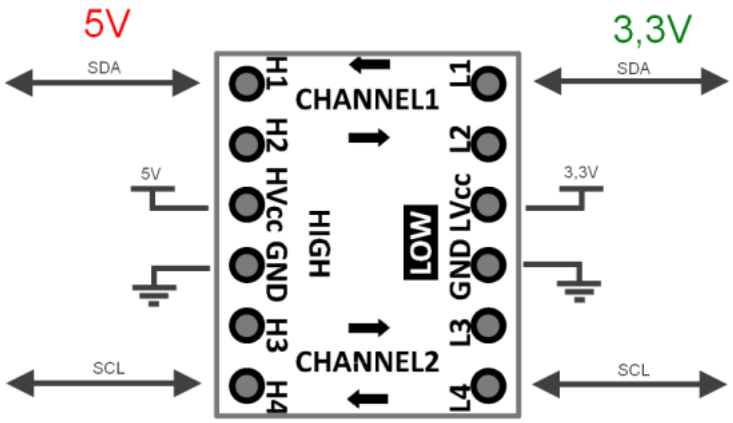
Figura 21 – Módulo para cartão SD



Fonte: FILIPEFLOP, 2017.

em um nível de tensão menor (3.3V) do que os 5V padrão destes pinos. Por isso foi utilizado um *Bidirectional Level Converter* para que a comunicação com o módulo pudesse ser feita sem que este sofresse algum dano.

Figura 22 – Conversor de Nível Bidirecional



Fonte: MSX elektronika, 2013.

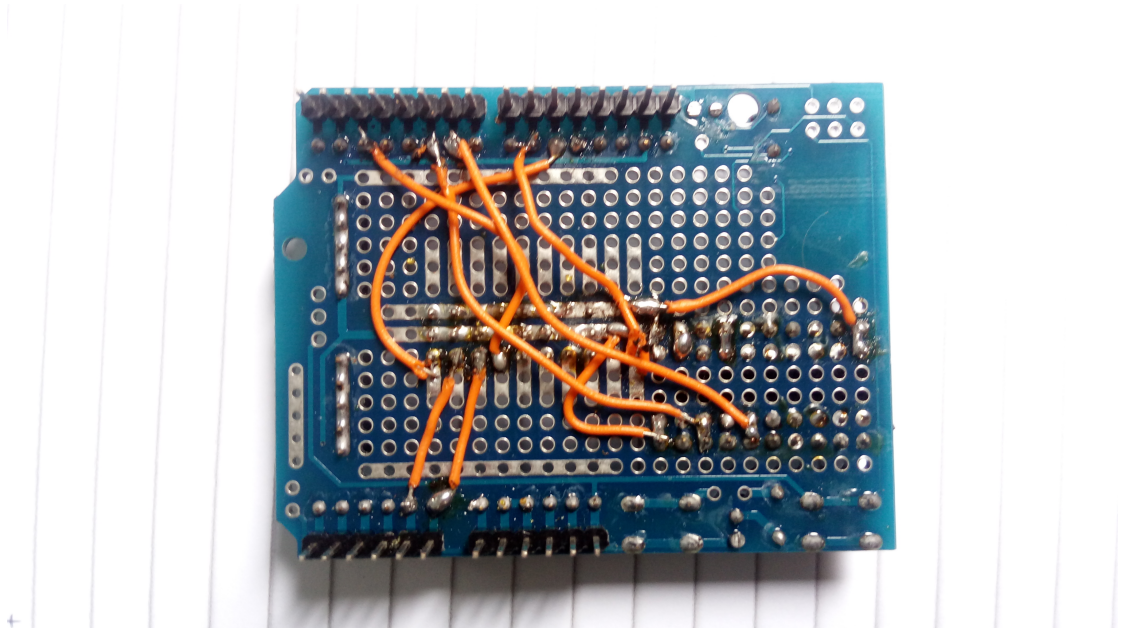
Como pode ser observado na figura 22, a referência de 5V é posta no pino HVcc e a de 3,3V no pino LVcc. Os pinos 4 (SS), 11 (MOSI) e 13 (SCK) vêm do *Arduino*, passam pelos pinos H*'i*, onde *i* varia de 1 a 4, e saem nos respectivos pinos L*'i* com a tensão reduzida para 3.3V. Daí, os sinais são encaminhados para os correspondentes pinos no módulo do cartão SD.

3.4 *Shield* e *Octal Buffer Driver* 74HC244N

A fim de facilitar a conexão entre o *Arduino* e os sensores e atuadores utilizados, recorreu-se a um *shield* já apropriado para o modelo UNO deste microcontrolador. Com este *shield*, pôde-se fazer soldas, mostradas na figura 23, mais facilmente por

baixo dele e disponibilizar pinos conectores com os quais os sensores e atuadores podem se conectar ao *Arduino* mais facilmente. Desta forma ficam a mostra apenas as conexões de dispositivos externos ao microcontrolador, facilitando a visualização.

Figura 23 – Soldas Embaixo do *Shield*



Fonte: Próprio Autor, 2017.

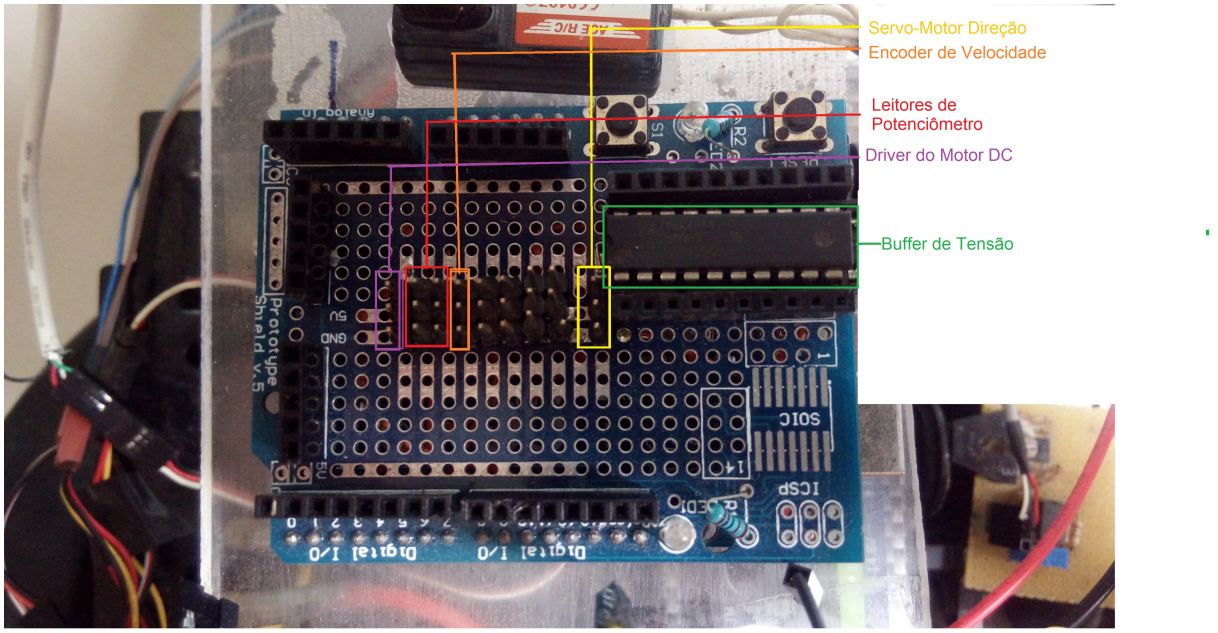
Visto de cima, os pinos postos para fazer a conexão com os sensores e atuadores externos são mostrados na figura 24. Por padrão, cada seqüência de pinos vem em três, sendo que um é para o GND, outro para o Vcc e por fim um é para o sinal. Há 5 trios de pinos, sendo eles para o servo-motor da direção, para o *encoder* de velocidade, para os dois leitores dos potenciômetros e para o *driver* de corrente utilizado para o motor de tração.

Também, na parte superior do *shield*, na figura 24, pode-se ver o circuito integrado 74HC244N. Este CI é um *buffer* de tensão de 8-bits o qual eleva sinais enfraquecidos, normalmente sendo utilizado em sistemas de transmissão de dados. Aqui, este dispositivo foi utilizado porque os sinais vindos do Receptor AM têm valores reduzidos, em torno de 1,8V, o qual não consegue ser diferenciado como um valor alto pelo *Arduino*. Assim este CI eleva os sinais provenientes do Receptor para no máximo 5V, mantendo a forma da onda dos dois canais.

3.5 *Encoder* e Estimação de Velocidade

Encoders são dispositivos utilizados para estimar posição ou velocidade de outros equipamentos. A técnica e tecnologia utilizada para efetuar tais estimativas variam, indo de técnicas mais simples e imprecisas a mais complexas e caras.

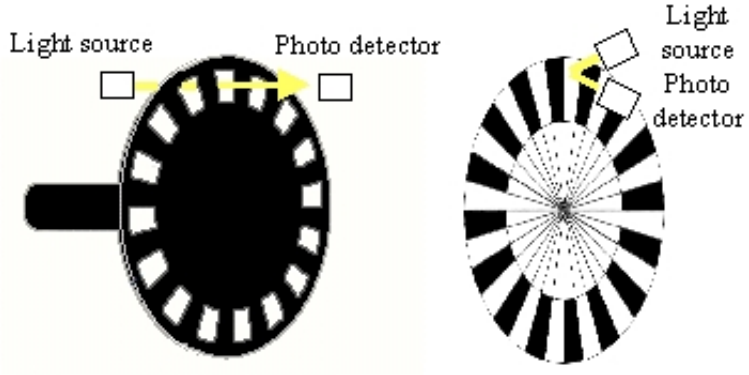
Figura 24 – Pinos para Conexões no *Shield*



Fonte: Próprio Autor, 2017 .

A ideia básica de um *encoder* consiste em um disco graduado com partes claras e escuras ou ainda com furos espaçados entre si. Um emissor e um sensor óptico são posicionados de forma que o sensor perceba um sinal luminoso quando luz emitida pelo emissor for refletida pela parte mais clara do disco ou ainda passe por algum dos furos, se for esse o caso. Quando a luz refletir sobre a parte escura do disco ou não encontrar um furo, o receptor óptico não perceberá a presença de luminosidade. Assim, cria-se uma onda quadrada que terá um período dependente da velocidade de giro do disco. Disto, consegue-se estimar a velocidade e/ou posição do equipamento medido, desde que se saiba a distância percorrida por pulso.

Figura 25 – *Encoder* Simples

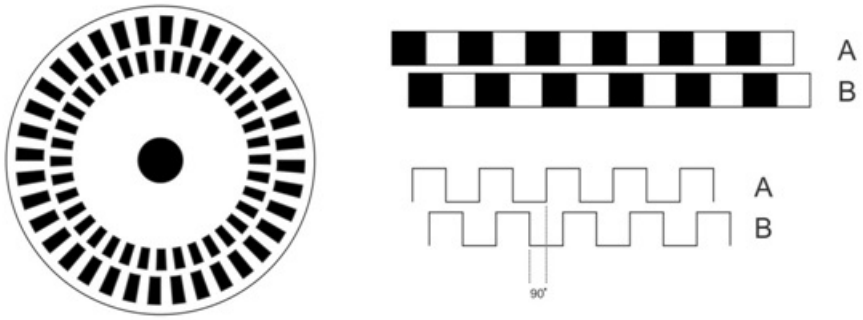


Fonte: Elektro Ekno, 2008 .

Existem outros tipos de *encoders* que atendem a necessidades mais específicas ao custo

de serem mais complexos. Como exemplo, pode-se citar o *encoder* de quadratura, o qual permite saber o sentido de rotação do disco. Isto é possível porque seu disco possui dois conjuntos concêntricos de ranhuras, uma mais externa e outra mais interna ao disco, defasadas de 90º. Neste caso o número de sensores dobra, pois são necessários dois emissores e dois receptores ópticos, um para cada conjunto de ranhuras.

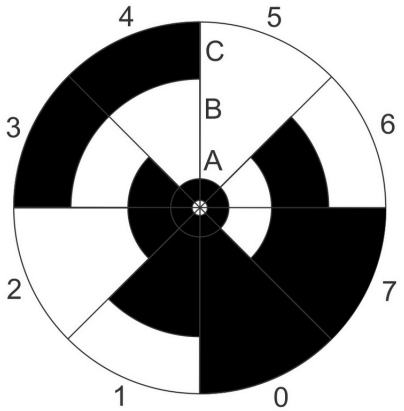
Figura 26 – *Encoder de Quadratura*



Fonte: Laboratório MOBILIS Computação Móvel, 2014.

Já os *encoders* absolutos permitem saber a posição do equipamento a qualquer momento. Isso é feito utilizando uma codificação binária e várias faixas de ranhuras concêntricas, cada uma representando um bit. Para provocar uma variação sutil de bits, de forma que apenas um mude por transição, comumente se utiliza a codificação *Gray*. Esse tipo de *encoder* é muito útil para quando cada posição do eixo tem um significado específico, mas apresenta uma sofisticação desnecessária, caso o que se busque seja um controle de velocidade, por isso esse tipo de sensor não se encaixa no presente projeto.

Figura 27 – *Encoder* Absoluto



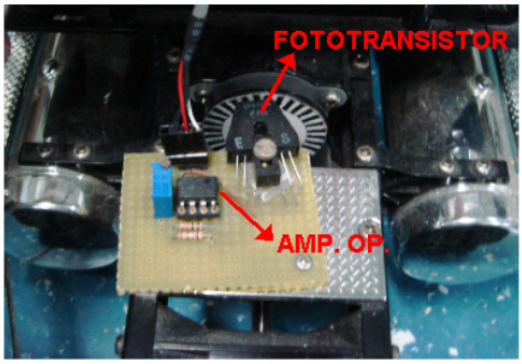
Posição	A	B	C
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

Fonte: Eletronicamente Falando, 2011.

Neste trabalho, reutilizou-se o *encoder* já instalado no elemento *truck*, o qual consiste de um disco incremental simples com 64 pulsos por volta. Ele foi mantido devido à dificuldade encontrada em posicionar outro *encoder* no eixo do motor, uma vez que este não fica exposto, mas sim dentro da caixa de redução, utilizada para converter a velocidade e torque do motor elétrico para valores apropriados ao *truck*.

O disco foi colado em uma pequena protuberância aparente do eixo da caixa de redução. O sensor emissor/receptor óptico foi posicionado em frente ao disco para detectar as mudanças de reflexão.

Figura 28 – *Encoder* Simples Existente no Protótipo



Fonte: Bertolani, 2011, p. 26.

Para estabelecer quanto o caminhão se movia por pulso foram necessários testes empíricos de voltas das rodas traseiras por pulsos. Como a velocidade de rotação do eixo do motor elétrico não é a mesma da velocidade de rotação das rodas, devido à caixa de redução, e a relação de transformação é desconhecida e variável, pois a caixa de redução possui marchas, mediu-se quantos pulsos eram dados para um giro completo das rodas traseiras. Foram feitas 20 medidas de pulsos por volta completa da roda e calculou-se a média dos valores encontrados. Como resultado, obteve-se 66 pulsos por volta. As rodas do *truck* possuem um diâmetro de $0,083[m]$, logo um perímetro de $0,261[m]$, portanto:

$$f_m = \frac{0,261}{66} = 0,00394 \left[\frac{m}{pulsos}\right] \tag{3.2}$$

Onde f_m representa o fator de metros deslocados por pulsos.

Para que essa relação não seja modificada, a marcha fica sempre travada em seu valor mais forte (maior torque). Como o que se deseja neste projeto é aplicar um controle fino sobre as manobras realizadas em ré, não é essencial que o caminhão atinja altas velocidades, por isso esta restrição pode ser tomada sem conseqüências relevantes.

3.6 Rádio e Receptor AM

Para enviar os sinais de comando ao *Arduino* foi utilizado um rádio AM de dois canais e freqüência de 27[MHz], o qual pode ser visto na figura 29. Junto com o rádio, há o receptor AM de nove pinos (2 dos canais, um de sinal, 3 Vcc's e 3 GND's), o qual pode ser visto na figura 30.

Figura 29 – Rádio AM de Dois Canais

Fonte: Próprio Autor, 2017.

Figura 30 – Receptor AM de Dois Canais

Fonte: Próprio Autor, 2017.

Os sinais produzidos pelos dois canais consistem em ondas PWM com um período de 96[ms], dos quais o tempo de sinal em alto dura entre 1[ms] a 2[ms], ou seja, o usuário consegue fazer os sinais dos canais variarem em torno de 1[ms] em alto.

Com esta variação, mediu-se o tempo em microssegundos de duração dos sinais em alto e, relacionando estes tempos a valores apropriados de saídas PWM para os motores, fez-se com que o usuário pudesse controlar a direção e aceleração do caminhão.

Para o canal responsável por controlar a aceleração do *truck*, aqui referido como ch1, encontrou-se um tempo mínimo de 1210[μs] e máximo de 1860[μs]. Já para o

canal responsável pelo controle da direção, ou ch2, encontrou-se um tempo mínimo de 1110[μs] e máximo de 2180[μs].

Empiricamente, observou-se que os valores PWM de máximo e mínimo seguros para o motor elétrico foram de 155 a 200. Lembrando que a saída PWM do *Arduino* possui 8 bits, logo seu máximo valor seria de 255. Este intervalo foi linearmente relacionado com o intervalo de variação do canal destinado à aceleração. Também empiricamente, observou-se uma variação segura para a saída PWM da direção. Este intervalo foi de 70 a 230, bem maior do que o encontrado para o motor elétrico, o qual também foi linearmente relacionado ao canal destinado à direção.

Um problema encontrado foi que, por vezes o sinal recebido se perdia ou sofria alguma interferência, de forma a sair do intervalo esperado e, assim gerando uma saída indesejada. Para solucionar isso, os valores possíveis para os canais foram saturados de forma que quaisquer valores fora do *range* esperado limitar-se-iam aos limites estipulados.

Como não se deseja que o motor elétrico seja muito forçado e para que o meio do intervalo represente a situação em que o motor está desativado, os limites para o ch1 foram de 1400[μs] a 1820[μs]. Já para o ch2 os limites foram de 1110[μs] a 2220[μs]. Considerando que a função que obtém esses tempos tem uma resolução de 10[μs], tem-se que:

$$\text{Resolução} = \frac{62}{\frac{2220}{10} - \frac{1110}{10}} = 0.559^{\circ} \tag{3.3}$$

Onde, utiliza-se 62º no numerador da equação 3.3 devido ao fato da direção variar entre ±31º.

No momento em que o ch2 foi usado para passar a referência do ângulo da articulação e não mais para a direção, a resolução foi:

$$\text{Resolução} = \frac{180}{\frac{2220}{10} - \frac{1110}{10}} = 1.622^{\circ} \tag{3.4}$$

Utiliza-se 180º como o intervalo de interesse, primeiramente porque não era desejável que as articulações ultrapassassem ±90º, havendo o risco de ângulos superiores a estes comprometerem a integridade física do protótipo. Em segundo lugar, não era desejável que as articulações alcançassem ângulos muito acentuados, pois, a partir de certo limite, torna-se impossível para o sistema, mesmo havendo somente um *trailer*, diminuir o ângulo apenas movendo o RMMA para trás. Empiricamente, constatou-se que ângulos próximos de ±80º seriam irreversíveis para o caso de um elemento *truck* ativo e um elemento *trailer* passivo, assim o intervalo de interesse foi tomado entre -90º a +90º por dar certa margem aos limites observados.

4 ELABORAÇÃO DO MODELO E CONTROLADOR *FUZZY*

4.1 Considerações e Simplificações

Ao se observar os trabalhos de (BERTOLANI, 2011), (BERTOLANI, 2013), (MIRANDA, 2011) e (PANDOLFI, 2012) vê-se que eles utilizaram a mesma estrutura física, o que torna a comparação entre eles direta. Neste trabalho, entretanto, modificou-se tal estrutura, substituindo o antigo sistema embarcado pelo *board* de desenvolvimento da família *Arduino*, e alterando-se as rotinas de controle e os sinais de comando enviados à placa. Logo, muitas comparações e análises precisam ser refeitas sob este novo cenário. A possibilidade de se adquirir dados de forma muito mais prática do que na estrutura antiga abre várias possibilidades que, ou não existiam anteriormente, ou eram extremamente dispendiosas e cansativas. Como este trabalho apresentou um foco considerável na reestruturação física do protótipo e em permitir que este fosse reutilizado em outros projetos, as partes de modelagem e controle serão focadas na metodologia *fuzzy*. Outras abordagens são possíveis, mas estas merecem trabalhos que se foquem exclusivamente nisto, o que não é o caso deste projeto.

Partindo-se do entendimento de que este é um novo sistema, decidiu-se por abordar o caso mais simples, composto de uma configuração com apenas dois graus de liberdade, sendo eles o elemento *truck* e o elemento *trailer*. A evolução natural deste projeto é a introdução do segundo elemento *trailer*, mas como uma análise inicial, observar como o novo sistema responde ao problema simplificado ajuda a compreender quais melhoras e limitações passaram a existir após as mudanças, sendo, portanto importante que se dê esse passo inicial.

4.2 Levantamento do Modelo *Fuzzy*

4.2.1 Obtenção das regras para Pequenas Variações nos Ângulos de Giro

Após a reestruturação física descrita na seção anterior, desejava-se a elaboração de um modelo que pudesse descrever o sistema real de forma satisfatória. Para isso foram realizados experimentos em que a referência de direção e velocidade eram passadas diretamente às rodas dianteiras e as rodas traseiras, respectivamente, através do *Joystick* de dois canais. Basicamente, nesta etapa, fez-se uso do rádio AM para controlar a velocidade e a direção das rodas dianteiras de forma direta pelo usuário, sem haver um sistema de controle

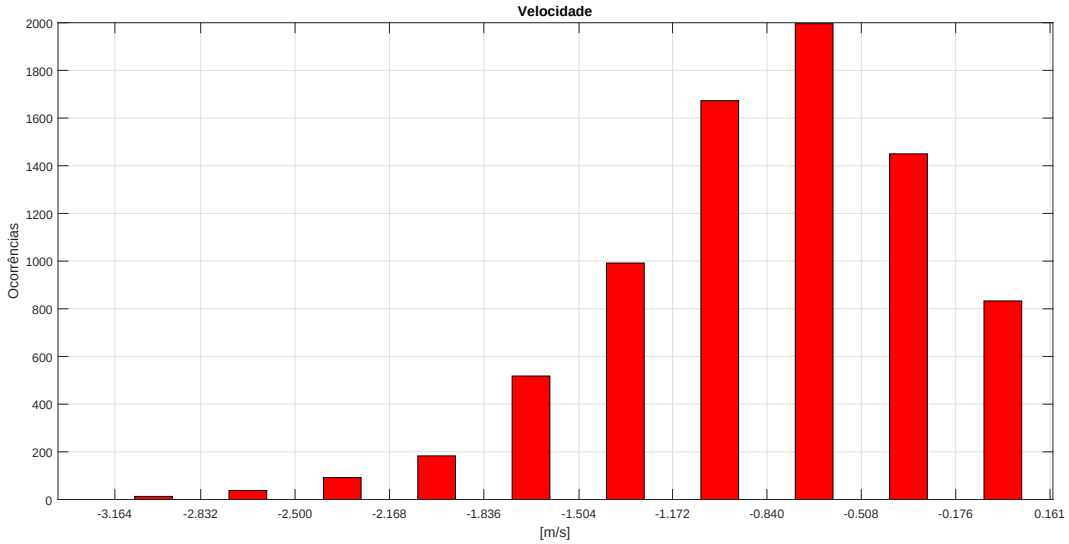
realimentado. Assim, operou-se manobras de ré em que o usuário realimentava visualmente a direção para evitar as situações de *jackknife*. Enquanto estas manobras eram feitas, coletou-se as informações de velocidade, ângulo das rodas dianteiras e ângulo do elemento *trailer* com o carro principal. Em alguns momentos foi necessário mover o RMMA para frente, quando, devido a um erro do operador, o protótipo entrava em *jackknife*, entretanto estes dados foram desconsiderados, pois apenas o comportamento em ré era relevante.

A facilidade de exercer o controle da aceleração e da direção por um *Joystick* possibilitou que fossem coletados dados em grandes quantidades e em posições mais significativas. Isso foi de extrema importância, uma vez que, para representar de forma satisfatória um sistema, o conjunto de regras *fuzzy* precisa abranger todas as possibilidades de configuração com leis consistentes com a realidade. Ao se tornar possível prolongar os experimentos de coleta de dados e controlar-se diretamente a direção e velocidade, pôde-se colocar o protótipo em diversos tipos de configuração repetidas vezes. Após obter-se estes dados, decidiu-se por analisar as regiões em que os dados coletados mais apareciam, dentro de seus universos de discurso, ou seja, dentro dos valores que estes poderiam assumir. Isto foi feito para ter-se uma ideia do quão bem os dados coletados representam as regiões de interesse neste trabalho.

Os dados considerados relevantes nesta etapa foram: velocidade v_n; ângulo das rodas dianteiras, aqui chamado de "direção" γ_n; ângulo entre o elemento *trailer* e o caminhão, ou simplesmente "ângulo" θ_n e a variação entre o ângulo atual e o próximo obtido na medida seguinte, ou simplesmente "variação" $\Delta\theta_{n+1}$ ($\Delta\theta_{n+1} = \theta_{n+1}$ - θ_n). Através das medidas feitas destas variáveis, dividiram-se as amostras dentro de um intervalo, tendo como valor máximo a maior medida feita e como valor mínimo a menor. Estes intervalos ainda foram subdivididos em dez partes e verificou-se quantas medidas existiam dentro de cada subintervalo. As figuras de 31 a 34 mostram os valores encontrados.

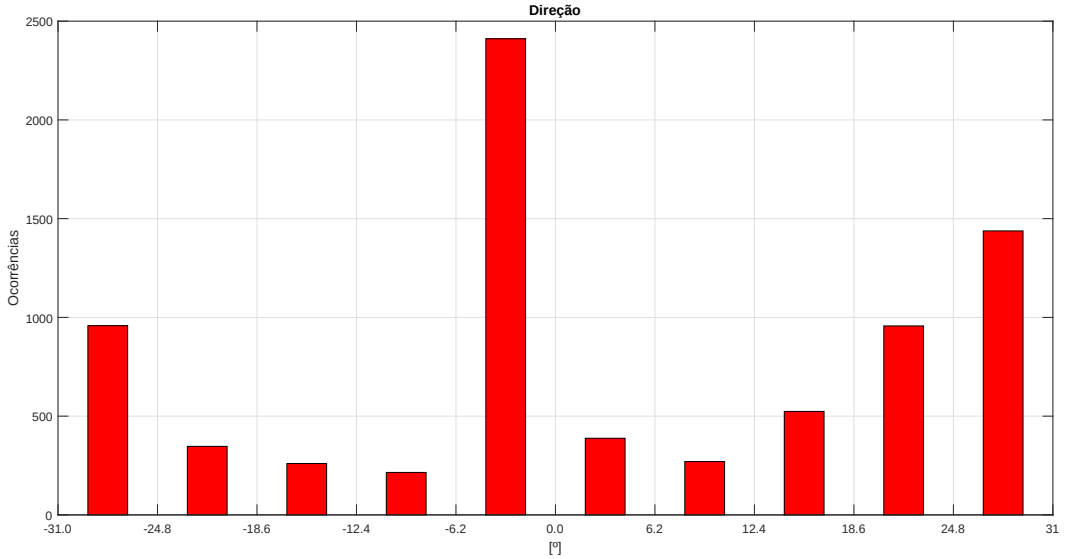
De fato, a velocidade influencia na dinâmica de variação do ângulo de formação do protótipo, entretanto consegue-se exercer um controle eficiente sobre ele mesmo considerando-a constante. Esta simplificação não representa inteiramente a realidade, mas demonstra-se ser uma aproximação válida. A figura 31 ainda assim é interessante para que se tenha ideia da faixa de velocidade na qual os testes foram feitos. A figura 32 mostra que a maioria das posições observadas da direção concentraram-se próximas de 0º (rodas dianteiras alinhadas com o elemento *truck*) ou nos extremos máximos que a direção poderia chegar, ou seja, de -31º a +31º, representando uma abertura de 31º a direita (sentido horário) e de 31º a esquerda (sentido anti-horário) respectivamente, em relação ao *truck*. Deste modo, não há sentido em desconsiderar algum desses intervalos e a variável lingüística que representa esta variável leva em conta todo esse espaço de excursão.

Figura 31 – *Freqüência da Ocorrência de Amostras para as Medidas de Velocidade*



Fonte: Próprio Autor, 2017.

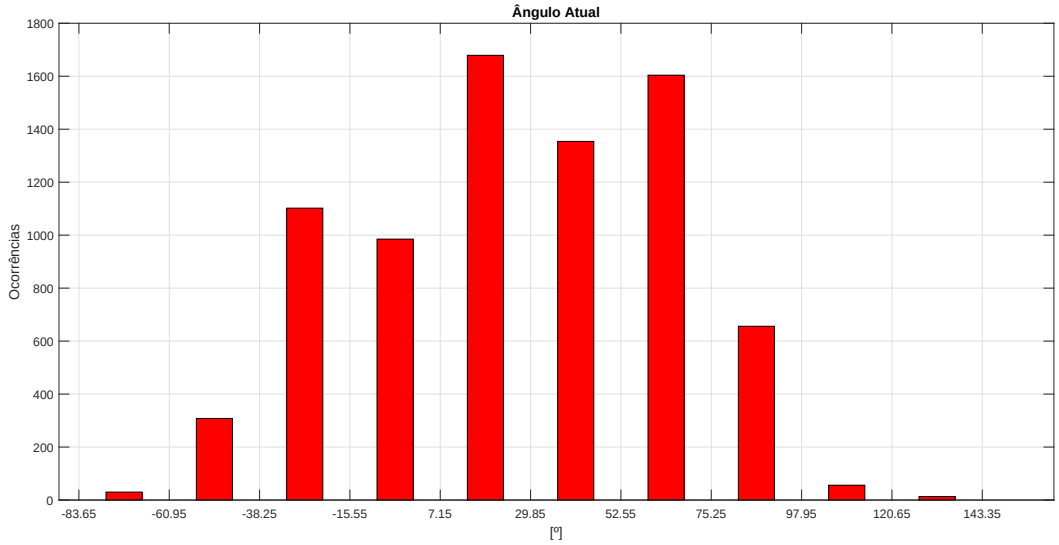
Figura 32 – *Freqüência da Ocorrência de Amostras para as Medidas de Direção*



Fonte: Próprio Autor, 2017.

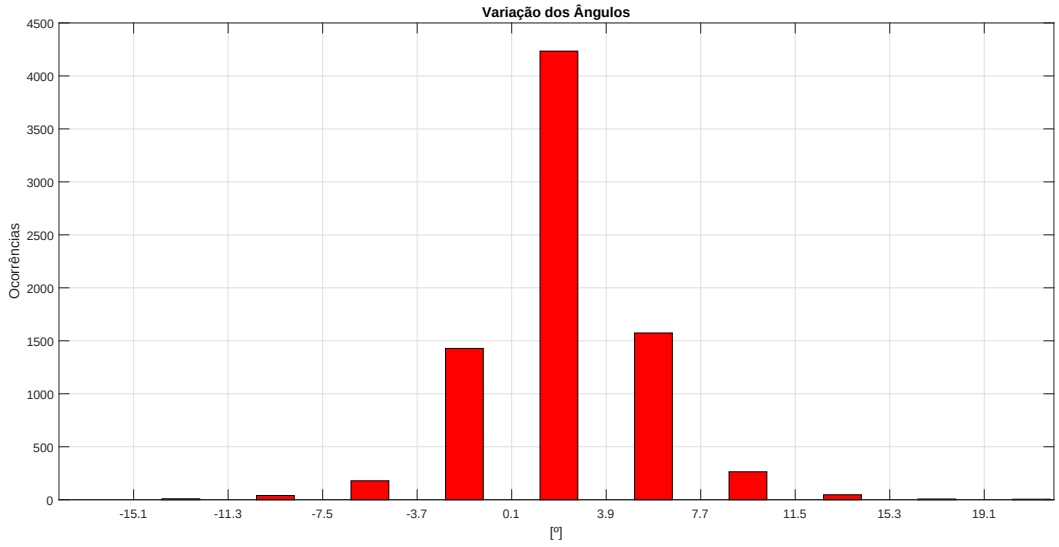
Já a figura 33 mostra que a maioria das medidas concentram-se em ângulos de formação para a esquerda, ou seja, o *trailer* estava virado à esquerda em relação ao elemento *truck*. Ainda assim, há uma quantidade relevante de dados entre -60º e -38º aproximadamente, e como não deseja-se realizar manobras com ângulos muito acentuados, foi considerado o intervalo de -50º a 50º para a variação do ângulo de formação. Este intervalo concentra a maioria das medidas feitas, logo representa bem a dinâmica de movimento em ré do RMMA, e não poria o conjunto próximo das regiões que representam risco de *Jackknife*.

Figura 33 – *Freqüência da Ocorrência de Amostras para as Medidas do Ângulo de Formação*



Fonte: Próprio Autor, 2017.

Figura 34 – *Freqüência da Ocorrência de Amostras para as Medidas de Variação do Ângulo*



Fonte: Próprio Autor, 2017.

Por fim, para a variação do ângulo de formação, observa-se na figura 34 que a maioria das medidas de $\Delta\theta_{n+1}$ concentra-se entre 0º e 4º aproximadamente, e dificilmente superam 8º em todos os sentidos. Sendo assim, para concentrar as regras em mudanças mais finas, limitou-se o intervalo desta variável entre -6º e 6º.

A escolha destes intervalos é relevante, pois tanto o modelo quanto o controlador *fuzzy*

utilizam variáveis lingüísticas que precisam ter suas fronteiras definidas. Assim buscou-se limitar o universo de discurso das variáveis dentro de regiões onde haviam dados que as representassem satisfatoriamente. Após estas definições passou-se para o próximo passo, o qual consiste exatamente em definir as variáveis lingüísticas das entradas, as regras entre elas e as conseqüências ou saídas destas regras.

4.2.2 Obtenção de um Modelo *Fuzzy* para Validação do Método

Uma vez que tenha-se determinado qual conjunto de dados será utilizado para gerar-se o modelo *fuzzy* e quais serão os limites das variáveis lingüísticas envolvidas, deve-se decidir como este modelo será feito. O primeiro aspecto a ser considerado é qual técnica de modelagem *fuzzy* será utilizada. Há duas possibilidades mais óbvias, já bem estabelecidas na literatura e discutidas anteriormente: Wang-Mendel (WANG; MENDEL, 1992) e Takagi-Sugeno (TAKAGI; SUGENO, 1985). Muito embora haja algumas bibliotecas para *Arduino* que implementam a metodologia de *deffuzzyficação* Wang-Mendel, isto ocuparia uma parte considerável da memória do controlador, além de exigir um tempo maior de processamento, o que acabava por afetar algumas leituras de variáveis. Já a metodologia de Takagi-Sugeno apresenta vantagens de criar saídas de uma forma mais objetiva, utilizando-se de aproximações por mínimos quadrados para estimar as melhores saídas, as quais já são valores *crisp*, e não precisam ser *defuzzyficadas*. Por isso, decidiu-se por utilizar esta metodologia em detrimento da de Wang-Mendel, sem recorrer a bibliotecas já existentes, a fim de poupar memória do microcontrolador.

Como ponto de partida, utilizou-se da função *genfis2* do *Software* $Matlab^{©}$. Esta função utiliza os conjuntos de dados de entrada e saída para gerar regras cujas saídas sejam equações lineares, aproximadas através do método de mínimos quadrados, que representam da melhor maneira possível as relações de causa-efeito passada através deles. A função é utilizada da seguinte forma:

$$fismat = genfis2(Xin, Xout, radii, xBounds, options, user\_centers)$$

Onde:

- Xin**:** Representa a matriz que contém os dados das entradas, sendo cada coluna uma entrada. No caso específico deste trabalho $Xin_{7788\times2}$.
- $Xout$**:** Representa a matriz que contém os dados das saídas, sendo cada coluna uma saída. No caso específico deste trabalho $Xin_{7788\times1}$.

- *radii*: Esta função assume que os valores das variáveis lingüísticas serão representados por uma função normal dada na equação 4.1. A variável radii é um vetor que representa o valor de σ para cada variável lingüística. Como há duas entradas e uma saída a dimensão de radii foi 1×3. Desejava-se que os valores das variáveis lingüística tivessem o *crosspoint* em 0,5, logo todas as entradas de radii foram 0,3, o que causa o efeito desejado.

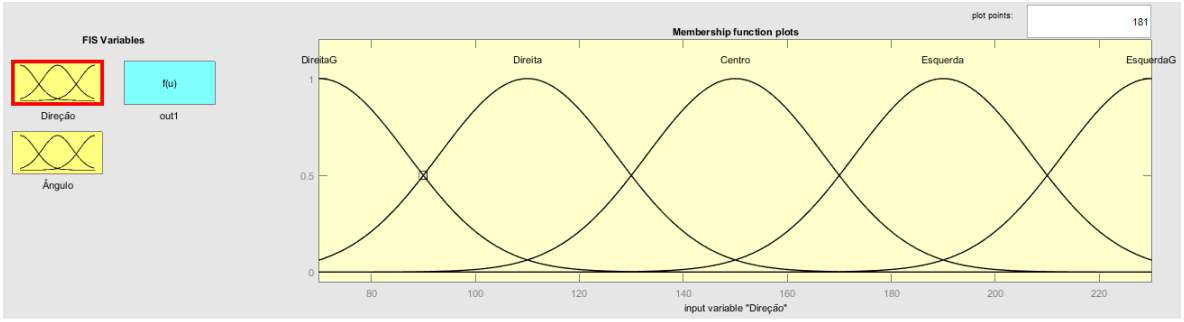
- *xBounds*: Essa matriz determina os limites máximo e mínimo de cada variável lingüística. A primeira linha representa os valores mínimos e a segunda os valores máximos. Estes limites são os discutidos anteriormente para γ, θ_n e θ_{n+1}. *xBounds*, portanto, para este caso, é uma matriz 2×3.

- *options*: Esta entrada modifica parâmetros específicos do algoritmo base da função e não foi utilizada. Por isso, ela recebeu uma matriz vazia.

- *user_centers*: Esta matriz determina as subdivisões de cada variável lingüística. Aqui, arbitrariamente decidiu-se por dividir cada variável em 5 valores igualmente espaçados. A forma da divisão das variáveis em seus respectivos espaços de discurso pode ser vista nas figuras 35 e 36.

$$\mathrm{f(x,\sigma,c)} = \mathrm{e}^{-\frac{(x-c)^2}{2\sigma^2}} \tag{4.1}$$

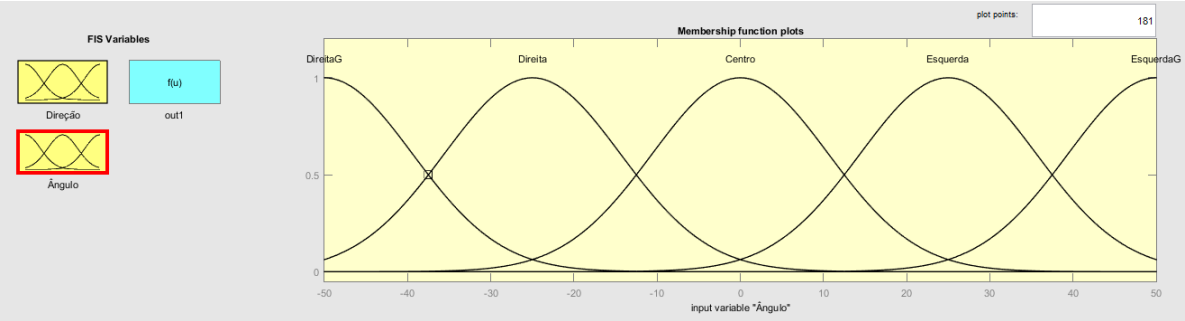
Figura 35 – *Divisão da Variável Lingüística Direção*



Fonte: Próprio Autor, 2017.

Deve-se observar que os limites da variável Direção vão de 70 a 230. Isso ocorre, pois esses são os valores em PWM que acionam o servo motor que controla a direção e equivalem

Figura 36 – *Divisão da Variável Lingüística Ângulo*



Fonte: Próprio Autor, 2017.

-31° e 31°, respectivamente. Tais valores podem ser convertidos para graus através da equação 4.2.

$$\text{Direção} = 0,3875\text{PWM} - 58,125 \tag{4.2}$$

A função *genfis2* gerou cinco regras, cujas saídas são dadas nas equações lineares de 4.3 a 4.7.

$$\gamma_n(DireitaG) \text{ e } \theta_n(DireitaG) \Rightarrow \theta_{n+1} = -0,006539\alpha_n + 1,011\theta_n + 1,689 \tag{4.3}$$

$$\gamma_n(Direita) \text{ e } \theta_n(Direita) \Rightarrow \theta_{n+1} = -0,03387\alpha_n + 1,019\theta_n + 4,563 \tag{4.4}$$

$$\gamma_n(Centro) \text{ e } \theta_n(Centro) \Rightarrow \theta_{n+1} = -0,07423\alpha_n + 1,022\theta_n + 11,010 \tag{4.5}$$

$$\gamma_n(Esquerda) \text{ e } \theta_n(Esquerda) \Rightarrow \theta_{n+1} = -0,02072\alpha_n + 0,9961\theta_n + 3,786 \tag{4.6}$$

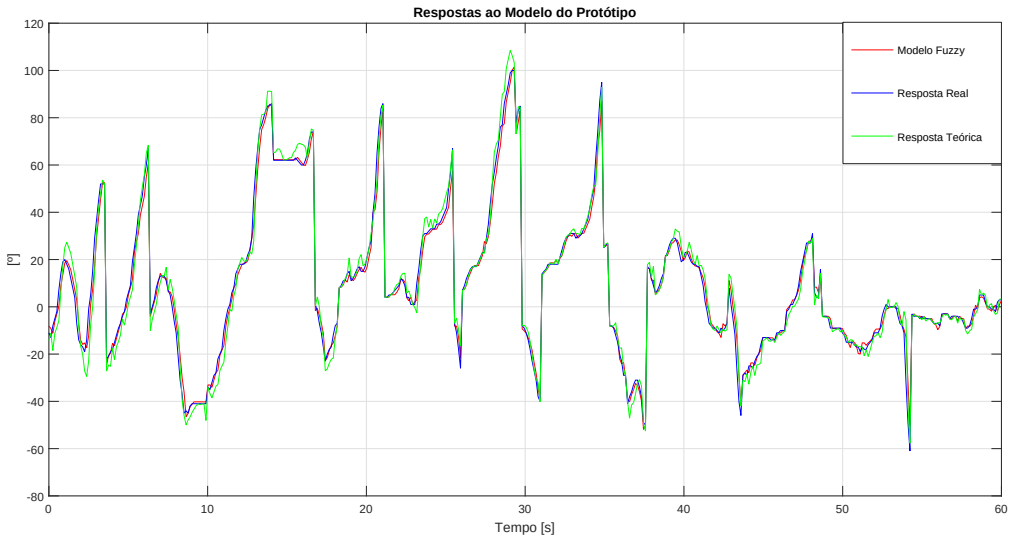
$$\gamma_n(EsquerdaG) \text{ e } \theta_n(EsquerdaG) \Rightarrow \theta_{n+1} = 0,01509\alpha_n + 1,027\theta_n - 4,608 \tag{4.7}$$

A função *genfis2* parece condicionar o número de regras criadas à quantidade de valores das variáveis lingüísticas, independentemente da quantidade e variedade dos dados. Entretanto,

uma boa quantidade de dados é necessária para que as saídas criadas consigam representar bem o comportamento do modelo em todo o universo de discurso através das equações de saída obtidas pelo método dos mínimos quadrados.

Para se validar o modelo *fuzzy* obtido através destes dados, utilizou-se de um outro conjunto de dados obtidos através dos testes feitos com o protótipo e, usando os novos valores de γ_n e θ_n como entradas para o modelo *fuzzy*, comparou-se as saídas com os valores reais medidos de θ_{n+1} ($\theta_{n+1} = \theta_n + \Delta\theta_{n+1}$). Além disso, utilizou-se da equação 1.3 para comparar os valores medidos e os obtidos através do modelo com o que seria esperado pelo modelo teórico baseado nas equações de giro. A figura 37 mostra os resultados destas comparações.

Figura 37 – Comparação entre as Respostas Obtidas Através do Modelo *Fuzzy* e da Equação Analítica com os Valores Medidos



Fonte: Próprio Autor, 2017.

Há algumas variações bruscas nos valores dos ângulos. Isso acontece porque nestes momentos o RMMA havia começado a se mover para frente e ambos modelos, analítico e *fuzzy*, apenas são válidos para movimentos em ré, por isso esses momentos foram desconsiderados até o protótipo voltar a se mover para trás. Apesar disso o modelo *fuzzy* conseguiu acompanhar melhor as respostas reais obtidas do que o modelo analítico, havendo menos desvios em transições mais bruscas.

A fim de avaliar-se o quão próximo as respostas obtidas pelo modelo *fuzzy* e pelo modelo analítico ficaram dos valores medidos, utilizou-se da função de *fitness* descrita na equação

4.8.

$$fitness = \frac{1}{\sum_{i=1}^{N} \|ref_i - y_i\|} \tag{4.8}$$

Onde ref_i são os valores da referência tomada como base da comparação, neste caso os dados medidos, e y_i são os valores que serão comparados com a referência, neste caso as saídas dos modelos. Esta é uma comparação absoluta, no sentido de que a resposta dependerá numericamente da ordem de grandeza dos valores usados. Como é feito o inverso da soma dos erros, quanto maior for o valor do *fitness*, maior será a proximidade dos dados, pois menor é o erro entre eles.

Para o modelo *fuzzy* o valor do *fitness* foi de $9,0625 \times 10^{-4}$, enquanto que para o modelo analítico o *fitness* foi de $6,2161 \times 10^{-4}$. Isso comprova que o modelo *fuzzy* encontrado representa melhor o sistema do que o modelo analítico, o que poderia ser sugerido visualmente pela figura 37.

A fim de se obter uma comparação proporcional desses dados, utilizou-se outra função de *fitness*, porém que relativiza os dados em função da média da referência, dando um valor em porcentagem. A equação 4.9 descreve essa função.

$$fitness = 100 \times \left(1 - \frac{\sum_{i=1}^{N} \|ref_i - y_i\|}{\sum_{i=1}^{N} \|ref_i - ref_i^*\|}\right) \tag{4.9}$$

Onde ref_i e y_i representam o mesmo que na equação 4.8 e ref_i^* representa a média dos valores de ref_i. Por essa função de *fitness* o modelo *fuzzy* apresentou uma semelhança de $90,42\%$ com os dados medidos, já o modelo analítico apresentou uma semelhança de $86,04\%$. Novamente, isso comprova que houve uma melhora na representação do sistema pelo modelo *fuzzy* quando comparado ao modelo analítico.

Este resultado superior do modelo *fuzzy* pode ser atribuído justamente ao fato dele conseguir incorporar não-linearidades e atrasos existentes na planta, o que o modelo analítico não inclui sozinho.

4.3 Elaboração do Controlador *Fuzzy*

Para elaborar agora o controlador *fuzzy*, foram utilizados os mesmos dados usados para a construção do modelo, por estes já se mostrarem bons o suficiente para representar os principais aspectos do sistema. O que alterou-se nesta abordagem são quais dados foram considerados entradas, quais foram considerados saídas e a forma destes serem utilizados.

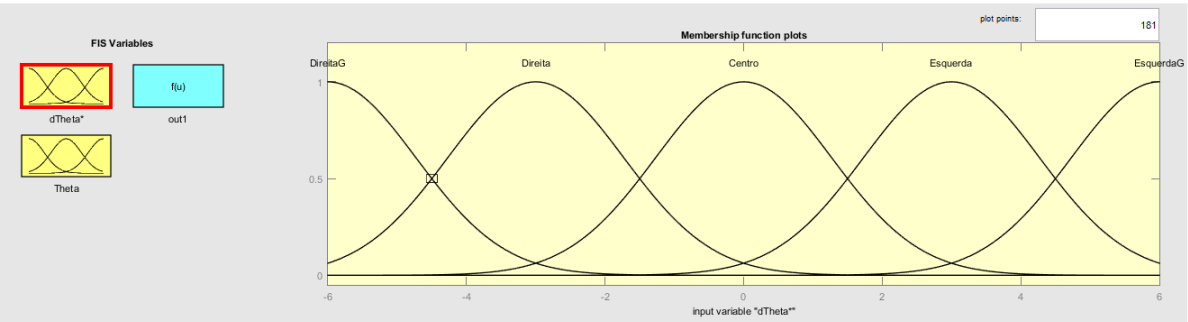
O modelo foi feito assumindo-se que, ao observar um instante específico n de dados, pode-se supor quais serão os valores de θ_{n+1}. Logo, pode-se inverter esta ideia e, assumir que, se é desejado que certa conseqüência θ_{n+1} aconteça quando o RMMA se encontra com um ângulo θ_n, o valor de α_n que causará tal efeito é conhecido, desde que se tenha dados suficientes que cubram a maior parte das possibilidades de causa e efeito. Em outras palavras, os valores de θ_n e θ_{n+1} foram considerados as entradas, pois supõe-se que θ_n tornou-se θ_{n+1} por causa de α_n. Logo, sabe-se qual valor a direção deve assumir para causar uma mudança desejada no ângulo da formação.

Portanto, o novo bloco de controle contará com duas entradas. Uma é a referência, ou ângulo em que deseja-se chegar. A outra será a posição atual, e assim o controlador deverá indicar qual o valor da direção que levará a configuração à posição desejada.

A dificuldade encontrada nesta abordagem está no fato de que, como olha-se medidas espaçadas apenas por um intervalo de amostras, não observa-se grandes diferenças em termos de valor absolutos entre θ_n e θ^*_{n+1}. Estes, quando observados em uma janela de duas amostras consecutivas, apresentam valores próximos, pois a variação do ângulo de posição é gradual e não instantânea. por isso, ao invés de observar-se o valor absoluto de θ^*_{n+1}, considerar-se-á a variação ou diferença entre a posição de referência e a atual ($\Delta\theta^*_{n+1} = \theta^*_{n+1} - \theta_n$). Assim, buscou-se concentrar as regras geradas próximas da referência, de forma a tornar o sistema mais estável próximo deste ponto e generalizar o comportamento nas situações em que a referência estava distante. A figura 9 ilustra o bloco de entradas e saída do controlador.

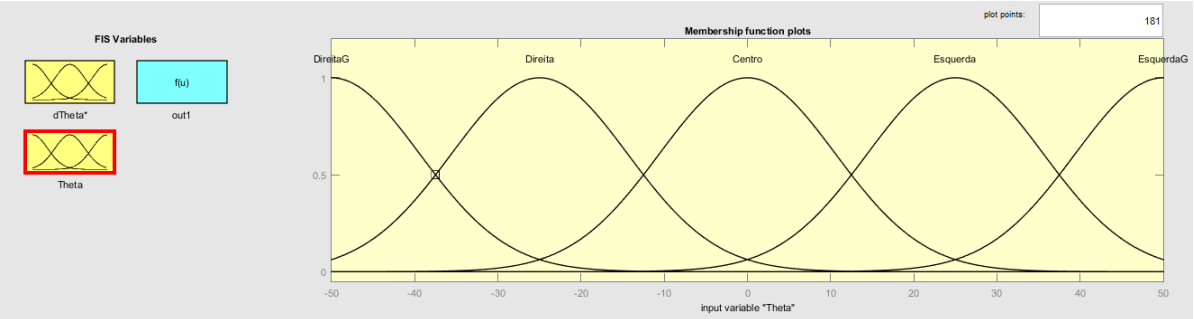
As variáveis lingüísticas de entrada e as equações de saída para o controlador *fuzzy* são mostradas nas figuras 38 e 39 e nas equações de 4.10 a 4.14, respectivamente.

Figura 38 – Diferença entre Referência e Ângulo Atual



Fonte: Próprio Autor, 2017.

Figura 39 – Ângulo Atual



Fonte: Próprio Autor, 2017.

$$\Delta\theta_{n+1}(DireitaG) \text{ e } \theta_n(DireitaG) \Rightarrow \gamma_{n+1} = -3,423\Delta\theta_{n+1} - 0,1602\theta_n + 107,7 \quad (4.10)$$

$$\Delta\theta_{n+1}(Direita) \text{ e } \theta_n(Direita) \Rightarrow \gamma_{n+1} = -5,663\Delta\theta_{n+1} + 0,9013\theta_n + 140,2 \quad (4.11)$$

$$\Delta\theta_{n+1}(Centro) \text{ e } \theta_n(Centro) \Rightarrow \gamma_{n+1} = -0,649\Delta\theta_{n+1} + 0,9065\theta_n + 143,6 \quad (4.12)$$

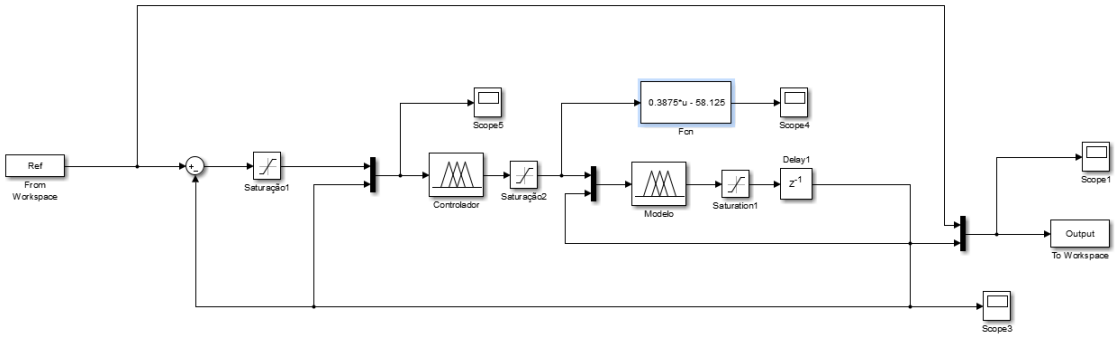
$$\Delta\theta_{n+1}(Esquerda) \text{ e } \theta_n(Esquerda) \Rightarrow \gamma_{n+1} = -3,209\Delta\theta_{n+1} + 0,7381\theta_n + 160,4 \quad (4.13)$$

$$\Delta\theta_{n+1}(EsquerdaG) \text{ e } \theta_n(EsquerdaG) \Rightarrow \gamma_{n+1} = -3,528\Delta\theta_{n+1} + 0,2951\theta_n + 170,4 \quad (4.14)$$

Uma vez que tenha-se gerado o controlador, este foi testado, via simulação, utilizando-se do modelo *fuzzy* gerado anteriormente como a representação da planta do sistema. A malha de controle é mostrada na figura 40.

A primeira saturação, situada após a subtração da referência com a posição atual, varia entre -6º e 6º, por esses serem os limites estabelecidos anteriormente. A segunda saturação, colocada após o controlador, representa os limites físicos da direção, podendo excursionar

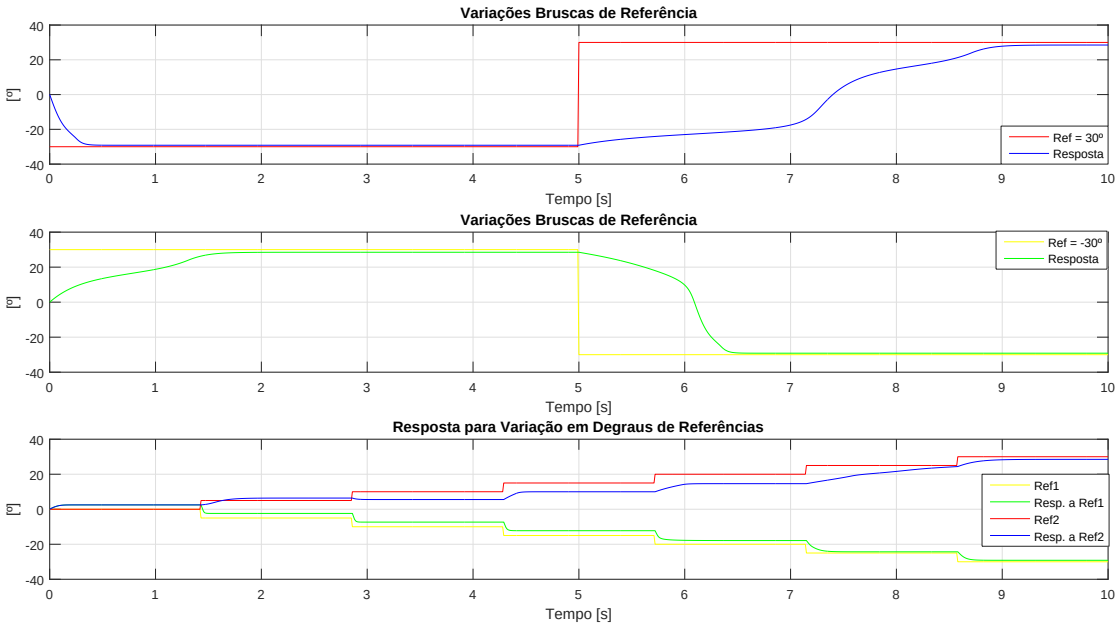
Figura 40 – Malha de Controle Proposta



Fonte: Próprio Autor, 2017.

entre -31° e 31°. Já a terceira saturação, posicionada depois do modelo, representa os limites físicos de variação do ângulo de formação indo de -90° a 90°, os quais já representariam uma situação de *jackknife*, mas, com apenas um *trailer*, estão próximos dos limites em que o sistema de controle conseguiu desfazer o arco da formação, assim assumiu-se esses valores como limitantes. Os resultados encontrados para estas simulações são mostrados na figura 41.

Figura 41 – Simulações Utilizando o Controlador e o Modelo *Fuzzy*



Fonte: Próprio Autor, 2017.

Pela figura 41, vê-se que o controlador proposto consegue fazer o modelo responder de tal forma a chegar nas referências determinadas. No primeiro caso observou-se como seria

a respostas para uma variação brusca de referência. Inicialmente o modelo tinha como saída inicial 0º e a referência estava em -30º. Após 5 segundos a referência foi invertida para +30º. No segundo caso foi feito o oposto, com a referência inicialmente igual a +30º e após 5 segundos sendo invertida para -30º, com o valor inicial da saída do modelo igual a 0º. No terceiro caso, variou-se a referência em degraus menores de 5º, partindo de 0º até 30º em ambos os sentidos.

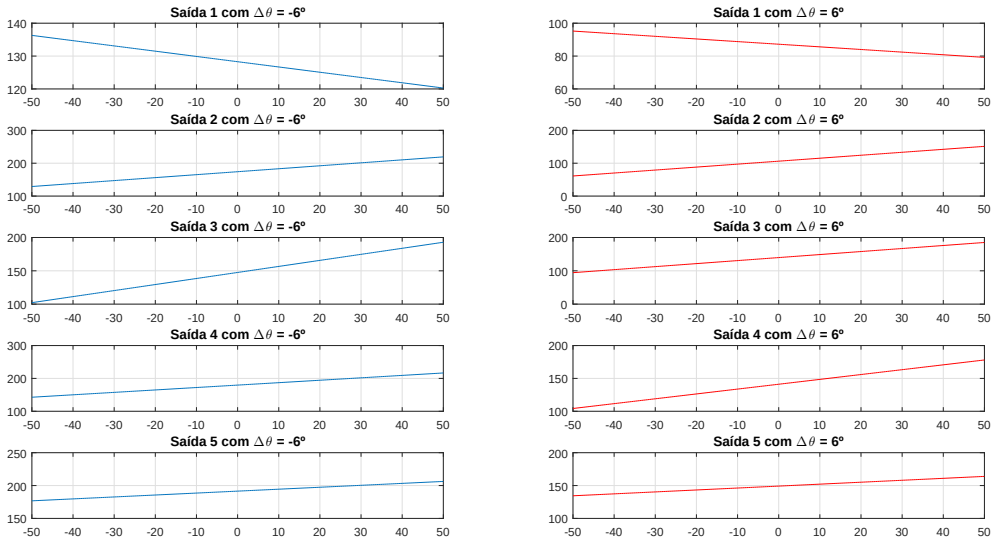
O sistema não respondeu da mesma forma para as referências de +30º e -30º, chegando mais rapidamente às referências negativas (*trailer* virado para a direita em relação ao *truck*). Especificamente no caso em que se requereu uma mudança brusca de -30º para 30º, observou-se uma resposta ainda mais lenta do que quando foi requerido o inverso, de 30º para -30º. Isto parece demonstrar que, dentre os dados utilizados para gerar o controlador, muito embora eles concentrem-se mais nos valores positivos do ângulo θ, a representação das transições de posição está melhor descrita nos dados com valores negativos de θ.

Esta observação parece se confirmar olhando para o terceiro caso, pois novamente o comportamento para a direita (sinal negativo), demonstrou-se melhor do que para a esquerda (sinal positivo). No caso das referências positivas houve uma região onde o modelo demorou mais para responder à mudança de referência, causando um atraso na resposta e um maior erro estacionário. A resposta para referências negativas, no obstante, apresentou um erro estacionário menor do que no outro caso, além de responder mais rapidamente às mudanças de referência.

Estas respostas foram obtidas simplesmente através dos dados coletados, não havendo interferência de especialistas no processo. Entretanto, uma das vantagens da técnica de controle *fuzzy* está justamente em poder combinar ambas formas de configurar o controlador. Além do mais, pelas respostas simuladas na figura 41, vê-se que, embora o sistema de controle cumpra sua função básica de levar o sistema para a referência, ele o faz com relativa demora. Logo, há espaço para melhorar a resposta incluindo o conhecimento sobre o comportamento do protótipo.

Para tanto, reutilizou-se as equações de 4.10 a 4.14 como base para se determinar as novas regras do controlador. Essas equações descrevem um plano, considerando-se todas as variações possíveis para $\Delta\theta^*_{n+1}$ e θ_n. Todavia, ao se fixar $\Delta\theta^*_{n+1} = -6^o$ quando deseja-se uma variação grande para a direita e $\Delta\theta^*_{n+1} = 6^o$ quando deseja-se uma variação grande para a esquerda, os planos tornam-se retas e, observando os valores de θ_n, consegue-se determinar quais equações darão as saídas necessárias. A figura 42 mostra as retas obtidas desta forma as quais foram utilizadas de base para se determinar as novas regras do controlador *fuzzy*.

Figura 42 – Retas Obtidas das Saídas Fazendo-se $\Delta\theta = -6^o$ e $\Delta\theta = 6^o$



Fonte: Próprio Autor, 2017.

Através destas retas avaliou-se quais valores de θ_n dariam as saídas adequadas em PWM nos casos em que $\Delta\theta^*_{n+1} = 6^o$ e $\Delta\theta^*_{n+1} = -6^o$. Por exemplo, caso $\Delta\theta^*_{n+1} = 6^o$, ou seja, $\Delta\theta^*_{n+1} = EsquerdaG$ e $\theta_n = DireitaG$, tem-se a situação em que a formação está entre -50º e -37,5º (caso $\theta_n < -50^o$ será considerado $\theta_n = -50^o$) e deseja-se que ela aumente seu ângulo, em pelo menos +6º. Só que nesta posição o protótipo tende a fazer θ_n diminuir, aproximando-o do *jackknife*. Para evitar que isso ocorra, a direção deve ser posta o máximo possível para a direita ($\gamma = -31^o$), assim escolhe-se a reta da figura 42 em que $\Delta\theta^*_{n+1} = -6^o$, $-50^o < \theta_n < -37,5^o$ e o valor em PWM é o menor possível, logo reta escolhida neste caso foi "Saída 3 com $\Delta\theta^*_{n+1} = -6^o$".

O mesmo raciocínio foi feito para os outros casos, levando-se em conta que em posições nas quais o ângulo de formação θ_n não era tão grande em módulo a direção γ não precisava estar em um de seus valores máximo (também em módulo) $\pm 31^o$. Portanto, tentou-se graduar as transições de forma a fazer o valor de $\|\gamma\|$ diminuir conforme $\|\theta_n\|$ diminuía e $\Delta\theta^*_{n+1} = \pm 6^o$ se mantinham constantes.

As equações de 4.15 a 4.22 foram feitas por este processo e incorporadas ao controlador *fuzzy*.

$$\Delta\theta_{n+1}(EsquerdaG) \text{ e } \theta_n(DireitaG) \Rightarrow \alpha_{n+1} = -5,663\Delta\theta_{n+1} + 0,9013\theta_n + 140,2 \quad (4.15)$$

$$\Delta\theta_{n+1}(EsquerdaG) \text{ e } \theta_n(Direita) \Rightarrow \alpha_{n+1} = -5,663\Delta\theta_{n+1} + 0,9013\theta_n + 140,2 \quad (4.16)$$

$$\Delta\theta_{n+1}(EsquerdaG) \text{ e } \theta_n(Centro) \Rightarrow \alpha_{n+1} = -0,649\Delta\theta_{n+1} + 0,9065\theta_n + 143,6 \quad (4.17)$$

$$\Delta\theta_{n+1}(EsquerdaG) \text{ e } \theta_n(Esquerda) \Rightarrow \alpha_{n+1} = -0,649\Delta\theta_{n+1} + 0,9065\theta_n + 143,6 \quad (4.18)$$

$$\Delta\theta_{n+1}(DireitaG) \text{ e } \theta_n(EsquerdaG) \Rightarrow \alpha_{n+1} = -3,209\Delta\theta_{n+1} + 0,7381\theta_n + 160,4 \quad (4.19)$$

$$\Delta\theta_{n+1}(DireitaG) \text{ e } \theta_n(Esquerda) \Rightarrow \alpha_{n+1} = -5,663\Delta\theta_{n+1} + 0,9013\theta_n + 140,2 \quad (4.20)$$

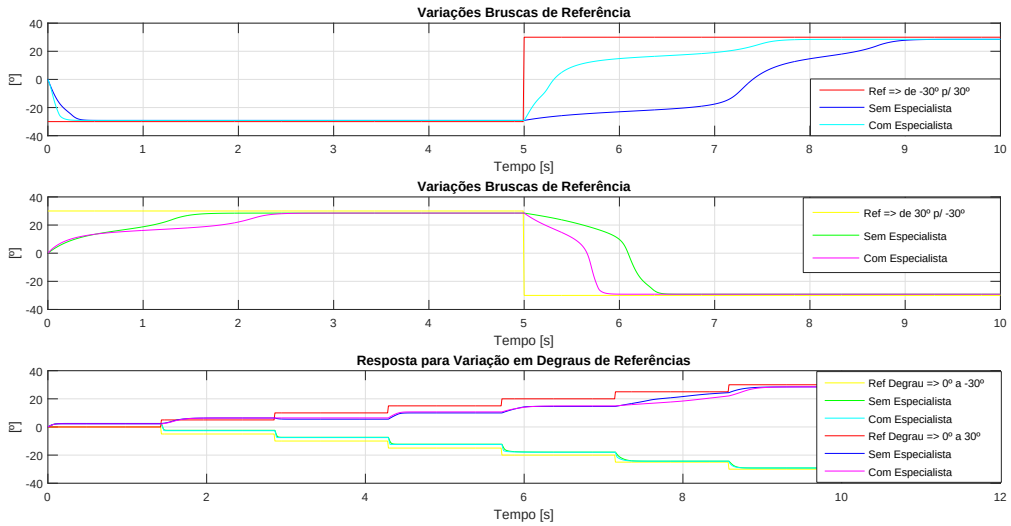
$$\Delta\theta_{n+1}(DireitaG) \text{ e } \theta_n(Centro) \Rightarrow \alpha_{n+1} = -3,528\Delta\theta_{n+1} + 0,2951\theta_n + 170,4 \quad (4.21)$$

$$\Delta\theta_{n+1}(DireitaG) \text{ e } \theta_n(Direita) \Rightarrow \alpha_{n+1} = -3,209\Delta\theta_{n+1} + 0,7381\theta_n + 160,4 \quad (4.22)$$

Com este novo controlador, as mesmas simulações mostradas na figura 41 foram repetidas, com o intuito de verificar se houve melhoras na resposta do sistema. A figura 43 mostra estas novas simulações.

Pela figura 43 vê-se que, embora para algumas situações o desempenho tenha piorado, na maioria dos casos, especialmente no mais crítico, onde se requereu uma variação brusca de referência ao protótipo, a resposta obtida com as novas regras foi melhor. Os erros estacionários mantiveram-se iguais em ambos os casos, mas no geral, o controlador com mais regras respondeu de forma mais rápida às mudanças de referência. Por ter-se conseguido agregar esta melhora ao desempenho do sistema, o controlador com as regras geradas via dados e as regras geradas via especialista será o escolhido para ser implementado no controlador.

Figura 43 – Repetição das Simulações Comparando o Controlador com e sem o Conhecimento Especialista



Fonte: Próprio Autor, 2017.

Como esta metodologia é *model free*, não se pode afirmar que esta solução seja a única, ou a mais otimizada em termos de tempo de resposta, ou erro estacionário, etc., mas ainda assim, ela atende a necessidade de executar manobras amplas e levar e manter o RMMA em diversas referências dentro de uma faixa extensa de variação do ângulo de formação, o que demonstrava-se ser um desafio em outros projetos aqui citados. Portanto, esta metodologia demonstrou-se eficiente em resolver problemas com essa natureza, ao menos em termos de simulação. O próximo passo é implementar o mesmo controle no sistema real, de forma a verificar se o sistema também apresentará um desempenho satisfatório.

5 RESULTADOS

Para implementar o controlador proposto na planta, basicamente escreveu-se a forma de inferência e *defuzzyficação* de *Takagi-Sugeno* descrita na seção 2 em forma de código para placa. Esta metodologia é implementada pelo *software* $Matlab^{©}$, sendo usada nas simulações mostradas na seção anterior, assim reescreveu-se o procedimento do método para a linguagem da placa. Este código pode ser visto no apêndice B deste projeto.

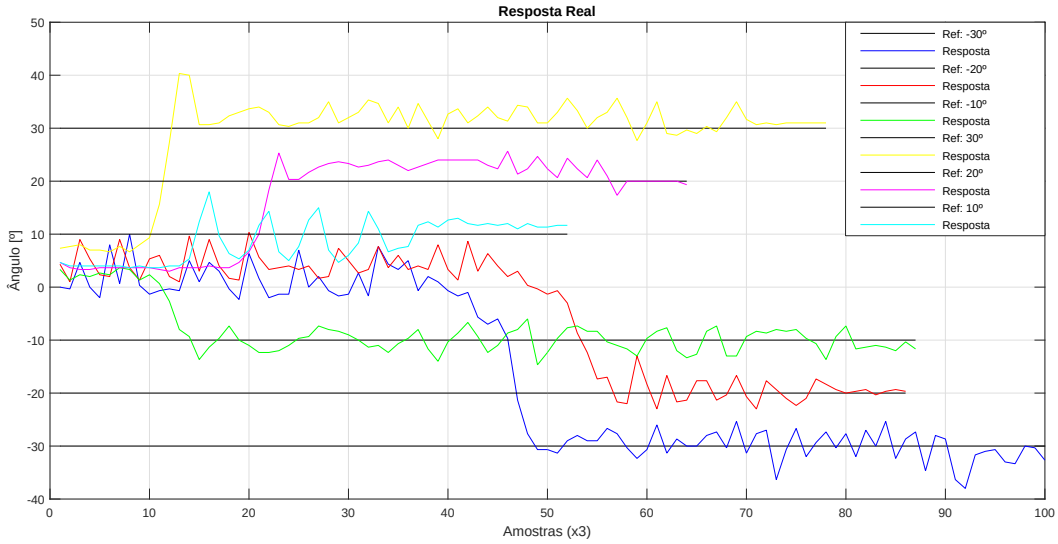
Uma vez que tenha-se implementado o controlador *fuzzy* proposto e a metodologia de inferência de *Takagi-Sugeno*, foram realizados experimentos sobre a planta com o intuito de verificar o comportamento do novo sistema. Utilizou-se de dois tipos diferentes de manobras: *tracking* e "manobra de oito". A manobra de *tracking* consiste em chegar e manter certo ângulo de giro, saindo de diversas posições iniciais. Este teste busca observar, o quão bem o sistema de controle consegue levar o protótipo para qualquer referência e como ele é mantido nela. Já a "manobra de oito" consiste em executar uma manobra de desvio de obstáculos executando um movimento de oito, ou seja, contornar dois obstáculos descrevendo um caminho na forma de um oito. Em termos práticos esse teste busca observar se efetivamente o controlador ajuda o usuário a realizar manobras consideradas difíceis, caso ele usasse somente seu conhecimento especialista do sistema para agir diretamente sobre a direção, como um motorista faria em um caminhão sem qualquer outro sistema de controle que não o exercido pelo operador.

5.1 Manobras de *Tracking*

Para realizar esta manobra, no código, fixou-se o valor da referência como constante e posicionou-se o RMMA com diversos ângulos de formação. Ao ligar-se o protótipo, utilizou-se do rádio AM apenas para movê-lo para trás deixando a encargo do sistema de controle chegar na referência determinada no código. Estes testes foram realizados para referências de $\pm 5^{o}$, $\pm 10^{o}$, $\pm 15^{o}$, $\pm 20^{o}$, $\pm 25^{o}$ e $\pm 30^{o}$, com os ângulos de formação θ_n iniciando com valores próximos a 0^{o} $\pm 30^{o}$ e $\pm 60^{o}$. As respostas obtidas com estes testes podem ser vistas nas figuras de 44 a 49.

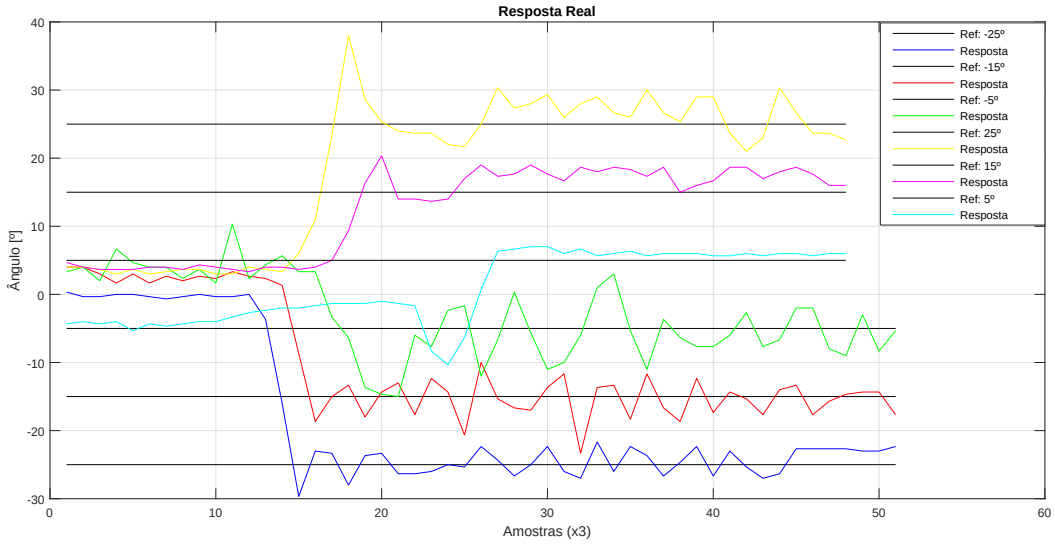
Como foi dito anteriormente, nestes testes buscou-se observar se o RMMA conseguiria sair de quaisquer posições e chegar a determinadas referências pré-determinadas. Além disso, observou-se como o protótipo se comportaria uma vez que este houvesse alcançado a referência desejada. Pôde-se observar que em todos os casos o protótipo conseguiu chegar nas referências estabelecidas, independentemente da posição de partida, e também teve

Figura 44 – Respostas Partindo-se Próximo de Zero com Referências em -30º, -20º, -10º, 10º, 20º e 30



Fonte: Próprio Autor, 2017.

Figura 45 – Respostas Partindo-se Próximo de Zero com Referências em -25º, -15º, -5º, 5º, 15º e 25º

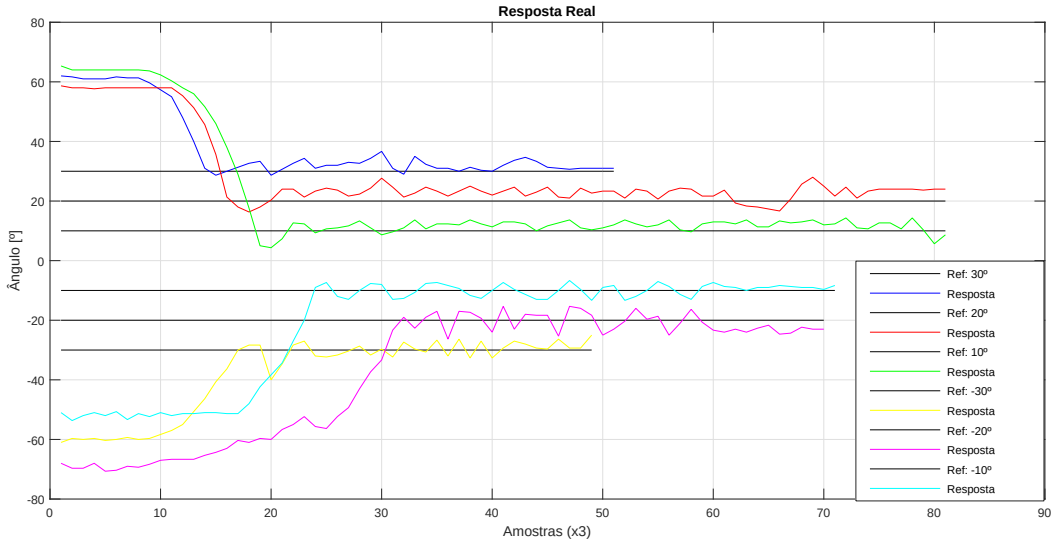


Fonte: Próprio Autor, 2017.

êxito em se manter na referência ao chegar nela, tendo uma oscilação em torno de ±5º do valor desejado. Em alguns casos a oscilação passou de forma considerável deste entorno, com mais de ±10º de oscilação, mas logo voltava à referência.

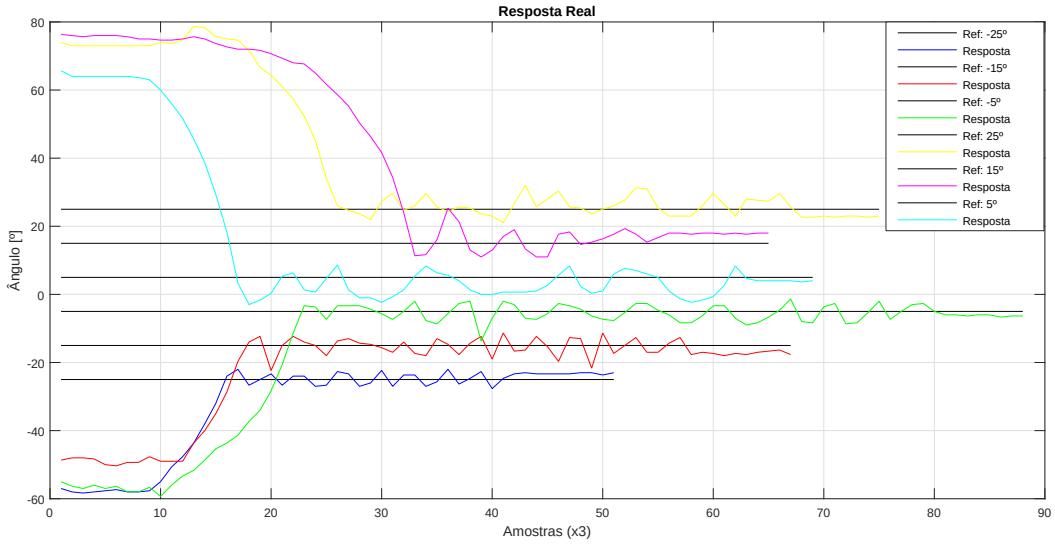
As oscilações em torno da referência não são inesperadas, pois este é um sistema real, com não-linearidades tais como ruídos e perturbações, que tendem a tirar o sistema do ponto

Figura 46 – Respostas Partindo-se do Sentido de Giro Desejado com Referências em -30º, -20º, -10º, 10º, 20º e 30



Fonte: Próprio Autor, 2017.

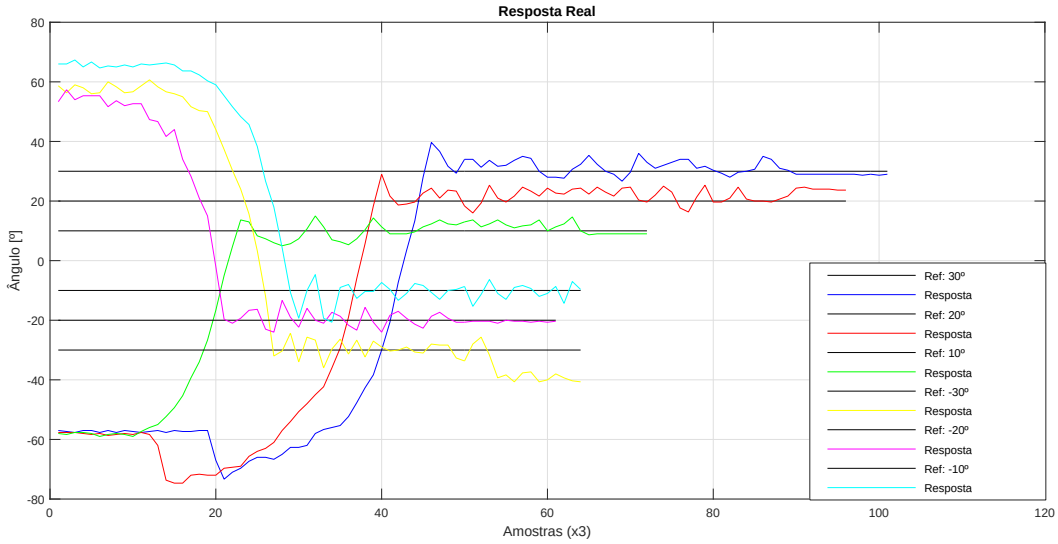
Figura 47 – Respostas Partindo-se do Sentido de Giro Desejado com Referências em -25º, -15º, -5º, 5º, 15º e 25º



Fonte: Próprio Autor, 2017.

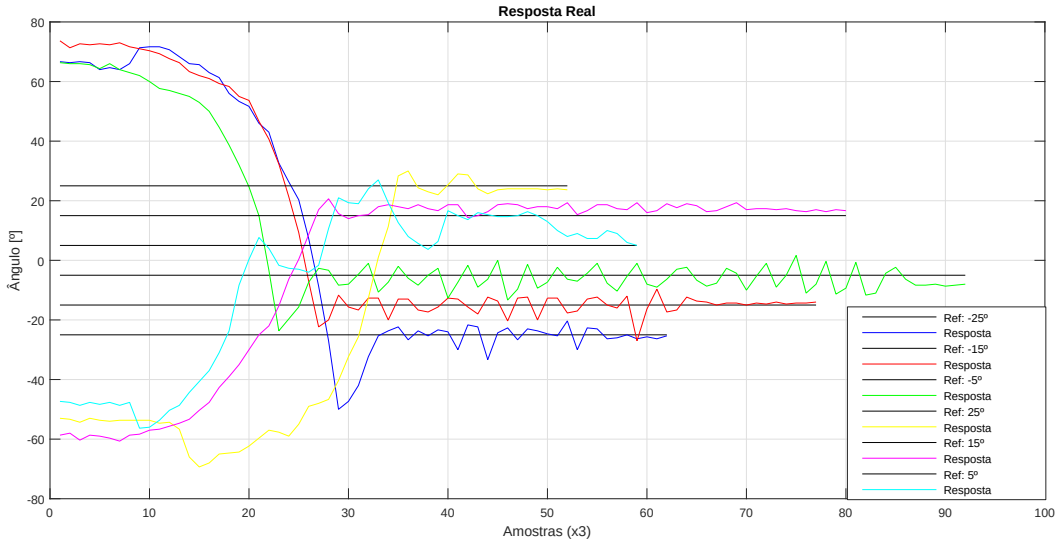
de equilíbrio em torno da referência. Além do mais, este tipo de controle também depende da velocidade, a qual foi controlada manualmente pelo operador do protótipo, logo os experimentos estavam sujeitos a uma aceleração mais brusca em alguns momentos. Isto não foge da realidade, já que o sistema de controle aqui proposto serviria para auxiliar no posicionamento da direção, mas ainda seria incumbência do motorista controlar a velocidade.

Figura 48 – Respostas Partindo-se do Sentido de Giro Oposto com Referências em -30º, -20º, -10º, 10º, 20º e 30º



Fonte: Próprio Autor, 2017.

Figura 49 – Respostas Partindo-se do Sentido de Giro Oposto com Referências em -25º, -15º, -5º, 5º, 15º e 25º



Fonte: Próprio Autor, 2017.

Deve-se ainda explicar que em algumas curvas o sistema parece ser mais lento demorando a responder à referência. O que de fato ocorria é que, para cada experimento, o protótipo era desligado e ligado. Ao ser ligado, o sistema de gravação de dados já entrava em funcionamento e nem sempre o comando de ré era dado imediatamente. Logo, os momentos em que o protótipo esteve parado na posição inicial aconteciam porque ele ainda não estava em movimento. A partir do momento em que o comando de ré era dado a configuração já

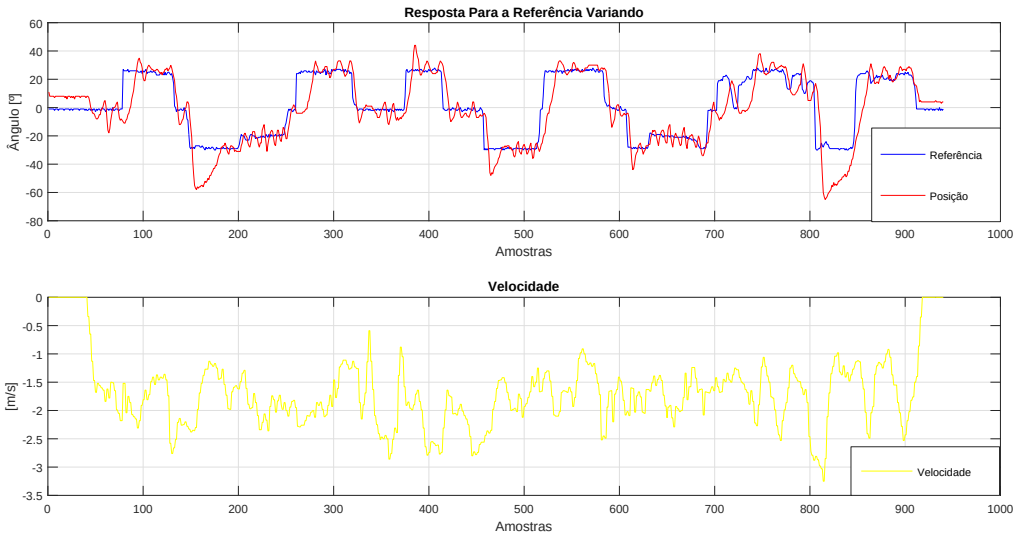
era alterada, com o sistema de controle tentando levar o RMMA à referência desejada.

Outro aspecto notável é que para algumas manobras, quando partia-se de ângulos próximos à -60º, o ângulo de formação primeiro aumentava em módulo para depois começar a diminuir. Isto ocorria porque, em casos de valores elevados de $\|\theta\|$ o sistema primeiro precisa aumentar $\|\theta\|$ para depois se estabilizar em torno de um novo valor. Após esta estabilização $\|\gamma\|$ era posto em seu valor máximo para a direção apropriada, fazendo com que $\|\theta\|$ voltasse a diminuir. Isto obviamente possui limites, havendo posições iniciais que não permitem ao sistema de controle diminuir o módulo do ângulo de formação. Entretanto, com estes experimentos, consegue-se afirmar que dentro do intervalo de -60º a +80º é possível reduzir o módulo do ângulo de formação e levar o protótipo a referências dentro destes limites.

Em termos gerais, este experimento comprovou que o controlador *fuzzy* produzido tem a capacidade de levar o protótipo de diversas formações para referências distantes e mantê-la nesta posição, mesmo havendo certa oscilação, cumprindo assim, um dos objetivos deste trabalho.

Outra experiência feita foi utilizar o segundo canal do rádio controle para mudar a referência do RMMA em tempo real. Assim, buscou-se observar o comportamento do sistema em uma situação que impunha mais dinamismo ao sistema de controle. A resposta obtida pode ser vista na figura 50.

Figura 50 – Resposta para uma Sequência de Mudanças de Referências



Fonte: Próprio Autor, 2017.

Na figura 50, observa-se maiores oscilações do que nos casos em que as referências foram fixas. Em dois momentos o protótipo passou bastante da referência, mas conseguiu retornar

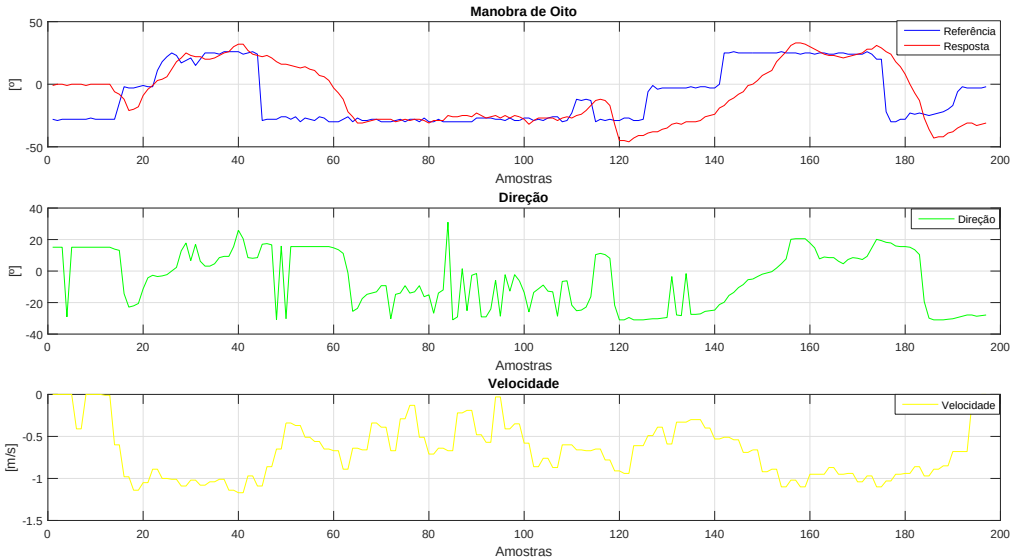
às posições desejadas sem perder a estabilidade. A inércia do protótipo pode explicar essa ultrapassagem tão elevada da referência, uma vez que movendo-se para trás com velocidade, o RMMA não conseguiria parar de imediato na referência. Apesar disto, o resultado é satisfatório , pois a resposta observada "move-se" em direção às referências, mesmo que oscile em torno delas.

Com este experimento também demonstra-se a capacidade do sistema de transitar entre referências, mesmo que distantes, de forma dinâmica, o que seria requerido em um caminhão de tamanho convencional. Logo, o sistema de controle demonstrou-se eficiente na realização de manobras de *tracking*, conseguindo atingir referências dentro de um espaço amplo de variação e manter-se nelas com alguma oscilação, mas ainda assim de forma aceitável.

5.2 Manobras Desviando-se de Obstáculos

Neste experimento, posicionou-se dois obstáculos afastados entre si em torno de 2 metros e executou-se com o RMMA uma manobra de "oito", contornando-os. Esta manobra tem como objetivo mostrar que o sistema de controle auxilia o manobrista a realizar movimentos complexos, justificando o sistema de controle com uma funcionalidade prática. No teste realizado, conseguiu-se fazer o protótipo contornar os obstáculos apenas movendo-o para trás, como era desejado. Os valores de referência passados durante a manobra, bem como a resposta obtida, além dos valores de direção e velocidade são mostrados na figura 51.

Figura 51 – Resposta para Manobra Desviando-se de Obstáculos



Fonte: Próprio Autor, 2017.

Neste experimento, limitou-se os valores máximos da referência entre ±30º, por estes terem

sido os limites usados em todas as outras experiências realizadas e por esses valores serem altos o suficiente para permitir a realização da manobra.

As mudanças nas referência vieram do usuário o qual não precisou se preocupar em como mover a direção de forma a causar o efeito desejado no ângulo de formação θ. Ao invés disso, foi necessário apenas que o usuário dissesse qual valor de θ era desejado que o sistema de controle se encarregou de chegar nele. Isto reduz o raciocínio necessário ao usuário, tornando a tarefa de realizar este tipo de manobra mais direta. Portanto, efetivamente pode-se dizer que este sistema de controle consegue auxiliar o usuário a realizar manobras que, de outra forma, seriam complexas, necessitando de um operador experiente para executá-las sem maiores transtornos.

5.3 Esforço de Controle

Em estudos de técnicas de controle comumente preocupa-se com o esforço de controle necessário para determinado sistema agir sobre a planta. Isto é relevante porque, por vezes, há limitações físicas que não permitem que certa ação seja executada e, caso chegue-se a esses limites, a planta pode ser forçada de uma forma que represente um risco a integridade física da mesma.

No caso específico deste projeto, a limitação física do controlador é o quanto as rodas dianteiras podem variar o ângulo. Como já mencionado, esta limitação é de $\pm 31^{o}$ aproximadamente. Internamente, o código não permite que os valores em PWM que comandam a direção excedam os valores que ultrapassariam os $\pm 31^{o}$, primeiro porque não adiantaria, já que não consegue-se aberturas maiores que essas e, em segundo lugar, para impedir que algum valor fora da faixa de variação aceitável faça a direção se mover de forma inesperada.

Outra consideração interessante é avaliar o valor de γ necessário para manter a formação girando com determinado ângulo θ. Em (PANDOLFI, 2012) é descrita a equação analítica 5.1 que descreve o valor de γ que mantém o sistema girando em ré por um ângulo θ desejado.

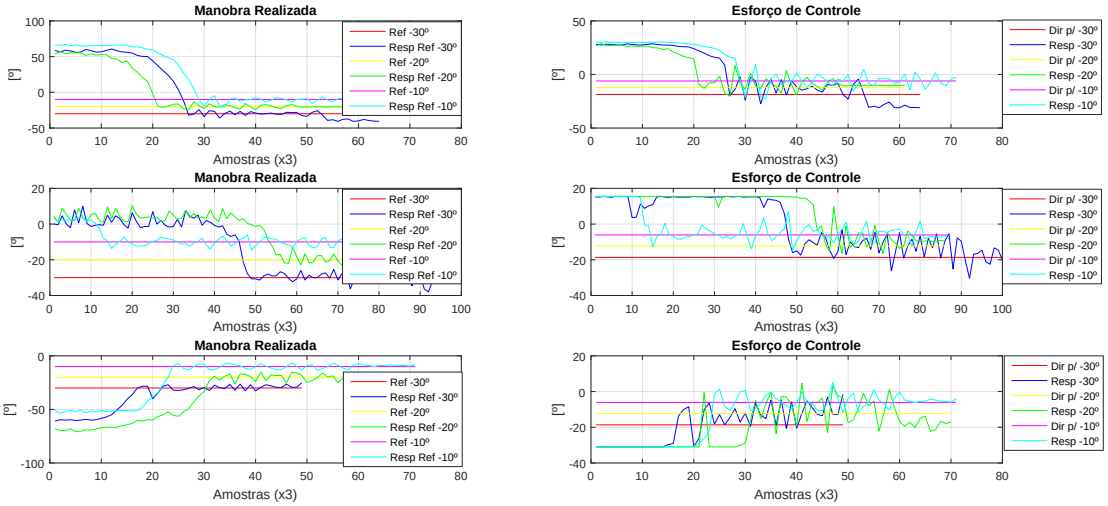
$$\gamma = \frac{1}{2} arcsin \left| \left(\frac{2A_1 sin\theta_1}{A_2 + B_1 \cos\theta_1} \right) \right| sgn(\theta_1) \tag{5.1}$$

Sendo que $sgn(x) = -1$ se $x > 0$, $sgn(x) = 1$ se $x < 0$ e $sgn(x) = 0$ se $x = 0$. Deve-se observar que originalmente o ângulo θ_1 dentro da função $sgn(x)$ era multiplicado por -1. Esta mudança ocorreu por se ter definido o lado esquerdo como positivo e não o direito,

como nos trabalhos anteriores. Isto é apenas uma questão de orientação e não afeta de forma alguma o controle, desde que se respeite as referências dadas previamente.

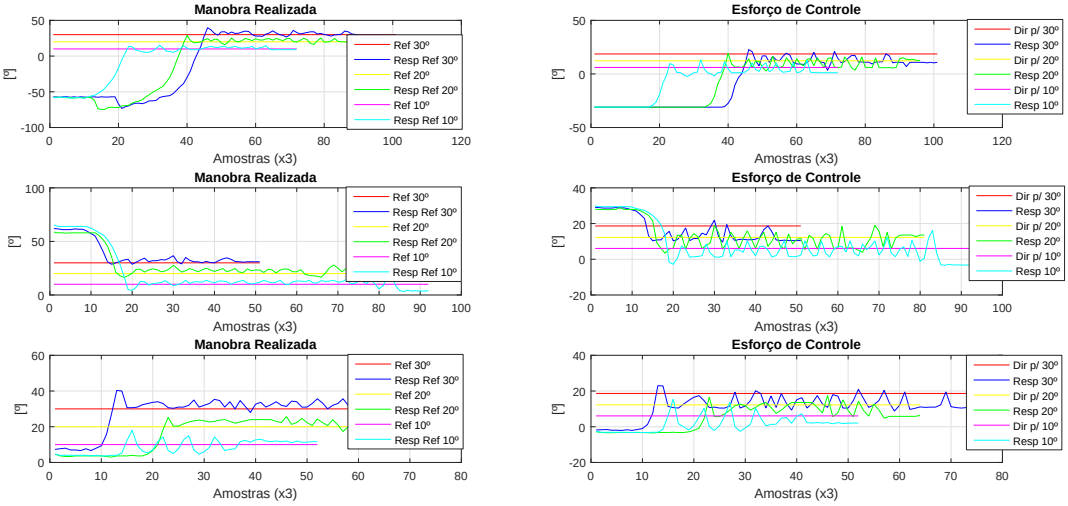
Utilizando-se da equação 5.1 e dos dados coletados do esforço de controle, observou-se o quanto a direção se aproximou de seu valor teórico no momento em que as manobras de *tracking* chegavam nas referências dadas. Estas comparações podem ser vistas nas figuras de 52 a 55.

Figura 52 – Esforço de Controle em Manobras Iniciadas com Diversas Configurações e Referências em -30º, -20º e -10º



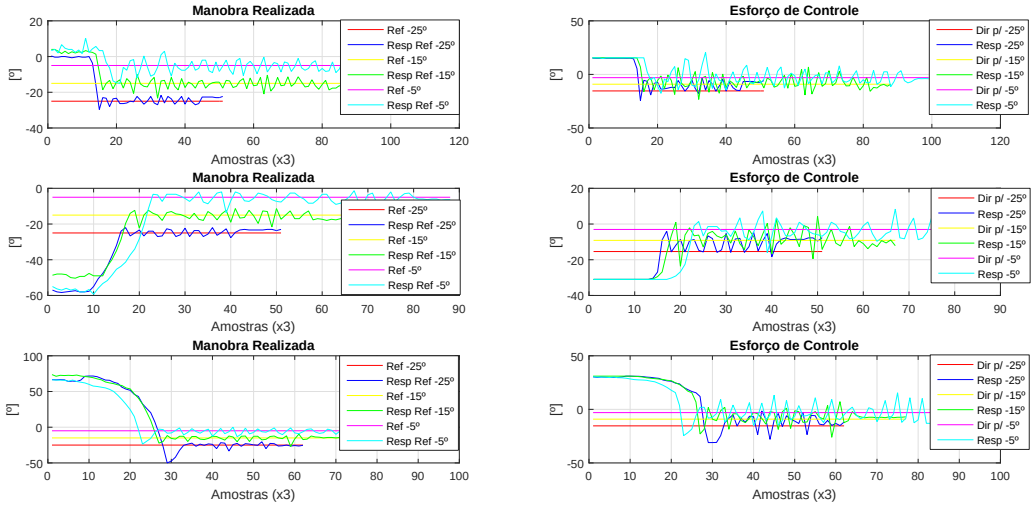
Fonte: Próprio Autor, 2017.

Figura 53 – Esforço de Controle em Manobras Iniciadas com Diversas Configurações e Referências em 30º, 20º e 10º



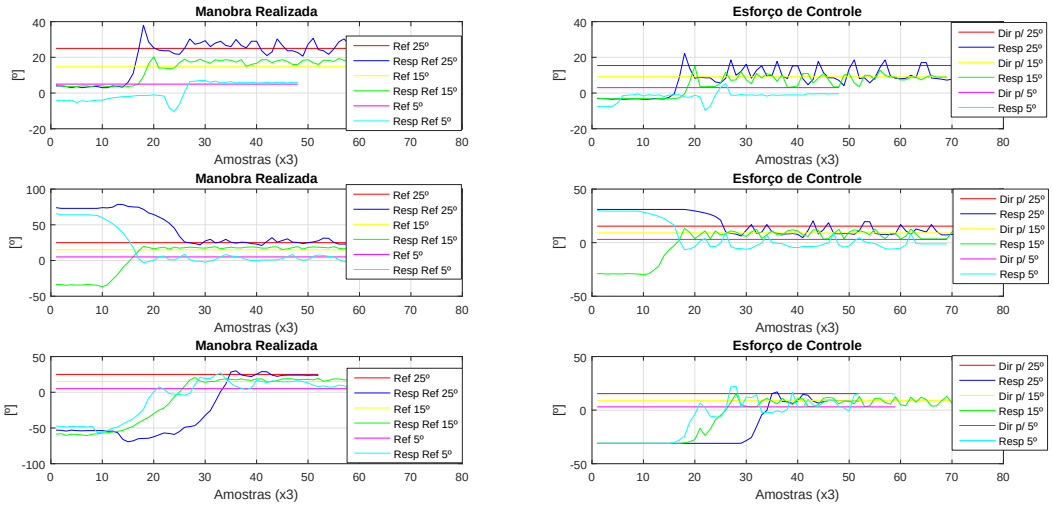
Fonte: Próprio Autor, 2017.

Figura 54 – Esforço de Controle em Manobras Iniciadas com Diversas Configurações e Referências em -25º,-15º e -5º



Fonte: Próprio Autor, 2017.

Figura 55 – Esforço de Controle em Manobras Iniciadas com Diversas Configurações e Referências em 25º,15º e 5º



Fonte: Próprio Autor, 2017.

Pelas respostas nas figuras de 52 a 55, observa-se que em alguns casos o valor de γ não foi muito próximo do esperado pela equação 5.1, chegando a ficar $\pm 10^{o}$ distante do valor teórico. Levando-se em conta as diferenças existentes entre plantas reais e modelos, esta diferença não é surpreendente, levando-se em conta as não-linearidades e ruídos existentes em sistemas reais.

Também pode-se observar que nenhuma das manobras exigiu da direção valores próximos

de seu limite enquanto a formação estava na referência. Entretanto, nas transições, a não ser em casos onde a formação já partia próxima de 0^o, o sistema de controle precisou utilizar a máxima abertura da direção γ em algum dos sentidos. Apesar disso, este esforço não foi excessivo e conforme o valor de $\|\theta\|$ diminuía a abertura de γ também diminuía.

Assim, conclui-se que o esforço de controle não representou um problema físico ao sistema, entretanto houve divergências entre os valores teóricos e práticos do ângulo γ da direção, o que reforça a dificuldade de modelos teóricos em representar sistemas reais que apresentem expressivas não-linearidades.

6 CONCLUSÃO E TRABALHOS FUTUROS

Após a realização das mudanças no *hardware* e no *software* do sistema de controle embarcado, bem como da implementação do controlador *fuzzy* proposto, pôde-se chegar as seguintes conclusões:

- Embarcar o sistema de controle do RMMA no microcontrolador, bem como a introdução do rádio AM e do módulo de gravação e leitura SD, possibilitou que a aquisição de dados do protótipo fosse feita de forma mais efetiva e rápido, tendo como conseqüência a aquisição de dados mais relevantes a respeito da dinâmica de funcionamento do sistema;
- Com os dados coletados foi possível criar um modelo *fuzzy* que conseguiu representar de forma satisfatória o sistema, mesmo em situações com ângulos de formação acentuados;
- Ao se conseguir dados mais expressivos sobre o sistema, tornou-se possível implementar um controle *fuzzy* que abrangeu um extenso espaço de variação do ângulo de formação do protótipo, o que, em termos práticos, demonstrou-se efetivo na realização de manobras complexas em ré.

A partir da estrutura construída neste projeto, abrem-se possibilidades para trabalhos futuros os quais dependam de uma quantidade de dados expressiva para sua implementação. Pode-se destacar como exemplo:

- Abordagens envolvendo redes neurais, as quais necessitam de uma grande quantidade de dados para que a rede treinada seja representativa do problema;
- Controladores híbridos *fuzzy*-PID ou *fuzzy*-neural, de forma a tentar melhorar o desempenho do sistema;
- Introduzir o segundo elemento *trailer* e aplicar a mesma metodologia para gerar um controlador *fuzzy* que atenda este sistema;
- Utilizar o modelo gerado para identificar o sistema e gerar funções de transferências que representem a planta em certos pontos de operação.

Em resumo, a troca do sistema embarcado reduziu atrasos de comando, já que o processamento do sistema de controle era feito internamente ao controlador e não mais por um computador externo.

O sistema de controle apresentou um desempenho satisfatório, por permitir aos usuários realizar manobras complexas, reduzindo o grau de conhecimento especialista necessário para executá-las efetivamente.

Além disto, a placa de desenvolvimento *Arduino* UNO utilizada neste projeto possui preços acessíveis, logo é de fácil reposição, havendo uma grande quantidade de materiais didáticos a seu respeito, além ter seu código facilmente transportado para outros modelos mais avançados da mesma família, caso haja esta intenção em outros projetos, assim muito da estrutura e do *software* se manteriam.

Em uma análise geral, este projeto foi bem sucedido, por ter demonstrado que a reestruturação pela qual o protótipo passou realmente permitiu melhoras na implementação de controladores que consigam atuar em praticamente todo o espaço de variação do ângulo de formação de um protótipo RMMA com dois elementos, um ativo e outro passivo. Além disso, abriu-se oportunidades para que novas técnicas sejam tentadas e outras já exploradas possam ser refeitas, agora com a possibilidade de adquirir-se dados mais expressivos para cada problema proposto.

7 ALOCAÇÃO DE RECURSOS

Quadro 2 – Recursos e Gastos

Material	**Preço (R$)**
Microcontrolador Arduino UNO	60,00
Bateria de Lítio	120,00
Conversor de Nível Bidirecional	2,00
Rádio AM de Dois Canais	130,00
Gravador SD	9,00
Reparo de Engrenagem	60,00
SoftWare Matlab	Disponibilizado pela UFES
Shield Arduino Uno	20,00
Fita Termo Retrátil	6,00
Potenciômetros Lineares	10,00
Buffer Driver 74HC244N	4,00
Total	421,00

Fonte: Próprio Autor, 2017

REFERÊNCIAS

AMERICA, I. T. *RC Globe Liner*. 2017. In blog. Disponível em: <https://www.tamiyausa.com/product/item.php?product-id=56304>. Acesso em: 13 Julho 2017. Citado na página 20.

ASKAR-ZADEH, L. Fuzzy sets. *Fuzzy Sets-Information and Control*, IEEE, p. 339, 1965. Citado na página 35.

ATMEL. *8-bit Microcontroller with 32KBytes In-System Programmable Flash*. 2017. In blog. Disponível em: <http://www.atmel.com/Images/doc2503.pdf>. Acesso em: 10 Julho 2017. Citado na página 48.

BERTOLANI, D. N. *Robôs Móveis Multi-Articulados - Proposta de Algoritmo Feedback Não Linear e Validação Comparativa com Soluções Fuzzy em Protótipo Real*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, Vitória - Espírito Santo, 2011. Citado 6 vezes nas páginas 22, 23, 24, 26, 27 e 60.

BERTOLANI, D. N. *Sistonia do Controle de Configuração de Robôs Móveis Multiarticulados Via Algoritmo Genético*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, Vitória - Espírito Santo, 2013. Citado na página 60.

FERREIRA, E. de P. Fundamentos e aplicações da lógica fuzzy ao controle de sistemas. p. 13, Fevereiro 2009. Citado na página 35.

FERREIRA, E. de P. Fundamentos e aplicações da lógica fuzzy ao controle de sistemas. p. 63, Fevereiro 2009. Citado na página 37.

FERREIRA, E. de P. Fundamentos e aplicações da lógica fuzzy ao controle de sistemas. p. 65, Fevereiro 2009. Citado na página 37.

FERREIRA, E. de P. Fundamentos e aplicações da lógica fuzzy ao controle de sistemas. p. 76, Fevereiro 2009. Citado na página 39.

GILAT, A.; SUBRAMANIAM, V. *Numerical Methods for Engineers and Scientists: An Introduction with Applications Using MATLAB*. [S.l.]: Wiley Bicentennial-Knowledge for Generations, 2008. 183, 184 p. Citado na página 45.

KULITZ, H. R. *Modelagem e Controle Fuzzy de Robôs e Veículos Multi-Articulados*. Tese (Doutorado) — Universidade Federal do Espírito Santo, Vitória - Espírito Santo, 2003. Citado na página 22.

LOBO, A. *O porto de Roterdã e os robôs*. 2016. In blog. Disponível em: <http://www.ilos.com.br/web/roterda-o-porto-e-os-robos/>. Acesso em: 01 Fevereiro 2016. Citado 2 vezes nas páginas 17 e 18.

MATHWORKS. *Fuzzy Logic Toolbox*. 2015. In blog. Disponível em: <https://edoras.sdsu.edu/doc/matlab/toolbox/fuzzy/fuzzyt27.html>. Acesso em: 14 Maio 2017. Citado na página 46.

MIRANDA, V. M. *Ferramentas de Auxílio ao Desenvolvimento de Preditores e Controladores Neurais Completos a Horizonte Fixo para Robôs Móveis MultiArticulados*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, Vitória - Espírito Santo, 2011. Citado 2 vezes nas páginas 28 e 60.

OLIVEIRA, T. R. B. de. *Desenvolvimento de Algoritmos e Ferramentas de Síntese e Validação de Estratégias para o Controle em Manobras ou Navegação de Veículos ou Robôs Móveis Multi-Articulados*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, Vitória - Espírito Santo, 2010. Citado na página 27.

PANDOLFI, F. *Modelagem e Controle de Robôs Móveis Multi-Articulados no Espaço de Configurações*: Soluções não-lineares e fuzzy. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, Vitória - Espírito Santo, 2012. Citado 5 vezes nas páginas 21, 25, 27, 60 e 82.

TAKAGI, T.; SUGENO, M. *Fuzzy* identification of systems and its applications to modeling and control. January/February 1985. Citado 4 vezes nas páginas 40, 41, 44 e 64.

WANG, L.-X.; MENDEL, J. M. Generating fuzzy rules by learning from examples. November/December 1992. Citado 3 vezes nas páginas 40, 41 e 64.

A CONTROLE DIRETO SOBRE AS RODAS DIANTEIRAS

define FASTADC 1

// defines for setting and clearing register bits

ifndef cbi

define cbi(sfr, bit)

$(\_SFR\_BYTE(sfr) \&= \sim \_BV(bit))$

endif

ifndef sbi

define sbi(sfr, bit)

$(\_SFR\_BYTE(sfr) \mid= \_BV(bit))$

endif

// Load the SD Library

include <SD.h>

// Create the needed objects Sd2Card SDcard; SdVolume volume;

/* RC PulseIn Serial Read out

By: Nick Poole

SparkFun Electronics

Date: 5

License: CC-BY SA 3.0 - Creative commons share-alike 3.0 use this code however you'd like, just keep this license and attribute. Let me know if you make hugely, awesome, great changes. */

```
// ———————— Begin to declare variables ————————————-

int ch1; // Here's where we'll keep channel 1 values

int ch2; // Here's where we'll keep channel 2 values

int ch1 _ max = 1820; // maximum value allowed to channel 1

int ch1 _ min = 1400; // minimum value allowed to channel 1

int ch2 _ max = 2205; // maximum value allowed to channel 2

int ch2 _ min = 1115; // minimum value allowed to channel 2

int mover; // it keeps the values to drive the motor

int girar; // it keeps the values to turn the wheels

int gear = 155; // it sets the value for the gear

int Pot0 = A1; // It measures the angle seen by the potentiometer 1.

int angle1 = 0; // It receives the angle of the traillers in degrees

volatile int pulses = 0; // It counts the pulses given by the encoder

unsigned long old _ time = 0; // It saves the past time

unsigned long new _ time = 0; // It saves the current time

float velocidade = 0; // It receives the current speed of the truck

float velocidade _ old = 0; // It receives the past speed of the truck

float factor = 0.00394; // Factor of meters/pulses const int chipSelect = 4;

// ———————— End to declare variables ————————————-

void setup() {

# if FASTADC // set prescale to 16

sbi(ADCSRA,ADPS2) ;
```

```
cbi(ADCSRA,ADPS1) ;

cbi(ADCSRA,ADPS0) ;

# endif

// Initialize parameters for saving at the SD Card

Serial.begin(250000); // set baud speed at the monitor terminal

//Inicia a comunicacao com o modulo SD

if (!SD.begin(chipSelect))

{

Serial.println("Falha ao acessar o cartao !");

Serial.println("Verifique o cartao/conexoes e reinicie o Arduino...");

return;

}

Serial.println("Cartao iniciado corretamente !");

Serial.println("Aguardando acionamento do push-button...");

Serial.println();

//————————Define Inputs————————————

pinMode(2, INPUT); // Set pin 2 as an input

pinMode(5, INPUT); // Set pin 5 as an input

pinMode(6, INPUT); // Set pin 6 as an input

pinMode(3, OUTPUT); // Set pin 3 as an output

//————————Enable Interruption————————-

attachInterrupt(digitalPinToInterrupt(2), counter _ pulses, FALLING); // Interruption
for reading the encoder
```

```
//Serial.begin(250000); // set baud speed at the monitor terminal

File dataFile = SD.open("arquivo.txt", FILE _ WRITE);

// Grava os dados no arquivo

if (dataFile)

{

dataFile.print("Velocidade [m/s]");

dataFile.print(" ");

dataFile.print("Direcao [PWM]");

dataFile.print(" ");

data Fileprint("Angulo [ 0 ] ");

dataFile.println();

dataFile.close();

}

analogWrite(3,gear); // It sets the gear at the strongest one

}

void counter _ pulses()

{

pulses ++; // It counts one transition at the encoder

}

void loop() {

//// ———— Encoder Section ————————

new _ time = millis();
```

```
if (new _ time - old _ time >= 300) // It updates the counter at each 0.3 seconds

{ detachInterrupt(digitalPinToInterrupt(2)); //Desable interruption during calculation

velocidade = (factor * pulses * 1000)/(new _ time - old _ time); // It converts pulses to linear speed

pulses = 0; // It resets pulses

// Here it checks the direction of the moviment

if(mover > 180)

{

velocidade = velocidade*(-1);

}

if ((velocidade _ old < 0) & & (velocidade > 0))

{

velocidade = velocidade*(-1);

}

velocidade _ old = velocidade;

// Update time

old _ time = millis(); // It updates the old _ time variable

// Re-open Interruption attachInterrupt(digitalPinToInterrupt(2), counter _ pulses, FALLING);

}

//// ————————- Gravar Dados ————————————-

Serial.print(velocidade);

Serial.print(" ");
```

```
Serial.print(girar);

Serial.print(" ");

Serial.print(angle1);

Serial.print(" ");

Serial.print(mover);

Serial.println(";");

// ————————- Channel 1 ————————————-

ch1 = pulseIn(6, HIGH, 300000); // It controls speed

// It sets ch1 in the middle when there is a measurement equal to 0

if (ch1 == 0)

{

ch1 = ch1 _ min + ((ch1 _ max - ch1 _ min)/2);

}

// It sets the top saturation for the driver

if (ch1 > ch1 _ max)

{

ch1 = ch1 _ max;

}

// It sets the botton saturation for the driver

if (ch1 < ch1 _ min)

{

ch1 = ch1_ min;
```

```
}

mover = map(ch1, ch1 _ min,ch1 _ max, 192, 163); // Map the channel 1

analogWrite(10,mover); // Put ”mover” at pin 10

// ——————- Channel 2 ————————————-

ch2 = pulseIn(5, HIGH, 300000); // It controls direction

// It sets ch2 in the middle when there is a measurement equal to 0

if (ch2 == 0)

{

ch2 = ch2 _ min + ((ch2 _ max - ch2 _ min)/2);

}

// It sets the top saturation for the direction

if (ch2 > ch2 _ max)

{

ch2 = ch2 _ max;

}

// It sets the botton saturation for the direction

if (ch2 < ch2 _ min)

{

ch2 = ch2 _ min;

}

girar = map(ch2, ch2 _ min,ch2 _ max, 70, 230); // Map the channel 2 to the PWM
output

analogWrite(9,girar); // Put ”girar” at pin 11
```

```
// delay(100);

// ”I put this here just to make the termina window happier” - He said...

// ——————- Read Potentiometer 0 ——————————-

Pot0 = analogRead(0); // It reads the analog input at A0

angle1 = map(Pot0, 20,560, -90, 90); // Map the Potentiometer to the corresponding angle

// ——————- Gravar Dados ——————————-

// Serial.println(angle1);

File dataFile = SD.open(”arquivo.txt”, FILE _ WRITE);

// Grava os dados no arquivo

if (dataFile){

dataFile.print(velocidade);

dataFile.print(” ”);

dataFile.print(girar);

dataFile.print(” ”);

dataFile.print(angle1);

dataFile.println(”;”);

dataFile.close();

}

else

{

// Mensagem de erro caso ocorra algum problema na abertura do arquivo Serial.println(”Erro
ao abrir arquivo.txt !”);

}
```

```
}
```

B IMPLEMENTAÇÃO DO CONTROLADOR FUZZY ATUANTE SOBRE O ÂNGULO DE CONFIGURAÇÃO

define FASTADC 1

// defines for setting and clearing register bits

ifndef cbi

define cbi(sfr, bit)

$(\_SFR\_BYTE(sfr) \&= \sim \_BV(bit))$

endif

ifndef sbi

define sbi(sfr, bit)

$(\_SFR\_BYTE(sfr) \mid= \_BV(bit))$

endif

// Load the SD Library

include <SD.h>

// Create the needed objects Sd2Card SDcard; SdVolume volume;

/* RC PulseIn Serial Read out

By: Nick Poole

SparkFun Electronics

Date: 5

License: CC-BY SA 3.0 - Creative commons share-alike 3.0 use this code however you'd like, just keep this license and attribute. Let me know if you make hugely, awesome, great changes. */

// ———————— Begin to declare variables ————————-

int ch1; // Here's where we'll keep channel 1 values

int ch2; // Here's where we'll keep channel 2 values

int ch1 _ max = 1820; // maximum value allowed to channel 1

int ch1 _ min = 1400; // minimum value allowed to channel 1

int ch2 _ max = 2215; // maximum value allowed to channel 2

int ch2 _ min = 1115; // minimum value allowed to channel 2

int delta _ ref _ max = 6; // maximum value allowed to the reference

int delta _ ref _ min = -6; // minimum value allowed to the reference

int mover; // it keeps the values to drive the motor

int girar; // it keeps the values to turn the wheels

int gear = 155; // it sets the value for the gear

int Pot0 = A1; // It measures the angle seen by the potentiometer 1.

int angle1 = 0; // It receives the angle of the traillers in degrees float ang _ dir = 0;

int referencia = 0; // It gives the reference for the angle of the trailer

int delta _ referencia = 0;

volatile int pulses = 0; // It counts the pulses given by the encoder

unsigned long old _ time = 0; // It saves the past time

unsigned long new _ time = 0; // It saves the current time

float velocidade = 0; // It receives the current speed of the truck

```
float velocidade _ old = 0; // It receives the past speed of the truck

float factor = 0.00394; // Factor of meters/pulses

const int chipSelect = 4;

// ———————— End to declare variables ————————-

// ————————— Beggin of the initial set ups ————————

void setup()

# if FASTADC

// set prescale to 16

sbi(ADCSRA,ADPS2) ;

cbi(ADCSRA,ADPS1) ;

cbi(ADCSRA,ADPS0) ;

# endif

// Initialize parameters for saving at the SD Card

Serial.begin(250000); // set baud speed at the monitor terminal

//It Starts the Communication with the SD Module

if (!SD.begin(chipSelect))

{

Serial.println( " Falha ao acessar o cartao ! " );

Serial.println( " Verifique o cartao/conexoes e reinicie o Arduino... " );

return;

}

Serial.println( " Cartao iniciado corretamente !" );
```

```
Serial.println( " Aguardando acionamento do push-button... " );

Serial.println();

//————————Define Inputs————————————

pinMode(2, INPUT); // Set pin 2 as an input

pinMode(5, INPUT); // Set pin 5 as an input

pinMode(6, INPUT); // Set pin 6 as an input

pinMode(3, OUTPUT); // Set pin 3 as an output

//————————Enable Interruption————————-

attachInterrupt(digitalPinToInterrupt(2), counter _ pulses, FALLING); // Interruption
for reading the encoder

//Serial.begin(250000); // set baud speed at the monitor terminal

File dataFile = SD.open( " arquivo.txt " , FILE _ WRITE);

// Grava os dados no arquivo

if (dataFile)

{

dataFile.print( " Velocidade [m/s] " );

dataFile.print( " " );

dataFile.print( " Direcao [^0] " );

dataFile.print(" ");

dataFile.print(" Angulo [^0] ");

dataFile.print(" ");

dataFile.print("Referência [^0]");

dataFile.println();
```

```
dataFile.close();

}

analogWrite(3,gear); // Set the gear at the strongest value

}

// ———— End of the initial set ups ————

// ———————— Interruption ————————

void counter _ pulses()

{

pulses ++;

}

// ————————- Start main code ————————-

void loop() {

//// ————— Encoder Section ———————

new _ time = millis();

if (new _ time - old _ time >= 300) // It updates the counter at each 0.3 seconds

{

detachInterrupt(digitalPinToInterrupt(2)); //Desable interruption during calculation

velocidade = (factor * pulses * 1000)/(new _ time - old _ time); // It converts pulses to
linear speed

pulses = 0; // It resets pulses

// Defines the direction of the moviment

if(mover > 180)

{
```

```
velocidade = velocidade*(-1);

}

if ((velocidade _ old < 0) & & (velocidade > 0))

{ velocidade = velocidade*(-1);

}

velocidade _ old = velocidade;

// Update time

old _ time = millis(); // It updates the old _ time variable

// Re-open Interruption attachInterrupt(digitalPinToInterrupt(2), counter _ pulses, FAL-
LING);

// ————————- Channel 1 ————————————-

ch1 = pulseIn(6, HIGH, 300000); // It controls speed

// Places ch1 at the middle when ch1 = 0

if (ch1 == 0)

{

ch1 = ch1 _ min + ((ch1 _ max - ch1 _ min)/2);

}

// It sets the top saturation for the driver

if (ch1 > ch1 _ max)

{

ch1 = ch1 _ max;

}

// It sets the botton saturation for the driver
```

```
if (ch1 < ch1 _ min)

{

ch1 = ch1 _ min;

}

mover = map(ch1, ch1 _ min,ch1 _ max, 190, 155); // Map the channel 1

analogWrite(10,mover); // Put "mover"at pin 10

// ————————- Read Potentiometer 0 ————————————-

Pot0 = analogRead(0); // It reads the analog input at A0 angle1 = map(Pot0, 20,560,
-90, 90); // Map the Potentiometer to the corresponding angle

// ————————- Channel 2 ————————————-

ch2 = pulseIn(5, HIGH, 300000); // It controls direction

// Places ch2 at the middle when ch1 = 0

if (ch2 == 0)

{ ch2 = ch2 _ min + ((ch2 _ max - ch2 _ min)/2); }

// It sets the top saturation for the direction

if (ch2 > ch2 _ max)

{

ch2 = ch2 _ max;

}

// It sets the botton saturation for the direction

if (ch2 < ch2 _ min)

{

ch2 = ch2 _ min;
```

```
}

// If the truck is moving forward then the control upon the direction is direct

if (mover < 168)

{

girar = map(ch2, ch2 _ min, ch 2_ max, 70, 230); // Map the channel 2 to the PWM output

analogWrite(9,girar); // Put " girar " at pin 11 }

else // If the truck is moving backward then the control will be under the angle of the articulations

{

referencia = map(ch2, ch2 _ min, ch2 _ max, -30, 30);

// referencia = 5; delta _ referencia = referencia - angle1;

// It sets the top saturation for the reference

if (delta _ referencia > delta _ ref _ max)

{

delta _ referencia = delta _ ref _ max;

}

// It sets the botton saturation for the reference

if (delta _ referencia < delta _ ref _ min)

{

delta _ referencia = delta _ ref _ min;

}

girar = Control(delta _ referencia, angle1);
```

```
analogWrite(9,girar); // Put " girar " at pin 11

}

// delay(100); // " I put this here just to make the terminal window happier " - He said...

// ——————- Gravar Dados ——————————-

// Serial.println(angle1);

ang _ dir = 0.3875*girar - 58.125;

File dataFile = SD.open(" arquivo.txt " , FILE _ WRITE);

// Grava os dados no arquivo

if (dataFile) {

dataFile.print(velocidade);

dataFile.print(" ");

dataFile.print(ang _ dir);

dataFile.print(" ");

dataFile.print(angle1);

dataFile.print(" ");

dataFile.print(referencia);

dataFile.println(" ; ");

dataFile.close();

}

else

{

// Mensagem de erro caso ocorra algum problema na abertura do arquivo Serial.println("
Erro ao abrir arquivo.txt ! ");
```

```
}

// Takagi-Sugeno Inference

}

int Control(int in1, int in2)

{

// ——- Parameters for the Linguistic Variables Parameters:———

const float sigma11 = 1.2728;

const float sigma12 = 1.2728;

const float sigma13 = 1.2728;

const float sigma14 = 1.2728;

const float sigma15 = 1.2728;

const float sigma21 = 10.6066;

const float sigma22 = 10.6066;

const float sigma23 = 10.6066;

const float sigma24 = 10.6066;

const float sigma25 = 10.6066;

const int c11 = -6;

const int c12 = -3;

const int c13 = 0;

const int c14 = 3;

const int c15 = 6;

const int c21 = -50;
```

```
const int c22 = -25;

const int c23 = 0;

const int c24 = 25;

const int c25 = 50;

// —— Parameters for the Output: ——————————

const float a1 = -3.4227;

const float b1 = -0.1602;

const float c1 = 107.7498;

const float a2 = -5.6629;

const float b2 = 0.9013;

const float c2 = 140.2122;

const float a3 = -0.6490;

const float b3 = 0.9065;

const float c3 = 143.6462;

const float a4 = -3.2095;

const float b4 = 0.7381;

const float c4 = 160.4009;

const float a5 = -3.5279;

const float b5 = 0.2951;

const float c5 = 170.3535;

// ——————- Calculate the Membership Curves ———

float MS1 _ in1 = exp(((-1)*pow(in1 - c11,2))/(2*pow(sigma11,2)));
```

```
float MS2 _ in1 = exp(((-1)*pow(in1 - c12,2))/(2*pow(sigma12,2)));

float MS3 _ in1 = exp(((-1)*pow(in1 - c13,2))/(2*pow(sigma13,2)));

float MS4 _ in1 = exp(((-1)*pow(in1 - c14,2))/(2*pow(sigma14,2)));

float MS5 _ in1 = exp(((-1)*pow(in1 - c15,2))/(2*pow(sigma15,2)));

float MS1 _ in2 = exp(((-1)*pow(in2 - c21,2))/(2*pow(sigma21,2)));

float MS2 _ in2 = exp(((-1)*pow(in2 - c22,2))/(2*pow(sigma22,2)));

float MS3 _ in2 = exp(((-1)*pow(in2 - c23,2))/(2*pow(sigma23,2)));

float MS4 _ in2 = exp(((-1)*pow(in2 - c24,2))/(2*pow(sigma24,2)));

float MS5 _ in2 = exp(((-1)*pow(in2 - c25,2))/(2*pow(sigma25,2)));

// ———— Weight of the Variables: ————————

float W1 = MS1 _ in1*MS1 _ in2;

float W2 = MS2 _ in1*MS2 _ in2;

float W3 = MS3 _ in1*MS3 _ in2;

float W4 = MS4 _ in1*MS4 _ in2;

float W5 = MS5 _ in1*MS5 _ in2;

// Rules Made by Expertises:

float W6 = MS5 _ in1*MS1 _ in2;

float W7 = MS5 _ in1*MS2 _ in2;

float W8 = MS5 _ in1*MS3 _ in2;

float W9 = MS5 _ in1*MS4 _ in2;

float W10 = MS1 _ in1*MS5 _ in2;

float W11 = MS1 _ in1*MS4 _ in2;
```

```
float W12 = MS1 _ in1*MS3 _ in2;

float W13 = MS1 _ in1*MS2 _ in2;

// Output Equations:

float Z1 = a1*in1 + b1*in2 + c1;

float Z2 = a2*in1 + b2*in2 + c2;

float Z3 = a3*in1 + b3*in2 + c3;

float Z4 = a4*in1 + b4*in2 + c4;

float Z5 = a5*in1 + b5*in2 + c5;

// Outputs Made by Expertises:

float Z6 = a2*in1 + b2*in2 + c2;

float Z7 = a2*in1 + b2*in2 + c2;

float Z8 = a3*in1 + b3*in2 + c3;

float Z9 = a3*in1 + b3*in2 + c3;

float Z10 = a4*in1 + b4*in2 + c4;

float Z11 = a2*in1 + b2*in2 + c2;

float Z12 = a5*in1 + b5*in2 + c5;

float Z13 = a4*in1 + b4*in2 + c4;

// Final Output

float Z = (Z1*W1 + Z2*W2 + Z3*W3 + Z4*W4 + Z5*W5 + Z6*W6 + Z7*W7 + Z8*W8
+ Z9*W9 + Z10*W10 + Z11*W11 + Z12*W12 + Z13*W13)/(W1 + W2 + W3 + W4 +
W5 + W6 + W7 + W8 + W9 + W10 + W11 + W12 + W13);

// It sets the top saturation for the control response

if (Z > 230)
```

```
{
Z = 230;
}
// It sets the botton saturation for the control response
if (Z < 70)
{
Z = 70;
}
return Z;
}
```