UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ENGENHARIA ELETRÔNICA

HUGO LEONARDO MINANTI DOS ANJOS

CONTROLADOR FUZZY EM FPGA PARA APLICAÇÂO EM SISTEMAS INTELIGENTES DE NAVEGAÇÂO

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO
2018

HUGO LEONARDO MINANTI DOS ANJOS

CONTROLADOR FUZZY EM FPGA PARA APLICAÇÂO EM SISTEMAS INTELIGENTES DE NAVEGAÇÂO

Trabalho de conclusão de curso, apresentado à disciplina de TCC 2, do curso Superior de Engenharia Eletrônica do Departamento Acadêmico de Eletrônica - DAELN - da Universidade Tecnológica Federal do Paraná - UTFPR, como requisito parcial para a obtenção do título de engenheiro em eletrônica.

Orientador: Prof. Dr. Márcio Rodrigues da Cunha

CAMPO MOURÃO

2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Campo Mourão
Coordenação do Curso de Engenharia Eletrônica

TERMO DE APROVAÇÃO

DO TRABALHO DE CONCLUSÃO DE CURSO INTITULADO

CONTROLADOR FUZZY EM FPGA PARA APLICAÇÂO EM SISTEMAS INTELIGENTES DE NAVEGAÇÂO

por

HUGO LEONARDO MINANTI DOS ANJOS

Trabalho de Conclusão de Curso apresentado no dia 10 de agosto de 2018 ao Curso Superior de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná, Campus Campo Mourão. O Candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho _____ (aprovado, aprovado com restrições ou reprovado).

_____
Prof. Dr. Eduardo Giometti Bertogna
(UTFPR)

_____
Prof. Me. Lucas Ricken Garcia
(UTFPR)

_____
Prof. Dr. Marcio Rodrigues da Cunha
(UTFPR)
Orientador

A Folha de Aprovação assinada encontra-se na Coordenação

AGRADECIMENTOS

Primeiramente, agradeço a Deus, por ter me dado a oportunidade de chegar até aqui e me guiado pelo caminho. Agradeço também aos meus pais, Josué Alves dos Anjos e Sandra Aparecida Minanti dos Anjos, que lutaram por mim e fizeram tudo a favor do meu crescimento pessoal e profissional. Não sei o que seria de mim sem eles.

Aos familiares, que foram muito importantes, me ajudando sempre que foi preciso, me auxiliando no transporte e também financeiramente, e eu não teria chegado tão longe sem a ajuda deles.

Ao meu orientador e professor, Márcio Rodrigues da Cunha, pela sua grande contribuição no meu processo de formação de novos conhecimentos, e pela tolerância com o meu desempenho no desenvolvimento deste trabalho.

A toda equipe do Departamento de Engenharia Eletrônica e a todos os professores que tive, que de forma direta e indireta contribuíram para minha formação.

Aos amigos, que me encorajaram nesta jornada, e sempre estiveram do meu lado dando todo o apoio necessário durante todo o processo de formação.

Sou eternamente grato também a alguns professores que tive durante a minha vida, que desde o início de minha vida escolar tiveram muita sabedoria na sua maneira de ensinar e educar.

RESUMO

O escopo deste trabalho foi o desenvolvimento e a implementação de um sistema de navegação inteligente utilizando-se da lógica *fuzzy*, desenvolvida na linguagem de descrição de *hardware* VHDL, e a sua implementação usando o *software* de desenvolvimento Quartus II, da fabricante ALTERA. O objetivo do sistema de navegação proposto foi de que o protótipo caminhe sem colidir em nenhum obstáculo, usando para isso, três sensores de distância localizados a frente e aos lados do protótipo, e se locomovendo através de dois motores de corrente contínua. Os resultados deste trabalho, obtidos por meio de simulação e implementação prática foram comparados aos resultados obtidos com as simulações feitas no *software* do MATLAB.

PALAVRAS CHAVE: Lógica *Fuzzy*. Controle inteligente. Sensores ultrassônicos. VHDL. FPGA.

ABSTRACT

This work proposes the development and implementation of an intelligent navigation system using fuzzy logic, developed in the hardware description language VHDL, and its implementation using the development software Quartus II, from the manufacturer ALTERA. The proposed navigation system aims the prototype walk without colliding in any obstacle using, for this, three distance sensors located at the front and sides of the prototype, and moved by two direct current motors. The results of this work, obtained through simulation and practical implementation, were compared to the results obtained with the simulations made in MATLAB software.

KEYWORDS: Fuzzy Logic. Intelligent control. Ultrassonic sensors. VHDL. FPGA.

LISTA DE FIGURAS

SUMÁRIO

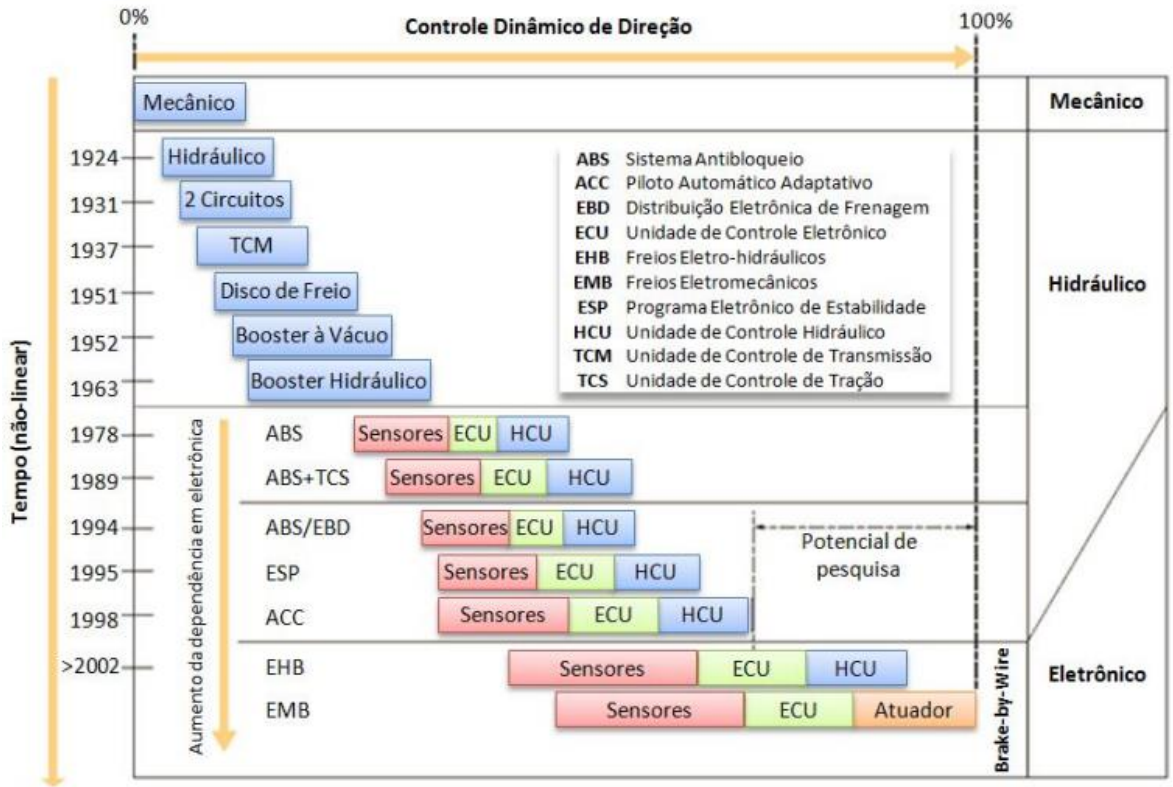
1 INTRODUÇÃO

Um panorama geral das tecnologias presentes num veículo automotivo pode ser observado na Figura 1, que mostra de forma simples quando certos avanços tecnológicos passaram a ser utilizados nos automóveis.

Na Figura 1, é possível notar a evolução dos automóveis, que deixaram de ser puramente mecânicos, se tornaram hidráulicos e recentemente muito dependentes de dispositivos eletrônicos.

No início da eletrônica automotiva, em meados da década de 70, os carros começaram a desfrutar das vantagens proporcionadas pelos diversos sistemas eletrônicos que começaram a surgir, tais como vários tipos de sensores, a Unidade de Controle Eletrônico (ECU), a Unidade de Controle Hidráulico (HCU) e o Sistema Anti-bloqueio (ABS), (FÓRMULA UFSCAR, 2016).

Figura 1 - Linha do tempo das evoluções tecnológicas em um veículo.



Fonte: FÓRMULA UFSCAR, 2016.

Como previamente apresentado, os carros possuem diversos tipos de sensores, que são elementos fundamentais para realizar o monitoramento e o controle de diversos equipamentos e sistemas, visto que sensores são dispositivos sensíveis a um tipo de energia do ambiente, como por exemplo: luz, calor, movimento, som, etc. Desta forma existem muitos tipos de sensores, tais como óticos, ultrassônicos, térmicos, etc. (THOMAZINI e ALBUQUERQUE, 2010).

Devido a suas características, os sensores podem medir todos os tipos de grandezas, tais como velocidade, temperatura, pressão, comprimento, etc. Além dos sensores que obtém as informações do sistema em estudo, sistemas de controle contam com atuadores, que fazem o trabalho contrário ao dos sensores, atuando sobre o sistema controlado através de algum dispositivo, tais como motores, válvulas, resistências, dentre outros, com o intuito de mantê-lo em condições especificadas (THOMAZINI e ALBUQUERQUE, 2010).

A ECU é o sistema mais importante dos automóveis, conhecida como o coração do veículo, uma vez que é a responsável por manter o motor funcionando. A ECU é basicamente um microprocessador, que recebe as informações dos diversos sensores espalhados pelo veículo, e através de valores pré-programados ela comanda os atuadores presentes no veículo (como por exemplo, a bomba de combustível e a ignição das velas) para manter o motor funcionando da forma mais eficiente possível (DIAS, 2012).

O ABS é um sistema que monitora constantemente as rodas do veículo que utilizam freio a disco, de modo que quando o freio é acionado a ponto de as rodas chegarem à eminência de um travamento, o sistema entra em ação impedindo o travamento das rodas. Ele é um item de segurança de extrema importância, pois permite que o veículo seja manuseado normalmente durante uma freada, diminuindo substancialmente a distância de frenagem (LIMA, 2009).

Comumente, os sistemas eletrônicos automotivos são embarcados, ou seja, são sistemas que possuem *hardware* e *software* dedicados a uma função, e diferentemente dos computadores comuns, são projetados para realizar tarefas específicas. Uma tecnologia muito utilizada no desenvolvimento de sistemas embarcados são os Dispositivos Lógicos Programáveis (PLD), que permitem a criação de *hardwares* específicos (PORTAL SISTEMAS EMBARCADOS, 2015).

Os PLDs podem ser utilizados para o desenvolvimento de sistemas complexos através dos Dispositivos Lógicos Programáveis complexos (CPLD) e dos Arranjos de

Portas Programáveis em Campo (FPGA), que permitem o desenvolvimento de *hardware*s maiores e mais robustos (PEDRONI, 2010).

O rumo que a tecnologia automotiva vem tomando, aponta para o desenvolvimento de veículos autônomos, capazes de transitar sozinhos através do uso de câmeras, sensores e de sistemas de localização como o Sistema de Posicionamento Global (GPS). O surgimento dos veículos autônomos traz à tona uma perspectiva de controle diferente, o controle inteligente (PEQUENINO, 2017).

Sistemas de controle inteligente são sistemas que fazem o uso da Inteligência Artificial (IA), tais como redes neurais artificias (RNA), que emulam as funções biológicas mais simples do cérebro humano, sendo capazes de aprender, e a lógica nebulosa também conhecida como lógica *fuzzy*, que é capaz de emular de certa forma a maneira humana de pensar (SIMÕES e SHAW, 2007).

A navegação de carros autônomos é uma área bastante explorada atualmente. Criar um protótipo capaz de se locomover em um ambiente qualquer é um trabalho importante, e pode ser utilizada em diversas aplicações como missões de salvamento em lugares perigosos, ambientes em chamas, usos industriais, dentre outros (NOGUEIRA, 2013).

2 OBJETIVOS

2.1 OBJETIVOS GERAIS

Desenvolver em FPGA um sistema de controle inteligente utilizando a lógica *fuzzy*, que permita a um carrinho se locomover em um ambiente qualquer, desviando de possíveis obstáculos sem auxílio externo. Utilizando para isso sensores ultrassónicos capazes de detectar objetos ao redor do veículo.

2.2 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo principal deste trabalho, foi necessário a realização de algumas tarefas, que são os objetivos específicos, listados a seguir:

- Entender o funcionamento de sensores ultrassônicos e seus métodos de utilização;
- Entender a lógica *fuzzy* e o uso deste conceito em sistemas de controle;
- Programar em VHDL um *hardware*, que seja capaz de processar os dados obtidos de sensores ultrassônicos, de modo que seja possível trabalhar com eles na FPGA;
- Programar um controlador *fuzzy* na FPGA, que tenha a capacidade de controlar um carrinho, através dos dados obtidos pelos sensores ultrassônicos.
- Realizar testes práticos com um protótipo controlado por este sistema.

2.3 JUSTIFICATIVA

Diversas empresas têm investido muito dinheiro no desenvolvimento de novas tecnologias, com o intuito de desenvolver veículos que possuam autonomia para transitar sem o auxílio de um motorista. Todas essas tecnologias estão ligadas de

alguma forma a inteligência artificial, por serem os sistemas mais próximos de imitar o comportamento humano e, portanto, substituí-lo (PEQUENINO, 2017).

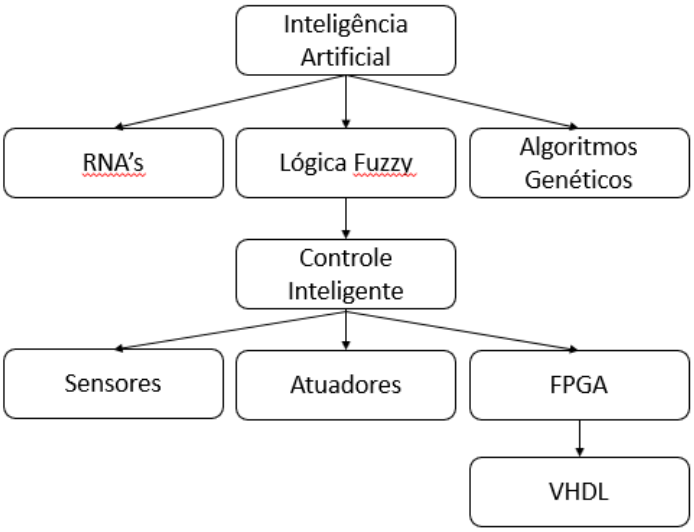
Existe no mercado uma ampla gama de dispositivos capazes de trabalhar com IA, e a FPGA é um deles. Apesar de ser uma tecnologia recente a IA já é bastante presente no cotidiano da vida moderna, a Cortana da Microsoft é uma IA que utiliza algoritmos e FPGA para o reconhecimento de fala (SHAH, 2016).

Assim sendo, o desenvolvimento de um sistema inteligente em FPGA para um protótipo de veículo autônomo, se torna um projeto viável e capaz de render bons resultados.

3 FUNDAMENTAÇÃO TEÓRICA

O diagrama da Figura 2 apresenta os conceitos necessários para o desenvolvimento do sistema de controle abordado neste trabalho, e que serão explicados nos próximos tópicos.

Figura 2 - Diagrama dos conceitos utilizados.



Fonte: Autoria própria.

3.1 SENSORES E ATUADORES

Todo sistema de controle é dependente de algum tipo de sensor ou atuador, de modo que o sensor traga informações para o controlador e o atuador execute alguma ação sobre a variável controlada através de comandos do controlador (WENDLING, 2010).

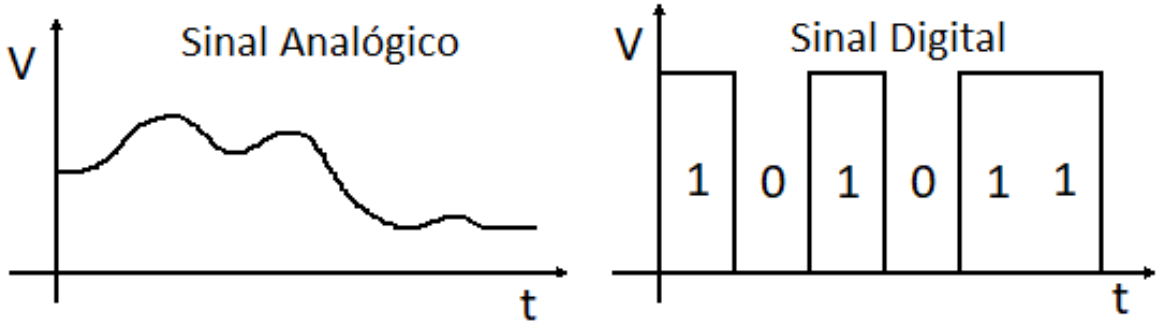
As variáveis controladas podem representar qualquer tipo de evento físico, tal como temperatura, pressão, vazão, etc. Para que o sistema possa interferir em alguma variável, existem diversos tipos de atuadores, eles podem ser válvulas, motores, solenoides, relés, entre outros (WENDLING, 2010).

Os sensores, por sua vez, são dispositivos sensíveis a algum tipo de energia do ambiente, tais como energia luminosa, térmica, sonora, etc. E baseado nisso

podem ser utilizados para detectar a presença de objetos, medir distâncias, temperaturas, dentre outras aplicações (THOMAZINI e ALBUQUERQUE, 2010).

A saída de um sensor pode ser analógica ou digital, e nem sempre estes sinais podem ser diretamente utilizados em um controlador, na maioria dos casos é necessário que haja o condicionamento deste sinal, para que ele seja adequado ao controlador utilizado, desde o nível de tensão até o tipo do sinal. Sensores analógicos apresentam sinais de saída com qualquer nível de tensão dentro da sua faixa de trabalho como mostra a Figura 3 (WENDLING, 2010).

Figura 3 - Representação dos sinais analógico e digital.



Fonte: Autoria Própria.

Diferentemente dos sensores digitais, que disponibilizam em sua saída apenas dois valores, que são os níveis lógicos alto “1” e baixo “0” conhecidos como bits, além disso podem ter várias saídas ou trabalhar com sequencias de bits em uma saída única para representar um número maior de valores, o que pode ser observado na Figura 3. Normalmente, os sinais digitais são obtidos através de algum circuito eletrônico que converte o sinal analógico (WENDLING, 2010).

3.1.1 SENSORES ULTRASSÔNICOS

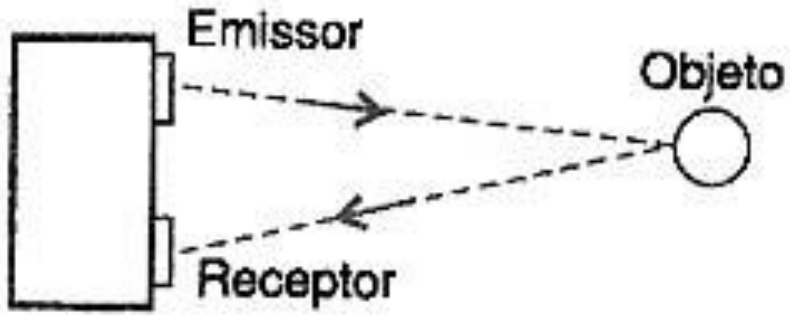
Os sinais ultrassônicos são sinais sonoros, porém com frequências bem elevadas, inaudíveis ao ouvido humano. Os sensores ultrassônicos contêm cristais de materiais piezelétricos que ressonam a certa frequência e transformam energia sonora em sinais elétricos e vice-versa (THOMAZINI e ALBUQUERQUE, 2010).

Sensores ultrassônicos possuem um transmissor e um receptor, como indicado na Figura 4. O transmissor é responsável por emitir as ondas ultrassônicas, e o receptor recebe estas mesmas ondas de forma direta ou refletida. Através do tempo de atraso entre o sinal enviado e recebido ou da ausência de recepção do sinal, o sensor é capaz de indicar a presença de objetos ou até mesmo a distância entre eles (THOMAZINI e ALBUQUERQUE, 2010).

Figura 4 - Sensor ultrassônico com emissor e receptor.

Fonte: THONSEM, 2015.

Figura 5 - Esquema de emissão e recepção de ondas ultrassônicas.



Fonte: BRAGA, 2017.

Conforme ilustrado na Figura 5, uma das aplicações dos sensores ultrassônicos é a detecção de objetos através da reflexão do sinal emitido. Essa é uma das melhores aplicações do sensor ultrassônico, pois o mesmo é capaz de identificar todos os tipos de objetos, inclusive os visualmente transparentes, o que não seria possível com um

sensor óptico infravermelho, por exemplo. Apesar dessa vantagem, o sensor trabalha com ondas sonoras e a sua precisão pode ser afetada por alguns fatores externos como a temperatura e a pressão (THOMAZINI e ALBUQUERQUE, 2010).

3.2 CONTROLE INTELIGENTE

Sistemas de controle comuns apresentam uma resposta devido à sua entrada de acordo com sua função de transferência. Já os sistemas de controle inteligentes apresentam como resposta soluções para os problemas encontrados, mesmo que sejam problemas novos ou inesperados, obtendo assim um tipo de comportamento que de certa forma é único (SIMÕES e SHAW, 2007).

Os sistemas inteligentes são capazes de copiar de certa forma alguns comportamentos dos seres humanos, tais como reconhecer padrões e tomar decisões. Para que seja possível fazer com que as máquinas tenham a capacidade de trabalhar de maneira parecida com o método humano de pensar, foram desenvolvidos vários modelos de inteligências artificiais, tais como a lógica *fuzzy*, as RNAs, os algoritmos genéticos, dentre outros (SIMÕES e SHAW, 2007).

3.2.1 INTELIGÊNCIA ARTIFICIAL

De acordo com Altero (2008), o termo inteligência artificial surgiu a partir de uma reunião em 1956 com cientistas de diversas áreas, cujo interesse era o desenvolvimento de máquinas inteligentes e a forma como os processos existentes poderiam ser afetados por elas.

O Instituto de Tecnologia de Massachusetts (MIT) propôs a divisão da história da IA em três partes, a primeira parte como época clássica que vai de 1956 até 1970, a segunda como época romântica de 1970 a 1980 e a terceira como época moderna que vai de 1980 a 1990 (ALTERO, 2008).

Durante a época clássica o objetivo era simular a inteligência humana. Já durante a época romântica o objetivo se tornou mais simples, simular o comportamento humano em situações restritas. E por último, na época moderna

deram uma atenção especial para as ferramentas de desenvolvimento de sistemas especialistas (ALTERO, 2008).

Com o passar dos anos, foram desenvolvidas diversas maneiras de programar uma IA, tais como as redes neurais artificiais, a lógica *fuzzy* e os algoritmos genéticos. Além disso, as IAs possuem diversas aplicações, dentre elas estão os jogos, o reconhecimento de padrões, o processamento de linguagem natural, a visão de computador, a prova de teoremas, os sistemas especialistas e as bases de dados inteligentes (ALTERO, 2008).

3.2.2 LÓGICA *FUZZY*

A lógica booleana, que se baseia somente em valores verdadeiros e falsos é bastante útil, porém, quando os problemas se tornam complexos, pode ser até impossível representa-los. Por causa disso, em 1965, Zadhe propôs a utilização de todos os valores existentes entre o estado verdadeiro e o falso, o que deu início a logica *fuzzy*, que utiliza os conceitos da lógica clássica, porém, com o acréscimo de operadores capazes de trabalhar com a infinidade de valores adotados neste novo modelo (ALTERO, 2008).

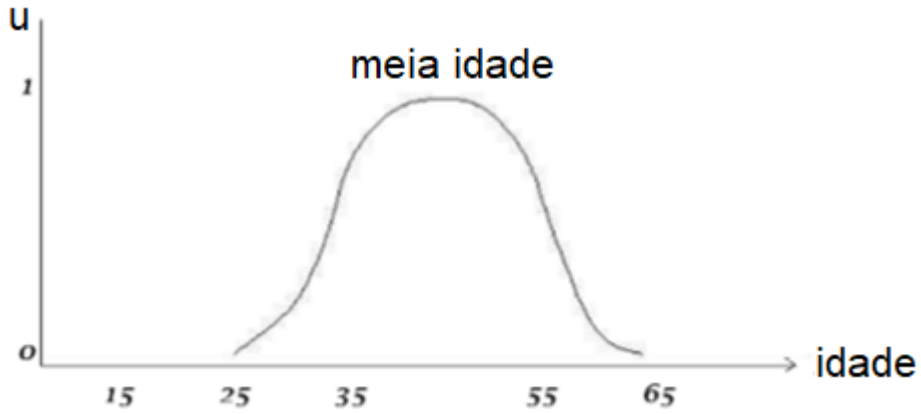
A lógica *fuzzy* não trabalha com valores absolutos, ela trabalha com variáveis linguísticas, que são variáveis que contêm conjuntos numéricos, como por exemplo, a variável velocidade que pode conter os conjuntos “baixa” onde se encontram os valores considerados baixos, “média” que engloba os valores médios e “alta” contendo os valores altos, esses conjuntos recebem o nome de conjuntos nebulosos ou conjuntos *fuzzy* (SIMÕES e SHAW, 2007).

Os conjuntos *fuzzy* transformam valores específicos de alguma variável em um grau de porcentagem conhecido como grau de pertinência, que representa o quanto aquele valor pertence ao conjunto associado a ele. Voltando ao exemplo anterior sobre velocidade, a velocidade de quarenta quilômetros por hora, pode ter um grau de pertinência de 70% no conjunto “baixa” e 30% no conjunto “média”, sendo assim um valor pode pertencer a mais de um conjunto desde que a soma dos graus de pertinência seja próxima ou igual a 100% (SIMÕES e SHAW, 2007).

Os conjuntos *fuzzy* podem ser representados graficamente através das funções de pertinência que os representam, conforme apresentado na Figura 6 com o exemplo

da idade. É importante salientar que com a lógica *fuzzy* é possível trabalhar com sistemas não lineares (SIMÕES e SHAW, 2007).

Figura 6 - Exemplo de gráfico de um conjunto *fuzzy*.



Fonte: RIGNEL; CHENCI; LUCAS, 2011.

Além de tudo o que foi dito, a lógica *fuzzy* é muito útil para sistemas de controle inteligente, pois a forma como ela trabalha os dados é muito parecida com a forma com que os humanos executam suas tarefas cotidianas, utilizando palavras ao invés de números, além de permitir a criação de sistemas de controle que possuam múltiplas entradas e múltiplas saídas (SIMÕES e SHAW, 2007).

3.2.2.1 FUZZIFICAÇÃO

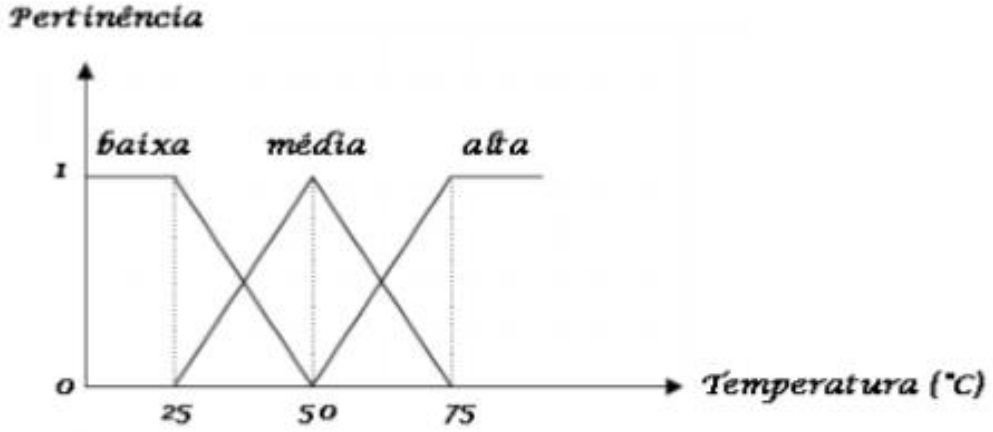
O ato de transformar os números concretos em valores imprecisos ou vagos, como são os graus de pertinência, é conhecido por fuzzificação. O cérebro faz isso de forma natural, pois é comum para um humano olhar para um copo, por exemplo, e determinar se ele está meio cheio ou meio vazio (SIMÕES e SHAW, 2007).

A fuzzificação é feita levando em conta a experiência de usuários ou o conhecimento de um especialista acerca da variável que está sendo fuzzificada. Nessa etapa que se determinam as variáveis linguísticas e seus conjuntos, e aquela experiência ou conhecimento é utilizado para determinar o que pode ser considerado um valor médio, baixo ou alto (SIMÕES e SHAW, 2007).

A Figura 7, apresenta um exemplo de fuzzificação da variável temperatura, onde os valores de temperatura são transformados em conjuntos numéricos que

contem valores que vão de 0 a 1, que são os graus de pertinência de cada valor de temperatura (ALTERO, 2008).

Figura 7 - Exemplo de fuzzyficação de temperatura.



Fonte: SIMÕES; SHAW, 2007.

Conforme mostrado no exemplo da Figura 7, temperaturas abaixo de 25 graus pertencem 100% ao conjunto “baixa”, temperaturas acima de 75 graus pertencem 100% ao conjunto “alta” e temperaturas intermediárias têm graus de pertinência variáveis de 0% a 100% dentro dos conjuntos “baixa”, “média” e “alta”, e são com estes graus de pertinência que os controladores *fuzzy* vão trabalhar (SIMÕES e SHAW, 2007).

3.2.2.2 OPERAÇÕES ENTRE CONJUNTOS

Existem diversas formas de trabalhar com os conjuntos *fuzzy*, as operações mais básicas são a negação (Lógica *not*) descrita na expressão (1), a união (lógica *or*) cujo comportamento está descrito na expressão (2) e a intersecção (lógica *and*) dada pela expressão (3) (ALTERO, 2008).

$$\neg\mu_A(x) = 1 - \mu_A(x) \tag{1}$$

$$\mu_{A+B}(x) = \max\{\mu_A(x), \mu_B(x)\} \tag{2}$$

$$\mu_{A.B}(x) = \text{mín}\{\mu_A(x), \mu_B(x)\} \tag{3}$$

As equações (1), (2) e (3), apresentam, como exemplo, conjuntos genéricos “A” e “B”, nos quais o “x” é valor concreto da variável, o “μ” é referente ao grau de pertinência da variável, o “mín” e “máx” são, respectivamente, os valores mínimos e máximos dentro das chaves e o símbolo “¬" indica a negação do conjunto.

Além destes, existem alguns operadores bem mais complexos que fogem um pouco da aplicação para o projeto. As equações apresentadas representam as operações básicas que o controlador irá realizar com os conjuntos *fuzzy* a fim de executar as ações necessárias através de regras de inferência, que são regras que determinam os valores de saída com base nas situações de entrada. Essas regras também devem ser criadas por meio da experiência ou do conhecimento de algum profissional da área (ALTERO, 2008).

3.2.2.3 DEFUZZIFICAÇÃO

A defuzzificação é o oposto da fuzzificação, enquanto uma é responsável por transformar valores concretos em valores nebulosos dentro de conjuntos *fuzzy*, a outra é responsável por transformar esses mesmos valores nebulosos em valores concretos (ALTERO, 2008).

Assim como acontece com a fuzzificação, na defuzzificação a experiência ou o conhecimento de algum profissional também se faz necessário, pois após todos os tratamentos feitos pelas regras de inferência, a saída que também é composta por conjuntos com valores válidos em mais de um conjunto. E é na defuzzificação que se converterão esses valores em valores reais (SIMÕES e SHAW, 2007).

Existem diversos métodos de defuzzificação, os mais utilizados são os métodos do Centro-da-Área (C-o-A) descrito na equação (4), do Centro-do-Máximo (C-o-M) descrito na equação (5) e o método da Média-do-Máximo (M-o-M) descrito na equação (6) (SIMÕES e SHAW, 2007).

$$\mathrm{u} = \frac{\sum_{\mathrm{j}=1}^{\mathrm{N}} \mathrm{u_i}\mu_{\mathrm{out}}(\mathrm{u_i})}{\sum_{\mathrm{j}=1}^{\mathrm{N}} \mu_{\mathrm{out}}(\mathrm{u_i})} \tag{4}$$

$$\mathrm{u} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{u_i}\cdot\sum_{\mathrm{k}=1}^{\mathrm{N}} \mu_{\mathrm{out}}(\mathrm{u_i})}{\sum_{\mathrm{i}=1}^{\mathrm{N}} \sum_{\mathrm{k}=1}^{\mathrm{N}} \mu_{\mathrm{out}}(\mathrm{u_i})} \tag{5}$$

$$\mathrm{u} = \sum_{m=1}^{M} \frac{u_m}{M} \tag{6}$$

Nas equações (4), (5) e (6), a variável “u” refere-se a um valor concreto da saída, enquanto o “μ” consiste em um grau de pertinência de um conjunto de saída, ou seja, “u” representaria o eixo “x” e “μ” o eixo “y” do gráfico de saída. Todos esses métodos são baseados em cálculos de médias para determinar o valor real da saída, já que os conjuntos de saída na maioria das vezes vão possuir mais de um valor inferido a eles (SIMÕES e SHAW, 2007).

3.2.2.4 REGRAS DE INFERÊNCIA

Um Sistema de Inferência *fuzzy* é geralmente representado na forma de termos antecedentes e consequentes utilizados de forma condicional como mostrado a seguir:

- SE <antecedente> ENTÃO <consequente>

Através do uso das operações nebulosas no antecedente, pode-se determinar a valor de inferência referente aos conjuntos na saída.

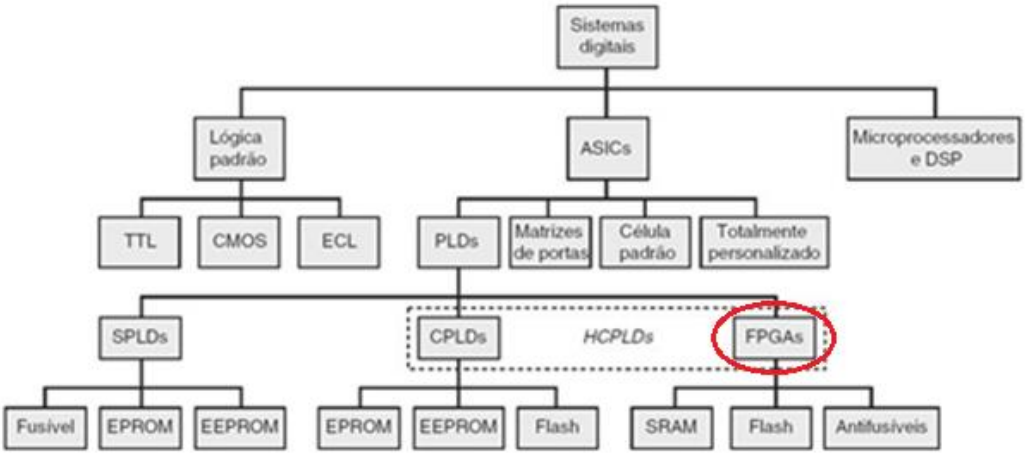
O antecedente é formado por uma ou mais entrada linguística e um conjunto *fuzzy* desta mesma variável, que apresenta um valor de inferência, é através da aplicação dos operadores nesses antecedentes que se determina o valor de inferência nos conjuntos de saída (MINUSSI, 2009).

O consequente é formado por uma ou mais variáveis linguísticas de saída e um conjunto *fuzzy* dessa variável, os consequentes recebem um valor de inferência de acordo com a manipulação dos antecedentes, todas as regras são determinadas simultaneamente, e é importante existir um número suficiente de regras para que o sistema possa reagir a todas as possibilidades do sistema (MINUSSI, 2009).

3.3 FPGA

Os sistemas digitais englobam diversas tecnologias, tais como os Circuitos Integrados de Aplicação Específica (ASICs), onde se situam os PLDs. Os CPLDs junto das FPGAs fazem parte do grupo dos PLDs, como mostra o diagrama da Figura 8.

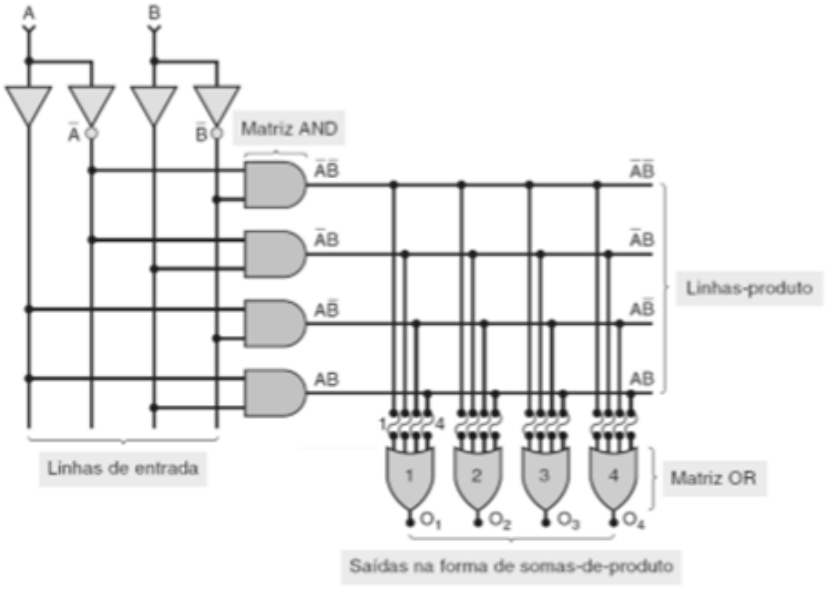
Figura 8 - Diagrama de árvore dos sistemas digitais.



Fonte: TOCCI, 2011.

O funcionamento dos PLDs é baseado na capacidade dos transistores de funcionar como chave, com isso é possível utilizar matrizes de portas lógicas para gerar diversos tipos de saídas alterando apenas os estados dos transistores. Esse tipo de arquitetura é mostrado na Figura 9 (TOCCI, 2011).

Figura 9 - Exemplo de arquitetura de um PLD.



Fonte: TOCCI, 2011.

Por ser um tipo de PLD, a FPGA se baseia no mesmo princípio de funcionamento e por este motivo possui todas as vantagens dos PLDs, ou seja, permitem o desenvolvimento de um *hardware* embutido, capaz de realizar as funções

de diversos circuitos integrados (CI), ocupa menos espaço em placas, tem um consumo menor, e geralmente um menor custo de fabricação (TOCCI, 2011).

Os PLDs podem ser voláteis e não voláteis, o que significa que é possível regravar várias vezes um PLD volátil, e que é possível gravar apenas uma vez em um PLD não volátil. Ambos os tipos têm as suas vantagens e desvantagens, contudo as FPGAs em sua maioria são voláteis (TOCCI, 2011).

3.3.1 VHDL

VHDL (*VHSIC* Linguagem de descrição de *Hardware*) é uma linguagem, através da qual é possível programar FPGAs e CPLDs independentemente de suas marcas e fabricantes. Ela foi a primeira linguagem de *hardware* a ser padronizada pelo IEEE, e surgiu nos anos oitenta como resultado de uma iniciativa do Departamento de Defesa dos Estados Unidos (PEDRONI, 2010).

O código em VHDL serve para descrever estruturas e comportamentos, que serão atribuídos a algum meio físico, como as FPGAs, por exemplo, através de um compilador. Os códigos em VHDL também podem ser simulados, a partir de diversas ferramentas que os fabricantes fornecem de forma gratuita, permitindo além da simulação, também a síntese de códigos em VHDL como é o caso do Quartus da Altera (PEDRONI, 2010).

4 METODOLOGIA

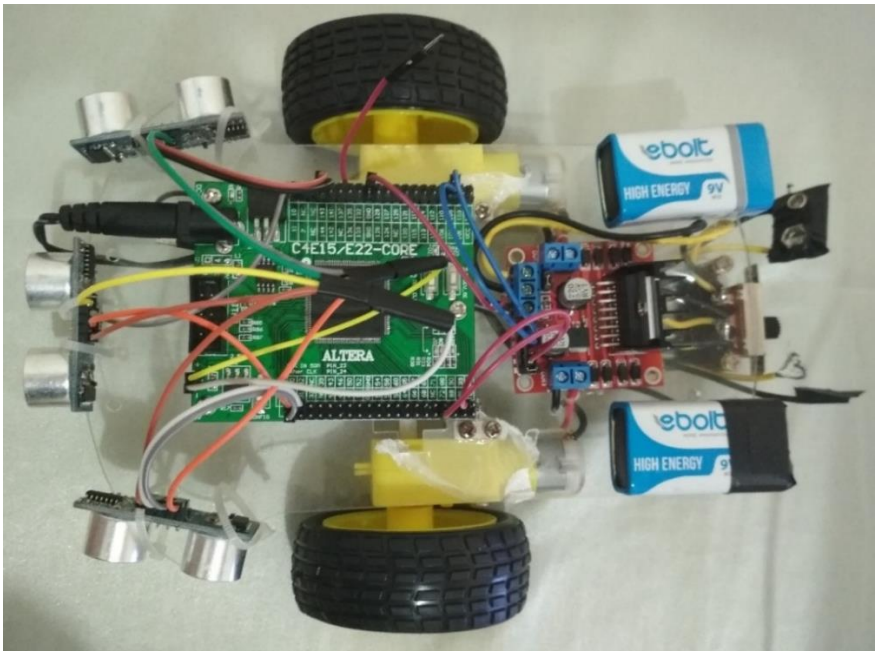
Este trabalho teve a finalidade de desenvolver um sistema de controle inteligente. Em particular, foi utilizado um controlador baseado em lógica fuzzy, pois é uma técnica de controle inteligente que não exige algoritmos complexos e nem modelagem matemática dos sistemas.

Para este trabalho s utilizados os itens listados abaixo:

- 3 sensores ultrassonicos HC-SR04;
- 1 ponte H dupla L298n;
- 2 motores DC;
- 1 placa com chip FPGA ALTERA cyclone IV.
- Chassis de acrílico com duas rodas;
- Motores de corrente contínua
- Baterias de alimentação.

Todos estes itens foram reunidos para a construção do protótipo como mostra a Figura 10.

Figura 10 – Protótipo de carro utilizado como modelo.



Fonte: Autoria própria.

4.1 DIAGRAMAS DE BLOCOS DO SISTEMA

Uma demonstração básica do que foi realizado é representado na Figura 11.

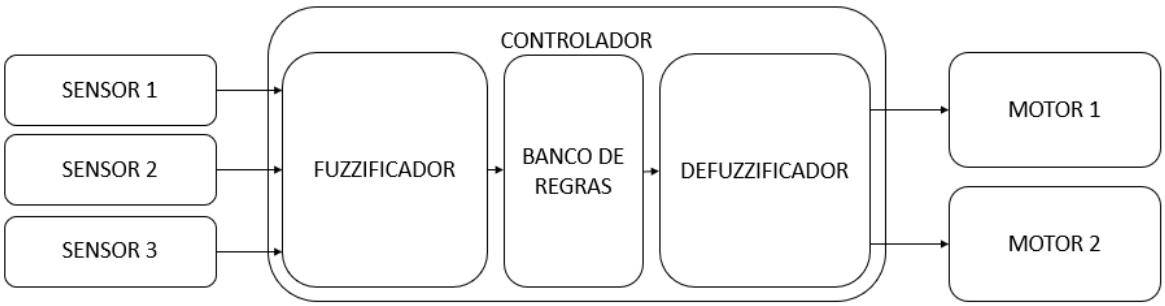
Figura 11 - Esquema básico do sistema de controle

Fonte: Autoria própria.

O diagrama básico mostra que o controlador vai comandar os motores baseado nos dados colhidos dos sensores.

A forma como o controlador executa as suas funções de controle é mostrada na Figura 12.

Figura 12 – Esquema do controlador.



Fonte: Autoria própria.

O funcionamento do controlador apresentado na Figura 12 é descrito a seguir:

1. Os sensores medem a distância dos objetos presentes no ambiente enquanto o protótipo se move;

2. Os dados dos sensores são lidos pelo controlador que se divide em três partes principais, fuzzificador, banco de regras e deffuzificador;

3. O fuzzificador transforma os valores de entrada em variáveis linguísticas com alguns conjuntos *fuzzy*;

4. Os valores fuzzificados são submetidos a um conjunto de regras, onde se executam algumas operações entre os conjuntos *fuzzy* para gerar os valores de saída fuzzificados;

5. Os valores de saída fuzzificados vão para o defuzificador, onde serão transformados novamente em valores reais, neste caso específico um número entre -100% e 100%, que indica qual a velocidade de cada um dos motores e o sentido de giro.

Baseado nestes conceitos, o controlador contará com 3 variáveis de entrada e 2 duas de saídas, essa é uma grande vantagem dos controladores *fuzzy*, pois eles podem trabalhar com diversas entradas e saídas.

4.2 FPGA

Neste trabalho foi utilizado como plataforma do projeto, uma placa de desenvolvimento com um chip FPGA EP4CE22E22C8N da altera como mostra a Figura 13.

Figura 13 - Placa FPGA utilizada.

Fonte: Autoria própria.

As especificações deste chip podem ser obtidas através do próprio Quartus, e podem ser observadas na Figura 14.

Figura 14 - Características do chip FPGA.

Family	Cyclone IV E
Device	EP4CE22E22C8
Timing Models	Final
Total logic elements	0 / 22,320 (0 %)
Total combinational functions	0 / 22,320 (0 %)
Dedicated logic registers	0 / 22,320 (0 %)
Total registers	0
Total pins	0 / 80 (0 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

Fonte: Autoria própria.

Além destas características, existem as características da placa que serão listadas a seguir:

- Entrada para alimentação de 5 V;
- Terminais de entrada e saída;
- 1 oscilador de 50 Mhz;
- Terminais para alimentação de dispositivos externos de até 5 V;
- Botão de *reset*.

Esta placa apresenta tudo o que é necessário para a elaboração deste projeto, e para a realização dos testes que serão executados posteriormente.

4.3 MOTORES CC

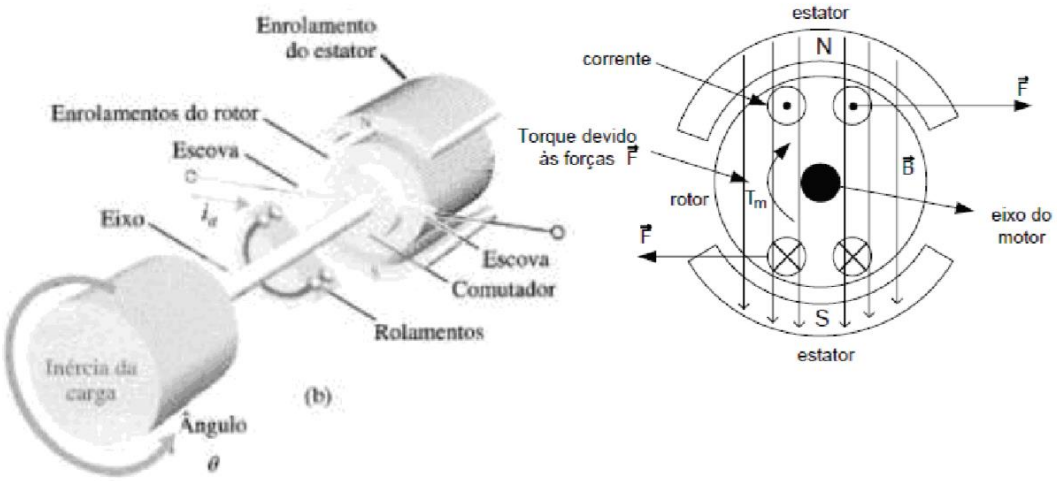
O chassis utilizado neste projeto vem com um conjunto de rodas e motores acoplados, estes motores são motores CC (corrente contínua) de 12 V que podem ser facilmente controlados por meio da variação de sua tensão média de alimentação.

O funcionamento deste motor se dá por meio da indução e da repulsão eletromagnética e ele pode ser dividido em três partes principais, rotor, estator e comutador (NOGUEIRA, 2013).

- Rotor: é a parte central que gira, e contém algumas bobinas;
- Estator: é a parte estática do motor, que pode conter imãs permanentes ou bobinas;
- Comutador: é a conexão através de escovas com o rotor, responsáveis por alimentar as bobinas do rotor e manter a polarização correta das mesmas para que o motor gire continuamente.

As partes do motor podem ser visualizadas na Figura 15.

Figura 15 - Partes internas de um motor CC.

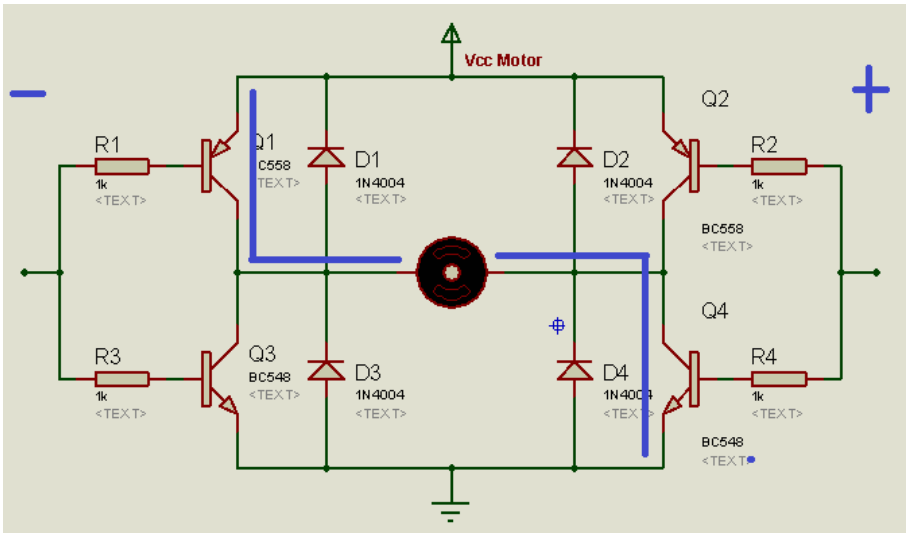


Fonte: NOGUEIRA, 2013.

O acionamento de um motor CC pode ser feito de várias formas, porém para poder realizar este acionamento nos dois sentidos de giro é preciso utilizar um método de acionamento que proporcione isso, como é o caso da ponte H (BERNARDO, 2016).

O esquema da ponte H pode ser visto na Figura 16.

Figura 16 - Esquemático de uma ponte H com transistores, sentido A.

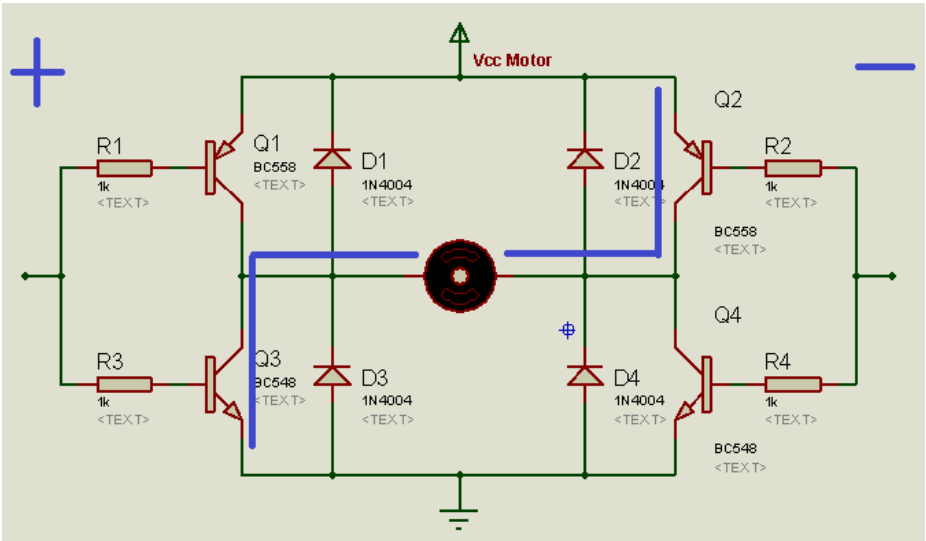


Fonte: BERNARDO, 2016.

O acionamento do motor em um sentido de giro se dá através do chaveamento de dois transistores de forma simultânea, que alimentam o motor com uma polarização como visto na Figura 16.

A mudança do sentido de giro se dá através do chaveamento de outros dois transistores que acabam invertendo a polarização do motor fazendo com que o mesmo gire no sentido contrário, como se observa na Figura 17. (BERNARDO, 2016).

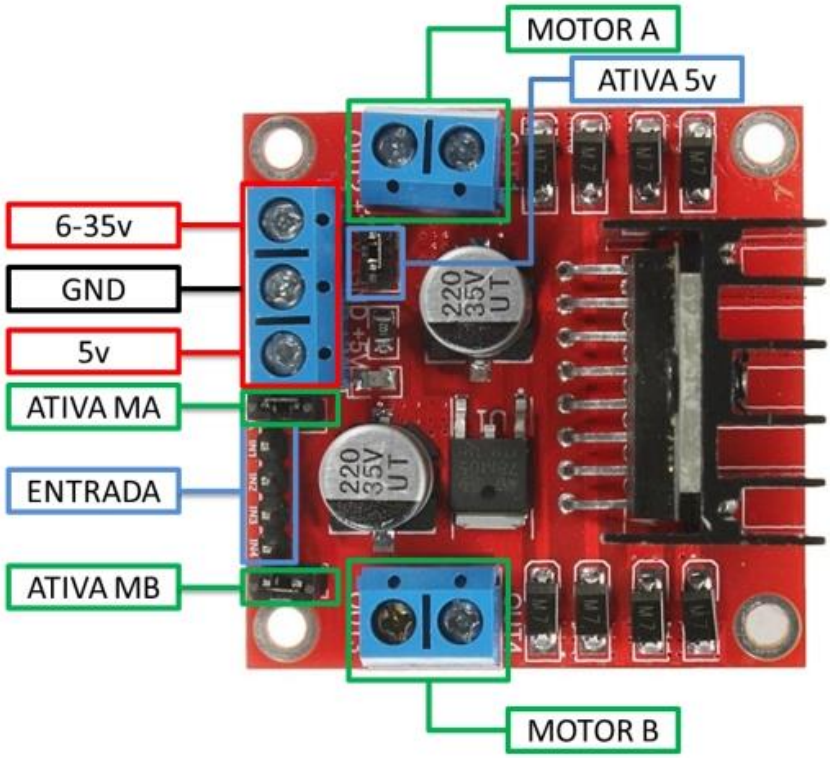
Figura 17 - Esquemático de uma ponte H com transistores, sentido B.



Fonte: BERNARDO, 2016.

Como este projeto conta com dois motores, se fez necessário o uso de duas pontes H, por isso o uso do driver de acionamento com dispositivo integrado L298 mostrado na Figura 18, pois o mesmo possui duas pontes.

Figura 18 – Driver de acionamento de motores.



Fonte: THOMSEN, 2013.

Como mostra a Figura 18, este dispositivo possui algumas características importantes que estão listadas a seguir:

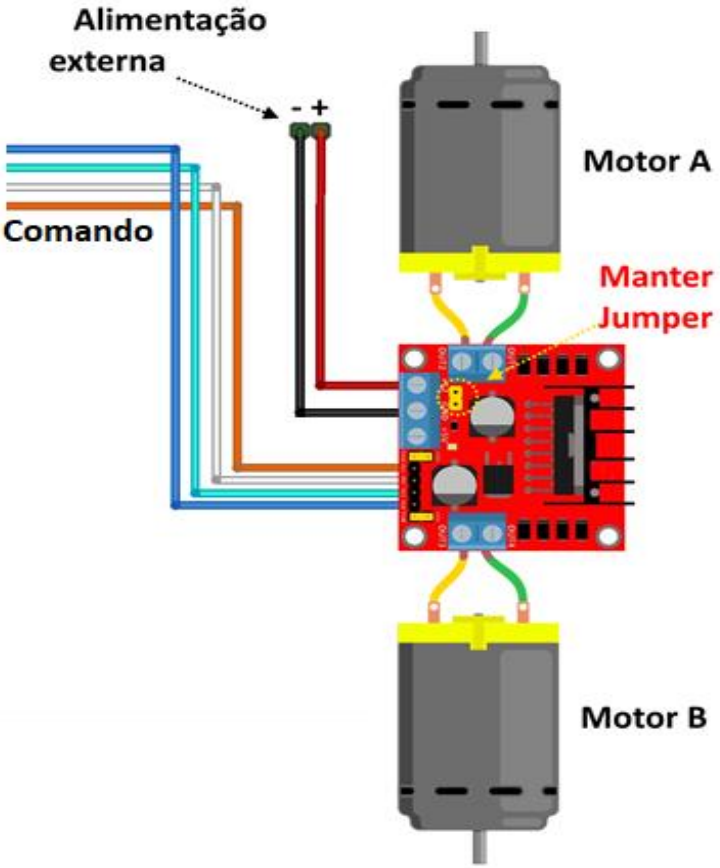
- Entrada para alimentação entre 6 V e 35 V;
- Terminal de 5 V que pode funcionar como entrada ou saída para alimentação de dispositivos externos;
- Saídas de alimentação para dois motores;
- Duas entradas de comando para cada motor, uma para cada sentido de rotação, totalizando 4 entradas de comando;

É importante salientar que para que o motor gire no sentido desejado é preciso acionar um de seus comandos por vez, se por acaso forem acionados os dois

comandos dos dois sentidos de giro ao mesmo tempo, o motor permanecerá parado (THOMSEN, 2013).

A maneira correta de ligação deste *driver* de acionamento é apresentada na Figura 19.

Figura 19 - Esquema de ligação do driver.

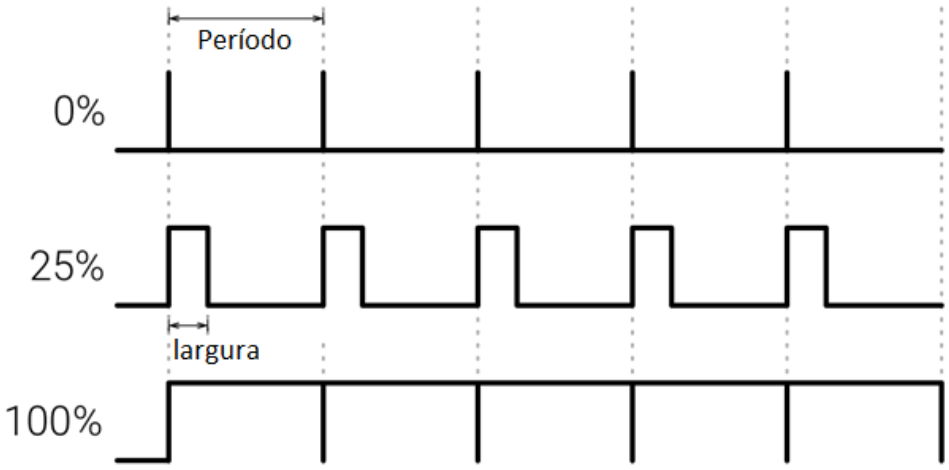


Fonte: Adaptado de THOMSEN, 2013.

Somente acionar os motores não é o bastante para um sistema de controle, é preciso de alguma forma poder variar a velocidade deles para que o controlador possua algum tipo de controle sobre os mesmos (NOGUEIRA, 2013).

O controle da velocidade de um motor CC pode ser feito de maneira simples utilizando a técnica de Modulação por largura de pulso ou PWM, que consiste em variar a tensão média do motor por meio da variação da largura de um pulso enviado com uma frequência constante como mostra a Figura 20 (HENRY, 2018).

Figura 20 - representação de um PWM.



Fonte: HENRY, 2018.

Algumas das características mais importes do uso do PWM serão listados a seguir:

- PERÍODO: É determinado pela frequência com que os pulsos serão enviados, no caso deste projeto em específico será utilizada uma frequência de 1 Khz;
- LARGURA: É a largura do pulso que será enviado, pode variar entre 0% e 100%, e determina qual a porcentagem de tensão será entregue a carga, uma vez que ele altera a tensão média do sistema de forma linear.

Com todas essas informações já é possível realizar os testes com os motores, o driver e a placa FPGA como será mostrado mais à frente.

4.4 SENSORES

O sensor utilizado neste trabalho é o sensor ultrassónico HC-SR04, por se tratar de um dispositivo barato, que consome pouca energia, e é de fácil utilização por ser alimentado com 5 V que está disponível no *driver* do motor e até mesmo na placa da FPGA (THONSEM, 2015).

A Figura 21, apresenta o modelo de sensor utilizado.

Figura 21 - Sensor HC-SR04.

Fonte: THONSEM, 2015.

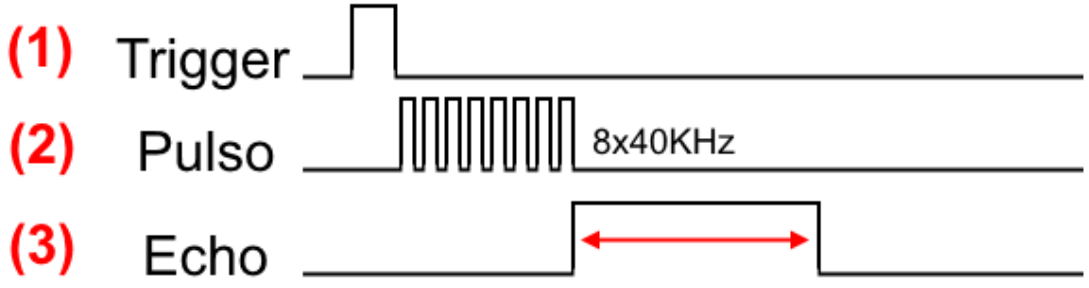
Como mostra a Figura 21 este sensor possui apenas 4 terminais, cujas funções são as seguintes:

- VCC: Tensão de alimentação 5V;
- GND: Terra da alimentação;
- TRIG: Terminal de disparo da medição;
- ECHO: Terminal de leitura da distância.

Esse sensor utiliza sinais ultrassônicos de 40 Khz para medir a distância entre o sensor e algum objeto. Com base na velocidade do som ele é capaz de medir distâncias que vão de 2 cm a 4 m (THONSEM, 2015).

O funcionamento deste sensor é extremamente simples, ele funciona basicamente em três passos que podem ser vistos na Figura 22.

Figura 22 - Diagrama de funcionamento do sensor.



Fonte: THONSEM, 2015.

A Figura 22 apresenta os passos de funcionamento do sensor que estão explicados abaixo:

1. Um pulso de 10 us (microsegundos) é enviado ao terminal *trigger*, o que indica o início do ciclo de medição.
2. Automaticamente após o pulso, o módulo envia 8 pulsos de 40 KHz e aguarda o retorno deste sinal.
3. Caso haja um retorno de sinal o terminal Echo ficara em nível lógico pelo tempo que levou até o retorno do sinal, o tempo máximo de espera é de 38 ms (milisegundos), após isso ele espera por outro disparo.

Com essas informações é possível calcular a distância entre o sensor e o objeto a frente através da equação (7), onde “D” é a distância, “T” é o tempo em que Echo fica em nível lógico alto, e “V” é a velocidade do som (THONSEM, 2015).

$$D = \frac{\mathrm{T}*\mathrm{V}}{2} \tag{7}$$

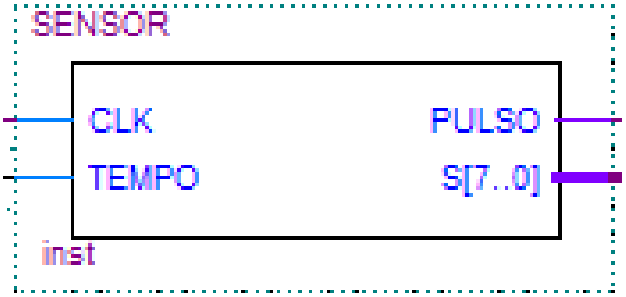
4.5 PROCEDIMENTO DE TESTES

Antes do projeto do controlador é necessário realizar os testes com os dispositivos periféricos, para que os mesmos estejam preparados para serem utilizados em conjunto com a placa FPGA.

4.5.1 TESTE DO SENSOR

Para a utilização dos sensores foi desenvolvido um bloco em VHDL capaz de emitir um pulso de 10 us e de ler o tempo cujo o terminal Echo fica em nível lógico alto, e que através de contadores é capaz de determinar a distância de objetos em centímetros, este bloco é mostrado na Figura 23.

Figura 23 - Bloco do sensor.



Fonte: Autoria própria.

O bloco do sensor pode ser utilizado para os três sensores pois os mesmos são iguais. Este bloco possui duas entradas, e duas saídas, com as funções definidas da seguinte forma:

- CLK: Entrada de clock de 50 Mhz;
- TEMPO: Entrada ligada ao terminal Echo do sensor;
- PULSO: Saída responsável por dar um pulso de 10 us ao terminal TRIG do sensor;
- S: É a saída do bloco que indica a distância dos objetos em centímetros, para essa aplicação ela possui 8 bits e pode varias de 0 a 255.

Para realizar a medida utiliza-se três contadores, um contador para dar o pulso de 10 us, outro contador que conta o tempo em que o terminal echo ficou em nível lógico alto e o terceiro serve para delimitar o limite de 38 ms para o início da próxima medida.

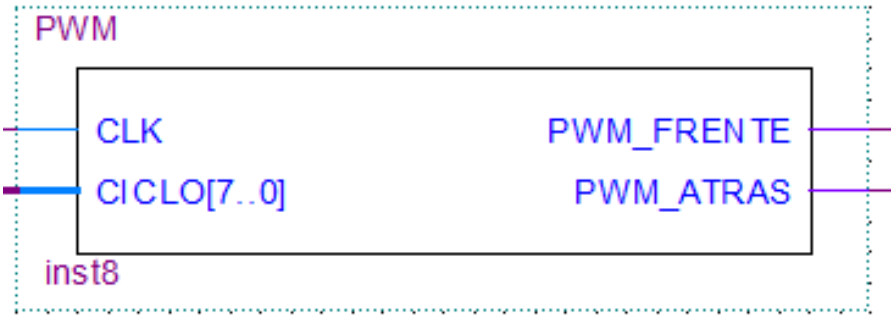
O contador de tempo é regulado uma frequência aproximada de 17 Khz, , isso por que a velocidade do som é de aproximadamente 340 m/s, e utiliza-se metade dela como mostrou a equação (7) anteriormente, pois o som percorre a distância de ida e volta, assim tem-se uma velocidade de aproximadamente 170 m/s, que equivale a 17000 cm/s, assim os 17 Khz do contador irão contar 1 cm a cada ciclo de clock.

4.5.2 TESTE DO DRIVER E DOS MOTORES

Para o teste dos motores e do *driver* foram necessários um bloco PWM desenvolvido em VHDL e uma bateria para alimentação dos motores.

A Figura 24 apresenta o bloco PWM, que contém duas entradas, que são CICLO e CLK, e duas saída que se chamam PWM_FRENTE e PWM_ATRAS, que são responsáveis por girar o motor nos dois sentidos, as duas nunca são usadas simultaneamente. Neste bloco CICLO é uma entrada inteira limitada entre -100 e 100, representando a porcentagem de largura de pulso.

Figura 24 - Bloco PWM.



Fonte: Autoria Própria.

Este bloco pode ser utilizado para os dois motores pois os mesmos são iguais. Este bloco possui duas entradas, e duas saídas, com as funções definidas da seguinte forma:

- CLK: Entrada de clock de 50 Mhz;
- CICLO: É a largura do pulso, que neste projeto pode assumir valores inteiros entre -100 e 100, onde valores negativos indicam o sentido de giro inverso;
- PWM_FRENTE: Saída definida para que o motor leve o protótipo para frente;
- PWM_ATRAS: Saída definida para que o motor leve o protótipo para trás.

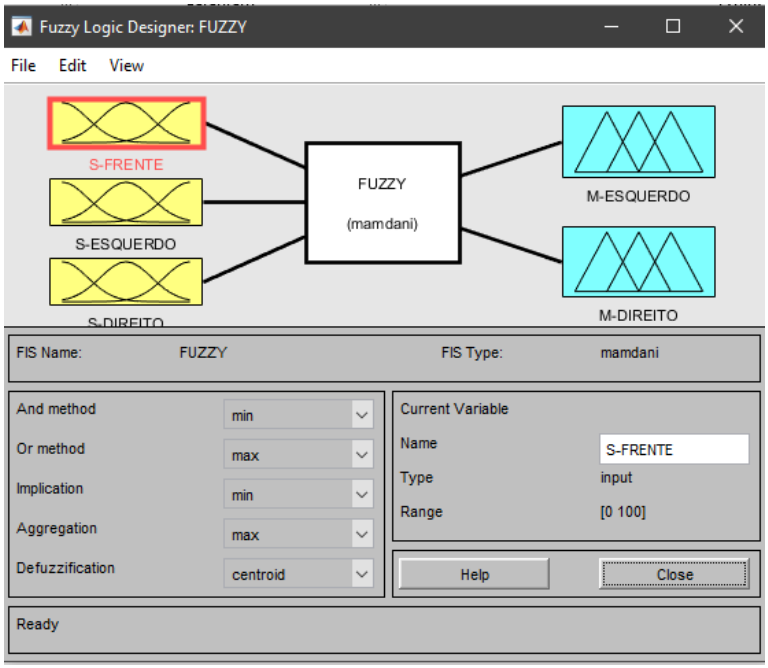
Neste bloco utiliza-se um contador e um timer, o contador serve para determinar o início e o fim de um pulso do PWM, e o timer define qual o tamanho do pulso. Como a FPGA trabalha com clock de 50 Mhz, e a frequência definida para o PWM neste projeto foi de 1 Khz é preciso que o ciclo tenha um tamanho de 50000, ou seja, a cada 50000 cilcos de clock de 50 Mhz o pulso do PWM é reiniciado.

Como a saída do controlador terá um valor limitado entre -100 e 100 e o tamanho do pulso do PWM é de 50000 ciclos, é preciso realizar uma normalização desse valor para que atenda aos 50000, por isso a saída do controlador é multiplicada por 500 dentro do bloco.

4.6 PROJETO DO CONTROLADOR NO MATLAB

Existem diversas formas de desenvolver controladores *fuzzy*, porém, neste trabalho foi utilizada uma ferramenta do *software* MATLAB que pode ser visualizada na Figura 25, o controlador desenvolvido nesta ferramenta será utilizada como base para a criação do controlador em VHDL.

Figura 25 - Ferramenta *fuzzy* do MATLAB.



Fonte: Autoria própria.

Utilizando essa ferramenta é possível determinar todas as características do sistema *fuzzy*, dentre estas características as principais são:

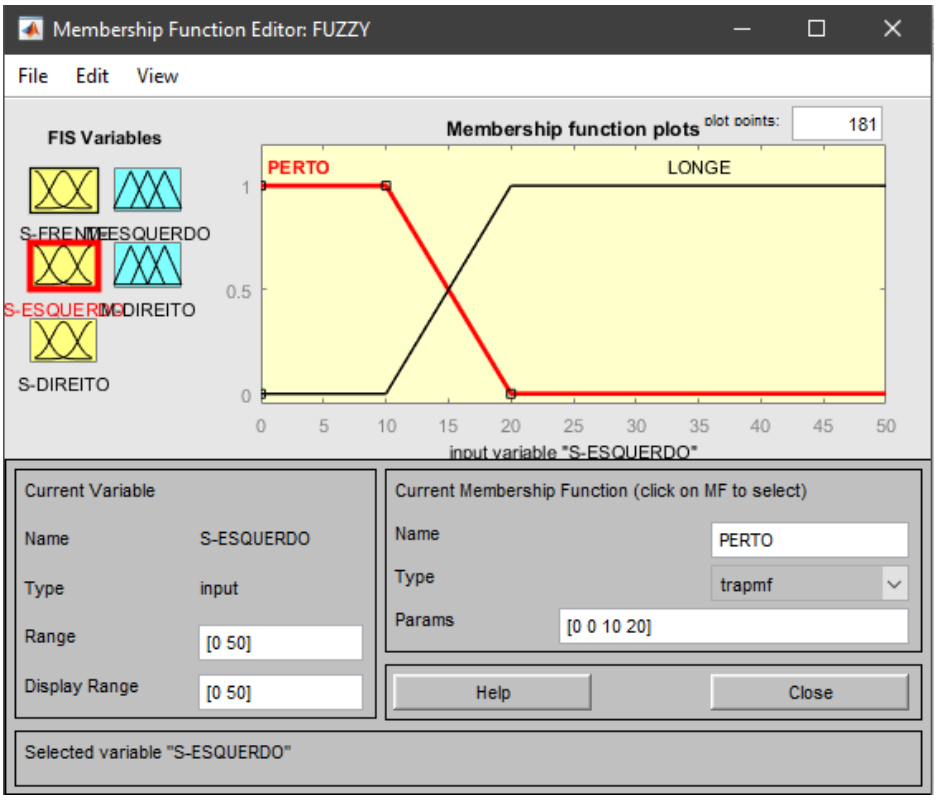
1. Variáveis de entrada e de saída;
2. Conjuntos das variáveis de entrada e de saída
3. Método de defuzzificação.

Para este projeto é necessário o uso das seguintes variáveis:

- S-FRENTE: Sensor que mede a distância de objetos a frente;
- S-ESQUERDO: Sensor que mede a distância de objetos a esquerda;
- S-DIREITO: Sensor que mede a distância de objetos a direita;
- M-ESQUERDO: Motor da esquerda;
- M-DIREITO: motor da direita.

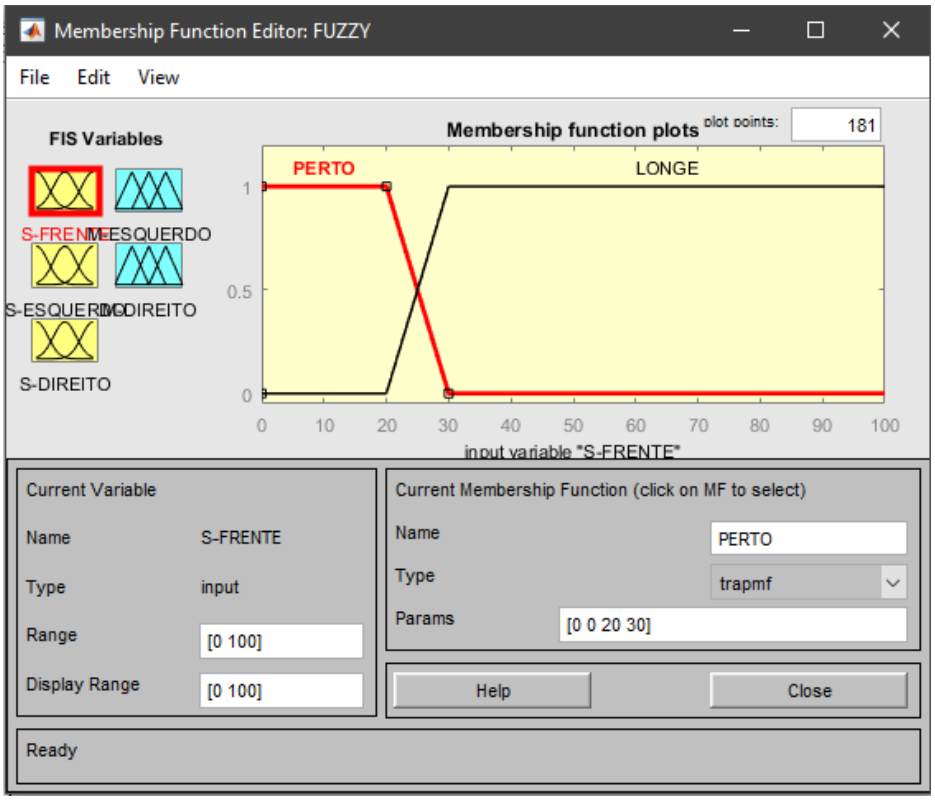
Para todos os sensores foram determinados dois conjuntos, que são os conjuntos PERTO e LONGE. Para os sensores laterais estes conjuntos são iguais, pois as suas funções são as mesmas, já para os conjuntos do sensor frontal foram utilizadas distâncias um pouco maiores como pode ser visto comparando a Figura 26 com a Figura 27.

Figura 26 - Conjuntos dos sensores laterais.



Fonte: Autoria própria.

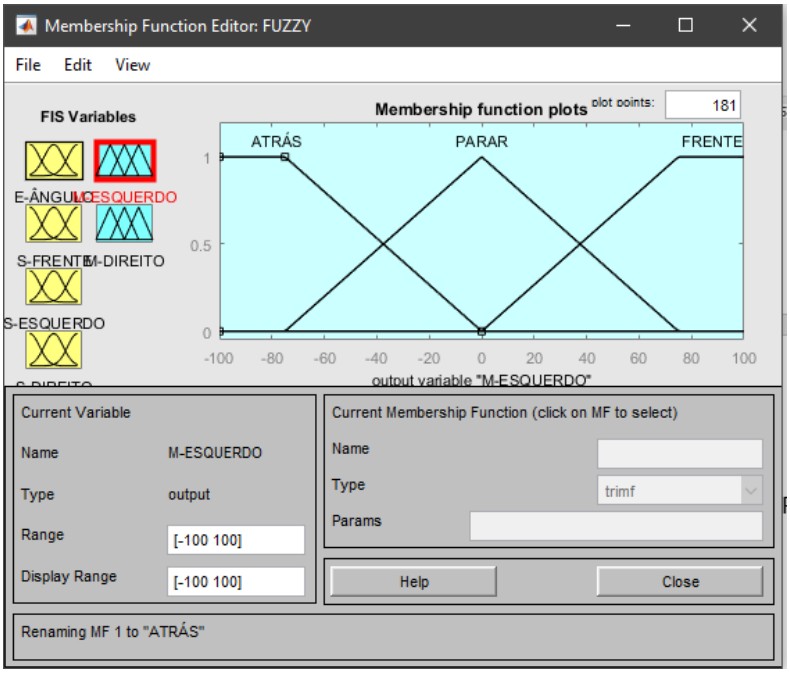
Figura 27 - Conjuntos do sensor frontal.



Fonte: Autoria Própria.

As variáveis de saída são idênticas e apresentam três conjuntos que são ATRÁS, PARAR e FRENTE, determinados como mostra a Figura 28

Figura 28 - Conjuntos das variáveis de saída.



Fonte: Autoria própria.

Além das variáveis e dos conjuntos, também é necessário determinar as regras do sistema, para este sistema com três variáveis e nove conjuntos de entrada, existe uma combinação de 8 regras possíveis, sempre utilizando o operador de intersecção, que é o mais utilizado, porém quando se cria as 8 regras, existem repetições de saída como mostra a Tabela 1.

Tabela 1 - Regras do sistema.

	VARIÁVEIS DE ENTRADA			VERIÁVEIS DE SAÍDA	
REGRA	FRENTE	ESQUERDA	DIREITA	MOTOR ESQUERDO	MOTOR DIREITO
1	PERTO	PERTO	PERTO	FRENTE	ATRÁS
2	PERTO	PERTO	LONGE	FRENTE	ATRÁS
3	PERTO	LONGE	PERTO	ATRAS	FRENTE
4	PERTO	LONGE	LONGE	FRENTE	ATRÁS
5	LONGE	PERTO	PERTO	FRENTE	FRENTE
6	LONGE	PERTO	LONGE	FRENTE	PARAR
7	LONGE	LONGE	PERTO	PARAR	FRENTE
8	LONGE	LONGE	LONGE	FRENTE	FRENTE

Fonte: Autoria própria.

As regras que possuem saídas repetidas podem ser substituídas por uma única regra, assim, este projeto pode ser simplificado para a utilização de apenas 5 regras, como mostra a Tabela 2.

Tabela 2 - Regras utilizadas no controlador.

	VARIÁVEIS DE ENTRADA			VERIÁVEIS DE SAÍDA	
REGRA	FRENTE	ESQUERDA	DIREITA	MOTOR ESQUERDO	MOTOR DIREITO
1	PERTO	----------	---------	FRENTE	ATRÁS
2	PERTO	LONGE	PERTO	ATRAS	FRENTE
3	LONGE	----------	----------	FRENTE	FRENTE
4	LONGE	PERTO	LONGE	FRENTE	PARAR
5	LONGE	LONGE	PERTO	PARAR	FRENTE

Fonte: Autoria própria.

Esta ferramenta permite ainda que se realize simulações, o que é muito importante, pois permite a avaliação do sistema sem que seja necessário implementa-lo, e assim corrigir alguns detalhes que possam apresentar falhas antes de pôr em prática o que foi desenvolvido.

A Figura 29 mostra um exemplo de simulação utilizando o método do centro de massa para a defuzzificação, este método é o mais utilizado e que apresenta o melhor comportamento, porém é o mais difícil de implementar ele na prática de forma completamente exata (SIMÕES e SHAW, 2007).

Figura 29 - Simulação das saídas no MATLAB.

Fonte: Autoria própria.

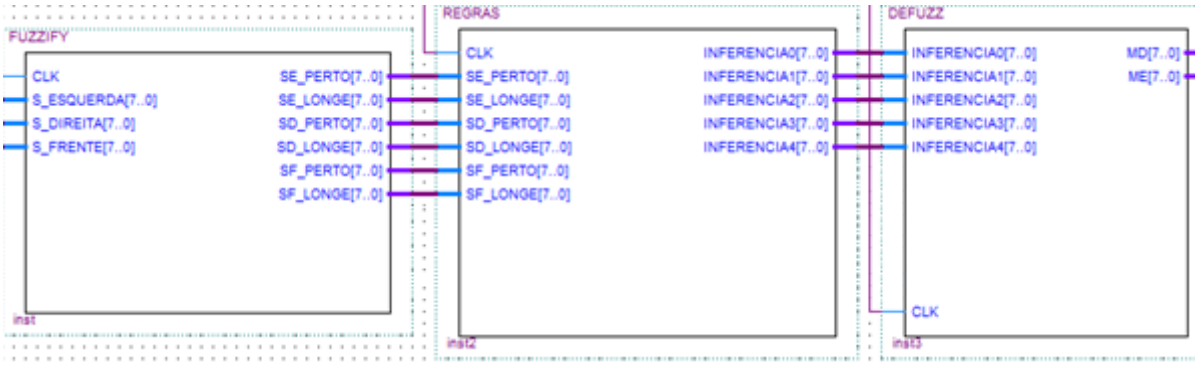
Na Figura 29 é possível observar as entradas em amarelo, e as saídas em azul, e o comportamento de cada regra, que estão presentes uma em cada linha de gráficos como enumerado a esquerda da imagem.

Todos estes procedimentos encerram o projeto do controlador na ferramenta do MATLAB, o próximo passo foi a implementação do controlador em VHDL.

4.7 PROJETO DO CONTROLADOR EM VHDL

Como visto anteriormente, o controlador é dividido em três partes que são, fuzzificação, regras e defuzzificação, por este motivo cada uma dessas partes foram divididas em blocos separados como ilustra a Figura 30.

Figura 30 - Blocos do controlador.



Fonte: Autoria própria.

O bloco a esquerda é responsável pela fuzzificação, e contem 4 entradas que estão listadas a seguir:

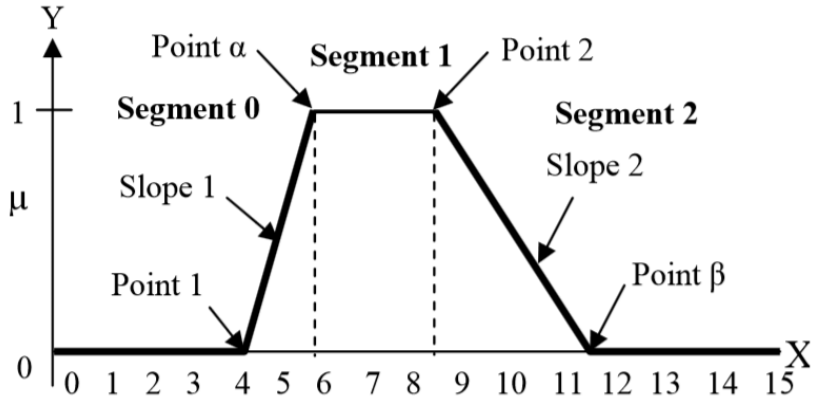
- CLK: Entrada de clock 50 Mhz;
- S_ESQUERDA: Entrada do sensor esquerdo;
- S_DIREITA: Entrada do sensor direito;
- S_FRENTE: Entrada do sensor frontal;

O bloco de fuzzificação também conta com 6 saídas, uma para cada conjunto pertencente as variáveis de entrada.

A fuzzificação é realizada através de uma função que recebe um vetor de inteiros, que indicam onde são as extremidades do conjunto que são todos de forma trapezoidal, no caso dos triângulos os valores da parte de cima são iguais.

A função recebe este vetor de 4 números inteiros que indicam os 4 cantos do trapézio, dividindo o mesmo em 3 segmentos, e duas retas importantes, a de subida e a de descida, como mostra a Figura 31.

Figura 31 - Exemplo de conjunto de fuzzificação.



Fonte: VUONG e MADNI, 2011.

Para o cálculo dos valores de inferência utilizam-se as expressões (8), (9), (10) e (11), considerando que o valor de inferência “μ” assume valores de 0 a 255, pois o mesmo é uma variável de 8 bits e o valor de entrada é o que vem diretamente dos sensores, ou seja, é a distância em centímetros.

$$Slope\ 1 = \frac{255}{\text{point}\ \alpha - \text{point1}} \tag{8}$$

$$Slope\ 2 = \frac{255}{\text{point}\ \beta - \text{point2}} \tag{9}$$

$$\mu = 255 - (\text{entrada} - \text{point2}) * \text{slope2} \tag{10}$$

$$\mu = (\text{entrada} - \text{point1}) * \text{slope1} \tag{11}$$

Este método é utilizado por Vuong e Madni (2011) e é um dos mais simples encontrados na literatura, uma vez que utiliza apenas operações comuns de soma, subtração, multiplicação e divisão, por este motivo ele foi utilizado neste projeto.

O bloco de regra é responsável por receber os dados codificados e realizar as operações entre os conjuntos como manda o conjunto de regras.

Para cada regra existe uma variável de saída no bloco de regras e que pode assumir valores entre 0 e 255, pois são variáveis de 8 bits como foi observado na Figura 30 anteriormente.

Neste projeto utiliza-se o operador de intersecção já apresentando anteriormente, onde todos os valores envolvidos nesta operação são comparados, restando somente o menor valor.

Essas variáveis de saída recebem o retorno de uma função que realiza a operação de intersecção entre os valores correspondentes da regra como mostra a Figura 32.

Figura 32 - Regras usadas em VHDL.

```
INFERENCIA0 <= MINIMO(SE_PERTO, BLANK, BLANK);
INFERENCIA1 <= MINIMO(SF_PERTO, SE_LONGE, SD_PERTO);
INFERENCIA2 <= MINIMO(SF_LONGE, BLANK, BLANK);
INFERENCIA3 <= MINIMO(SF_LONGE, SE_PERTO, SD_LONGE);
INFERENCIA4 <= MINIMO(SF_LONGE, SE_LONGE, SD_PERTO);
```

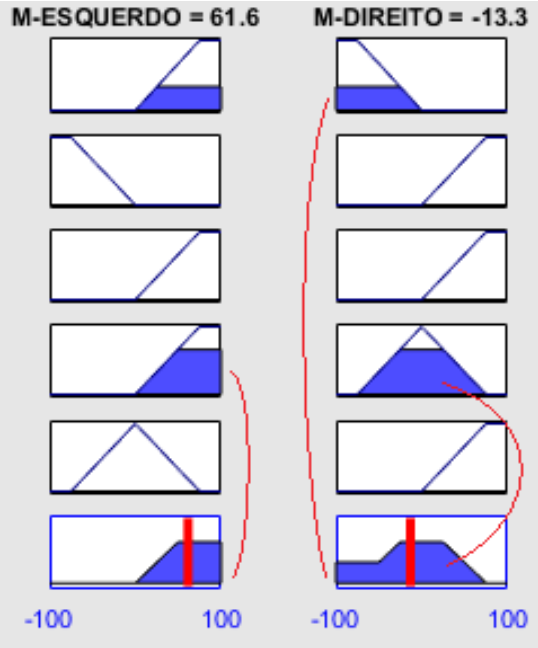
Fonte: Autoria própria.

Cada uma das variáveis de saída contém um valor de inferência referente a uma regra, e esses valores são passados para o deffuzificador que aplicará estes valores aos consequentes, ou seja, a saída do sistema.

O deffuzificador recebe os valores de inferência das regras, e aplica eles na saída, após isso é utilizado o método do centro de massa para a defuzzificação.

Para o cálculo do centro de massa é preciso calcular a área dos três conjuntos de saída já com seus respectivos valores de inferência como mostra a Figura 33.

Figura 33 - Demonstração de defuzzificação.



Fonte: Autoria própria.

Como mostra a Figura 33, somente o maior valor de cada conjunto é levado em consideração como mostra a saída do motor esquerdo, e os conjuntos diferentes resultantes são somados como mostra a saída do motor direito.

Para o cálculo da defuzzificação é preciso calcular a área resultante, e em todos os casos deste projeto, as saídas serão trapezoidais. A equação (12) apresenta a forma de se calcular a área de um trapézio, onde “A” é a área, “B” e “b” são referentes as bases maiores e menores e “h” a altura do mesmo.

$$A = \frac{B+b}{2} * h \tag{12}$$

Para realizar a defuzzificação de forma mais próxima possível em VHDL, é necessário trabalhar com esta equação deixando-a em função da altura, pois a altura representa o valor de inferência do conjunto. Assim recebe-se o valor de inferência e multiplica-o pela primeira parte da equação (que se pode chamar de constante de área) pois a mesma só depende das bases que serão sempre as mesmas.

O deffuzificador cria uma variável para cada conjunto de saída, e cada uma dessas variáveis recebe o maior valor de inferência de acordo com as regras como

visto anteriormente na saída do motor esquerdo na Figura 33. Isso é feito em VHDL através de uma função que retorna o maior valor como mostra a Figura 34

Figura 34 - Método de inferência de saída utilizado em VHDL.

```
ME_FRENTE_MAX <= MAXIMO (INFERENCIA0, INFERENCIA3, INFERENCIA4);
ME_PARA_MAX <= INFERENCIA5;
ME_ATRAS_MAX <= INFERENCIA2;

MD_FRENTE_MAX <= MAXIMO (INFERENCIA2, INFERENCIA3, INFERENCIA5);
MD_PARA_MAX <= INFERENCIA4;
MD_ATRAS_MAX <= INFERENCIA1;
```

Fonte: Autoria própria.

Os valores máximos de inferência obtidos conforme mostra a Figura 34, são utilizados para o cálculo da área apresentado anteriormente.

Uma das formas de facilitar o cálculo da defuzzificação é trabalhar com um conjunto central na saída como é o caso do conjunto PARAR, pois estes conjuntos centrais são metade positivos e metade negativos, o que totalizam uma área nula

Outra forma de facilitar o cálculo é por meio do espelhamento, como foi observado na Figura 28 os conjuntos FRENTE e ATRÁS são praticamente os mesmos, porém um é negativo e o outro é positivo, o que facilita o cálculo. Desta forma pode-se calcular a área apenas uma vez e utilizar para os dois modelos com sinal trocado.

Com os valores das áreas é possível calcular o valor de saída do sistema já defuzzificado através da equação (13), onde se calcula a média ponderada das áreas.

$$Saída = \frac{-Área*uATRÁS+0+Área*uFRENTE}{uATRÁS+uPARAR+uFRENTE} \quad (13)$$

Na equação (13), “uATRÁS”, “uPARAR” e “uFRENTE” são valores referentes ao valor máximo de inferência obtidos anteriormente, e a “Área” é apenas uma constante de área gerada com os pontos das bases do trapézio do conjunto FRENTE. Existe um zero na equação pois a área que representa o conjunto PARAR é nula.

Com tudo pronto foi necessário realizar os testes e verificar se tudo funcionaria corretamente.

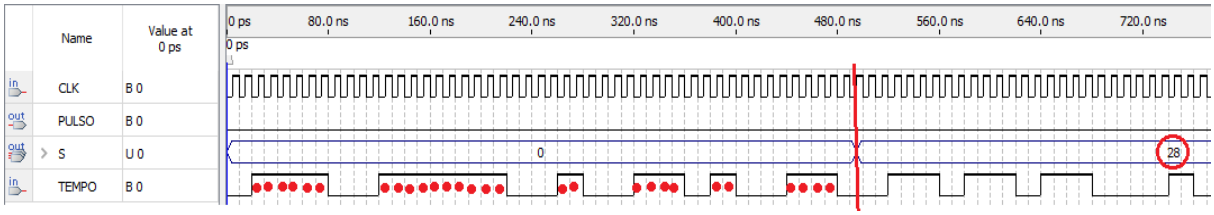
5 RESULTADOS

Neste capítulo serão apresentados os resultados da implementação do projeto em VHDL, suas simulações e uma análise das dificuldades encontradas para a realização prática.

5.1 TESTE DO SENSOR EM VHDL

Os testes realizados com o bloco do sensor demonstraram que ele é capaz de funcionar perfeitamente com o sensor. Esta característica é apresentada na simulação da Figura 35.

Figura 35 - Simulação do sensor em VHDL.



Fonte: Autoria própria.

Para a realização desta simulação foi necessário a redução dos valores dos contadores do bloco, pois o mesmo apresenta um ciclo completo de 38 ms, e o simulador mostra apenas 1 ms de simulação.

É possível verificar que o bloco conta de forma correta o número de clocks pelo qual o sinal de entrada permanece em tempo nível lógico alto, pode-se verificar na simulação que o bloco apresenta o valor 28 na saída, o que corresponde corretamente aos ciclos indicados com bolas vermelhas.

É importante verificar que o valor da saída só é atualizado ao final da medição informando somente a distância total ao controlador.

Esta simulação mostra que o sensor é capaz de funcionar perfeitamente em conjunto com este bloco em VHDL, podendo fornecer os dados necessários ao controlador.

Outro resultado interessante é notar a quantidade de *hardware* que fo exigido para a criação deste bloco. Voltando os valores dos contadores para os projetados durante a metodologia é possível verificar consumo de *hardware* gerado por um único bloco, o que está indicado na Figura 36.

Figura 36 - Consumo de *hardware* do bloco do sensor.

Family	Cyclone IV E
Device	EP4CE22E22C8
Timing Models	Final
Total logic elements	120 / 22,320 (< 1 %)
Total combinational functions	115 / 22,320 (< 1 %)
Dedicated logic registers	62 / 22,320 (< 1 %)
Total registers	62
Total pins	11 / 80 (14 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

Fonte: Autoria própria.

Na Figura 36 é possível observar que o bloco consumiu menos de 1% do *hardware* disponível na FPGA, o que é algo muito positivo, pois mostra que é possível utilizar diversos sensores, no caso deste projeto serão utilizados 3 sensores, além de deixar bastante *hardware* disponível para o resto do projeto.

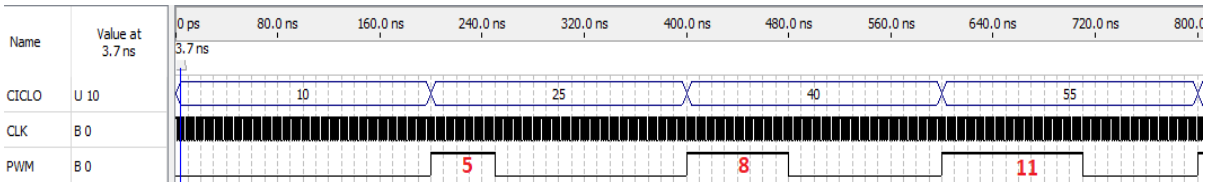
5.2 TESTE DO PWM EM VHDL

O teste do bloco de PWM tem que ser feito de duas maneiras, através de simulação como foi feito com o sensor e na prática utilizando uma fonte de tensão e o *driver* de acionamento dos motores.

Primeiro é preciso avaliar o funcionamento do PWM, para isso utiliza-se a simulação, que assim como no bloco anterior também foram necessárias reduções nos valores de contadores para uma simulação dentro de 1 ms.

A simulação do bloco de PWM pode ser vista na Figura 37.

Figura 37 - Simulação do PWM em VHDL.



Fonte: Autoria própria.

Como a Figura 37 mostra, o PWM funciona corretamente, para cada tamanho de pulso definido pela entrada CICLO, a saída muda e acompanha o valor indicado, para 25% o PWM ficou em nível lógico alto por 5 ciclos de clock de um total de 20, para 40% foram 8 ciclos de clock, isso mostra a funcionalidade correta deste bloco.

Assim como no bloco do sensor, é importante verificar o consumo de *hardware* deste bloco também, o que pode ser analisado na Figura 38.

Figura 38 - Consumo de *hardware* do bloco de PWM.

Top-level Entity Name	PWM
Family	Cyclone IV E
Device	EP4CE22E22C8
Timing Models	Final
Total logic elements	123 / 22,320 (< 1 %)
Total combinational functions	123 / 22,320 (< 1 %)
Dedicated logic registers	19 / 22,320 (< 1 %)
Total registers	19
Total pins	11 / 80 (14 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

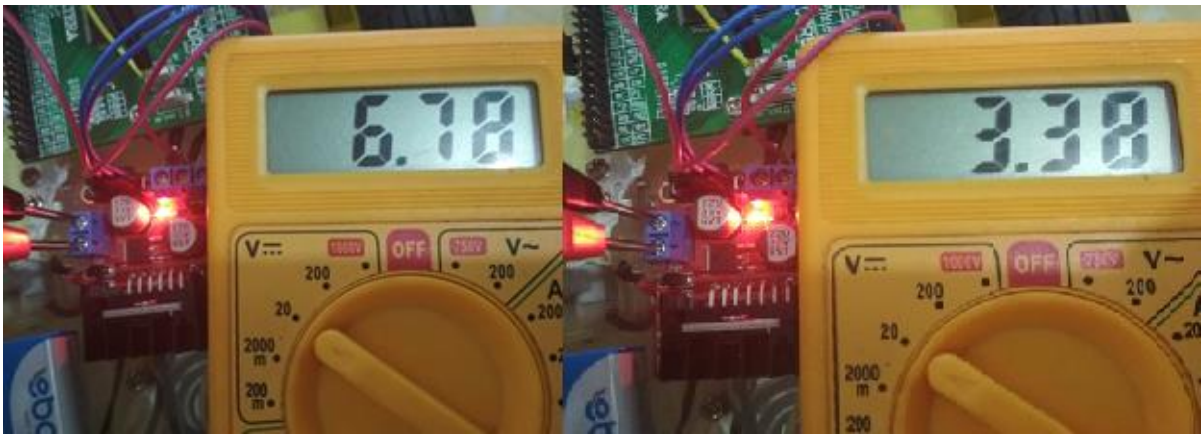
Fonte: Autoria própria.

Assim como o bloco do sensor o bloco PWM também consumiu menos de 1% do *hardware* disponível, o que é muito importante, pois abre uma margem maior para uso de *hardware* por parte do controlador e mostra uma eficiência grande no uso da FPGA.

Também é possível realizar o teste do PWM junto ao driver de acionamento do motor, usando o PWM como comando do driver, e usando um multímetro para medir a tensão na saída é possível verificar o funcionamento do PWM.

É possível ver um comparativo do PWM em funcionamento na Figura 39.

Figura 39 - Comparativo do PWM 100% e 50%.



Fonte: Autoria própria.

No comparativo é possível verificar que o PWM se comporta de forma correta pois o multímetro informa que o PWM com pulso de 100% fornece aos motores tensão de 6.78 V, enquanto que para um pulso de 50% ele informa que a tensão entregue é de 3.38 V, quase que exatamente a metade da tensão, visto que a metade exata seria 3.39 V.

O resultado obtido com os testes práticos do PWM são muito satisfatórios e apontam que o controlador terá todas as capacidades de atuar sobre a velocidade dos motores do protótipo.

5.3 TESTE DO CONTROLADOR EM VHDL

Era esperado do controlador em VHDL um consumo de *hardware* razoável, porém o mesmo apresentou um consumo de *hardware* muito abaixo do esperado, consumindo menos de 7% do *hardware* disponível como apresenta a Figura 40.

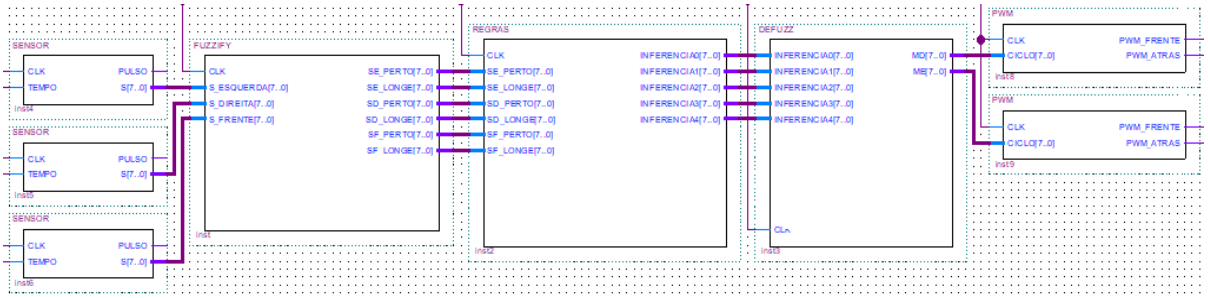
Figura 40 - Consumo de *hardware* do controlador.

Top-level Entity Name	CONTROLADOR
Family	Cyclone IV E
Device	EP4CE22E22C8
Timing Models	Final
Total logic elements	1,477 / 22,320 (7 %)
Total combinational functions	1,442 / 22,320 (6 %)
Dedicated logic registers	223 / 22,320 (< 1 %)
Total registers	223
Total pins	50 / 80 (63 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

Fonte: Autoria própria.

Essa informação é apenas do controlador, ainda é preciso liga-lo a três blocos de sensores na entrada e dois de PWM na saída como mostra a Figura 41.

Figura 41 - Controlador completo em VHDL.



Fonte: Autoria própria.

O controlador completo por sua vez acaba consumindo um pouco mais de *hardware* em relação ao controlador sozinho, como mostra a Figura 42.

Figura 42 - Consumo de *hardware* do controlador completo em VHDL.

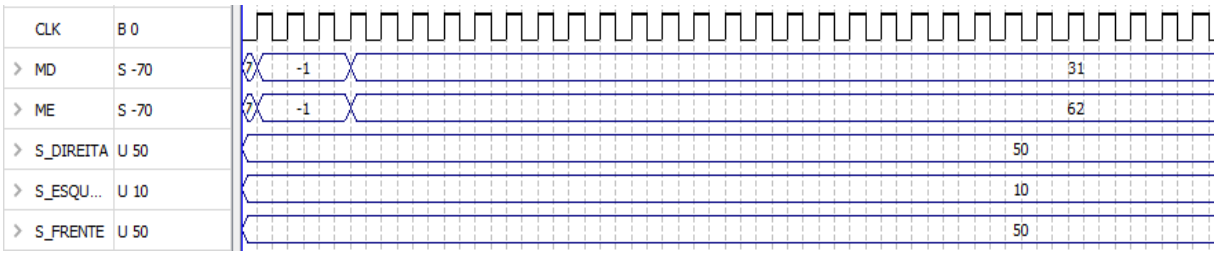
Family	Cyclone IV E
Device	EP4CE22E22C8
Timing Models	Final
Total logic elements	1,661 / 22,320 (7 %)
Total combinational functions	1,633 / 22,320 (7 %)
Dedicated logic registers	329 / 22,320 (1 %)
Total registers	329
Total pins	11 / 80 (14 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

Fonte: Autoria própria.

O fato de o controlador completo ter ocupado apenas 214 elementos lógicos a mais do que o controlador sozinho mostra que a inclusão de vários blocos iguais no sistema como é o caso do bloco do sensor e do PWM, não causam grande impacto no consumo de *hardware* do sistema.

O controlador em VHDL assim como os outros pode ser simulado como mostra a Figura 43. Porém diferente das outras simulações não é necessário realizar nenhuma alteração no controlador para que ele possa ser simulado.

Figura 43 - Simulação controlador em VHDL.



Fonte: Autoria própria.

Como é possível observar na simulação tanto o controlador em MATLAB quanto o projetado em VHDL apresentam saídas com números reais, isso torna o trabalho de análise mais fácil, para verificarmos a eficiência deste controlador basta comparar os resultados obtidos por ele com os resultados do controlador em VHDL usando entradas aleatórias, e isso pode ser observado na Tabela 3.

Tabela 3 - Comparação entre saídas simuladas em VHDL e MATLAB.

ENTRADA			SAÍDA - MATLAB		SAÍDA - VHDL	
FRENTE	ESQUERDA	DIREITA	M-ESQUERDO	M-DIREITO	M-ESQUERDO	M-DIREITO
5	15	25	65,6	-65,6	62	-62
10	5	30	65,6	-65,6	62	-62
35	40	5	0	65,6	0	62
26	9	44	60,7	7,88	62	-22
62	53	41	65,6	65,6	62	62
38	16	9	16,4	60,7	51	62
17	8	6	65,6	-65,6	62	-62
5	7	58	65,6	-65,6	62	-62

Fonte: Autoria própria.

A comparação observada na Tabela 3 mostra um ótimo resultado para o controlador descrito em VHDL.

Utilizando esta tabela comparativa é possível calcular o erro gerado pelas diferenças dos valores gerados.

A equação (14) mostra como se calcula o erro.

$$e = \frac{VHDL - MATLAB}{MATLAB} * 100 \quad (14)$$

Na equação (14), “e” representa o erro em porcentagem, “VHDL” é o valor simulado em VHDL, e “MATLAB” é o valor simulado no MATLAB.

Utilizando a equação (14) é possível realizar um comparativo com os erros de cada simulação, como mostra a Tabela 4.

Tabela 4 - Comparativo com o valor do erro.

SAÍDA - MATLAB		SAÍDA - VHDL		ERRO	ERRO
M-ESQUERDO	M-DIREITO	M-ESQUERDO	M-DIREITO	M-ESQUERDO	M-DIRETO
65,6	-65,6	62	-62	-5,49	-5,49
65,6	-65,6	62	-62	-5,49	-5,49
0	65,6	0	62	0,00	-5,49
60,7	7,88	62	-22	2,14	-379,19
65,6	65,6	62	62	-5,49	-5,49
16,4	60,7	51	62	210,98	2,14
65,6	-65,6	62	-62	-5,49	-5,49
65,6	-65,6	62	-62	-5,49	-5,49

Fonte: Autoria própria.

O comparativo mostra que a maioria dos erros não é superior a 5,5%, apenas em dois casos, um para cada motor o controlador em VHDL mostrou um erro exorbitante ultrapassando a casa dos 200%.

Apesar destes dois casos apresentarem erros extremamente altos eles não alteram o comportamento dinâmico do protótipo, pois em ambos os casos o protótipo se comporta de maneira correta.

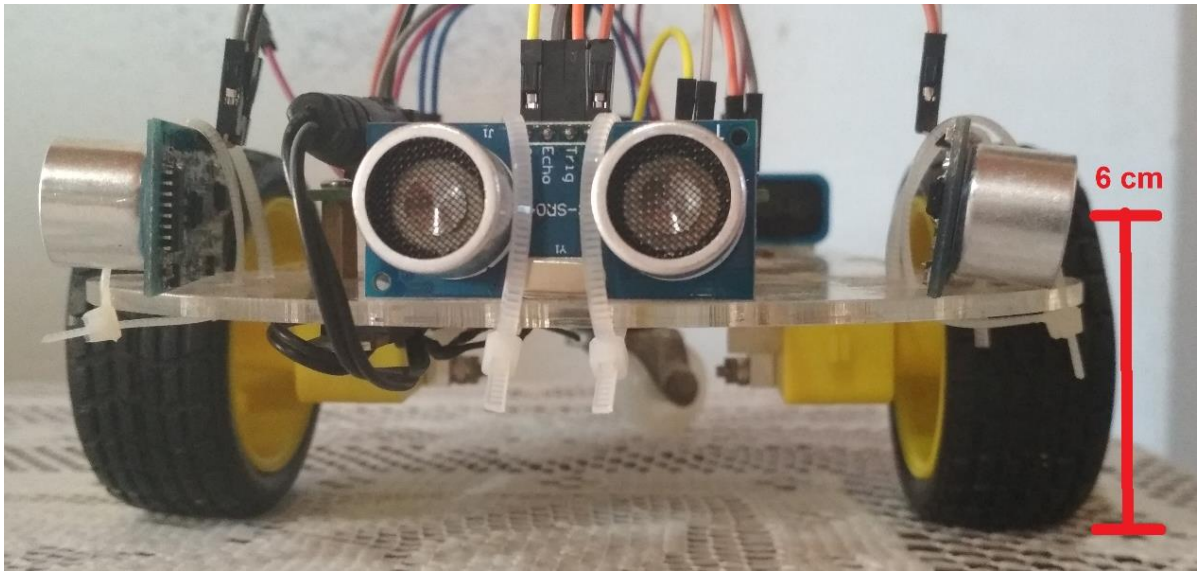
Esse comportamento do controlador evidencia mais uma das vantagens dos controladores *fuzzy,* eles só precisam responder de forma coerente ao que se propõe e podem ser configurados somente com o conhecimento daquilo que é preciso fazer, sem o uso de nenhum cálculo complexo e sem ter que modelar matematicamente nenhum elemento utilizado no sistema.

5.4 TESTE DO PROTÓTIPO

Os testes realizados com o protótipo mostram que ele se comporta de maneira correta em todas as situações simples, como desviar de paredes e desviar de objetos grandes.

O principal problema está relacionado com os objetos baixos, pois os sensores estão localizados em partes altas do protótipo, cerca de 6cm do chão como mostra a Figura 44.

Figura 44 - Vista frontal do protótipo.



Fonte: Autoria própria.

Essa altura dos sensores faz com que os sensores não sejam capazes de detectar objetos pequenos que interferem na locomoção do mesmo como um pedaço de madeira, pedras, etc.

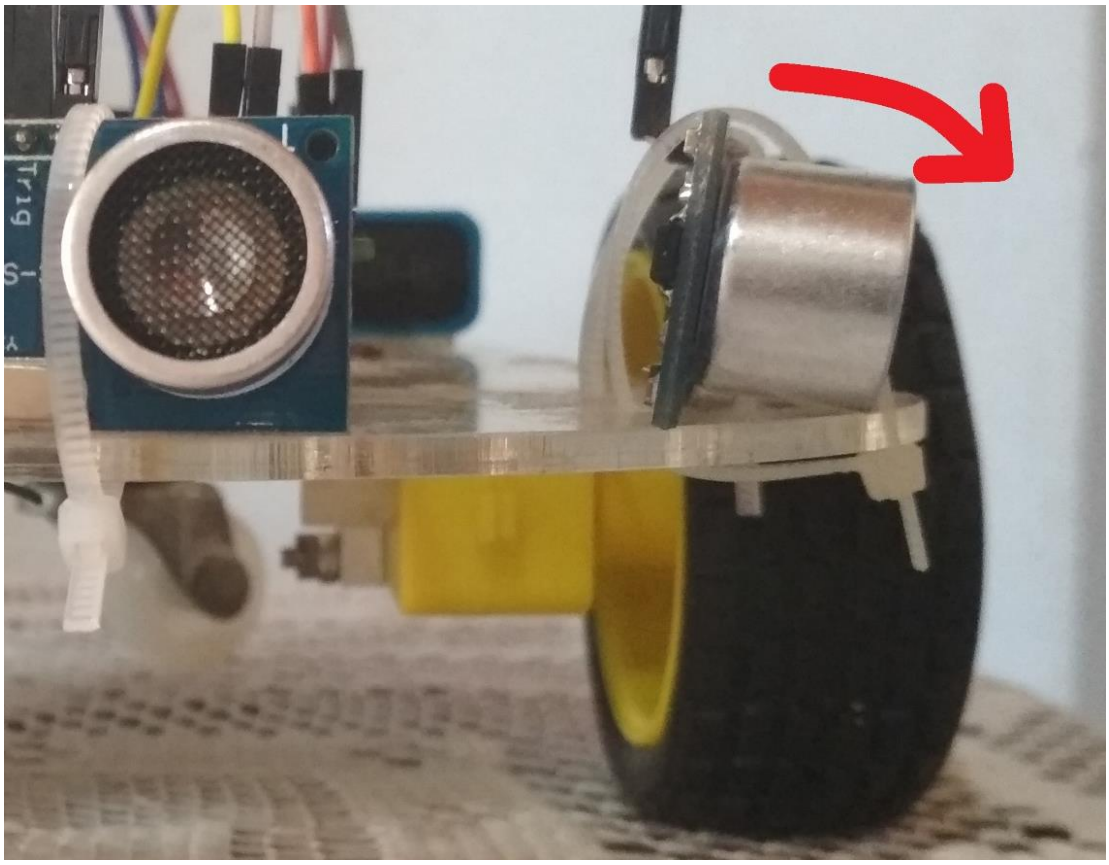
Outro fato a se notar é que este protótipo não foi feito para desviar de todos os tipos de obstáculos, como é o caso dos buracos, por este motivo ele não pode se locomover em alguns ambientes.

É preciso tomar cuidado também com ambientes que possuem beiradas com queda livre, pois o protótipo não é capaz de identificar essa situação e pode acabar caindo.

Pode ser que objetos muito finos possam acabar entrando em algum ponto cego dos sensores, pois estes sensores tem um alcance lateral um pouco baixo para objetos que estão muito próximos.

A questão dos objetos pequenos pode ser melhorada inclinando um pouco os sensores para baixo como mostra Figura 45.

Figura 45 - Ajuste dos três sensores.

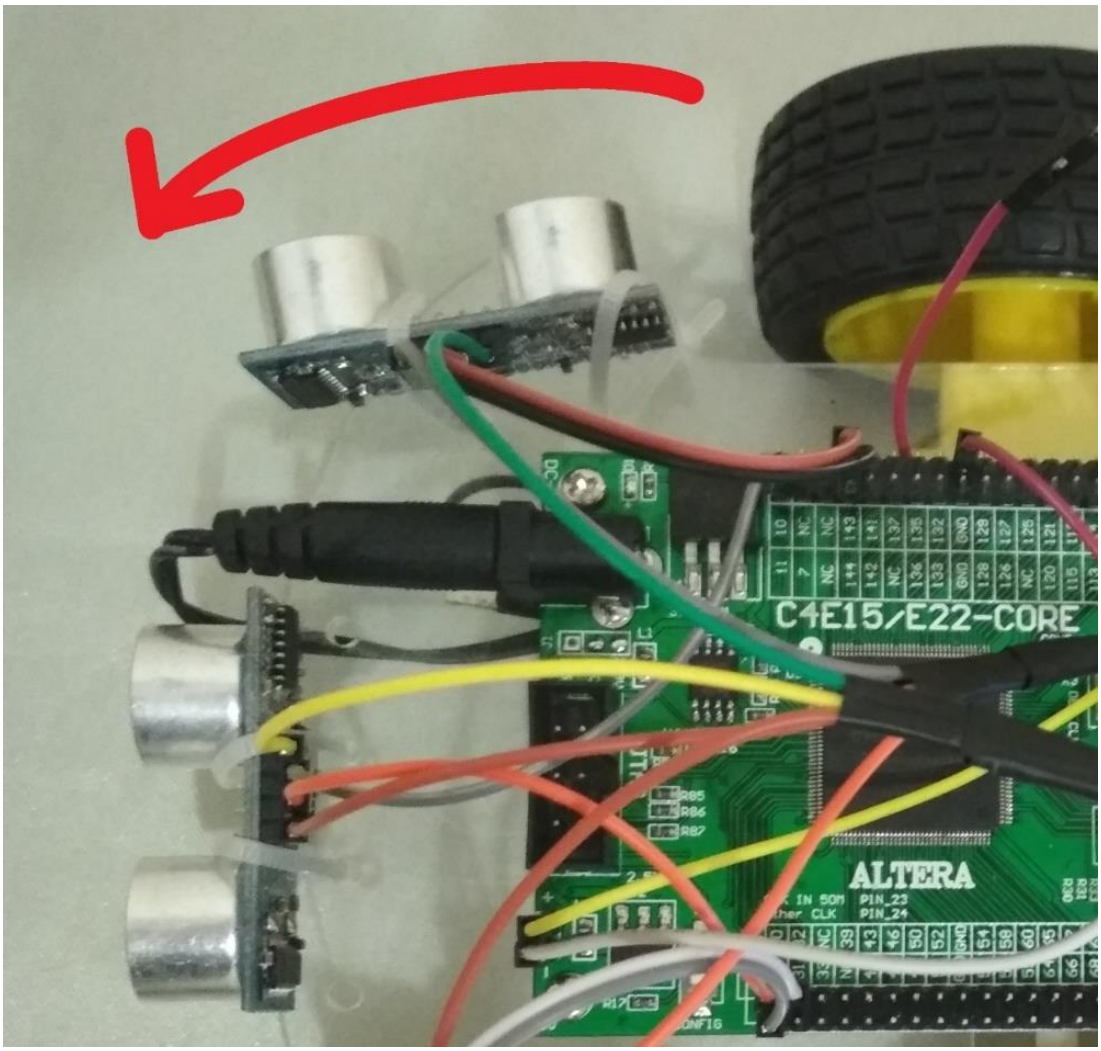


Fonte: Autoria própria.

O ajuste mostrado na Figura 45 não resolve o problema de objetos próximos, porém melhora a situação com objetos pequenos que estejam mais afastados. Uma mudança que provavelmente resolveria este problema seria colocar os sensores na parte de baixo do protótipo, porém neste caso não foi possível fazer isso devido ao posicionamento dos terminais dos sensores.

Um outro ajuste que melhora o problema com pontos cegos dos sensores é inclinar um pouco os sensores laterais para que os mesmos não fiquem totalmente perpendiculares ao sensor frontal como indica a Figura 46.

Figura 46 - Ajuste dos sensores laterais.



Fonte: Autoria própria.

Assim como o ajuste anterior, este também não resolve completamente o problema, porém diminui o número de ocorrência de pontos cegos. Para corrigir completamente o problema dos pontos cegos seria necessário utilizar mais sensores, ou utilizar sensores com um alcance lateral maior.

6 CONCLUSÃO

O objetivo deste trabalho foi projetar um sistema de navegação utilizando lógica *fuzzy* para um protótipo de carrinho inteligente capaz de desviar de obstáculos. Para isso foi desenvolvido um controlador *fuzzy* em VHDL, para ser utilizado com um chip FPGA visando ser um sistema embarcado ao protótipo.

As soluções encontradas no processo do levantamento bibliográfico foram indispensáveis para a realização do projeto. Nesta revisão foi visto que, diferentes métodos de implementação são utilizados com o intuito de aperfeiçoar essas técnicas.

Com os estudos realizados, foi proposto um projeto para que os objetivos fossem alcançados. A metodologia de realização da programação na linguagem VHDL segue uma orientação encontrada em diversos artigos analisados, a maioria fazia uso de módulos ou blocos de programas separados que permitem uma análise dos mesmos e a verificação de erros mais facilmente.

Os blocos foram desenvolvidos separadamente, para que se entendesse corretamente o funcionamento de cada elemento do sistema. Assim foi possível atender os objetivos de entender o funcionamento dos sensores, e utiliza-los em conjunto com uma placa FPGA. Além disso foi possível entender o funcionamento da ponte H e dos motores envolvidos no processo de atuação do sistema de controle.

Para cada bloco são realizadas algumas simulações que fornecem a análise do funcionamento de cada elemento envolvido no sistema, mostrando o controle e visando obter a arquitetura completa do sistema de navegação do robô de forma otimizada, tudo feito antes de iniciar-se a fase de teste com um protótipo.

Os resultados das simulações se mostram coerentes quando comparados com os resultados do processo *fuzzy* obtido pelo *software* MATLAB.

O problema do erro muito grande na comparação do sistema projetado em VHDL com o sistema teórico do MATLAB, mostrou que mesmo com uma margem grande de erro o sistema ainda responde de forma coerente ao que se espera, porém um pouco mais abrupto do que o controlador teórico.

A habilidade do programador em VHDL que é adquirida com a prática no projeto, facilita o desenvolvimento e diminui o tempo necessário para buscar soluções. As dificuldades na implementação de um projeto em VHDL, também traz consigo o cuidado que se deve ter com comandos ou funções que não são sintetizáveis.

Outro aspecto importante de ressaltar quando se trabalha com dispositivos lógicos programáveis é a ocupação da área do *hardware*, ou seja, o consumo de *hardware*, que neste projeto foi de 7% do total (1.661/22.320).

A análise dos resultados mostra que a aplicação do controlador *fuzzy* usado no projeto, para um sistema de navegação inteligente é viável e apresenta uma resposta bastante coerente.

As simulações do sistema presentes na Tabela 3, observa-se que nos nove casos simulados, os resultados são corretos e indicam que o protótipo pode se desviar dos obstáculos.

Todos os preparativos e simulações tornaram possível a construção de um protótipo, que assim como esperado foi capaz de desviar da maioria dos objetos exceto em alguns casos por limitações físicas dos sensores, que foram compensadas até certo ponto.

O protótipo resultante deste projeto demonstrou que é completamente possível a utilização de algoritmos inteligentes em FPGA, e ainda mostra que há bastante espaço para melhorias, visto que foram usados poucos recursos do *hardware* utilizado. Assim é possível fazer diversas melhorias no sistema, como adicionar sensores para identificação de buracos, utilizar técnicas mais modernas como algoritmos genéticos e redes neurais, utilizar sensores mais complexos, dentre outros.

REFERÊNCIAS

ALTERO, A. O. **Inteligencia Artifical:** teórica e prática. 1ª. ed. São Paulo: Livraria da Física, v. Único, 2008.

BERNARDO, J. 6 alternativas de como controlar motores com a Ponte H. **EletronWorld**, 06 out. 2016. Disponivel em: <http://eletronworld.com.br/eletronica/6-alternativas-de-como-controlar-motores-com-a-ponte-h/>. Acesso em: 15 mar. 2018.

BRAGA, N. C. Como funcionam os sensores ultrassônicos. **Instituto Newton C. Braga**, 2017. Disponivel em: <http://www.newtoncbraga.com.br/index.php/como-funciona/5273-art691>. Acesso em: 13 Maio 2017.

DIAS, A. Injeção Eletrônica: UCE(ECU) – Unidade de comando eletrônico. **Carros Infoco**, 2012. Disponivel em: <http://www.carrosinfoco.com.br/carros/2012/07/injecao-eletronica-uceecu-unidade-de-comando-eletronico/>. Acesso em: 12 Abril 2017.

FÓRMULA UFSCAR. Eletrônica Embarcada: Evolução dos sistemas Elétricos/Eletrônicos. **Projeto Fórmula Route UFSCar**, 2016. Disponivel em: <http://www.formula.ufscar.br/blog/eletronica-embarcada-evolucao-dos-sistemas-eletricoseletronicos/>. Acesso em: 2 Abril 2017.

HENRY, J. Android Things. **Developers**, 22 mar. 2018. Disponivel em: <https://developer.android.com/things/sdk/pio/pwm>. Acesso em: 25 abr. 2018.

LIMA, J. ABS – Como Funciona e Como Usar. **Auto Aftermarket News**, 2009. Disponivel em: <http://www.autoaftermarketnews.com/tecnica.php?cat=71&id=79&marca=0>. Acesso em: 12 Abril 2017.

MINUSSI, C. R. **Lógica Nebulosa (Lógica *Fuzzy*)**. Ilha Solteira: Unesp/FE/DEEE, 2009. 119 p.

NOGUEIRA, M. M. **Aplicando Lógica *Fuzzy* no Controle de Robôs Móveis usando Dispositivos Lógicos Programáveis e a Linguagem VHDL.** Ilha Solteira: [s.n.], 2013.

PEDRONI, V. A. **Eletrônica Digital Moderna e VHDL**. 1ª. ed. Rio de Janeiro: Elsevier, v. Único, 2010.

PEQUENINO, K. Ford investe 940 milhões em inteligência artificial para carros autónomos. **Público**, 2017. Disponivel em: <https://www.publico.pt/2017/02/13/tecnologia/noticia/ford-investe-mil-milhoes-de-dolares-em-inteligencia-artificial-1761863>. Acesso em: 12 Abril 2017.

PORTAL SISTEMAS EMBARCADOS. Sistemas Embarcados: A computação invisível. **Portal Sistemas Embarcados**, 2015. Disponivel em: <http://www.sistemasembarcados.org/2015/11/14/artigo-sistemas-embarcados-o-que-e-definicao/>. Acesso em: 12 Abril 2017.

RIGNEL, D. G. S.; CHENCI, G. P.; LUCAS, C. A. Uma Introdução a Lógica *Fuzzy*. **RESIGeT**, v. I, p. 28, Março 2011.

SHAH, A. Intel promete novos chips para inteligência artificial no próximo ano. **IDG Now**, 2016. Disponivel em: <http://idgnow.com.br/internet/2016/11/23/intel-promete-novos-chips-para-inteligencia-artificial-no-proximo-ano/>. Acesso em: 13 Maio 2017.

SIMÕES, M. G.; SHAW, I. S. **Controle e Modelagem *Fuzzy***. 2ª. ed. São Paulo: Blucher, v. Unico, 2007.

THOMAZINI, D.; ALBUQUERQUE, P. U. B. **Sensores industriais:** fundamentos e aplicações. 7ª. ed. São Paulo: Érica, v. Único, 2010.

THOMSEN, A. Driver Motor Ponte H L298n. **Filipe Flop**, 2013. Disponivel em: <https://www.filipeflop.com/blog/motor-dc-arduino-ponte-h-l298n/>. Acesso em: 22 fev. 2018.

THONSEM, A. Como utilizar o sensor ultrasônico HC-SR04. **BuildBot**, 17 jan. 2015. Disponivel em: <http://buildbot.com.br/blog/como-utilizar-o-sensor-ultrasonico-hc-sr04/>. Acesso em: 03 maio 2018.

TOCCI, R. **Sistemas Digitais:** Princípios e Aplicações. 11ª. ed. São Paulo: Pearson, v. Único, 2011.

VUONG, P. T.; MADNI, A. M. VHDL Implementation For a *Fuzzy* Logic Controller. **BEI Technologies, Inc.**, Sylmar, p. 8, 2011.

WENDLING, M. **Sensores**. UNESP. Guartinguetá, p. 19. 2010.