DISSERTAÇÃO DE MESTRADO

Arquiteturas de Hardware Dedicadas de Controladores Nebulosos para Auxílioà Locomoção de Deficientes Visuais

Tiago Romeiro de Jesus

Brasília, Agosto de 2017

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS MECATRÔNICOS

Arquiteturas de Hardware Dedicadas de Controladores Nebulosos para Auxílioà Locomoção de Deficientes Visuais

Tiago Romeiro de Jesus

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM SISTEMAS MECATRÔNICOS

APROVADA POR:

Prof. Dr. Daniel M. Muñoz Arboleda, PPMEC/UnB
Orientador

Prof. Dr. Carlos Humberto Llanos, PPMEC/UnB
Membro Interno

Prof. Dra. Suélia Rodrigues Fleury Rosa, FGA/UnB
Membro Externo

BRASÍLIA/DF, 18 AGOSTO DE 2017

Jesus, Tiago Romeiro de
Arquiteturas de Hardware Dedicadas de Controladores Nebulosos para Auxílioà Locomoção de Deficientes Visuais / Tiago Romeiro de Jesus. –Brasil, 2017.
94 p.

Orientador: Daniel Murício Muñoz Arboleda

Dissertação (Mestrado) – Universidade de Brasília – UnB
Faculdade de Tecnologia – FT
Programa de Pós-Graduação em Sistemas Mecatrônicos – PPMEC, 2017.

1. Tecnologia Assistiva. 2. Deficiente Visual. 3. Lógica Fuzzy. 4. Hardware Reconfigurável. 5. VHDL. 6. Gerador de Códigos. 7. Ponto Flutuante I. Daniel Maurício Muñoz Arboleda, orientador. II. Universidade de Brasília. III. Faculdade de Tecnologia.

Dedicatória

A Deus por me acalmar no caos da humanidade e aos meus pais por me tornarem o que sou, juntamente com meus irmãos e amigos.
A minha amada esposa, Shirley, que me proporciona a alegria de viver cada dia como se fosse o último.

Tiago Romeiro de Jesus

Agradecimentos

Agradeço ao meu orientador prof. Daniel Maurício Muñoz Arboleda no zelo pela orientação e por não medir esforços em transmitir seus conhecimentos sempre que necessário, além da amizade adquirida.

Obrigado aos membros da banca, prof. Carlos Humberto Llanos e a profa. Suélia de Siqueira Rodrigues Fleury Rosa, pela aceitação em avaliar a presente dissertação e cujas avaliações darão o toque final ao trabalho.

Agradeço imensamente à família Santos e Samolão, em nome dos meus amigos Lívia e Jr, por me receber sempre que precisei vir a Brasília e, principalmente, por me fazer sentir como parte da família.

Um abraço especial aos meus amigos de tempo da Casa do Estudante Universitário (CEU), que até hoje me proporcionam momentos de alegria: Thiago Ribeiro, Tiago Castro, Thiago Miranda, Diego, Eliete, Sanderson e Willton.

Agradeço imensamente ao meu amigo, José Adriany Victor de Aquino, sem o qual talvez eu não estaria aqui hoje finalizando o mestrado.

Ao coordenador do PPMEC, prof. Edson e a servidora Gláucia, pela atenção dada nos momentos necessários.

A todos os docentes que seguem tentado fazer com que nossos alunos possam "ler o mundo para poder transformá-lo" (Paulo Freire). Em especial aos professores do Eixo Tecnológico de Controle e Processos Industriais do Campus Jataí e aos professores do PPMEC.

Ao Instituto Federal de Goiás pela licença para cursar o mestrado e pelo suporte financeiro.

Tiago Romeiro de Jesus

RESUMO

Este trabalho apresenta um sistema de auxílio à locomoção de deficientes visuais em ambientes fechados em formato de óculos, denominado *Bleye* (*blind people eyes*) onde são utilizados sensores de medição de distância para o desenvolvimento de um controlador *fuzzy* para desvio de obstáculos, chamado FLOA (*fuzzy logic-based obstacle avoidance*). Inicialmente o sistema foi desenvolvido usando código estruturado, para servir como referência e, posteriormente, foi explorado o paralelismo intrínseco dos algoritmos envolvidos usando arquiteturas de hardware mapeadas em dispositivos FPGAs (*Field Programmable Gate Array*) utilizando representação numérica em ponto flutuante. Um ambiente de co-simulação usando o Matlab® e o Questasim® foi utilizado para realizar comparações numéricas com um código desenvolvido em texto estruturado que serviu como modelo de referência. A regularidade das arquiteturas em hardware desenvolvidas para o FLOA permitiu a criação de um gerador automático de código VHDL (*Very high speed integrated circuits Hardware Description Language*) a partir de um modelo de alto nível de controladores *fuzzy* Takagi-Sugeno genéricos. Com essa ferramenta de geração de código VHDL, denominada *fis2hdl*, foi possível comparar o impacto no consumo de recursos em relação: (a) tamanho da palavra, (b) número de entradas e (c) número de saídas. Os testes realizados com a implementação das arquiteturas de hardware propostas mostram que: (a) o sistema possui um tempo de execução de 17.5 μs, em contraste com implementações em software usando um *Desktop Intel core i7* operando a 2.4 GHz e um *Arduino Mega* operando a 16 MHz, os quais tem um que teve um tempo de execução de 1 ms e 752 ms, respectivamente; (b) a escolha da representação numérica de 27 bits se mostrou eficiente em relação ao consumo de recursos. No pior cenário foram consumidos 8256 LUTs, 2759 FFs e 10 DSPs atingindo uma precisão de 4.89×10^{-5} se comparado com uma implementação de 64 bits e uma frequência de operação de 50 MHz; (c) No pior caso o consumo estimado de energia foi de 189 mW, sendo 92 mW de potência dinâmica. Os resultados mostram que o sistema Bleye/FLOA é eficaz quanto ao tempo de execução e o menor consumo de recursos quando comparados com uma solução de 64 bits. Finalmente, uma analise de escalabilidade realizada com auxílio da ferramenta *fis2hdl* permitiu verificar que o consumo de recursos de hardware aumenta mais significativamente com a variação do número de entradas do sistema, pois este parâmetro afeta diretamente o número de elementos da base de regras.

ABSTRACT

This work presents a locomotion aid system for the visually impaired in indoor environments, called Bleye (blind people eyes), where distance measuring sensors were used to develop a fuzzy system for obstacle avoidance, called FLOA (fuzzy logic-based obstacle avoidance). Initially the system was developed using structured code to serve as a reference, and later the intrinsic parallelism of the involved algorithms was explored using hardware architectures mapped on FPGAs (Field Programmable Gate Arrays) and custom floating-point numerical representations. A co-simulation environment using Matlab® and Questasim® was used to perform numerical comparisons against the code developed in structured text that served as reference model. The regularity of the proposed hardware architectures for the FLOA allowed the creation of an automatic VHDL (Very high speed integrated circuits Hardware Description Language) code generator, called *fis2hdl*, from a high-level representation of generic Takagi-Sugeno fuzzy controllers. This generator tool enables to compare the impact on resource consumption in relation to: (a) word size, (b) number of inputs and (c) number of outputs. The tests performed with the implementation of the proposed hardware architectures show that: (a) the system has a execution time of 17.5μs, in contrast to software implementations based on an Desktop Intel core i7 operating at 2.4 GHz and an Arduino Mega at 16 MHZ, which achieved an execution time of 1 ms and 752 ms, respectively; (b) the choice of the 27-bit numerical representation was efficient in relation to resources consumption. In the worst case scenario, 8256 LUTs, 2759 FFs and 10 DSPs were consumed achieving an accuracy of 4.89×10^{-5} compared to a 64-bit implementation and an operational frequency of 50 MHz; (c) In the worst case, the estimated energy consumption was 189 mW (92 mW of dynamic power). The results show that the Bleye/FLOA system is efficient in terms of runtime and resource consumption if compared to a 64-bit solution. Finally, a scalability analysis was performed using the *fis2hdl* tool, verifying that the number of inputs notably affects the increment on the hardware resources consumption since this parameter increases the number of elements of the rule base.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS Siglas

ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuits
BLEYE	Blind People Eyes
BRAM	Block RAMDelay-Locked Loop
CAT	Comitê de Ajuda Técnica
CPLD	Complex Programmable Logic Device
DAC	Digital to Analog Converter
DFLC	Digital Fuzzy Logic Controller
DSP	Digital Signal Processing
ENM	Departamento de Engenharia Mecânica
FGA	Faculdade do Gama
FIR	Finite Inpulse Response
FK	Filtro de Kalman
FLOA	Fuzzy Logic Based Obstacle Avoidance
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GPS	Global Position System
IDEA	International Data Encryption Algorithm
IEEE	Institute of Electrical and Electronics Engineers
IIR	Infinite Impulse Response
IP	Intellectual Property
L	Longe
LEIA	Laboratory of Embedded Systems and Integrated Circuits
LUT	Look-Up Table
MP	Muito Perto
P	Perto
PAR	Place And Route
PIBIC	Programa Institucional de Bolsas de Iniciação Científica
PID	Proporcional, Integral e Derivativo
PWM	Pulse Width Modulation
RAM	Random Access Memory
RFID	Radio Frequency IDetification
RISC	Reduced Instruction Set Computer
RNA	Rede Neural Artificial
SoC	System on-Chip
SUS	Sistema Único de Saúde
TA	Tecnologia Assistiva
TCC	Trabalho de Conclusão de Curso
TS	Takagi-Sugeno

TTL	Transistor-Transistor Logic
UnB	Universidade de Brasília
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
ZVD	Zona Virtual Deformável

Capítulo 1

INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O objetivo da *Tecnologia Assistiva* (TA) é desenvolver equipamentos, dispositivos e sistemas que sejam capazes de superar a lacuna que faz com que as pessoas com deficiências não se tornem um cidadão pleno, capazes de fazer tudo o que os demais podem fazer (1). O comitê de ajudas técnicas (CAT) estabeleceu em 2006 a seguinte definição (2):

> "Tecnologia Assistiva é uma área do conhecimento, de característica interdisciplinar, que engloba produtos, recursos, metodologias, estratégias, práticas e serviços que objetivam promover a funcionalidade, relacionada à atividade e participação, de pessoas com deficiência, incapacidade ou mobilidade reduzida, visando sua autonomia, independência, qualidade de vida e inclusão social."

Com o empoderamento das pessoas com deficiência o conceito de deficiência tem mudado bastante e, atualmente, o mais aceito é do modelo social, onde a deficiência é definida como a perda ou redução de oportunidades, barreiras físicas e sociais, para que esses sujeitos tenham igualdade de condições. Esse conceito se difere do conceito médico, que considera como deficiência "qualquer perda ou anormalidade de estrutura ou função psicológica, física ou anatômica". Mesmo com a atualização da definição, em 2001, o conceito ainda está focado de forma individual, sem levar em consideração o meio externo. Isso é evidenciado quando se observa que a grande maioria dos projetos não levam em consideração as necessidades das pessoas com deficiência (1).

Apesar da importância da avaliação dos resultados dos projetos em TA – cujos dados poderiam ser utilizados para, primeiro, diminuir as altas taxas de abando do dispositivo (Fuhrer et al., 2003 apud (1)), segundo, aumentar a responsabilidade dos governos (DeRuyter, 1997 apud (1)) e até mesmo dar evidências para que o mesmo possam financiá-los – a maioria dos projetos não são avaliados adequadamente, pois há uma crença na obviedade dos benefícios (1). A maioria dos projetos de TA tem o foco nos avanços tecnológicos, sem a devida preocupação em verificar a interação dos mecanismos desenvolvidos com o usuário, levando a utilização de mais de uma tecnologia para atingir o objetivo e a confecção de dispositivos muito grandes ou pesados – mesmo com o avanço na miniaturização de dispositivos. Todos esses fatorem

levam a uma alta taxa de rejeição dos projetos pelos usuários finais (Phillips e Zhau, 1993 apud (1)). No caso específico de deficientes visuais o estudo feito em Nottingham revelou que três em cada dez pessoas com deficiências visuais não saem de casa sem acompanhamento (Clark-Carter et al. 1981 apud (1)). Em idosos o resultado é ainda mais alarmante, pois dois em cada três idosos não se locomovem sozinhos (Yerasimou, 2002 apud (1)). 86% destes deficientes não se sentem seguros em utilizar o transporte público e isso, afeta a oportunidade de emprego (Campion et al., 2003 apud (1)). Além disso, Yerasimou (2002 apud (1)) relata que um ponto crucial para os deficientes visuais é o medo da perda de orientação e alta ocorrência de acidentes durante a viagem devido à colisão com obstáculos.

Historicamente os deficientes visuais sempre se utilizaram de vários métodos e ferramentas para seu apoio e mobilidade, como simples $varas$ e animais adestrados. Os sistemas atuais só tiveram avanços há poucas décadas, inclusive as bengalas, que só tiveram seu uso generalizado a partir de 1950 e os cães guias, a partir de 1920 (ver Figura 1.1).

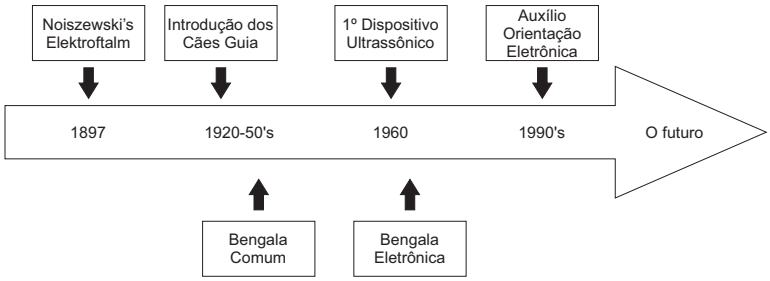


Figura 1.1: Histórico dos dispositivos de TA para deficientes visuais - adaptado (Hersh, 2008)

Interessante notar que o primeiro dispositivo eletrônico foi desenvolvido em 1897, quando Noiszewski criou o Elektroftalm para auxiliar à mobilidade dos cegos através de uma única célula de selênio colocada na testa do deficiente para indicar se havia luz através de uma saída sonora (3), mas os grandes avanços só ocorreram a partir da Segunda Guerra Mundial com o advento do sensoriamento remoto utilizando ultrassons e radiação infravermelha. Já no ano de 1960 iniciou o desenvolvimento de bengalas a laser. Outro foco, no início do século 21, consistiu no desenvolvimento de dispositivos para orientação espacial que indicam para o usuário sua posição em relação a um determinado sistema, usando, por exemplo, radiação infravermelha. Essa tecnologia é bem sucedida, mas possui uma instalação relativamente cara, assim como os sistemas atuais que utilizam câmeras e sistemas de posicionamento global (GPS) (1).

Um ponto fundamental no desenvolvimento de equipamentos de TAs com tecnologia nacional é buscar um diminuição de custos em relação a equipamentos importados, além de poder melhorar o atendimento coberto pelo SUS (Sistema Único de Saúde). É válido ressaltar que, além de construções tecnológicas para a melhoria de vida dos deficientes, é importante a relação dos processos históricos, ambientais e sociais que envolvem a vida do mesmo, pois o sofrimento humano aborda questões que vão além da parte biológica (2)

1.2 DESCRIÇÃO DO PROBLEMA

Atualmente no *LEIA*/UnB (*Laboratory of Embedded Systems and Integrated Circuits Applications*) está sendo desenvolvido um dispositivo de auxílio à locomoção de deficientes visuais em ambientes fechados, denominado *Bleye* (blind people eyes), em formato de óculos. A proposta é que o sistema seja capaz de indicar para o deficiente visual onde se localiza uma loja ou local dentro de ambiente fechado, como shoppings, centro culturais, hospitais, universidades, etc. A localização global do deficiente será realizada por triangulação utilizando antenas bluetooth ou wifi. Já a detecção de obstáculos que possam impedir a livre movimentação do deficiente será feita através de sensores como ultrassom, infravermelho e um sistema de visão estéreo. O sistema conta com a ajuda de dispositivos auxiliares, como sistemas inerciais para indicar a intenção de movimento e de atuadores piezoelétricos para indicar ao usuário a direção de movimento. Dois trabalhos relacionados a esse projeto foram desenvolvidos: (a) um PIBIC (Programa Institucional de Bolsas de Iniciação Científica) no qual foi especificado um filtro de Kalman para Fusão sensorial e (b) um TCC (Trabalho de Conclusão de Curso) no qual se integraram os FKs (Filtros de Kalman) em um FPGA (4). Por outro lado, um trabalho de mestrado está trabalhando com a parte de visão computacional para determinação do mapa de disparidade entre as imagens direita e esquerda de um kit estereoscópico.

O diagrama do sistema *Bleye* está representado na Figura 1.2 e o mesmo irá demandar uma alta capacidade computacional para lidar com algoritmos de visão estéreo, filtros Kalman e lógica *fuzzy*, sendo que sua execução deve ocorrer em apenas algumas dezenas de milissegundos, de preferência em menos de 100 ms, para garantir a segurança do usuário. Além disso, é importante que o sistema seja ajustado de acordo com a experiência do usuário e seja possível alterar o número de entradas/saída, mantendo a escalabilidade.

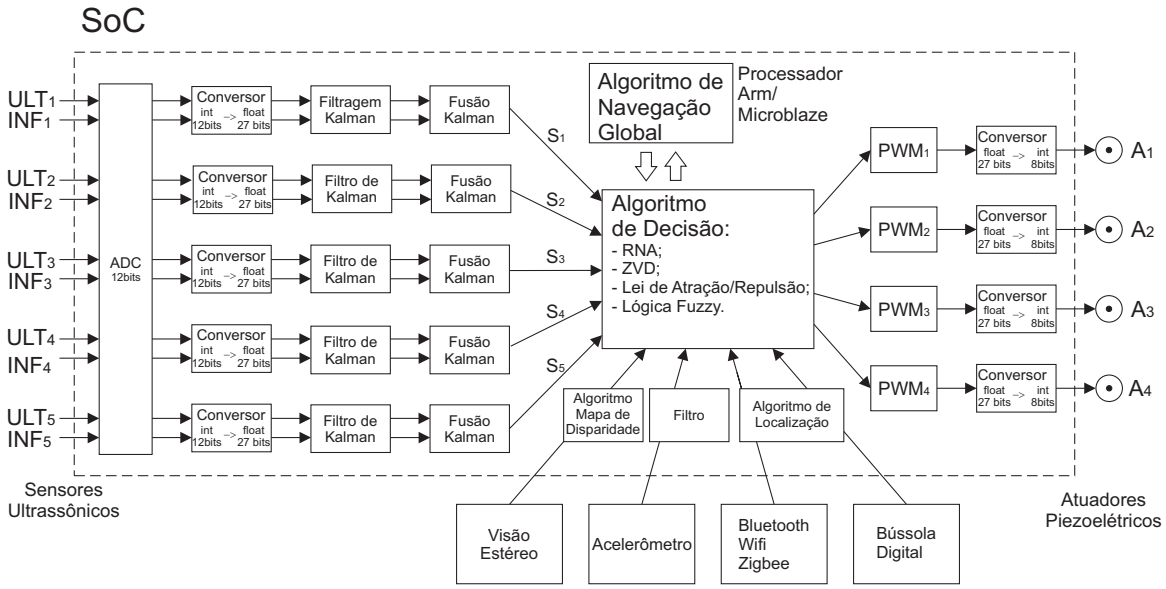


Figura 1.2: Esquema geral do sistema proposto - *Bleye*

Diante do exposto, o presente trabalho apresenta um módulo de auxílio a deficientes visuais em formato de óculos, o Bleye, cujo sistema de controle está baseado em um controlador *fuzzy* Takagi-Sugeno (TS) de ordem zero, denominado *FLOA* (*fuzzy logic based obstacle avoidance*), e implementado em arquiteturas de hardware dedicadas usando dispositivos FPGAs.

1.3 OBJETIVOS

1.3.1 OBJETIVO GERAL

Desenvolver um sistema baseado em lógica *fuzzy* para desvio de obstáculos usado na navegação de deficientes visuais em ambientes fechados, FLOA, embarcado em arquiteturas reconfiguráveis de forma a se obter um sistema de alto desempenho em termos de velocidade de execução e com baixo consumo de hardware.

1.3.2 OBJETIVOS ESPECÍFICOS

Espera-se alcançar os seguintes objetivos específicos:

- Projetar um sistema de controle *fuzzy* do tipo Takagi-Sugeno (TS) para o desvio de obstáculos (FLOA);
- Implementar o FLOA numa arquitetura reconfigurável, explorando o paralelismo intrínseco do algoritmo;
- Implementar o FLOA em código estruturado no Matlab® para servir de referência;
- Desenvolver uma ferramenta de geração automática de código VHDL.

1.4 METODOLOGIA

A proposta desta dissertação está na utilização de múltiplos sensores ultrassônicos e em quatro atuadores piezoelétricos, posicionados num dispositivo em formato de óculos (vide Figura 1.3), para indicar ao usuário qual a direção de movimento que o mesmo deve seguir para evitar a colisão com obstáculos estáticos e em ambientes fechados.

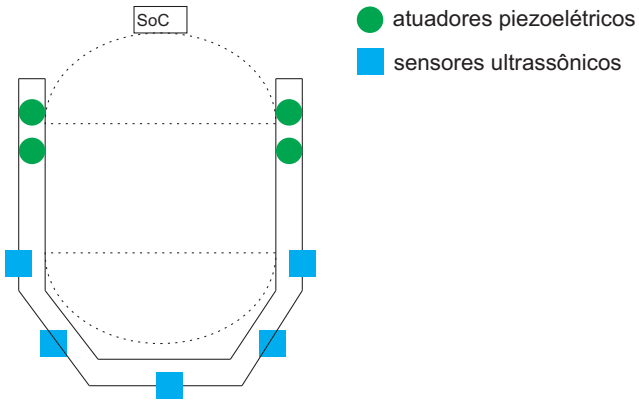


Figura 1.3: Disposição dos dispositivos no óculos: (i) 5 sensores ultrassônicos dispostos 45° entre si fixados na haste/aro do óculos; (ii) 4 atuadores piezoelétricos dispostos ao longo da haste; (iii) SoC fixado na ponteira dos óculos, dando a possibilidade de ficar oculto no vestuário.

A decisão de qual direção de movimento o usuário deve seguir para desviar de obstáculos será realizada por um controlador *fuzzy* com modelo de inferência Takagi-Sugeno (TS). A escolha desse controlador se deve a alguns pontos importantes:

1. O controle se baseia em conhecimentos heurísticos e pode ser alterado de forma relativamente simples após a realização de testes reais do sistema com deficientes visuais;

2. Em comparação com os outros controladores *fuzzy*, o controlador *fuzzy*–TS é o que possui os cálculos mais simples de serem implementados em plataformas embarcadas com restrições em termos de capacidade computacional;

3. Comparado a outros controles que vêm sendo amplamente utilizados para a detecção de obstáculos, como Zonas Virtuais Deformáveis (ZVD), Lei da Atração/ Repulsão e Redes Neurais Artificiais (RNAs) a Lógica *fuzzy* também já está consolidada e, apesar disso, pouco utilizada para esse caso específico. Essa simplicidade é uma característica do controlador *fuzzy*–TS e irá facilitar a implementação do algoritmo em FPGA.

A escolha do sistema em FPGA se deve ao fato da busca por um dispositivo com dimensões reduzidas e de alto poder computacional. Outro fator preponderante é poder explorar o paralelismo dos algoritmos, visto que o sistema global, representado pela Figura 1.2, irá receber várias informações do meio - através dos sensores de ultrassom, infravermelho, câmeras, antenas, etc. e alguns desses módulos possuem entradas múltiplas que poderão ser tratadas de forma paralela, como o módulo de filtragem de sinais, o módulo de localização e o módulo de detecção de obstáculos - demandando um sistema mais robusto. Além disso, alguns módulos do sistema, como o de visão computacional, consomem uma quantidade razoável da área no dispositivo, pois possuem algoritmos de alta complexidade.

Os sistemas foram descritos em linguagem de descrição de hardware VHDL e com isso, observou-se que existia algumas vantagens e/ou desvantagens ao se utilizar os controladores *fuzzy*-TS, como as descritas a seguir:

1. A regularidade da estrutura dos códigos desenvolvidos desde o início da criação do projeto do controlador *fuzzy*–TS, principalmente no desenvolvimento do código em Matlab e em Linguagem C;

2. Descrever manualmente sistemas complexos em VHDL é processo propenso a erros, que só são verificadas após a síntese lógica do projeto;

3. Dificuldade em se alterar o código VHDL em caso de necessidade de alteração nos números entradas/ saídas/ regras no controlador *fuzzy* (escalabilidade).

O projeto foi desenvolvido utilizando arquiteturas de hardware em ponto flutuante; apesar da saída ser o valor do duty cycle do PWM (Pulse Width Modulation), e não ser necessário uma precisão tão grande. Deve-se ressaltar que a maioria dos trabalhos pesquisados utilizam inteiros ou ponto fixo para descrever controladores *fuzzy* em VHDL. A utilização do ponto flutuante é justifica porque na maioria das aplicações que demandam uma alta capacidade computacional, como em robótica, devem ser calculados com alta precisão (5) e (6).

Para atender aos propósitos citados acima, o projeto de dissertação foi desenvolvimento em etapas que estão descritas sucintamente abaixo e que serão melhor explanadas no decorrer da dissertação:

1. ª Etapa: Projeto do controlador *fuzzy*–TS. Detalhes do projeto serão apresentados no capítulo 3;
2. ª Etapa: Implementação, usando a toolbox fuzzy do *Matlab*® do controlador *fuzzy*–TS para verificação dos requisitos. O capítulo 3 também apresenta os respectivos resultados de simulação usand o toolbox fuzzy do Matlab;
3. ª Etapa: Desenvolvimento, em texto estruturado, do código do controlador *fuzzy*–TS.
4. ª Etapa: Validação do código desenvolvido em texto estruturado, comparando com a implementação realizada na etapa 2. Os resultados dessa comparação sao apresentados no capítulo 4;
5. ª Etapa: Implementação das arquiteturas de hardware do controlador *fuzzy*–TS em VHDL, tendo como base o código em texto estruturado desenvolvido na etapa 3 (Capítulo 04). O capítulo 4 apresenta detalhes das arquiteturas propostas.
6. ª Etapa:Validação da arquitetura de hardware desenvolvida em VHDL, comparando-a com o código em texto estruturado;
7. ª Etapa: Desenvolvimento de um gerador automático de código VHDL para controladores *fuzzy*–TS de ordem zero. Os requisitos, desenvolvimento, resultados e analises do gerador de código, assim como as suas limitações serão apresentados no capítulo 5.

Nas etapas acima foram utilizadas ferramentas computacionais que já são bastante difundidos na literatura, como o software *Matlab*® e suas toolboxs, o software Vivado da *Xilinx*®, necessário para desenvolver e testar os códigos em VHDL e o *QuestaSim* da *Mentor*® que é um software de co-simulação que será utilizado para validação do algoritmo em VHDL.

1.5 CONTRIBUIÇÕES DO TRABALHO

Este trabalho originou duas publicações listadas no Apêndice A, além de uma publicação para conferência internacional, COBEM 2017, que está sendo finalizada.

Nesse trabalho foi desenvolvido um controlador *fuzzy*–TS embarcado em um sistema reconfigurável em formato de óculos para o auxílio a locomoção de deficientes visuais em ambientes fechados. O sistema mostrou se eficiente quanto ao tempo de execução sendo que as duas arquiteturas propostas, paralela e sequencial, são mais de 40000 vezes superior ao mesmo controlador implementado em uma arquitetura de software (Arduíno Mega). A arquitetura paralela se mostrou eficiente quando utilizada, sendo superior a cinco vezes mais rápido do que a arquitetura sequencial. O sistema testado mostrou-se de acordo com o esperado, mas espera-se uma melhora com testes reais.

A principal contribuição do trabalho foi o desenvolvimento de um gerador automático de códigos VHDL do controlador *fuzzy*–TS de ordem zero a partir da obtenção dos dados de uma ferramenta de alto nível. No caso o toolbox *fuzzy* do Matlab. Isso foi possível devido a regularidade das arquiteturas

desenvolvidas. A utilização da representação aritmética em ponto flutuante deixa o trabalho na fronteira do conhecimento, pois até o presente momento não foi encontrado nenhum gerador automático de código de controladores fuzzy com aritmética em ponto flutuante.

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

2.1 LÓGICA *FUZZY*

A lógica *fuzzy* é uma das várias técnicas de Inteligência Artificial (AI) que vem se desenvolvendo desde meados do século passado. A AI se baseia na inteligência humana, considerando principalmente o seu comportamento, para solucionar problemas (7). O fator mais importante da AI é a capacidade de aprendizagem dos sistemas, melhorando seu desempenho pela aplicação de novos métodos e conhecimento ou pela melhoria dos métodos existentes.

Utilizar a lógica clássica, de Aristóteles, para descrever o modelo do raciocínio humano é praticamente impossível. Lukasiewicz, em 1920, quebrou essa premissa com a ideia de um conjunto de valores entre verdadeiro e falso para refletir o grau de verdade (8). Zadeh (9), em 1965, apresentou o conceito de lógica *fuzzy* expandiu a teoria dos conjuntos clássicos, que permitia uma maior aproximação com a forma como a mente humana processa as informações. Embora a construção de regras básicas seja fácil, ajustar o sistema não é tão simples e demanda excessivos testes e erros: aprendendo com experiência e se adaptando a mudanças das condições operacionais (10).

A teoria dos conjuntos *fuzzy* permite fazer uma associação gradual nos processos de tomadas de decisão, através da avaliação do grau de pertinência dos termos *fuzzy*. Muitos sistemas de AI, ao lidar com sistemas difusos, tem seu projeto facilitado (11).

Os conceitos de conjuntos *fuzzy* são apresentados a seguir.

2.1.1 TEORIA DOS CONJUNTOS *FUZZY*

A teoria de conjuntos se baseia no conceito elementar de pertinência, $\mu(x)$, de um elemento "x" a um determinado conjunto. Na lógica clássica, um determinado elemento pertence ou não pertence a um determinado conjunto. Na Lógica *fuzzy*, além deste conceito, se faz necessário indicar o grau de pertinência deste elemento ao conjunto (12). Por exemplo, na Figura 2.1, dado o conjunto Universal de cores, U, suponha que queremos determinar a pertinência dos elementos x_1 e x_2 em relação ao subconjunto de cores vermelha, V, sendo que $V \subset U$. Apesar do elemento x_2 pertencer ao conjunto vermelho, o mesmo não

pode ser dito, com tanta propriedade, do elemento x_1 , visto que o mesmo está numa região entre as cores vermelho e amarelo (alaranjado). Essa questão não interessa para a lógica booleana, e o resultado deve ser $x_1 \in V$ ou $x_1 \notin V$. Isso limita o alcance computacional em solucionar determinados problemas. Nesses casos a lógica *fuzzy* se mostra eficiente, onde o resultado parece impreciso (*fuzzy*) e o convencional não consegue resolver facilmente.

No caso da Figura 2.1, tanto x_1 quanto x_2 pertenceriam ao conjunto V, $\{x_1,x_2\} \in V$, faltando indicar o grau de pertinência de cada elemento - o quão vermelho são os elementos x_1 e x_2. Para determinar essa grau de pertinência os valores 0 e 1 seriam o limite - nada vermelho e totalmente vermelho, respectivamente - podendo existir infinitos valores esses limites (indicando o grau de vermelho). Por exemplo, a pertinência do elemento x_1 pode ser igual a 0.8, $\mu_V(x_1)$, e a pertinência de x_2 igual a 0.4, $\mu_V(x_2)$, como exemplo.

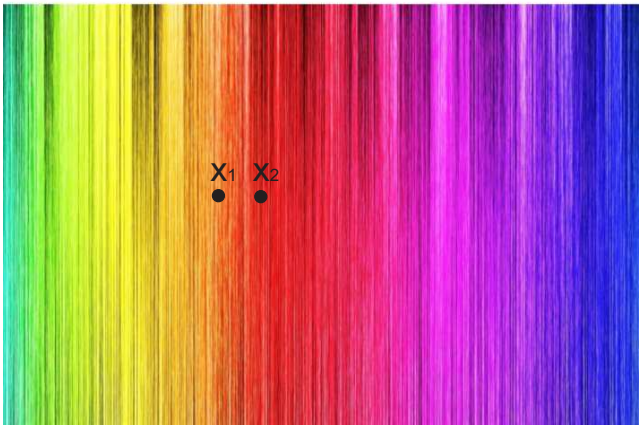


Figura 2.1: Conjunto de Cores - U

2.1.1.1 DEFINIÇÃO DE CONJUNTOS *FUZZY*

Um subconjunto *fuzzy* A de X fica definido por sua função de pertinência μ, que associa a cada elemento de X um grau de pertinência $\mu_A(x)$, com o qual x pertence ao subconjunto A (13), sendo representado pelo par ordenado (12):

$$\{[x, \mu_A(x))], \forall x \in X\} \tag{2.1}$$

No conjunto *fuzzy*, as operações não ocorrem diretamente com seus elementos, mas sim com seus graus de pertinência. Sendo assim, um conjunto A, definido por A = {(x_1,0),(x_2,0.5),(x_3,1),(x_4,1)}, pode ser representado, com clareza, apenas por seu vetor de pertinência: A = {0; 0.5; 1; 1} (12).

Definição: Um conjunto é completamente definido por seu vetor de pertinências (12).

2.1.1.2 PROPRIEDADES FUNDAMENTAIS

A $\cap$ $\bar{A}$ necessariamente não é vazia;

A $\cup$ $\bar{A}$ necessariamente não é o conjunto Universo.

As demais propriedades de conjuntos são validas para os conjuntos *fuzzy* (comutativa, associativa,

distributiva, etc).

2.1.1.3 OPERAÇÕES *FUZZY*

As principais operações do conjunto *fuzzy* são a interseção, a união e o complemento, sendo definidos pelas equações (2.2), (2.3) e (2.4), respectivamente.

$$(\forall x \in X) \Longrightarrow \mu_{\mathrm{A} \cap \mathrm{B}}(x) = min(\mu_{\mathrm{A}}(x), \mu_{\mathrm{B}}(x)) \tag{2.2}$$

$$(\forall x \in X) \Longrightarrow \mu_{\mathrm{A} \cup \mathrm{B}}(x) = max(\mu_{\mathrm{A}}(x), \mu_{\mathrm{B}}(x)) \tag{2.3}$$

$$(\forall x \in X) \Longrightarrow \mu_{\mathrm{A}^{\mathrm{C}}}(x) = \bar{\mu_{\mathrm{A}}}(x) = 1 - \mu_{\mathrm{A}}(x) \tag{2.4}$$

2.1.2 VARIÁVEIS LINGUÍSTICAS

Segundo Campos (13), a variável linguística é definida por três elementos V, X e Tv, em que "V" indica uma variável de um conjunto *X*. Tv é um subconjunto *fuzzy* indicando termos, nomes, rótulos ou, simplesmente, "valores linguísticos" que caracterizam a variável V. A variável linguística é indicada por um substantivo, enquanto o termo linguístico representa um adjetivo/ advérbio que caracteriza essa variável linguística (14) e (15).

No caso do conjunto da Cor Vermelha (V), poderíamos definir como termos linguísticos: T{V} = {claro, escuro}.

2.1.3 FUNÇÕES DE PERTINÊNCIA

A maioria das funções de pertinência que representam os subconjuntos *fuzzy* possuem uma transição suave, sem mudanças abruptas, e as mais utilizadas são as triângulares, trapezoidais, gaussianas, entre outras (13), como representado na Figura 2.2

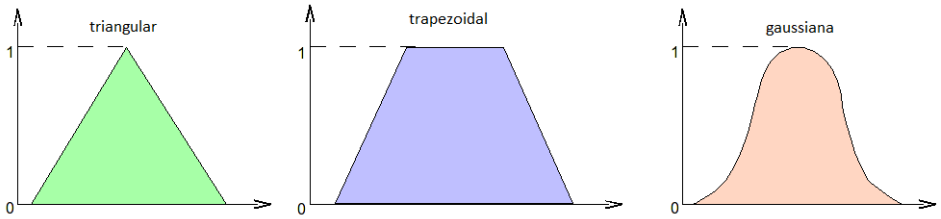


Figura 2.2: Algumas das principais funções de pertinência utilizadas

2.1.4 SISTEMAS *FUZZY*

A implicação conectiva P $\rightarrow$ Q (P implica Q) é uma forma clássica, onde a proposição P também é referida como a hipótese ou o antecedente e a proposição Q também é referida como a conclusão ou o

consequente. Essa premissa pode ser reescrita como indicado na equação (2.5).

$$if P(antecedente), then Q(consequente) \tag{2.5}$$

Essa é a forma mais comum de representar o conhecimento humano e de expressá-lo naturalmente (16).

Ao usar as propriedades básicas e as operações definidas para conjuntos *fuzzy* qualquer estrutura de regra composta pode ser decomposta e reduzida a uma série de regras canônicas simples conforme indicado na Tabela 2.1.

Tabela 2.1: Regras canônicas de sistemas *fuzzy* - Adaptado de (Ross, 2010)

Regra 1:	IF condition C_1, THEN restriction R_1
⋮	
Regra n:	IF condition C_n, THEN restriction R_n

Essa simplicidade, que expressa a linguagem humana e baseadas em conjuntos *fuzzy*, descreve sistemas complexos com um conjunto de restrições de saída (consequentes) baseados em certas condições de entrada (antecedentes), que são conectadas por conectivos “e”, “or” ou “else” (16)

Quando se tem múltiplas regras de entrada (antecedentes) a saída é determinada através da agregação, que pode utilizar a operação de intersecção ou de união para determinar a saída (consequente). Três modelos comumente utilizados para inferência de sistemas *fuzzy* são: (a) Mamdani, (b) Takagi-Sugeno e (c) Tsukamoto.

O método de Mamdani é o mais utilizado na prática e citado na literatura (16). A base de regras segue a equação (2.5) e os “*n*” conjuntos de regras IF–THEN estão de acordo com a forma Mamdani, descrita pela (2.6) (16) e (17).

$$IF\ x_1^k\ AND\ x_2^k\ THEN\ y^k\ é\ B^k,\ for\ k = 1, 2, ..., n. \tag{2.6}$$

Apesar da equação (2.6) representar apenas duas entradas (x_1 e x_2) e uma saída (y) a mesma é válida para múltiplas entradas e/ou saídas.

Na determinação da saída (defuzzificação) pode ser utilizado vários métodos, sendo os mais comuns (15):

- Média dos Máximos: a saída é a média dos máximos valores de pertinência definidos pela inferência;
- Máximo das pertinências: a saída é o maior valor de pertinência inferido no processo;
- Centro da área: a saída é o centro de gravidade de distribuição;

O próximo método, denominado Takagi-Sugeno (TS), foi idealizado tentando sistematizar a geração das regras *fuzzy*. A principal diferença para o método Mamdani é que a saída é uma função dependente das entradas, como apresentado na equação (2.7) (16).

$$IF\ x_1\ AND\ x_2\ THEN\ y\ = f(x_1,x_2) \tag{2.7}$$

onde $f(x_1,x_2)$ é uma função polinomial das entradas x_1 e x_2.

No caso de $f(x_1,x_2)$ ser uma constante, o sistema de inferência é denominado modelo Sugeno de ordem zero, sendo um caso especial do sistema Mamdani no qual cada consequência da regra é especificada como um singleton *fuzzy* (16).

A saída do sistema no método TS é a média ponderada ((18), (16)) definida pela equação (2.8). Essa forma de defuzzificação é realizada de forma direta, sendo computacionalmente mais eficiente quando comparado ao modelo Mamdani (15).

$$\frac{\sum(\mu_i \times f(x_1, ..., x_n))}{\sum \mu_i} \tag{2.8}$$

onde μ indica a menor pertinência das "n" entradas e $f(x_1, ..., x_n)$ a função de saída relativa à regra ativada (k).

Já o método de Tsukamoto é muito parecido como de Takagi-Sugeno, mas tem como consequente funções monotônicas, que, em geral, são utilizadas apenas em casos bastante específicos. Também tem a vantagem da saída ser determinada de forma a evitar a parte de defuzzificação (16).

De acordo com o estudo feito por Zavala, (8), o método de inferência mais comum é o Mamdani, pois sua principal vantagem é ser mais fácil para projetar. Em contrapartida no método de inferência de Sugeno cada regra tem como saída um valor numérico, dispensando o processo de defuzzificação e diminuindo a complexidade computacional. Além disso, os controladores *fuzzy*–(TS) são fáceis de serem caracterizados, dispensando escolhas do tipo de implicação e de defuzzificação (19).

2.2 CONTROLADORES *FUZZY*

2.2.1 INTRODUÇÃO

Nos controladores clássicos o objetivo é determinar o modelo da planta para, então, calcular os parâmetros do controlador que irá automatizá-la – o foco está no modelo do controlador (vide Figura 2.3).

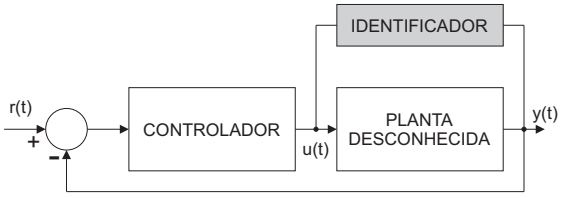


Figura 2.3: Diagrama de blocos de um controlador clássico: a ideia principal é identificar o modelo da planta para determinar os parâmetros do controlador - adaptado (Simões, 2007).

Já o controlador *fuzzy* tem seu foco no operador da planta, que detém o conhecimento para operá-la (vide Figura 2.4). A partir dessas informações, determina-se um modelo *fuzzy* para operar a planta.

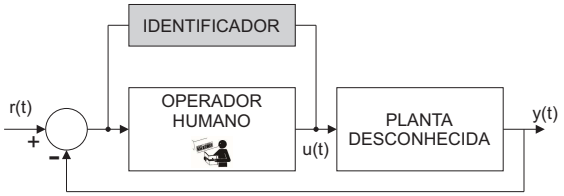


Figura 2.4: Diagrama de blocos de um controlador *fuzzy*: o modelo é baseado em como o operador do sistema controla a planta, identificando informações relativas ao processo de controle para automatizar a planta - adaptado (Simões, 2007).

2.2.2 DEFINIÇÕES E CONCEITOS

Os controladores baseados na Lógica *fuzzy* podem ser representados como indicado na Figura 2.5, sendo seus principais módulos (a) a fuzzificação, (b) a base de conhecimento, (c) a inferência e (d) a defuzzificação.

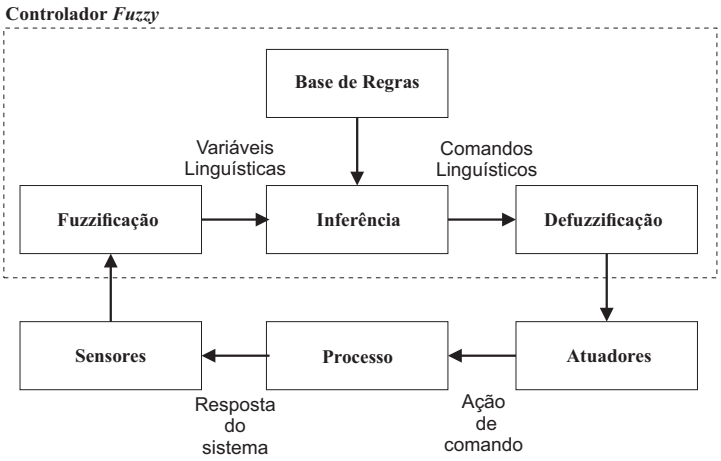


Figura 2.5: Diagrama de blocos de um controlador *fuzzy* - adaptado (Campos, 2004) e (Ross, 2010).

Para que seja possível utilizar uma variável real em sistemas *fuzzy* essas devem ser fuzzificadas, isto é, traduzidas para uma função que possa ser interpretada pelo sistema *fuzzy* (16). Assim, os sinais dos sensores e dos atuadores devem ser convertidos em variáveis linguísticas e seus respectivos termos linguísticos. Nessa fase se define o universo de discurso e são atribuídos a cada termo linguístico a sua função de pertinência. É de fundamental importância a escolha do número de termos linguísticos, visto que um número excessivo pode sobrecarregar o sistema, mas possibilita um ajuste mais fino no controle (13) e (16).

O módulo de inferência utiliza a base de conhecimento para definir o comportamento do sistema e determinar o valor de saída, que pode ou não continuar sendo uma variável linguística, dependendo do tipo de inferência escolhido (ver seção 2.1.4). A base de conhecimento contém o banco de regras que associa as variáveis linguísticas de entrada com as variáveis linguísticas de saída. O desenvolvimento do banco de regras é a parte mais crítica do projeto, pois o desempenho do controlador poderá ser comprometido se não for bem elaborada. O banco de regras pode ser definido através do comportamento do operador do sistema ou pela modelagem do processo e essas informações podem ser obtidas de forma manual ou automática (13).

Em grande parte dos processos se faz necessário que a saída *fuzzy* precise ser única, em oposição a

um conjunto *fuzzy*. Essa etapa é denominada defuzzificação, onde ocorre a conversão de uma quantidade *fuzzy* para uma quantidade precisa, o valor real. Assim como foi dito que a fuzzificação é a conversão de uma quantidade precisa em uma quantidade *fuzzy* (16). Os métodos de defuzzificação já foram tratados na seção 2.1.4.

2.3 SISTEMAS RECONFIGURÁVEIS

Uma tarefa computacional pode ser dividida tanto em software, quanto em hardware e podem se diferenciar quanto ao processamento das informações, pois as aplicações em software são implementadas no processador *RISC* (*Reduced Instruction Set Computer*), usando, por exemplo, a linguagem *C / C++*, enquanto as aplicações em hardware são implementados em circuitos analógicos e/ou digitais, usando linguagem de descrição de hardware, como o *VHDL* (*VHSIC Hardware Description Language*) (20). Aplicações de hardware baseadas em circuitos digitais podem fazer uso de *ASIC*s (*Application specific integrated circuits*), lógica TTL (*Transistor-Transistor Logic*) ou *CPLD*s (*Complex Programmable Logic Devices*). Dentre os CPLDs, as plataformas mais comuns são os sistemas reconfiguráveis, tais como *FPGA*s, que consistem numa matriz de blocos lógicos reprogramáveis que podem ser interconectados fisicamente de forma a obter o resultado desejado (21).

Atualmente, muitos sistemas de tempo real necessitam de alta capacidade computacional, aliando flexibilidade (inerentes aos dispositivos baseados em softwares), velocidade e capacidade de processamento paralelo, o que são características inerentes das arquiteturas reconfiguráveis (21) e (22). Ao se comparar processadores RISC e DSP (*Digital signal processing*), com um FPGA em aplicações para filtros do tipo *FIR* (*Finite Impulse Response*) e *IIR* (*Infinite Impulse Response*) (23) e criptografia (24) mostram o alto desempenho das FPGAs, em relação aos outros dois (taxa de transferência), só perdendo em consumo energético para o DSP, em aplicações com filtros, como pode ser observado na Tabela 2.2.

Tabela 2.2: Comparação do rendimento e da eficiência energética entre RISC, DSP e FPGA para o algoritmo de criptografia IDEA, e dos filtros digitais do tipo FIR e IIR - Adaptado (Pless, 2003).

Type	Device	IDEA		FIR		IIR	
		Mbit/s	Mbit/s	Mtap/s	Mtap/Ws	Mtap/s	Mtap/Ws
RISC	Strong ARM AS-110	32.0	32.0	9.9	26.7	1.5	3.6
DSP	TI TMS320C6x	53.1	8.9	-	-	-	-
	TI TMS320C2xx	-	-	20.0	769.2	1.0	52.4
FPGA	Xililnx XC4020XL	528.0	167.9	-	-	-	-
	Xililnx XC40003A	-	-	30.0	454.5	2.1	9.7

Apesar de o hardware dedicado ter um desempenho muitas vezes maior que os hardwares reconfiguráveis, devido ao seu elevado grau de especialização, ele é pouco flexível, sendo este o forte dos processadores baseados em software. Mesmo assim, estudos mostram que quando o processo pode ser combinado em estruturas paralelas, a eficiência dos sistemas reconfiguráveis são muito superiores e, em alguns casos, podem chegar a ser mais eficientes energéticamente do que os DSPs (vide colunas FIR e IIR da Tabela 2.1).

Na última decada tem se difundido o uso de dispositivos reconfiguráveis baseados em sistemas em chip (SoCs), os sao circuitos integrados com alto grau de integração, incorporando em um único chip elemetos tais como FPGAs, processadores embarcados, blocos de memória, blocos de interface, blocos analógicos e componentes que lidam com funções de processamento de aplicação específica (22), (25) e (26).

2.3.1 FPGAs e SOC

FPGA é um dispositivo lógico que contém uma matriz bidimensional de células lógicas, que podem ser interligadas entre si para executar uma função. Essa interligação é realizada por switches programáveis, como mostrado na Figura 2.6 (27).

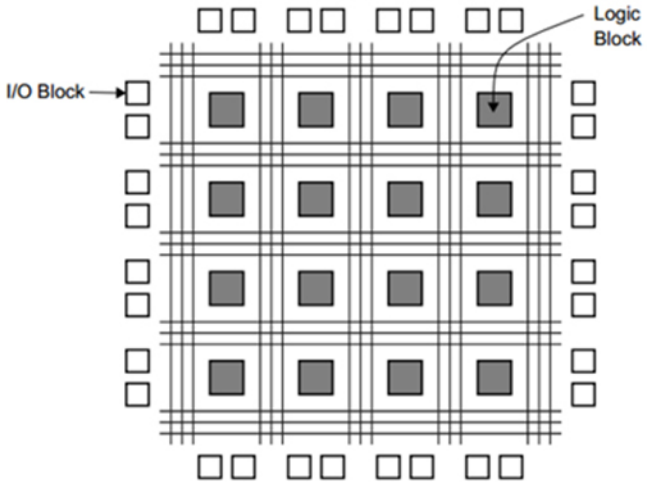


Figura 2.6: Estrutura geral de um FPGA - (CHU, 2008).

As células lógicas são o elemento básico de construção dos projetos em FPGA, pois as mesmas podem implementar circuitos combinacionais ou sequenciais, mas apesar de sua importância, os blocos lógicos (que são grupos de células lógicas com finalidade específicas, como carry, adicionadores e subtratores) são mais representativos, pois estão organizados estrategicamente e interligados com vias de comunicação mais rápidas, resultando em projetos com redução de atrasos e com melhor desempenho (28). Adicionalmente, os FPGAs atuais contam com blocos de Processamento Digital de Sinais (DSP) e blocos de memória RAM (RAM blocks) que permitem a elaboração de circuitos mais complexos. Outra definição, então, pode ser: FPGA é um dispositivo lógico programável, que consiste de uma matriz bidimensional de blocos lógicos ligados por uma rede de encaminhamento programável, como mostrado na Figura 2.7.

Os fabricantes de FPGAs, visando atender ao mercado, incorporaram circuitos integrados de aplicação específica (ASIC) tais como processadores de software e conversores de dados ADC, formando os sistemas SoC. Na Figura 2.8 podemos verificar a inclusão de vários dispositivos, mantendo a base como blocos lógicos que interagem com os demais elementos do chip. A adição destes dispositivos serve para fins específicos e, apesar de poderem ser implementados dentro da lógica reconfigurável, é mais eficiente em termos de desempenho e consumo energético, incorporá-los on-chip (28).

Dentre estes elementos ASIC, destacam-se processador de software que aumentam a flexibilidade das soluções e conversores de dados ADC ou DAC; entre outros. Todos visam melhorar os projetos, a fim de

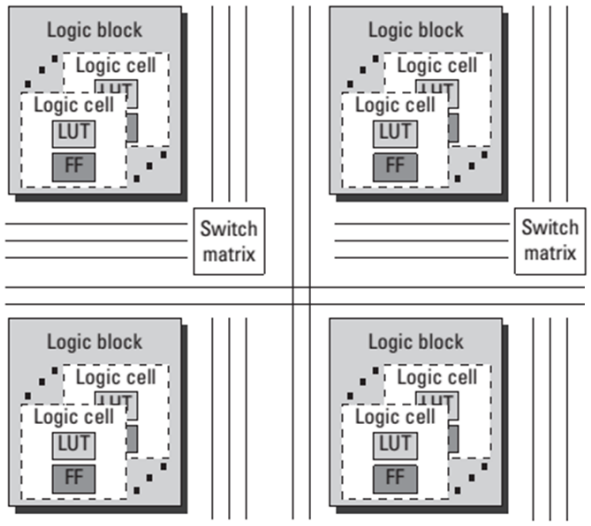


Figura 2.7: Diagrama de Blocos de um FPGA, mostrando as células lógicas, elementos de armazenamento, geradores de função e lógica de roteamento (Sass, 2010).

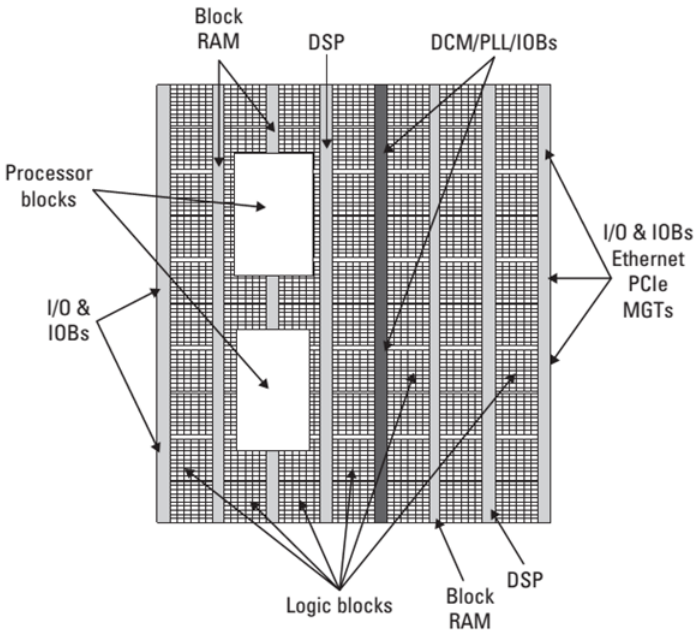


Figura 2.8: Visão de uma Plataforma FPGA (SoC) - (Sass, 2010).

liberar recursos da lógica programável e diminuir o consumo de energia (28).

2.4 ESTADO DA ARTE

Pesquisas relacionadas a pessoas com deficiência visual são extensivamente estudadas desde os anos 70 e, ultimamente, se destacam os dispositivos com sensores baseados em visão (29), (30), (31), (32) e (33), obtenção de informações do ambiente através de etiquetas ou RFID-tags (34), (35), (36) e (37), laser (38) e o cão-guia robô (39), (40) e (41). É nessa área do cão-guia, também, onde se encontra a maioria dos trabalhos combinados com a Lógica *fuzzy*, que se baseia na experiência humana para as tomadas de decisão. Grande parte das soluções apresentadas atualmente são onerosas, ineficientes em relação ao consumo energético, relativamente grandes e utilizam apenas um elemento sensor. Os que possuem dimensões pequenas possuem um baixo poder computacional e poucos são os trabalhos que utilizam a lógica *fuzzy* para solucionar problemas relacionados a tomadas de decisão por parte do deficiente visual. Entre esses trabalhos se destaca o de Wang (42), onde a detecção de obstáculos se baseia no controle *fuzzy* e na utilização de múltiplos sensores ultrassônicos.

Foi no controle automático onde a lógica *fuzzy* teve sua maior atuação (8). O primeiro trabalho foi realizado por Mamdani, em 1972, aplicando técnicas de controle, baseado na lógica *fuzzy*, em uma máquina a vapor. A saída do controle foi inferida usando uma base de regras (17). Em 1984, Sugeno apresentou um projeto de estacionamento automático onde o esquema de controle *fuzzy* tinha como consequentes equações lineares de primeira ordem ((18)). Atualmente, os sistemas de controle baseados na lógica *fuzzy* são bastante utilizados em várias aplicações como máquinas de lavar roupa – aumentando seu desempenho –, controle de tráfego ferroviário – como o japonês, e em sistemas industriais para controlar: (a) Purificação de água, (b) Reatores nucleares, (c) Robótica, (d) Reconhecimento de padrões, entre outros (8).

Zavala e Camacho (8), em 2012, fizeram um estudo sobre o uso de Lógica fuzzy no controle de sistemas com implementação em hardware (analógico e/ou digital) desde 2012 (vide Tabela 2.3). Foram listadas as características de mais de 50 trabalhos que utilizavam a lógica fuzzy, como: (a) Tecnologia: *VLSI* (*Very Large Scale Integration*), *FPGA*, *CMOS* (*Complementary Metal Oxide Semiconductor*), entre outros; (b) Arquitetura (serial, paralela ou mista); (c) Tipo de inferência; (d) Número de entradas; (e) Número de termos linguísticos por entrada; (f) Número de saídas; (g) Número de regras; (h) Tamanho das palavras (bits); entre outros. De acordo com o levantamento é possível inferir algumas características, como: (a) O número de entradas varia de duas a oito; (b) O número de saídas varia de uma a quatro; (c) O número de termos linguísticos varia num máximo 16 para as entradas; e (d) 256 singletons para a saída; (e) A sobreposição de termos linguísticos é limitado a dois. Vinte e quatro destes trabalhos foram desenvolvidos em FPGAs, sendo maioria em código VHDL e utilizando a técnica de pipeline, que garante uma considerável aceleração do tempo de execução (43).

Tabela 2.3: Estudo das principais características dos processadores *fuzzy* implementados em hardware - adaptado (Zavala, 2012).

Reported Years	References	Technology Used	Architecture	Speed on FLIPS	Inference Type	Defuzzification Method	Design Method	Number of Inputs	MF's allowed per input	Number of Outputs	MF's allowed per output	Number of Rules	Bits per Rule	Clock Cycles for an inference	Overlapped MF's	Discourse Universe Resolution	Membership Universe Resolution	Comments
1985-1986	[18] [41] [42]	CMOS	Parallel - Serial	80K	Max-Min	COG	---	2	31	1	31	16	124	256	2	5-6	4	ROM memory usage to store rules. All rules are executed in parallel but each rule is processed sequentially.
1986-1995	[19] [43]-[46]	Analog (Voltage)	2 dedicated processors	1M	Max-Min	COG	---	3	7	1	7	49	---	---	2	---	---	Processing based on voltage levels handling.
1987-1988	[25]	Mixed-Signal	Systolic Multiprocessor	---	Total relationship	---	---	2	7	1	7	150	---	---	2	---	---	Previously calculates the values for a fuzzy total relationship.
1990-1991	[47] - [49]	VLSI	Dynamically reconfigurable cascadable	580K	Max-Min	COG	High level C	2	7	1	7	102	12	18	2	6	4	Processes 102 rules in parallel. RAM rule Storage
1991	[40]	VLSI	Parallel with pipeline	---	Max-Min	Variable	---	9	8	1	8	64	---	---	2	8	6	Introduces the active rule driven schema and uses RAM memory to store Membership functions.
1992	[27]	Mixed Signal	Pipelined Josephson	790M (simulated)	Max-Min	COG	Schematic	2	---	1	---	4	---	---	2	5	4	COG defuzzifier realized with comparisons without additions or multiplications.
1992	[20]	VLSI	RISC + FALU	---	Variable	Variable	---	Variable	2-16	Variable	2-16	---	---	---	2	4	6	Introduces the Fuzzy Arithmetic and Logic Unit FALU which includes vector instructions for fuzzy calculation.
1993-1995	[50] - [52]	FPGA	3 stage pipelined	---	Max-Min	COG	Schematic	4	8	1	8	256	15	---	2	8	8	Uses a Look up table LUT to realize Fuzzification.
1993	[53] - [54]	VLSI	Co-processor	200K	Max-Min	COG	---	2-4	6	1	7	15000	---	100	2	16	12	Includes a membership function generator.

1998	[72]	FPGA	Pipelined	---	Max-Min	COG, MOM	AHDL	3	5	3	5	---	---	---	8	Selectable defuzzifier and shape of membership function.
1999-2000	[73] [74]	FPGA	2 stage pipelined	---	Max-min	COG	VHDL	2	9	1	7	---	---	12	2	7-16	4-13	Rule association matrix
2000	[75]	FPGA	Multi-Stage	---	Max-Min	Center Average	Verilog XL	5	---	2	---	294	16	128	---	10	4	High Speed Multi-Stage Fuzzy Hardware HSMFH
2002	[76]	FPGA	Pipelined	818K	Max-min	COG	HDL	2	7	1	7	49	---	---	2	8	8	Automatic synthesis of a FLC based on ANN
2002	[77]	Mixed-Signal	Neuro-fuzzy Parallel	---	Cascade-Sum	COG	Schematic	3	5	1	5	125	---	---	---	---	---	Current mode inference engine with digital registers to store parameters given via I^2C.
2003	[39]	FPGA	Pipelined	170K	Takagi-Sugeno	COG	VHDL	3	3	1	---	27	---	---	---	---	---	Rule base as petri-nets distribution
2003	[78]	FPGA	Parallel and sequential	45M	Max-Min	CAD	VHDL	2	4	1	4	16	---	---	2	8	8	Sequential inference. Distributed arithmetic for defuzzification
2003	[79]	Mixed-signal	Parallel	0.5M	Mamdani	COG	Schematic	2	3	1	---	9	---	---	2	0-32 uA	0-16uA	Linguistic Hedge approach that allows good performance with minimal resources
2004	[80] [81]	Digital	6 stage Pipelined	4M	Max-Min	COG	---	8	8	4	8	1024	---	96	2	8	4	Scheduling unit for improvements on active rule detection.
2005	[82]	FPGA	Multi-cycle	31K	Min-Max	COG	VHDL	2	7	1	7	49-343	8	36	---	7	4	Multi-cycle execution

1998	1998	1996	1995-2002	1997	1997-2000	1996	1996	1996	1995
[71]	[32]	[70]	[67] - [69]	[28] [66]	[21] [64] [65]	[23]	[38]	[63]	[62]
Analog	VLSI	VLSI	FPGA	Digital	FPGA	Voltage mode CMOS	VLSI	FPGA	FPGA
Parallel	Pipelined	Direct Data Stream	Parallel	Ternary Neural	Pipelined RISC+ Fuzzy	Parallel	Pipelined	Pipelined	Parallel pipelined
500K	400K	300M	500K -3M	---	Variable	6M	86M	---	9M
Max-Min	Max-Min	Max-Min	Variable	Max-Min	Variable by Software	Max-Min	Max-Min	Max-Min	Max-Min
COG	COG with LUT	---	COG optimized	COG	Variable	COG	MOA / COA	Yager	COG
---	Cadence DFWII	GDT Designers	VHDL	---	VHDL	HSPICE	---	Statecharts VHDL	---
2	2	4	3	2	variable	3	4	2	2
7	16	Variable	8	---	Variable	3	7	7	3
1	1	2	1	1	Variable	1	1	1	1
4	7	Variable	8	---	Variable	5	8	7	9 Singleton
49	256	---	---	20	---	13	64	---	9
---	---	---	---	---	---	---	18	---	---
---	---	---	9	---	---	---	---	---	24
2	2	Variable	2	---	Variable	2	2	2	2
---	8	16	8	---	Variable	---	8	8	6
---	8	6	6	---	Variable	---	6	8	4
Voltage input - Current output Operational transconductance Amplifiers	Trapezoidal membership functions shapes with power of two slopes.	DDS direct data stream to accelerate calculations.	Inferences are calculated totally in parallel.	Use of tri-valued logic and neural Networks for maximum operations.	Uses a MIPS-I processor together with an FPGA fuzzy processor based on cooperative state machines.	Stages realized with voltage operations	Any shape for membership functions by usage of a LUT	Fuzzy system is represented by state diagrams.	Use of an EPROM as LUT for defuzzifier avoiding multiplication and division

Trabalhos envolvendo o auxílio para locomoção para deficientes visuais utilizando FPGAs são em sua grande maioria aplicados ao uso de visão estéreo ((44), (45) e (46) e (47)) e alguns com o uso de sensores ultrassônicos, como (48).

Existem alguns geradores automáticos de código VHDL de sistemas *fuzzy* como o *Digital Fuzzy Logic Controller* (DFLC) que trabalha com palavras com resolução de 8 bits na entrada e de 12 bits na saída. O sistema está limitado a duas variáveis de entrada, com no máximo sete termos linguísticos cada (8 bits de resolução) o que possibilita ter no máximo 49 base de regras. A inferência é baseada no modelo Takagi-Sugeno de ordem zero com a defuzzificação sendo feita pela Média Ponderada (49).

Outro gerador de códigos VHDL de sistemas *fuzzy* bastante conhecido e difundido na literatura é o *xFuzzy* com várias citações em projetos e livros (50). Esse sistema integra várias ferramentas para o desenvolvimento de projetos *fuzzy* que vão desde a descricao inicial de forma gráfica, passando pela simulação e a sintese em C, C++, Java e VHDL. Adicionalmente essa ferramente permite o desenvolvimento de projetos *fuzzy* do tipo Mamdani e Sugeno. As limitações do gerador de codigo VHDL do xFuzzy consistem: (a) na utilização de ponto fixo para representação numérica, (b) as entradas e as saídas devem possuir o mesmo número de termos linguísticos e tipo das funções de pertinência e (c) de permitir a sobreposição de no máximo duas funções de pertinência. A ferramenta encontra-se disponível no link <http://www2.imse-cnm.csic.es/Xfuzzy/>.

2.5 CONCLUSÕES DO CAPÍTULO

Nesse capítulo foi realizada uma revisão dos principais tópicos que serão utilizados no decorrer da dissertação: (a) Lógica *fuzzy*, (b) Controladores *fuzzy* e (c) Sistemas Reconfiguráveis. Assim como uma pesquisa sobre o atual estado da arte referentes a Lógica fuzzy, a implementação em descrição de hardware e geradores automáticos de código VHDL.

Capítulo 3

PROJETO DO CONTROLADOR *FUZZY* TAKAGI-SUGENO PARA DESVIO DE OBSTÁCULOS

Neste capítulo serão utilizados os conceitos formulados na fundamentação teórica para projetar um controlador *fuzzy* do tipo Takagi-Sugeno (TS) aplicado ao auxilio da locomoção de deficientes visuais em ambientes fechados. Na definição do controlador devem ser definidos os parâmetros que identificam a estrutura do sistema, como o tipo de controlador, as variáveis de entrada e saída, as variáveis linguísticas, os termos linguísticos, as funções de pertinência e a base de regras. Estas escolhas devem estar de acordo com a proposta descrita na seção 1.4.

3.1 FUZZIFICAÇÃO

As variáveis de entrada do sistema serão os sinais dos cinco sensores ultrassônicos e as variáveis de saída serão os sinais dos quatro atuadores piezoeléctricos, como indicados na Tabela 3.1. Nessa tabela também constam os símbolos que representam cada sensor/ atuador e o universo de discurso das variáveis de entrada/ saída.

Tabela 3.1: Variáveis de entrada (sensores) e saída (atuadores) do sistema com seus respectivos símbolos e universo de discurso.

Sensor/ Atuador	Símbolo	Universo de Discurso
Sensor 1	S1	2–400 cm
Sensor 2	S2	2–400 cm
Sensor 3	S3	2–400 cm
Sensor 4	S4	2–400 cm
Sensor 5	S5	2–400 cm
Atuador 1	A1	0–100
Atuador 2	A2	0–100
Atuador 3	A3	0–100
Atuador 4	A4	0–100

Essas variáveis devem ser fuzzificadas para que possam ser utilizadas pelo controlador, para isso, os va-

lores reais das variáveis de entrada serão representadas pela variável linguística “distância”, cujo universo de discurso está definido entre 2 e 400 cm (sensores ultrassônicos HC–SR04). Esse universo de discurso foi dividido em termos linguísticos, como indicados na Tabela 3.2. Cada um desses termos linguísticos deve estar relacionada a uma função de pertinência. Definiu-se que os termos seriam melhor representados pelas funções trapezoidais, como indicados na Figura 3.1.

Tabela 3.2: Termos línguisticos da variável de entrada (distância) com seus respectivos símbolos

Termo *fuzzy*	Símbolo
Muito Perto	MP
Perto	P
Longe	L

É importante ressaltar que a escolha da melhor função de pertinência, assim como os ajustes dos seus respectivos parâmetros (inclinações e interceptos) podem ser feitos após a implementação inicial através de um processo de otimização.

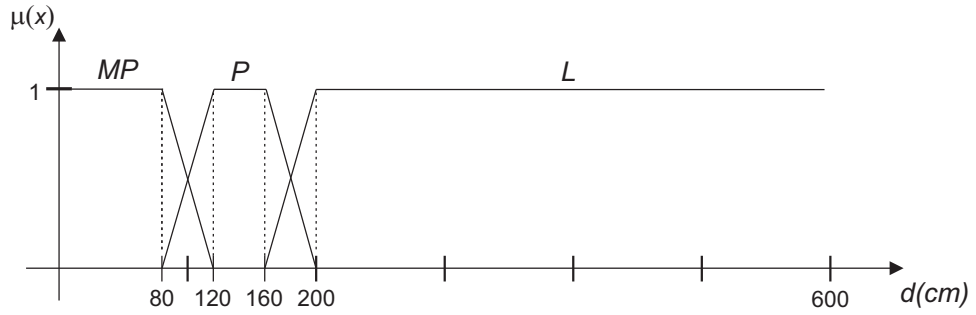


Figura 3.1: Funções de pertinência dos termos linguísticos de entrada (distância): todas as funções de pertinência são do tipo trapezoidal.

Também é necessário que os sinais de saída dos sensores piezoelétricos sejam representados por uma variável linguística. Como a vibração dos sensores piezoelétricos é controlada por um sinal *PWM*, cuja intensidade pode variar de 0 a 100% através da modificação do valor do *duty cycle*, esse sinal será utilizado como a variável linguística de saída. O universo de discurso será definido entre 0 e 100%. Na maioria das aplicações, como a saída do controlador *fuzzy*–TS é uma função dependente das entradas, não é necessário definir termos linguísticos para a saída. No presente caso, onde a saída será representada por funções constantes, o uso de termos linguísticos para a variável PWM irá facilitar (a) o preenchimento da tabela relativa à base de regras e (b) possíveis alterações em seus valores. Sendo assim, os termos linguísticos da variável de saída, seus símbolos e os respectivos valores numéricos estão representados na Tabela 3.3.

Tabela 3.3: Termos linguísticos da variável de saída (PWM) com seus respectivos símbolos e valores numéricos

Termo *fuzzy*	Símbolo	Valor
Desligado	OFF	0
Baixo	B	30
Médio	M	60
Forte	F	90

3.2 INFERÊNCIA E DEFUZZIFICAÇÃO

Para o caso particular, onde cada uma das cinco entradas possuem três termos linguísticos, existem 243 possibilidades que formarão a base de conhecimento. Essas possibilidades são apresentadas no Apêndice I. A partir da inferência desse conjunto de condições de entrada o controlador irá gerar ações de controle (12). Essas ações são as saídas da defuzzificação, que no caso do controlador *fuzzy*–TS é feita de forma direta, pois a saída da equação 2.8 é um valor numérico real, e não uma variável linguística.

De acordo com a vibração dos atuadores piezoelétricos o usuário saberá qual é a direção a seguir. A Tabela 3.4 indica quais as direções possíveis, de acordo com as ativações dos sensores.

Tabela 3.4: Acionamento dos atuadores indicando qual a direção a seguir pelo deficiente visual para desviar dos obstáculos.

Siga	Pare	Esquerda	Direita	Diagonal Esquerda	Diagonal Direita
A1 ● ● A3	A1 ● ● A3	A1 ● ○ A3	A1 ○ ● A3	A1 ● ○ A3	A1 ○ ● A3
A2 ○ ○ A4	A2 ● ● A4	A2 ● ○ A4	A2 ○ ● A4	A2 ○ ● A4	A2 ● ○ A4

3.3 SIMULAÇÕES E VALIDAÇÃO DO CONTROLADOR EM CÓDIGO ESTRUTURADO

O controlador *fuzzy*–TS projetado nesse capítulo foi implementado no toolbox *fuzzy* do *Matlab*® (vide Figura 3.2).

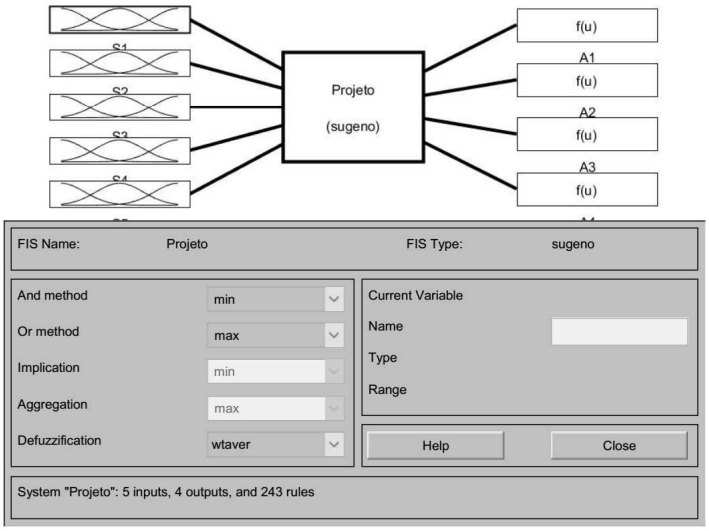


Figura 3.2: Projeto desenvolvido no toolbox *fuzzy* do *Matlab*®. Pode-se verificar o número de entradas, o número de saídas, o tipo de inferência utilizada, o tipo AND (MIN), o tipo OR (MAX) e o tipo de defuzzificação (wtaver = média ponderada).

O objetivo dessa implementação é validar um projeto desenvolvido em código estruturado, que será

usado como modelo e para validar o projeto desenvolvido em arquiteturas de hardware. Os dois projetos desenvolvidos foram comparados de forma a se obter o erro quadrático médio no intuito de validar o código estruturado (vide Figura 3.3).

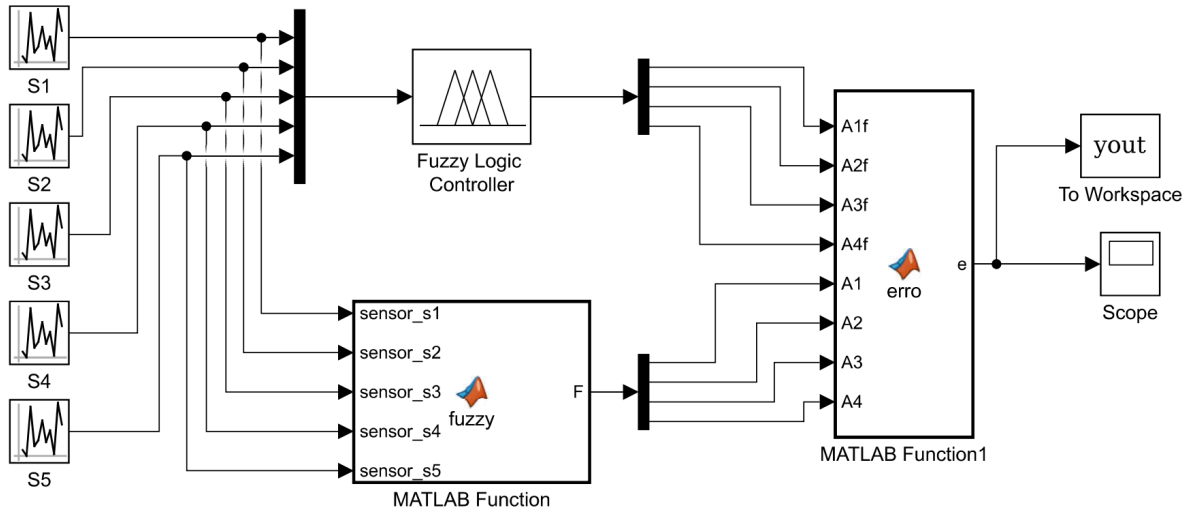


Figura 3.3: Projeto desenvolvido no toolbox *Simulink* do *Matlab*®. As entradas são geradas de forma randômica e a saída yout é a diferença entre os sinais do bloco *fuzzy* e os sinais do bloco Matlab Function. Esse valor é usado para determinar o erro quadrático médio.

A figura 3.4 apresenta a saída do sistema desenvolvido no toolbox *fuzzy*.

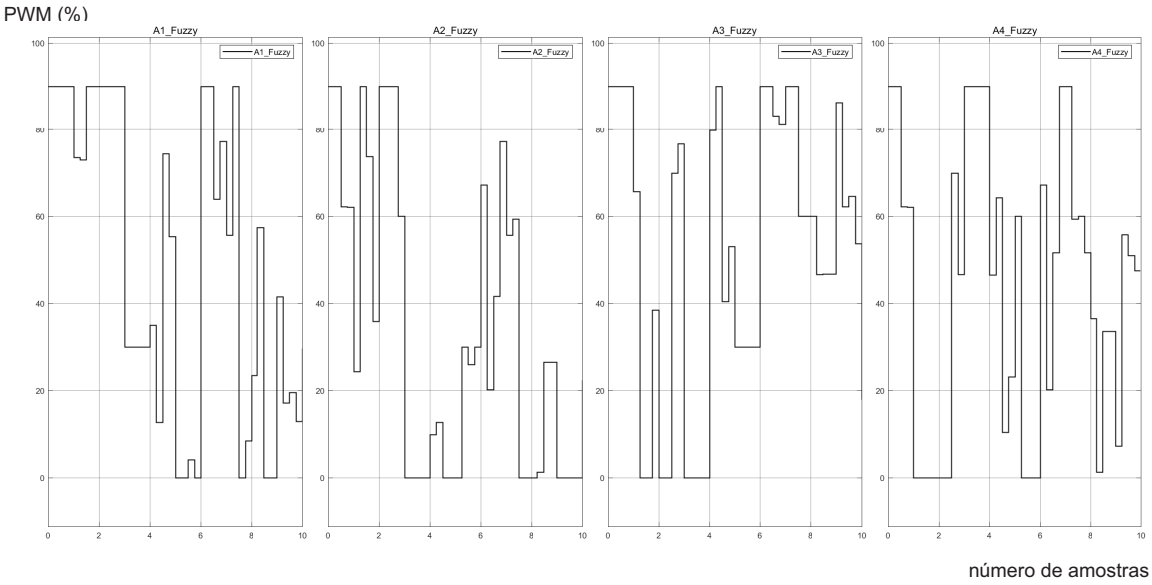


Figura 3.4: Gráfico demonstrando a saída do controlador *fuzzy*-Ts desenvolvido no toolbox *Simulink* do *Matlab*®.

Já na figura 3.5 está apresentado a saída do sistema desenvolvido em código estruturado *fuzzy*.

Observando as figuras verifica-se que as saídas são praticamente idênticas. Para confirmar essa hipótese foram gerados 86400 amostras randômicas para determinar o erro entre os dois sistemas. No gráfico apresentado na Figura 3.6 é possível verificar o erro a cada amostragem. A melhor forma de se obter

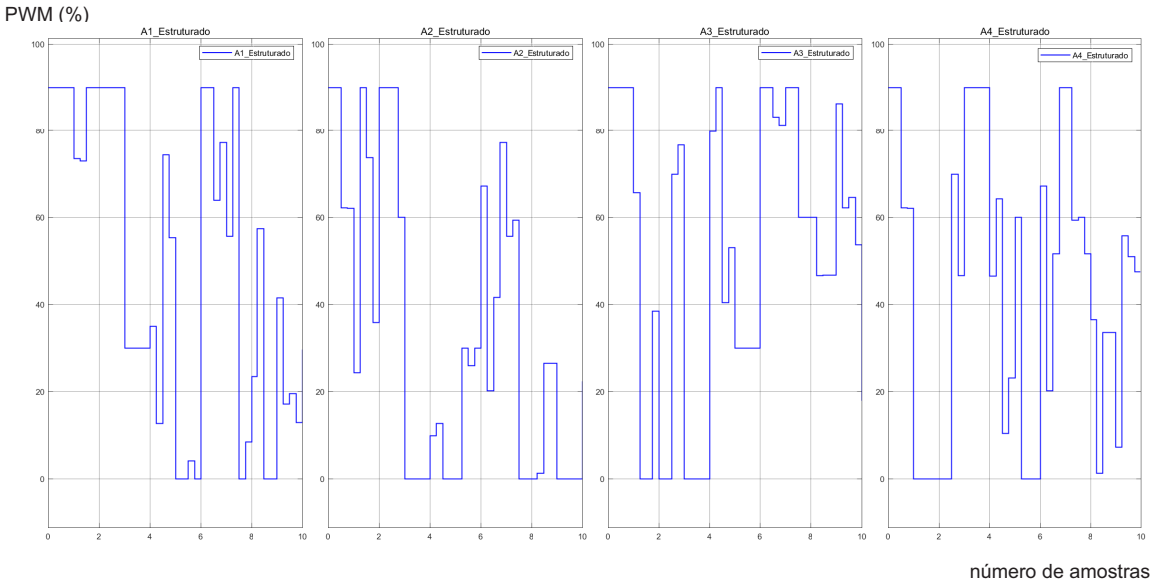


Figura 3.5: Gráfico demonstrando a saída do controlador *fuzzy*–Ts desenvolvido em código estruturado.

o valor do erro é utilizando o erro quadrático médio, visto que os dados possuem valores negativos e positivos. Esse valor foi calculado em 3.5182×10^{-33}.

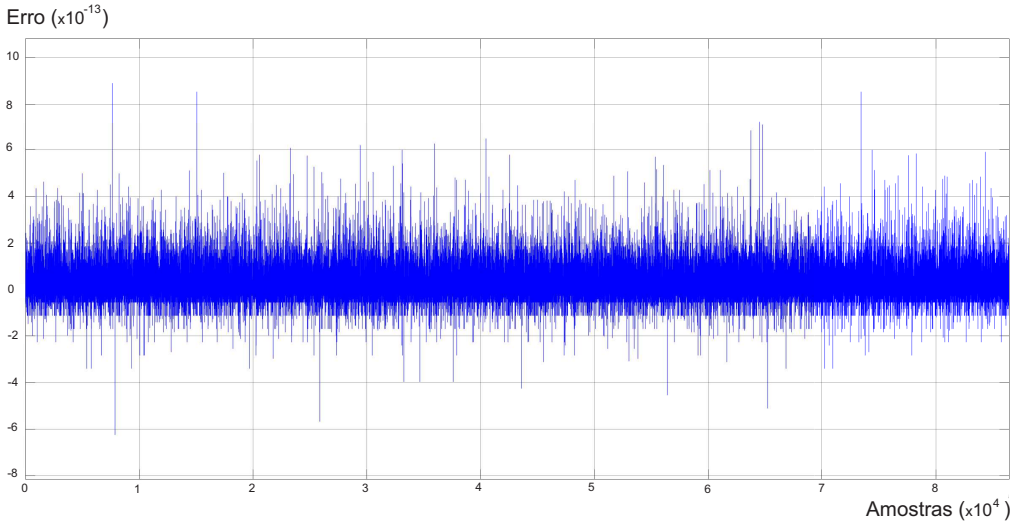


Figura 3.6: Erro para 86400 amostras, com o erro quadrático médio igual a 3.5182×10^{-33}.

3.4 CONCLUSÕES DO CAPÍTULO

Nesse capítulo foi apresentado o desenvolvimento do controlador *fuzzy*–TS e uma implementação usando o toolbox *fuzzy* do Matlab. Esse projeto serviu para validar o código em texto estruturado, que é utilizado como referência e modelo para o desenvolvimento da arquitetura em hardware do controlador *fuzzy*–TS. O baixo valor do erro quadrático médio confirma que o projeto desenvolvido em código estruturado está de acordo com o esperado pelo toolbox *fuzzy*.

Capítulo 4

IMPLEMENTAÇÃO EM FPGA DO CONTROLADOR FUZZY TAKAGI-SUGENO

Este capítulo apresenta as arquiteturas implementadas em hardware do controlador *fuzzy*–TS modelado no capítulo precedente. O modelo do controlador TS foi implementado em duas configurações: paralela e sequencial, no intuito de comparar o tempo de execução dos cálculos e a utilização de recursos de hardware. Nesse capítulo foi utilizado o conceito de "regra ativa" onde são processados apenas os antecedentes que realmente são necessários (estão ativos), economizando cálculos desnecessários e recursos tais como blocos de DSPs (8). Essa solução foi explorada em alguns trabalhos (51), (52) e (53).

Esse projeto tem como diferencial a utilização de ponto flutuante para realização dos cálculos, que foram desenvolvidos pelos pesquisadores do grupo LEIA – GRACO (Grupo de Automação e Controle) do Departamento de Engenharia Mecânica (ENM) da Universidade de Brasília. Os IPCores trabalham com representação aritmética em ponto flutuante usando o padrão IEEE754 ((6), (54), (55), (56) e (57)) e têm sido utilizados em diversas aplicações tais como controle, filtragem de sinais, aprendizagem de maquina, identificação de sistemas, entre outras. A utilização des tes IPCores garantem uma flexibilidade no tamanho das palavras utilizadas e, como os blocos DSPs da Artix 7 são de 9x9 ou 18x18, é interessante a utilização de múltiplos desses valores buscando um menor consumo de recursos.

4.1 PLATAFORMA NEXYS4 - ARTIX7

Para realização da implementação das arquiteturas de hardware desenvolvidas foi utilizada a plataforma Nexys4 DDR para desenvolvimento de circuitos digitais, que possui um FPGA de média capacidade (Artix-7 XC7A100T-1CSG324C), conversor ADC on *chip* (*xadc*) de 1 *MSPS* (*Mega Samples per Second*), memórias externas, USB, Ethernet, entre outros, podendo trabalhar desde projetos de circuitos combinacionais simples até potentes processadores incorporados. Possui, também, vários periféricos integrados, incluindo um acelerômetro, um sensor de temperatura, um microfone digital, um amplificador de alto-

falante e vários dispositivos de E/S que permitem que o Nexys4 DDR seja usado para uma ampla gama de projetos sem a necessidade de outros componentes (58). Na Artix-7 vem inclusos:

- 4860 Kbits de *memory* RAM;
- 63400 *look-up-table* (LUT);
- 126800 *Flip-Flop* (FF);
- 240 *Digital Signal Processing* (DSP).

4.2 ARQUITETURAS PARALELA E SEQUENCIAL DO CONTROLADOR *FUZZY*–TS PROPOSTO

A arquitetura paralela do controlador *fuzzy*–TS está apresentada na Figura 4.1.

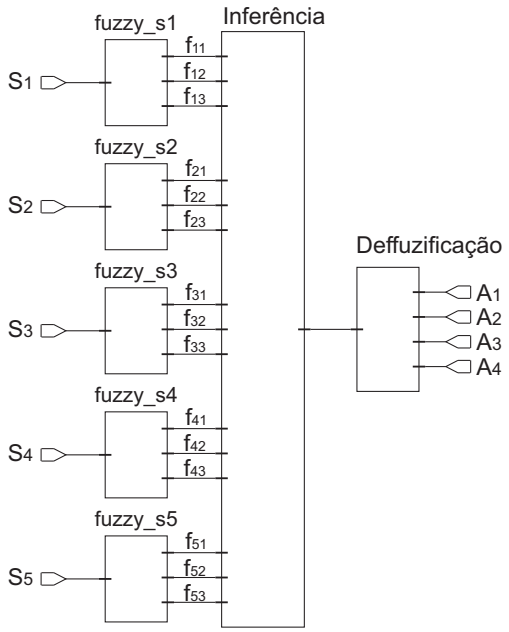


Figura 4.1: Arquitetura geral do controlador TS paralelo. A arquitetura faz uso de cinco blocos *fuzzy* – um para cada sensor – juntamente com o *bloco de inferência* e o *bloco de defuzzificação*

Essa arquitetura possui um bloco de fuziffficação para cada sensor (S_1, ..., S_5) e cada bloco tem como saída três valores de pertinência (f_{i1}, ...,f_{i3}), relativos aos três termos linguísticos de cada entrada. O *bloco de inferência* recebe um total de 15 valores de pertinências que, combinados e de acordo com a base de regras, irá determinar as quatro saídas do sistema (A_1,...,A_4).

Observando a arquitetura sequencial, mostrada na Figura 4.2, verifica-se a utilização de apenas um *bloco de fuzzificação*, que será ativado e reutilizado por uma máquina de estados finitos (FSM - *Finite State Machine*). A FSM controla o início dos cálculos das pertinências e o armazenamento nas respectivas variáveis de cada sensor (*f11*, *f12*, ..., *f53*). Por ser feita de forma sequencial espera-se que essa etapa seja até 5 vezes mais lenta que no modo paralelo.

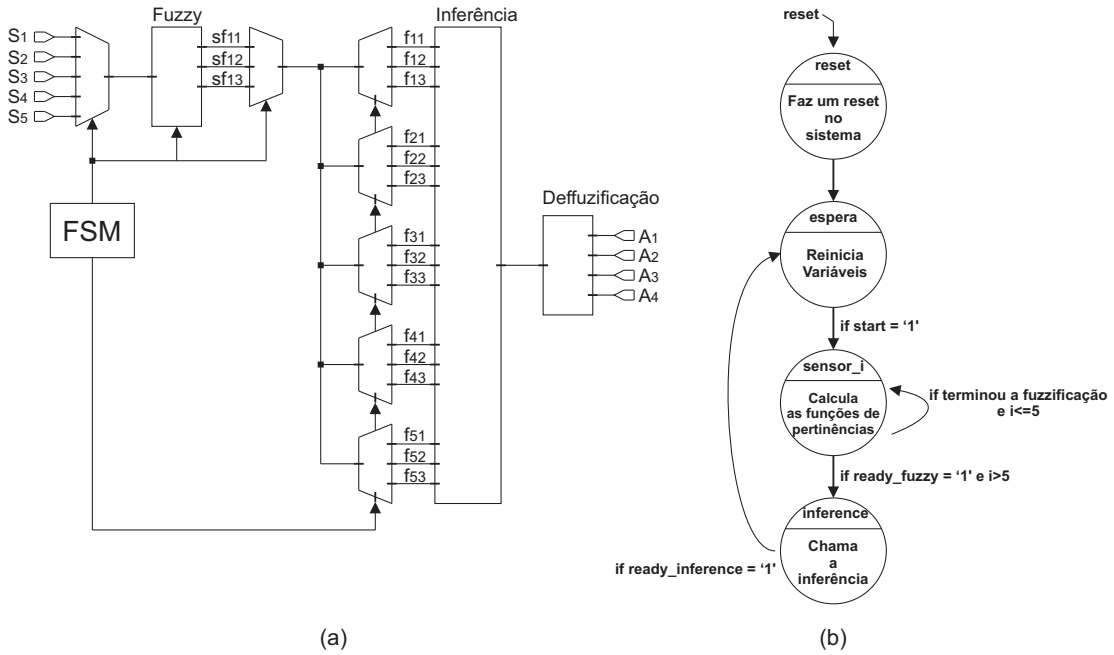


Figura 4.2: (a) Arquitetura geral do controlador TS sequencial. (b) FSM que irá controlar a reutilização do único *bloco de fuzzificação* e o *bloco de inferência*.

Ambas as arquiteturas possuem o mesmo *bloco de inferência*, pois devido a escolha de se priorizar o consumo de recursos não foi possível realizar todas as tomadas de decisões de forma paralela. Isso se deve pela limitação do número de blocos de DSPs para efetuar todos os cálculos necessários de forma simultânea.

Nas próximas seções serão explicados o funcionamento dos blocos de *fuzzificação* e de *inferência*.

4.3 BLOCO DE FUZZIFICAÇÃO

Cada variável linguística de entrada possui três termos linguísticos e o *bloco de fuzzificação* será o responsável por calcular os valores numéricos da pertinência para cada termo linguístico das entradas do sistema. Ao todo serão quinze valores de pertinência calculados ($f_{11}, f_{12}, ..., f_{53}$). Para determinar o valor da pertinência de cada termo linguísto é necessário dividir a função de pertinência trapezoidal em regiões, como apresentado na Figura 4.3.

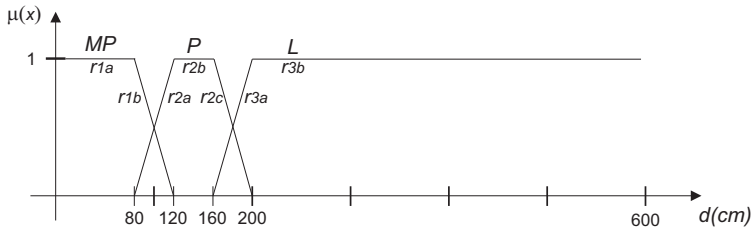


Figura 4.3: Divisão em regiões dos termos linguísticos para o cálculo de suas respectivas pertinências

Os cálculos das pertinências ocorrem de forma paralela e são determinados pelas funções de pertinência ativadas de acordo com o valores medidos pelos sensores. A Figura 4.4 apresenta a arquitetura de hardware

proposta para implementar o *bloco de fuzzificação*.

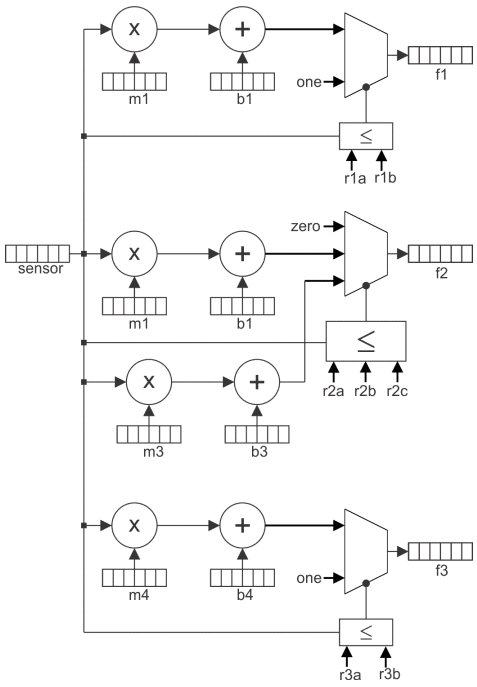


Figura 4.4: Arquitetura do *bloco de fuzzificação*. Aqui verifica-se que o valor de entrada do sensor é utilizado para determinar as pertinências (f_1,f_2 e f_3) de cada termo *fuzzy* (*MP*, *P* e *L*) da variável de entrada (distância). Os parâmetros r_{1a}, r_{1b}, ... r_{3b} correspondem às regiões das funções de pertinência de cada termo linguístico.

Devido à escolha da forma trapezoidal para as funções de pertinência algumas regiões não necessitam de cálculo para determinar o valor da pertinência, pois já são conhecidas (zero ou um), como a 1ł e a 2ł partes do termo *MP* (r_{11} e r_{12}), a 2ł parte do termo *P* (r_{22}) e a 2ł e 3ł partes do termo *L* (r_{32} e r_{33}). Já as outras são determinadas usando as funções que representam cada reta, de acordo com os valores de inclinação (m_1, m_2, etc.) e dos interceptos (b_1, b_2, etc.). No final, um comparador irá verificar em qual região o sensor está apontando e armazenará o valor correto na respectiva pertinência, como indicado na Figura 4.4.

4.4 BLOCO DE INFERÊNCIA

A arquitetura de hardware proposta para o *bloco de inferência* pode ser observada na Figura 4.5.

Essa parte do controlador envolve verificar todas as bases de regras, determinando as que estão ativas e as relacionando com a respectiva função de saída. Esses valores serão utilizados pelo *bloco de defuzzificaçã*o para determinar a saída do sistema. O controle do *bloco de inferência* é feito pela FSM mostrada na Figura 4.6 a qual calcula o numerador e denominador da Equação 2.8.

As 243 comparações não foram realizadas em paralelo por dois motivos: (a) diminuir o consumo de recursos economizando área para a implementação dos vários módulos que compõem o sistema global (vide Figura 1.2) e que futuramente deverão ser implementados na FPGA; (b) a limitação do número de blocos de DSPs disponíveis na FPGA utilizada (Artix7LX100T), limitado a 240 DSPs. Apesar dessa

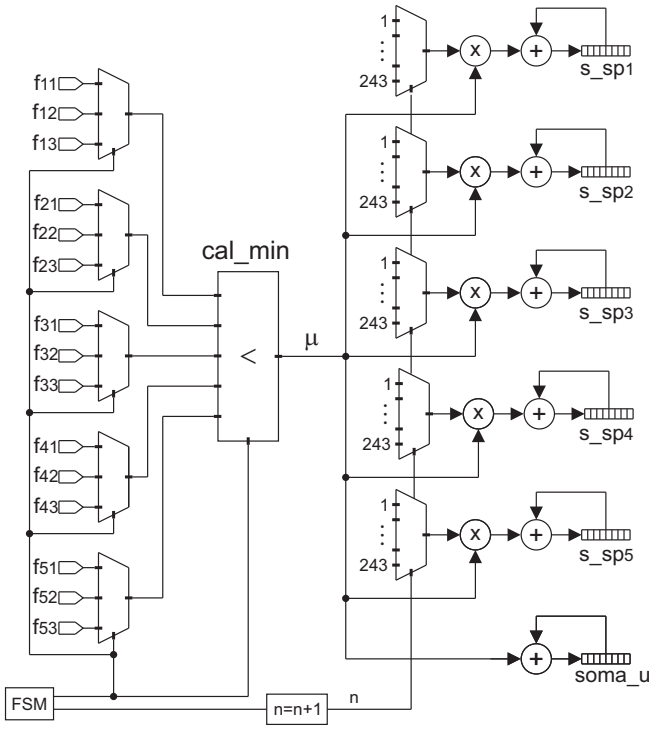


Figura 4.5: Arquitetura do bloco de inferencia. A FSM controla os multiplexadores de entrada e de endereçamento dos pesos da base de regras. Um contador é usado para percorrer sequencialmente as 243 regras. O calculo do valor da menor pertinência é feito de forma sequencial e são usados cinco multiplicadores, em paralelo, para calcular o valor da menor pertinência pelo respectivo peso oriundo da base de regras. Esse valor é acumulado, assim como o valores das menores pertinência. Tudo isso de acordo com a Equação (2.8) e com a FSM representada na Figura 4.6.

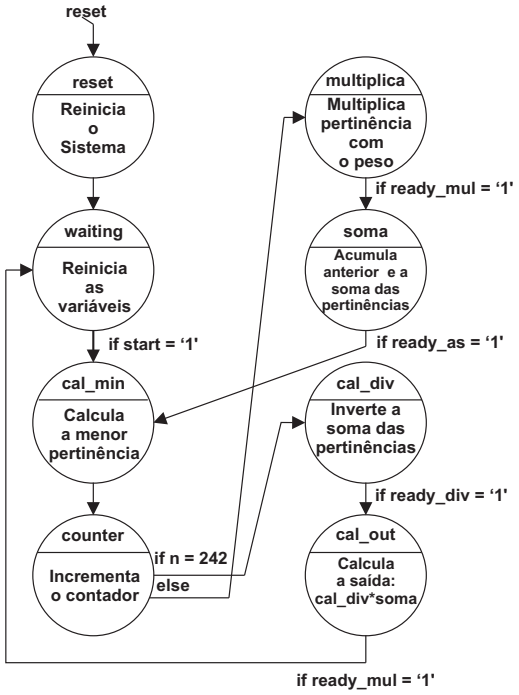


Figura 4.6: FSM do *bloco de inferência* que, entre outras operações, permite calcular o numerador e denominador da equação (2.8).

quantidade de DSPs ser bastante expressiva, o numéro não é suficiente para realizar todos os cálculos de forma simultâna. É importante salientar que foram utilizados os IPCores desenvolvidos no LEIA para realizar os cálculos aritméticos, sendo que a unidade de multiplicação faz uso de blocos DSP visando a diminuição do consumo de LUTs.

A forma mais apropriada de armazenar a base de regras é usando blocos RAM (BRAM) próprios da plataforma. Contudo, neste trabalho a base de regras foi armazenada em LUTs, devido ao processo de inicialização dos dados dos blocos BRAM, fato que dificulta a parametrização do código VHDL. Se faz necessário um aprofundamento nesse assunto de forma a retirar a base de regras das LUTs buscando diminuir o consumo de recursos.

Nessa etapa do controlador o *bloco de inferência* receberá 5 valores de pertinência, relativas a cada sensor de entrada e, através da interseção dos elementos, será determinado o menor valor entre as cinco possíveis sendo esses valores acumulados para posterior utilização na defuzzificação. Além disso, cada valor de pertinência calculado nessa etapa deve ser multiplicado pelo respectivo valor de saída determinado pela base de regras e, também, deve ser acumulado. Essas etapas podem ser observadas na Figura 4.5, juntamente com a FSM que controla o bloco.

4.5 BLOCO DE DEFUZZIFICAÇÃO

A parte de defuzzificação não existe no controlador *fuzzy*–TS, pois a saída já é um valor real, mas esse é o nome mais adequado para essa parte do controle. As saídas desse bloco estão de acordo com a média ponderada, dada pela equação (2.8). Cada entrada do bloco é multiplicada pelo valor correspondente da saída – dado pela tabela da base de regras – e esses valores são acumulados e divididos pela soma das entradas do bloco. A representação da arquitetura do *bloco de defuzzificação* pode ser observada na Figura 4.7.

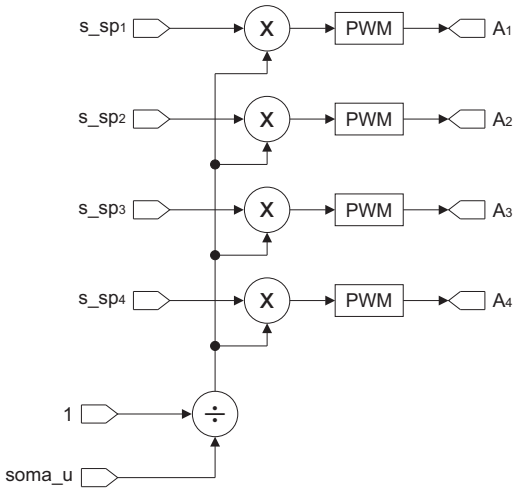


Figura 4.7: Arquitetura do bloco defuzzificação

A saída do *bloco de inferência* (*s_sp*$_1$, ..., *s_sp*$_4$ e *soma_u*) são utilizados para determinar a saída do módulo de detecção de obstáculos em duas etapas: (a) uma divisão que irá determinar o valor de $(soma\_u)^{-1}$ e (b) a multiplicação do resultada da divisão pelo somatório das ativações multiplicadas pelos respectivos

valores de saída (s_sp_1, ... e s_sp_4), determinando o valor das saídas (A_1, ... e A_4).

4.6 COMPARAÇÃO DO PROJETO DESENVOLVIDO EM CÓDIGO ESTRUTURADO COM O PROJETO DESENVOLVIDO EM VHDL

Com o contralador desenvolvido em software funcionando corretamente, foram desenvolvidas em VHDL as arquiteturas paralela e sequencial do controlador *fuzzy*–TS proposto. Os testes para validação foram feitos utilizando a ferramenta de co-simulação QuestaSim® que facilita o acesso as variáveis do projeto e a simulação de arquiteturas de hardware descritas em VHDL em conjunto com outras ferramentas do *Matlab*®, como o HDL Verifier e a toolbox Simulink. O esquema da co-simulação está representada na figura 4.8.

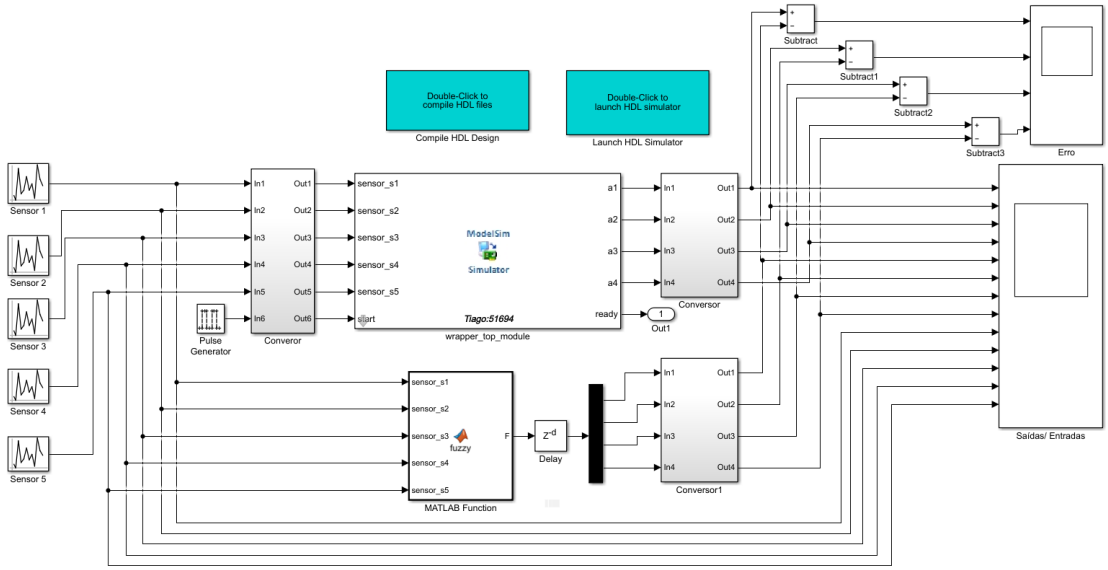


Figura 4.8: Esquemático para validação do projeto desenvolvido em hardware

O erro quadrático médio associado as arquiteturas paralela e sequencial estão representados na Tabela 4.1.

Tabela 4.1: Erro quadrático médio das arquiteturas propostas com tamanho de 27 bits

Arquitetura	Erro quadrático Médio
Paralela	4.89×10^{-5}
Sequencial	4.89×10^{-5}

O mesmo valor de erro quadrático médio para ambas as arquiteturas é explicado pelo fato das mesmas utilizar os mesmos módulos, apenas de uma forma diferente. O baixo valor do erro quadrático médio indica que os controladores, apesar de utilizarem uma representação numérica intermediária (27 bits), possuem uma alta precisão, adequada para o propósito do projeto.

4.7 CONSUMO DE RECURSOS DAS ARQUITETURAS PROPOSTAS

A síntese das arquiteturas em paralelo e sequencial fornecem dados relativos ao uso dos recursos de hardware pelo sistema e estão indicados na tabela 4.2.

Tabela 4.2: Dados relativos à utilização de recursos após a síntese das arquiteturas (a) paralela e (b) sequencial.

(a) Paralela		
Recursos	Utilizado	Utilização (%)
LUT	8256	13.02
FF	2759	2.18
DSP	10	4.17

(b) Sequencial		
Recursos	Utilização	Utilização (%)
LUT	4079	6.43
FF	1835	1.45
DSP	7	2.92

Ao observar o número de blocos de DSPs utilizados pode-se inferir que o software Vivado está otimizando o código. Observa-se a partir das figuras 4.4 e 4.5 que o bloco de fuzzificação deveria fazer uso de três multiplicadores e o bloco de inferência de cinco multiplicadores, além de mais quatro multiplicadores relativos a cada saída do sistema. Além disso, a implementação poderia ser mais eficiente se a base de regras fossem mapeadas nas BRAMs disponíveis na plataforma FPGA.

4.8 COMPARAÇÃO DO TEMPO DE EXECUÇÃO

Na Tabela 4.3 é apresentada uma comparação do tempo de execução do controlador *fuzzy*–TS em diversas plataformas computacionais. Observa-se que as arquiteturas de hardware possuem um tempo de execução muito superior em relação às implementações em software. Tendo como padrão o tempo de execução da arquitetura paralela (FPGA), verifica-se que o mesmo tem um fator de aceleração superior a 41 em comparação com o DevC++ e mais de 30 mil vezes em relação a plataforma Arduino. Nota-se que o tempo de execução da arquitetura paralela é praticamente idêntico ao da arquitetura sequencial e esse fato pode ser explicado levando em consideração que o módulo de inferência não foi desenvolvido de forma paralela, pois um dos critérios era a utilização dos IPCores desenvolvidos no LEIA, visando um menor consumo de recursos, e pela limitação do SoC utilizado, pois seriam necessárias 243 comparações e vários cálculos matemáticos. Mesmo assim, foi possível constatar a vantagem da arquitetura paralela na parte da fuzzificação, onde são necessário 50 ns para a execução, enquanto na arquitetura sequencial são necessários 290 ns (fator de aceleração de 5.8).

Tabela 4.3: Tempo de execução dos algoritmos do controlador *fuzzy*–TS

Plataforma	Tempo de execução us	Fator de aceleração
ARDUINO Mega - 16 MHz (32 bits)	7.52×10^5	31.07×10^3
DevC++® Windows 64 bits Core i7 - 2.40 GHz (32 bits)	1.00×10^3	41.32
FPGA arquitetura sequencial - 50 MHz (27 bits)	2.50×10^1	1.03
FPGA arquitetura paralela - 50 MHz (27 bits)	2.42×10^1	1

4.9 TESTES PRÁTICOS COM O PROJETO DESENVOLVIDO EM VHDL

Para verificação das arquiteturas de hardware propostas foram utilizados os switches emulando os sensores ultrassônicos, enquanto os leds foram utilizados para emular os sensores piezoelétricos. Dos 16 switches disponíveis na plataforma Nexys4, os 12 primeiros foram utilizados como bits de entrada simulando os 12 bits de saída do conversor ADC (vide Figura 1.2) e os 3 últimos são usados para indicar o sensor. Logo após, esse sinal de 12 bits é convertido para um sinal de 27 bits, pois internamente todos os cálculos são realizados com 27 bits. Na saída os leds devem receber um valor de duty cycle entre 0 e 100, que necessitam de 8 bits para serem representados. O sistema final para teste está representado na Figura 4.9.

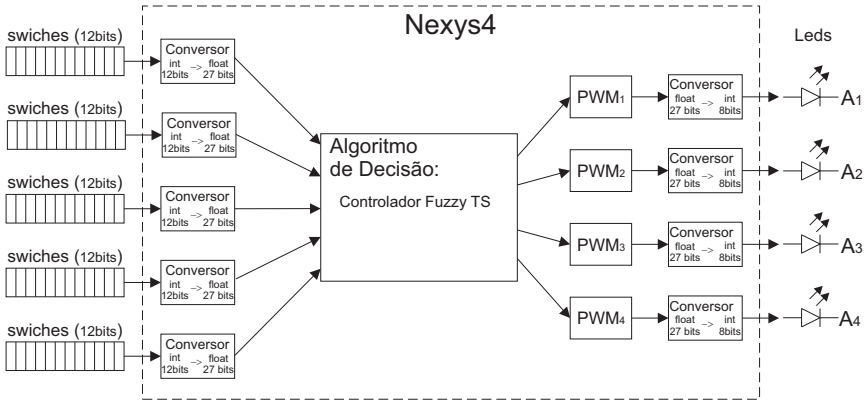


Figura 4.9: Esquemático do projeto final para testes

Durante o processo de implementação foi necessário acrescentar algumas restrições de timming, dentre as quais a mais importante foi a diminuição da frequência de clock para 50 MHz.

A figura 4.10 mostra o resultado obtido após processo de place and route (PAR), onde pode se observar a área ocupada pela fuzzificação (amarelo) e a área ocupada pela inferência (vermelha). Essa figura deixa evidente que a demanda de recursos de hardware por parte da inferência é mais significativa que a parte de fuzzificação e, por isso, a inferência acaba sendo a parte crítica do projeto.

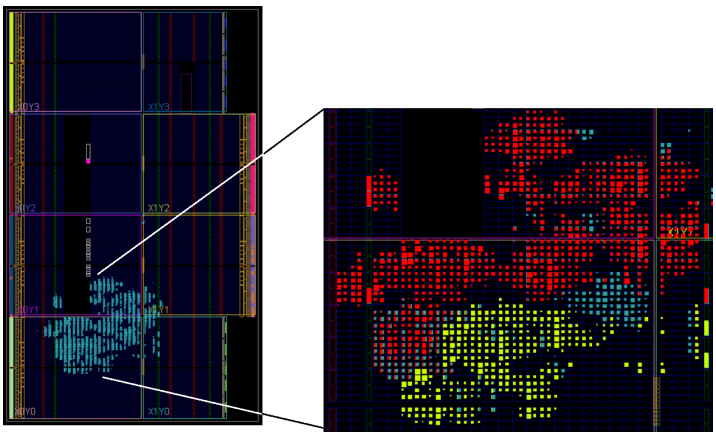


Figura 4.10: Área de recursos alocados pela arquitetura sequencial. No zoom destaca-se a área vermelha, relativa a área ocupada pela inferência e a área amarela, relativa a área ocupada pela fuzzificação.

A Figura 4.11 mostra a área ocupada pela arquitetura paralela na FPGA. É possível observar que a área ocupada pela arquitetura paralela é maior que a sequencial, o que coincide com os dados relativos à Tabela 4.2. Nessa implementação a parte da fuzzificação (amarelo) demanda mais recursos, visto que são implementados cinco blocos de fuzzificação em paralelo .

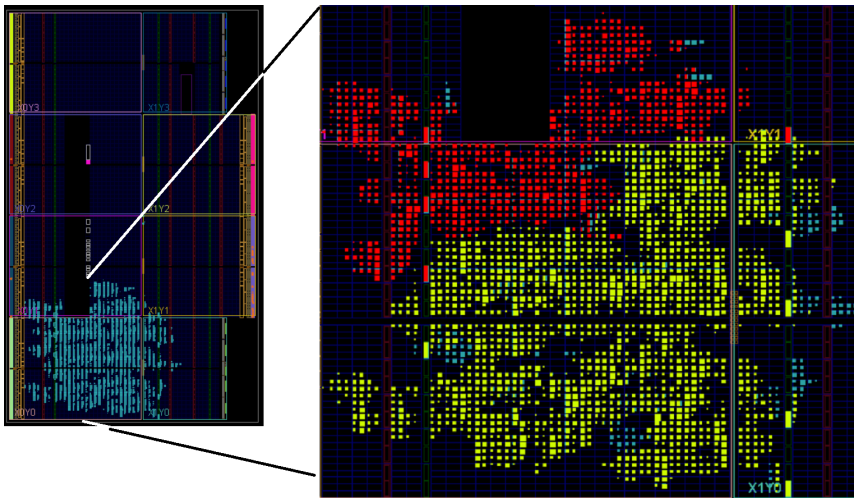


Figura 4.11: Área de recursos alocados pela arquitetura paralela. No zoom destaca-se a área vermelha, relativa a área ocupada pela inferência e a área amarela, relativa a área ocupada pela fuzzificação.

Devido a menor área necessária para a implementação da arquitetura sequencial espera-se que a mesma consuma uma menor potência, como observado na Tabela 4.4, que apresenta o consumo estimado pelas arquiteturas propostas após a implementação da arquitetura.

Tabela 4.4: Consumo de potência estimado após implementação das arquiteturas propostas.

Arquitetura	Potência Dinâmica (W)	Potência Estática (W)	Potência Total (W)
Paralela	0.092	0.097	0.189
Sequencial	0.004	0.104	0.108

Os testes realizados com a plataforma Nexys 4 Artix-7 FPGA pode ser verificados no vídeo disponibilizado no link <https://www.youtube.com/watch?v=eJUcsxSTD8w>. Nele são demonstrados alguns testes que comprovam a eficácia da utilização do controlador *fuzzy*–TS como responsável por indicar os desvios necessários para evitar possíveis obstáculos.

4.10 CONCLUSÕES DO CAPÍTULO

Nesse capítulo foi detalhado a implementação em hardware dos blocos do controlador *fuzzy*–TS: a fuzzificação, a inferência e a defuzzificação.

O controlador *fuzzy*–TS foi projetado para cinco sensores de entrada, espaçados de 45 graus entre si, contudo o ângulo de abertura dos sensores ultrassônicos é de aproximadamente 15 graus o que indica a necessidade de se usar mais sensores em anel para fazer a varredura de 180 graus (Figura 1.2).

Modificar a arquitetura de hardware para processar mais entradas envolve instanciar mais componentes de fuzzificação, assim como retrabalhar o componente da inferência que inclui a base de regras. Esse

processo geralmente é dispendioso e propenso a erros.

A dificuldade de modificar a arquitetura de hardware em termos de número de entradas e saídas, assim como o tamanho da palavra, o que afetaria diretamente a precisão do resultado.

O próximo capítulo descreve o desenvolvimento de uma ferramenta de geração de código VHDL a partir de uma descrição em alto nível de abstração usando o toolbox *fuzzy* do Matlab.

Capítulo 5

DESENVOLVIMENTO DE UM GERADOR AUTOMÁTICO DE CÓDIGO VHDL PARA CONTROLADORES FUZZY TAKAGI-SUGENO DE ORDEM ZERO

No decorrer do projeto verificou-se uma regularidade na implementação em VHDL do controlador *fuzzy*–TS. Esse fato possibilitou o desenvolvimento de uma ferramenta de geração automática de código VHDL de controladores *fuzzy*–TS genéricos, sendo esse resultado uma das contribuições mais importantes do presente trabalho.

5.1 REQUISITOS DO GERADOR AUTOMÁTICO DE CÓDIGO VHDL

Abaixo são descritos os requisitos para o funcionamento do gerador automático de código VHDL:

1. Captura dos parâmetros do controlador a partir de um modelo de alto nível desenvolvido no toolbox *fuzzy* do *Matlab*®;
2. Estrutura parametrizável em termos do número de entradas e saídas do sistema *fuzzy*;
3. Realizar cálculos em ponto flutuante baseado no padrão IEEE754;
4. O gerador deve ser parametrizável em função do tamanho da palavra desejada informando o tamanho do expoente e da mantissa;
5. O gerador deve criar um modelo de simulação do sistema de acordo com o universo de discurso fazendo uso de arquivos testbech com capacidade de leitura e escrita de dados;

6. A frequência de operação deve ser no mínimo de 100 MHz (frequência típica do sinal de clock dos FPGAs comerciais de baixo custo).

5.2 DESENVOLVIMENTO DO GERADOR DE CÓDIGO VHDL

O algoritmo do gerador automático de código VHDL foi projetado tendo como base a arquitetura paralela geral do controlador *fuzzy*–TS apresentado na Figura 5.1.

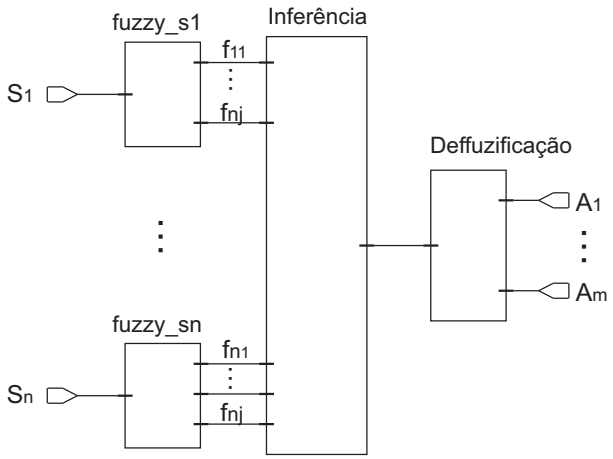


Figura 5.1: Arquitetura de um controlador *fuzzy*–TS geral, com n entradas m saídas e k funções de pertinência nos termos linguísticos de entrada.

O sistema proposto gera automaticamente um código VHDL a partir da captura dos dados de um modelo de alto nível desenvolvido no toolbox *fuzzy* do *Matlab*. Esses dados do modelo do controlador são armazenadas em um arquivo do tipo struct, que consiste numa matriz que contem as seguintes informações: (a) nome; (b) tipo de inferência do controlador; (c) tipo de interseção (AND); (d) tipo de união (OR); (e) forma de implicação; (f) forma de agregação; (g) dados relativos a entrada; (h) dados relativos a saída e (j) dados relativos a base de regras, conforme mostrado na Figura 5.2.

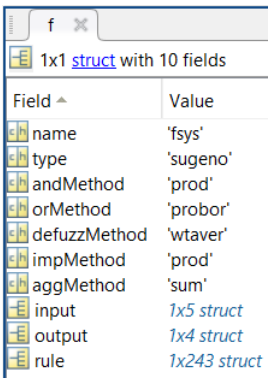
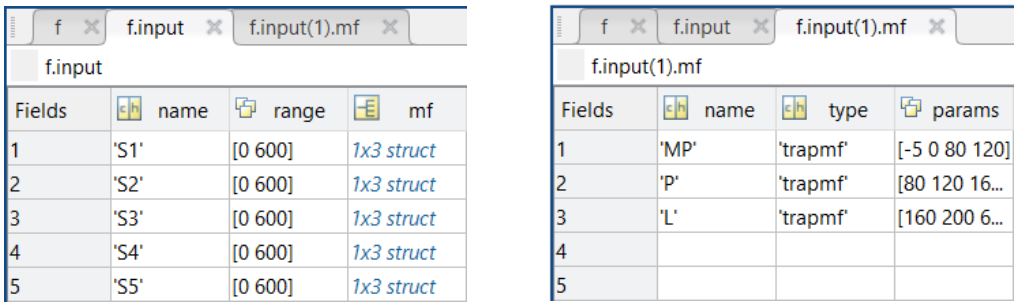


Figura 5.2: Arquivo struct que armazena as informações do controlador desenvolvido no toolbox *fuzzy*.

No caso das entradas é necessário informar o número de termos linguísticos e o tipo e as características das funções de pertinência de cada termo linguístico. Expandindo a estrutura da variável de entrada é

possível verificar essas informações (vide Figura 5.3).



f | f.input | f.input(1).mf

f.input

Fields	name	range	mf
1	'S1'	[0 600]	1x3 struct
2	'S2'	[0 600]	1x3 struct
3	'S3'	[0 600]	1x3 struct
4	'S4'	[0 600]	1x3 struct
5	'S5'	[0 600]	1x3 struct

(a)

f | f.input | f.input(1).mf

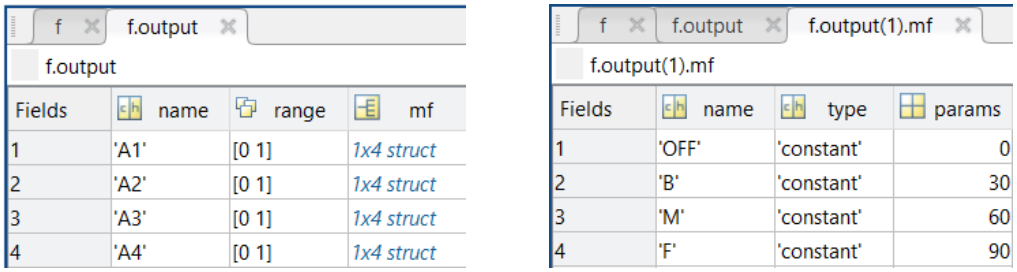
f.input(1).mf

Fields	name	type	params
1	'MP'	'trapmf'	[-5 0 80 120]
2	'P'	'trapmf'	[80 120 16...
3	'L'	'trapmf'	[160 200 6...
4			
5			

(b)

Figura 5.3: Arquivos structs que armazenam: (a) Informações relativas às variáveis linguística de entrada e (b) Respectivos termos linguísticos.

Já para as variáveis linguísticas de saída também é necessário informar o número de termos linguísticos e a função de saída do sistema. No caso do controlador *fuzzy*–TS de ordem zero só terá o valor relativo ao termo linear da equação. Ao expandir a estrutura da variável de saída é possível constatar esses dados (vide Figura 5.4).



f | f.output

f.output

Fields	name	range	mf
1	'A1'	[0 1]	1x4 struct
2	'A2'	[0 1]	1x4 struct
3	'A3'	[0 1]	1x4 struct
4	'A4'	[0 1]	1x4 struct

(a)

f | f.output | f.output(1).mf

f.output(1).mf

Fields	name	type	params
1	'OFF'	'constant'	0
2	'B'	'constant'	30
3	'M'	'constant'	60
4	'F'	'constant'	90

(b)

Figura 5.4: Arquivos structs que armazenam: (a) Informações relativas às variáveis linguística de saída e (b) Dados relativos às funções de saída.

A ferramenta desenvolvida gera o código VHDL dos seguintes módulos ou componentes:

1. *top_module*.vhd: arquivo principal que instancia os outros módulos;
2. *fuzzy*.vhd: arquivo que faz o cálculo das pertinências de cada variável de entrada;
3. *inference*.vhd: arquivo responsável por fazer o cálculo da inferência e calcular a saída;
4. *tb_top_module*.vhd: arquivo que irá gerar um testbench automático com vetores de teste aleatórios de acordo com o universo de discurso.
5. *fpupack*.vhd: biblioteca que tem todos os parâmetros da arquitetura;
6. *entities*.vhd: arquivo em que é declarado todas as entidades utilizadas;

Adicionalmente, são gerados os IPCores desenvolvidos no LEIA necessários para realizar os cálculos aritméticos:

1. *addsubfsm_v6*.vhd: responsável pelos cálculos de soma e de subtração;

2. *multiplierfsm_v2*.vhd: bla bla bla.

3. *divNR*.vhd: responsável pelos cálculos de divisão;

4. *fixMul_v6*.vhd: IPCore de auxílio a divisão.

Um exemplo de geração de códigos pode ser visto em <https://www.youtube.com/watch?v=AQNXxJWOuiE>.

5.3 TESTES COM O GERADOR AUTOMÁTICO DE CÓDIGO VHDL

O gerador automático de código foi utilizado para verificar o impacto no consumo de recursos ao se alterar alguns parâmetros do controlador: (a) o tamanho das palavras, onde foram realizadas sínteses com diferentes formatos numéricos padrões do IEEE754, 32 bits (float) ou 64 bits (double), além de representações intermediárias, mantendo constante o número de entradas (cinco) e de saídas (quatro); (b) o número de entradas e de saídas, mantendo constante o tamanho das palavras em 27 bits.

Primeiro foi realizada a síntese alterando o tamanho das palavras. O resultado da síntese está apresentado na Tabela 5.1, onde é possível verificar que o aumento do número de bits aumenta significativamente o consumo de LUTs, FF e DSPs, mostrando que o formato numérico é um requisito fundamental ao se projetar sistemas complexos.

Tabela 5.1: Consumo de recursos para sistemas *fuzzy* com diversos tamanhos de palavra. O sistema *fuzzy* testado possui 5 entradas e 4 saídas.

Número de bits	LUT	FF	DSP
27	8350	2639	10
32	9841	3108	20
48	17844	4635	30
64	24723	6201	70

Também foram realizados testes de síntese para verificar o impacto no consumo de recursos com o aumento do número de entradas e saídas. A Tabela 5.2 apresenta os resultados da síntese lógica.

Tabela 5.2: Consumo de recursos para sistemas *fuzzy* com alterações no número de entradas e/ou saídas. Em todos os casos os sistemas tinham palavras com 27 bits.

Número de entradas	Número de saídas	LUT	FF	DSP
2	1	3713	1171	8
2	2	4115	1332	9
3	1	5501	1549	11
3	2	6333	1710	12
3	3	5484	1871	13
4	1	6470	1928	14
4	2	6952	2096	15
6	3	9755	3029	10
7	3	11603	3445	11
8	4	14877	4075	13
10	5	38005	4942	16

Na Figura 5.5, referente ao consumo de LUTs, verifica-se que o aumento do número de saídas, mantendo as entradas constantes, não afeta o consumo de recursos tanto quanto a alteração no número de entradas, mantendo o número de saídas constantes. O aumento do consumo de recursos é maior quando ocorre o aumento no número de entradas e de saídas (diagonal 2x1, 3x2, 4x3,..., 10x5).

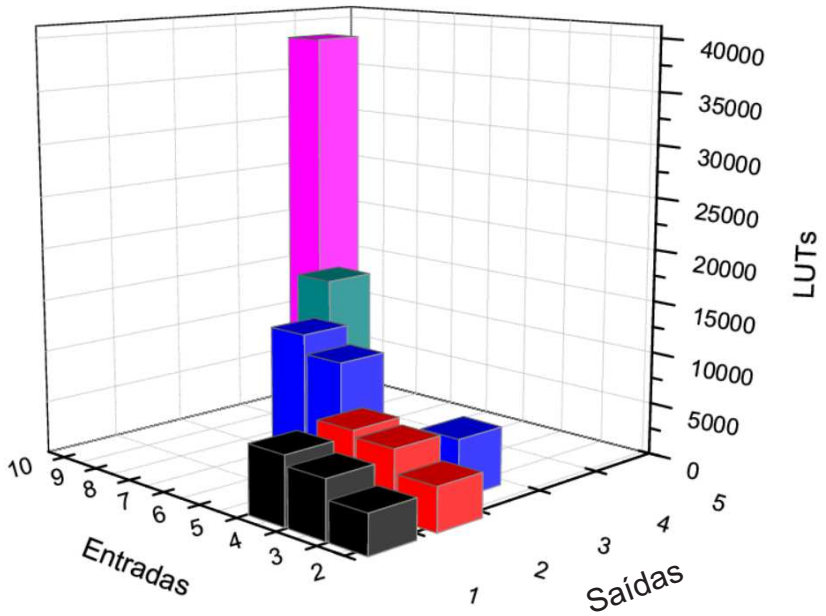


Figura 5.5: Gráfico que relaciona o número de *look-up-table* (LUTs) com os números de entradas e saídas.

Na Figura 5.6, que se refere ao consumo de FFs, verifica-se o mesmo que no caso das LUTs: a maior variação ocorre na diagonal, mas o impacto do aumento no número de entradas é mais relevante, porque afeta diretamente o número de elementos da base de regras, que são armazenadas nos FFs.

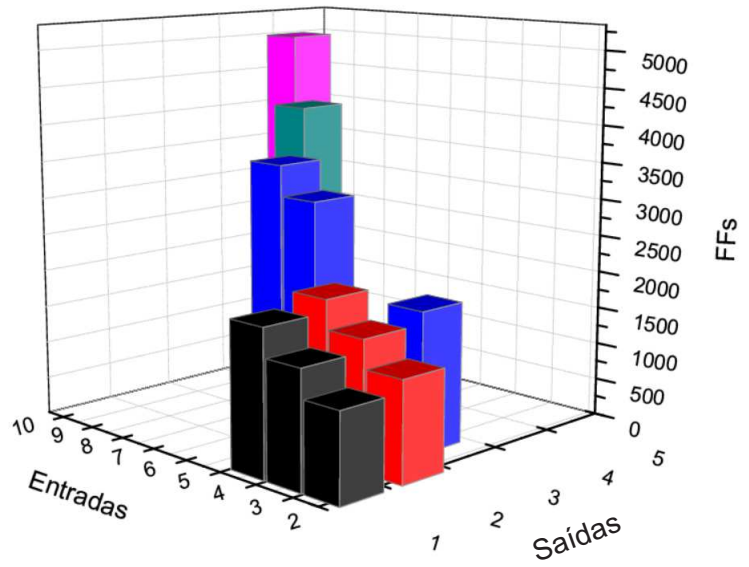


Figura 5.6: Gráfico que relaciona o número de *Flip-Flop* (LUTs) com os números de entradas e saídas.

Como os DSPs estão diretamente ligados aos cálculos aritméticos é de se esperar que qualquer aumento no número de entradas e saídas altere o seu valor. Na Tabela 5.2 e na Figura 5.7 se verifica que, a partir de

6 entradas, o Vivado começa a otimizar de forma mais intensa a síntese e, ao invés de ocorrer um aumento no gasto de DSP, ocorre uma diminuição significativa com posterior aumento. Novamente, o crescimento do número de entradas é mais impactante do que o aumento das saídas. Isso se deve ao fato do bloco de fuzzificação estar diretamente relacionando com o número de entradas, no qual são mais usados os cálculos de multiplicação.

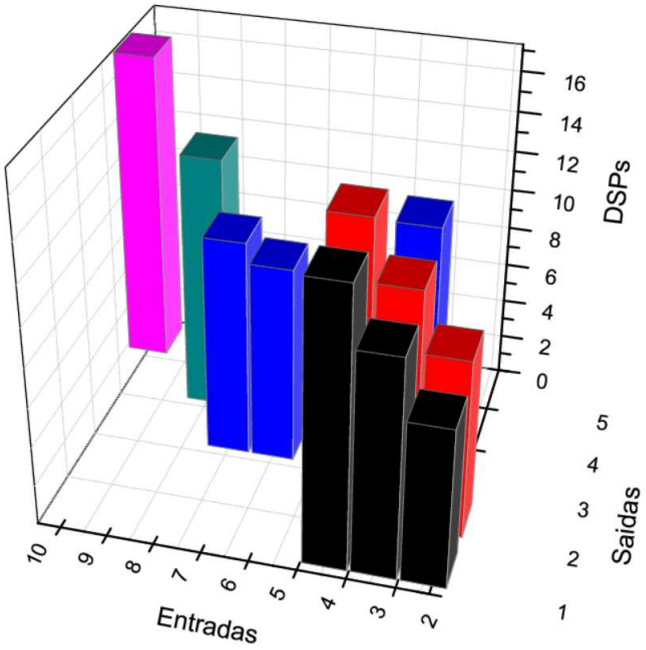


Figura 5.7: Gráfico que relaciona o número de *Digital Signal Processing* (DSPs) quando os números de entradas e saídas variam.

5.4 LIMITAÇÕES DO GERADOR AUTOMÁTICO DE CÓDIGO VHDL DESENVOLVIDO

A versão atual do gerador de código VHDL desenvolvido possui algumas limitações que estão descritas a seguir:

1. A ferramenta desenvolvida gera o código VHDL somente para controladores *fuzzy*–TS de ordem zero, pois são mais simples de serem implementados. Os modelos do controlador *fuzzy*–TS de ordem superior podem ser mais eficientes do ponto de vista de consumo de recursos, visto que seria necessário armazenar apenas os parâmetros das funções de saídas;

2. As entradas e as saídas devem possuir as mesmas características (número de termos linguísticos e tipo das funções de pertinência);

3. As funções de pertinência devem ser do tipo trapezoidal. Para trabalhos futuros se faz necessário que seja adicionado novas funções de pertinência;

4. Só pode haver a sobreposição de no máximo duas funções de pertinência;

5. O universo de discurso das entradas deve ser sempre positivo.

5.5 CONCLUSÕES DO CAPÍTULO

Nesse capítulo foi apresentado as ideias gerais para o desenvolvimento do gerador automático VHDL para controladores *fuzzy*–TS. Devido a criação do gerador foi possível verificar a influência do tamanho das palavras (bits) e do número de entradas e de saídas no consumo de recursos. É possível concluir que o numero de entradas tem maior impacto no consumo de recursos do que o número de saídas.

Capítulo 6

CONCLUSÕES

Este capítulo resume os resultados e contribuições dessa dissertação de mestrado, seguidos de uma discussão sobre o encaminhamento dos trabalhos futuros.

6.1 CONCLUSÕES GERAIS

Nesse trabalho foi desenvolvido um sistema de detecção de obstáculos para deficientes visuais utilizando dispositivos reconfiguráveis, cuja tomada de decisão de qual direção o usuário deveria seguir para evitar os obstáculos foi realizada por um controlador *fuzzy* Takagi-Sugeno.

A implementação do controlador em arquiteturas reconfiguráveis confirmou o correto funcionamento das arquiteturas de hardware propostas, indicando ao usuário a direção de movimento no intuito de desviar o obstaculo. Adicionalmente, esta implementação permitiu verificar a vantagem em relação ao tempo de execução, quando comparados aos sistemas em software, e a escolha da utilização da representação numérica de 27 bits mostrou uma forma de reduzir o consumo de recursos da plataforma mantendo um alto nível de precisão nos cálculos.

O projeto foi além do proposto inicialmente ao desenvolver um gerador automático de código VHDL a partir de uma implementação inicial no toolbo *fuzzy* do Matlab. Ao gerar códigos em ponto flutuante o software desenvolvido colocou o trabalho no limiar da arte, visto que não foi encontrado nenhum outro gerador de códigos automático com essa capacidade.

A seguir são feitas algumas conclusões acerca do que foi desenvolvido nos capítulos anteriores e a indicação de melhorias em trabalhos futuros.

6.2 IMPLEMENTAÇÃO EM FPGA DO CONTROLADOR FUZZY TAKAGI-SUGENO

No capítulo 4 foram implementadas duas arquiteturas em hardware do controlador *fuzzy*–TS, uma com a defuzzificação sendo feita de forma paralela e outra de forma sequencial. O propósito do desenvolvimento

das duas arquiteturas foi mostrar o ganho expressivo no tempo de execução quando essa parte do algoritmo é executado de forma paralela. O módulo de fuzzificação da arquitetura paralela é 5 vezes mais rápido que no modo sequencial. Em contrapartida, a arquitetura sequencial consome cerca de 53,7% do total de recursos da arquitetura paralela, mostrando ser mais eficiente no consumo de recursos. Também é importante ressaltar que após a analise de *timming* foi necessário reduzir a frequência do clock pela metade (50 MHz).

6.3 DESENVOLVIMENTO DE UM GERADOR AUTOMÁTICO DE CÓDIGO VHDL

O capítulo 5 apresentou o desenvolvimento de uma ferramente de geração automática de código VHDL de controladores *Fuzzy*. A ideia de fazer esse gerador surgiu ao longo do projeto ao se verificar que o código possuía uma regularidade, da necessidade de se evitar possíveis erros de digitação e de poder escalonar facilmente o projeto. Foi realizado uma analise do consumo de recursos alterando alguns parâmetros do controlador, como: (a) tamanho das palavras, onde foi verificado o aumento significativo no consumo de recursos com o aumento do tamanho da palavra; (b) o número de entradas e/ou saídas, indicando que o consumo de recursos aumenta mais significativamente com a variação do número de entradas, pois este parâmetro afeta diretamente o número de elementos da base de regras. É visível que o maior impacto ocorre no número de LUTs, que aumenta de forma significativa com as alterações nos números de entradas e saídas (vide Figura 5.5).

6.4 TRABALHOS FUTUROS

A seguir são relacionados propostas de trabalho futuro buscando melhorar o projeto proposto.

- TESTES PRÁTICOS: Foi constatada uma necessidade de melhorar as bases de regras, principalmente utilizando especialistas (deficientes visuais) para testar o controlador proposto. Além disso, como foi citado na introdução, o sistema só será aprovado, se atender as necessidades dos usuários finais, que deve ocorrer após a submissão do projeto ao comitê de ética da UnB, e a realização de testes reais;
- PARALELIZAR ARQUITETURA DE INFERÊNCIA: Apesar da limitição do SoC utilizado, em relação ao número de regras da base de conhecimento, é interessante a realização da parte de inferência de forma paralela. Uma solução possível seria a paralelização por conjuntos de regras;
- UTILIZAÇÃO DA MEMÓRIA BRAM: Todas as 243 regras foram armazenadas utilizando as LUTs e isso aumenta o consumo de recursos da plataforma, o que não é adequado. Portanto, um trabalho futuro é o armazenamento dessas informações nos blocos de memórias RAM (BRAM), liberando espaço para que as LUTs sejam utilizadas apenas para descrever o hardware;
- MELHORIAS NO GERADOR AUTOMÁTICO DE CÓDIGO VHDL: O gerador de códigos VHDL mostrou-se uma ferramenta útil ao facilitar o desenvolvimento de projetos de descrição de hardware.

Entretanto, a versão atual do gerador é limitada apenas aos controladores *Fuzzy* Takagi-Sugeno de ordem zero. Sendo assim, se faz necessário a expansão do projeto para controladores *Fuzzy* T-S de qualquer ordem e para controladores baseados no método de inferência Mamdani. Isso aproximaria a ferramenta desenvolvida do gerador de código VHDL do grupo x*Fuzzy* (50). Adicionalmente, a ferramenta desenvolvida pode ser incorporada ao toolbox *fuzzy* do *Matlab*®.

REFERÊNCIAS BIBLIOGRÁFICAS

1 HERSH, M.; JOHNSON, M. A. *Assistive technology for visually impaired and blind people*. New York, USA: Springer London, 2008.

2 ANDRADE, A. O.; RODRIGUES, S. S.; LEITE, C. R. M. *Novas Tecnologias Aplicadas à Saúde: Integração de áreas transformando a sociedade*. Rio Grande do Norte, Brasil: EDUERN, Editora Universitária, 2017.

3 SPIRKOVSKA, L. *Summary of Tactile User Interfaces Techniques and Systems*. NASA; Washington, DC, United States: NASA Ames Research Center, 2005.

4 FARIA, L. A. F. *Dispositivo de Auxílio à Locomoção de Deficientes Visuais Utilizando Fusão Sensorial em Arquiteturas Reconfiguráveis*. Dissertação (Mestrado) — FGA - Universidade de Brasília, Gama, Brazil, 2014.

5 KUNG, Y. S. et al. Fpga-implementation of inverse kinematics and servo controller for robot manipulator. In: *2006 IEEE International Conference on Robotics and Biomimetics*. [S.l.: s.n.], 2006. p. 1163–1168.

6 SÁNCHEZ, D. F. et al. Parameterizable floating-point library for arithmetic operations in fpgas. In: *Proceedings of the 22Nd Annual Symposium on Integrated Circuits and System Design: Chip on the Dunes*. New York, NY, USA: ACM, 2009. (SBCCI '09), p. 40:1–40:6. ISBN 978-1-60558-705-9. Disponível em: <http://doi.acm.org/10.1145/1601896.1601948>.

7 BARR, A.; FEIGENBAUM, E. A. *The Handbook of Artificial Intelligence - Vol. I*. Califórnia, EUA: William Kaufmann,Inc, 1981.

8 ZAVALA, A. H.; NIETO, O. C. Fuzzy hardware: A retrospective and analysis. *IEEE Transactions on Fuzzy Systems*, v. 20, n. 4, p. 623–635, Aug 2012. ISSN 1063-6706.

9 ZADEH, L. A. Fuzzy sets. *Information and Control*, v. 8, n. 3, p. 338 – 353, 1965. ISSN 0019-9958. Disponível em: <http://www.sciencedirect.com/science/article/pii/S001999586590241X>.

10 ZADEH, L. A. Fuzzy logic, neural networks, and soft computing. *Commun. ACM*, ACM, New York, NY, USA, v. 37, n. 3, p. 77–84, mar. 1994. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/175247.175255>.

11 BARR, A.; FEIGENBAUM, E. A. *The Handbook of Artificial Intelligence - Vol. II*. Califórnia, EUA: William Kaufmann,Inc, 1982.

12 SIMÕES, M. G.; SHAW, I. S. *Controle e Modelagem Fuzzy*. São Paulo, BRA: Blucher: FAPESP, 2007.

13 CAMPOS, M. N.; SAITO, K. *Sistemas Inteligentes em Controle e Automação de Processos*. Rio de Janeiro, BRA: Ciência Moderna, 2004.

14 SOUZA, O. N. *Introdução à Teoria dos Conjuntos Fuzzy*. 2010. Disponível em: <http://www.ime.unicamp.br/~valle/PDFfiles/osmar10.pdf>.

15 GUERRA, R. *Projeto e simulaço ãdo controle de atitude autônomo de satélites usando lógica nebulosa*. São José dos Campo, Brasil: Instituto Nacional de Pesquisas Espaciais (INPE), 1998.

16 ROSS, T. J. *Fuzzy Logic With Engineering Applications*. SÃčo Paulo, BRA: John Wiley Sons, 2010.

17 MAMDANI, E. H. Application of fuzzy algorithms for control of simple dynamic plant. *Electrical Engineers, Proceedings of the Institution of*, v. 121, n. 12, p. 1585–1588, December 1974. ISSN 0020-3270.

18 SUGENO, M.; MURAKAMI, K. Fuzzy parking control of model car. In: *The 23rd IEEE Conference on Decision and Control*. [S.l.: s.n.], 1984. p. 902–903.

19 JACQUES, M. A. P. et al. The impact of different approximate reasoning methods on fuzzy signal controllers. In: ____. *The 13th Mini-EURO Conference and the 9th Meeting of the EURO Working Group on Transportation, Bari, Italy, June 10-13, 2002*. [S.l.]: Polytechnic of Bari, Engineering Faculty, Italy, 2002. p. 184–192.

20 MENG, Y. An agent-based reconfigurable system-on-chip architecture for real-time systems. In: *Second International Conference on Embedded Software and Systems (ICESS'05)*. [S.l.: s.n.], 2005. p. 8 pp.–.

21 PLESSL, S. et al. The case for reconfigurable hardware in wearable computing. In: *Personal and Ubiquitous Computing*. [S.l.: s.n.], 2003. p. 1617–4917.

22 PELLIZZONI, R.; CACCAMO, M. Real-time management of hardware and software tasks for fpga-based embedded systems. *IEEE Transactions on Computers*, v. 56, n. 12, p. 1666–1680, Dec 2007. ISSN 0018-9340.

23 ABNOUS, A. et al. Evaluation of a low-power reconfigurable dsp architecture. In: ____. *10 IPPS/SPDP'98 Workshops Held in Conjunction with the 12th International Parallel Processing Symposium and 9th Symposium on Parallel and Distributed Processing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. cap. Parallel and Distributed Processing, p. 55–60.

24 MENCER, O.; MORF, M.; FLYNN, M. J. Hardware software tri-design of encryption for mobile communication units. In: *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. [S.l.: s.n.], 1998. v. 5, p. 3045–3048 vol.5. ISSN 1520-6149.

25 MIGNOLET, J. y. et al. Enabling hardware-software multitasking on a reconfigurable computing platform for networked portable multimedia appliances. In: *Proceedings of the International Conference on Engineering Reconfigurable Systems and Architecture 2002*. [S.l.: s.n.], 2002. p. 116–122.

26 SALEH, R. et al. System-on-chip: Reuse and integration. *Proceedings of the IEEE*, v. 94, n. 6, p. 1050–1069, June 2006. ISSN 0018-9219.

27 CHU, P. *FPGA Prototyping by Verilog Examples*. USA: John Wiley and Sons, 2008.

28 SASS, R.; SCHMIDT, A. G. *Embedded Systems Design with Platform FPGAs: Principles and Practices*. USA: Elsevier, 2010.

29 NAGARAJAN, R.; YAACOB, S.; SAINARAYANAN, G. Fuzzy clustering in vision recognition applied in navi. In: *2002 Annual Meeting of the North American Fuzzy Information Processing Society Proceedings. NAFIPS-FLINT 2002 (Cat. No. 02TH8622)*. [S.l.: s.n.], 2002. p. 261–266.

30 DUNAI, L. D. et al. Obstacle detectors for visually impaired people. In: *2014 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*. [S.l.: s.n.], 2014. p. 809–816. ISSN 1842-0133.

31 LANDA-HERNàNDEZ, A.; BAYRO-CORROCHANO, E. Cognitive guidance system for the blind. In: *World Automation Congress 2012*. [S.l.: s.n.], 2012. p. 1–6. ISSN 2154-4824.

32 ALGHAMDI, S.; SCHYNDEL, R. van; KHALIL, I. Safe trajectory estimation at a pedestrian crossing to assist visually impaired people. In: *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. [S.l.: s.n.], 2012. p. 5114–5117. ISSN 1094-687X.

33 HAN, S. B.; KIM, D.-H.; KIM, J. H. Fuzzy gaze control-based navigational assistance system for visually impaired people in a dynamic indoor environment. In: *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. [S.l.: s.n.], 2015. p. 1–7.

34 TATSUMI, H.; MURAI, Y.; MIYAKAWA, M. Rfid for aiding the visually impaired recognize surroundings. In: *2007 IEEE International Conference on Systems, Man and Cybernetics*. [S.l.: s.n.], 2007. p. 3719–3724. ISSN 1062-922X.

35 BALAKRISHNAN, G. et al. Stereopsis method for visually impaired to identify obstacles based on distance. In: *Image and Graphics (ICIG'04), Third International Conference on*. [S.l.: s.n.], 2004. p. 580–583.

36 ALAMY, L. E. et al. Bus identification system for visually impaired person. In: *2012 Sixth International Conference on Next Generation Mobile Applications, Services and Technologies*. [S.l.: s.n.], 2012. p. 13–17. ISSN 2161-2889.

37 SAMMOUDA, R.; ALRJOUB, A. Mobile blind navigation system using rfid. In: *2015 Global Summit on Computer Information Technology (GSCIT)*. [S.l.: s.n.], 2015. p. 1–4.

38 WATANABE, H. et al. Floor estimation by a wearable travel aid for visually impaired. In: *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. [S.l.: s.n.], 2015. p. 252–257.

39 MACNAMARA, S.; LACEY, G. A smart walker for the frail visually impaired. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. [S.l.: s.n.], 2000. v. 2, p. 1354–1359 vol.2. ISSN 1050-4729.

40 WEI, Y.; KOU, X.; LEE, M. Development of a guide-dog robot system for the visually impaired by using fuzzy logic based human-robot interaction approach. In: *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*. [S.l.: s.n.], 2013. p. 136–141. ISSN 2093-7121.

41 WEI, Y.; LEE, M. A guide-dog robot system research for the visually impaired. In: *2014 IEEE International Conference on Industrial Technology (ICIT)*. [S.l.: s.n.], 2014. p. 800–805.

42 WANG, X.; SHEN, X. h.; YAN, Y. s. Study of improved fuzzy control anti-collision method. In: *2012 IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC 2012)*. [S.l.: s.n.], 2012. p. 700–704.

43 YIZHEN, L.; LIN, L.; JUN, W. The application of pipeline technology: An overview. In: *2011 6th International Conference on Computer Science Education (ICCSE)*. [S.l.: s.n.], 2011. p. 47–51.

44 TOLEDO, E. J. et al. Fpga implementation of an augmented reality application for visually impaired people. In: *International Conference on Field Programmable Logic and Applications, 2005.* [S.l.: s.n.], 2005. p. 723–724. ISSN 1946-147X.

45 LEE, D.-J.; ANDERSON, J. D.; ARCHIBALD, J. K. Hardware implementation of a spline-based genetic algorithm for embedded stereo vision sensor providing real-time visual guidance to the visually impaired. *EURASIP J. Adv. Signal Process*, Hindawi Publishing Corp., New York, NY, United States, v. 2008, p. 107:1–107:10, jan. 2008. ISSN 1110-8657. Disponível em: <http://dx.doi.org/10.1155/2008/385827>.

46 LIN, K. W. et al. A wearable stereo vision system for visually impaired. In: *2012 IEEE International Conference on Mechatronics and Automation*. [S.l.: s.n.], 2012. p. 1423–1428. ISSN 2152-7431.

47 SEKHAR, V. C. et al. Design and implementation of blind assistance system using real time stereo vision algorithms. In: *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*. [S.l.: s.n.], 2016. p. 421–426.

48 TRENT, M. et al. An fpga-based portable real-time obstacle detection and notification system. In: *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. [S.l.: s.n.], 2016. p. 1954–1958.

49 DELIPARASCHOS, K. M.; NENEDAKIS, F. I.; TZAFESTAS, S. G. A fast digital fuzzy logic controller: Fpga design and implementation. In: *2005 IEEE Conference on Emerging Technologies and Factory Automation*. [S.l.: s.n.], 2005. v. 1, p. 4 pp.–262. ISSN 1946-0740.

50 MATIA FERNANDO; MARICHAL, G. N.; JIMENEZ, E. *Fuzzy Modeling and Control: Theory and Applications*. Madri, Spain: Atlantis Press, 2014.

51 IKEDA, H.; HIRAMOTO, Y.; KISU, N. A fuzzy inference processor with an "active-rule-driven" architecture. In: *1991 Symposium on VLSI Circuits*. [S.l.: s.n.], 1991. p. 25–26.

52 GABRIELLI, A.; GANDOLFI, E. A fast digital fuzzy processor. *IEEE Micro*, v. 19, n. 1, p. 68–79, Jan 1999. ISSN 0272-1732.

53 SURMANN, H. et al. Optimized fuzzy controller architecture for field programmable gate arrays. In: ____. *Second International Workshop on Field-Programmable Logic and Applications Vienna, Austria, 1992*. Berlin: Springer Berlin Heidelberg, 1993. cap. Field-Programmable Gate Arrays: Architecture and Tools for Rapid Prototyping.

54 SÁNCHEZ, D. *Implementação em VHDL de uma biblioteca parametrizável de operadores aritméticos em ponto flutuante para ser usada em problemas de robótica*. Dissertação (Mestrado) — Universidade de Brasília, Brasília, Brazil, 2009.

55 SÁNCHEZ, D. et al. Parameterizable floating-point library for arithmetic operations in fpgas. In: *Proc. International Symposium on Integrated Circuits and System Design*. Natal, Brazil: ACM, 2009. p. 253–258.

56 MUÑOZ, D. et al. FPGA-based floating-point library for CORDIC algorithms. In: *Proc. International Southern Programmable Logic Conference*. Porto de Galinhas, Brazil: IEEE, 2010. p. 55–60.

57 MUÑOZ, D. *Otimização por inteligência de enxames baseada em arquiteturas paralelas em aplicações embarcadas.* Dissertação (Mestrado) — Universidade de Brasília, Brasília, Brasil, 2012.

58 DIGILENT. *Nexys 4 DDR Reference Manual*. 2016. <https://reference.digilentinc.com/_media/nexys4-ddr:nexys4ddr_rm.pdf>.

APÊNDICES

A. PUBLICAÇÕES REALIZADAS

A.1 TRABALHOS ACEITOS EM CONGRESSOS

A.1.1 SBAI 2017

CONTROLE NEBULOSO EMBARCADO EM SISTEMAS RECONFIGURÁVEIS PARA DESVIO DE OBSTÁCULOS EM SISTEMA DE NAVEGAÇÃO PARA DEFICIENTES VISUAIS

TIAGO ROMEIRO DE JESUS[1,2], DANIEL MAURÍCIO MUÑOZ ARBOLEDA[1].

1. *Laboratório de Sistemas Embarcados e Circuitos Integrados Aplicados (LEIA), Departamento de Engenharia Mecânica, Universidade de Brasília.*
Campus Darcy Ribeiro – Asa Norte. Brasília – DF. 70 910 – 900
E-mails: tiago.jesus@ifg.edu.br; damuz@unb.br

2. *Laboratório de Automação e Controle, Departamento de Áreas Acadêmicas do Campus Jataí, Instituto Federa de Educação Ciência e Tecnologia de Goiás.*
Unidade Flamboyant – Residencial Flamboyant. Jataí – GO. 75 804 – 714
E-mail: tiago.jesus@ifg.edu.br

Abstract— Assistive technology pursuit the development of systems which are capable of insert a disabled person into society in order to help, facilitate and provide his inclusion and develop his quality of life. Although practically all devices developed to assist people with disabilities meet the proposed objectives, most of them do not consider the aspects experienced by visually impaired people and, consequently the users may reject these devices. In addition, these systems tend to be expensive, inefficient in relation to energy consumption, relatively large and usually make use of only one sensor to detect obstacles. This paper proposes a system to detect obstacles based on the user, using distance measuring sensors and Takagi-Sugeno Fuzzy controller for decision making. The intrinsic parallelism of the algorithms involved was explored using hardware architectures mapped into FPGAs devices. A co-simulation environment using Matlab® and Questasim® was developed allowing perform numerical comparisons. The hardware system enabled gains in runtime, compared with the software systems.

Keywords— Visually impaired; Navigation system; Fuzzy Control; Takagi-Sugeno; FPGA.

Resumo— A tecnologia assistiva busca o desenvolvimento de sistemas capazes de inserir uma pessoa com deficiência na sociedade para ajudar, facilitar e proporcionar sua inclusão e desenvolver sua qualidade de vida. Embora praticamente todos os dispositivos desenvolvidos para auxiliar as pessoas com deficiência satisfazem os objetivos propostos, a maioria deles não considera os aspectos vivenciados por deficientes visuais e, conseqüentemente, os usuários podem rejeitar esses dispositivos. Além disso, esses sistemas tendem a ser caros, ineficientes em relação ao consumo de energia, relativamente grande e geralmente fazem uso de apenas um sensor para detectar obstáculos. Este artigo propõe um sistema para detecção de obstáculos baseado no usuário, utilizando sensores de medição de distância e um controlador Fuzzy do tipo Takagi-Sugeno para as tomadas de decisões. O paralelismo intrínseco dos algoritmos envolvidos foi explorado usando arquiteturas de hardware mapeadas em dispositivos FPGAs. Um ambiente de co-simulação usando o Matlab® e Questasim® foi desenvolvido permitindo realizar comparações numéricas. O sistema em hardware permitiu ganhos em tempo de execução, se comparados com os sistemas em software.

Palavras-chave— Deficientes visuais; Sistemas de navegação; Controlador Fuzzy; Takagi-Sugeno; FPGA.

1 Introdução

A Tecnologia Assistiva (TA) é uma área do conhecimento que tem como objetivo promover a autonomia, independência, qualidade de vida e inclusão social de pessoas com alguma deficiência.

Pesquisas relacionadas a pessoas com deficiência visual são extensivamente estudadas desde os anos 70 e ultimamente se destacam os dispositivos com sensores baseados em visão (Nagarajan, 2202), (Dunai, 2014), (Landa, 2012), (Alghamdi, 2012) e (Han, 2015), obtenção de informações do ambiente através de etiquetas ou RFID-tags (Tatsumi, 2006), (Balakrishnan, 2004), (Alamy, 2012) e (Sammouda, 2014), laser (Watanabe, 2015) e o cão-guia robô (MacNamara, 2000), (Wei, 2013) e (Wei, 2014). É nessa área do cão-guia, também, onde se encontra a maioria dos trabalhos combinados com a Lógica Fuzzy, que se baseia na experiência humana para as tomadas de decisão. Grande parte das soluções apresentadas atualmente são onerosas, ineficientes em relação ao consumo energético, relativamente grandes e utilizam apenas um elemento sensor. Os que possuem dimensões pequenas possuem um baixo poder computacional e poucos são os trabalhos que utilizam a lógica fuzzy para solucionar problemas relacionados a tomadas de decisão por parte do deficiente visual. Entre esses trabalhos se destaca o de Wang (Wang, 2012), onde a detecção de obstáculos se baseia no controle fuzzy e na utilização de múltiplos sensores ultrassônicos.

Este trabalho propõe a implementação em arquitetura em hardware de um controlador nebuloso para desvio de obstáculos. Para atender a estes requisitos foi utilizado um sistema embarcado com FPGA, onde foram desenvolvidas duas implementações em hardware, uma com arquitetura paralela e outra com arquitetura sequencial, para efeito de comparação no ganho do tempo de execução. Foram, também, utilizados cálculos em ponto flutuante para melhorar a precisão e a faixa dinâmica das operações se comparado com sistemas implementados com pontos fixos

(Baturone, 2008). Para diminuir o consumo de recursos foram utilizados pontos flutuantes de 27 bits, cujos resultados foram comparados com uma arquitetura de 64 bits.

2 Lógica Fuzzy

A lógica fuzzy formaliza os conceitos que vão além do exato, atingindo valores aproximados ou imprecisos que são medidos através do seu grau de pertinência. É fortemente baseada na teoria de conjuntos, mas comumente é confundida com teoria de probabilidade (Zadeh, 2004).

A arquitetura geral dos controladores baseados na Lógica Fuzzy está mostrada na Figura 2.

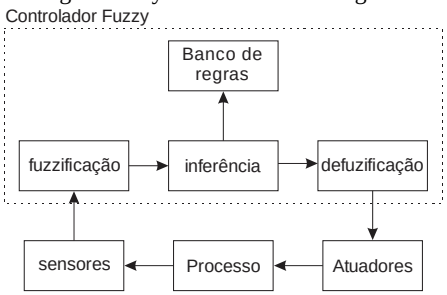


Figura 2. Diagrama do controlador Fuzzy

Os principais módulos do controlador são a fuzzificação, a base de conhecimento, a inferência e a defuzzificação. No módulo de fuzzificação os sinais dos sensores e dos atuadores devem ser convertidos em variáveis linguísticas. A seguir, o módulo de inferência utiliza o conjunto de regras definidos no módulo de base de conhecimentos para definir o comportamento do sistema e determinar o valor de saída, que continua sendo uma variável linguística. No final, o módulo de defuzificação converte esse resultado linguístico em um valor real (Campos, 2004), (Simões, 2007) e (Ross, 2010).

3 Modelo do Controlador Fuzzy

O sistema tem como base o desenvolvimento de um dispositivo, em formato de óculos, de auxílio à locomoção de deficientes visuais em recintos fechados. Para o sistema de detecção de obstáculos são utilizados cinco sensores ultrassônicos e quatro atuadores piezoeléctricos que através de vibração indicam a direção a ser seguida pelo deficiente visual (vide Figura 1)

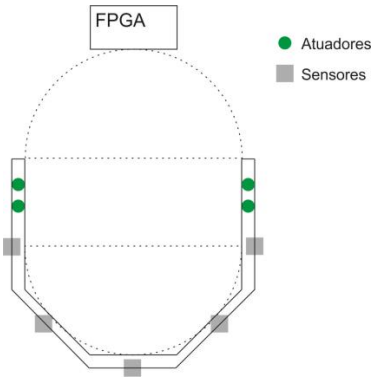


Figura 2. Vista superior do sistema para detecção de obstáculos

As variáveis de entrada são os sinais dos cinco sensores e as variáveis de saída são os sinais dos quatro atuadores, como indicado na Tabela 1.

Tabela 1. Indicação dos sensores e atuadores do sistema.

	Sensores					Atuadores			
	1	2	3	4	5	1	2	3	4
Símbolo	S1	S2	S3	S4	S5	A1	A2	A3	A4

A seguir as variáveis linguísticas são divididas em termos linguísticos, como indicado na Tabela 2 (sensores) e na tabela 3 (atuadores).

Tabela 2. Termos Fuzzy das variáveis linguísticas de entrada.

Termos Fuzzy da variável linguística de entrada	Símbolo
Muito Perto	MP
Perto	P
Longe	L

Tabela 3. Temos Fuzzy das variáveis linguísticas de saída.

Termos Fuzzy	Símbolo
Desligado	OFF
Baixo	B
Médio	M
Forte	F

Logo após, cada termo linguístico será relacionado a uma função de pertinência. No caso das entradas o universo de discurso está entre os valores de 20 cm e 400 cm, que define a escala dos sensores (vide Figura 3).

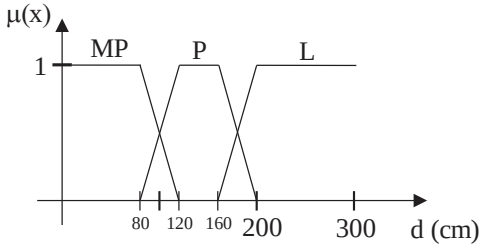


Figura 3. Funções de Pertinência para entrada

O controlador utilizado foi o de Takagi-Sugeno que, apesar de fácil implementação, requer atenção quando ao tempo de processamento, pois são analisadas várias sentenças para se obter o resultado final. Esse fato decorre do controlador Fuzzy Takagi-Sugeno seguir a regra do tipo equação 1, para variável i (Campos, 2004).

$$\text{Se}\left(x_1 = \text{A}_{1\text{i}} \text{ e } x_2 = \text{A}_{2\text{i}} \cdots\ x_1 = \text{A}_{3\text{i}}\right),\ \text{então}\ f\left(x_1, x_2 \cdots\ x_1\right) \quad (1)$$

No controlador proposto a função $f\left(x_1, x_2 \cdots\ x_1\right)$ está representada na Figura 4, que relaciona os termos Fuzzy da variável linguística de saída a valores constantes.

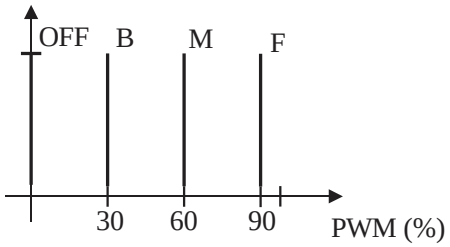


Figura 4. Funções de Pertinência para saída

A base de conhecimento está constituída a partir deste método, onde a quantidade de regras depende do número de entradas e do número de termos linguísticos. Portanto são necessárias 243 regras, que resultam dos três termos linguísticos definidos para a entrada combinados 5 vezes (número de sensores). O método de inferência do controlador se baseia na agregação das regras e a defuzzificação, na média ponderada de cada regra, como indicado na equação 2, onde $u(x_i)$ representa a pertinência das entradas e P_i o peso relativo as saídas (Campos, 2004).

$$y = \frac{\sum_{i=1}^{n} u(x_i) \times P_i}{\sum_{i=1}^{n} u(x_i)} \quad (2)$$

Os atuadores piezelétricos são ativados de forma a indicar a direção que o mesmo deve seguir, como indicado na Tabela 4.

Tabela 4. Sinal dos atuadores indicando a direção a ser seguida

Siga	Direita	Esquerda	Diagonal Direita	Diagonal Esquerda	Pare
● ● ○ ○	○ ● ○ ●	● ○ ● ○	○ ● ● ○	● ○ ○ ●	● ● ● ●

4 Implementação do Controlador Fuzzy em FPGA

Apesar da maioria dos trabalhos serem implementados na forma sequencial utilizando-se microcontroladores, PC's e processadores de propósito geral, o sistema nebuloso proposto foi mapeado em um dispositivo FPGA, usando a linguagem VHDL, explorando o paralelismo intrínseco dos algoritmos. Com isso espera-se diminuir consideravelmente o tempo de execução dos cálculos, realilzando operações de forma simultânea, tais como a fuzzificação dos sensores e a defuzzificação das saídas.

A arquitetura geral do controlador nebuloso está mostrada na Figura 3. Para efeitos de comparação, o controlador nebuloso foi implementado, também, em uma arquitetura sequencial, como indicado na Figura 4. A diferença entre as duas arquiteturas propostas é que a arquitetura paralela terá cinco blocos fuzzy em sua estrutura, enquanto a arquitetura sequencial terá apenas um bloco fuzzy que será ativado e reutilizado por uma máquina de (vide Figura 4).

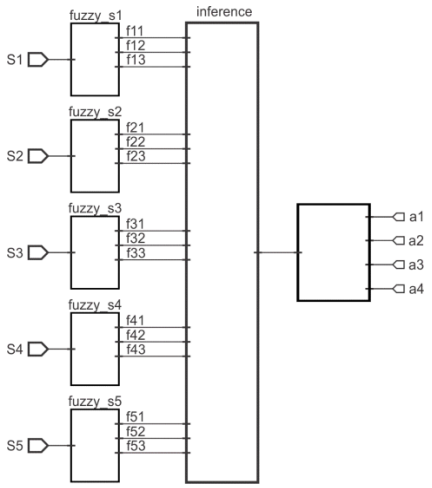


Figura 3. Esquema geral do Controlador com Arquitetura Paralela

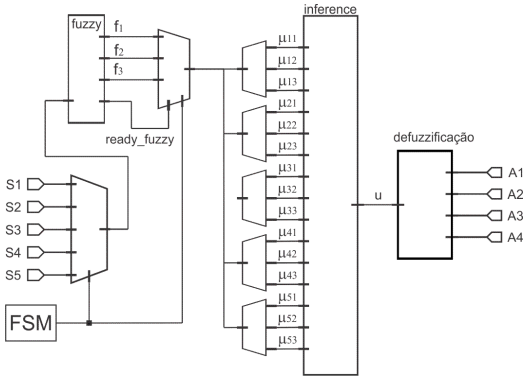


Figura 4. Esquema geral do Controlador Sequencial

O bloco fuzzy determina o valor do grau de pertinência dos sensores em relação a cada termo lin-

guístico da variável linguística. Como cada sensor possui três termos linguísticos, cada bloco Fuzzy tem três valores de saída. Como são cinco sensores, ao todo são quinze entradas para o bloco de inferência. A Figura 5 apresenta a arquitetura de hardware proposta para implementar o bloco Fuzzy. Observa-se que quatro multiplicações e quatro somas são realizadas em paralelo.

O bloco de inferência (vide Figura 6) determina os valores de ativação de cada regra através do operador de intersecção e os disponibiliza para o bloco de defuzzificação (vide Figura 7). Esse bloco armazena os valores de ativação, e dos pesos relativos a cada regra, para calcular a saída dos atuadores de acordo com a equação 2. Como cada regra tem um valor de peso relacionado, não foi possível implementar a inferência de forma paralela, pois para isso demandaria um número excessivo de recursos do hardware, que não estava disponível.

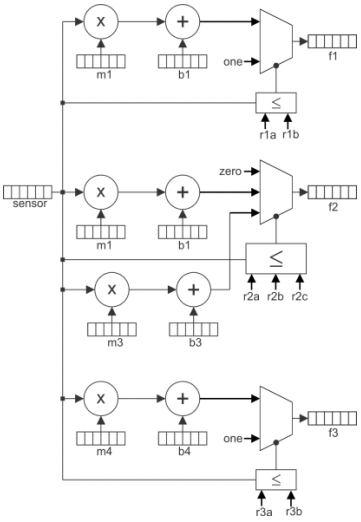


Figura 5. Bloco Fuzzy

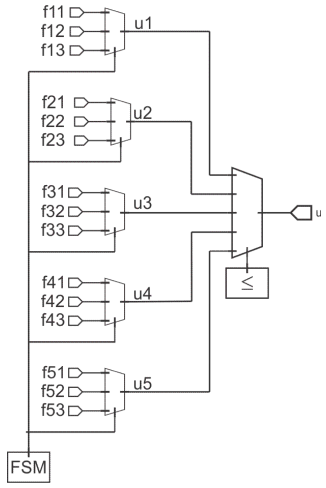


Figura 6. Bloco Inferência

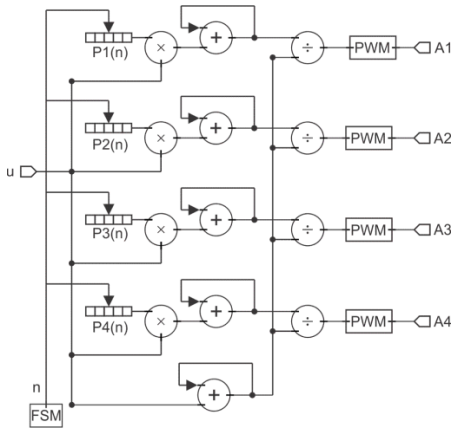


Figura 7. Bloco Defuzzificação

5 RESULTADOS

No intuito de validar o correto comportamento das arquiteturas de hardware propostas para o controlador fuzzy, uma implementação em software usando o Matlab® foi usada como modelo de referência. Simulações comportamentais foram realizadas usando obstáculos em diversas posições e os resultados obtidos pelo controlador foram comparados com a saída do modelo de referência. Para esta etapa foi utilizado o sistema de cosimulação cosimWizard do Matlab® que facilita a inserção do Questasim® como verificador de código em VHDL (HDL Verifier), possibilitando a simulação de vários casos. A Tabela 1 mostra o erro quadrático médio associado às arquiteturas em paralelo e sequencial, em relação ao código em Matlab.

Tabela 2: Erro quadrático médio das arquiteturas propostas

Arquitetura	Erro quadrático médio
Paralela	0,00004888056824
Sequencial	0,00004888056824

Esses valores de erros indicam que a escolha de uma arquitetura de 27 bits é viável, pois além do erro ser pequeno o uso da arquitetura de 27 bits tem um melhor custo benefício em termos de utilização de recursos de hardware, se comparados com plataformas de 32 e 64 bits [14].

A síntese das arquiteturas em paralelo e sequencial forneceram dados relativos ao uso dos recursos do hardware pelo sistema e estão indicados nas tabelas 2 e 3, que mostra um vantagem da arquitetura sequenciail.

Tabela 3: Dados relativos à utilização de recursos após síntese – arquitetura paralela

Recursos	Estimado	Utilização (%)
LUT	8404	13,2
FF	2619	2,07

DSP	10	4,17

Tabela 4: Dados relativos à utilização de recursos após síntese – arquitetura sequencial

Recursos	Estimado	Utilização (%)
LUT	3943	6
FF	1693	1
DSP	7	3

A Tabela 4 apresenta uma comparação numérica do tempo de execução do controlador fuzzy em diferentes plataformas. Pode-se observar que as arquiteturas de hardware aceleram o tempo de execução do controlador fuzzy se comparado com as implementações em software. Adicionalmente, o tempo de execução da arquitetura paralela é praticamente idêntico à arquitetura sequencial. Isso foi devido ao bloco de inferência não ter sido desenvolvido de forma paralela. A vantagem da arquitetura paralela é visível na parte de fuzzificação, onde os cálculos foram realizados em 50 ns em comparação com 290 ns da arquitetura sequencial.

Tabela 4. Falta legenda da tabela

Plataforma	Tempo de execução (μs)	Fator de aceleração
Matlab® Windows 64 bits Core i7 2.40 GHz	571,227	1
FPGA arquitetura paralela – 100 MHz	17,145	33,32
FPGA arquitetura sequencial – 100 MHz	17,495	32,65

6 Conclusão

Este trabalho fez uma análise comparativa entre três plataformas distintas para implementação de um controlador Fuzzy para detecção de obstáculos para deficientes visuais. Os resultados demonstraram a eficiência no tempo de execução das arquiteturas em hardwares quando comparadas com arquiteturas em software. A utilização de arquiteturas com ponto flutuante de 27 bits mostrou um excelente custo benefício ao mostrar um baixo erro quadrático médio, em relação às arquiteturas de 64 bits, e consumindo um número menor de recursos, se comparado com arquiteturas em hardware de 32/64 bits. Este trabalho também demonstrou que as arquiteturas desenvolvidas são regulares e por isso é interessante o desenvolvimento de um gerador automático de códigos. A finalização do projeto, como a integração do sistema com o óculos e a verificação de sua aceitação por parte dos usuários é fundamental.

Agradecimentos

Ao Instituto Federal de Goiás pelo afastamento para realizar as pesquisas e pelo apoio financeiro.

Referências Bibliográficas

Alamy, L. E., Lhaddad, S., Maalal, S., Taybi, Y. and Salih-Alj, Y. (2012). Bus identification system for visually impaired person, 2012 Sixth International Conference on Next Generation Mobile Applications, Services and Technologies, pp. 13-17.

Alghamdi, S., van Schyndel, R. and Khalil, I. (2012). Safe trajectory estimation at a pedestrian crossing to assist visually impaired people, 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 5114-5117.

Balakrishnan, G., Sainarayanan, G., Nagarajan, R. and Yaacob, S. (2004). Stereopsis method for visually impaired to identify obstacles based on distance, Image and Graphics (ICIG'04), Third International Conference on, pp. 580-583.

Barr, A. and Feigenbaum, E. A. (1982). The Handbook of Artificial Intelligence, Heuristech Press.

Baturone I., F. J. Moreno-Velo, V. Blanco, J. Ferruz Design of Embedded DSP-Based Fuzzy Controllers for Autonomous Mobile Robots IEEE Transactions on Industrial Electronics, Vol. 55, Nº 2, pp. 928-936, Feb. 2008.

Campos, M. N. De, Saito, Kaku. Sistemas Inteligentes em Controle e Automação de Processos. Rio de Janeiro: Ciência Moderna, 2004.

Dunai, L. D., Lengua, I. L., Tortajada, I. and Simon, F. B. (2014). Obstacle detectors for visually impaired people, 2014 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM), pp. 809-816.

Han, S. B., Kim, D.-H. and Kim, J. H. (2015). Fuzzy gaze control-based navigational assistance system for visually impaired people in a dynamic indoor environment, Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on, pp. 1-7.

Hersh, M. A. and Johnson, M. A. (2008). Assistive technology for visually impaired and blind people, Springer.

Landa-Hern´andez, A. and Bayro-Corrochano, E. (2012). Cognitive guidance system for the blind, World Automation Congress (WAC), 2012, pp. 1-6.

MacNamara, S. and Lacey, G. (2000). A smart walker for the frail visually impaired, Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on, Vol. 2, pp. 1354-1359 vol.2.

Nagarajan, R., Yaacob, S. and Sainarayanan, G. (2002). Fuzzy clustering in vision recognition applied in navi, Fuzzy Information Processing

Society, 2002. Proceedings. NAFIPS. 2002 Annual Meeting of the North American, pp. 261-266.

Ross, T. J. (2010). Fuzzy Logic With Engineering Applications, 3 edn, John Wiley Sons. Sammouda, R. and Alrjoub, A. (2015). Mobile blind navigation system using rfid, Computer Information Technology (GSCIT), 2015 Global Summit on, pp. 1-4.

Sammouda, R. and Alrjoub, A. (2015). Mobile blind navigation system using rfid, *Computer Infomation Technology (GSCIT), 2015. Global Summit on*, pp. 1-4.

Simões, M. G. and Shaw, I. S. (2007). Controle e Modelagem Fuzzy, Blucher.

Tatsumi, H., Murai, Y., Miyakawa, M. and Tokumasu, S. (2006). Marking in space and acquisition of environmental information by datacarrier for the visually impaired, 2006 World Automation Congress, pp. 1-6.

Wang, X., h. Shen, X. and s. Yan, Y. (2012). Study of improved fuzzy control anti-collision method, Signal Processing, Communication and Computing (ICSPCC), 2012 IEEE International Conference on, pp. 700-704.

Watanabe, H., Tanzawa, T., Shimizu, T. and Kotani, S. (2015). Floor estimation by a wearable travel aid for visually impaired, Robot and Human Interactive Communication (RO-MAN), 2015 24th IEEE International Symposium on, pp. 252-257.

Wei, Y., Kou, X. and Lee, M. (2013). Development of a guide-dog robot system for the visually impaired by using fuzzy logic based human-robot interaction approach, Control, Automation and Systems (ICCAS), 2013 13th International Conference on, pp. 136-141.

Wei, Y. and Lee, M. (2014). A guide-dog robot system research for the visually impaired, Industrial Technology (ICIT), 2014 IEEE International Conference on, pp. 800-805.

Zadeh, L.A. FUZZY LOGIC SYSTEMS: ORIGIN, CONCEPTS, AND TRENDS. Computer Science Division Department of EECS UC Berkeley. Disponível em <https://wi-consortium.org/wicweb/pdf/Zadeh.pdf>. Acessado em novembro, 2016.

A.1.2 COBENGE 2017

DISPOSITIVO DIDÁTICO DE DETECÇÃO DE OBSTÁCULOS PARA DEFICIENTES VISUAIS BASEADO EM LÓGICA *FUZZY* EMBARCADO EM PLATAFORMA ARDUINO

Igor Oliveira Vieira – igorevan@gmail.com
Instituto Federal de Educação, Ciência e Tecnologia de Goiás
Rua Orminda Vieira de Freitas, 775 - Residencial Flamboyant
75804-714 – Jataí – Goiás

Ruth Pereira Fernandes – ruthfernandesd2@yahoo.com.br
Instituto Federal de Educação, Ciência e Tecnologia de Goiás
Rua Orminda Vieira de Freitas, 775 - Residencial Flamboyant
75804-714 – Jataí – Goiás

Daniel M. Muñoz – damuz@unb.br
GRACO, Departamento de Engenharia Mecânica, Universidade de Brasília,
Campus Universitário Darcy Ribeiro
70910-900 – Brasília – DF

Tiago Romeiro de Jesus – tiago.jesus@ifg.edu.br
GRACO, Departamento de Engenharia Mecânica, Universidade de Brasília,
Campus Universitário Darcy Ribeiro, 70910-900, Brasília, DF
Instituto Federal de Educação, Ciência e Tecnologia de Goiás
Rua Orminda Vieira de Freitas, 775 - Residencial Flamboyant
75804-714 – Jataí – Goiás

Resumo: *O atual processo de ensino-aprendizagem em disciplinas de sistemas de controle é baseado em aulas expositivas abordando tópicos teóricos e requerendo uma infraestrutura laboratorial onerosa. O uso de kits didáticos de baixo custo pode melhorar o processo de aprendizagem na sala de aula através do uso de casos de estudo reais. Por outro lado, a motivação por parte dos estudantes nestas disciplinas pode ser melhorada fazendo uso de temas transversais tais como as Tecnologias Assistivas. O presente trabalho propõe um sistema para detecção de obstáculos para deficientes visuais utilizando um protótipo composto por cinco sensores ultrassônicos espaçados entre si em 45º, permitindo uma varredura de 180º. O dispositivo é controlado por um controlador Fuzzy do tipo Takagi-Sugeno que foi embarcado na plataforma Arduino onde foram realizados testes para validação do projeto. O sistema de controle poderá ser melhorado com o feedback dos usuários finais, permitindo que as regras de decisões sejam ajustadas, sendo essa uma das vantagens do sistema Fuzzy: ser baseado na experiência humana. O projeto proposto tem um baixo custo e pode ser utilizado de forma didática e interdisciplinar, envolvendo disciplinas como sistemas de controle, microcontroladores, instrumentação, entre outras. Além disso, esse projeto, além de atender a uma orientação da OMS, vai de encontro com os ideais dos Parâmetros Curriculares Nacionais (PCN), inserindo o tema de saúde e sua importância na qualidade de vida dos cidadãos.*

Palavras-chave: *Ensino na engenharia, Tecnologias assistivas, Detecção de obstáculos, Controlador Fuzzy, Sistemas embarcados.*

1. INTRODUÇÃO

A educação é fundamentalmente um processo de exploração, de descoberta, de observação e construção do conhecimento que vivenciamos no mundo. O processo de constituição do pensamento lógico e matemático parte do pressuposto de que existe uma relação interdependente entre o sujeito conhecedor e o objeto a se conhecer, envolvendo mecanismos complexos, englobando aspectos que entrelaçam e se complementam (PIAGET, 1969). O uso da computação como ferramenta de educação, tem se tornado a cada dia indispensável para a melhoria da qualidade e dinamismo como alternativa ao ensino tradicional produzindo material atraente ao aluno, concatenando teoria, tecnologia e interdisciplinaridade (OLIVEIRA, 2010).

O ensino dos conceitos de sistemas de controle envolve diversos desafios devido ao denso conteúdo teórico inerente à disciplina. De forma geral, os professores que atuam nessa área fazem uso de aulas expositivas e/ou simulações computacionais para apresentar exemplos que demonstram o uso das técnicas de controle aplicadas em casos práticos. Contudo, a absorção dos conceitos nesta área de conhecimento é melhor consolidada através de aulas práticas através de casos de estudo reais (FERNANDES & GUEDES, 2003).

No Instituto Federal de Goiás (IFG), Campus Jataí, se oferece uma disciplina de sistemas de controle na qual são apresentados conceitos gerais sobre o tema. Uma das técnicas de controle mais utilizadas nos últimos anos é a Lógica *Fuzzy*, devido à sua facilidade de implementação além de possibilitar lidar com variáveis do tipo linguísticas. No intuito de melhorar o processo de ensino-aprendizagem da referida disciplina foi planejado o uso de kits didáticos de baixo custo que possam ser usados para colocar em prática os conceitos de Lógica *Fuzzy*.

A Lógica *Fuzzy* foi desenvolvida em 1965 por Lotfi Zadeh, professor da Universidade da Califórnia em Berkeley, e apresenta a ideia de gerar conclusões e respostas baseadas em informações imprecisas, qualitativas, incompletas, ambíguas e vagas, proporcionando sistemas que usam um raciocínio similar ao dos seres humanos (ZADEH, 1965).

Visando melhorar a motivação por parte dos estudantes da disciplina foi escolhido um tema com impacto social. Em particular trata-se do uso de tecnologias assistivas (TA) para auxílio à locomoção de deficientes visuais. A tecnologia assistiva visa englobar produtos, recursos, metodologias, estratégias, práticas e serviços para promoverem a autonomia, independência, qualidade de vida e a inclusão social de pessoas com deficiências, incapacidades ou mobilidades reduzidas (COMITÊ DE AJUDAS TÉCNICAS, 2016).

Segundo a Organização Mundial da Saúde (OMS), estima-se que 285 milhões de pessoas no mundo são deficientes visuais. Sendo que 39 milhões não conseguem enxergar nada e 246 milhões possuem baixa visão (possuem algum tipo de visão residual) (OMS, 2014). Pessoas que perdem a visão têm suas vidas afetadas de forma significativa, não somente no sentido pessoal, como também no sentido econômico e social, resultando em um problema sério de saúde coletiva (WEST & SOMMER, 2001).

A contribuição do presente trabalho é o desenvolvimento de um kit didático de baixo custo com as seguintes características: (a) sistema baseado em um controlador *Fuzzy* para navegação de deficientes visuais em recintos fechados através da estimação da distancia até os

obstáculos; (b) sistema portável em termos de tamanho, peso e consumo energético; (c) sistema que possa ser usado em sala de aula para aplicar os conceitos de Lógica *Fuzzy* na disciplina de sistemas de controle. A proposta consiste no desenvolvimento de um dispositivo para auxílio à mobilidade de deficientes visuais. Tal dispositivo será composto por cinco sensores ultrassônicos, quatro atuadores piezoelétricos e um sistema microcontrolado baseado na plataforma Arduino.

O presente trabalho está organizado da seguinte maneira: A seção 2 apresenta a fundamentação teórica abordando temas como lógica *Fuzzy* e tecnologias assistivas. A seção 3 detalha o desenvolvimento do kit didático, incluindo a descrição do controlador *Fuzzy*, assim como uma descrição da plataforma computacional e dos sensores usados no desenvolvimento. Antes de concluir, a seção 4 apresenta os resultados de simulação numérica e os testes experimentais feitos com o protótipo desenvolvido.

2. FUNDAMENTAÇÃO TEÓRICA

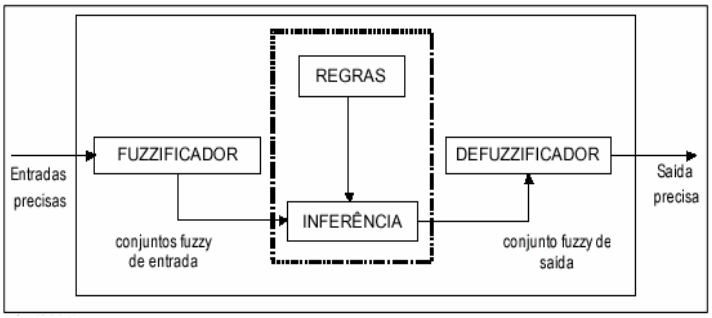
2.1. Lógica *Fuzzy*

Até o final do século XIX a comunidade científica não aceitava a incerteza em sistemas dinâmicos, pois representava um estado ineficaz que deveria ser evitado. O cientista tcheco Lofti A. Zadeh (professor da Universidade de Berkeley na Califórnia) apresentou a lógica *Fuzzy* como alternativa para problemas de lógica incompatíveis com a lógica clássica (CAVALCANTE, 2012).

A lógica *Fuzzy* conhecida também por teoria das possibilidades e lógica nebulosa diferencia-se por representar uma nova forma de manuseio de informações imprecisas, de uma forma muito distinta. Os sistemas *Fuzzy* são considerados sistemas inteligentes, pois representam esforços na direção da emulação da capacidade humana (CAVALCANTE, 2012). A lógica *Fuzzy* traduz expressões verbais, vagas, qualitativas e imprecisas, recorrentes na comunicação humana em valores numéricos, possibilitando a conversão da experiência humana em uma forma que seja compreensível pelos computadores e tornando possível a inclusão dessa experiência em tomadas de decisão em problemas complexos (SIMÕES & SHAW, 2007). Nessa lógica, ao contrário dos conjuntos binários clássicos, onde as variáveis podem assumir valores verdadeiro ou falso (0 ou 1), as variáveis podem ter um valor verdade que varia em grau entre 0 e 1 (NOVÀK *et al.*, 1999).

Um sistema de controle *Fuzzy* é definido como uma conjunção de conjuntos *Fuzzy* determinados por variáveis linguísticas de entrada e saída, acompanhadas de regras de controle *Fuzzy*, que fazem a ligação de um ou mais conjuntos *Fuzzy* de entrada a um conjunto *Fuzzy* de saída (ORTEGA, 2001). A estrutura básica de um controlador pode ser ilustrada na Figura 1. No módulo de fuzzificação os sinais dos sensores e dos atuadores devem ser convertidos em variáveis linguísticas. A seguir, o módulo de inferência utiliza o conjunto de regras definidos no módulo de base de conhecimentos para definir o comportamento do sistema e determinar o valor de saída, que continua sendo uma variável linguística. No final, o módulo de defuzzificação converte esse resultado linguístico em um valor real (SIMÕES & SHAW, 2007).

Figura 1 - Diagrama de um Controlador *Fuzzy*.



O modelo *Takagi-Sugeno* é um tipo de sistemas de inferência cujas regras são baseadas na seguinte Equação

$$Se\ x_1\ é\ A_{1i}\ e\ ...e\ x_n\ é\ A_{ni},\ então\ y = f_i(x_1,...,x_n) \tag{1}$$

onde $f(x_1,....,x_n)$ é uma função dependente das entradas do sistema x_i e cuja saída real é indicada pela Equação (2), não necessitando de uma etapa de defuzzificação.

$$y = \frac{\sum_{i=1}^{k} w_i f_i(x)}{\sum_{i=1}^{k} w_i}, \tag{2}$$

onde w_i representam as ativações de cada regra *Fuzzy*.

2.2. Tecnologias Assistivas para Deficientes Visuais

Existem diversos instrumentos no mercado que auxiliam na locomoção dos deficientes visuais. Alguns exemplos comuns que podem ser citados são bengalas e cães-guia. Apesar das bengalas serem acessíveis e mais utilizadas, elas não detectam obstáculos acima da linha da cintura. Além disso, o seu alcance é pequeno, aproximadamente 0,5 metros. Já o cão-guia possui maior eficiência por detectar e prever obstáculos que a bengala não localiza. Contudo, o seu treinamento é extenso, caro, a demanda solicitada não é suprida pela quantidade disponível desses animais e geralmente são incapazes de guiar o usuário em ambientes desconhecidos (HERSH & JOHNSON, 2008).

Um dos principais problemas das tecnologias assistivas é que, embora resolvam um problema tecnicamente, nem sempre obtêm aceitação por parte dos usuários. Isso ocorre porque a maioria desses produtos são desenvolvidos sem levar em consideração a avaliação por parte dos usuários e a real necessidade dos mesmos. Neste contexto é importante considerar que os usuários se relacionam com a tecnologia para exercer tarefas dentro de um contexto social, econômico, político e físico. Todos esses fatores devem ser levados em consideração para que dispositivos futuros sejam tanto tecnologicamente bem-sucedidos, como também amplamente utilizados (HERSH & JOHNSON, 2008).

2.3. Microcontroladores e Sistemas Embarcados

Os microcontroladores são conhecidos como a evolução dos clássicos circuitos digitais e são constituídos principalmente por um processador, memória, periféricos de entrada e saída, dispositivos de comunicação serial e temporizadores, além de serem programados por um software. Esse tipo de plataforma embarcada possibilita a realização de um conjunto de tarefas predefinidas designadas para um único sistema. Os microcontroladores são considerados um conjunto processador-software, sendo mais baratos, simples e compactos do que a lógica das portas digitais (PENIDO & TRINDADE, 2013).

Comumente os microcontroladores são utilizados para implementar sistemas embarcados. Esses sistemas geralmente são microprocessados e projetados de forma que a unidade de processamento seja inteiramente aplicada ao dispositivo ou sistema que será controlado. Ao contrário dos desktops e notebooks, que são dispositivos de finalidade geral, os sistemas embarcados realizam um grupo de tarefas preestabelecidas, satisfazendo requisitos e finalidades específicas (JÚNIOR & DUARTE, 2010).

Existem três grandes vantagens em se utilizar sistemas embarcados, o que os tornam bastante atrativos para diversos projetos:

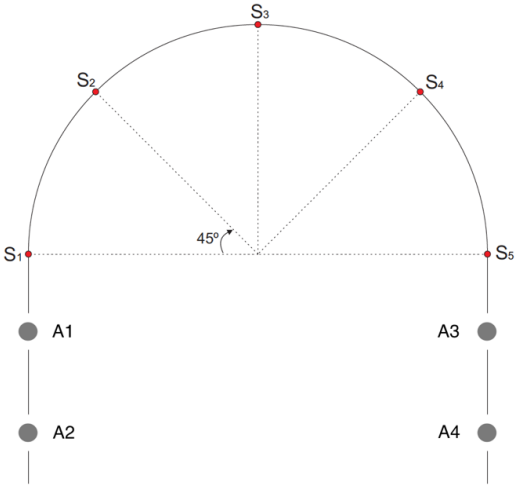
1. Um sistema embarcado realiza apenas um programa de forma repetida;
2. Um sistema embarcado possui custos não muito altos, tempo de resposta para processamento em tempo real e pouco consumo de potência;
3. Um sistema embarcado reage a mudanças e proporciona resultados em tempo real.

Os sistemas embarcados vêm sendo bastante implementados em sistemas de tecnologia assistiva, auxiliando na coleta de dados exteriores para manutenção da segurança de uma pessoa. Tais sistemas podem garantir independência e conforto ao usuário, seja ele deficiente físico ou visual. Adicionalmente, sistemas embarcados são bastante eficientes nas interações humano-computador e podem ser utilizados, por exemplo, na assistência de uma pessoa com deficiência visual a se locomover em ambientes desconhecidos.

3. DESENVOLVIMENTO DO DISPOSITIVO

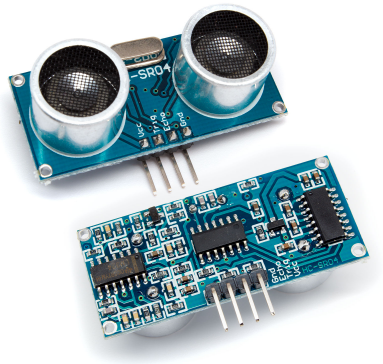
Para realização do protótipo, foram levantados dados e comparações. A partir dessa análise, foram decididos os componentes necessários que mais se adequam ao problema e ao projeto. Para tal protótipo, foram necessários cinco sensores ultrassônicos, quatro atuadores piezoelétricos e uma plataforma Arduino Mega. Um esquema generalizado da disposição dos sensores e atuadores no protótipo é ilustrado na Figura 2.

Figura 2 – Esquema da disposição dos sensores e atuadores no protótipo. S_i representam os sensores e A_i os atuadores.



Para a coleta dos dados de distância entre o usuário e os obstáculos foram utilizados sensores ultrassônicos. Ao todo foram utilizados cinco sensores, disponibilizados entre si em 45º, gerando um alcance de 180º. Os sensores ultrassônicos são dispositivos utilizados em diversas áreas de medição e utilizam-se da banda de frequência ultrassônica (maiores do que 20kHz) para seu funcionamento. As mudanças na variável de medição são determinadas pela variação no tempo que uma onda ultrassônica leva para ser emitida por um transmissor e captada por um receptor ou também pela variação da fase ou frequência da onda emitida (MORRIS, 2001). No projeto foi escolhido o sensor ultrassom HC-SR04 (HC-SR04 USER GUIDE, 2016), com tensão de alimentação de 5,5 V e alcance de medição entre 2 cm e 4 m. O sensor ultrassom utilizado é apresentado na Figura 3. As características do sensor são (a) baixo custo; (b) alcance entre 2 cm e 4m; (c) consumo de potência (corrente de alimentação de 15 mA); (d) desempenho estável; (e) medição acurada.

Figura 3 - Sensor Ultrassom HC-SR04.

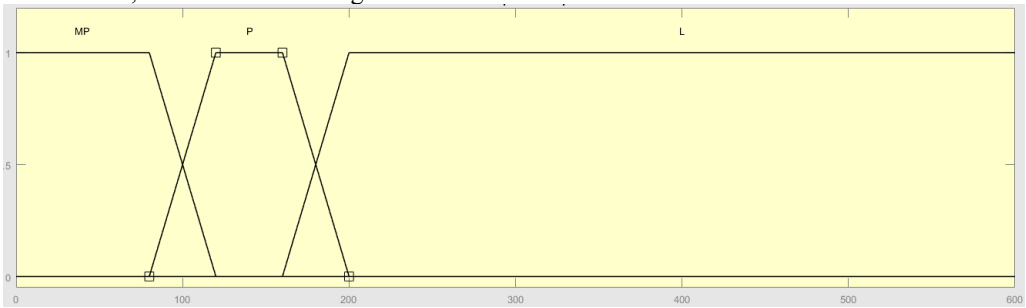


Para a construção do protótipo foi utilizada a plataforma Arduino Mega devido ao seu baixo custo e baixo consumo energético. O Arduino Mega é projetado com um microcontrolador ATmega1280, sendo designado como um computador em apenas um chip (PENIDO & TRINDADE, 2013).

Para o controle e processamento das informações foi utilizada a lógica *Fuzzy*. O controlador Takagi-Sugeno foi escolhido por ter cálculos mais simples de serem implementados no Arduino.

As variáveis linguísticas de entrada são representadas pelas distâncias dos sensores e cada variável foi dividida em termos linguísticos e esses representados pelas funções de pertinência de acordo com a Figura 4.

Figura 4 – Variável Linguística de entrada: distância em centímetros; termos linguísticos: MP – Muito Perto; P – Perto e L – Longe.



Como saída do sistema foram utilizados quatro LEDs emulando os atuadores piezoelétricos, disponibilizados no protótipo de acordo com a Figura 2. As saídas do controlador *Fuzzy* são valores de PWM (do inglês *Pulse Width Modulation*) que irão indicar a intensidade de luz nos LEDs. Foram escolhidas funções constantes com valores definidos de acordo com a Tabela 1. Essa intensidade do valor PWM indica, ao mesmo tempo, o quão próximo o objeto se encontra do usuário e o quão rápido deve ser a reação do mesmo.

Tabela 1 – Valores de saída das funções do controlador Takagi-Sugeno.

Nomenclatura	Valor PWM (Função constante)
OFF (Desligado)	0
B (Baixo)	30
M (Médio)	60
F (Forte)	90

Para o processo de inferência é necessário um conjunto de 243 regras de decisões pré-estabelecidas, derivadas dos três termos (Muito Perto, Perto e Longe) e das entradas dos cinco sensores, contabilizando 3^5 possibilidades. Parte da tabela de regras é apresentada na Figura 5.

Figura 5 – Base de regras parcial para as tomadas de decisões no sistema proposto.

	S1	S2	S3	S4	S5		A1	A2	A3	A4
1	MP	MP	MP	MP	MP	parar	OFF	OFF	OFF	OFF
2	MP	MP	MP	MP	P	direita virar	OFF	OFF	F	F
3	MP	MP	MP	MP	L	direita virar	OFF	OFF	F	F
4	MP	MP	MP	P	MP	parar	OFF	OFF	OFF	OFF
5	MP	MP	MP	P	P	direita diagonal	OFF	F	F	OFF
6	MP	MP	MP	P	L	direita diagonal	OFF	F	F	OFF
7	MP	MP	MP	L	MP	parar	OFF	OFF	OFF	OFF
8	MP	MP	MP	L	P	direita diagonal	OFF	F	F	OFF
9	MP	MP	MP	L	L	direita diagonal	OFF	F	F	OFF
10	MP	MP	P	MP	MP	parar	OFF	OFF	OFF	OFF
11	MP	MP	P	MP	P	parar	OFF	OFF	OFF	OFF
12	MP	MP	P	MP	L	direita virar	OFF	OFF	F	F
13	MP	MP	P	P	MP	parar	OFF	OFF	OFF	OFF
14	MP	MP	P	P	P	direita diagonal	OFF	F	F	OFF

4. RESULTADOS

Vários testes foram realizados para validar a proposta do controlador *Fuzzy*. Inicialmente foram realizadas simulações comportamentais usando o toolbox *Fuzzy* do Matlab® (vide Figuras 6). Em seguida o controlador *Fuzzy* foi implementado em código M (Matlab) e foram realizadas comparações numéricas entre ambas as implementações (vide Figuras 7 e 8).

Figura 6 – Toolbox *Fuzzy* do software Matlab® com o respectivo projeto.

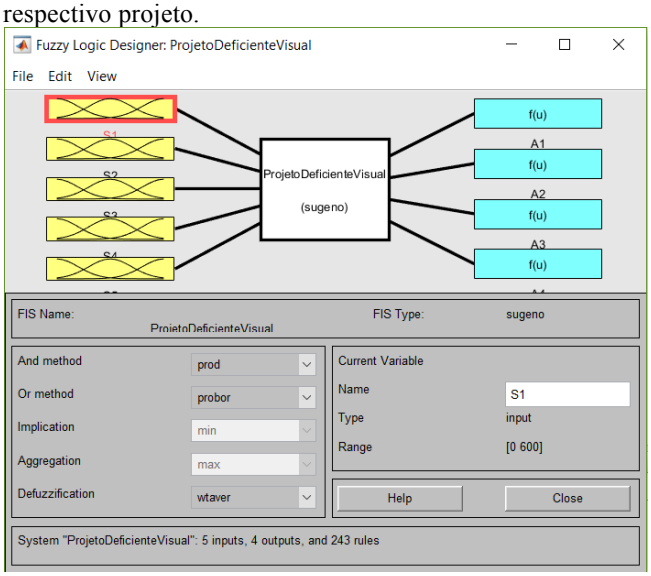


Figura 7 – Esquema de simulação para comparação numérica entre as implementações Matlab do controlador *Fuzzy* usando o toolbox *Fuzzy* e o código estruturado.

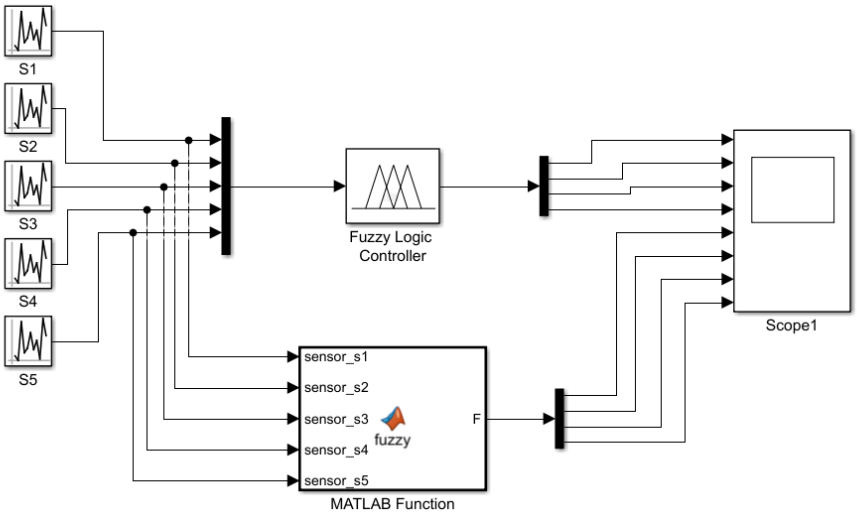
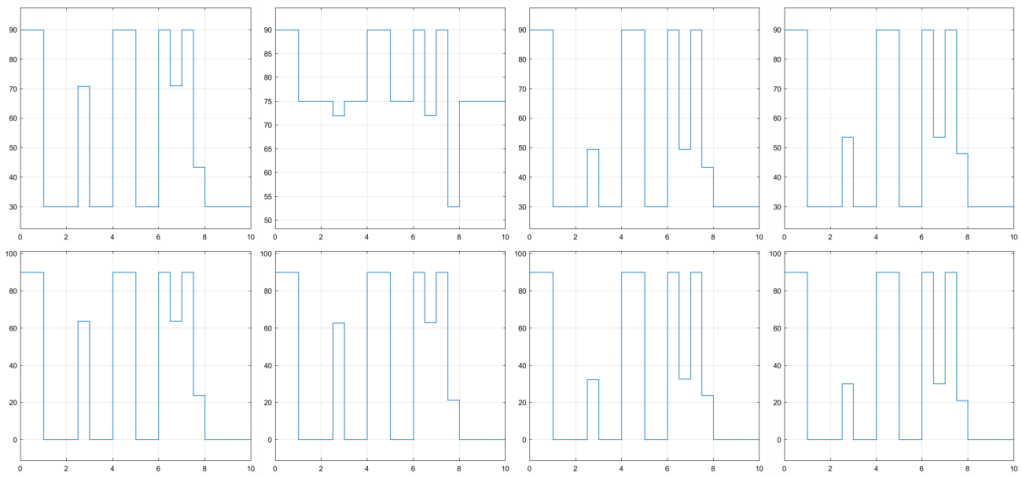


Figura 8 – Comparação dos resultados das implementações do controlador *Fuzzy*. Linha superior: sinal *dutycycle* dos atuadores obtidos pelo toolbox *Fuzzy* do Matlab. Linha inferior: sinal *dutycycle* dos atuadores obtidos pelo código estruturado.



Após a validação do controlador *Fuzzy* proposto, o mesmo foi embarcado no Arduíno Mega e um protótipo de baixo custo foi desenvolvido (vide Figura 9). Um vídeo demonstrando o funcionamento do protótipo construído está disponível em <https://youtu.be/X7OvCcgvUR8>.

Figura 9 – Protótipo de baixo custo desenvolvido.

Testes experimentais com obstáculos laterais, diagonais e frontais foram realizados. Na maioria dos testes foi evidenciado o correto funcionamento do sistema proposto, porém foi constatada a necessidade de ajuste do controlador *Fuzzy* no intuito de aprimorar o resultado. O ajuste foi realizado modificando a base de regras, sendo possível concluir que o mecanismo de regras de inferência da lógica *Fuzzy* permite realizar possíveis ajustes fáceis e rápidos. Este fato demonstra que o feedback do usuário pode melhorar a tabela de regras de forma simples, diferenciando-se de tecnologias de difícil aprimoramento por feedback do usuário após a implementação, tais como as Redes Neurais Artificiais (RNAs).

5. CONSIDERAÇÕES FINAIS E CONCLUSÕES

Esse trabalho propôs o desenvolvimento e validação do protótipo de um dispositivo didático para ensino de lógica *Fuzzy* em disciplinas de sistemas de controle. Em particular o protótipo desenvolvido permite o auxílio à mobilidade de deficientes visuais por meio da implementação de um controlador *Fuzzy* embarcado numa plataforma microcontrolada de baixo custo e de baixo consumo de energia. Os testes realizados com o protótipo desenvolvido se mostraram satisfatórios em termos da capacidade de desvio de obstáculos.

Em aplicações acadêmicas, a potencialidade da a utilização do protótipo como uma ferramenta didática para disciplinas de sistemas de controle pode permitir aos estudantes o uso de um dispositivo acessível e de fácil ajuste para aplicações baseadas em lógica *Fuzzy*. Adicionalmente, o dispositivo didático insere os estudantes a um tema transversal nas aulas relacionando ao uso de tecnologias focado principalmente na inserção digital dos deficientes visuais na sociedade (PCN, 2017).

Como trabalhos futuros pretende-se substituir os LEDs por atuadores piezoelétricos assim como implementar o dispositivo em formato de óculos de forma que possa ser testado por usuários com deficiência visual. Adicionalmente, espera-se fazer uso do dispositivo didático em sala de aula da disciplina de Sistemas de Controle do Instituto Federal de Goiás, Campus Jataí, visando coletar dados em relação ao processo de ensino-aprendizagem e à aceitação por parte dos estudantes.

REFERÊNCIAS BIBLIOGRÁFICAS

CAVALCANTE, J.H.F.et al. Lógica *Fuzzy* Aplicada Às Engenharias. João Pessoa: Câmara do Livro, 2012.

COMITÊ DE AJUDAS TÉCNICAS. Tecnologia Assistiva. Disponível em: <http://www.pessoacomdeficiencia.gov.br/app/sites/default/files/publicacoes/livro-tecnologia-assistiva.pdf > Acesso: 08 jun. 2016.

FERNANDES, N.; GUEDES, L. Disciplinas Práticas e Motivacionais nas Engenharias Mecânica e Mecatrônica. COBENGE, 2003

HC-SR04 User Guide. ELEC Freaks. Disponível em: <http://elecfreaks.com/estore/download/EF03085-HC-SR04_Ultrasonic_Module_User_Guide.pdf >. Acesso em: 25 jun. 2016.

HERSH, M. A.; JOHNSON, M. A. Assistive technology for visually impaired and blind people. Londres: Springer Verlag, 2008.

JÚNIOR, M. O.; DUARTE, R. O. Apostila sobre Introdução ao Projeto com Microcontroladores e Programação de Periféricos. Universidade Federal de Minas Gerais, 2010.

MORRIS, A. S. Measurament and Instrumentation Principles. Londres: Butterworth Heinemann, 2001.

NOVÀK, V.; PERFILIEVA, I.; MOCKOR, J. Mathematical principles of *Fuzzy* logic Dodrecht. Kluwer Academic, 1999.

OLIVEIRA, F. H. M. Uso de Realidade Aumentada na melhoria do processo de ensino-aprendizagem de motores elétricos. Instituto Federal de Goiás. 2010.

ORGANIZAÇÃO MUNDIAL DA SAÚDE. A deficiência visual e cegueira. Fact Sheet n. 282, agosto 2014. Disponível em: <http://www.who.int/mediacentre/ factsheets/fs282/en/>. Acesso em: 26 maio 2016.

ORTEGA, N.R.S. Aplicação da Teoria de Conjuntos *Fuzzy* a Problemas da Biomedicina. 2001. Tese de Doutorado – Instituto de Física da Universidade de São Paulo, São Paulo, 2001. Disponível em: < https://www.ime.usp.br/~tonelli/verao-*Fuzzy*/neli/principal.pdf >. Acesso em: 22 jun. 2016.

PCN (PARÂMETROS CURRICULARES NACIONAIS) – Saúde. Disponível em <http://portal.mec.gov.br/seb/arquivos/pdf/livro092.pdf>. Acesso em: 22 abr. 2017.

PENIDO, E.C.C; TRINDADE, R.S. Microcontroladores / Ouro Preto: Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais; Santa Maria: Universidade Federal de Santa Maria, Colégio Técnico Industrial de Santa Maria; Rede e-Tec Brasil, 2013.

PIAGET, Jean. Seis estudos de psicologia. Rio de Janeiro: Forense, v. 7, 1969.

SIMÕES, Marcelo Godoy; SHAW, Ian S. Controle e Modelagem *Fuzzy*. 2. ed., rev. e ampl. São Paulo: Blücher, 2007.

WEST, S.; SOMMER, A. Prevention of blindness and priorities for the future. Bull World Health Organ, vol. 79, n. 3, Genebra. Jan. 2001.

ZADEH, L.A.; *Fuzzy* Sets. Information and Control, v. 8, No. 3, pp. 338-353, 1965.

DIDACTIC DEVICE FOR OBSTACLE AVOIDANCE FOR VISUALLY IMPAIRED BASED ON *FUZZY* LOGIC EMBEDDED ON ARDUINO PLATFORM

Abstract: *Commonly, the teaching-learning process in control system courses is based on expositive talks regarding teoretical background and requiring expensive laboratorial infrastructure for practicing. The usage of low cost didactic kits can improve the learning process through the use of real case studies. In addition, the student's motivation in those courses can be improved by using transversal topics such as Assistive Technologies. This work proposes a system for detecting obstacles for the visually impaired using a prototype composed of five ultrasonic sensors spaced at 45°, allowing a scan of 180°. The device is controlled by the Takagi-Sugeno Fuzzy controller that was embedded on the Arduino platform where tests were carried out for project validation. The control system can be improved with feedback from end users, allowing decision rules to be adjusted, which is one of the advantages of the Fuzzy system: to be based on human experience. The proposed prototype has a low cost and can be used in a didactic and interdisciplinary way, involving disciplines such as control systems, microcontrollers, instrumentation, among others. In addition, this solution meets the WHO orientation, is in line with the ideals of the National Curricular Parameters (PCN), inserting the theme of health and its importance in the quality of life of citizens.*

Key-words: *Teaching in engineering, Asssitive technologies, Obstacle detection, Fuzzy controller, Embedded systems.*

B. BASE DE REGRAS

Tabela B.1: Banco de regras do Controlador *Fuzzy* Takagi-Sugeno

num	S1	S2	S3	S4	S5	Descrição	A1	A2	A3	A4
1	MP	MP	MP	MP	MP	parar	F	F	F	F
2	MP	MP	MP	MP	P	parar	F	F	F	F
3	MP	MP	MP	MP	L	direita virar	OFF	OFF	F	F
4	MP	MP	MP	P	MP	parar	F	F	F	F
5	MP	MP	MP	P	P	parar	F	F	F	F
6	MP	MP	MP	P	L	direita virar	OFF	OFF	F	M
7	MP	MP	MP	L	MP	parar	F	F	F	F
8	MP	MP	MP	L	P	parar	F	F	F	F
9	MP	MP	MP	L	L	diagonal direita	OFF	F	B	OFF
10	MP	MP	P	MP	MP	parar	F	F	F	F
11	MP	MP	P	MP	P	parar	F	F	F	F
12	MP	MP	P	MP	L	direita virar	OFF	OFF	F	F
13	MP	MP	P	P	MP	parar	F	F	F	F
14	MP	MP	P	P	P	parar	F	F	F	F
15	MP	MP	P	P	L	direita virar	OFF	OFF	F	M
16	MP	MP	P	L	MP	parar	F	F	F	F
17	MP	MP	P	L	P	parar	F	F	F	F
18	MP	MP	P	L	L	diagonal direita	OFF	F	B	OFF
19	MP	MP	L	MP	MP	seguir em frente	F	OFF	F	OFF
20	MP	MP	L	MP	P	seguir em frente	F	OFF	F	OFF
21	MP	MP	L	MP	L	seguir em frente	F	OFF	F	OFF
22	MP	MP	L	P	MP	seguir em frente	F	OFF	F	OFF
23	MP	MP	L	P	P	seguir em frente	F	OFF	M	OFF
24	MP	MP	L	P	L	seguir em frente	F	OFF	B	OFF
25	MP	MP	L	L	MP	seguir em frente	F	OFF	F	OFF
26	MP	MP	L	L	P	seguir em frente	F	OFF	M	OFF
27	MP	MP	L	L	L	seguir em frente	F	OFF	B	OFF
28	MP	P	MP	MP	MP	parar	F	F	F	F
29	MP	P	MP	MP	P	parar	F	F	F	F
30	MP	P	MP	MP	L	direita virar	OFF	OFF	F	F
31	MP	P	MP	P	MP	parar	F	F	F	F
32	MP	P	MP	P	P	parar	F	F	F	F
33	MP	P	MP	P	L	direita virar	OFF	OFF	F	M
34	MP	P	MP	L	MP	parar	F	F	F	F

Continua na próxima página

Tabela B.1 – *Continuação da página anterior*

	S1	**S2**	**S3**	**S4**	**S5**	atividade	**A1**	**A2**	**A3**	**A4**
35	MP	P	MP	L	P	parar	F	F	F	F
36	MP	P	MP	L	L	diagonal direita	OFF	F	B	OFF
37	MP	P	P	MP	MP	parar	F	F	F	F
38	MP	P	P	MP	P	parar	F	F	F	F
39	MP	P	P	MP	L	direita virar	OFF	OFF	F	F
40	MP	P	P	P	MP	parar	F	F	F	F
41	MP	P	P	P	P	parar	F	F	F	F
42	MP	P	P	P	L	direita virar	OFF	OFF	F	M
43	MP	P	P	L	MP	parar	F	F	F	F
44	MP	P	P	L	P	parar	F	F	F	F
45	MP	P	P	L	L	diagonal direita	OFF	F	B	OFF
46	MP	P	L	MP	MP	seguir em frente	F	OFF	F	OFF
47	MP	P	L	MP	P	seguir em frente	F	OFF	F	OFF
48	MP	P	L	MP	L	seguir em frente	F	OFF	F	OFF
49	MP	P	L	P	MP	seguir em frente	F	OFF	F	OFF
50	MP	P	L	P	P	seguir em frente	F	OFF	M	OFF
51	MP	P	L	P	L	seguir em frente	F	OFF	B	OFF
52	MP	P	L	L	MP	seguir em frente	F	OFF	F	OFF
53	MP	P	L	L	P	seguir em frente	F	OFF	M	OFF
54	MP	P	L	L	L	seguir em frente	F	OFF	B	OFF
55	MP	L	MP	MP	MP	parar	F	F	F	F
56	MP	L	MP	MP	P	parar	F	F	F	F
57	MP	L	MP	MP	L	direita virar	OFF	OFF	F	F
58	MP	L	MP	P	MP	parar	F	F	F	F
59	MP	L	MP	P	P	parar	F	F	F	F
60	MP	L	MP	P	L	direita virar	OFF	OFF	F	M
61	MP	L	MP	L	MP	parar	F	F	F	F
62	MP	L	MP	L	P	parar	F	F	F	F
63	MP	L	MP	L	L	diagonal direita	OFF	F	B	OFF
64	MP	L	P	MP	MP	parar	F	F	F	F
65	MP	L	P	MP	P	parar	F	F	F	F
66	MP	L	P	MP	L	parar	F	F	F	F
67	MP	L	P	P	MP	parar	F	F	F	F
68	MP	L	P	P	P	parar	F	F	F	F
69	MP	L	P	P	L	direita virar	OFF	OFF	F	M
70	MP	L	P	L	MP	parar	F	F	F	F
71	MP	L	P	L	P	parar	F	F	F	F
72	MP	L	P	L	L	diagonal direita	OFF	F	B	OFF

Continua na próxima página

Tabela B.1 – *Continuação da página anterior*

	S1	**S2**	**S3**	**S4**	**S5**	atividade	**A1**	**A2**	**A3**	**A4**
73	MP	L	L	MP	MP	seguir em frente	F	OFF	F	OFF
74	MP	L	L	MP	P	seguir em frente	F	OFF	F	OFF
75	MP	L	L	MP	L	seguir em frente	F	OFF	F	OFF
76	MP	L	L	P	MP	seguir em frente	F	OFF	F	OFF
77	MP	L	L	P	P	seguir em frente	F	OFF	M	OFF
78	MP	L	L	P	L	seguir em frente	F	OFF	B	OFF
79	MP	L	L	L	MP	seguir em frente	F	OFF	F	OFF
80	MP	L	L	L	P	seguir em frente	F	OFF	M	OFF
81	MP	L	L	L	L	seguir em frente	F	OFF	B	OFF
82	P	MP	MP	MP	MP	parar	F	F	F	F
83	P	MP	MP	MP	P	parar	F	F	F	F
84	P	MP	MP	MP	L	direita virar	OFF	OFF	F	F
85	P	MP	MP	P	MP	parar	F	F	F	F
86	P	MP	MP	P	P	parar	F	F	F	F
87	P	MP	MP	P	L	direita virar	OFF	OFF	F	M
88	P	MP	MP	L	MP	parar	F	F	F	F
89	P	MP	MP	L	P	parar	F	F	F	F
90	P	MP	MP	L	L	diagonal direita	OFF	F	B	OFF
91	P	MP	P	MP	MP	parar	F	F	F	F
92	P	MP	P	MP	P	parar	F	F	F	F
93	P	MP	P	MP	L	direita virar	OFF	OFF	F	F
94	P	MP	P	P	MP	parar	F	F	F	F
95	P	MP	P	P	P	parar	F	F	F	F
96	P	MP	P	P	L	direita virar	OFF	OFF	F	M
97	P	MP	P	L	MP	parar	F	F	F	F
98	P	MP	P	L	P	parar	F	F	F	F
99	P	MP	P	L	L	diagonal direita	OFF	F	B	OFF
100	P	MP	L	MP	MP	seguir em frente	F	OFF	F	OFF
101	P	MP	L	MP	P	seguir em frente	F	OFF	F	OFF
102	P	MP	L	MP	L	seguir em frente	F	OFF	F	OFF
103	P	MP	L	P	MP	seguir em frente	F	OFF	F	OFF
104	P	MP	L	P	P	seguir em frente	F	OFF	M	OFF
105	P	MP	L	P	L	seguir em frente	F	OFF	M	OFF
106	P	MP	L	L	MP	seguir em frente	F	OFF	F	OFF
107	P	MP	L	L	P	seguir em frente	F	OFF	M	OFF
108	P	MP	L	L	L	seguir em frente	F	OFF	B	OFF
109	P	P	MP	MP	MP	parar	F	F	F	F
110	P	P	MP	MP	P	parar	F	F	F	F

Continua na próxima página

Tabela B.1 – *Continuação da página anterior*

	S1	**S2**	**S3**	**S4**	**S5**	atividade	**A1**	**A2**	**A3**	**A4**
111	P	P	MP	MP	L	direita virar	OFF	OFF	M	F
112	P	P	MP	P	MP	parar	F	F	F	F
113	P	P	MP	P	P	parar	F	F	F	F
114	P	P	MP	P	L	direita virar	OFF	OFF	M	F
115	P	P	MP	L	MP	parar	F	F	F	F
116	P	P	MP	L	P	parar	F	F	F	F
117	P	P	MP	L	L	diagonal direita	OFF	M	B	OFF
118	P	P	P	MP	MP	parar	F	F	F	F
119	P	P	P	MP	P	parar	F	F	F	F
120	P	P	P	MP	L	direita virar	OFF	OFF	M	F
121	P	P	P	P	MP	parar	F	F	F	F
122	P	P	P	P	P	parar	F	F	F	F
123	P	P	P	P	L	direita virar	OFF	OFF	M	M
124	P	P	P	L	MP	parar	F	F	F	F
125	P	P	P	L	P	parar	F	F	F	F
126	P	P	P	L	L	diagonal direita	OFF	M	B	OFF
127	P	P	L	MP	MP	seguir em frente	M	OFF	F	OFF
128	P	P	L	MP	P	seguir em frente	M	OFF	F	OFF
129	P	P	L	MP	L	seguir em frente	M	OFF	F	OFF
130	P	P	L	P	MP	seguir em frente	M	OFF	F	OFF
131	P	P	L	P	P	seguir em frente	M	OFF	M	OFF
132	P	P	L	P	L	seguir em frente	M	OFF	M	OFF
133	P	P	L	L	MP	seguir em frente	M	OFF	F	OFF
134	P	P	L	L	P	seguir em frente	M	OFF	M	OFF
135	P	P	L	L	L	seguir em frente	M	OFF	B	OFF
136	P	L	MP	MP	MP	parar	F	F	F	F
137	P	L	MP	MP	P	parar	F	F	F	F
138	P	L	MP	MP	L	parar	F	F	F	F
139	P	L	MP	P	MP	parar	F	F	F	F
140	P	L	MP	P	P	parar	F	F	F	F
141	P	L	MP	P	L	diagonal direita	OFF	M	M	OFF
142	P	L	MP	L	MP	parar	F	F	F	F
143	P	L	MP	L	P	parar	F	F	F	F
144	P	L	MP	L	L	diagonal direita	OFF	M	B	OFF
145	P	L	P	MP	MP	parar	F	F	F	F
146	P	L	P	MP	P	parar	F	F	F	F
147	P	L	P	MP	L	parar	F	F	F	F
148	P	L	P	P	MP	parar	F	F	F	F

Continua na próxima página

Tabela B.1 – *Continuação da página anterior*

	S1	**S2**	**S3**	**S4**	**S5**	atividade	**A1**	**A2**	**A3**	**A4**
149	P	L	P	P	P	parar	F	F	F	F
150	P	L	P	P	L	diagonal direita	OFF	M	M	OFF
151	P	L	P	L	MP	parar	F	F	F	F
152	P	L	P	L	P	parar	F	F	F	F
153	P	L	P	L	L	diagonal direita	OFF	M	B	OFF
154	P	L	L	MP	MP	seguir em frente	M	OFF	F	OFF
155	P	L	L	MP	P	seguir em frente	M	OFF	F	OFF
156	P	L	L	MP	L	seguir em frente	M	OFF	F	OFF
157	P	L	L	P	MP	seguir em frente	M	OFF	F	OFF
158	P	L	L	P	P	seguir em frente	M	OFF	M	OFF
159	P	L	L	P	L	seguir em frente	M	OFF	M	OFF
160	P	L	L	L	MP	seguir em frente	M	OFF	F	OFF
161	P	L	L	L	P	seguir em frente	M	OFF	M	OFF
162	P	L	L	L	L	seguir em frente	M	OFF	B	OFF
163	L	MP	MP	MP	MP	esquerda virar	F	F	OFF	OFF
164	L	MP	MP	MP	P	esquerda virar	F	F	OFF	OFF
165	L	MP	MP	MP	L	direita virar	OFF	OFF	F	F
166	L	MP	MP	P	MP	esquerda virar	F	F	OFF	OFF
167	L	MP	MP	P	P	esquerda virar	F	F	OFF	OFF
168	L	MP	MP	P	L	direita virar	OFF	OFF	F	M
169	L	MP	MP	L	MP	esquerda virar	F	F	OFF	OFF
170	L	MP	MP	L	P	esquerda virar	F	M	OFF	OFF
171	L	MP	MP	L	L	diagonal direita	OFF	F	M	OFF
172	L	MP	P	MP	MP	esquerda virar	F	F	OFF	OFF
173	L	MP	P	MP	P	esquerda virar	F	F	OFF	OFF
174	L	MP	P	MP	L	direita virar	OFF	OFF	F	F
175	L	MP	P	P	MP	esquerda virar	F	F	OFF	OFF
176	L	MP	P	P	P	esquerda virar	F	M	OFF	OFF
177	L	MP	P	P	L	direita virar	OFF	OFF	F	M
178	L	MP	P	L	MP	esquerda virar	F	F	OFF	OFF
179	L	MP	P	L	P	esquerda virar	F	M	OFF	OFF
180	L	MP	P	L	L	diagonal direita	OFF	F	B	OFF
181	L	MP	L	MP	MP	seguir em frente	F	OFF	F	OFF
182	L	MP	L	MP	P	seguir em frente	F	OFF	F	OFF
183	L	MP	L	MP	L	seguir em frente	F	OFF	F	OFF
184	L	MP	L	P	MP	seguir em frente	F	OFF	F	OFF
185	L	MP	L	P	P	seguir em frente	F	OFF	M	OFF
186	L	MP	L	P	L	seguir em frente	F	OFF	B	OFF

Continua na próxima página

Tabela B.1 – *Continuação da página anterior*

	S1	**S2**	**S3**	**S4**	**S5**	atividade	**A1**	**A2**	**A3**	**A4**
187	L	MP	L	L	MP	seguir em frente	F	OFF	F	OFF
188	L	MP	L	L	P	seguir em frente	F	OFF	M	OFF
189	L	MP	L	L	L	seguir em frente	F	OFF	B	OFF
190	L	P	MP	MP	MP	esquerda virar	M	F	OFF	OFF
191	L	P	MP	MP	P	esquerda virar	M	F	OFF	OFF
192	L	P	MP	MP	L	direita virar	OFF	OFF	M	F
193	L	P	MP	P	MP	esquerda virar	M	F	OFF	OFF
194	L	P	MP	P	P	esquerda virar	M	M	OFF	OFF
195	L	P	MP	P	L	direita virar	OFF	OFF	M	M
196	L	P	MP	L	MP	esquerda virar	M	F	OFF	OFF
197	L	P	MP	L	P	esquerda virar	M	M	OFF	OFF
198	L	P	MP	L	L	diagonal direita	OFF	M	B	OFF
199	L	P	P	MP	MP	esquerda virar	M	F	OFF	OFF
200	L	P	P	MP	P	esquerda virar	M	F	OFF	OFF
201	L	P	P	MP	L	direita virar	OFF	OFF	M	F
202	L	P	P	P	MP	esquerda virar	M	F	OFF	OFF
203	L	P	P	P	P	esquerda virar	M	M	OFF	OFF
204	L	P	P	P	L	direita virar	OFF	OFF	M	M
205	L	P	P	L	MP	esquerda virar	M	F	OFF	OFF
206	L	P	P	L	P	esquerda virar	M	M	OFF	OFF
207	L	P	P	L	L	diagonal direita	OFF	M	B	OFF
208	L	P	L	MP	MP	seguir em frente	M	OFF	F	OFF
209	L	P	L	MP	P	seguir em frente	M	OFF	F	OFF
210	L	P	L	MP	L	seguir em frente	M	OFF	F	OFF
211	L	P	L	P	MP	seguir em frente	M	OFF	F	OFF
212	L	P	L	P	P	seguir em frente	M	OFF	M	OFF
213	L	P	L	P	L	seguir em frente	M	OFF	M	OFF
214	L	P	L	L	MP	seguir em frente	M	OFF	F	OFF
215	L	P	L	L	P	seguir em frente	M	OFF	M	OFF
216	L	P	L	L	L	seguir em frente	M	OFF	B	OFF
217	L	L	MP	MP	MP	esquerda diagonal	B	OFF	OFF	F
218	L	L	MP	MP	P	esquerda diagonal	B	OFF	OFF	F
219	L	L	MP	MP	L	direita virar	OFF	OFF	B	F
220	L	L	MP	P	MP	esquerda diagonal	B	OFF	OFF	F
221	L	L	MP	P	P	esquerda diagonal	B	OFF	OFF	M
222	L	L	MP	P	L	direita virar	OFF	OFF	F	F
223	L	L	MP	L	MP	esquerda diagonal	B	OFF	OFF	F
224	L	L	MP	L	P	esquerda diagonal	B	OFF	OFF	M

Continua na próxima página

Tabela B.1 – *Continuação da página anterior*

	S1	**S2**	**S3**	**S4**	**S5**	atividade	**A1**	**A2**	**A3**	**A4**
225	L	L	MP	L	L	diagonal direita	OFF	B	B	OFF
226	L	L	P	MP	MP	esquerda diagonal	B	OFF	OFF	F
227	L	L	P	MP	P	esquerda diagonal	B	OFF	OFF	F
228	L	L	P	MP	L	direita virar	OFF	OFF	B	F
229	L	L	P	P	MP	esquerda diagonal	B	OFF	OFF	F
230	L	L	P	P	P	esquerda diagonal	B	OFF	OFF	M
231	L	L	P	P	L	direita virar	OFF	OFF	B	M
232	L	L	P	L	MP	esquerda diagonal	B	OFF	OFF	F
233	L	L	P	L	P	esquerda diagonal	B	OFF	OFF	M
234	L	L	P	L	L	diagonal direita	OFF	B	B	OFF
235	L	L	L	MP	MP	seguir em frente	B	OFF	F	OFF
236	L	L	L	MP	P	seguir em frente	B	OFF	F	OFF
237	L	L	L	MP	L	seguir em frente	B	OFF	F	OFF
238	L	L	L	P	MP	seguir em frente	B	OFF	F	OFF
239	L	L	L	P	P	seguir em frente	B	OFF	M	OFF
240	L	L	L	P	L	seguir em frente	B	OFF	M	OFF
241	L	L	L	L	MP	seguir em frente	B	OFF	F	OFF
242	L	L	L	L	P	seguir em frente	B	OFF	M	OFF
243	L	L	L	L	L	seguir em frente	B	OFF	B	OFF