Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências
Faculdade de Engenharia

André Luís Jorge Garcia

Implementação eletrônica de sistemas Fuzzy

Rio de janeiro
2009

André Luís Jorge Garcia

Implementação eletrônica de sistemas Fuzzy

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação

Orientador: Prof. Dr. José Franco Machado do Amaral

Rio de janeiro
2009

CATALOGAÇÃO NA FONTE
UERJ/REDE SIRIUS/CTC/B

G216 Garcia, André Luis Jorge.
Implementação eletrônica de sistemas fuzzy/ André Luis Jorge Garcia. – 2009.
113f.: il.

Orientador: José Franco Machado do Amaral.
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.
Bibliografia: f.108
1. Arquitetura de computador. 2. Engenharia de computador. I. Amaral, José Franco Machado do. II. Universidade do Estado do Rio de Janeiro. Faculdade de Engenharia. III. Título.

CDU
004.382.7

Assinatura

06 de Agosto de 2009
Data

André Luís Jorge Garcia

Implementação eletrônica de sistemas Fuzzy

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação

Aprovado em 06 de Agosto de 2009

Banca Examinadora:

Prof. Dr. José Franco Machado do Amaral (Orientador)
Faculdade de Engenharia, UERJ

Prof. Dr Jorge Luís Machado do Amaral
Faculdade de Engenharia, UERJ

Prof. Dr. Antonio Carneiro de Mesquita Filho
Programa de Engenharia Elétrica, COPPE/UFRJ

Rio de Janeiro
2009

Dedicatória

Dedico aos meus amados pais, por nunca desistirem, sempre acreditando e apoiando com todo amor.

Agradecimentos

A Deus por estar sempre comigo;

À minha família pelo carinho e apoio recebido por todos estes anos, principalmente na compreensão de minhas amadas filhas nesta jornada;

Ao meu orientador Professor José Franco Machado do Amaral pelo estímulo, parceria e ensinamentos para a realização deste trabalho;

Aos meus amigos da UERJ , Samuel, Humberto, Antonio, Joaquim, Daniel, Renato e Marcos Paulo, pelos trabalhos e apoio durante este período;

À Universidade do Estado do Rio de Janeiro - UERJ por ter me recebido e colaborado na realização do mestrado;

Aos funcionários e professores do Programa de Pós-graduação em Engenharia Eletrônica - PEL, pela ajuda e ensinamentos;

Aos professores que participaram da Comissão examinadora;

A todos os amigos que de uma forma ou de outra me estimularam ou me ajudaram.

RESUMO

GARCIA, André Luís Jorge, Implementação Eletrônica de Sistemas Fuzzy. 2009. 115f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2009.

Este trabalho investiga a implementação de sistemas fuzzy com circuitos eletrônicos. Tais sistemas têm demonstrado sua capacidade de resolver diversos tipos de problemas em várias aplicações de engenharia, em especial nas relacionadas com controle de processos. Para processos mais complexos, o raciocínio aproximado da lógica fuzzy fornece uma maneira de compreender o comportamento do sistema, permitindo a interpolação aproximada entre situações observadas de entrada e saída. A implementação de um sistema fuzzy pode ser baseada em hardware, em software ou em ambos. Tipicamente, as implementações em software utilizam ambientes de programação integrados com simulação, de modo a facilitar o trabalho do projetista. As implementações em hardware, tradicionais ou evolutivas, podem ser analógicas ou digitais e viabilizam sistemas de maior desempenho. Este trabalho tem por objetivo pesquisar a implementação eletrônica de sistemas fuzzy, a fim de viabilizar a criação de sistemas reais capazes de realizar o mapeamento de entrada e saída adequado. O foco é a utilização de uma plataforma com uma arquitetura analógico-digital baseada em uma tabela de mapeamento armazenada em uma memória de alta capacidade. Memórias do tipo SD (*Secure Digital*) foram estudadas e utilizadas na construção do protótipo eletrônico da plataforma. Também foram desenvolvidos estudos sobre a quantização, especificamente sobre a possibilidade de redução do número de bits. Com a implementação realizada é possível desenvolver um sistema fuzzy num ambiente simulado (*Matlab*), configurar a plataforma e executar o sistema fuzzy diretamente na plataforma eletrônica. Os testes com o protótipo construído comprovaram seu bom funcionamento.

Palavras-chave: Sistemas Fuzzy, Implementação eletrônica, Plataforma Microcontrolada, Memória SD, Interface USB.

ABSTRACT

This work investigates the implementation of fuzzy systems using electronic currents. Such systems have been used before to solve several of problems of engineering applications, mainly involving process control applications. On more complex applications, the approximate reasoning of the fuzzy logic allows a way to understand the system behavior, allowing approximate interpolation among observed sets of input and output points. The implementation of a fuzzy system can be based in hardware, software or both. Typically, the software implementation uses a programming environment integrated with simulation, helping the designing work. The hardware implementations, traditional or evolutionary, can be analog or digital, mainly for high performance systems. This work aims to research an electronic implementation of a fuzzy system, capable to accomplish an adequate input to output mapping. The focus of this work is to design a platform with an analog-digital architecture based in a mapping table stored in a high capacity memory. Memories of the SD (Secure Digital) type were studied and used in the construction of a prototype of the electronic platform. Also studies were developed on the quantization, specifically to allow the reduction of the number of bits. With the accomplished implementation, it is possible to develop a fuzzy system in a simulated environment (Matlab), to configure the platform and to execute the fuzzy system directly in the electronic platform. The tests with the prototype was successful.

Keywords: Systems fuzzy, electronic implementation, microcontroller platform, memory card SD, interface USB.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

Os sistemas inteligentes baseados em lógica fuzzy têm demonstrado sua capacidade de resolver diversos tipos de problemas em várias aplicações de engenharia e na vida cotidiana. Cabe ressaltar o crescente interesse nas aplicações que tratam de controle de processos. Em 1965, Lofti A. Zadeh introduziu a teoria dos conjuntos fuzzy. Em seu artigo “Fuzzy Sets", ele formalizou suas idéias sobre uma nova ferramenta matemática que utiliza conhecimento e incertezas. A proposta de Zadeh era modelar o mecanismo do pensamento humano com valores lingüísticos, gerando uma nova classe de sistemas denominada de sistemas fuzzy. A lógica fuzzy é baseada na teoria dos conjuntos fuzzy e em regras fuzzy (do tipo se-então) e está intimamente relacionada à lingüística e à ciência da cognição. Os sistemas fuzzy são adequados para a criação de modelos a partir de um conhecimento explícito (racional e lingüisticamente tratável) de especialistas humanos, as denominadas informações subjetivas.

Considerando-se um determinado sistema real, pode-se relacionar sua complexidade com a precisão de seu modelo segundo o Princípio da Incompatibilidade (ZADEH, 1965).

“Conforme a complexidade de um sistema aumenta, a nossa habilidade de fazer declarações precisas e significativas sobre o comportamento do sistema diminui, até alcançar um limite além do qual precisão e relevância se tornam características mutuamente exclusivas.”

Para sistemas de menor complexidade ou de características bem definidas, e, portanto, de menor incerteza, expressões matemáticas podem fornecer descrições precisas sobre eles. No caso de sistemas um pouco mais complexos para os quais exista uma quantidade significativa de dados (informações objetivas), modelos computacionais não-lineares, tais como as Redes Neurais, fornecem um meio robusto e poderoso para reduzir alguma incerteza através do aprendizado baseado em padrões existentes nos dados disponíveis. Finalmente, para os sistemas mais complexos onde existem poucos dados numéricos e apenas informações imprecisas ou ambíguas (subjetivas) estão disponíveis, o raciocínio aproximado da lógica fuzzy fornece uma maneira de compreender (explicar) o comportamento do sistema, permitindo a interpolação aproximada entre situações observadas de entrada e saída.

Os sistemas fuzzy têm demonstrado sua capacidade de resolver diversos tipos de problemas em várias aplicações de engenharia, em especial nas relacionadas com controle de processos. Tipicamente, a implementação de um sistema fuzzy pode ser baseada em hardware, em software ou em ambos.

As implementações em software costumam utilizar ambientes com simulação, de modo a facilitar o trabalho do projetista. Nesses casos, a análise do sistema fica imune às interferências que poderiam ser analisadas se o sistema estivesse operando diretamente em uma plataforma de hardware.

As implementações em hardware podem ser com circuitos analógicos, com FPGAs, com FPAAs, com microprocessadores ou sistemas digitais puros. Elas viabilizam sistemas de maior desempenho. Também é possível a aplicação de técnicas evolutivas para projetar os sistemas fuzzy (AMARAL, 2003).

O objetivo principal deste trabalho é pesquisar a implementação eletrônica de sistemas fuzzy, a fim de viabilizar a criação de sistemas reais capazes de realizar o mapeamento de entrada e saída adequado. O foco é a concepção e a construção de uma plataforma analógico-digital baseada em uma tabela de mapeamento armazenada em uma memória SD (*Secure Digital*) de alta capacidade. Esta plataforma funciona como um mapeador não-linear analógico-digital (vide Figura 1). Além dos estudos realizados sobre a tecnologia de memória utilizada, foram desenvolvidos estudos sobre a quantização, mais especificamente sobre a possibilidade de redução do número de bits.

Figura 1 – Mapeador Analógico-Digital

Tais estudos levaram à construção de um protótipo eletrônico funcional da plataforma proposta. Com a implementação realizada é possível desenvolver um sistema fuzzy num ambiente simulado (*Matlab*), configurar a plataforma e executar o sistema fuzzy diretamente na plataforma eletrônica (vide Figura 2). Testes realizados com o protótipo construído comprovaram seu bom funcionamento.

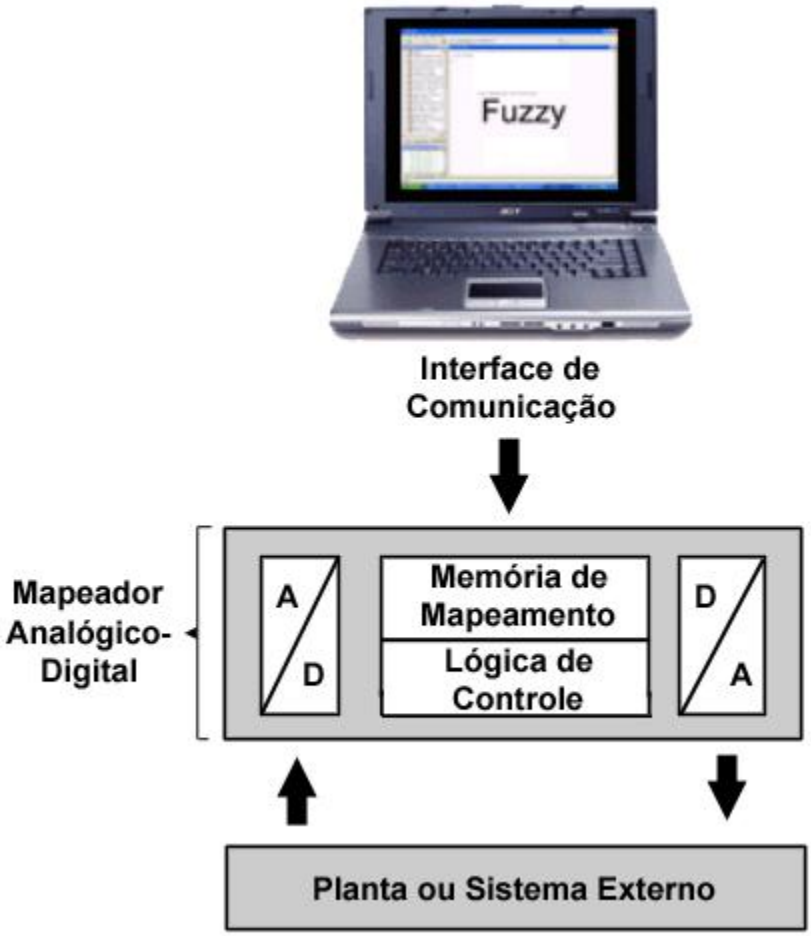


Figura 2 – Diagrama de Blocos da Plataforma de Desenvolvimento

O protótipo possui 3 canais analógicos de entrada que transformam os sinais externos em sinais digitais, de no máximo 8 bits, capazes de endereçar uma memória de mapeamento, devidamente preenchida de acordo com o sistema fuzzy implementado. Em sua saída é utilizado um conversor D/A para converter o sinal digital proveniente da memória de mapeamento em um sinal analógico capaz de atuar no sistema externo.

Algumas dificuldades do passado foram superadas pelo avanço da tecnologia e pela diminuição dos custos da implementação eletrônica. Como por exemplo, podemos citar a necessidade de uma grande quantidade de memória para armazenar todos os casos possíveis de inferência em uma plataforma baseada em mapeador analógico-digital. Com as memórias de grande velocidade e alta capacidade de armazenamento, este problema foi resolvido.

A proposta apresentada possui a característica de ter as entradas convertidas para um valor binário, sendo a quantidade de bits utilizados para esta conversão chamada de quantização. Quanto maior o número de bits usados na conversão, maior a precisão numérica dos cálculos. Porém, a desvantagem de aumentar o número de bits de uma conversão está no fato de necessitar de um maior tempo para executar esta conversão, além de uma alocação maior de memória nas implementações de hardware fuzzy via mapeamento digital.

A arquitetura escolhida é baseada em um mapeador analógico-digital, com o diferencial de ter um acumulador de uso das entradas, aproveitando o excedente de memória livre que tem a plataforma, conforme ilustrado na Figura 3. Neste trabalho também está proposta uma análise em relação à quantização binária das entradas, visando observar se as entradas estão com sua quantização binária em um nível adequado.

Um aspecto interessante da pesquisa foi a implementação de um acumulador de uso das entradas, alocando parte da memória interna do mapeador para guardar incidências de apresentação dos sinais em cada uma das entradas. A metodologia foi a de separar três faixas da memória, fora da área mapeada, e usar um somador para cada uma das possibilidades em cada entrada. Como temos cada entrada com no máximo 8 bits, formando uma quantização entre 0 e 255 decimal, teremos uma alocação de 256 somadores para cada entrada da plataforma. Estes acumuladores em tempo real verificam se todos os valores produzidos no sistema fuzzy estão realmente sendo utilizados pelo hardware durante sua aplicação. Este “acumulador”, ou supervisor, irá fornecer uma estatística de uso das entradas pelo sistema em um tempo determinado, o que pode ser vantajoso para analisar a operação do sistema. Ele também pode verificar que certas condições nunca vieram a se concretizar, facilitando a análise dessas situações.

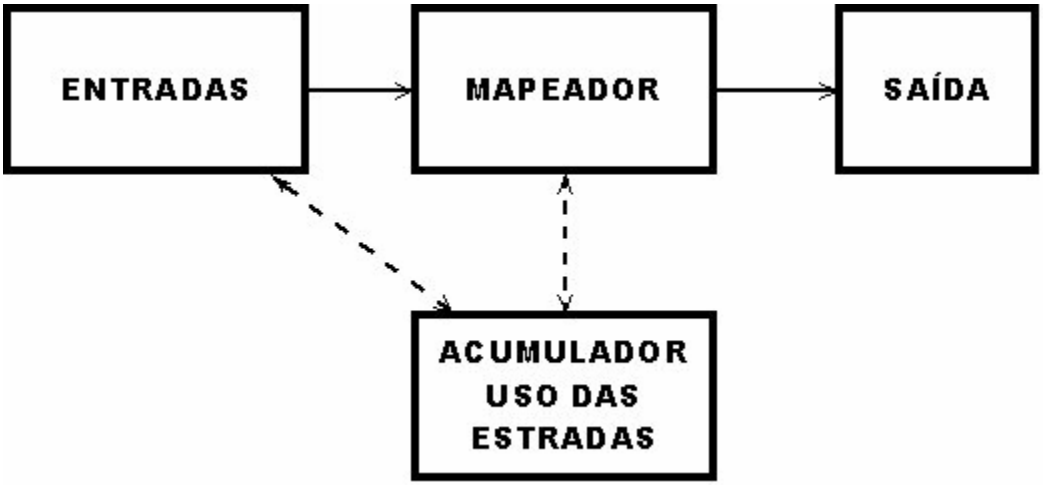


Figura 3 – Sistema Geral do Mapeador

Esta verificação é implementada facilmente utilizando esta técnica de mapeador analógico-digital, pois permite verificar se determinadas entradas foram muito usadas, pouco usadas ou nunca usadas.

Um estudo das memórias do tipo SD, com grande capacidade de armazenamento e rapidez quanto ao acesso dos dados, viabilizou a implementação da arquitetura escolhida.

No estudo da sensibilidade das entradas foi verificado os erros máximos em diversas quantizações, comparando as respostas no simulador com as apresentadas na plataforma. Para esta análise, a plataforma foi desenvolvida com a opção de se reduzir a quantização binária da conversão analógico-digital de suas entradas via comando de software inerente ao projeto da mesma, possibilitando a pesquisa de diversos casos.

Este trabalho representa um passo inicial e consistente no sentido de permitir que a implementação de sistemas fuzzy baseada em mapeador analógico-digital seja utilizada no projeto e em testes de sistemas inteligentes.

O texto está organizado em sete capítulos adicionais conforme descrito a seguir.

No capítulo 1 é realizada uma apresentação sucinta de conceitos básicos de sistemas fuzzy e suas aplicações, visando esclarecer esse tópico fundamental.

O capítulo 2 apresenta as diversas técnicas de implementações de sistemas fuzzy em plataformas via software e via hardware, abordando suas vantagens e desvantagens.

O capítulo 3 apresenta os cartões de memória do tipo SD, usados como mini HD para a plataforma, e a interface USB, fornecendo, assim, maiores subsídios para compreensão do hardware desenvolvido.

O capítulo 4 trata do funcionamento detalhado da plataforma eletrônica proposta e da construção do protótipo.

O capítulo 5 aborda e descreve o software de programação desenvolvido no ambiente PC que viabiliza a configuração da plataforma de hardware para que possa executar o sistema fuzzy previamente projetado no ambiente simulado.

O capítulo 6 ilustra a integração de um sistema fuzzy gerado no MATLAB® e a transferência do arquivo para a plataforma eletrônica através do software de transferência, estudando casos de aplicação e o procedimento da análise de sensibilidade.

A conclusão deste trabalho é apresentada no capítulo 7 juntamente com uma relação de possíveis trabalhos futuros.

CAPÍTULO 1

FUNDAMENTOS DE SISTEMAS FUZZY

A teoria dos conjuntos fuzzy foi introduzida, em 1965, por Lotfi A. Zadeh (Universidade da Califórnia, Berkeley). Em meados da década de 60, Zadeh observou que os recursos tecnológicos disponíveis eram incapazes de automatizar as atividades relacionadas a problemas de natureza industrial, biológica ou química, que compreendessem situações ambíguas, não passíveis de processamento através da lógica computacional fundamentada na lógica booleana. Procurando solucionar esses problemas, e baseado em estudos da lógica multivalor, proposta por Michalewicz em 1934, foi publicado em 1965, o primeiro artigo resumindo os conceitos dos conjuntos fuzzy e revolucionando o assunto com a criação da lógica fuzzy (ZADEH, 1965).

Em 1975, o Prof. Mamdani, do Queen Mary College, Universidade de Londres, após inúmeras tentativas frustradas em controlar uma máquina a vapor com tipos distintos de controladores, incluindo o controle proporcional integral e derivativo, mais conhecido como controlador PID, somente conseguiu fazê-lo através da aplicação do raciocínio fuzzy. Esse sucesso serviu de alavanca para muitas outras aplicações. A partir daí vieram várias outras aplicações, destacando-se, por exemplo, os controladores fuzzy de plantas industriais, refinarias, processos biológicos e químicos, trocadores de calor, máquinas diesel, tratamento de água e sistema de operação automática de trens.

Os controladores fuzzy são capazes de tomar decisões a partir de informações imprecisas e representam uma técnica que incorpora a forma humana de pensar em um sistema de controle. Um controlador fuzzy típico pode ser projetado para comportar-se conforme o raciocínio dedutivo: o processo que as pessoas usam para inferir conclusões baseadas em informações (SHAW e SIMÕES, 1999).

1.1 Aplicações de controle fuzzy

Diversas áreas estão sendo beneficiadas pela tecnologia decorrente da lógica fuzzy. Dentre essas áreas podem ser citadas algumas que tiveram relevância no avanço tecnológico e que merecem destaque. O controle de processos industriais foi a área pioneira. Na mesma

época que se fez o controle da máquina a vapor, conforme citado, vale ainda ressaltar outra aplicação industrial significativa que foi desenvolvida pela indústria de cimento *F. L. Smidth Corp.* da Dinamarca. Hoje em dia, uma grande variedade de aplicações comerciais e industriais está disponível, destacando-se neste cenário o Japão, os EUA e a Alemanha.

A seguir são apresentados outros exemplos onde os controladores fuzzy têm sido aplicados (SHAW e SIMÕES, 1999):

- em câmeras de vídeo, são aplicados ao foco automático e ao controle da íris da câmera; o primeiro pode manter em foco um objeto em movimento enquanto o segundo analisa as condições de iluminação para ajustar a velocidade automaticamente;
- em máquinas de lavar, com a utilização de sensores adequados (temperatura de água, concentração de detergente, peso das roupas, nível de água, tipo de tecido, tipo de sujeira, grau de sujeira) controlam os ciclos da máquina: bater, enxaguar e centrifugar. Há aproximadamente 270 tipos de ciclos de lavagem em uma lavadora “fuzzy”;
- em fornos de microondas, as informações obtidas pelos sensores (infravermelho, umidade, pressão atmosférica) permitem que se ajuste a intensidade e duração do cozimento para cada tipo de comida;
- incineração de lixo com o fim de manter a temperatura de queima constante, desta forma a geração de gases tóxicos é minimizado e se evita a corrosão da câmara de combustão;
- em aparelhos de ar-condicionado, a fim de produzir sinais de referência para as válvulas de água fria e quente, e também para o controle de umidade. A estratégia de controle usa diversos sensores diferentes para determinar a temperatura e a umidade, conseguindo um melhor aproveitamento de energia;
- controle de temperatura: selagem de embalagem, corrigindo a temperatura de fusão;
- equipamento para testes com carbono com objetivo de determinar a idade dos materiais, em que o controle fuzzy é capaz de corrigir rapidamente a temperatura do liquido refrigerante usado neste tipo de equipamento.

1.2 Sistemas de controle baseados em Lógica Fuzzy.

Lógica fuzzy é uma técnica de inteligência computacional que procura maneiras de máquinas emularem o raciocínio humano na solução de problemas diversos. Esta abordagem procura mimetizar a forma humana de atuar (BAUCHSPIESS, 2002). É um método capaz de

expressar de uma maneira sistemática quantidades imprecisas, vagas, mal definidas, por esta razão, ela é traduzida em português como: nebulosa ou difusa.

Segundo Lee (FABRO, 2003), os sistemas de controle fuzzy possuem uma série de vantagens quando comparados a outros sistemas de controle:

- simplificação do modelo que representa o processo;
- melhor tratamento das imprecisões inerentes aos sensores utilizados;
- facilidade na especificação das regras de controle, em linguagem próxima da natural;
- satisfação de múltiplos objetivos de controle;
- facilidade de incorporação do conhecimento de especialistas humanos.

Para se entender a concepção de um sistema de controle fuzzy, parte-se do esquema geral que descreve o modelo de um controlador e de uma planta ou processo que está sendo controlado, como ilustrado na Figura 4.

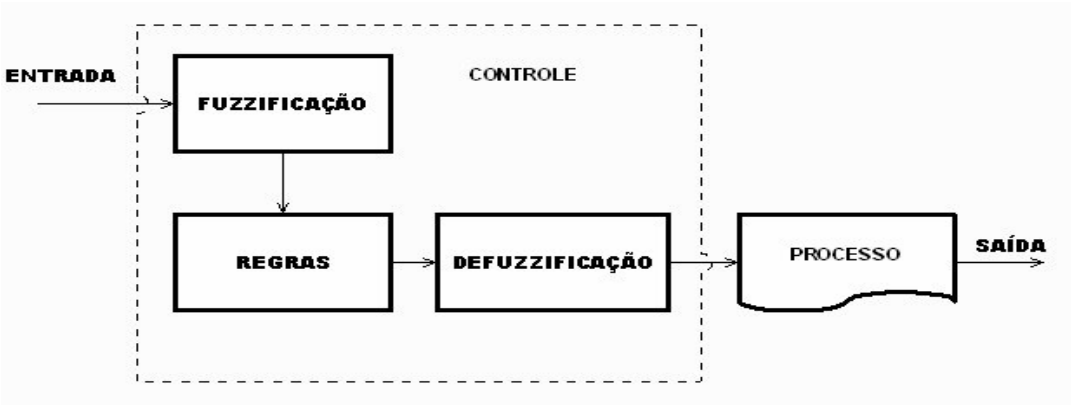


Figura 4 –Sistema Fuzzy

Para a confecção de um controlador fuzzy, pode-se imaginar que ao invés de usar apenas parâmetros de modelos matemáticos para construir o controlador, pode se visualizar este controle onde o operador humano, um especialista, tivesse a responsabilidade de controlar os parâmetros da planta.

Desta forma, o controlador fuzzy é desenvolvido para automatizar, como um especialista, o gerenciamento do processo. Com isto, o primeiro passo na construção de um sistema de controle fuzzy consiste na aquisição do conhecimento sobre o processo que se quer controlar. Como em qualquer processo de modelagem (LJUNG, 1999), deve-se inicialmente determinar qual ou quais são as variáveis de entrada e saída do processo.

As variáveis de entrada são aquelas nas quais o operador da planta baseia-se para fazer uma análise de desempenho do processo e para tomar decisões sobre os próximos passos a seguir e, em geral, a sua escolha é feita de maneira intuitiva por este especialista. Os dados dessas variáveis, em sistemas complexos, podem ser aproximados, isto porque existe uma aproximação inerente a o modelo fuzzy e à sua implementação.

As variáveis de saída são as variáveis controladas do processo. Estas são de mais fácil identificação já que na maioria dos casos elas estão relacionadas aos objetivos do controle e são as mesmas utilizadas nos controladores convencionais.

Após a definição de todas as entradas e saídas para o controlador fuzzy, deve-se especificar a base de conhecimento que formará o núcleo do sistema de controle fuzzy. As informações, neste caso, não precisam ser precisas, porém devem estar dentro do contexto dos objetivos a serem alcançados no processo para assegurar um bom desempenho do sistema de controle.

Em conclusão, os objetivos da planta a ser controlada devem ser bem compreendidos, podendo ter uma incerteza dos dados e certa ambigüidade em algumas situações do processo, sendo que a escolha de entradas e saídas é parte de fundamental importância para o desenvolvimento do controlador..

O diagrama em bloco de um sistema de controle fuzzy é mostrado na Figura 5. O controlador fuzzy é composto de quatro blocos, como descrito na seqüência (PASSINO e YURKOVICH, 1997):

1. Uma base de conhecimento: formada por uma base de dados e uma base de regras, contendo o conhecimento dos objetivos do processo a ser controlado;

2. Um mecanismo de inferência: que emula a decisão de um especialista, fazendo uma interpretação e aplicação do conhecimento sobre a melhor maneira de controlar a planta. Este mecanismo aplica a base de regras à base de dados corrente, gerando a resposta do controlador;

3. Interface de fuzzificação*:* que converte as entradas crisp do controlador em valores fuzzy de variáveis, de forma que o mecanismo de inferência possa facilmente identificar e aplicar as regras ativas em cada situação;

4. Interface de defuzzificação: que converte as conclusões do mecanismo de inferência, que são valores fuzzy de variáveis de saída, em valores crisp para a entrada atual do processo.

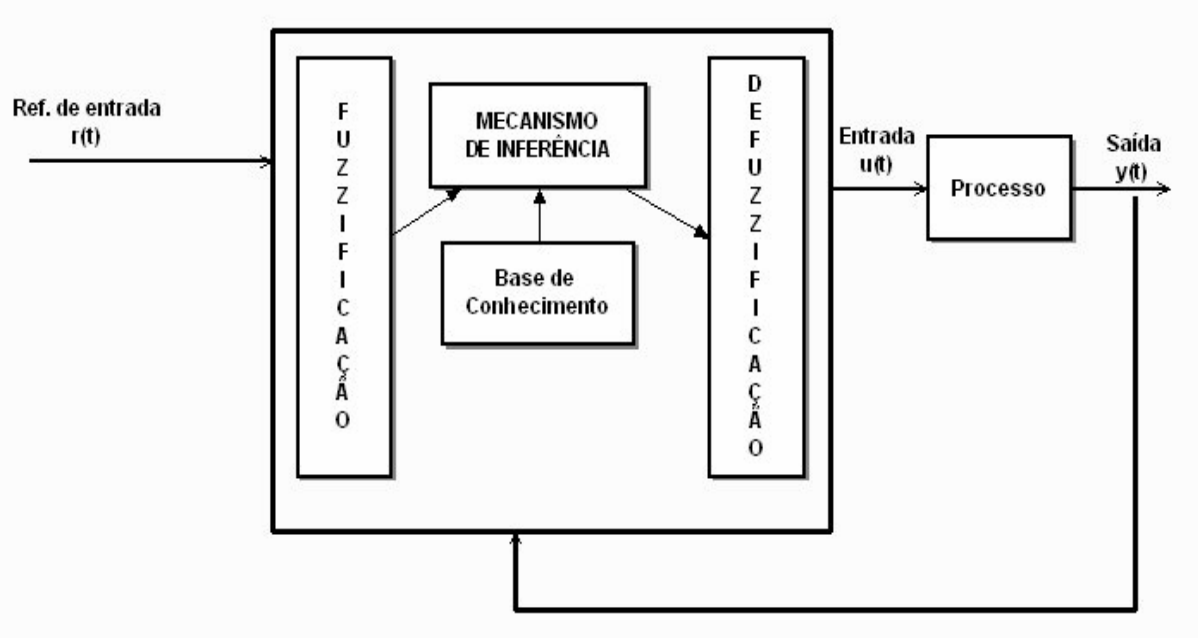


Figura 5 – Bloco Funcional Fuzzy

Os passos a seguir descrevem em detalhes cada um desses blocos e de certa maneira constituem um roteiro para se projetar um controlador fuzzy baseado em regras descritas por um especialista, apropriando-se do conhecimento humano, em como controlar o processo, a fim de usar a lógica fuzzy para automatizá-lo.

1.2.1 Fuzzificação e Funções de Pertinência

A fuzzificação é o processo que torna qualquer quantidade numérica – também chamada crisp na literatura – em quantidade fuzzy. É, portanto, uma função que garante certo grau de imprecisão a um valor numérico, mapeando o valor físico de uma variável de um processo em um universo normalizado de discurso (DRIANKOV; HELLENDOORN e REINFRANK, 1996). Isto é necessário para que a entrada do processo se torne compatível com a representação fuzzy adotada na base de regras.

Matematicamente, a fuzzificação pode ser descrita como um mapeamento:

$$\mu_A(x): U \rightarrow [0, 1] \qquad (1)$$

e representado por um conjunto de pares ordenados

$$A = \{\mu_A(x)/x\}, \; x \in U. \qquad (2)$$

Desta forma, a *função de pertinência* μ_A associa a cada elemento 'x' pertencente a *U* um número real $\mu_A(x)$ no intervalo [0, 1], que representa o grau com o qual o elemento 'x' pertence ao conjunto A, ou seja, o *grau de pertinência* de 'x' em A. O conjunto suporte de um conjunto fuzzy A é o subconjunto de pontos x de *U* tal que $\mu_A(x) > 0$. Um conjunto fuzzy cujo conjunto suporte é um único ponto de *U* com $\mu_A(x) = 1$ é chamado de conjunto unitário fuzzy ou *singleton*.

Esta função de transformação é denominada função de pertinência e é construída a partir da teoria de conjuntos fuzzy. Zadeh definiu os conjuntos fuzzy como uma classe com graus contínuos de pertinência, ou seja, dado um elemento **a**, do espaço **A**, pertencente a um conjunto fuzzy **X** – percebe-se aqui a relação com a teoria clássica dos conjuntos – existe uma função característica, ou de pertinência, dada por $\mu_X(a)$, que associa a esta pertinência um valor real no intervalo [0,1], conforme exemplo mostrado na Figura 6. Este exemplo ilustra a representação do conceito "pessoa alta" em um sistema clássico (crisp) e em um sistema fuzzy, considerando como alta uma pessoa com altura maior que 1,80 m.

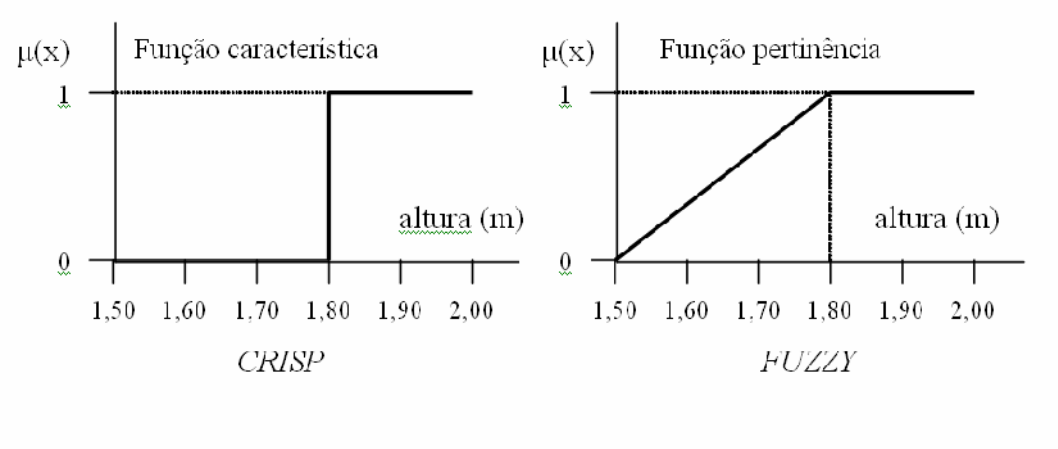


Figura 6 – Diferença entre Crisp e Fuzzy

Um conjunto fuzzy é um agrupamento impreciso e indefinido, onde a transição de não-pertinência para pertinência é gradual. A incerteza de um elemento, isto é, seu grau fracionário de pertinência, pode ser concebido como uma medida de possibilidade, ou seja, a possibilidade de que um elemento seja membro do conjunto. O conceito de possibilidade não é o mesmo que o de probabilidade. A probabilidade expressa a chance de que um elemento seja membro de um conjunto, sendo também expressa no intervalo numérico [0,1] (SHAW e SIMÕES, 1999).

Normalmente associa-se um rótulo lingüístico a um conjunto fuzzy. Uma função de

pertinência (FP) possível para o conjunto fuzzy *meia idade* é mostrada na Figura 7. Observe-se que, neste caso, a função de pertinência para idade em torno dos 45 anos tem valor próximo de 1, indicando que este valor pertence, definitivamente, ao conjunto *meia idade*, enquanto que para os valores próximos a 20 ou a 65 anos esta indicação não se verifica. Neste caso, uma afirmativa questionável, de que as idades próximas dos 36 ou dos 53 anos venham a pertencer a este conjunto, é traduzida por um valor da função de pertinência em torno de 0,5.

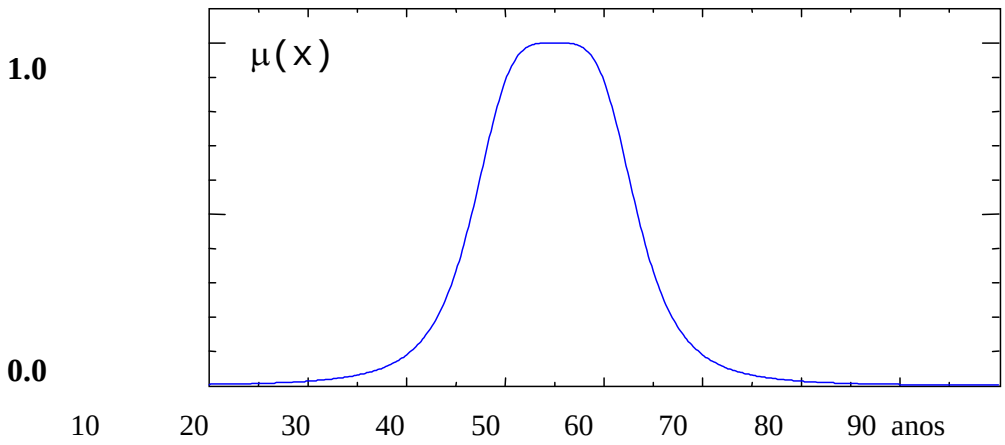


Figura 7 – Função de Pertinência para o conjunto fuzzy “meia idade”.

Em um conjunto *crisp*, uma lista objetiva das pessoas de meia idade é suficiente para sua definição. Para conjuntos fuzzy, entretanto, a definição só estará completa com o conhecimento do grau de pertinência de cada elemento. O grau de pertinência indica o quanto determinado elemento pertence ao conjunto fuzzy. Outro exemplo: o conjunto dos números naturais aproximadamente iguais a 20 poderia ser representado pelo conjunto fuzzy A, descrito como:

$$A = \{\mu_A(x)/x\} = \{0.15/12, 0.28/15, 0.6/18, 0.8/19, 1/20, 0.8/21, 0.6/24, 0.28/25, 0.15/28\}$$

Na lógica fuzzy existe o conceito de *variável lingüística* que é aquela cujos valores aparecem em sentenças na forma de linguagem "natural". Temperatura, velocidade, distância, altura, etc., são exemplos de variáveis lingüísticas. Em aplicações em engenharia, freqüentemente associamos termos lingüísticos a uma variável. Numa aplicação em controle, por exemplo, podemos nos referir a um certo erro de posicionamento como

“grande”, “médio”, “pequeno” ou “muito pequeno”. A questão semântica é muito importante e uma associação feita de forma adequada leva-nos a um melhor entendimento do problema que está sendo analisado. Para facilitar a compreensão, tome-se como exemplo a variável lingüística “Temperatura Ambiente”, cujos valores podem ser os termos lingüísticos “frio”, “quente”, “muito quente”, “nem quente nem frio”, “confortável”, com as respectivas funções de pertinência mostradas na Figura 8.

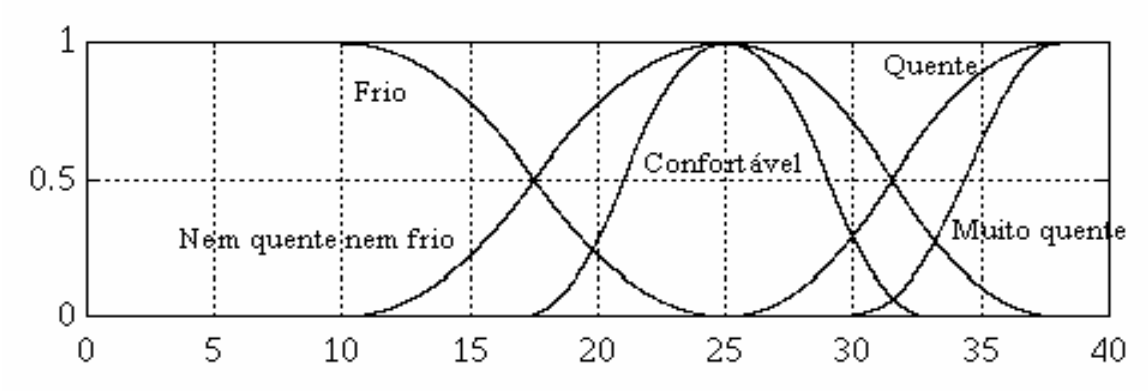


Figura 8 – Variável lingüística do tipo “Temperatura Ambiente”.

É possível realizar operações com conjuntos fuzzy. Tipicamente, o conjunto resultante da união de dois conjuntos fuzzy é aquele cuja função de pertinência é composta pelos *máximos* das funções de pertinência de A e de B para todo ‘x’ pertencente a U . A interseção mais comum, por outro lado, é o *mínimo*. Portanto, tem-se:

União (típica) $\mu_A(x) \cup \mu_B(x) = \text{Máx}\,[\mu_A(x), \mu_B(x)]$

Interseção (típica) $\mu_A(x) \cap \mu_B(x) = \text{Min}\,[\mu_A(x), \mu_B(x)]$

Cabe ressaltar que nem sempre esses dois operadores (máx e mín) são empregados. Estudos teóricos e experimentais sugerem que, em determinadas aplicações, outros operadores podem apresentar melhores resultados (GUPTA & QI, 1991). Com o intuito de obter uma maior flexibilidade na escolha de operadores que sejam mais adequados à modelagem de um determinado problema real, é conveniente tratar de operadores coletivamente, através de conceitos que os generalizem.

Uma ampla classe de modelos para operadores de interseção e união de conjuntos fuzzy é formada pelas normas triangulares e conormas triangulares (t- normas e t-conormas, respectivamente). Seguem-se as respectivas definições (PEDRYCZ, 1989).

Uma t-norma é uma função binária *t*: [0,1] x [0,1] → [0,1], que satisfaz às seguintes condições, ∀ x, y, z, w ∈ [0,1]:

Monotonia: $t(x, w) \leq t(y, z)$, para $x \leq y$ e $w \leq z$

Comutatividade: $t(x, y) = t(y, x)$

Associatividade: $t(t(x, y), z) = t(x, t(y, x))$

Condições limites: $t(x, 0) = 0$ e $t(x, 1) = x$

Uma s-norma é uma função binária s: [0,1] x [0,1] → [0,1], que satisfaz as propriedades de monotonia, associatividade e comutatividade como a t-norma e apresenta, nas condições limites, o seguinte comportamento:

Condições limites: $s(x, 0) = x$ e $s(x, 1) = 1$

As t-normas mais empregadas para definir a interseção de conjuntos fuzzy são:

produto: $\mu_{A \bullet B}(x) = \mu_A(x) \cdot \mu_B(x)$

interseção bold: $\mu_{A \cap B}(x) = min\,(0, \mu_A(x) + \mu_B(x) - 1)$

As s-normas mais empregadas para definir a união de conjuntos fuzzy são, além do *max*,

soma probabilística $\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$

soma truncada $\mu_{A \cup B}(x) = min\,(1, \mu_A(x) + \mu_B(x))$

A Figura 9 exemplifica o uso dos operadores *máx* e *min* aplicados a dois conjuntos fuzzy A e B.

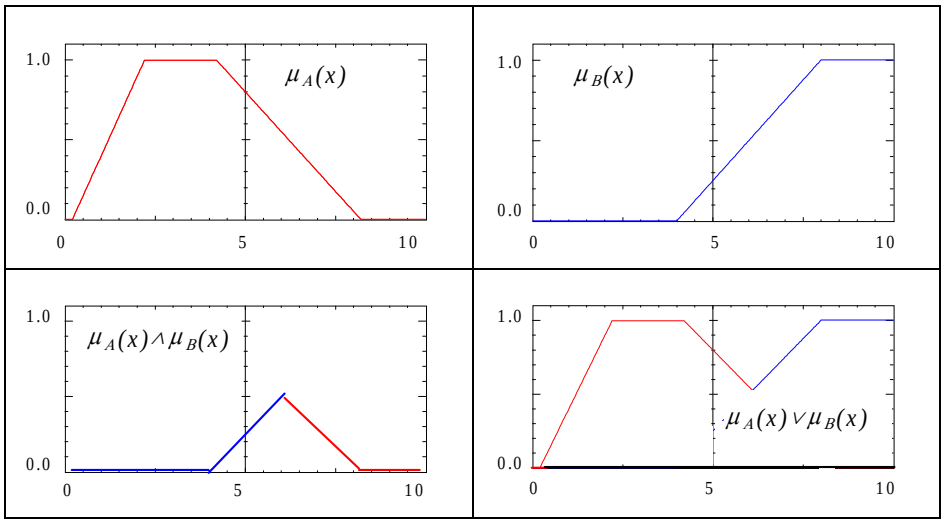


Figura 9 – Exemplo dos operadores de união (*máx*) e interseção (*min*) (SOUZA, 1999).

Com as informações apresentadas sobre conjuntos e lógica fuzzy pode-se conceituar

um sistema de inferência fuzzy, ou simplesmente, sistema fuzzy. Um sistema fuzzy é um sistema que usa uma coleção de funções de pertinência e regras do tipo se-então para inferir algo a partir dos dados. As regras têm geralmente o seguinte formato:

SE *x* é *baixo* E *y* é *alto* ENTÃO *z* é *médio*

Onde *x* e *y* são variáveis de entrada (nomes para valores conhecidos de dados), *z* é uma variável de saída (um nome para um valor a ser computado), "*baixo*", "*alto*" e "*médio*" são valores lingüísticos definidos por uma função de pertinência (conjunto fuzzy) em *x*, *y* e *z*, respectivamente.

O antecedente (premissa da regra) estima com que grau a regra se aplica, enquanto o conseqüente (conclusão da regra) associa uma função de pertinência para uma dada variável de saída. Um conjunto de regras num sistema fuzzy é conhecido como a *Base de Regras*.

Um sistema fuzzy de inferência básico (TANSCHEIT, 2000) é ilustrado no digrama em blocos da Figura 10.

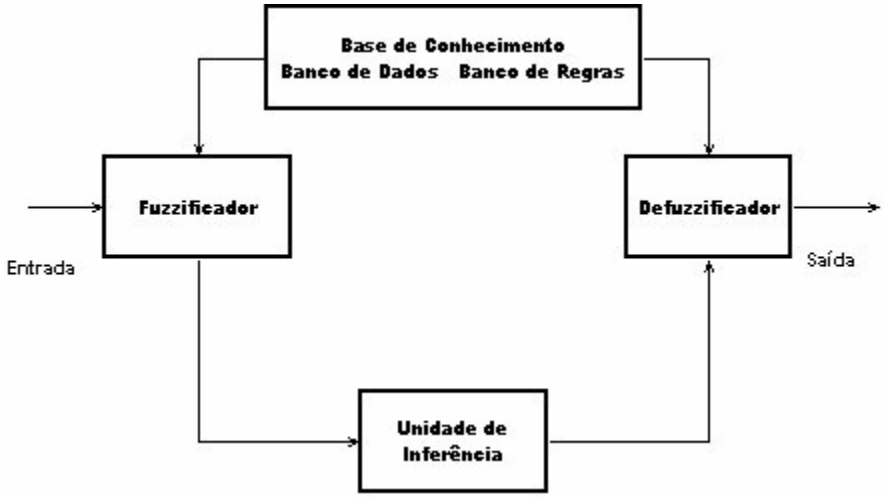


Figura 10 – Diagrama em blocos de um sistema de inferência fuzzy.

Fuzzificador

Neste bloco são recebidos os valores reais de cada entrada. As funções de pertinência definidas para cada variável de entrada recebem os valores reais das entradas para determinar o grau de pertinência de cada premissa da regra.

Vários perfis de funções de pertinência (FPs) são encontrados nas implementações de sistemas fuzzy (SOUZA, 1999). Os tipos mais comuns são: triangular, trapezoidal, gaussiano, sino e *singleton*.

FP Triangular

Esse perfil, apresentado na Figura 11, tem a vantagem de ser simples. É descrito por três variáveis: SL, C e SR (Spread Left, Center e Spread Right) e é definido pelas expressões:

$$\mu(x) = \frac{x - SL}{C - SL} \quad \text{, para } SL \le x \le C$$

$$\mu(x) = \frac{x - SR}{C - SR} \quad \text{, para } C \le x \le SR$$

$$\mu(x) = 0 \quad \text{, caso contrário}$$

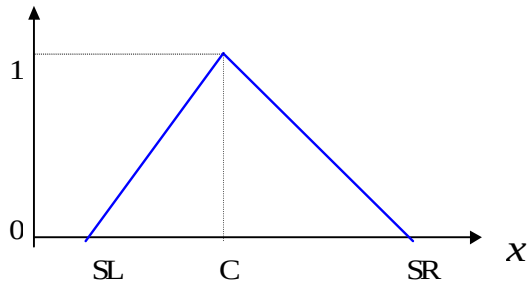


Figura 11 – Função de pertinência de formato triangular.

FP Trapezoidal

Este perfil também é simples. Pode ser descrito por quatro parâmetros. A Figura 12 ilustra um exemplo de tal FP.

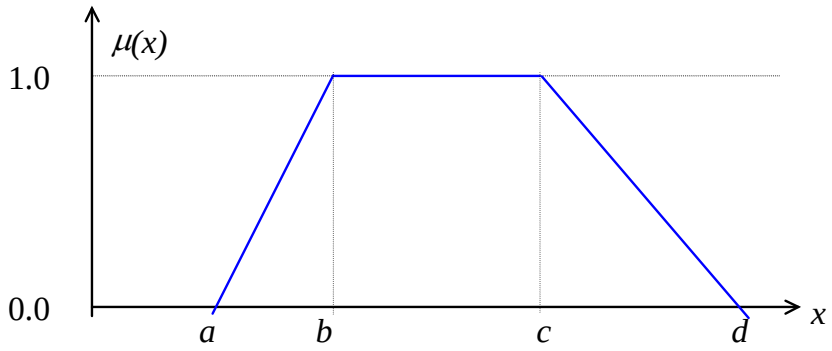


Figura 12 – Função de pertinência com formato trapezoidal.

FP Gaussiano

Este perfil é descrito pela expressão abaixo, onde m é a média e ν é o desvio padrão:

$$\mu(x) = e^{-\left(\frac{x-m}{\nu}\right)^2} \qquad (3)$$

Este formato de FP é empregado em aplicações de agrupamentos que utilizam medidas de similaridade (por ex. distância euclidiana). A Figura 13 ilustra este formato.

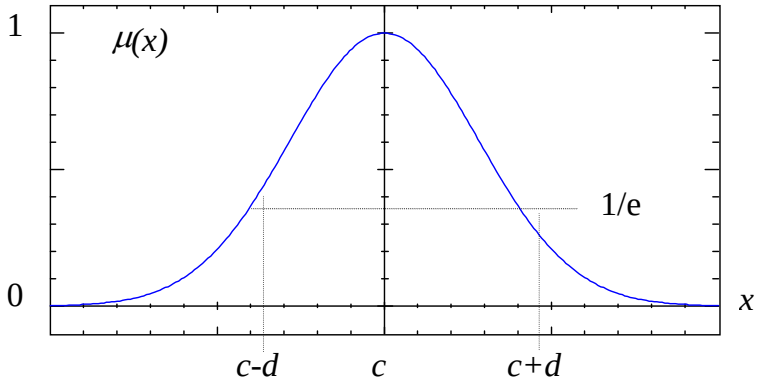


Figura 13 – Função de pertinência gaussiana com $m = c$ e v = d.

FP Sino

Este perfil de função de pertinência é definido pela expressão abaixo:

$$\mu(x) = \frac{1}{1+\left(\frac{x-c}{a}\right)^{2b}} \qquad (4)$$

Onde a variável c define o centro da FP, a define a largura e b o decaimento da FP. Conforme mostrado na Figura 14, seu perfil é similar ao formato anterior. Entretanto, o esforço computacional para o seu cálculo é menor por não envolver exponenciais.

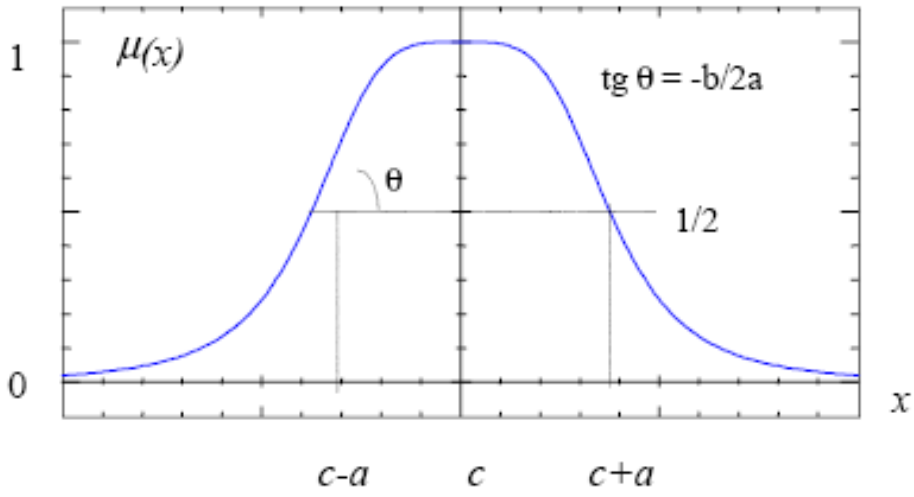


Figura 14 – Função de pertinência com formato sino

As FPs descritas anteriormente podem ser utilizadas tanto nos termos antecedentes quanto nos termos conseqüentes das regras. Entretanto, um tipo de FP muito freqüente entre os conseqüentes dos sistemas fuzzy é o chamado “Fuzzy Singleton”. Este formato compreende uma FP que apresenta o grau de pertinência igual a 1 apenas em um ponto de seu

domínio, e o grau 0 (zero) nos demais pontos. Sua principal vantagem é simplificar o processo de defuzzificação do sistema fuzzy. A Figura 15 ilustra um exemplo de FP "Fuzzy Singleton" ou simplesmente "Singleton". Nela, o grau de pertinência da FP assume o valor 1 apenas para $x = a$.

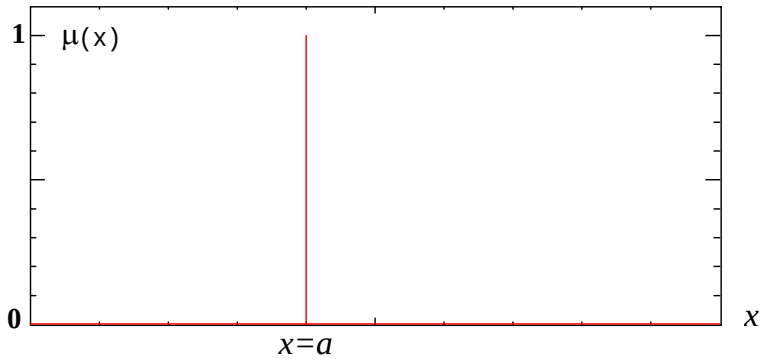


Figura 15 – Função de pertinência fuzzy Singleton.

Unidade de Inferência

Nesta unidade os valores das premissas de cada regra são computados e aplicados para gerar a conclusão de cada regra. Em seqüência um processo de composição de regras é empregado para gerar a entrada do defuzzificador.

As regras fuzzy formam a parte fundamental da estrutura de conhecimento em um sistema fuzzy de inferência. Existem dois formatos principais de regras fuzzy, cada um associado a um modelo de inferência. Os modelos são chamados pelos nomes de seus criadores, a saber, Mamdani e Takagi-Sugeno-Kang (TSK). Basicamente, a diferença entre eles recai no tipo de conseqüente e no procedimento de defuzzificação. A seguir são descritos os dois formatos. Por simplicidade, somente modelos de regras com duas entradas e uma saída são exemplificados.

Modelo de Inferência Mamdani

Regra : *Se* *x* *é* *A* *e* *y* *é* *B* *então* *z* *é* *C*

Este modelo foi proposto, inicialmente, como uma tentativa para controle de um conjunto turbina a vapor/boiler usando regras derivadas de um especialista humano (MAMDANI & ASSILIAN, 1975). A Figura 16 ilustra como a saída é derivada em um sistema de inferência do tipo de Mamdani.

A saída *z* é obtida pela defuzzificação do conjunto fuzzy de saída resultante da aplicação da operação de t-conorma (*máx*) sobre os conjuntos dos conseqüentes, que, por sua vez, foram modificados via t-norma, pelo grau de disparo fornecido pelos antecedentes.

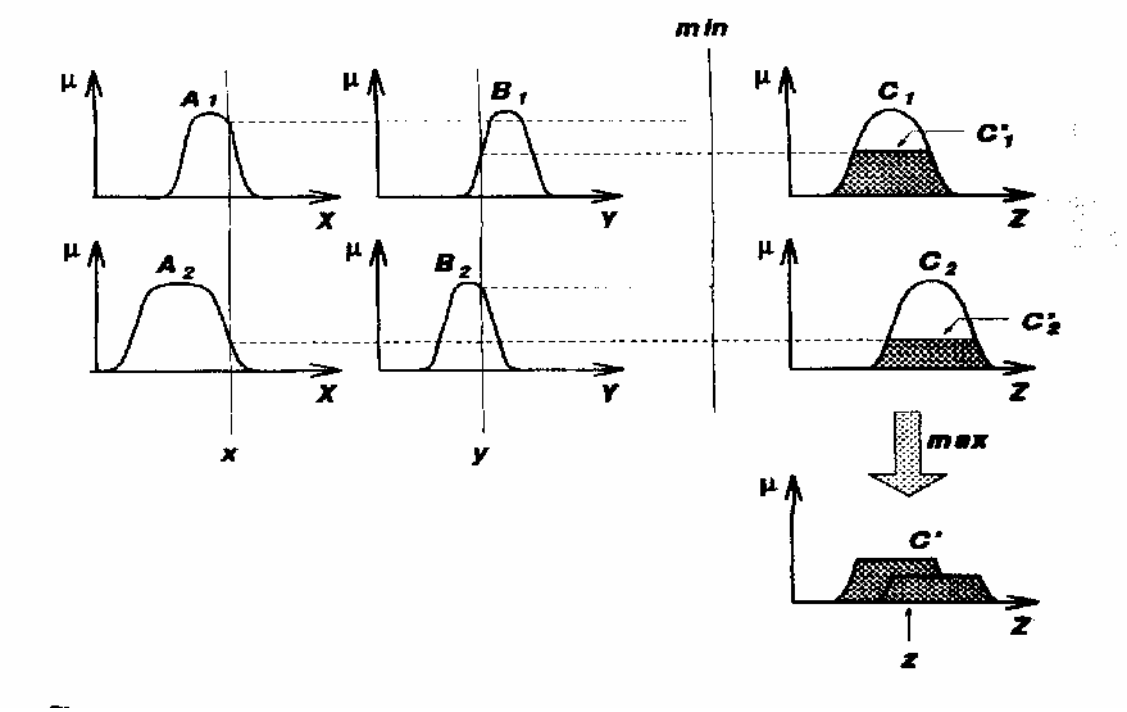


Figura 16 – Modelo de Mamdani com composição max/min (JANG & SUN, 1995).

Modelo de Inferência TSK

Regra : *Se x é A e y é B então z = f(x,y)*

Neste caso, a saída de cada regra é uma função das variáveis de entrada. Geralmente, a função que faz o mapeamento de entrada e saída, para cada regra, é uma combinação linear das entradas, isto é $z = px1 + qx2 + r$. No caso em que $p = q = 0$, temos $z = r$ (*fuzzy singleton*). A saída do sistema é obtida pela média ponderada (procedimento de defuzzificação) das saídas de cada regra, usando-se o grau de disparo destas como pesos da ponderação. A Figura 17 ilustra este modelo.

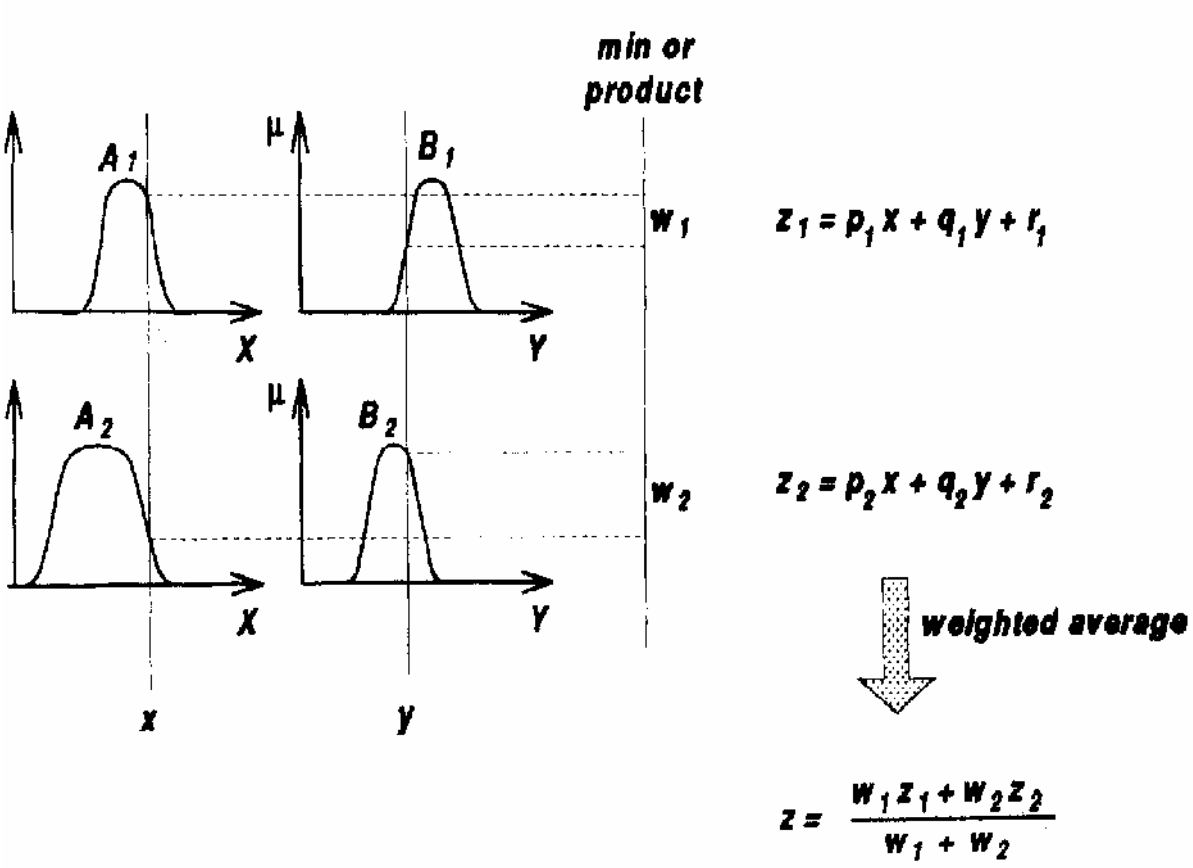


Figura 17 – Modelo de TSK (JANG & SUN, 1995).

Base de Conhecimento

Consiste numa Base de Dados (onde são definidos os conjuntos fuzzy, funções de pertinência) e numa Base de Regras (onde estão armazenadas as regras fuzzy) que realizarão a inferência do sistema.

1.2.2 Defuzzificador

Uma vez concluída a inferência deve-se determinar o valor real da saída do sistema. Este processo é chamado de defuzzificação. Dentre os vários métodos existentes os mais usados são: centróide (centro de gravidade), média dos máximos e média ponderada.

Defuzzificador Centro de Gravidade

É um dos métodos mais utilizados. Supondo-se um universo de discurso discreto, a saída *Z* é produzida pelo cálculo do centro de gravidade do conjunto fuzzy conseqüente μ_c obtido pela composição das regras. A expressão da saída *Z* é dada por:

$$Z = \frac{\sum_{i=0}^{m} \mu_c(zi) * Zi}{\sum_{i=0}^{m} \mu_c(zi)} \qquad (5)$$

Onde *m* é o número de intervalos de quantização da saída, *Zi* é o valor da variável de saída para o intervalo de quantização *i* e $\mu_c(zi)$ seu grau de pertinência

Defuzzificador Média dos Máximos

Neste tipo de defuzzificador, o conjunto de saída é examinado e se determina os valores para os quais ele é máximo. Computa-se então a média destes valores como valor de saída, conforme ilustrado na Figura 18.

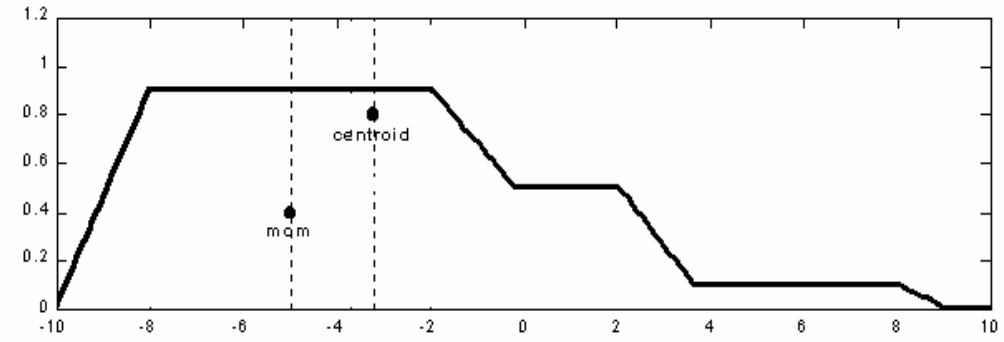


Figura 18 – Métodos de defuzzificação: Centróide e Média dos Máximos.

Defuzzificador Média Ponderada

Quando apenas singletons são usados como conseqüentes das regras fuzzy, este é o método naturalmente indicado, pois combina os conseqüentes com o nível de disparo de cada uma das regras gerando uma saída *Z* de acordo com a expressão:

$$Z = \frac{\sum_{i=1}^{n} \mu_i * Zi}{\sum_{i=1}^{n} \mu_i} \qquad (6)$$

Onde *n* é o número de regras fuzzy, μ_i é o nível de disparo da regra *i*, e *Zi* é o valor do singleton *i*. Este método também é utilizado na defuzzificação dos modelos TSK de primeira ordem (saída é combinação linear das entradas).

CAPÍTULO 2
IMPLEMENTAÇÕES DE SISTEMAS FUZZY

Sistemas baseados em lógica fuzzy têm demonstrado sua capacidade de resolver diversos tipos de problemas em várias aplicações de engenharia.

Um sistema fuzzy pode ter diversas formas de implementação. Foram descritos alguns modelos consagrados que possuem suas vantagens e desvantagens, limites e ponderações pertinentes a forma e ao resultado obtido com as implementações. Conforme a aplicação, pode-se ter um projeto de execução simulada da inferência fuzzy entre as entradas e a respectiva saída para a avaliação de resultados específicos, ou um projeto via hardware especifico, mutável ou não, quando se propõe uso de módulos físicos de emulação fuzzy que tenham a facilidade de serem alterados (vide Figura 19).

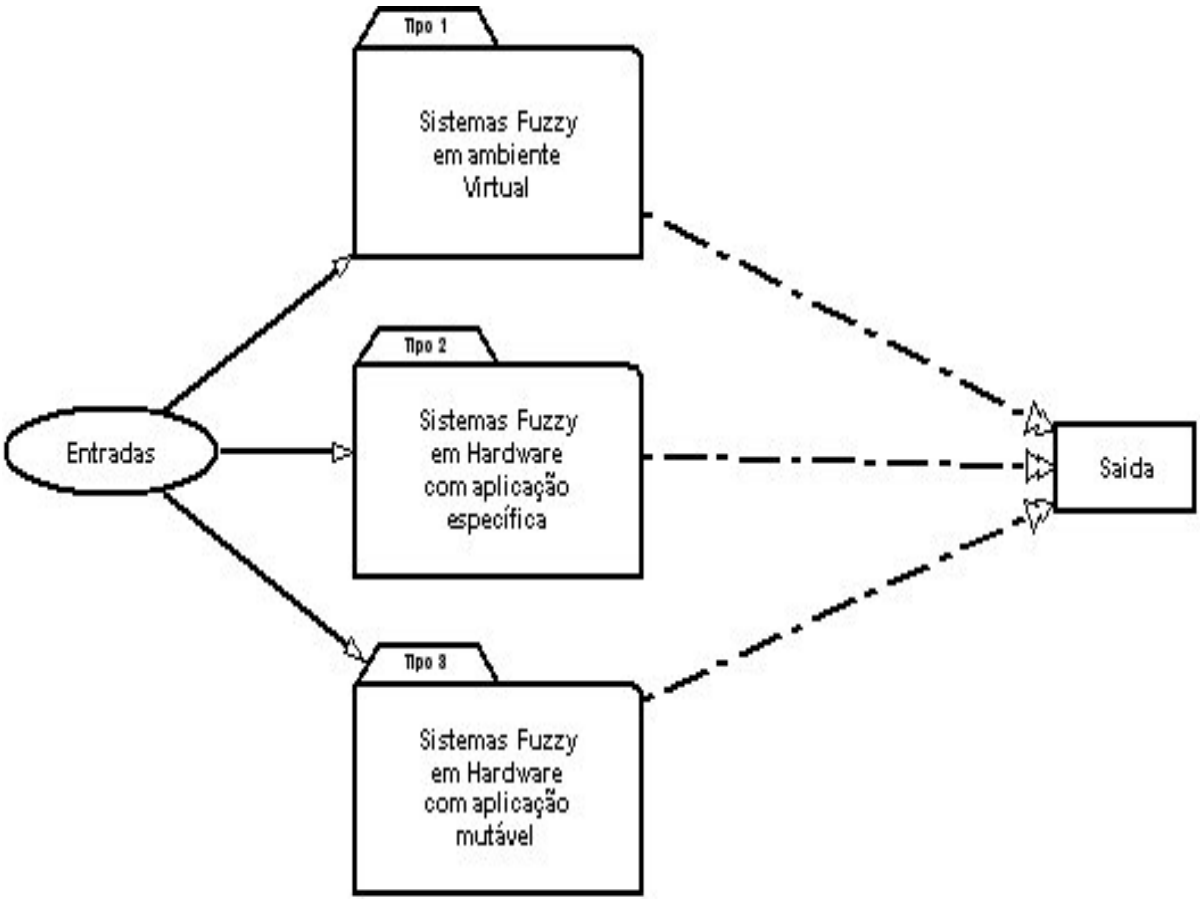


Figura 19 – Tipos de Implementações fuzzy

2.1 - Implementação via Software em Microcomputadores

Este tipo de implementação é focado diretamente no contexto de algum programa que faça toda a inferência fuzzy virtualmente, utilizando simulação, adequando as regras, fuzzificador, defuzzificador e tipos de inferência. Esta é uma etapa importante que antecede qualquer aplicação real via hardware dedicado ou integrado. Seria uma primeira etapa de qualquer projeto, pois pode sofrer alterações sem envolver meios físicos de hardware.

Um software muito utilizado para tal fim é o toolbox fuzzy do Matlab, permitindo a inclusão de diversos modelos de inferências, bem como grande facilidade no manuseio de regras e representações gráficas da saída, além de possuir uma flexibilidade no que tange ao manuseio dos arquivos que servirão como sinais de entrada do sistema. Esta etapa de análise do sistema fuzzy apresentado via software se faz necessário, pois em um sistema no qual se deseja alcançar uma excelência em resultados práticos irão suceder revisões sobre regras, padrões e métodos utilizados, para se almejar esta implementação via hardware.

Chegando a uma configuração de sistema via software que atenda as perspectivas desejadas em um meio “virtual”, não sujeito as influências reais de um sistema de aplicação via hardware, passa-se a segunda etapa que é a adequação deste sistema a uma plataforma hardware para executar o sistema.

2.2 – Implementação com Circuitos Analógicos

Este tipo de implementação é realizado com componentes eletrônicos analógicos, geralmente composto por diodos, amplificadores operacionais, resistores e transistores, formando um sistema eletrônico analógico projetado especificamente para o projeto.

Para obtermos a saída com inferência em uma das funções de pertinência (triangular, rampa), deve-se parametrizar os valores dos resistores para tal fim, lembrando que a saída Vo tem seu valor entre [0,1]. Os circuitos que seguem esta implementação nas regras e defuzzificação são portas OU (operação MAX) e portas AND (operação MIN), implementados como circuitos somadores ou comparadores.

Devido a ser um circuito analógico de aplicação direta, a grande vantagem é a velocidade de resposta Vo no que tange à apresentação dos sinais na entrada, podendo fazer parte direta de projetos como um todo ou parte em um sistema fuzzy (SAMS et al., 1994). As desvantagens deste sistema estão relacionadas com a composição eletrônica para sua implementação, pois nem sempre teremos os valores dos resistores para implementar o circuito, tendo que fazer composições em série ou em paralelo dos mesmos. A outra

desvantagem está no fato de alterar uma ou mais regras pelo especialista. Caso haja necessidade de alterar uma regra ou função de pertinência, a eletrônica da plataforma tem que mudar, com novos cálculos e troca significativa nos valores dos componentes. Neste exposto, ficaria trabalhoso ter uma plataforma de pesquisa que precisasse ser alterada as regras ou funções de pertinência, a não ser que se tivesse “blocos construtores” (AMARAL, 2003), definição esta dada ao conjunto de circuitos previamente modelados, para serem usados em sistemas com aplicações fuzzy diferentes, podendo ser selecionados conforme a necessidade.

Yamakawa (1993, pp.496-52) propõe um sistema de inferência fuzzy analógico com circuitos diferentes dos tradicionais circuitos analógicos, tais como amplificadores operacionais e seus derivados. Os modelos de circuitos empregados em máquinas de inferência são não-lineares com os seguintes características por ele definidas:

• considerando que a função de pertinência manipule valores entre 0 e 1 e uma resolução de 10%, o projetista não precisa se preocupar com a precisão, envolvendo a linearidade, baixo desvio térmico, baixo offset, etc.

• essencialmente um circuito fuzzy não necessita de ganho maior do que a unidade.

• um circuito fuzzy deve ser projetado em uma arquitetura simples (com poucos dispositivos) para obter alta velocidade de processamento. As características de entrada e saída de circuitos intrinsecamente fuzzy são não-lineares.

Uma implementação analógica simplificada é proposto por Baturone (BATURONE ,1994) e apresentada na Figura 20.

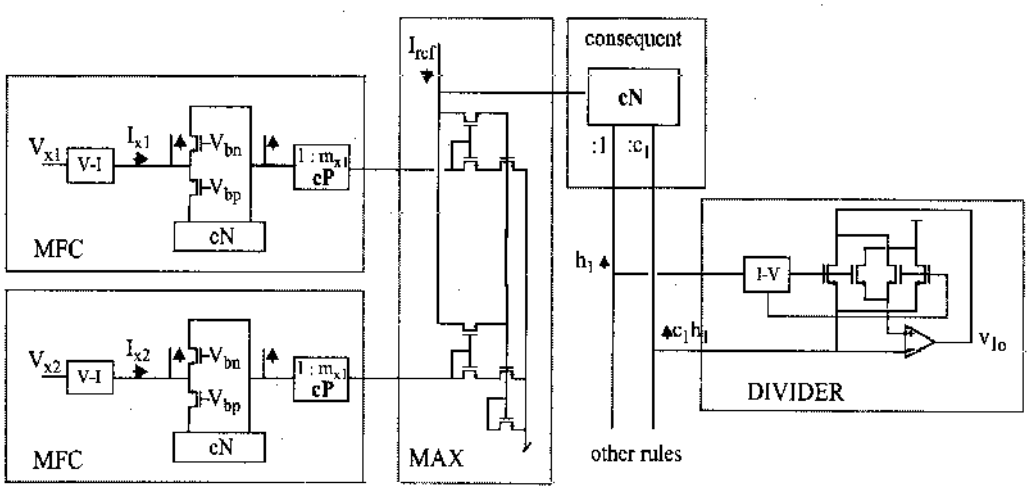


Figura 20 - Arquitetura de máquina de inferência fuzzy proposto por Baturone

Huertas (HUERTAS, 1992) propõe uma arquitetura que define um ciclo de operação cuja duração depende da precisão desejada, pois cada regra é implementada em uma RAM digital com conversor digital-analógico em corrente. Conforme maior precisão, maior número de bits nas regras e conversores d/a, maior duração de tempo em cada operação ilustrado na Figura 21. O núcleo desta arquitetura é um MFC (Conjunto de circuitos com funções de pertinência), hábil para gerar um grande número de funções de pertinência que permite algum tipo de reconfigurabilidade eletrônica. Ver Figura 22.

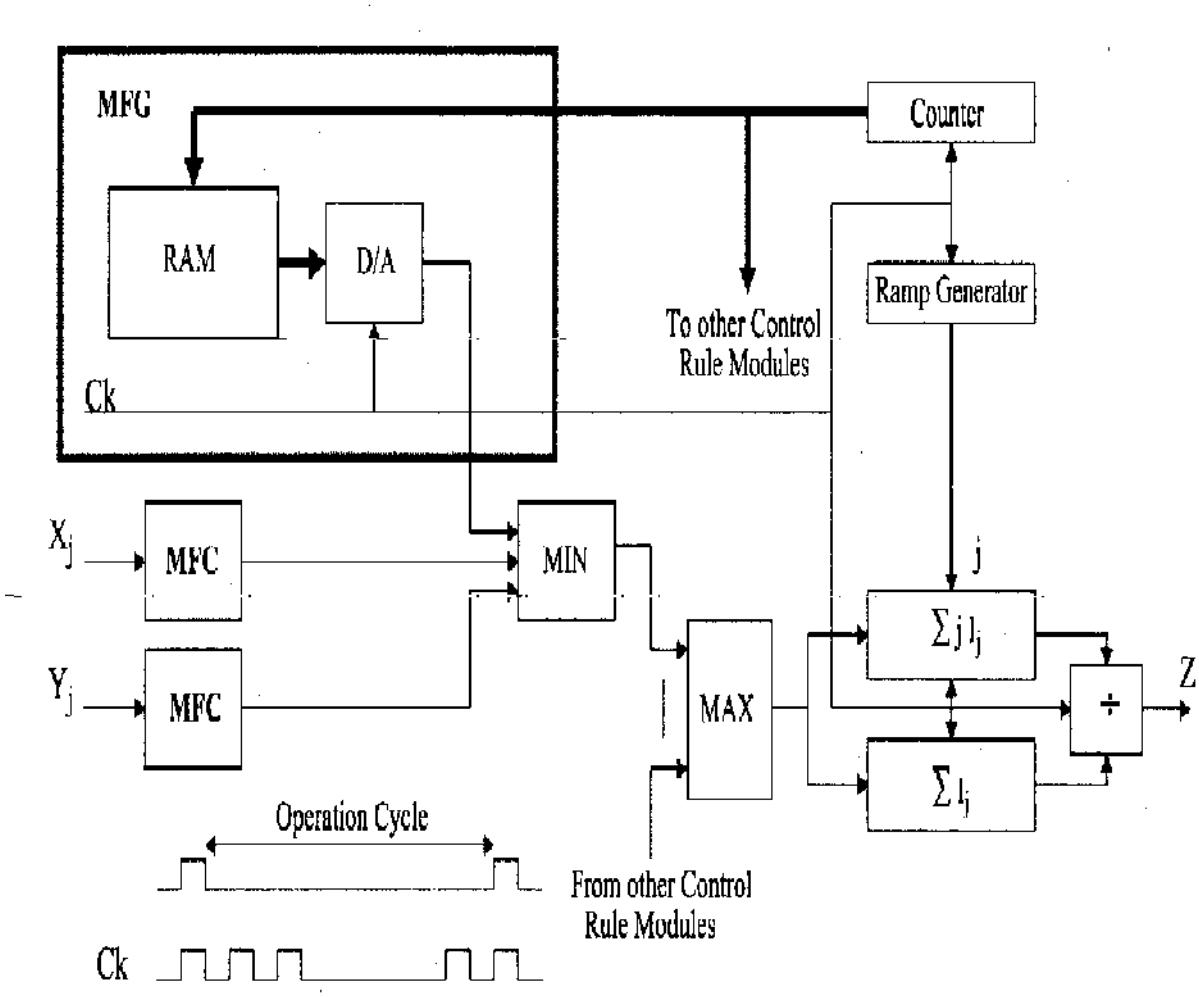


Figura 21 - Arquitetura de máquina de inferência fuzzy proposto por Huertas

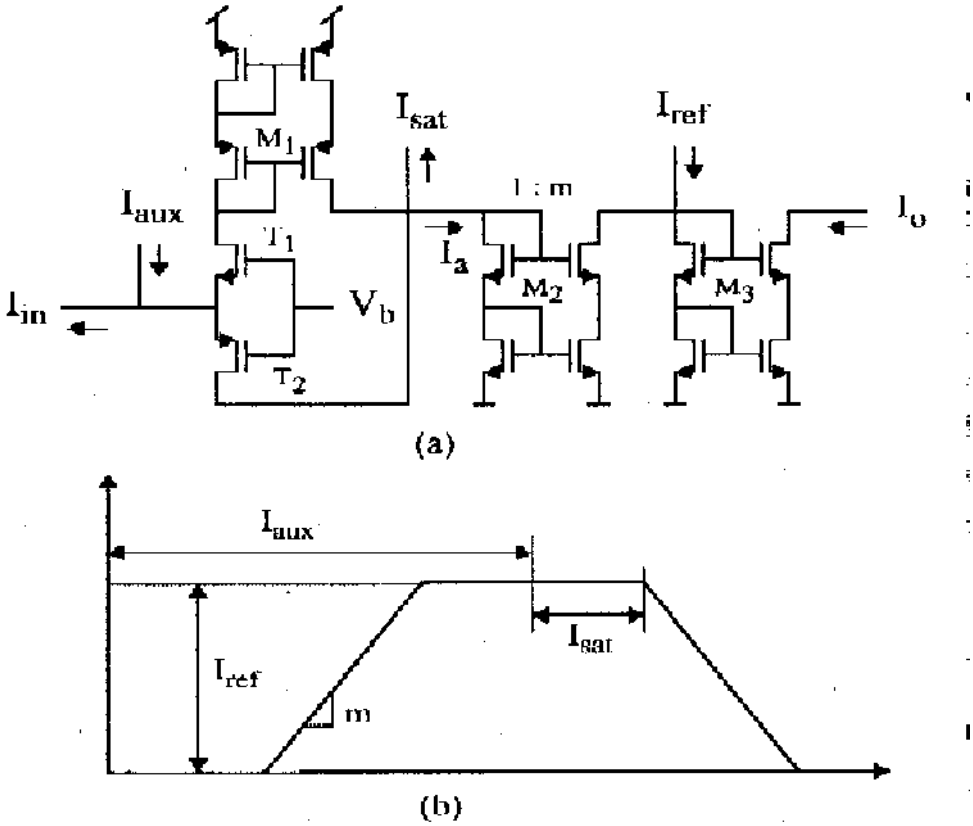


Figura 22 - MFC proposto por Huertas: a) Esquemático b) parâmetros ajustáveis para função trapezoidal

Na arquitetura proposta por Yamakawa, temos:

1. Função de pertinência

O primeiro bloco de uma máquina de inferência fuzzy é um circuito de função de pertinência (MFC). A característica de entrada e saída de um MFC exibe a forma da função de pertinência.

2. Circuito *min*

O segundo bloco é um circuito *min* o qual aceita mais do que um sinal de entrada e produz o menor valor deles em sua saída. Este circuito propõe uma função AND no antecedente das regras if-then. Se as variáveis no antecedente são conectadas com OR, o circuito deve ser um circuito *max*. A saída destes circuitos representa o grau de concordância entre o antecedente e a entrada e segue para o próximo estágio para ponderar a função de pertinência conseqüente.

Os circuitos *min* e *max* podem ser implementados empregando transistores bipolares como mostram as Figuras 23 e 24. O compensador (seguidor de emissor) compensa o

deslocamento de tensão de 0,7V para produzir valores de saída igual a mínima ou máxima em relação à tensão de entrada.

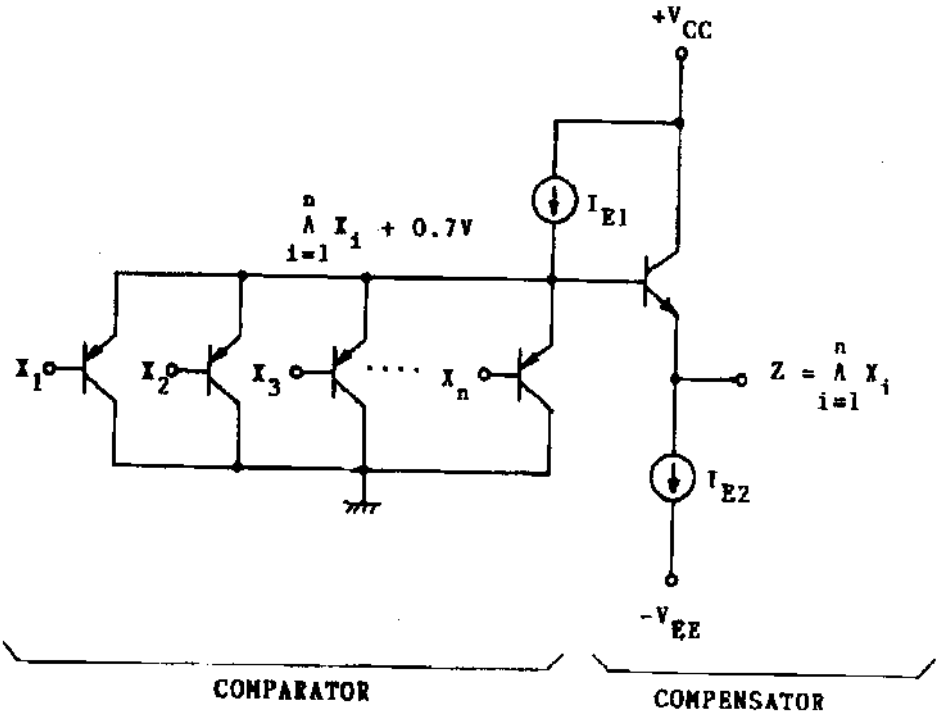


Figura 23 - Circuito *min* utilizando transistor bipolar

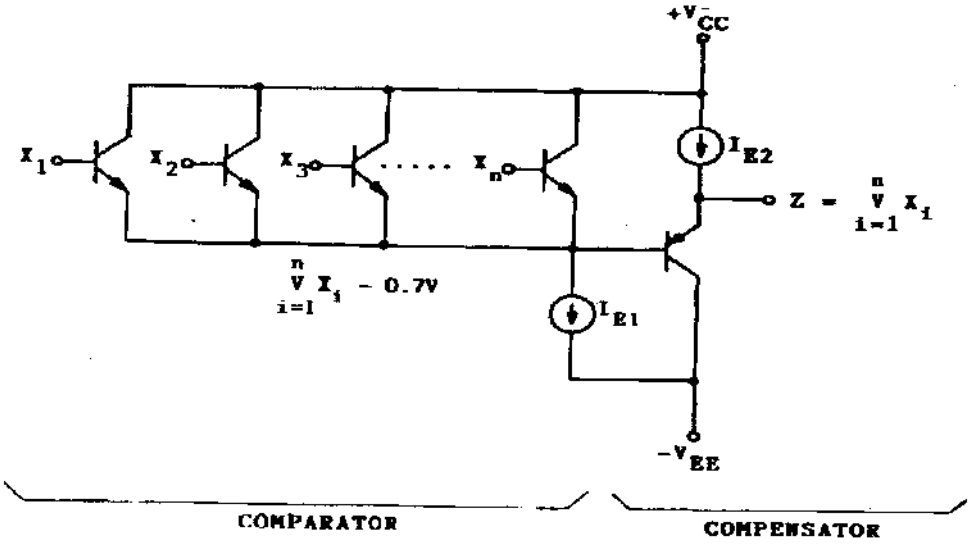


Figura 24 - Circuito *max* utilizando transistor bipolar

A exceção do estágio de defuzzificação, o núcleo da computação da máquina fuzzy pode todo ser implementado usando operadores elementares de min e max, mostrando a importância destes circuitos para implementação em hardware dedicado (CATANIA, V.; RUSSO, M, 1994)..

3. Circuito MFC

Funções de pertinência de variáveis no antecedente são implementadas por um MFC. Um circuito de função de pertinência é o qual as características de entrada e saída podem ser associadas a sinais externos ou dispositivos de captura de valores. A Figura 25 mostra a descrição do circuito de um MFC o qual pode realizar cinco funções: Z, S, trapezoidal, triangular e NA (não). A Figura 26 ilustra os gráficos da função de pertinência, sendo (a) função S, (b) função Z, (c) função triangular e (d) função trapezoidal.

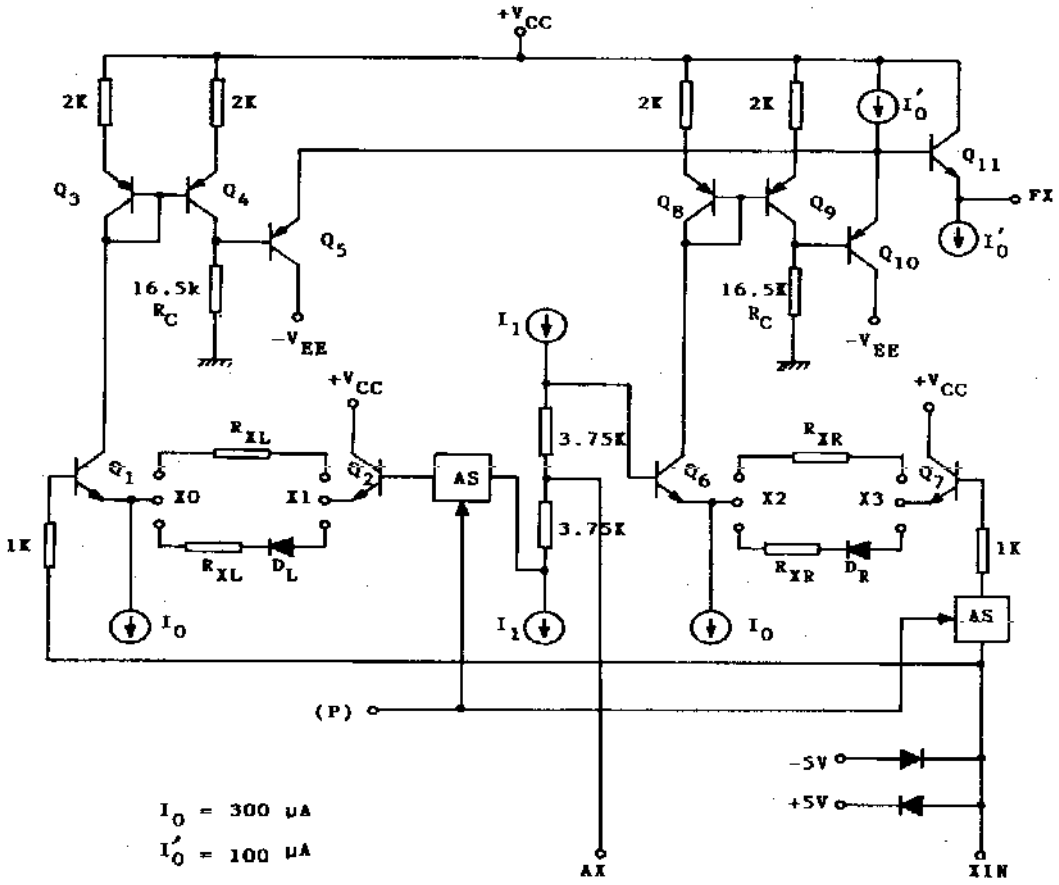
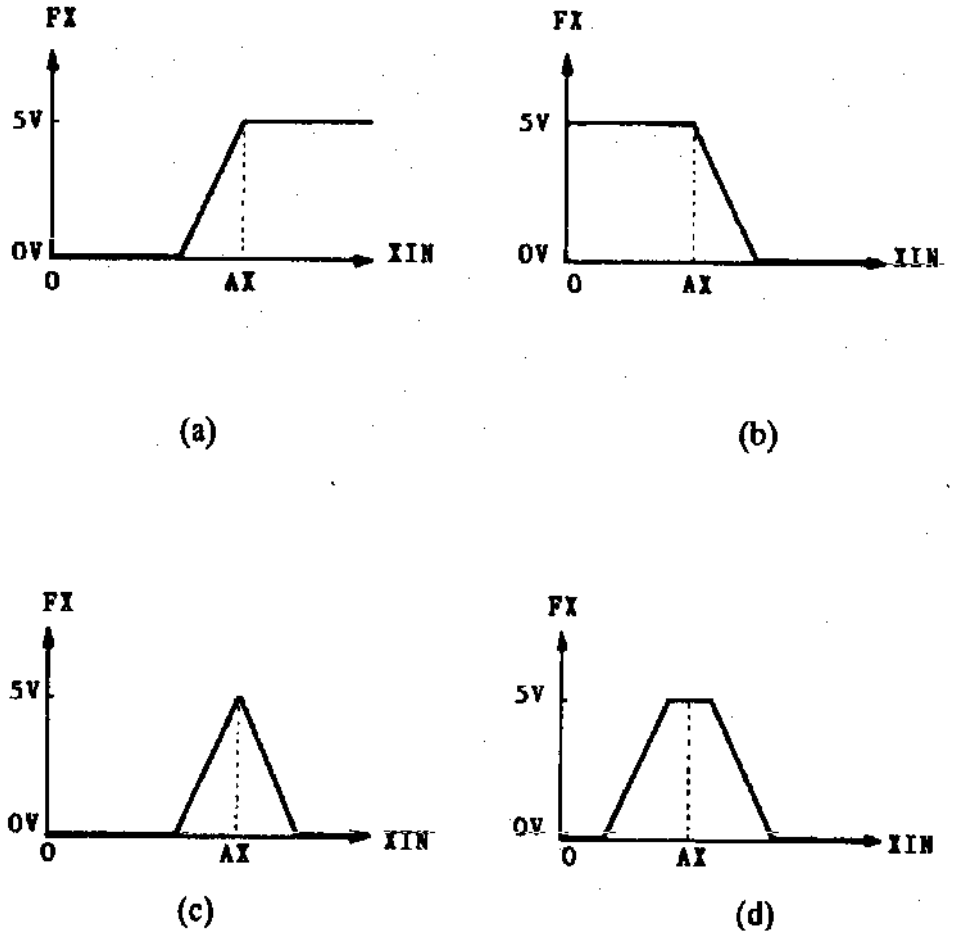


Figura 25 - Circuito MFC

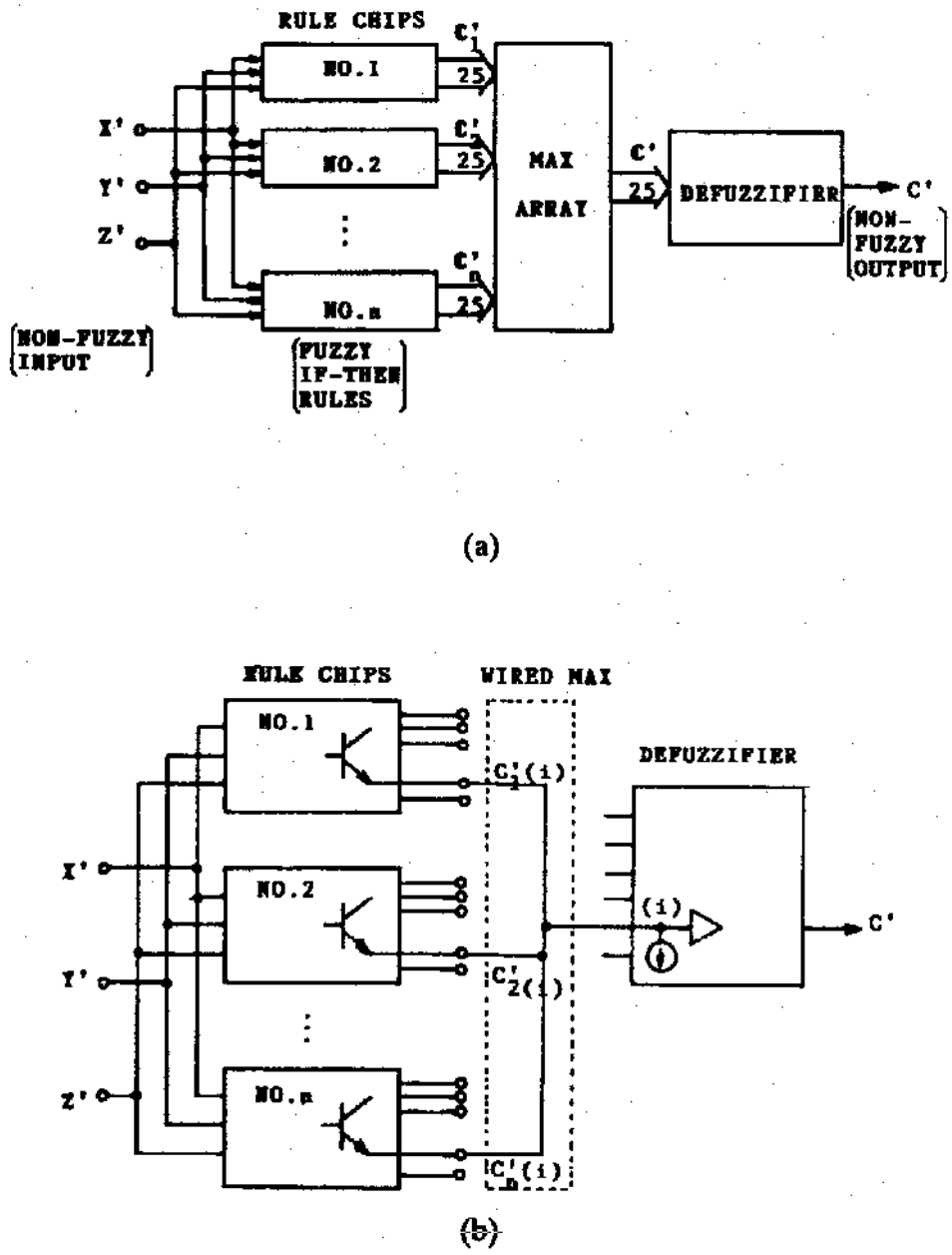


(a) S-function, (b) Z-function, (c) Λ-function (triangular function), and (d) Π- function (trapezoidal function).

Figura 26 - Funções de pertinência da proposta de Yamakawa .

6. Vetor de *max*

A saída de uma máquina de inferência fuzzy deve ser usualmente alimentada a um vetor *max* para agregar todas as conclusões individuais antes da defuzzificação, como mostra a Figura 27. Uma máquina de inferência fuzzy com estrutura de coletor aberto e um defuzzificador com a fonte de corrente no terminal de entrada facilita a sua implementação.



(a) Aggregation of individual conclusions (individual inference results) with a MAX array. (b) Aggregation by wired MAX.

Figura 27 - Vetor *Max* Proposto por Yamakawa

7. Defuzzificador

Para obter um valor determinístico de uma função de pertinência como conclusão final da inferência fuzzy, a defuzzificação deve ser ativada. A Figura 28 mostra um defuzzificador proposto por Yamakawa.

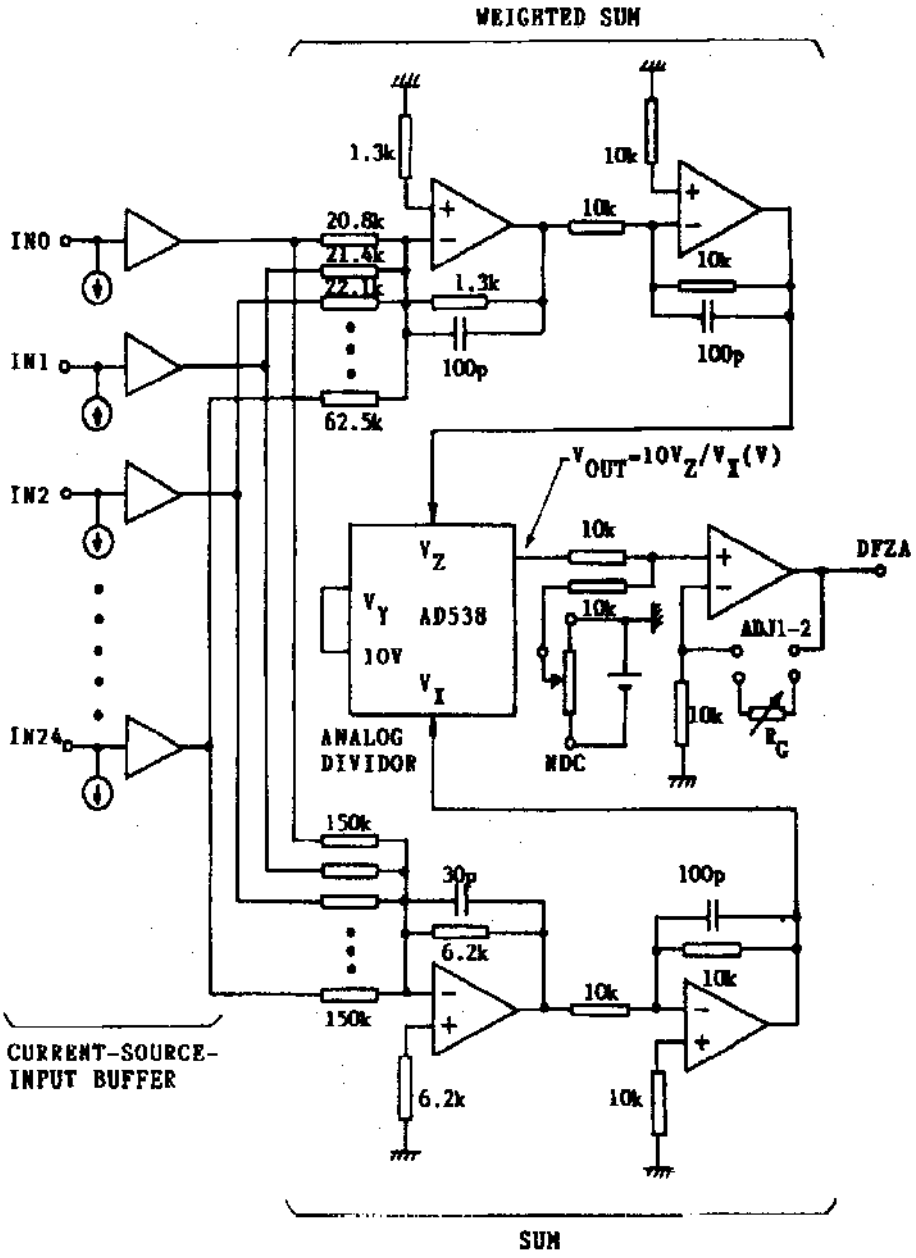


Figura 28 - Circuito defuzzificador de Yamakawa

Sistemas fuzzy típicos como Mamdani ou Takagi-Sugeno inferem uma saída exata (não fuzzy) agregando todas as respostas das regras e realizando uma divisão. Um dos principais problemas que aparecem quando se implementa um sistema fuzzy em hardware é a

seleção do circuito de divisão adequado. O divisor é o circuito que influencia e limita a velocidade, precisão e a capacidade de interfaceamento do bloco de saída do chip fuzzy. Circuitos multiplicadores e divisores podem ser empregados com combinações de conversores A/D e D/A.

Técnicas convencionais log-antilog ou bipolar translinear têm sido empregadas em chips fuzzy analógicos. Estas estruturas podem ser realizadas com tecnologia CMOS usando transistores MOS trabalhando na região abaixo da limiar. Entretanto, a baixa corrente que eles operam fazem o circuito ficar lento. Baixa velocidade é também a limitação de multiplicadores e divisores baseados na técnica de divisão de tempo.

O princípio translinear MOS é outra abordagem. Neste caso, a principal limitação é a baixa resolução por causa do desempenho dos circuitos resultantes que são sensíveis ao desvio da lei quadrática do modelo dos transistores em saturação. Uma outra técnica amplamente empregada é alterar o multiplicador usando realimentação local ou inserindo no caminho de realimentação um amplificador inversor. A precisão é limitada pelos offsets associados com entrada e saída das variáveis.

Outra alternativa é a técnica de aproximação sucessiva (BATURONE, 2000) que emprega dois acumuladores.

Enfim, existem na literatura diversas propostas de arquiteturas de implementação de sistemas fuzzy analógicos, cada uma propondo uma abordagem diferente de implementação, visando obter os resultados do processo em um menor tempo de execução eletrônica.

2.3 – Utilização de FPGA e FPAA

Os chips baseados em tecnologia FPGA (*Field Programmable Gate Array*) ou FPAA (*Field Programmable Analog Array*) se apresentam como um tipo de hardware programável, contendo uma estrutura interna em blocos formada por circuitos operacionais, conversores A/D, chaves programáveis, e outros tipos de componentes programáveis, permitindo uma aplicação dedicada a um fim específico, programado por um especialista via software, como uma aplicação fuzzy, ou alguma técnica de computação evolucionária, como o AG (Algoritmo Genético). Neste último caso, o hardware é reconfigurado pelos bits de configuração determinados pelo algoritmo evolucionário implementado em software (AMARAL, 2003). Esta tendência de se usar FPGA em circuitos inteligentes são motivo de pesquisa na área de sistemas auto-programáveis (LOHN & COLOMBANO, 1998; MILLER & THOMPSON,

1998). O FPGA geralmente se aplica a circuitos no domínio do processamento digital de sinais e computação reconfigurável.

2.4 Implementação baseada em Mapeamento Digital

O hardware de um sistema fuzzy via Mapeamento Digital fornece de modo automático a resposta desejada em função dos sinais apresentados na entrada da plataforma. Cada combinação de sinais na entrada já está com sua devida saída armazenada na memória interna da plataforma. Este "mapa de resultados" (*look*-up *table*) é carregado de um sistema externo de inferência fuzzy, como o Matlab, e fica armazenado na memória interna da plataforma. Os sinais analógicos presentes na entrada da plataforma são convertidos em digitais e formam um endereço para apontar na memória da plataforma qual é o dado a ser apresentado na saída da plataforma, conforme mostrado na Figura 29

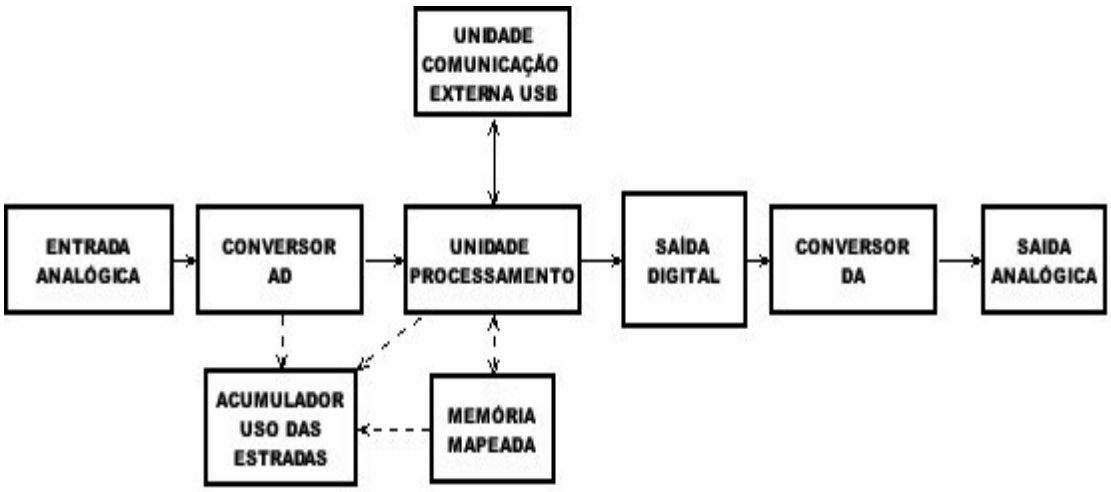


Figura 29 – Estrutura Funcional em Blocos da Plataforma

Uma vantagem deste sistema é a forma rápida da resposta a cada uma das combinações na entrada do sistema. Outra é a possibilidade de verificar se realmente o processo de execução real está usando todo o mapa de decisões. Esta característica exploratória será utilizada para a pesquisa nesta dissertação.

A desvantagem é a dependência de um sistema externo para formalizar o mapeamento digital para ser carregado na plataforma. Além do que, a transferência não é de forma tão rápida quando se usa um arquivo que tenha combinações de entradas com quantização binária elevadas, acima de quatro bits. Como proposto nesta dissertação, têm-se três entradas com uma conversão analógico-digital de até oito bits em cada entrada, podendo ter um total de

16777216 combinações possíveis para serem transferidos para a memória interna da plataforma. Com isso temos uma necessidade de memória considerável para a plataforma poder guardar o mapeamento, além de uma memória extra para servir de acumulador no sistema proposto de sensibilidade do uso das variáveis de entrada.

Os sistemas via hardware possuem tolerâncias em seus componentes físicos, além de estarem sujeitos às variações climáticas, ruídos ESD, EMI, ou mesmo ao desgaste físico em uso contínuo. Por serem fatores reais intrínsecos ao hardware, e de magnitude variável, não estão no escopo de variáveis de entrada, e sim em níveis de tolerância permitida dentro do sistema como um todo. Tais fatores somente podem ser testados em uma das plataformas reais de execução.

Sistemas com mapeamentos dinâmicos em hardware têm limitações quando se trata do armazenamento dos dados, devido à quantificação binária das entradas das variáveis. Quanto maior o número de bits para cada entrada, e quanto maior for a quantidade de entradas, maior será a necessidade de se ter mais memória de acesso rápido para armazenar a grande quantidade de dados.

2.5 Implementação com Microprocessadores e Microcontroladores

Na implementação direta em unidades microprocessadas é feita toda a gama de cálculos da inferência fuzzy via software local inerente ao microprocessador utilizado. Possui a versatilidade de ter na própria unidade as entradas, máximo e mínimo, regras, fuzzificaçao, defuzzificação e a saída. O tempo de processamento dos dados fica intimamente ligado ao clock de processamento e à quantização binária da conversão das entradas. Quanto maior a quantização, maior será o tempo das operações na unidade aritmética do processador. Alguns cálculos podem gerar tempos de centenas de clocks de máquina por operação, principalmente os cálculos envolvendo divisões sucessivas com ponto fixo. Comparativamente ao sistema utilizado nesta plataforma, que também utiliza um microcontrolador de oito bits, no caso de implementação dos cálculos com processamento matemático interno, o tempo de execução de cada inferência pode chegar a 1 milissegundo, devido as rotinas de tratamento dos restos do ponto flutuante em microcontroladores de 8 bits. Este tempo corresponde a 30 vezes o tempo de acesso ao cartão SD, em que os dados estão gravados e indexados serialmente na memória.

Uma solução mais adequada em relação aos erros de processamento e tempo de execução seria usar microcontroladores DSP (*Digital Signal Processing*) de 32 bits. São unidades com processamento e controle de execução em unidades internas distintas, tornando

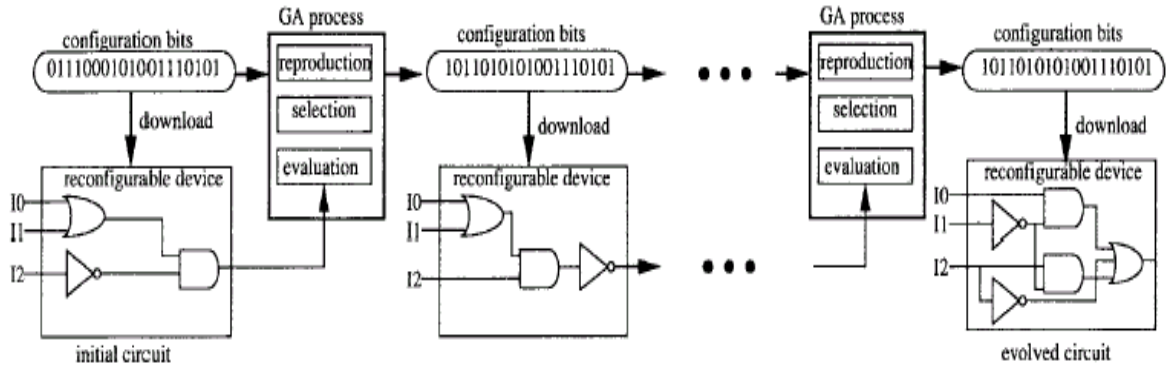
mais rápido o processamento matemático e viabilizando o uso direto dos microcontroladores na inferência fuzzy. Apesar de preços maiores que os microcontroladores convencionais, nas aplicações em que se faça necessário este tipo de inferência, tais dispositivos justificam o investimento. Vale ressaltar que o uso de sistemas fuzzy com quatro variáveis de entrada iria sobrecarregar mais os cálculos de inferência em sistemas microprocessados com cálculos diretos na plataforma, enquanto que no mapeador teríamos apenas que acrescentar mais dados à memória do cartão SD referente a expansão de mais um campo de oito bis no endereçamento.

2.6 Implementação com Eletrônica Evolucionária

A Computação Evolucionária é uma área de estudo que busca a síntese de algoritmos evolucionários inspirados em aspectos essenciais da evolução natural e da genética e os aplica na solução de problemas em geral. No universo dos algoritmos evolucionários, o Algoritmo Genético (AG) (HOLLAND, 1975; GOLDBERG, 1989) encontrou uma maior difusão, com ênfase em aplicações relacionadas à otimização. A utilização de Algoritmos Genéticos tem sido ampliada nos últimos anos: além de problemas de otimização, estes algoritmos também têm sido empregados em síntese de sistemas (BLICKLE, 1996; HARVEY, 1993), principalmente na área de engenharia.

Grandes avanços têm sido obtidos na síntese de circuitos eletrônicos utilizando-se Computação Evolucionária (YAO & HIGUCHI, 1996). O grande interesse demonstrado por esta nova área de estudos levou à criação de uma nova nomenclatura para a mesma: Eletrônica Evolucionária (SIPPER, 1997).

No Hardware Evolucionário aplica-se um Algoritmo Evolucionário para buscar (ou otimizar), dentre todos os possíveis circuitos do espaço de busca, a solução, isto é, o circuito, que obedece ao critério de aptidão estabelecido - por exemplo, a configuração de um circuito que apresente uma determinada saída desejada. Os circuitos são geralmente representados por uma codificação binária, o cromossomo. Inicialmente, uma população é criada, e todos os indivíduos (circuitos) avaliados. Parte desta população é mantida, os melhores indivíduos têm mais chances de serem selecionados para reproduzir e formar a população subseqüente, até que o objetivo seja atingido. A Figura 30 ilustra o processo básico de evolução quando aplicado ao desenvolvimento de circuitos digitais.



Basic concept of evolvable hardware.

Figura 30 – Figura básica de Hardware Evolutivo (HIGUCHI et al., 1999).

Este processo de aprendizado e reconfiguração dos componentes eletrônicos nesta técnica evolutiva permitem a criação de um sistema que parte de uma análise do problema para criar uma solução com o mérito voltado ao alvo final esperado. No processo evolutivo, existe uma evolução do sistema no “GA Process” e os resultados vão reconfigurando o circuito, que pode ser um FPGA, até ter um circuito final, apropriado aos objetivos do sistema.

Quanto ao tipo de projeto, a eletrônica evolucionária é dividida em otimização, síntese e auto-reparo de circuitos. Nestes processos utilizam-se componentes eletrônicos com funções reprogramáveis, associados às técnicas de implementação evolucionária.

O projeto eletrônico evolucionário fuzzy pode ser um circuito digital ou um circuito analógico.

A síntese de circuitos digitais é feita basicamente por duas formas básicas de representação: portas e funções lógicas (HIGUCHI et al., 1997). No primeiro caso, a unidade básica de circuito digital manipulada pelo sistema evolutivo é uma porta lógica, isto é, AND, OR, NOR, NAND ou XOR. No segundo caso, a unidade básica manipulada pelo algoritmo evolutivo é uma função digital mais complexa, como uma unidade lógica aritmética ou um produto de variáveis lógicas. Em geral, cada gene codifica a natureza do bloco de construção do circuito e a origem das entradas do mesmo (THOMPSON, 1996). Normalmente, a avaliação consiste na comparação com a especificação do circuito digital, dada na forma de uma tabela verdade (TOCCI & WIDMER, 2000).

Existem algumas áreas de aplicação nas quais a implementação de um sistema fuzzy em hardware analógico pode oferecer vantagens significativas. Estas aplicações estão

relacionadas com requisitos específicos, tais como, maior desempenho e baixo consumo. Cabe ressaltar também a utilização de implementações especiais de sistemas fuzzy em hardware analógico para funcionar como blocos nos chamados sistemas dedicados (*embedded systems*) e dentro de soluções do tipo *System-on-Chip* (isto é, quando o sistema completo, incluindo o módulo fuzzy, é implementado no mesmo chip).

As implementações de sistemas fuzzy em hardware analógico (YAMAKAWA, 1993; GUO ET AL., 1996) têm seguido as normas comuns do projeto de circuitos eletrônicos, baseadas principalmente na experiência e na criatividade do projetista. No entanto, grandes avanços têm sido obtidos na síntese de evolucionária de circuitos eletrônicos (YAO & HIGUCHI, 1996). O projeto evolutivo de um circuito engloba a concepção de sua topologia e a seleção dos tipos e valores dos componentes. Ele abrange o amplo campo de pesquisa de utilização de algoritmos evolucionários em otimização e/ou síntese de circuitos eletrônicos (ZEBULUM ET AL., 2001). Recentemente, sua aplicação na síntese de circuitos eletrônicos analógicos que implementam sistemas fuzzy vem sendo explorada (AMARAL, 2003) e apresenta grandes potencialidades.

Diversos “circuitos fuzzy” foram sintetizados pela técnica evolutiva. Tais circuitos foram evoluídos em plataformas de desenvolvimento flexíveis projetadas para viabilizar a continuidade das pesquisas nesta área. A área de Eletrônica Evolucionária tem progredido bastante e num futuro não muito distante, sistemas evolucionários inteligentes poderão realizar tarefas úteis no mundo real (YAO & HIGUCHI, 1996).

CAPÍTULO 3
MEMÓRIA SD E INTERFACE USB

O surgimento das memórias seriais de alta velocidade de acesso aos dados e de grande capacidade de armazenamento, como as do tipo SD (*Secure Digital*), foi fator determinante para a realização desta pesquisa, pois viabilizou a implementação da arquitetura baseada em mapeador analógico-digital proposta. O estudo deste tipo de memória, que por suas características permitiu o aumento do número de bits das conversões A/D, é apresentado neste capítulo. Além disso, o capítulo aborda a interface USB (*Universal Serial Bus*) utilizada para conexão da plataforma eletrônica ao computador de desenvolvimento.

3.1 – Memória SD

Optou-se por utilizar a memória tipo cartão SD (Cartão de Memória Digital Seguro) que tem seu endereçamento, envio e recepção de dados serialmente, via SPI de altíssima velocidade, na faixa de MHz de transmissão, tornando a busca dos dados requisitados pelo UPPC (Unidade de Processamento e Periféricos Central – microcontrolador principal da plataforma) em no máximo 0,002 segundos em um cartão de 2 Gb. O SD tem um microcontrolador interno que controla a memória flash (apague, leia, escreva e controle de erro) dentro do cartão de memória. Os dados são transferidos entre o cartão de memória e o controlador de principal em unidades de 512 bytes por bloco, de forma que isto pode ser visto como um disco rígido genérico nos programas de aplicação. É uma unidade completa similar ao sistema de unidade de disco dos computadores pessoais – Hard Disk – HD, que precisa receber uma formatação antes de ser utilizada.

O sistema de arquivos pode ser formatado em FAT 16 ou FAT 32, sendo este último definido para o cartão SD usado neste trabalho. Existem dois modelos de memória em cartões com invólucros parecidos, o SD e o MMC (*Multi Media Card*). As memórias armazenam os dados em um chip não volátil que, mesmo sem a alimentação da fonte, preservam as informações nelas armazenadas. São memórias que podem ser regravadas.

Tendo em vista que a necessidade real da plataforma será de 16.777.216 posições de memória (3 entradas de 8 bits), muito em relação aos sistemas de memória usados anteriormente, temos agora, com um cartão de 2Gb, mais de 2.000.000.000 posições de memória disponíveis. Isto permitirá em trabalhos futuros, uma implementação de histórico de atividades armazenados na própria memória da plataforma, ou qualquer tipo de evento extra que se queira armazenar para futuras consultas.

3.1.1 Pinagem e funcionamento da Memória SD

Conforme ilustrado nas Figuras 31 e 32, existem dois modelos de memória em cartões com invólucros parecidos, o SD aplicado neste trabalho e o MMC (*Multi Media Card*). Porém, cada um tem sua pinagem específica, motivo este de cuidados na hora de estabelecer as vias de dados de comunicação do mesmo.

Figura 31 – Cartões SD e MMC

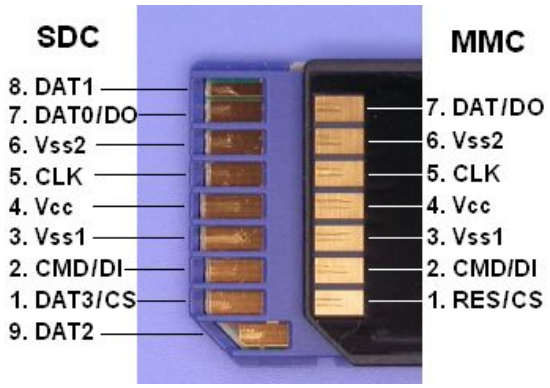


Figura 32 – Pinagem SD e MMC

A faixa de alimentação está entre 2,7 e 3,6 volts e três pinos do cartão são usados para este fim. Os demais seis pinos são usados na comunicação SPI do cartão.

A sequência de comando – FRAME enviado pelo UPPC tem um comprimento fixo de seis bytes no pacote. Quando uma sequência de comando é transmitida ao cartão, uma resposta para o comando (R1, R2 ou R3) será mandado de volta. Por ser uma transferência de dados síncrona, o UPPC tem que continuar lendo bytes até que receba qualquer resposta válida. A resposta de comando (NCR) é de 0 a 8 bytes para cartões do tipo SD. O sinal de CS

deve ser nível baixo durante uma transação (comando, resposta e dados, se for o caso). O campo de CRC é opcional em modo de SPI, por ser uma transmissão de dados baseado em sincronismo de clock, mas é requerido como campo para compor a sequência de comando. O sinal de DI deve ser mantido alto durante transferência de leitura. Ver Figura 33

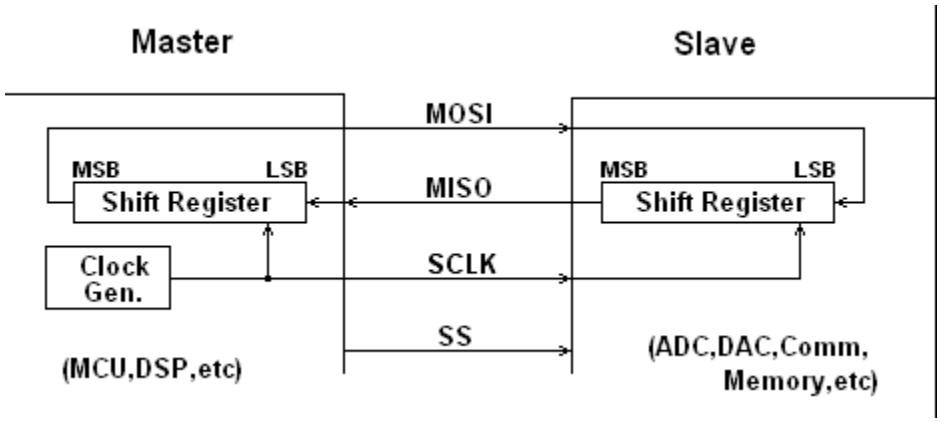


Figura 33 – Gráfico Comunicação SPI

A resposta que o cartão providencia para os comandos recebidos mostra o estado que se encontra o mesmo, se a recepção foi correta, ou mesmo se não recebeu nada, com o não envio de nenhuma resposta para o UPPC. Na Figura 34 temos um resumo de duas respostas que determinam condições iniciais e de respostas nas vias de leitura ou escrita no cartão SD.

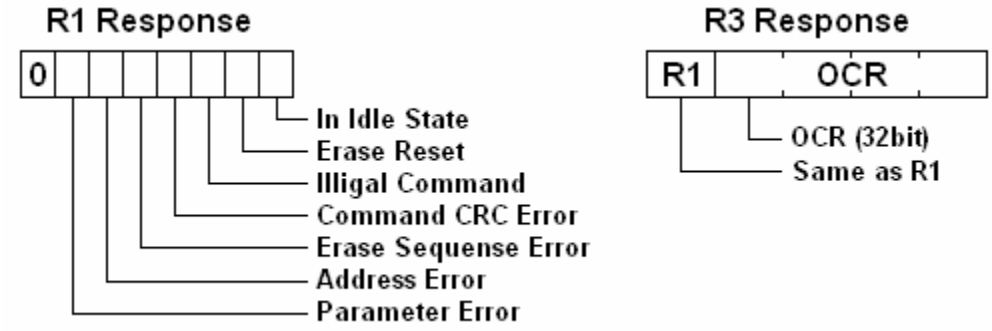


Figura 34 – Registradores de Resposta do Cartão SD

Os comandos de execução SPI são definidos em 64 comandos. Cada comando é representado em abreviação como GO_IDLE_STATE ou CMD <n>, sendo <n> o número

do índice de comando e o valor pode ser 0 a 63. Na Tabela 1 são apresentados os comandos normalmente usados para ler/escrever (*read/write*) de todos os cartões genéricos SD e inicialização dos mesmos.

Tabela 1 – Tabela de Comandos da memória SD

CommandI	**Argument**	**Response**	**Data**	**Abbreviation**	**Description**
CMD0	None(0)	R1	No	GO_IDLE_STATE	Software reset.
CMD1	None(0)	R1	No	SEND_OP_COND	Initiate initialization process.
ACMD41(*1)	*2	R1	No	APP_SEND_OP_COND	For only SDC. Initiate
CMD8	*3	R7	No	SEND_IF_COND	For only SDC V2. Check voltage range.
CMD9	None(0)	R1	Yes	SEND_CSD	Read CSD register.
CMD10	None(0)	R1	Yes	SEND_CID	Read CID register.
CMD12	None(0)	R1b	No	STOP_TRANSMISSION	Stop to read data.
CMD16	Block length[31:0]	R1	No	SET_BLOCKLEN	Change R/W block size.
CMD17	Address[31:0]	R1	Yes	READ_SINGLE_BLOCK	Read a block.
CMD18	Address[31:0]	R1	Yes	READ_MULTIPLE_BLOCK	Read multiple blocks.
CMD23	Number of blocks[15:0]	R1	No	SET_BLOCK_COUNT	For only MMC. Define number of blocks to transfer with next multi-block read/write command.
ACMD23(*1)	Number of blocks[22:0]	R1	No	SET_WR_BLOCK_ERASE_COUNT	For only SDC. Define number of blocks to pre-erase with next multi-block write command.
CMD24	Address[31:0]	R1	Yes	WRITE_BLOCK	Write a block.
CMD25	Address[31:0]	R1	Yes	WRITE_MULTIPLE_BLOCK	Write multiple blocks.
CMD55(*1)	None(0)	R1	No	APP_CMD	Application specific command.
CMD58	None(0)	R3	No	READ_OCR	Read OCR.
*1:ACMD<n> means a command sequense of CMD55-CMD<n>. *2: Rsv(0)[31], HCS[30], Rsv(0)[29:0] *3: Rsv(0)[31:12], Supply Voltage(1)[11:8], Check Pattern(0xAA)[7:0]					

3.1.2 – Procedimentos iniciais do Cartão SD

Após inserir o cartão ou ligar alimentação do mesmo, quando ele já estiver inserido, que é o caso desta plataforma, quando a tensão de alimentação alcançou 2,2 volts, tem que esperar pelo menos por um milissegundo. Depois, com a entrada DI e CS em nível alto, deve-se aplicar 74 pulsos de clock no SCLK e o cartão estará estável e pronto para aceitar comandos nativos.

A taxa de transmissão, também conhecida como BAUD RATE, deve estar inicialmente entre 100 e 400 KHz, sendo esta freqüência aplicada ao sinal de CLOCK. Então envia-se um CMD0 com CS em nível baixo para inicializar o cartão (*reset*). O cartão verifica CS quando um CMD0 é recebido. Se o sinal de CS for baixo, o cartão entra em modo de SPI. Considerando que o CMD0 deve ser enviado como um comando nativo, o campo de CRC tem que ter um valor válido. Quando o cartão entrar em modo de SPI, a característica de verificar CRC ficará inativa e o CRC não é mais conferido, pois nesta transmissão via sincronismo de clock, não existe a necessidade de conferência de CRC, de forma que a rotina de transmissão de comando tem que ser escrita com o CRC válido somente para CMD0 e CMD8. Quando o CMD0 é aceito, o cartão entrará em estado inicial de RESET e ficará INATIVO e restrito a alguns comandos específicos. O cartão, neste caso, enviará uma resposta no registro R1 com valor (0x01).

Em estado inativo, o cartão aceita só CMD0, CMD1 e CMD58, ou CMD0, CMD41 e CMD58 em alguns cartões. Qualquer outro comando será rejeitado. Por este tempo, verifica a faixa de tensão de funcionamento indicada no OCR. Este registro vem como resposta do CMD58, e se estiver fora da especificação de tensão, significa que o cartão está danificado. O cartão, ao receber um CMD1, vai enviar uma resposta em R1 de 0x00, demonstrando que o cartão foi inicializado e está pronto para receber os comandos de LER/ESCREVER (read/write). Como temos cartões que em vez de receber CMD1 devem receber o CMD41, então se deve enviar CMD41 primeiro e, se for rejeitado, tentar novamente com CMD1. O processo inicial de inicialização pode demorar algumas centenas de milissegundos, dependendo da capacidade de memória do mesmo.

O campo de TRAN_SPEED no CSD indica a taxa máxima de clock do cartão. A taxa máxima é 25 MHz para SD na maioria dos casos depois da inicialização.

Os blocos podem ser configurados com tamanhos de 512 bytes até 2 GB no cartão, de forma que o tamanho do bloco deve ser re-inicializado com CMD16. No caso deste trabalho,

fixou-se em blocos de 512 bytes. Isto significa que quando precisamos ler um byte em um determinado endereço, devemos acessar o bloco inteiro deste endereço, carregá-lo na RAM do UPPC, então teremos acesso ao byte daquele endereço. Como exemplo, caso a entrada defina um endereço do tipo 130BE, este endereço fará parte de um bloco de 512 endereços, de 1305A até 1356B. Todos os bytes neste bloco serão copiados para dentro da RAM do UPPC e para ter acesso ao byte desejado no 130BE.

3.1.3 – Pacote de Dados

O bloco de dados é transferido como um pacote de dados que consiste em Símbolo (Token), Bloco de Dados (Data Block) e CRC. O formato do pacote de dados é ilustrado na Figura 35.

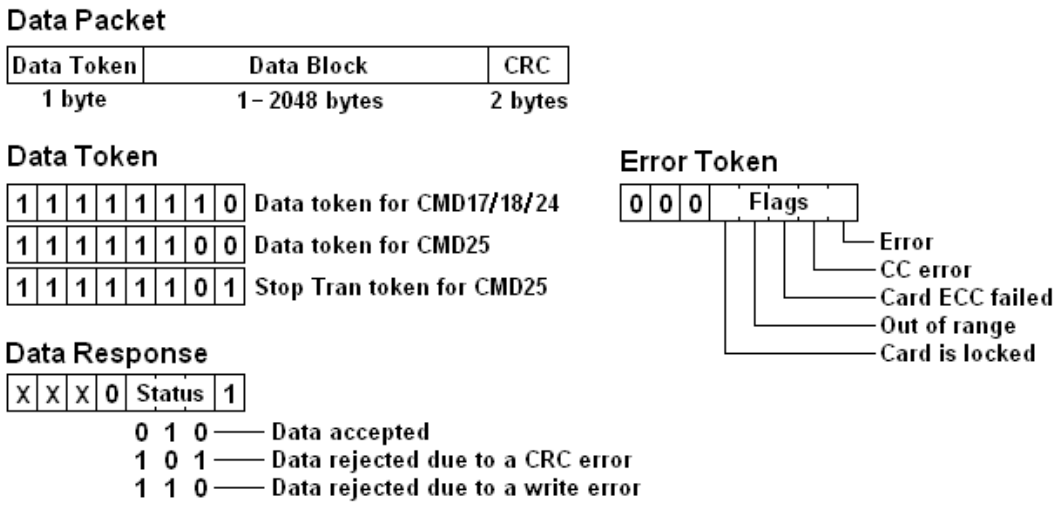


Figura 35 – Bloco de dados da Memória SD

3.1.4 – Lendo um Bloco do SD

O CMD17 tem como argumento o endereço requisitado para devida leitura do bloco. Significa que se colocarmos como argumento do CMD17 o valor 0, este vai ler o primeiro bloco de 512 Bytes alocados na memória do cartão SD, isto porque foi definido este valor de bloco a ser lido. Quando se envia este comando, espera-se um retorno de resposta do comando, para verificar se foi aceito ou não. Se esta resposta não retornar, o UPPC deve

indicar falha no dispositivo. Caso retorne e com definição do estado sem erro, logo depois virá o pacote de dados com os respectivos bytes a serem guardados no UPPC. Ver Figura 36.

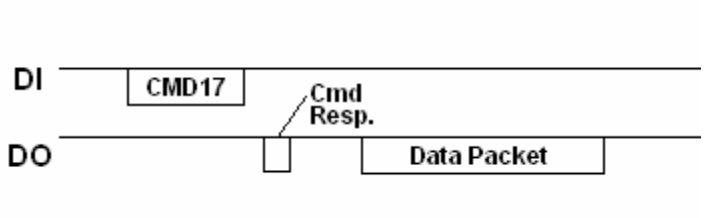


Figura 36 – Sincronização DI / DO de Leitura

3.1.5 – Escrevendo no SD

Ao enviar o comando de escrita CMD24, têm-se o tempo de 1 byte para receber a resposta do SD de comando aceito ou não. Caso o comando seja aceito, deve-se enviar o Pacote de Dados, que será recebido pelo SD e gravado na respectiva posição de memória enviada no argumento do CMD24. Ao término do envio deste pacote, o SD responderá novamente para verificar se houve algum erro ou se ocorreu tudo corretamente. Ver Figura 37.

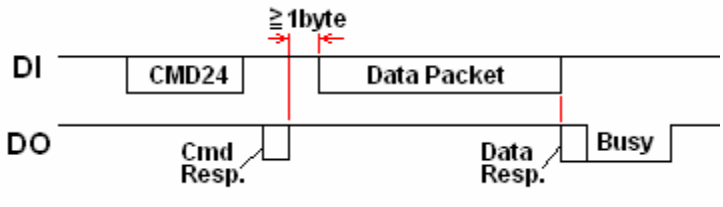


Figura 37 - Sincronização DI / DO de Escrita

O comando de escrever Blocos seguidos de dados no SD é o CMD25, com argumento apontando para o endereço inicial de escrita dos blocos. Após enviar o comando, o UPPC obterá uma resposta do SD indicando se está tudo certo para receber o pacote ou não, caso tenha ocorrido algum erro. Estando tudo certo, é enviado o primeiro pacote de dados. Após este envio, o UPPC receberá novamente uma resposta, com o mesmo objetivo anterior. Antes de enviar o segundo pacote em seqüência, o UPPC deve estar controlando a subida do nível lógico do DO, de 0 para 1. Pois enquanto está em 0, indica que o cartão SD está ocupado. Este processo continua até que seja enviado um Símbolo (Token dentro do pacote de dados) que

indique fim de transmissão, que é o código 11111101. Neste momento, o SD termina o último byte que estava sendo transmitido e que deve ser descartado pelo UPPC. A Figura 38 mostra o gráfico desta operação, que será utilizada quando o PC descarregar o arquivo Fuzzy para dentro do SD, através do UPPC.

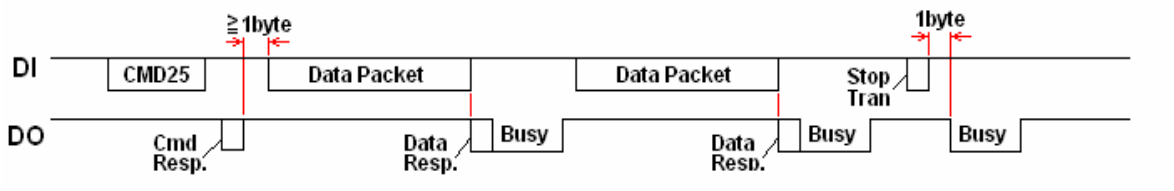


Figura 38 - Sincronização DI / DO de Escrita Ininterrupta

3.2 – Interface USB (Universal Serial Bus)

A interface serial USB é amplamente utilizada na transferência de dados devido ao fato de ser de alta velocidade e apenas usar quatro terminais de interligação. esta interface foi escolhida com o propósito de transferir os dados do PC para a Plataforma através de uma interface padronizada nos PCs.

O tradicional meio de transferência de dados do computador para sistemas externos via comunicação serial RS232 tornou-se um padrão conhecido amplamente no meio acadêmico e para os profissionais no âmbito da engenharia aplicada. Porém, com o advento das conexões USB (*Universal Serial Bus*), as portas seriais do tipo RS232 que tinham sua saída do PC via conector DB9 estão praticamente fora de uso nos novos PCs. Então, um foco de estudos para este trabalho foi a aplicação da interconexão Computador - Plataforma via USB. Tal estudo objetivou, principalmente, tornar viável a utilização da plataforma eletrônica aqui desenvolvida com um grande número de computadores modernos atuais, inclusive notebooks, que, em sua maioria, só possuem uma ou mais interfaces do tipo USB.

A Figura 39 mostra o diagrama em blocos da comunicação via USB, integrada no próprio microcontrolador usado neste projeto.

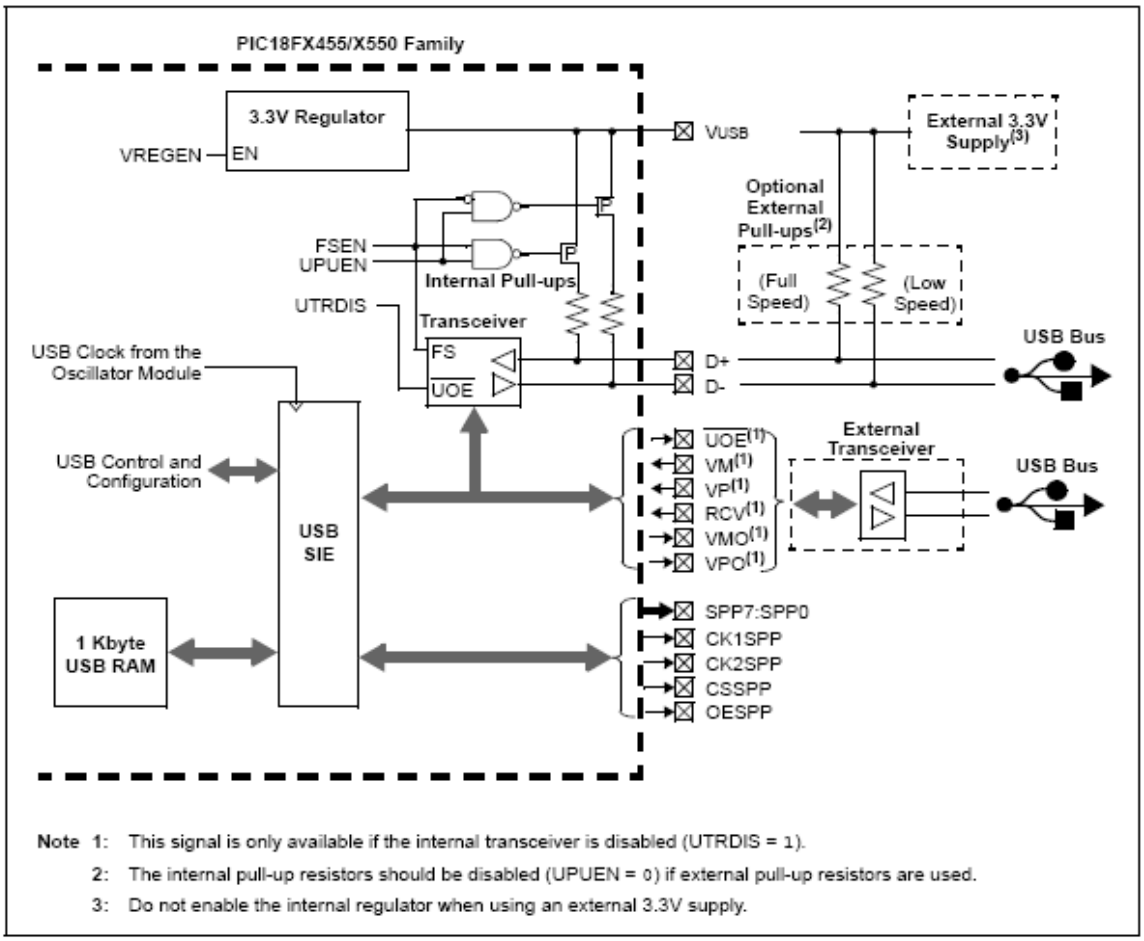


Figura 39 – USB integrado no Microcontrolador PIC18F4550

Para utilização da comunicação USB, o UPPC tem internamente um Buffer – área reservada de memória, que é de 1000 Bytes, para receber e transmitir dados. Este controle é feito através de diversos registradores e Flags de controle num total de 22 registradores. Conseqüentemente, para se utilizar esta função USB interna do microcontrolador, foi decidido implementar o software através da linguagem C, usando o compilador CCS que permite a integração diretamente com todos os registradores necessários. Além disso, decidiu-se instalar um script no PC transformando uma porta serial virtual Com4 em saída USB para a plataforma desenvolvida.

CAPÍTULO 4
IMPLEMENTAÇÃO DA PLATAFORMA EM BLOCOS

Este capítulo descreve o funcionamento detalhado da plataforma eletrônica proposta e da construção do protótipo. Também é apresentada e discutida a interconexão entre a Plataforma e o sistema gerador do PC, sendo que este tinha como função produzir um arquivo dentro dos padrões pré-estabelecidos para reconhecimento da Plataforma. Este arquivo, pronto em uma pasta do PC, será enviado via interface USB para a Plataforma, que armazenará o mesmo na memória do cartão SD. Esse arquivo pode também ser gravado no cartão SD diretamente pelo PC através de um gravador de cartões SD.

A Figura 40 apresenta uma visão geral do sistema integrado.

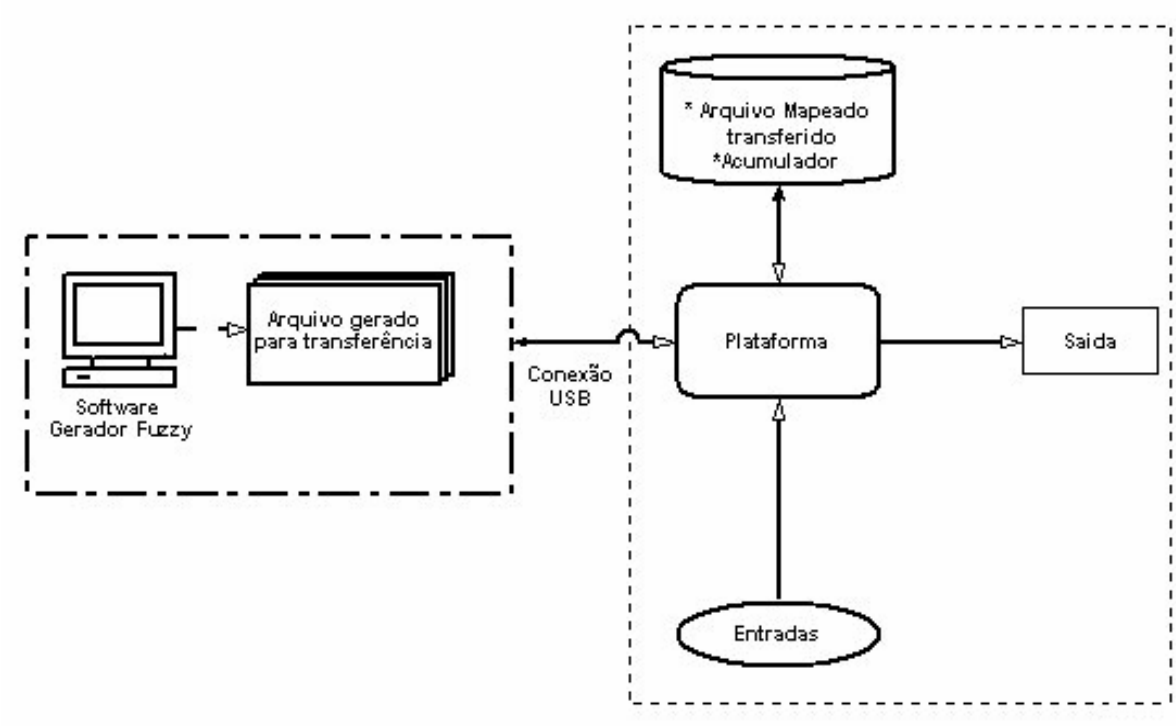


Figura 40 – Sistema integrado ao PC

No capítulo 5 serão abordados o software de geração do arquivo de mapeamento com base no sistema fuzzy bem como o aplicativo que transfere esses dados para a Plataforma.

4.1 – Canais de Entrada e Conversor A/D do UPPC.

A plataforma possui três canais analógicos de entrada em que os sinais amostrados deverão variar de -5 até 5 volts que serão devidamente convertidos em valores digitais binários entre 00000000 e 11111111. A tendência de trabalhar com sinais de baixa tensão já é uma realidade consagrada, pois reduz a potência dissipada nos componentes e o consumo da alimentação.

Essa faixa de tensão entre -5 e +5 volts vai ser deslocada para outra faixa, que é de 0 a 5 volts, por ser esta a faixa correta de entrada do conversor analógico digital do UPPC. O circuito de condicionamento destes valores está ilustrado na Figura 41 A opção de poder trabalhar com entradas negativas objetivou ampliar a atuação de entrada da plataforma.

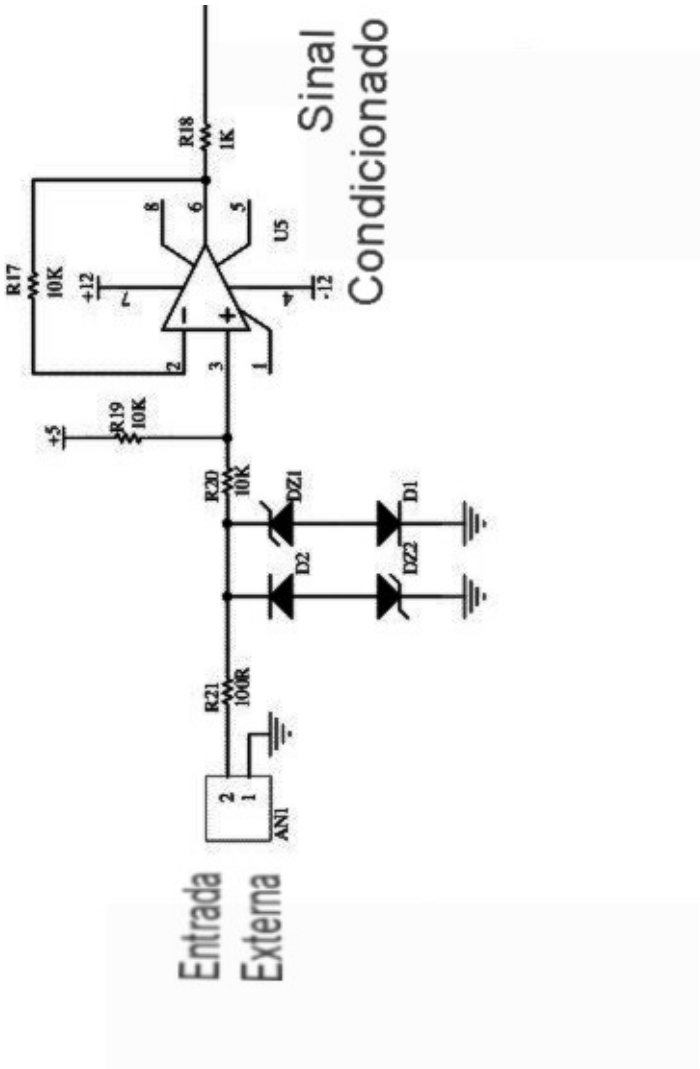


Figura 41 – Condicionador do Sinal de Entrada

A Fórmula abaixo indica como podemos estabelecer o valor analógico que irá para a entrada do conversor A/D do UPPC após o circuito de condicionamento, sendo V_in a entrada na plataforma, e $V_{(uppc)}$ a entrada no microcontrolador:

$$\mathbf{V_{(uppc)} = (V\_in + 5) \times 0,5} \qquad (7)$$

Supondo, como exemplo, uma das entradas com uma tensão de -2,15 Volts. Este valor corresponderia na fórmula ao V_in. O valor transposto para entrada no UPPC será de:

$$\mathbf{V_{(uppc)} = (-2,15+5) \times 0,5 = 1,425\ Volts} \qquad (8)$$

Este valor será convertido internamente pelo conversor A/D do UPPC. O conversor A/D do UPPC tem sua quantização definida, inicialmente, em 8 bits, podendo ser alterado para até 5 bits para testes de quantização que serão feitos em capítulo posterior.

A conversão deste valor analógico para digital vai obedecer a fórmula para uma quantização de 8 bits. O valor decimal estará entre 0 e 255 quantizados, que são os valores mínimos e máximos de conversão de 8 bits. Para conversão do valor decimal em binário nos exemplos demonstrados neste trabalho é usada a técnica de divisão sucessiva por 2. Porém, esta conversão é direta no interior do UPPC.

$$\mathbf{Valor_{(dec)} = Inteiro(51 * V_{(uppc)})} \qquad (9)$$

O valor analógico do exemplo da Eq 01, que é de 1,425 volts, terá seu valor de 73(d). No registrador interno do UPPC este valor será de 01001001b .

4.2 – Estrutura e Endereço de Busca da Memória Mapeada

Foi definido anteriormente como se faz a conversão dos dados analógicos para digital na Plataforma. A estrutura de dados mapeada na memória da Plataforma é endereçada diretamente pela conversão das três entradas da plataforma, depois das devidas conversões no UPPC. A primeira entrada analógica, chamada de IN1, forma um bloco de 8 bits, sendo este valor atribuído aos bits menos significativos do endereço do dado a ser lido. A segunda entrada, chamada de IN2, outro bloco de 8 bits, constitui o bloco intermediário do endereço. Finalmente, a terceira entrada IN3 formará o último bloco de oito bits e mais significativo do endereço global do dado da inferência fuzzy a ser apresentado na saída da Plataforma. Como temos 3 entradas convertidas em 8 bits cada uma, o endereçamento da plataforma terá uma

extensão de 24 bits, perfazendo um total de 16777216 endereços alocados na memória da Plataforma, estando a mesma quantizada para 8 bits.

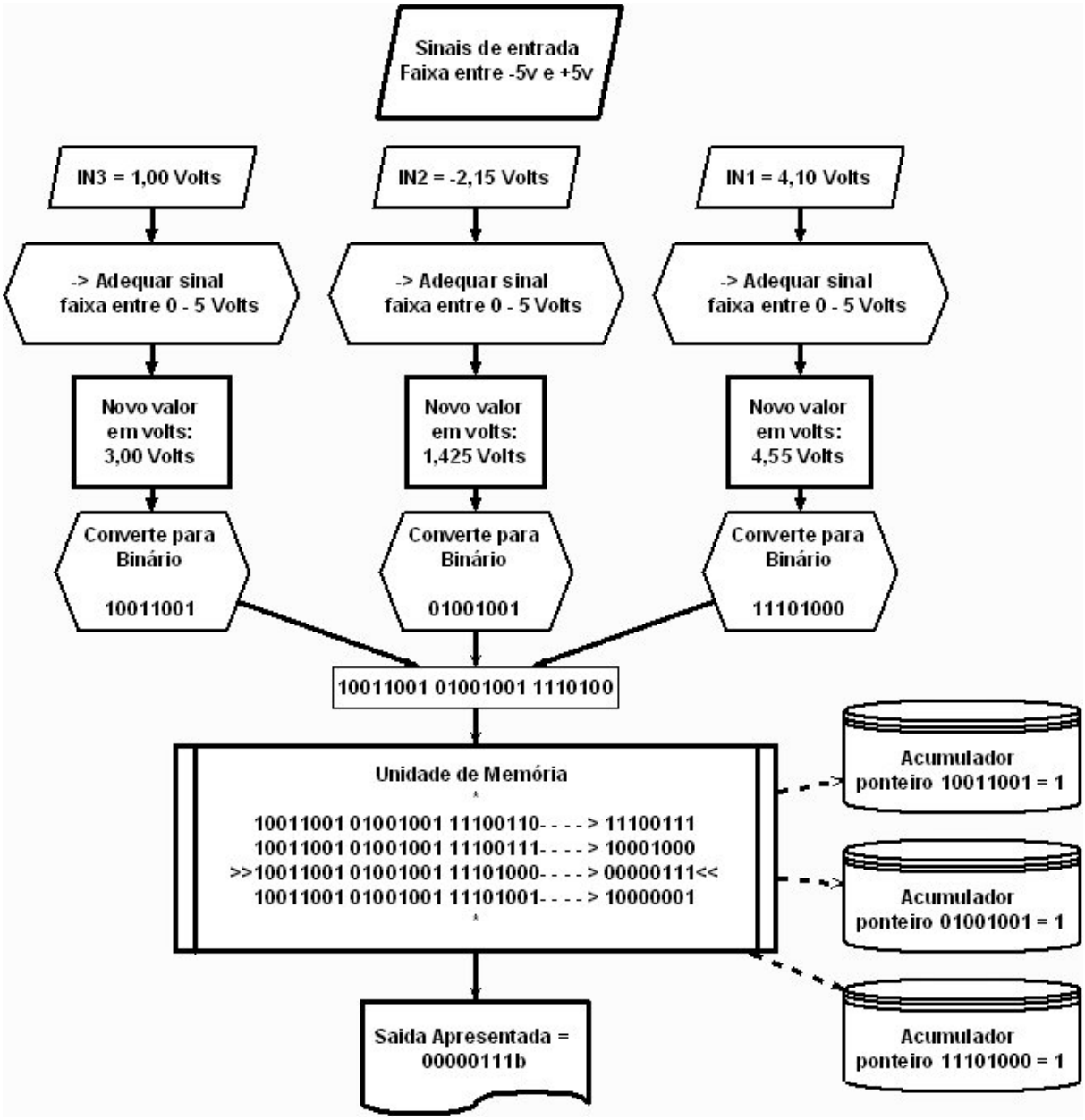


Figura 42 – Formação do Endereço da Memória

A Figura 42 ilustra um exemplo de consulta a um dado da plataforma com as entradas estabelecidas em IN1 = 4,1 volts, IN2 = -2,15 volts e IN3 = 1 volt. De acordo com as devidas conversões, o endereço é formado pela seqüência binária 10011001 01001001 1110100. Este endereço será o ponteiro para se localizar na memória SD da plataforma o dado desejado de 8 bits, que, neste exemplo, é de 00000111b. Como pode ser observada nesta figura, cada uma das combinações binárias de entrada terá um acumulador de uso. Como a entrada binária

11101000 da IN1 for utilizada, o acumulador soma 1 a sua posição de memória, assim como as demais entradas utilizadas. Este processo permitirá uma pesquisa estatística do uso ou não de certas faixas de cada entrada especificamente, que será demonstrado em um tópico à frente.

4.3 – Acumulador de Incidência nas Entradas

Conforme apresentado no início desta dissertação, uma proposta de averiguar o uso ou não de certas faixas de entrada da plataforma está no escopo deste trabalho. A Figura 42 dá uma noção parcial de como funciona esta proposta de acúmulo desses dados imprescindíveis a uma futura análise do comportamento do sistema fuzzy sugerido no mapeamento da plataforma. Para uma compreensão mais detalhada, foram feitas algumas tabelas de uso das entradas da plataforma durante a execução do processo em si de um projeto fictício. Na Tabela 2 temos uma aquisição de valores das entradas IN1, IN2 e IN3, ao longo de um período de tempo, supondo um projeto já definido e todos os seus parâmetros colocados na memória da plataforma.

Tabela 2 – Tabela de Registro de Eventos

Entradas do sistema catalogadas após 15 eventos								
Evento		IN1 (VOLTS)	IN1 DEC		IN2 (VOLTS)	IN2 DEC	IN3 (VOLTS)	IN3 DEC
1		-1	102		0	127	-1	102
2		0	127		0	127	-2	76
3		0	127		1	153	4	229
4		-3,1	48		1	153	4	229
5		2	178		1	153	-2	76
6		2	178		0	127	-1,2	96
7		-1	102		1	153	-1,6	86
8		-1	102		0	127	-1,4	91
9		-3,1	48		2	178	-1,5	89
10		0	127		1	153	-1,5	89
11		4	229		0	127	-2	76
12		-1	102		2	178	-2	76
13		-1	102		0	127	-2,25	70
14		2	178		1	153	-2,36	67
15		2	178		1	153	-2,56	62

Nesta tabela têm-se anotado todas as entradas apresentadas durante 15 mudanças de estado de qualquer uma das variáveis de entrada, que foi chamado de evento. Esses eventos foram tomados em um determinado tempo, suficiente para supor que todas as condições de

funcionamento do sistema já foram atingidas durante esse período. Por exemplo, três dias para a tabela acima.

Analisando apenas a entrada IN1, verifica-se que seu valor se repete em 1, 7, 8, 12 e 13. Esta repetição demonstra que ao longo desse tempo esta incidência com esse valor foi relevante, em comparação as outras catalogadas. A Tabela 3 é apresentada a seguir, aglutinando as incidências que se repetiram nos eventos apresentados, colocando em ordem crescente de valores já transformados em decimais das entradas.

Tabela 3 – Ordenação por nível de entrada dos eventos

Evento	IN1 Dec		Evento	IN2 Dec		Evento	IN3 Dec
4	48		1	127		15	62
9	48		2	127		14	67
1	102		6	127		13	70
7	102		8	127		2	76
8	102		11	127		5	76
12	102		13	127		11	76
13	102		3	153		12	76
2	127		4	153		7	86
3	127		5	153		9	89
10	127		7	153		10	89
5	178		10	153		8	91
6	178		14	153		6	96
14	178		15	153		1	102
15	178		9	178		3	229
11	229		12	178		4	229

Esta tabela ordenada crescentemente pelos valores decimais que foram apresentados ao longo do tempo mostra claramente uma repetibilidade de certos números, que podem ser estudados estatisticamente. Outro fator é a não incidência de certos valores, que também é um motivo de estudo. Seria pelo fato dos sensores de aquisição de dados não estarem em conformidade com o projeto? Ou seria pelo fato do especialista ter informado situações que realmente nunca acontecem? As Tabelas 4, 5 e 6 descrevem os valores que ficaram acumulados internamente no Acumulador de Incidência das Entradas.

Tabela 4 – Valores alocados nos acumuladores de IN1

Acumuladores de incidência em IN1		
Num dec:	Num binario:	Qtd de incidencias
48	00110000	2
102	01100110	5
127	01111111	3
178	10110010	4
229	11100101	1

Tabela 5– Valores alocados nos acumuladores de IN2

Acumuladores de incidência em IN2		
Num dec:	Num binario:	Qtd de incidencias
127	01111111	6
153	10011001	7
178	10110010	2

Tabela 6– Valores alocados nos acumuladores de IN3

Acumuladores de incidência em IN3		
Num dec:	Num binario:	Qtd de incidencias
62	00111110	1
67	01000011	1
70	01000110	1
76	01001100	4
86	01010110	1
89	01011001	2
91	01011011	1
96	01100000	1
102	01100110	4
229	11100101	2

Como esta plataforma possui três entradas de até oito bits de quantização cada uma, existem 256 possibilidades individuais de incidência para cada entrada. Então, para as três entradas, temos 768 possibilidades de incidência, sendo que para cada uma dessas deverá haver um acumulador. Nesta plataforma, este acumulador é uma variável de 8 bits, que tem sua contagem máxima de eventos em 255. Se a contagem de eventos chega a esse valor, esta

variável para de acrescer, pois chegou ao limite máximo, representando uma incidência máxima desse valor de entrada no projeto.

4.4 – Unidade de Processamento e Periféricos Central - UPPC

Esta Unidade de Processamento vai receber as entradas da Plataforma, fazer a conversão A/D internamente, enviar os dados devidos às saídas, acionar o display, ler o teclado, receber ou enviar arquivos via USB para o PC, ativar e armazenar dados na memória auxiliar SD e consultar esses dados.

Existe uma gama muito grande de processadores hoje em dia. Um estudo detalhado das necessidades da Plataforma, da acessibilidade aos meios de programação, o custo e a comunicação com periféricos auxiliares apontou para o microcontrolador PIC18F4550 da MICROCHIP para ser utilizado neste trabalho.

Um microcontrolador engloba, além do processador em si e de pequena quantidade de memória, uma série de periféricos e meios coadjuvantes internos que normalmente estariam em um meio externo, contribuindo com uma otimização de espaço e de custo da plataforma. Por aglutinar várias funções em um dispositivo, as desvantagens do microcontrolador são as limitações desses itens internos, como a pequena memória de armazenamento, velocidade nos periféricos reduzida, além de alguns pinos com funções sobrepostas, impedindo simultaneidade no uso das mesmas em uma mesma aplicação.

Alguns periféricos internos já incorporados neste microcontrolador estão sendo usados e serão abordados em tópicos logo a seguir.

O microcontrolador PIC18F4550, com as características gerais mostrados na Figura 43, será chamado de UPPC - Unidade de Processamento e Periféricos Central em todo escopo deste trabalho.

Este microcontrolador pode ser programado pela sua linguagem de montagem, através da utilização de um montador fornecido pelo fabricante, ou por linguagens de nível mais alto para facilitar a programação de certas funções complexas, como, por exemplo, trabalhar com ponto flutuante ou mesmo fazer uma rotina de divisão com resto.

Quanto ao clock de trabalho, foi utilizado no modo PLL de 40 MHz.

No caso desta plataforma foi utilizada a linguagem C com o compilador da CCS adequado para o microcontrolador em questão.

Features	PIC18F2455	PIC18F2550	PIC18F4455	PIC18F4550
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	24576	32768	24576	32768
Program Memory (Instructions)	12288	16384	12288	16384
Data Memory (Bytes)	2048	2048	2048	2048
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/ Compare/PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Universal Serial Bus (USB) Module	1	1	1	1
Streaming Parallel Port (SPP)	No	No	Yes	Yes
10-Bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Comparators	2	2	2	2
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-pin PDIP 28-pin SOIC	28-pin PDIP 28-pin SOIC	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP

Figura 43 – Funções e periféricos do PIC18F4550

Na Figura 44 temos o encapsulamento do microcontrolador e a respectiva pinagem, com todas as funções abreviadas em cada um dos terminais.

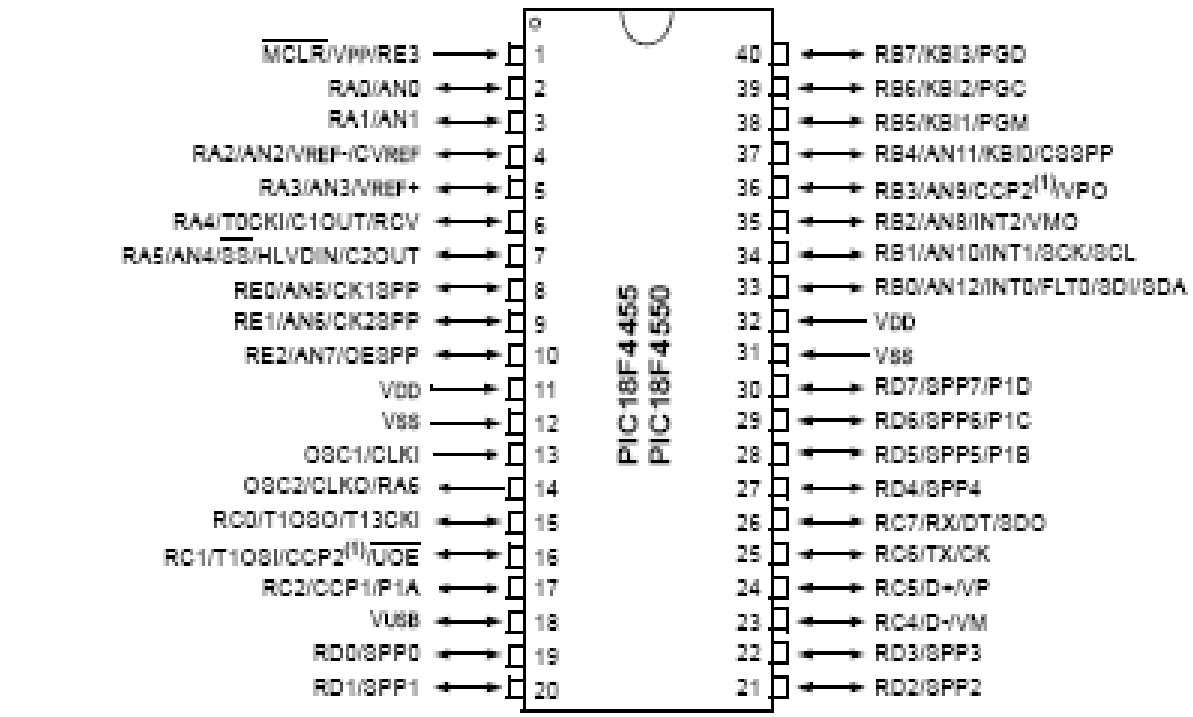


Figura 44 – Pinagem e encapsulamento do PIC18F4550

4.5 – Conversor Analógico-Digital Interno

O conversor Analógico-Digital (A/D) é um dos periféricos internos da UPPC. Ele opera com uma precisão de 8 bits. As três entradas analógicas da Plataforma serão convertidas diretamente na UPPC, mas uma de cada vez, multiplexadas e armazenada internamente em registradores. Os pinos do UPPC que receberão os sinais a serem convertidos são:

> In1 - Pino 2 – porta RA.0 – AN0

> In2 - Pino 3 – porta RA.1 – AN1

> In3 - Pino 4 – porta RA.2 – AN2

> In4 – Pino 5 – porta RA.3 – DIGITAL

> In5 – Pino 7 – porta RA.5 - DIGITAL

Como precisamos de três portas analógicas que serão utilizadas com o conversor A/D interno, temos que configurar a porta A com os três primeiros bits, RA.0, RA.1 e RA.2, sendo entradas analógicas, e os demais podem ficar como digitais. O registrador utilizado para tal fim é o **ADCON1.** Fazendo o registrador ADCON1 = 0000**1100b,** sendo que a configuração dos últimos 4 bits determinam que os três primeiros sinais da Porta A serão analógicos e os demais serão digitais.

Para seleção individual de cada porta a ser usada no conversor A/D interno, temos o registrador ADCON0 que faz este controle de acordo com a Tabela 7:

Tabela 7 – Seleção das Entradas A/D do Microcontrolador

ADCON0 = xx0000xx B – seleciona: In1 - Pino 2 – porta RA.0 – AN0 ADCON0 = xx0001xx B – seleciona: In2 - Pino 3 – porta RA.1 – AN1 ADCON0 = xx0010xx B – seleciona: In3 - Pino 4 – porta RA.2 – AN2

A quantização binária inicial é de oito bits. Porém, para os estudos de sensibilidade propostos nos capítulos 5 e 6, existe a necessidade de uma diminuição de quantização do sistema. Para tal, essa quantização será determinada via software interno e de acordo com alguns parâmetros programados via display. Os níveis binários de conversão podem variar entre 5 e 8 bits na conversão.

4.6 Funcionamento Global da Plataforma

O diagrama de blocos funcional do Mapeador está ilustrado na Figura 45. Esta divisão em blocos define os módulos interdependentes com a UPPC, que faz todo o controle de fluxo e operações de controle.

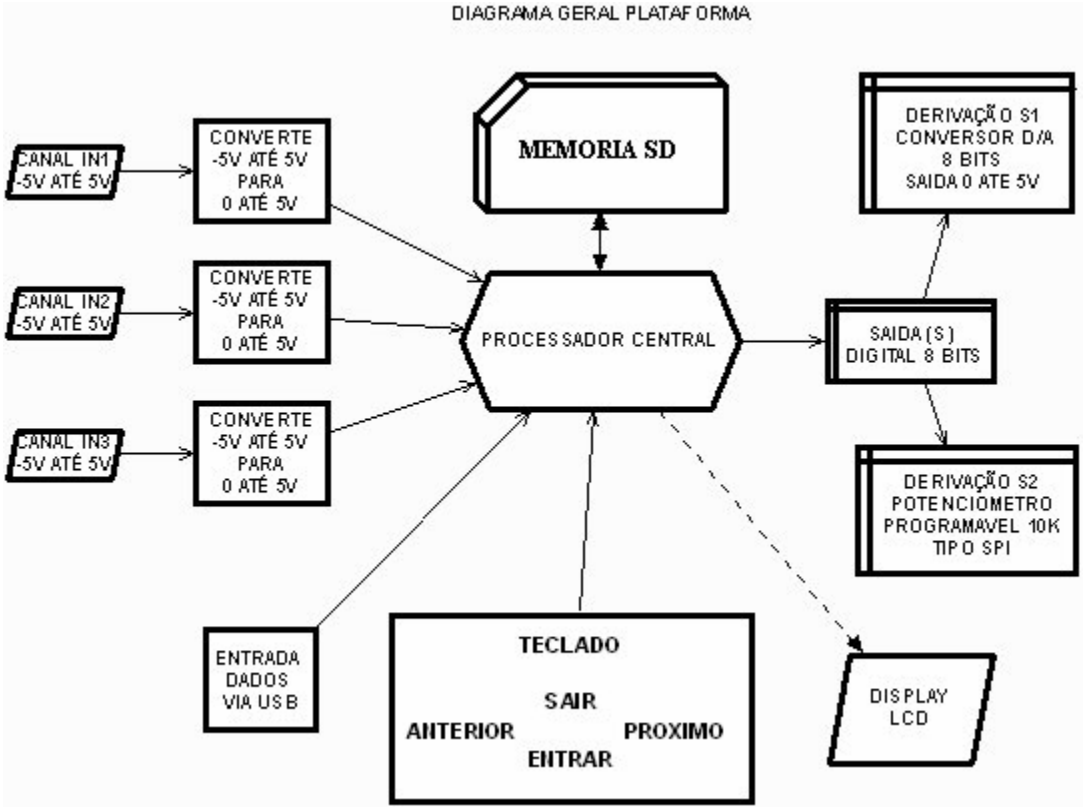


Figura 45 – Esquema de Interconexão dos Módulos da Plataforma

O princípio de funcionamento está fundamentado na carga (*download*) de um arquivo pré-definido em um microcomputador, ou qualquer outro sistema que obedeça aos padrões de formatação estabelecidos, e na gravação do mesmo na memória SD da plataforma. Após esta fase, basta colocar a plataforma em modo de execução. O que se espera é obter uma resposta na saída de acordo com a incidência dos três sinais de entrada na plataforma.

Como existem diversas possibilidades de configurações possíveis implementadas nesta plataforma, foi colocado um teclado para selecionar as funções desejadas e um display de cristal líquido – LCD, que facilita a visualização das mesmas.

4.7 - Funções de execução da Plataforma

Através do teclado é possível selecionar as diversas funções de execução da plataforma. Para ir de uma função para outra, utiliza-se as teclas de Anterior ou Próximo. Quando se deseja acionar a função em questão, basta apertar a tecla Enter. O modo natural do

programa da plataforma é o de execução, isto é, ao sair de qualquer uma das funções, automaticamente a plataforma entra no modo de execução do sistema fuzzy mapeado.

Função LER

Neste modo a plataforma estará preparada para dar início à carga do arquivo de mapeamento advindo do PC, através da conexão USB. Esta opção entra em funcionamento assim que se conecta o cabo a uma entrada USB do PC. A nova tela do display indicará que a plataforma está ESPERANDO O PROGRAMA. Nesta etapa, a plataforma somente poderá sair da rotina de espera retirando-se o cabo USB do PC.

Esta função está diretamente relacionada com o programa de transferência no PC, onde está o arquivo mapeado gerado pelo Matlab. Quando começa a receber o arquivo via USB, a tela do display muda para RECEBENDO CARGA.

Acumulador

Os valores acumulados do uso das entradas estarão arquivados no cartão, e podem ser acessados somente através do leitor de cartão SD conectado em um PC. Não estão acessíveis para leitura direto na plataforma

Função Quantizar IN1

Esta função permite programar a quantização da entrada IN1. Ela é habilitada assim que a plataforma é ligada. O display mostra na primeira linha: “Quantização: IN1”, sendo que o valor vai de 5 até 8. Na segunda linha mostra “5 6 7 8” aguardando a seleção. Se colocar um número diferente ao número que está na primeira linha, e apertar [Enter], o valor será alterado. Se isso acontecer, a plataforma terá novos valores de quantização para IN1.

Função Quantizar IN2

A mesma característica do subitem 4.8.3, porém com a entrada IN2.

Função Quantizar IN3

A mesma característica do subitem 4.8.3, porém com a entrada IN3.

4.8 – Saída de Dados da Plataforma

A saída de dados da plataforma se dá através de uma porta paralela de 8 bits. Esta saída é puramente digital, apresentando os dados instantaneamente de acordo com o dado

advindo da busca na memória SD, que pode ser visualizado através de oito leds na plataforma. Para facilitar o uso desta plataforma em diversas pesquisas, foram também implementadas duas novas saídas: a primeira é a saída S1 da Figura 51, com um conversor D/A de oito bits, utilizando o conversor DA0800, com faixa de saída entre 0 e 5 volts analógicos, aproveitando a saída digital paralela citada anteriormente para a sua entrada e devida conversão. A segunda é a derivação S2 da mesma figura, sendo este hardware um componente que funciona como um potenciômetro programável via comunicação SPI (*Serial Peripheral Interface*). O seu funcionamento detalhado está no subitem 4.9.1. Foi escolhido o modelo de potenciômetro programável MCP41010 (Microchip), que tem como característica principal o valor máximo nominal de 10 Kohms. É um CI de 8 pinos, sendo que três deles fazem a função de potenciômetro propriamente dito, outros três pinos servem de entrada da comunicação serial para programá-lo e dois pinos de alimentação, conforme a Figura 46. Como é um potenciômetro programável via software, inúmeras aplicações podem advir desse hardware, sendo que o mesmo já foi utilizado para configurar o peso em uma implementação de Rede Neural Artificial desenvolvida anteriormente (GARCIA ET AL, 2007).

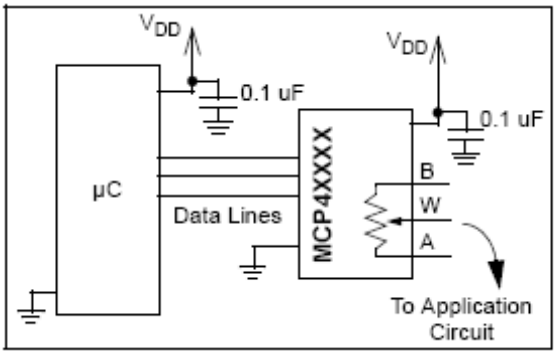


Figura 46 – Esquema de ligação do MCP41010

Os dados de programação entram serialmente através do pino SI, obedecendo ao sincronismo do clock do pino SCK, sendo que a variação de resistência do potenciômetro está correlacionado a um código de oito bits cuja faixa é de 0 e 255. Temos, então, as extremidades PA0 e PB0 valendo 10 Kohms e o *tap* central PW0 variável conforme o valor do código de programação que entra via SPI.

O valor inicial resistivo para o código de programação 00 será de 52 Ohms para o MCP41010 – 10 Kohms, conforme informativo da folha de dados da Microchip relativo a este componente. Para sabermos qual a variação de valores do *tap* central em relação às

extremidades do potenciômetro, temos a fórmula definindo uma regra prática de conversão do código de programação para com os valores apresentados na saída, retirando apenas os valores inteiros do resultado:

$$R_{WA}(D_n) = \frac{(R_{AB})(256 - D_n)}{256} + R_W \quad (10)$$

$$R_{WB}(D_n) = \frac{(R_{AB})(D_n)}{256} + R_W \quad (11)$$

RWA(Dn) = valor da resistência entre o tape W e a extremidade RA;

RWB(Dn) = valor da resistência entre o tape W e a extremidade RA;

RAB(Dn) = valor da resistência entre o tape RA e RB;

Dn = Código em decimal que entrou no chip para programação do mesmo

RW = Valor residual da resistência inicial, neste caso, de 52 ohms

Como primeiro exemplo, temos um valor hipotético que foi fornecido via SPI pela UPPC de 00111011(binário) = 3B (hex) = 59 (decimal). Então, conforme a fórmula, teremos:

RWA(59) = {[10000 x (256 – 59)] / 256} + 52 = 7747 Ohms

RWB(59) = [(10000 x 59) / 256] + 52 = 2356 Ohms

4.9 – Protótipo da Plataforma

Por se tratar de um protótipo de pesquisa, com algumas funções acrescidas ao longo de deste trabalho de dissertação, todo o desenvolvimento foi baseado na concatenação de módulos independentes, conforme ilustrado na Figura 47.

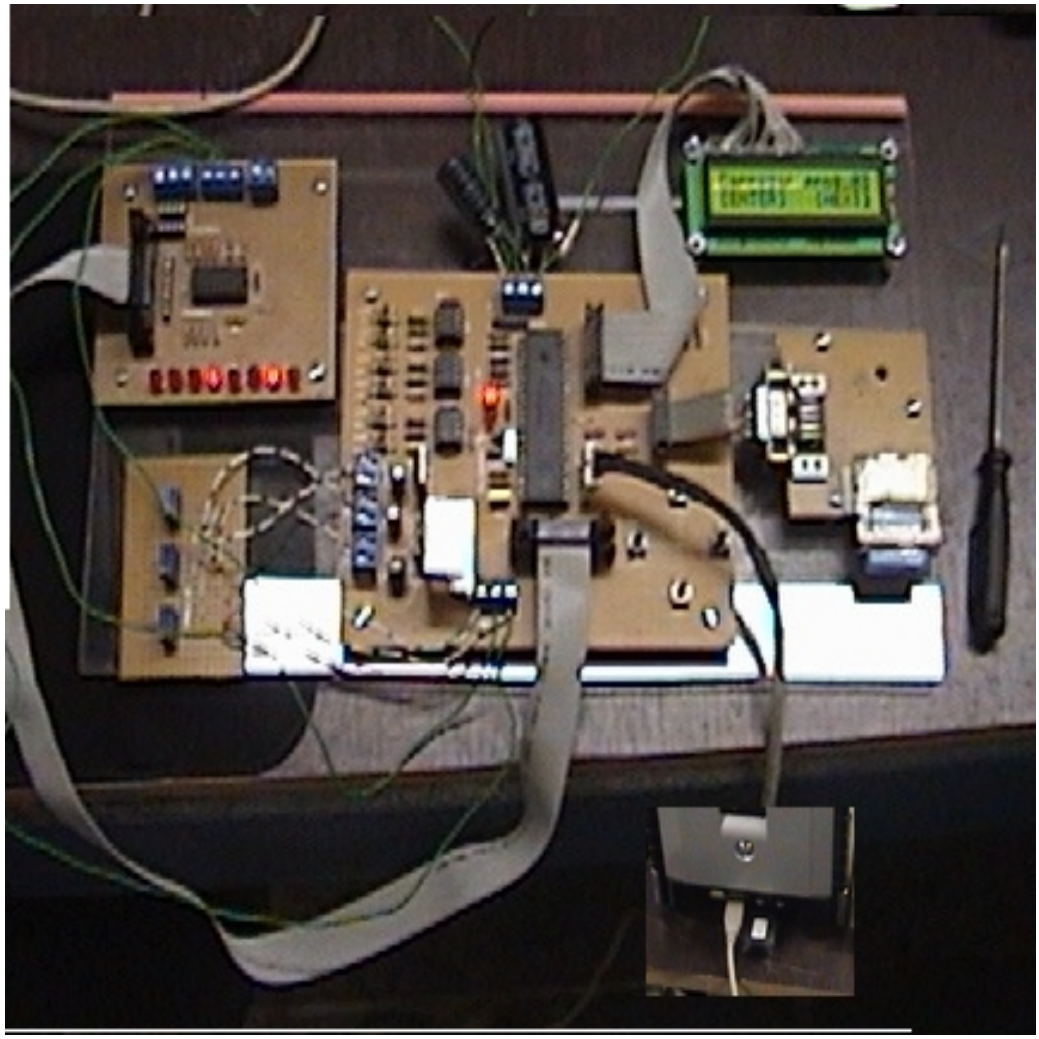


Figura 47 – Protótipo da Plataforma

CAPÍTULO 5
IMPLEMENTAÇÃO DO SOFTWARE NO PC

Todo o procedimento de modelagem do sistema fuzzy foi desenvolvido no ambiente do Matlab. No Matlab há uma ferramenta que é chamada "*Fuzzy Logic Toolbox*". Nela pode ser simulado o sistema fuzzy com as devidas entradas, regras, defuzificador e saída. Nesta etapa do trabalho, foi muito importante verificar as diversas possibilidades de entradas quanto a sua quantização binária, bem como os diferentes valores correspondentes em sua saída. Em um nível binário de cinco bits por entrada, o desenvolvimento do mapa completo ao simular o sistema fuzzy via software foi feito em um tempo razoável, aproximadamente doze minutos. Porém, ao se executar o processo de mapeamento com uma quantização binária de oito bits por entrada, o processo demorou em torno de cinco horas.

O editor FIS (*Fuzzy Inference System*) deste toolbox, após a devida inferência fuzzy implementada, vai fornecer uma função com o nome definido pelo usuário que obedecerá a todos os parâmetros nela contida quanto ao sistema fuzzy proposto. Porém, para se determinar todo o mapeamento das entradas devemos colocar passo a passo às devidas entradas e armazenar as respectivas saídas em um arquivo. Este processo seria completamente inviável se fosse executado manualmente, pois em um sistema que utiliza a quantização de oito bits por entrada, a quantidade de dados a ser pesquisada por essa função será de 2^{24} valores distintos, dando um total de 16777216 números armazenados.

O procedimento de estruturação do mapeador, quanto a formatar e preencher o arquivo no PC com os dados a serem transferidos para a plataforma, obedeceu às operações concatenadas em blocos, conforme ilustrado na Figura 48.

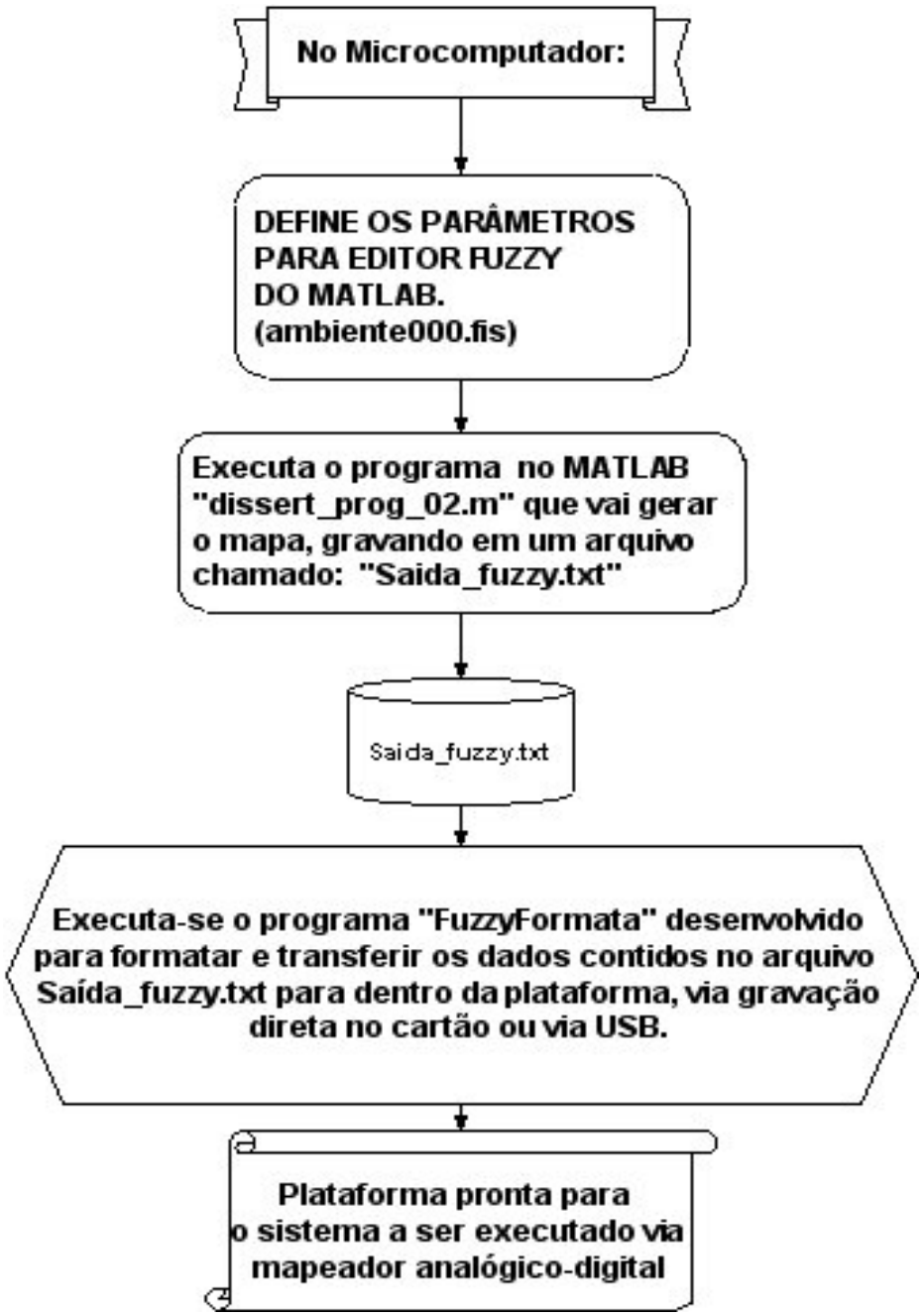


Figura 48 – Blocos dos Softwares de Transferência.

5.1 – Parametrizando o Toolbox Fuzzy no MATLAB

Esta ferramenta do Matlab fornece condições para o usuário desenvolver toda a inferência fuzzy via software. Além disso, o toolbox tem uma Interface Gráfica com o Usuário, o que o torna mais fácil de usar. Tem uma aparência consistente, com controles intuitivos como: saída, caixas de listagens, entrada de parâmetros e muitos outros diretamente na aplicação gráfica (CHAPMAN, 2003).

Com o intuito de esclarecer como foram obtidos os resultados nas pesquisas, se faz necessário demonstrar o uso básico deste toolbox do Matlab, nas funções puramente dedicadas no desenvolvimento deste trabalho.

Em um sistema de inferência fuzzy, os nomes das variáveis, a faixa de entrada dos dados, bem como as regras e o tipo de defuzificador devem estar pré-definidos, independente do procedimento selecionado para implementação. Como parâmetros desta dissertação, temos uma plataforma mapeada com três entradas analógicas que serão convertidas em um sinal digital de até oito bits cada, tendo uma saída com um sinal digital de oito bits. Como o mapeador interno da plataforma está com seu endereçamento diretamente relacionado com o conjunto de bits das entradas de forma digital, o que interessa, para fins de formatar esse mapeamento, são apenas os dados digitais, tanto na entrada como na saída, conforme ilustrado na Figura 49.

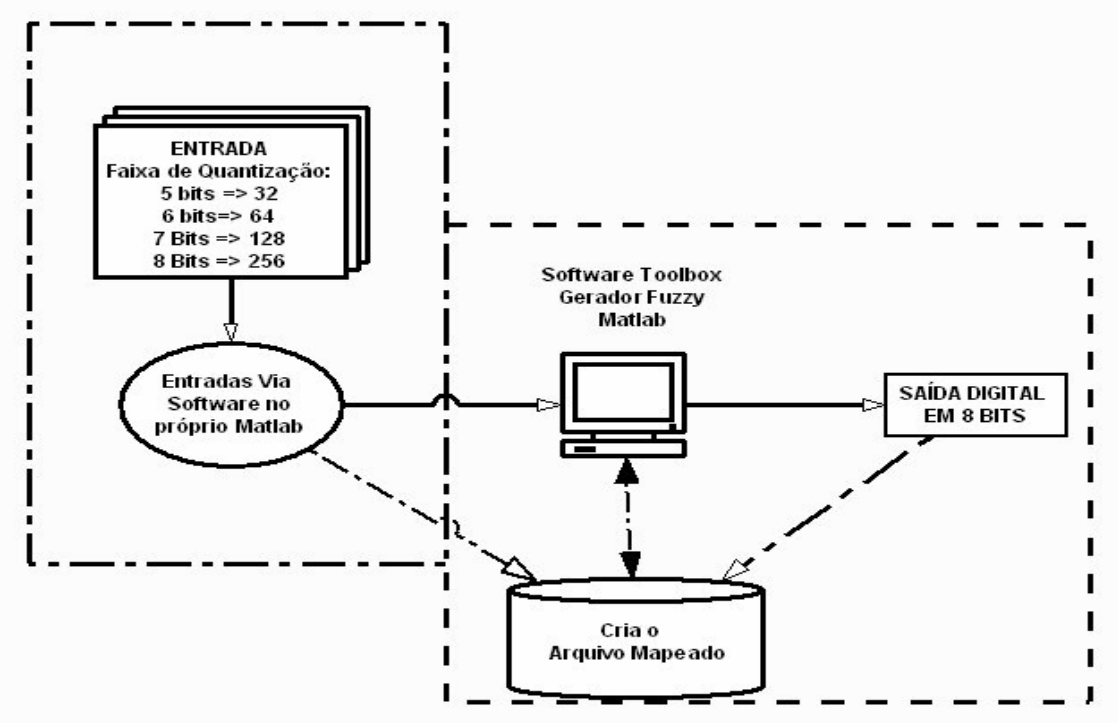


Figura 49 – Bloco Fuzzy no Matlab

Conforme ilustrado, temos uma quantização binária de cinco até oito bits por variável de entrada. Esta quantização vai definir o tamanho final do mapeador. Esses parâmetros de quantização serão utilizados tanto no software do mapeador quanto na formação do arquivo de mapeamento no Matlab.

Para compor o mapa digital a ser transferido para plataforma, foi concebido um sistema com ajuste de velocidade de um exaustor com inferência fuzzy através de três entradas analógicas de temperatura, quantizadas em 8 bits cada uma.

IN1-Temp-LadoA (Sensor de temperatura em um extremo do ambiente)

IN2-Temp-LadoB (Sensor de temperatura no outro extremo do ambiente)

IN3-Temp-mediano (Sensor de temperatura no meio do ambiente)

O modelo Mamdani foi escolhido e o cálculo de defuzificação é baseado no método do centróide. O nome do sistema gerado foi “Ambiente000an”. A Figura 50 ilustra a Tela Gráfica do sistema Fuzzy analógico proposto.

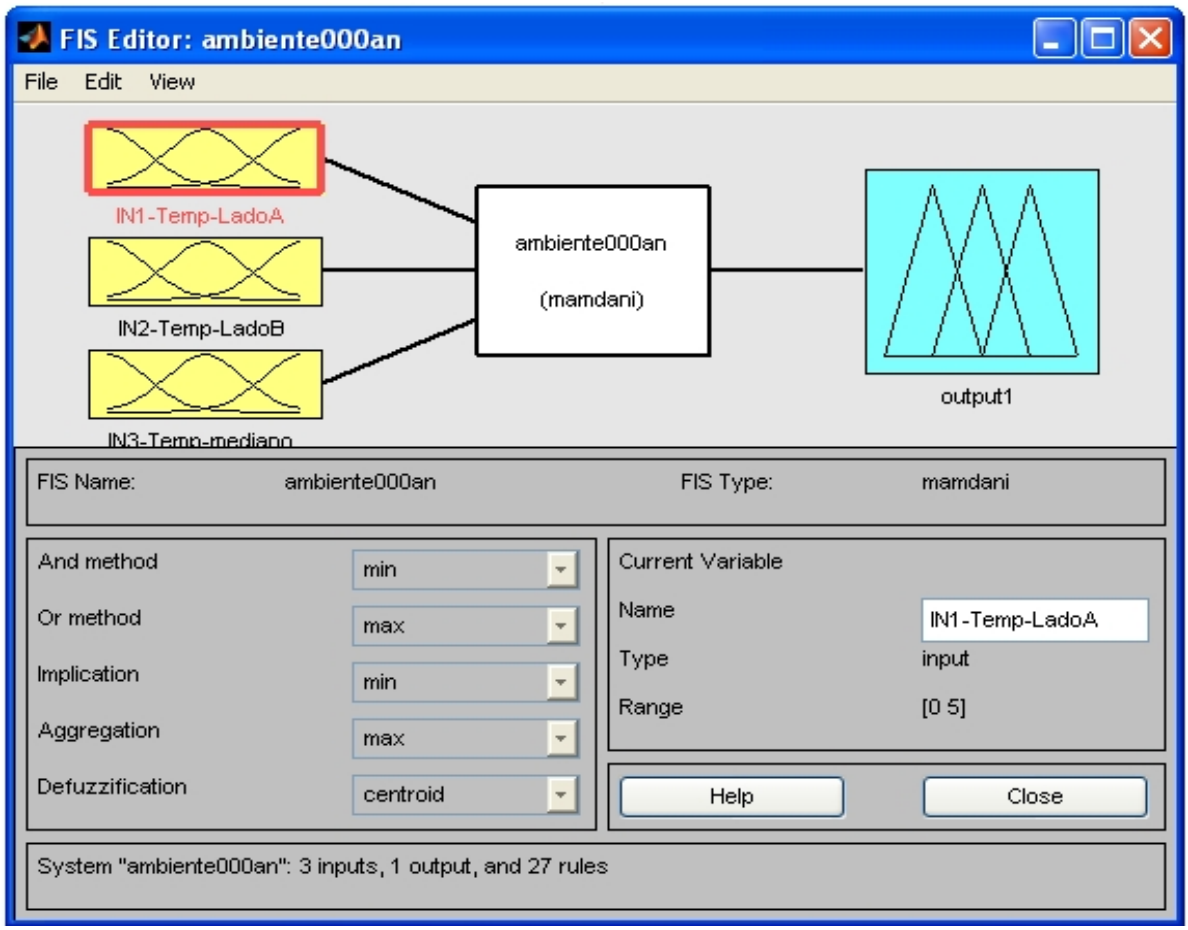


Figura 50 – Tela Inicial do “Toolbox Fuzzy” no Matlab

As entradas analógicas nas entradas da plataforma estarão na faixa entre -5 e + 5 volts. Após a conversão digital, cada valor decimal digital corresponderá a um valor em volts. De um valor decimal para o seguinte ou para o anterior existe uma diferença fixa em volts. Esta diferença é devida à quantização. Quanto menor este valor, melhor a conversão analógica -

digital. Em uma quantização de 8 bits, teremos a faixa de -5 a 5 volts transposta para uma faixa de 0 a 255 decimal. O valor em volts referente a faixa de quantização é de:

10,0 volts / 255 = 0,0392 Volts

Esta variação está relacionada ao erro de quantização, que conseqüentemente vai provocar um erro na saída do sistema fuzzy quantizado em relação ao sistema fuzzy analógico.

A curva de pertinência da entrada IN1 está ilustrada na Figura 51.

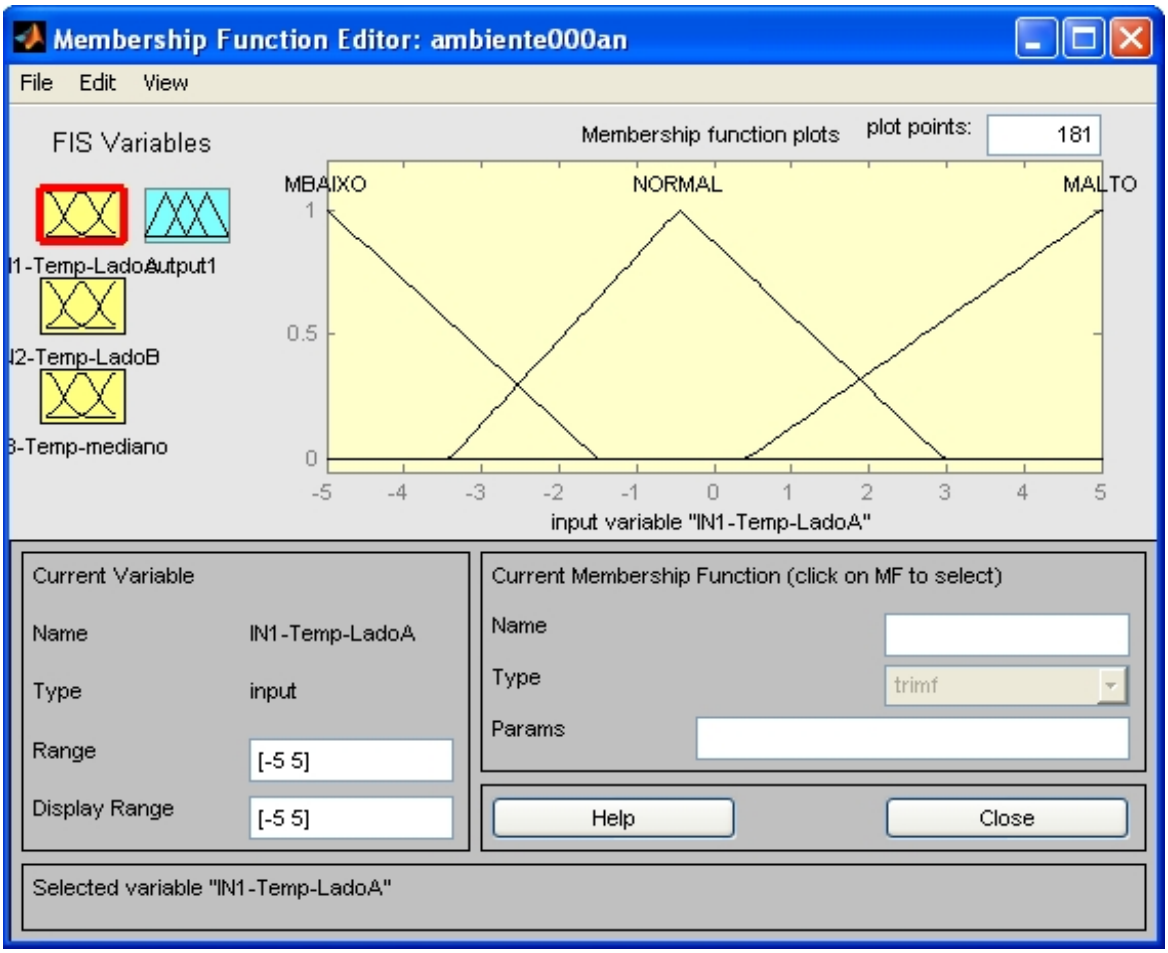


Figura 51 – Tela Gráfica entrada IN1 “Toolbox Fuzzy” no Matlab

A saída fuzzy analógica será sempre entre 0 e 5 volts, pois representa a faixa de valores fornecida pelo conversor D/A da plataforma. Porém, esta faixa de saída, após quantizada, também estará com níveis de saída com saltos de 0,0196 Volts para cada valor adjacente decimal, que pode provocar erros em relação aos valores calculados sem a quantização. A Figura 52 ilustra a tela gráfica do Matlab nesta saída fuzzy.

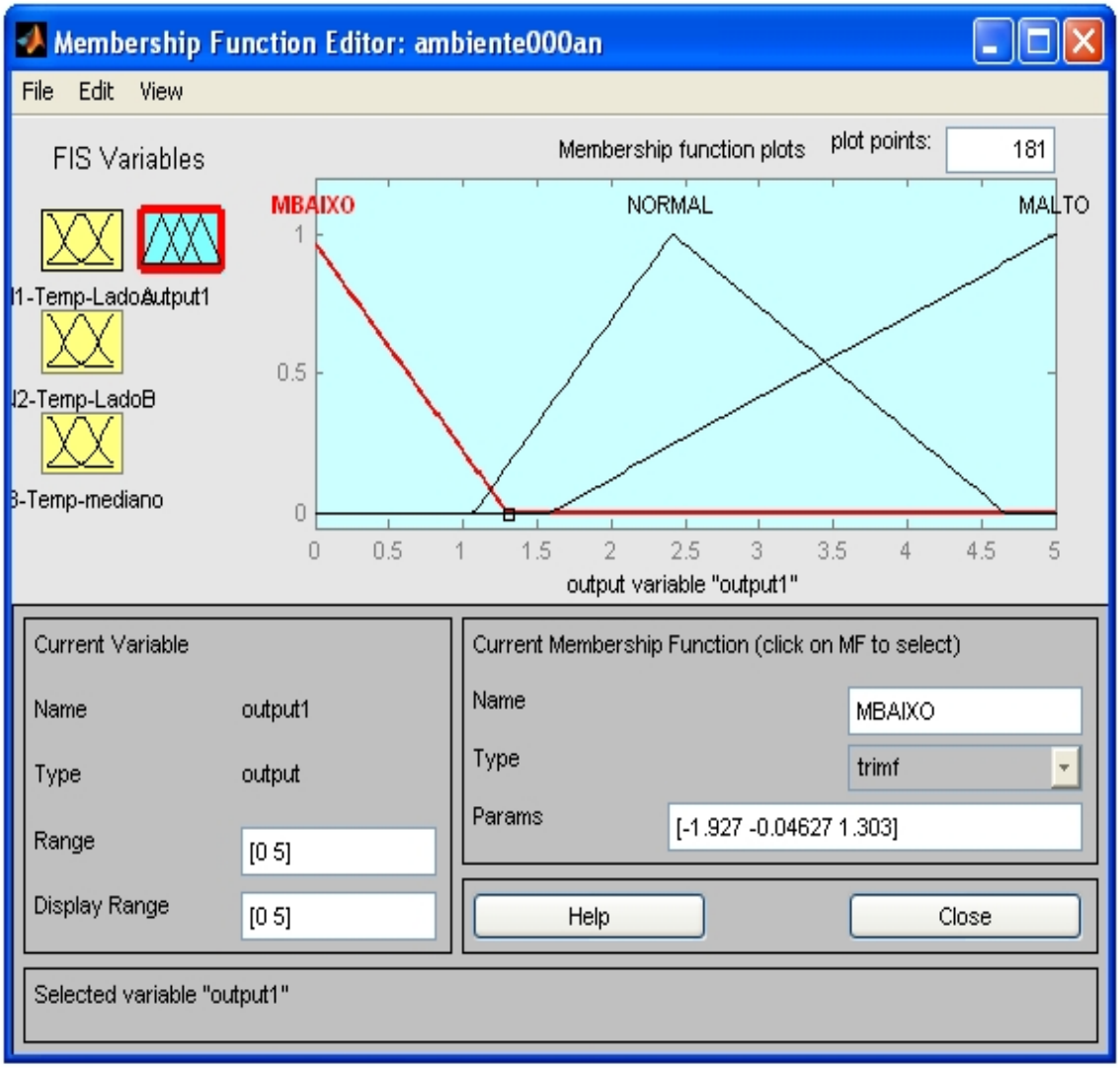


Figura 52 – Tela Gráfica saída analógica do “Toolbox Fuzzy” no Matlab

Após as entradas e saídas terem seus parâmetros estabelecidos, faz-se necessário compor as regras de atuação no sistema apresentado. Foram criadas as seguintes regras, conforme ilustrado na Figura 53.

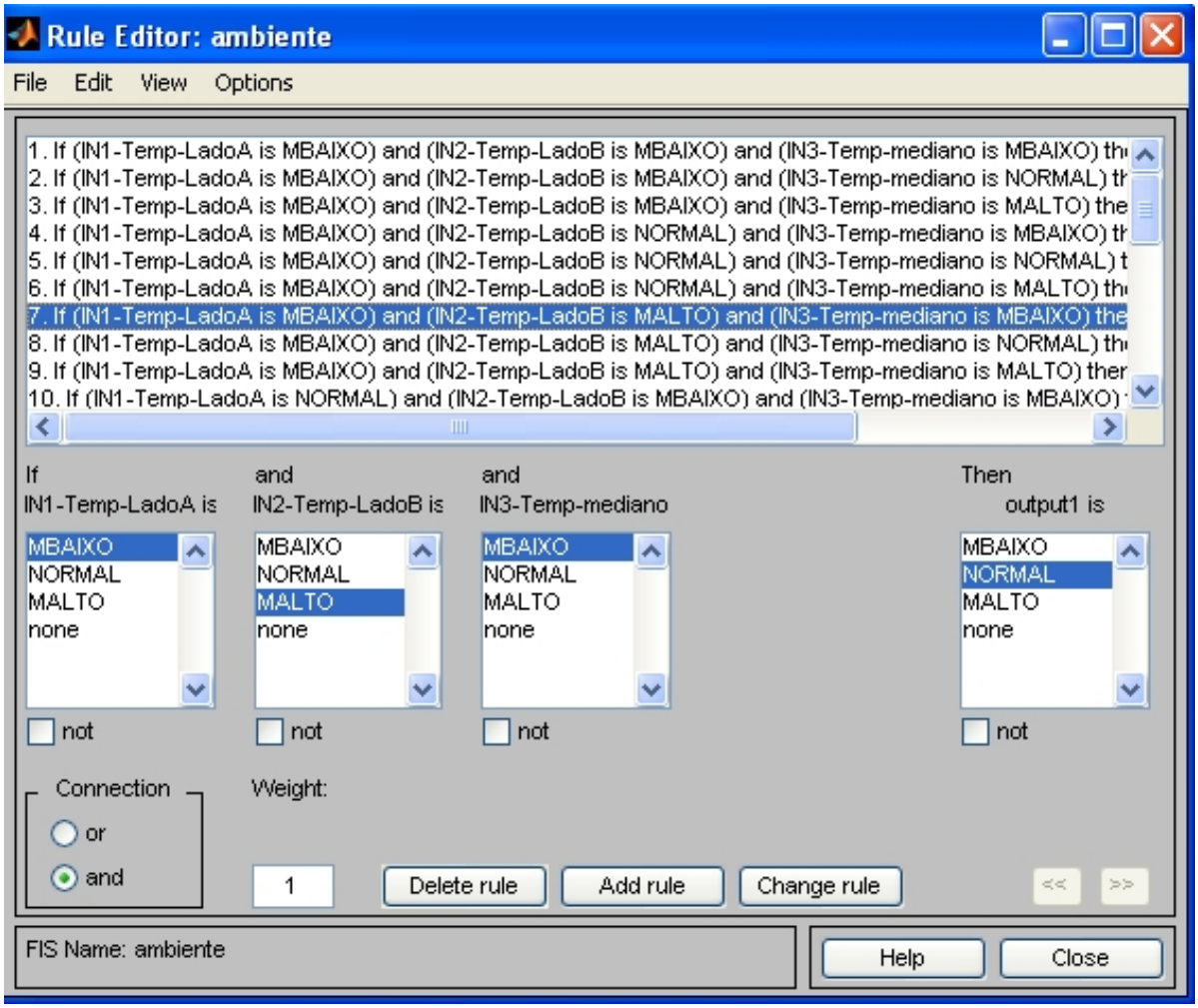


Figura 53 – Tela Gráfica das regras (Rules) do “Toolbox Fuzzy” no Matlab

Após a criação das regras, todo este sistema foi gravado no arquivo chamado ambiente000an.fis. Para gerar o arquivo de mapeamento desejado para transferência, as entradas serão quantizadas nos valores correspondentes a 8 bits, ficando a faixa de entrada entre 0 e 255 decimais com equivalência direta entre os -5 e 5 volts de entrada.

5.2 – Gerando o Arquivo de Mapeamento

Através de um arquivo executável no Matlab, o dissert_prog_02.m, faz-se a escolha inicialmente da quantização desejada para cada entrada analógica. O programa cria automaticamente uma matriz que vai conter todas as inferências calculadas através da função ambiente.fis anteriormente descrita. Depois de gerar esta matriz, a mesma é exportada para um arquivo formatado em um padrão para ser utilizado em um outro software de transferência entre PC e Plataforma via USB, ou gravação direta no PC. A metodologia funcional é

demonstrada a seguir. Primeiramente coloca-se o nome do programa na janela de comandos do Matlab:

>> dissert_prog_02

Após dar início à execução, o programa apresenta os dados da função fuzzy e pede que insira a quantização estabelecida anteriormente. No caso, foi de 8 bits por entrada.

USANDO PARÂMETROS NO ARQUIVO AMBIENTE000AN.FIS

a =

name: ‘ambiente’

type: ‘mamdani’

andMethod: ‘min’

orMethod: ‘max’

defuzzMethod: ‘centroide’

impMethod: ‘min’

aggMethod: ‘max’

input: [1x3 struct]

output: [1x1 struct]

rule: [1x27 struct]

Entre valor quantizado IN1 (LSB) variável: 8

Entre valor quantizado IN2 variável: 8

Entre valor quantizado IN3 (MSB) variável: 8

Faixa de IN1 = 32 , na proporção entre -5 até +5 Volts

Faixa de IN2 = 32 , na proporção entre -5 até +5 Volts

Faixa de IN3 = 32 , na proporção entre -5 até +5 Volts

Qtd total de dados no futuro arquivo mapeado = 32768

AGUARDE FINAL DA EXECUÇÃO!!!!!

Atenção:

Verifique o arquivo saida_fuzzy.txt onde estão os dados Mapeados

FIM DA EXECUÇÃO DO PROGRAMA

A formatação do arquivo “saida_fuzzy.txt” está de acordo com a ordem crescente das variáveis de entrada do sistema. Na primeira linha, à esquerda, está o dado correspondente à entrada IN3 = 0000000 , IN2 = 0000000 e IN1 = 0000000. Após a vírgula, na mesma linha,

seguindo da esquerda para direita, o dado do endereço correspondente à IN3 = 0000000 , IN2 = 0000000 e IN1 = 0000001.

Para exemplificar, na Tabela 8 temos a primeira página do arquivo “saida_fuzzy.txt ” com sua devida formatação, destacando os seis primeiros dados neste arquivo e seus respectivos endereços no mapa.

Tabela 8 – Parte do arquivo mapeado em ordem crescente

18,18,18,18,20,20,20,20,20,23,23,23,26,23,20,20,18,18,20,20,23,26,26,23,23,23,20,20,20,18,18,18,18,18,18,18,18,20,20,20,20,23,23,23,26,23,20,20,18,18,20,20,23,26,26,23,23,23,20,20,20,18,18,18,18,18,18,18,18,20,20,20,20,23,23,23,26,23,20,20,18,18,20,20,23,26,26,23,23,23,20,20,20,20,18,18,18,18,18,18,18,18,20,20,20,20,23,23,23,26,23,20,20,18,18,20,20,23,26,26,23,23,23,20,20,20,18,18,18,18,20,20,20,20,20,20,20,20,23,23,23,26,23,20,20,20,20,20,20,23,26,26,23,23,23,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,23,23,23,26,23,20,20,20,20,20,20,23,26,26,23,23,23,20,20,20,20,20,20,20,23,23,23,23,23,23,23,23,23,23,26,23,23,23,23,23,23,23,23,23,23,26,26,23,23,23,23,23,23,23,23,23,23,23,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26,28,28,28,28,28,28,28,28,28,28,28,28,28,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,28,28,28,28,28,28,28,28,28,28,28,28,28,28,48,84,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,99,28,28,28,28,28,28,28,28,28,28,28,28,28,28,43,69,94,107,115,115,115,115,115,117,130,130,130,130,130,130,130,130,130,26,26,26,26,26,26,26,26,26,26,26,26,36,48,76,92,102,110,112,115,117,125,133,138,138,138,138,138,138,138,138,23,23,23,23,23,23,23,23,23,23,23,26,23,31,41,59,82,94,102,107,112,120,127,133,138,138,138,138,138,138,138,138,20,20,20,20,20,20,20,20,23,23,23,26,23,28,36,48,74,87,97,105,112,120,127,133,138,138,138,138,138,138,138,138,138,20,20,20,20,20,20,20,20,23,23,23,26,23,28,36,46,69,82,94,105,112,120,127,133,138,138,138,138,138,138,138,138,138,18,18,18,18,20,20,20,20,23,23,23,26,23,28,36,43,64,82,94,105,112,120,127,133,138,138,138,138,135,135,135,135			
No arquivo, da esquerda para direita têm-se os dados na ordem dos endereços	IN3	IN2	IN1
18	00000000	00000000	00000000
18	00000000	00000000	00000001
18	00000000	00000000	00000010
18	00000000	00000000	00000011
20	00000000	00000000	00000100
20	00000000	00000000	00000101

As repetições numéricas na Tabela 8 sugerem que possa ter algum tipo de compactação de dados para que a transferência seja em um tempo reduzido.

5.3 – Programa de Transferência para a Plataforma.

O arquivo de mapeamento deverá agora ser transferido para a Plataforma. A plataforma sempre irá consultar os dados na ordem crescente da memória do cartão, conforme a Tabela 8 anteriormente apresentada.

Com a finalidade de garantir que o mapa tenha seu primeiro endereço exatamente no início do cartão, um programa de transferência foi criado com esta função específica. A via de transferência pode ser a comunicação USB ou copiando diretamente os dados do PC para o cartão através de um gravador de cartão SD conectado ao PC.

Para habilitar o uso da transferência pela interface USB, um *driver* fornecido pela CCS teve que ser instalado no PC. Este *driver* cria a porta COM4 que é a saída USB do PC que comunica com a entrada USB da plataforma.

O programa, feito com interface visual gráfica, foi desenvolvido em Delphy e tem como tela inicial a apresentada na Figura 54.

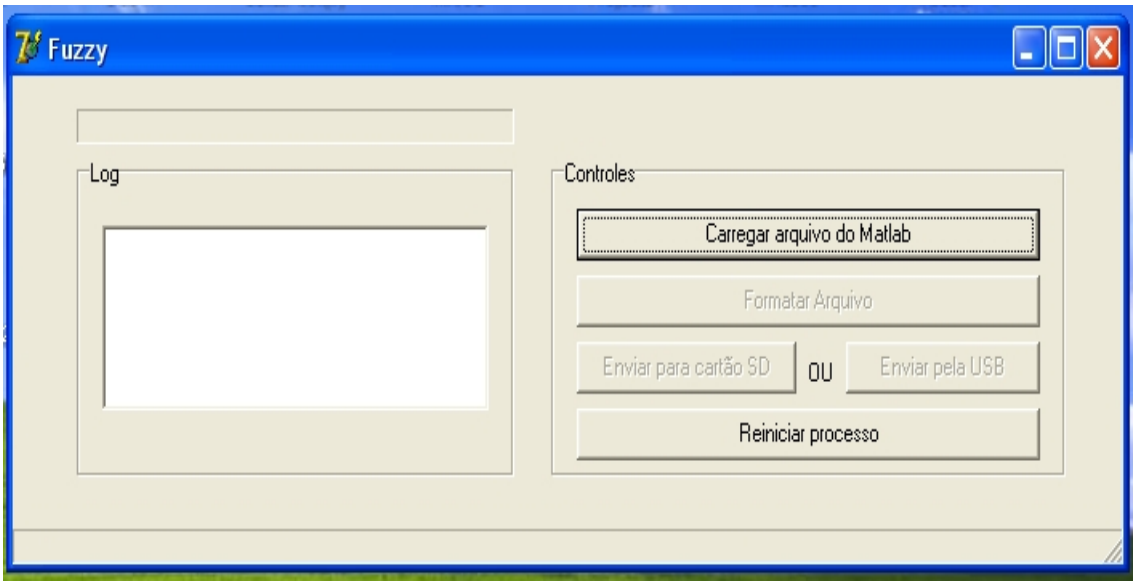


Figura 54 – Tela Programa de transferência de Dados pelo PC

Uma análise da quantização pretendida de bits nas entradas, ao se converter um sistema analógico em digital, é o passo inicial para verificar se é viável ou não esta implementação. Adicionalmente, pode-se fazer um estudo de uma possível redução do número de bits de quantização nas mesmas.

Esta análise é feita sempre tomando por base o sistema fuzzy embrionário como padrão. No exemplo do tópico anterior, o sistema proposto a ser digitalizado é de um sistema

com três entradas analógicas entre 0 e 5 volts e com saída entre 0 e 5 volts. A idéia é digitalizar as entradas com 8 bits e a saída também com 8 bits, para configuração do mapeador.

Para ter o estudo das quantizações, as medidas dos erros máximos e médios permitidos serão os parâmetros de análise. Deseja-se determinar qual é o nível mínimo de quantização por entrada capaz de gerar um sistema final com erro aceitável, conforme a especificação do projeto.

5.4 – Acumulador de uso de Entradas na Memória do Cartão SD

Uma outra característica implementada na plataforma que visa estudos futuros é a possibilidade de geração dos acumuladores de uso de entradas. A plataforma pode gravar, opcionalmente, um somatório de incidências de ocorrência de valores em entrada. Quando um valor de entrada é apresentado, é somado 1 a um registrador referente a esta entrada. Esses dados são gravados em um bloco único de 768 valores, representando os 256 valores diferentes por entrada INx, considerando a quantização máxima permitida de 8 bits. O início do espaço alocado no cartão SD para este fim é estabelecido após o último endereço dos 16 Mbytes iniciais do cartão.

Como exemplo, com uma quantização de 5 bits por entrada, para o bloco da entrada IN1, que vai conter 256 valores, somente os 32 primeiros valores serão válidos. Os outros 224 valores serão nulos. Ao começar o segundo bloco com mais 256 valores, que pertence a entrada IN2, novamente os 32 primeiros valores são válidos, descartando o restante. Assim sucessivamente para a entrada IN3. Esses dados podem ser úteis ao projetista, para uma possível otimização, pois fornecem uma supervisão de uso das entradas do sistema,. Na Tabela 9 temos um exemplo de arquivo.

Tabela 9 – Exemplo do arquivo Acumulador do uso de entradas

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
IN1 5bits	1	0	0	0	50	80	255	255	255	255	14	0	0	0	0	56	56
	2	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IN2 5bits	1	255	255	255	255	180	180	180	180	100	100	100	101	101	87	87	87
	2	251	251	200	200	200	210	210	210	220	200	100	80	80	97	97	97
	....	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IN3 5bits	1	190	189	178	178	60	255	254	30	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAPÍTULO 6
UM SISTEMA FUZZY COM O ESTUDO DA QUANTIZAÇÃO

O caso a ser estudado é o ajuste de velocidade de um exaustor com sensores de temperatura nas entradas, com uma faixa analógica prevista em cada entrada entre -5 e +5 volts, e uma saída de 0 a 5 volts que vai controlar a velocidade de um exaustor. Esta saída tem uma tolerância de +/- 15 % em relação à saída fuzzy, que é o controle de velocidade do exaustor. Este erro foi determinado em função do tipo de atuador a ser controlado, que é um exaustor para refriamento.

Primeiramente foi feito o teste na plataforma tomando os erros de medição entre o sistema quantizado no Matlab e o sistema inserido no mapeador digital.

Em seguida foram feitas pesquisas no Matlab para analisar a implementação com diferentes níveis de quantizações nas entradas. O objetivo era verificar as possibilidades de redução do número de bits para a especificação deste sistema.

Inicialmente foi gerado no Matlab o sistema fuzzy analógico desejado, com as especificações das funções de pertinências e das regras, para depois quantizar este sistema em 8 bits e transferir os dados para a plataforma A Figura 55 apresenta a janela genérica do sistema apresentado pelo toolbox do Matlab.

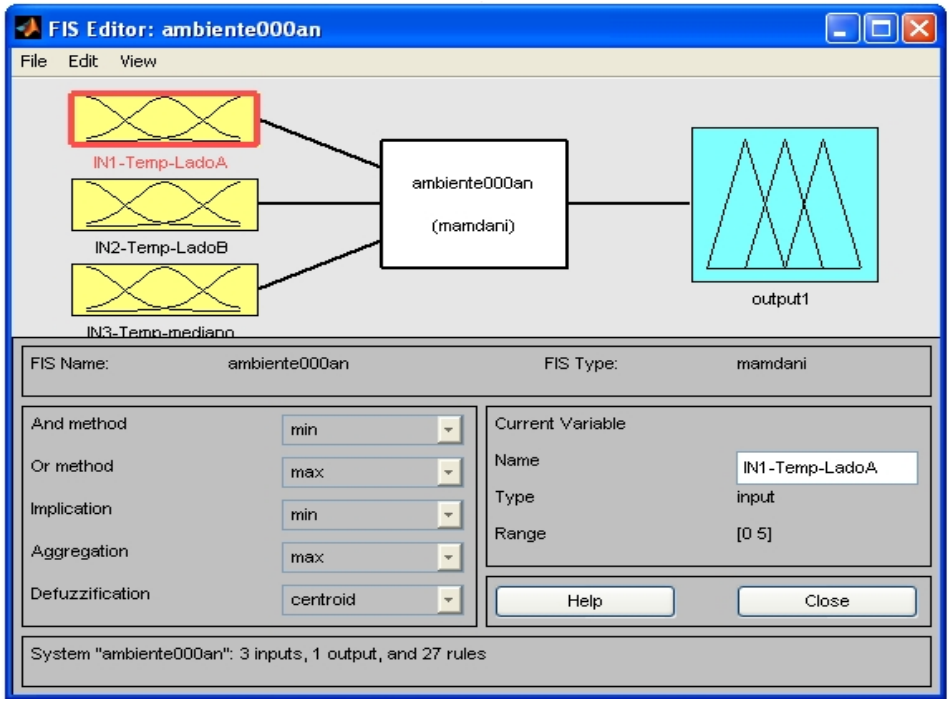


Figura 55 – Janela inicial Toolbox do Controlador Velocidade “ambiente000an.fis”

O sensor 1, colocado no lado A de um ambiente hipotético, foi conectado na entrada IN1 da plataforma. O sensor 2, colocado no lado B, foi conectado na entrada IN2 da plataforma. Finalmente o sensor 3, colocado em uma posição intermediária, foi conectado na entrada IN3, sendo este último com as curvas de baixo, médio e alto em níveis de coincidência maiores em relação às funções de pertinências das outras duas entradas.

Este sistema proposto foi nomeado de ambiente000an.fis, com modelo Mamdani e método de defuzificação do tipo centróide. Possui três entradas e uma saída, sendo cada entrada com quantização máxima de 8 bits, provendo uma faixa de valores entre 0 e 255 decimais para cada entrada. A saída tem uma quantização de 8 bits, e também possui uma faixa entre 0 e 255 decimais. As Figuras 56, 57 e 58 ilustram as devidas entradas analógicas no Toolbox do Matlab, e a Figura 59 ilustra a saída deste sistema no Matlab.

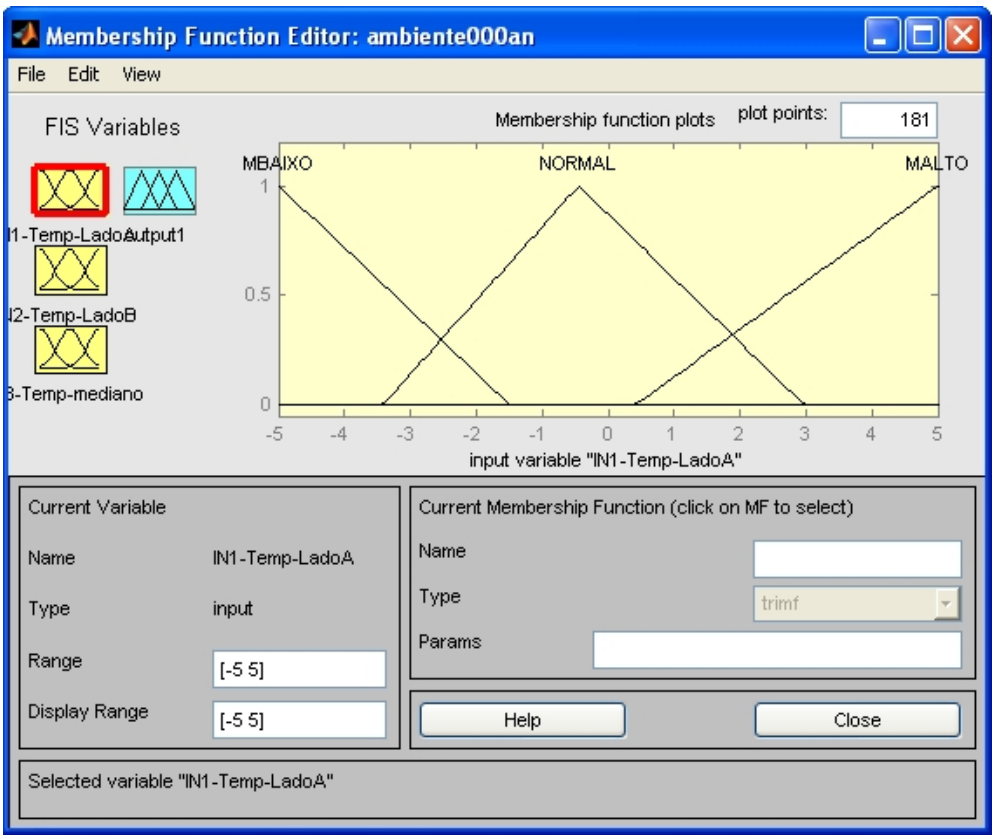


Figura 56 – Entrada IN1 Analógica

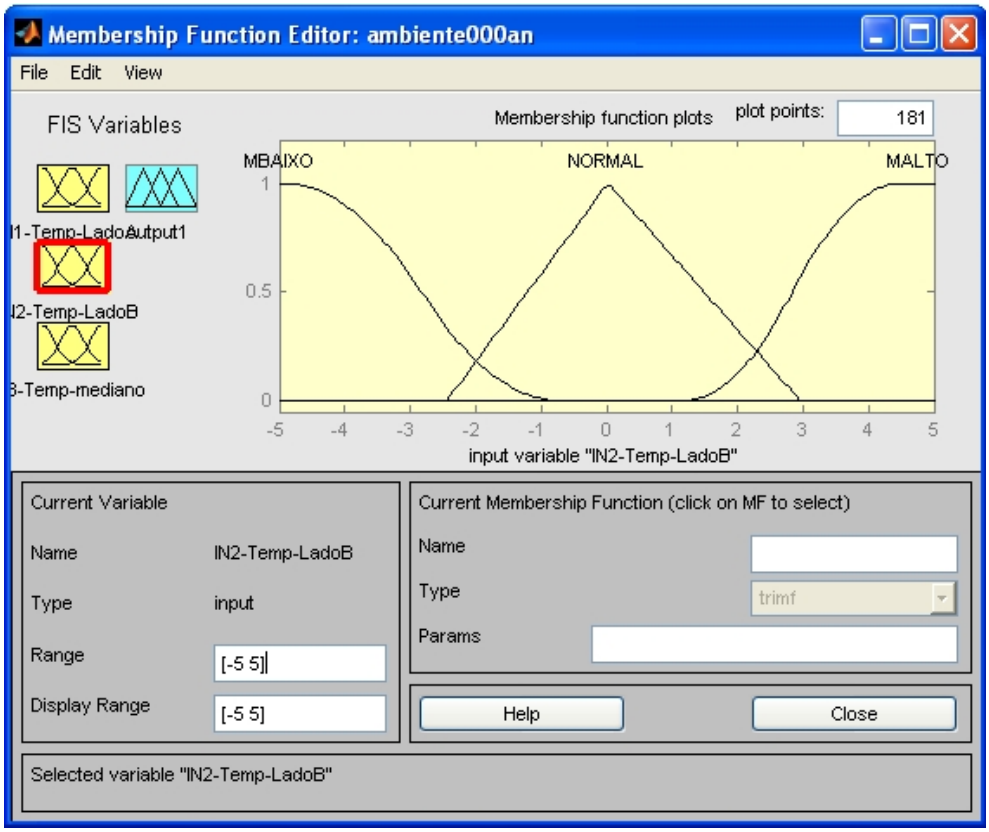


Figura 57 – Entrada IN2 Analógica

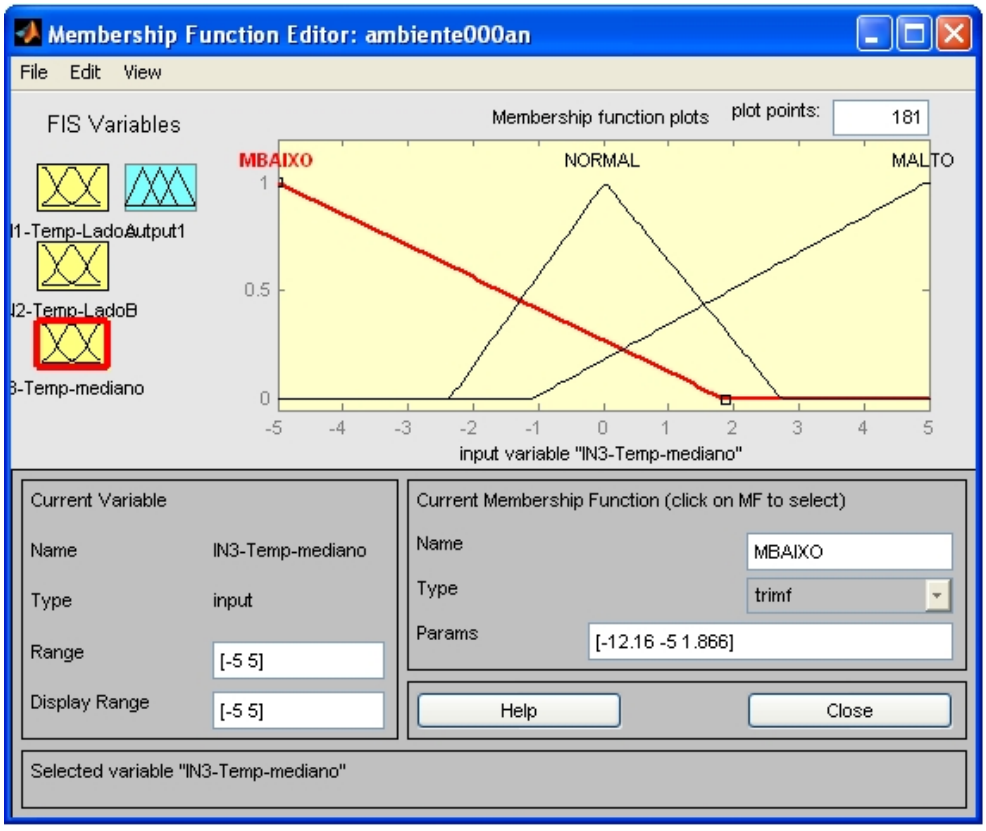


Figura 58 – Entrada IN3 Analógica

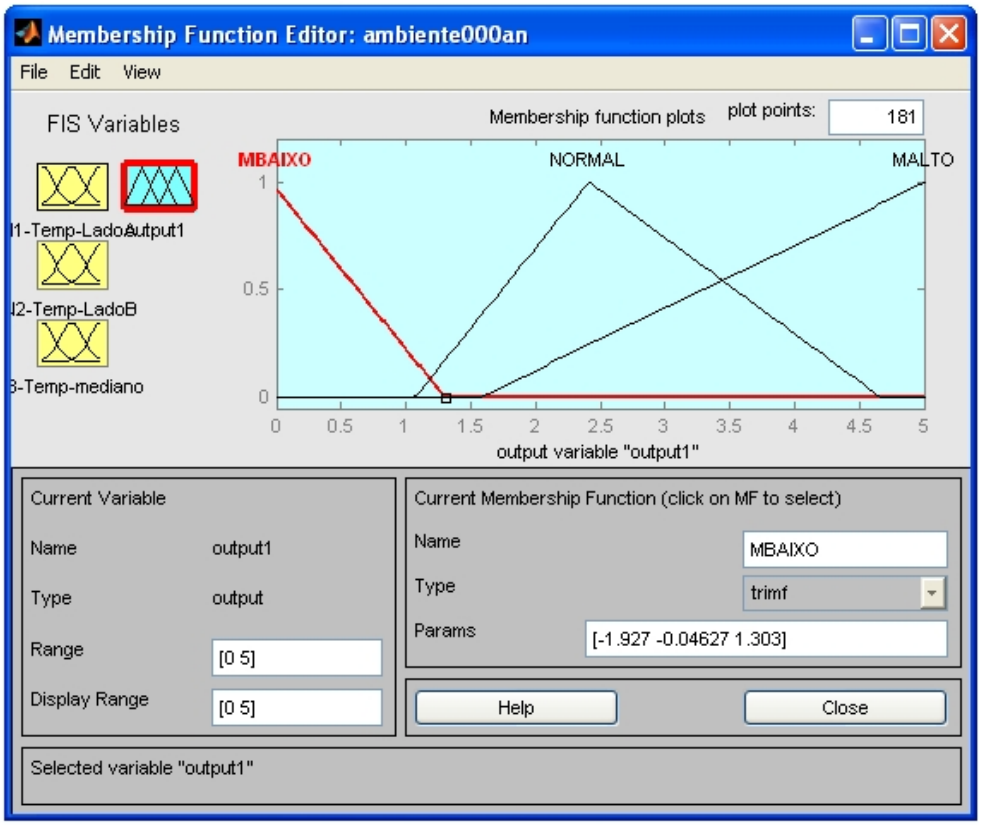


Figura 59 – Saída fuzzy Analógica

6.1 – Criando as regras do Controlador de Velocidade por Temperatura.

As regras foram criadas baseadas em um controlador fictício, formando 27 regras que estão relacionadas abaixo:

- 1. If (IN1-Temp-LadoA is MBAIXO) and (IN2-Temp-LadoB is MBAIXO) and (IN3-Temp-mediano is MBAIXO) then (output1 is MBAIXO) (1)
- 2. If (IN1-Temp-LadoA is MBAIXO) and (IN2-Temp-LadoB is MBAIXO) and (IN3-Temp-mediano is NORMAL) then (output1 is MBAIXO) (1)
- 3. If (IN1-Temp-LadoA is MBAIXO) and (IN2-Temp-LadoB is MBAIXO) and (IN3-Temp-mediano is MALTO) then (output1 is MBAIXO) (1)
- 4. If (IN1-Temp-LadoA is MBAIXO) and (IN2-Temp-LadoB is NORMAL) and (IN3-Temp-mediano is MBAIXO) then (output1 is MBAIXO) (1)
- 5. If (IN1-Temp-LadoA is MBAIXO) and (IN2-Temp-LadoB is NORMAL) and (IN3-Temp-mediano is NORMAL) then (output1 is MBAIXO) (1)
- 6. If (IN1-Temp-LadoA is MBAIXO) and (IN2-Temp-LadoB is NORMAL) and (IN3-Temp-mediano is MALTO) then (output1 is NORMAL) (1)

- 7. If (IN1-Temp-LadoA is MBAIXO) and (IN2-Temp-LadoB is MALTO) and (IN3-Temp-mediano is MBAIXO) then (output1 is NORMAL) (1)
- 8. If (IN1-Temp-LadoA is MBAIXO) and (IN2-Temp-LadoB is MALTO) and (IN3-Temp-mediano is NORMAL) then (output1 is NORMAL) (1)
- 9. If (IN1-Temp-LadoA is MBAIXO) and (IN2-Temp-LadoB is MALTO) and (IN3-Temp-mediano is MALTO) then (output1 is MALTO) (1)
- 10. If (IN1-Temp-LadoA is NORMAL) and (IN2-Temp-LadoB is MBAIXO) and (IN3-Temp-mediano is MBAIXO) then (output1 is MBAIXO) (1)
- 11. If (IN1-Temp-LadoA is NORMAL) and (IN2-Temp-LadoB is MBAIXO) and (IN3-Temp-mediano is NORMAL) then (output1 is NORMAL) (1)
- 12. If (IN1-Temp-LadoA is NORMAL) and (IN2-Temp-LadoB is MBAIXO) and (IN3-Temp-mediano is MALTO) then (output1 is NORMAL) (1)
- 13. If (IN1-Temp-LadoA is NORMAL) and (IN2-Temp-LadoB is NORMAL) and (IN3-Temp-mediano is MBAIXO) then (output1 is NORMAL) (1)
- 14. If (IN1-Temp-LadoA is NORMAL) and (IN2-Temp-LadoB is NORMAL) and (IN3-Temp-mediano is NORMAL) then (output1 is NORMAL) (1)
- 15. If (IN1-Temp-LadoA is NORMAL) and (IN2-Temp-LadoB is NORMAL) and (IN3-Temp-mediano is MALTO) then (output1 is NORMAL) (1)
- 16. If (IN1-Temp-LadoA is NORMAL) and (IN2-Temp-LadoB is MALTO) and (IN3-Temp-mediano is MBAIXO) then (output1 is NORMAL) (1)
- 17. If (IN1-Temp-LadoA is NORMAL) and (IN2-Temp-LadoB is MALTO) and (IN3-Temp-mediano is NORMAL) then (output1 is NORMAL) (1)
- 18. If (IN1-Temp-LadoA is NORMAL) and (IN2-Temp-LadoB is MALTO) and (IN3-Temp-mediano is MALTO) then (output1 is MALTO) (1)
- 19. If (IN1-Temp-LadoA is MALTO) and (IN2-Temp-LadoB is MBAIXO) and (IN3-Temp-mediano is MBAIXO) then (output1 is MBAIXO) (1)
- 20. If (IN1-Temp-LadoA is MALTO) and (IN2-Temp-LadoB is MBAIXO) and (IN3-Temp-mediano is NORMAL) then (output1 is NORMAL) (1)
- 21. If (IN1-Temp-LadoA is MALTO) and (IN2-Temp-LadoB is MBAIXO) and (IN3-Temp-mediano is MALTO) then (output1 is NORMAL) (1)
- 22. If (IN1-Temp-LadoA is MALTO) and (IN2-Temp-LadoB is NORMAL) and (IN3-Temp-mediano is MBAIXO) then (output1 is NORMAL) (1)

- 23. If (IN1-Temp-LadoA is MALTO) and (IN2-Temp-LadoB is NORMAL) and (IN3-Temp-mediano is NORMAL) then (output1 is NORMAL) (1)
- 23. If (IN1-Temp-LadoA is MALTO) and (IN2-Temp-LadoB is NORMAL) and (IN3-Temp-mediano is NORMAL) then (output1 is NORMAL) (1)
- 24. If (IN1-Temp-LadoA is MALTO) and (IN2-Temp-LadoB is NORMAL) and (IN3-Temp-mediano is MALTO) then (output1 is NORMAL) (1)
- 25. If (IN1-Temp-LadoA is MALTO) and (IN2-Temp-LadoB is MALTO) and (IN3-Temp-mediano is MBAIXO) then (output1 is NORMAL) (1)
- 26. If (IN1-Temp-LadoA is MALTO) and (IN2-Temp-LadoB is MALTO) and (IN3-Temp-mediano is NORMAL) then (output1 is MALTO) (1)
- 27. If (IN1-Temp-LadoA is MALTO) and (IN2-Temp-LadoB is MALTO) and (IN3-Temp-mediano is MALTO) then (output1 is MALTO) (1)

Essas regras ficarão fixas ao longo da pesquisa sobre a quantização das entradas em questão.

Após configurar todo o sistema fuzzy, através do programa dissert_prog_02.m, foi gerado o mapa de combinações dos valores de entrada e suas respectivas saídas a serem transferidos para a plataforma, formando o mapeamento no arquivo saída_fuzzy.txt.

A geração do arquivo para transferência é realizada em blocos de 512 valores, e sua configuração já foi apresentada na Tabela 8 e 9 do capítulo 5. O tempo de execução para geração deste arquivo foi de aproximadamente 4 horas. Depois de gerado este arquivo, ele sofre duas alterações: a retirada das vírgulas entre os números e a transformação dos números em valores ASCII. Para tal fim, foi criado um programa específico para completar a operação de transferência.

6.2 Programa Formatador do Arquivo Mapeado para Transferência

O programa “FuzzyFormata” tem todas as funções para formatar o arquivo saída_fuzzy.txt produzido pelo Matlab no padrão a ser enviado ao cartão SD. Conforme a Figura 60, uma tela é apresentada para se selecionar a função inicial que é de carregar o programa produzido pelo Matlab, que denominamos “saída_fuzzy.txt”.

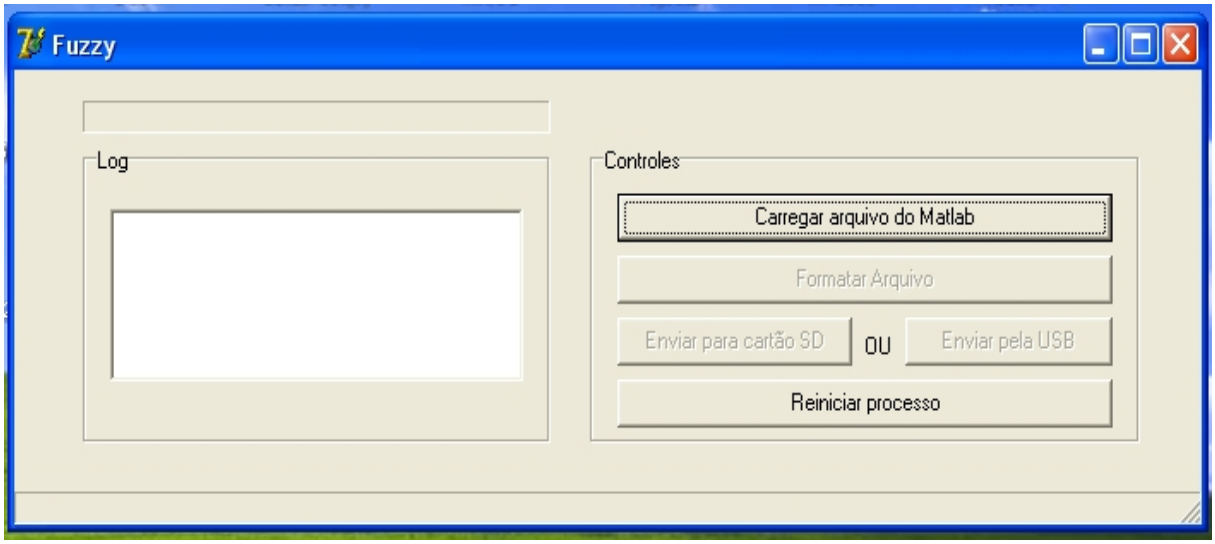


Figura 60 – Tela Inicial do Programa “FuzzyFormata”

Ao se clicar no ícone “ Carregar arquivo do Matlab”, uma tela de busca de arquivos é aberta. Nesta primeira etapa, o arquivo “saída_fuzzy.txt” deve ser então selecionado. Ver Figura 61.

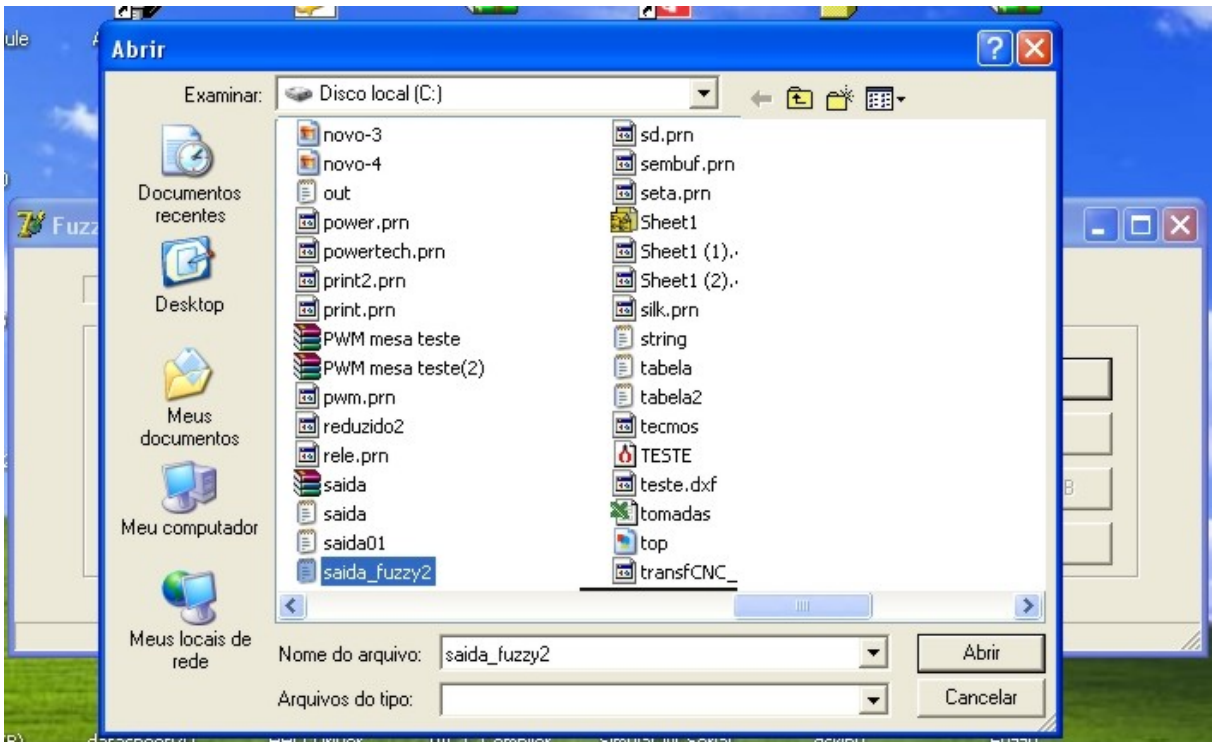


Figura 61 – Carregando arquivo saída_fuzzy.txt

Com o arquivo selecionado, uma nova tela do programa é apresentada na Figura 62, que tem um histórico de operações executadas (*Log*). Esta indicação de operação facilita a visualização das operações executadas pelo programa

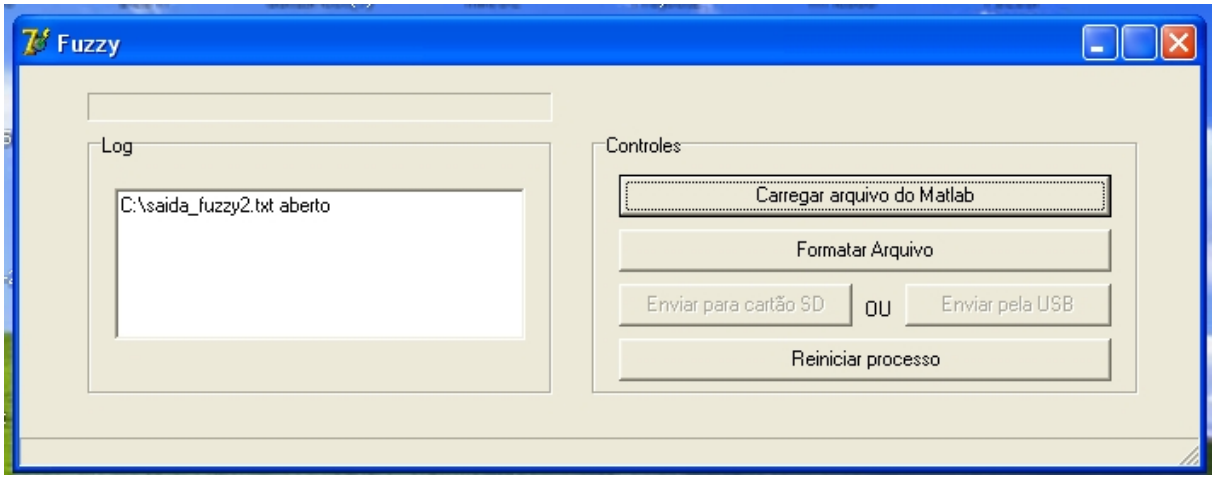


Figura 62 – Log que demonstra o arquivo "saída_fuzzy.txt" aberto.

A segunda etapa será na formatação do arquivo. Esta formatação vai retirar as vírgulas produzidas pelo Matlab no arquivo saída_fuzzy.txt, bem como colocar os valores que estão em decimal no padrão ASCII de transferência para o Cartão SD. Ver Figura 63.

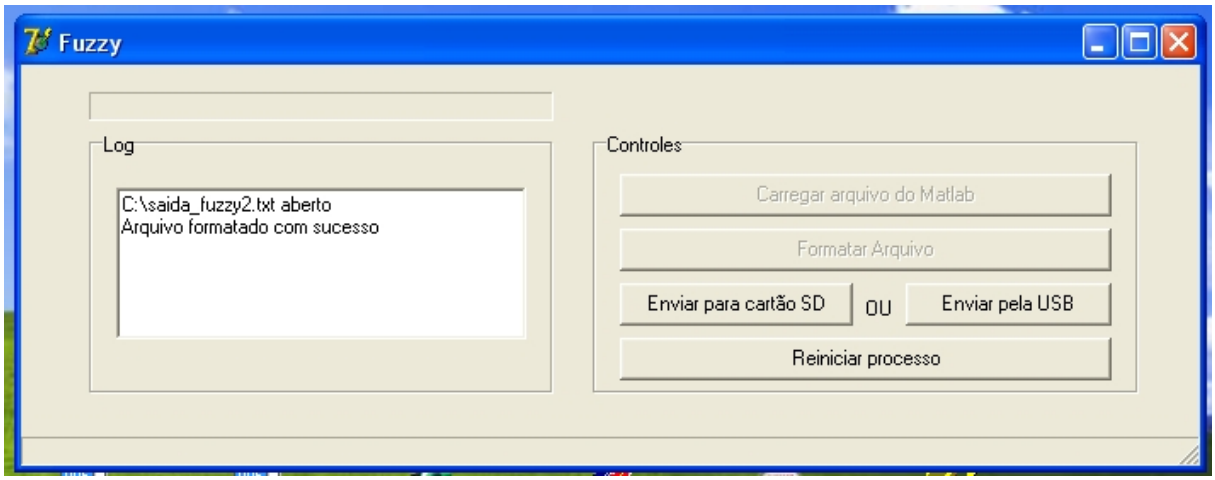


Figura 63 – Log que demonstra o arquivo "saída_fuzzy.txt" formatado

A terceira etapa pode ser escolhida em duas formas: Ou "Enviar para o cartão SD" na qual transfere para o cartão SD através de um leitor/gravador de cartão SD conectado

diretamente no PC, ou através da transferência via conexão USB. No primeiro caso, o programa abrirá uma tela de formatação padrão do Windows que irá formatar o cartão SD. Isso se faz necessário para garantir que o arquivo mapeado ao ser transferido para o cartão comece na primeira linha de endereçamento do mesmo, porque é essa linha que será o valor inicial de busca para o valor 0000...0000 nas entradas da plataforma. Caso tenha alguma coisa gravada neste endereço, a transferência será alocada em outra posição, dando erro generalizado ao se buscar os dados no cartão pela plataforma. A segunda opção seria enviar pelo USB. Este processo é mais demorado e tem um tempo geral de execução em torno de vinte minutos. Ver Figuras 64 e 64.

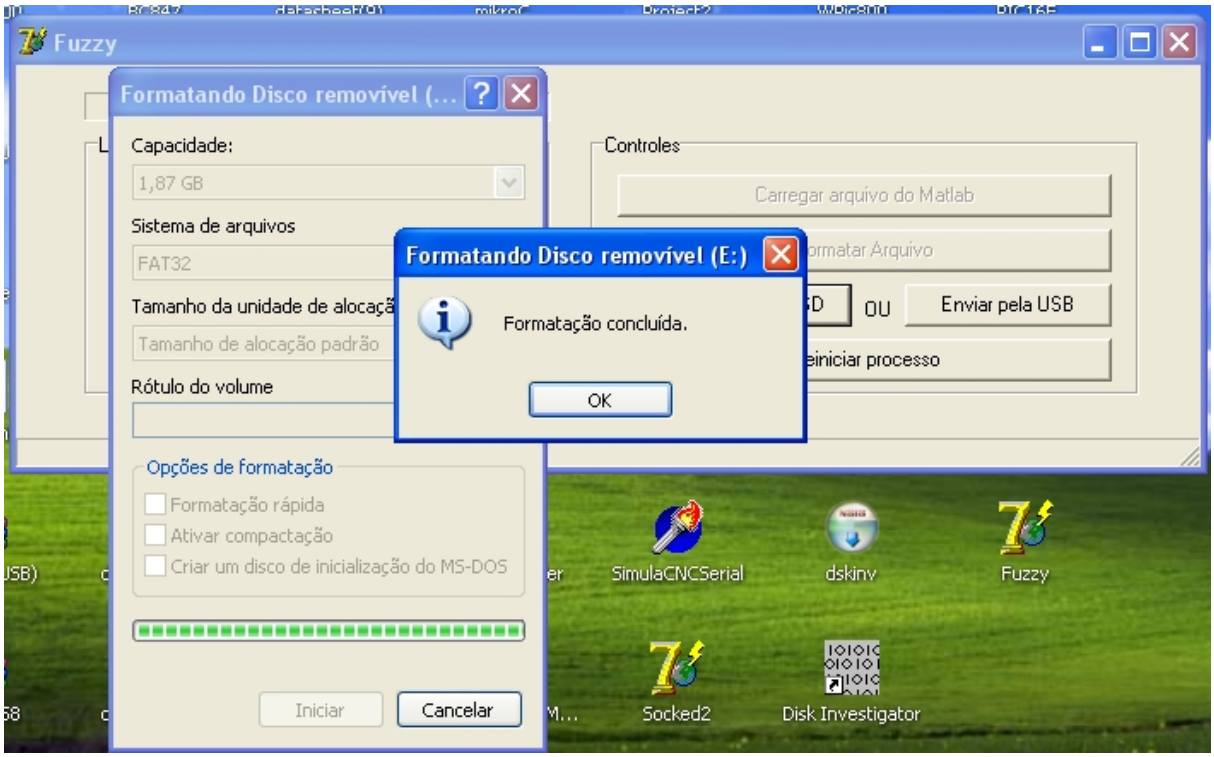


Figura 64 – Formatação do Cartão SD

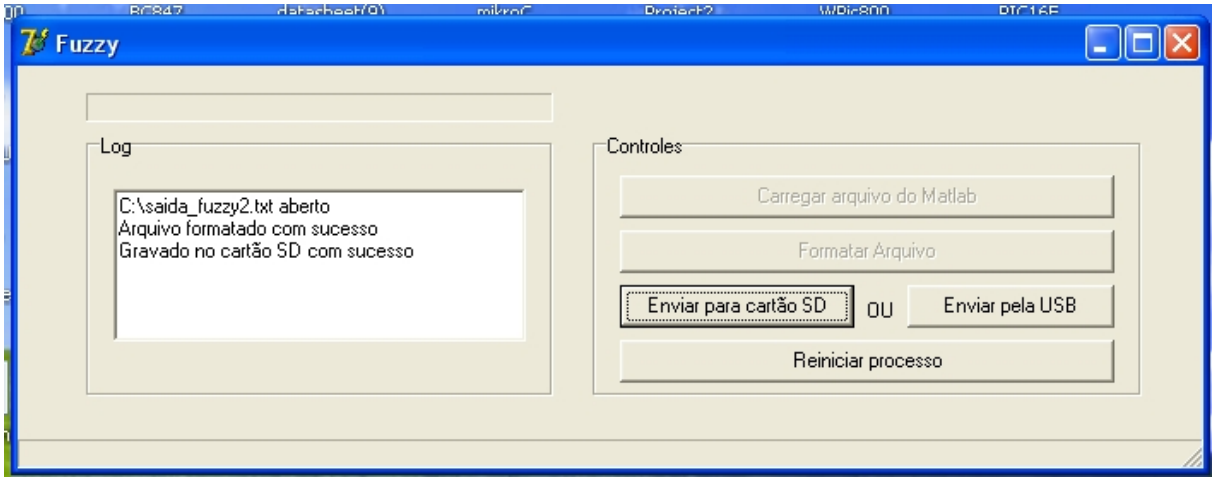


Figura 65 – Gravando no Cartão SD após Formatação Física.

A última opção deste sistema é de Reiniciar o Processo, em que clicando neste botão, teremos o sistema reinicializado para novo procedimento, zerando o histórico de eventos.

Abrindo o arquivo em um editor ASCII, temos ilustrado na Figura 66 o início do arquivo mapeado após a devida formatação. Após a transferência, temos todo o mapa gerado anteriormente no Matlab agora no interior do cartão. Em um editor de memória de disco podemos visualizar onde está gravado dentro do cartão SD o arquivo transferido. Ver a Figura 67. Vale ressaltar que o início do cartão até o setor 7711 foi utilizado como cabeçalho de formatação. O início de dados do cartão começa no setor 7712. Nota-se que na formatação usada pelo sistema operacional Vista, este valor inicial do setor é alterado na formatação. Como o sistema na plataforma foi projetado para ler neste setor 7712, tem que ser usado o sistema operacional XP na formatação do cartão SD.

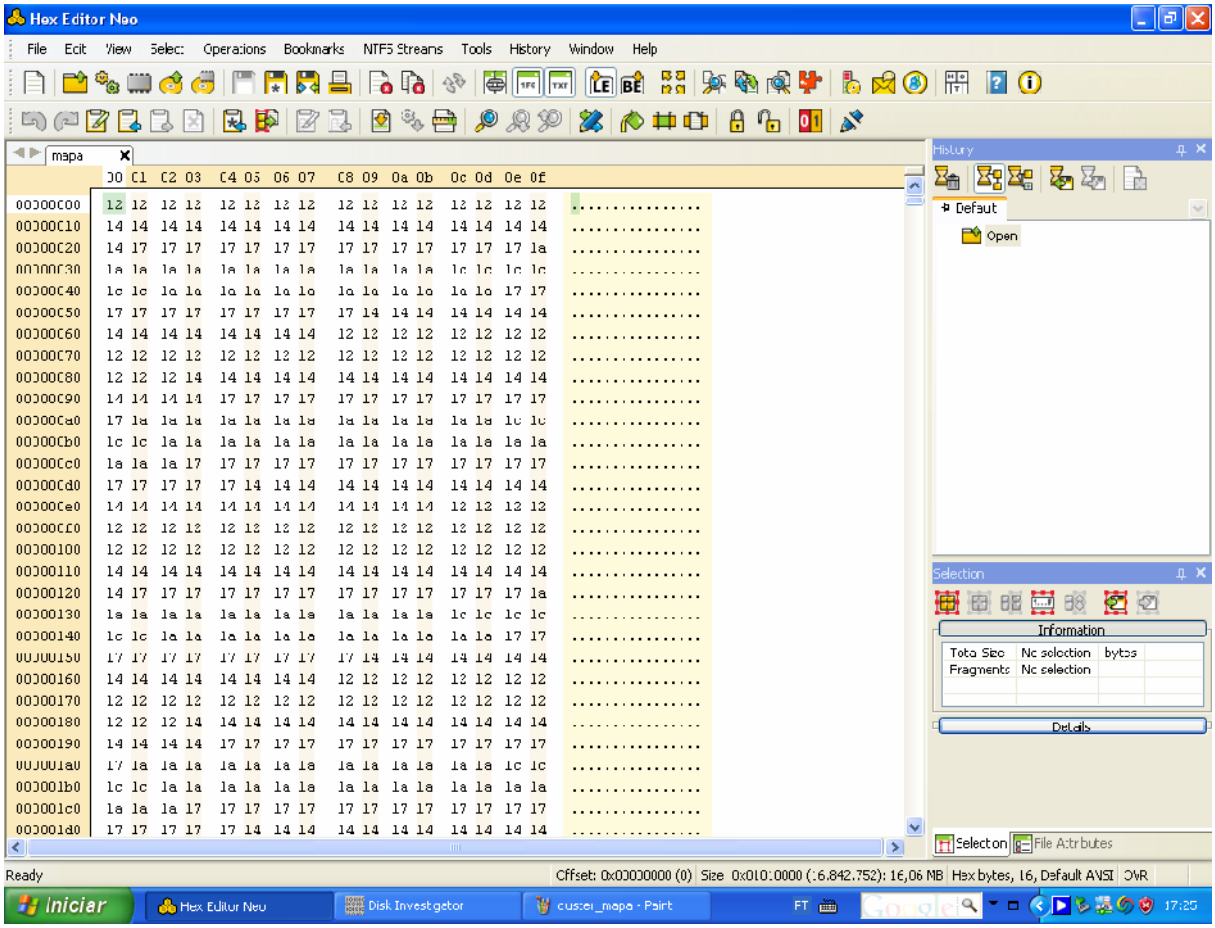


Figura 66 – Inicio do arquivo formatado para entrar no cartão SD.

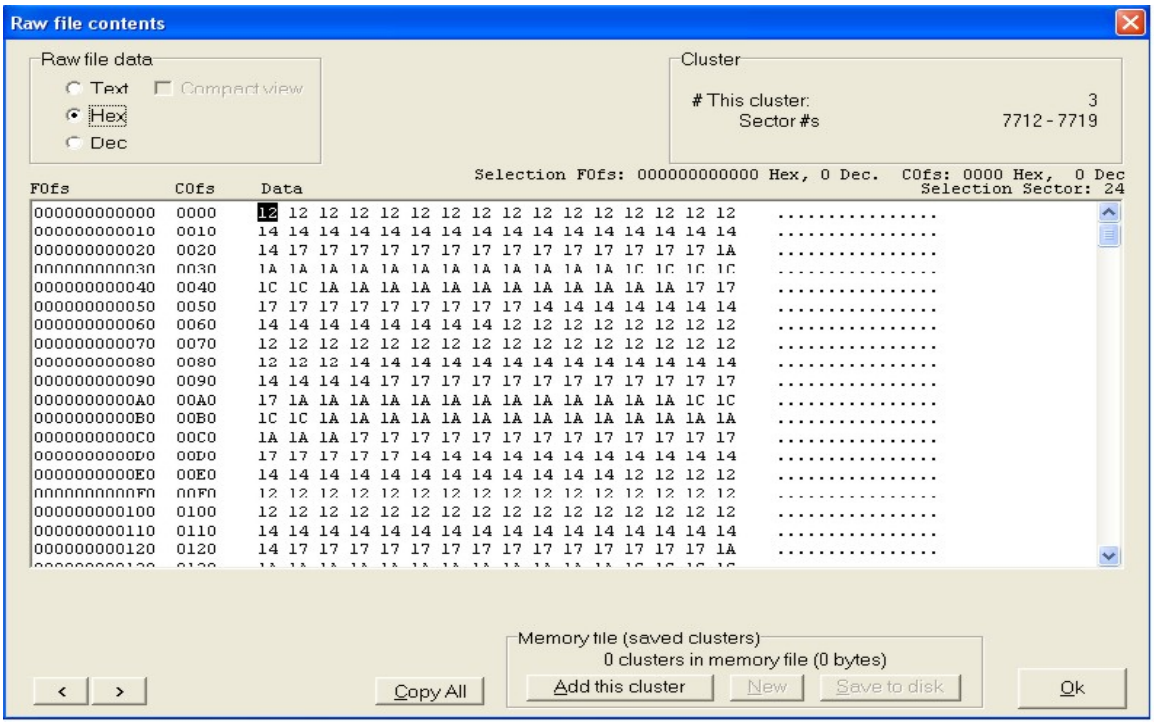


Figura 67 – Inicio do arquivo alocado no interior do Cartão SD.

6.3 Testando o Hardware do Mapeador

Foi feita uma amostragem de 125 pontos distintos para verificar os erros entre os valores quantizados em 8 bits produzidos no Matlab e os valores medidos na plataforma. A tabela dos pontos de teste foi produzida com a variação de cada entrada com passos de 2,50 volts. No primeiro teste – Teste 1 – foi fornecido à plataforma o valor de -5.00 volts para IN1, -5.00 volts para IN2 e – 5.00 volts para IN3, obtendo a primeira saída. No segundo teste – Teste 2, valor de -5.00 volts para IN1, -5.00 volts para IN2 e – 2,50 volts para IN3, obtendo a segunda saída. E assim sucessivamente, até ter o último teste – Teste 125, com 5.00 volts para IN1, 5.00 volts para IN2 e 5.00 volts para IN3. Os sinais de entrada foram fornecidos por um três divisores de tensão, onde foram usados três potenciômetros de 10 K multivoltas. A tabela 10 fornece os dados dos erros mais relevantes dos testes.

Tabela 10 – Tomada de valores reais com devidos erros

Linha 1	Número do teste	16	41	66	67	91
Linha 2	Entrada Analógica IN1	-5	-5	-5	-2,5	-5
Linha 3	Entrada Analógica IN2	2,5	2,5	2,5	2,5	2,5
Linha 4	Entrada Analógica IN3	-5	-2,5	0	0	2,5
Linha 5	Saida digital binária no Matlab	01111111	01111111	10000111	10000111	10100110
Linha 6	Saida digital binária apresentada na Plataforma	10000010	10000010	10001010	10001010	10001010
Linha 7	Erro decimal entre as linhas 5 e 6 acima	3	3	3	3	2
Linha 8	Saída Analógica no Matlab com conversor D/A de 8 bits	2,5	2,5	2,65	2,65	3,25
Linha 9	Saída Analógica na Plataforma após o conversor D/A de 8 bits	2,56	2,56	2,71	2,71	3,31
Linha 10	Erro analógico entre as linhas 8 e 9 acima	0,06	0,06	0,06	0,06	0,06
Linha 11	Erro percentual analógico em %	2,4	2,4	2,264151	2,264151	1,846154

Na linha 11 temos o valor percentual dos erros medidos entre os valores simulados no Matlab e os valores transmitidos para a plataforma. Os testes 16 e 41 apresentaram maior erro percentual. Com a imprecisão dos instrumentos associada à tolerância dos componentes, os resultados foram bastante satisfatórios, superando as expectativas iniciais para um sistema de baixo custo. Os resultados mostram a viabilidade de utilização deste sistema para aplicações em pesquisas de sistemas fuzzy mapeados.

6.4 Verificando a Quantização do sistema em relação à Sensibilidade e Robustez

A conversão de analógica para digital é o passo inicial para um processamento digital de qualquer tipo e natureza, e neste processo a quantização define qual é o nível de precisão e fidelidade aos dados convertidos. Conforme BRUNO DE SÁ, 2009, na síntese de filtros digitais, os coeficientes e os sinais devem ser quantizados antes de serem armazenados e processados por um processador digital de sinais. Caso a sensibilidade do filtro seja considerada, os efeitos da quantização afetam a resposta do filtro e é possível obter estruturas mais ou menos sensíveis aos efeitos causados pela quantização.

Em um sistema fuzzy baseado numa arquitetura de mapeador, a pesquisa por uma sensibilidade dentro do erro máximo tolerável pode fornecer uma redução quanto ao uso de memória mapeada. Este tipo de redução do uso de memória será especialmente vantajoso quando se tem o intuito de se produzir um hardware com memórias altamente velozes. Por outro lado, com um hardware de processamento de sinais, uma vantagem seria a redução no tempo de processamento devido à redução no número de bits usados nos cálculos.

Também cabe ressaltar que se faz necessário uma pesquisa para verificar se o nível de quantização aplicado a um sistema analógico vai produzir erros na saída deste sistema, e se esses erros estão dentro da tolerância admitida em sua especificação.

A verificação de sensibilidade foi implementada via software no Matlab. O sistema fuzzy analógico Controlador de Velocidade por Temperatura foi a base para comparação para as diferentes quantizações de suas entradas, para obter dados comparativos e verificar qual seria a menor quantização possível de se implementar.

Foi produzida uma tabela de valores analógicos que serviu de entrada para o sistema fuzzy analógico concomitante para com os demais sistemas quantizados em valores distintos nas entradas. Esta tabela de entradas teve um incremento de valor de 0,04 volts entre -5 e 5 volts, que é a faixa de valores na entrada do conversor A/D proposto na plataforma. Todas as combinações de valores nas entradas resultaram 15.813.251 testes comparativos entre a inferência analógica e as demais. Ver Figura 68.

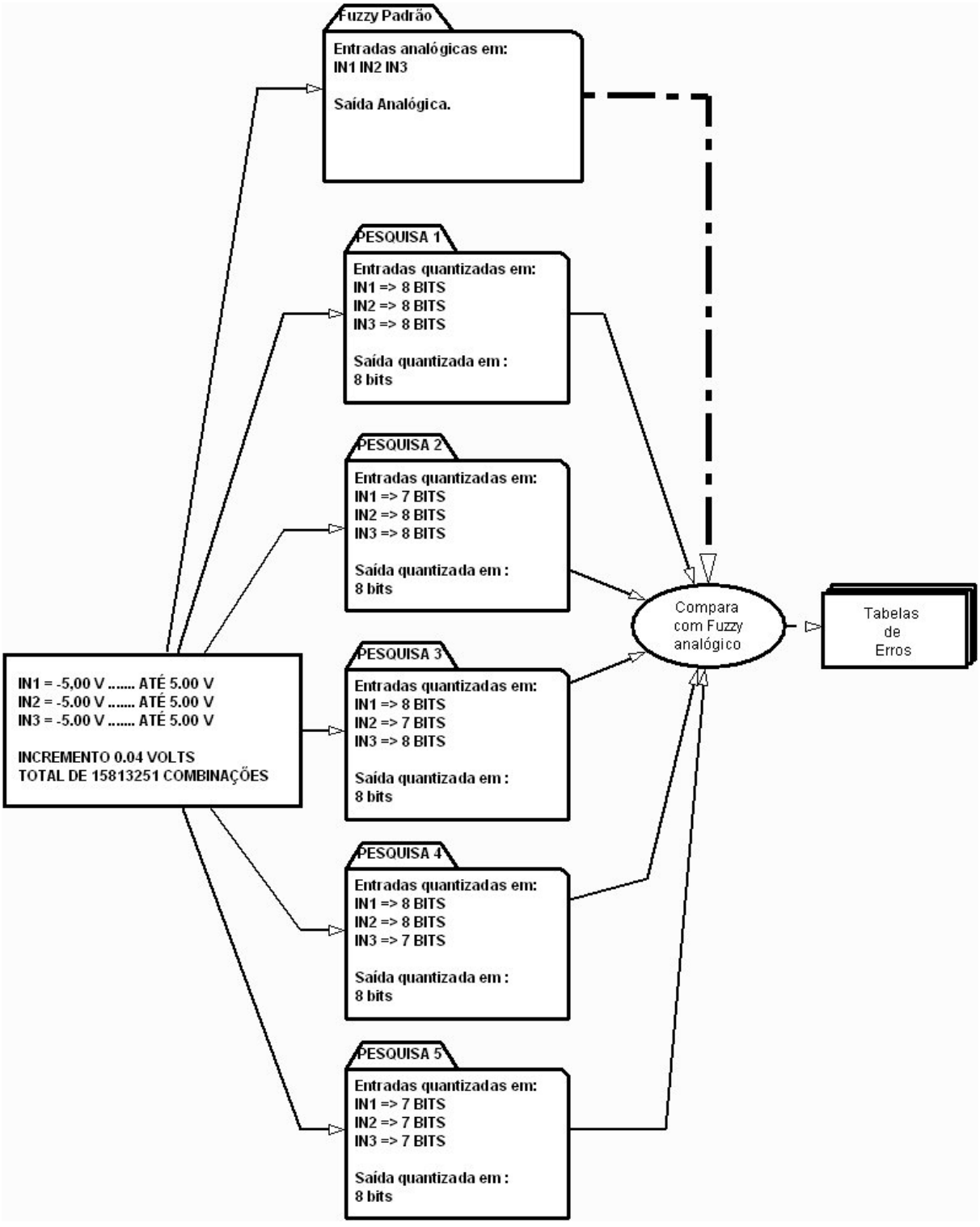


Figura 68 – Pesquisas comparativas de quantizações

A comparação dos valores foi realizada entre as saídas do sistema fuzzy analógico com os sistemas quantizados.

6.4.1 Resultados das Quantizações

A primeira tabela de resultados dos erros surgiu do confronto entre as saídas do sistema fuzzy analógico versus sistema quantizado com as entradas IN1 em 8 bits, IN2 em 8 bits e IN3 em 8 bits. Ver a tabela 11.

Tabela 11 – Resumos dos erros entre sistema fuzzy Analógico versus quantizado com IN1 em 8 bits, IN2 em 8 bits e IN3 em 8 bits

Entrada IN1 analógica	-2,6
Entrada IN2 analógica	-5
Entrada IN2 analógica	-2,36
Entrada IN1 analógica no conversor A/D-8bits	1,2
Entrada IN2 analógica no conversor A/D-8bits	0
Entrada IN2 analógica no conversor A/D-8bits	1,32
Saída Fuzzy Analógica	0,614468
Saída Fuzzy Analógica com as entradas IN1 em 8bits, IN2 em 8bits e IN3 em 8bits	0,529412
Erro Absoluto máximo em volts	0,085056
Percentual de erro máximo (%)	13,84%
Erro Médio em volts	0,0074

Nesta tabela está armazenado o valor de erro percentual máximo absoluto encontrado em todas as comparações. Apesar do valor absoluto ser pequeno, 0,085 volts, este erro representa 13,84 % em relação a saída esperada. Como está 1,16% abaixo da tolerância permitida de 15%, é possível estabelecer a quantização deste sistema analógico para um digital com as suas entradas em 8 bits, e a sua saída também em 8 bits. Observa-se nesta pesquisa que um controlador fuzzy digital com 3 entradas e 27 regras, visando controlar a velocidade de um exaustor, ficou perto do limite da tolerância máxima de sua saída, em uma quantização de 8 bits. Em relação aos erros médios, que foi a soma de todos os erros divididos pela quantidade de medidas, o resultado foi de um valor bem menor que o erro máximo absoluto.

A segunda, terceira, quarta e quinta tabelas de resultados dos erros surgiram das relações entre as saídas do sistema fuzzy analógico versus as variações do sistema fuzzy nas demais quantizações. Todos os erros percentuais máximos absolutos foram acima da tolerância definida para este sistema, demonstrando que somente a quantização para entradas de 8 bits pode ser implementada para atender a especificação de tolerância inferior a 15%. Uma observação nessas tabelas sobre o erro médio é que são todos bem abaixo do erro

máximo absoluto em cada uma delas, indicando que a maioria dos erros produzidos nos confrontos de resultados foram bem pequenos. As tabelas 12, 13, 14 e 15 concatenam os dados resultantes dessas interações. Caso a tolerância para este projeto fosse de 30 %, o sistema poderia ser quantizado com IN1 em 7 bits, IN2 em 8 bits e IN3 em 8 bits.

Tabela 12 – Resumos dos erros entre sistema fuzzy analógico versus quantizado com IN1 em 7 bits, IN2 em 8 bits e IN3 em 8 bits

Entrada IN1 analógica	-3,4
Entrada IN2 analógica	-1,96
Entrada IN2 analógica	-5
Entrada IN1 analógica no conversor A/D-7bits	0,8
Entrada IN2 analógica no conversor A/D-8bits	1,52
Entrada IN2 analógica no conversor A/D-8bits	0
Saída Fuzzy Analógica	0,839967
Saída Fuzzy Analógica com as entradas IN1 em 7bits, IN2 em 8bits e IN3 em 8bits	0,588235
Erro Absoluto máximo em volts	0,251732
Percentual de erro máximo (%)	29,97%
Erro Médio em volts	0,0096

Tabela 13 – Resumos dos erros entre sistema fuzzy analógico versus quantizado com IN1 em 8 bits, IN2 em 7 bits e IN3 em 8 bits

Entrada IN1 analógica	-3,32
Entrada IN2 analógica	-2,44
Entrada IN2 analógica	-5
Entrada IN1 analógica no conversor A/D-8bits	0,84
Entrada IN2 analógica no conversor A/D-7bits	1,28
Entrada IN2 analógica no conversor A/D-8bits	0
Saída Fuzzy Analógica	0,604354
Saída Fuzzy Analógica com as entradas IN1 em 8bits, IN2 em 7bits e IN3 em 8bits	0,901961
Erro Absoluto máximo em volts	0,297607
Percentual de erro máximo (%)	49,24%
Erro Médio em volts	0,0110

Tabela 14 – Resumos dos erros entre sistema fuzzy analógico versus quantizado com IN1 em 8 bits, IN2 em 8 bits e IN3 em 7 bits

Entrada IN1 analógica	1,96
Entrada IN2 analógica	-5
Entrada IN2 analógica	-2,36
Entrada IN1 analógica no conversor A/D-8bits	3,48
Entrada IN2 analógica no conversor A/D-8bits	0
Entrada IN2 analógica no conversor A/D-7bits	1,32
Saída Fuzzy Analógica	0,605504
Saída Fuzzy Analógica com as entradas IN1 em 8bits, IN2 em 8bits e IN3 em 7bits	0,901961
Erro Absoluto máximo em volts	0,296457
Percentual de erro máximo (%)	48,96%
Erro Médio em volts	0,0086

Tabela 15 – Resumos dos erros entre sistema fuzzy analógico versus quantizado com IN1 em 7 bits, IN2 em 7 bits e IN3 em 7 bits

Entrada IN1 analógica	-3,32
Entrada IN2 analógica	-2,44
Entrada IN2 analógica	-5
Entrada IN1 analógica no conversor A/D-7bits	0,84
Entrada IN2 analógica no conversor A/D-7bits	1,28
Entrada IN2 analógica no conversor A/D-7bits	0
Saída Fuzzy Analógica	0,604354
Saída Fuzzy Analógica com as entradas IN1 em 7bits, IN2 em 7bits e IN3 em 7bits	0,901961
Erro Absoluto máximo em volts	0,297607
Percentual de erro máximo (%)	49,24%
Erro Médio em volts	0,0137

6.4.2 Análise do Tempo de Resposta da Plataforma

Foi medido o tempo de resposta da plataforma com o intuito de se ter dados comparativos aos demais sistemas. Foi fornecido às entradas um conjunto de valores analógicos no qual se conhecesse a saída esperada. Depois outro conjunto diferente de valores analógicos foi apresentado na entrada, tendo também uma saída diferente da primeira, e conhecida. A diferença entre a mudança do sinal de entrada para a mudança de sinal na saída forneceu o tempo de resposta do sistema como um todo. O valor do tempo obtido foi de 190 microssegundos. A Figura 69 ilustra o gráfico das entradas e o gráfico das mudanças na saída do sistema.

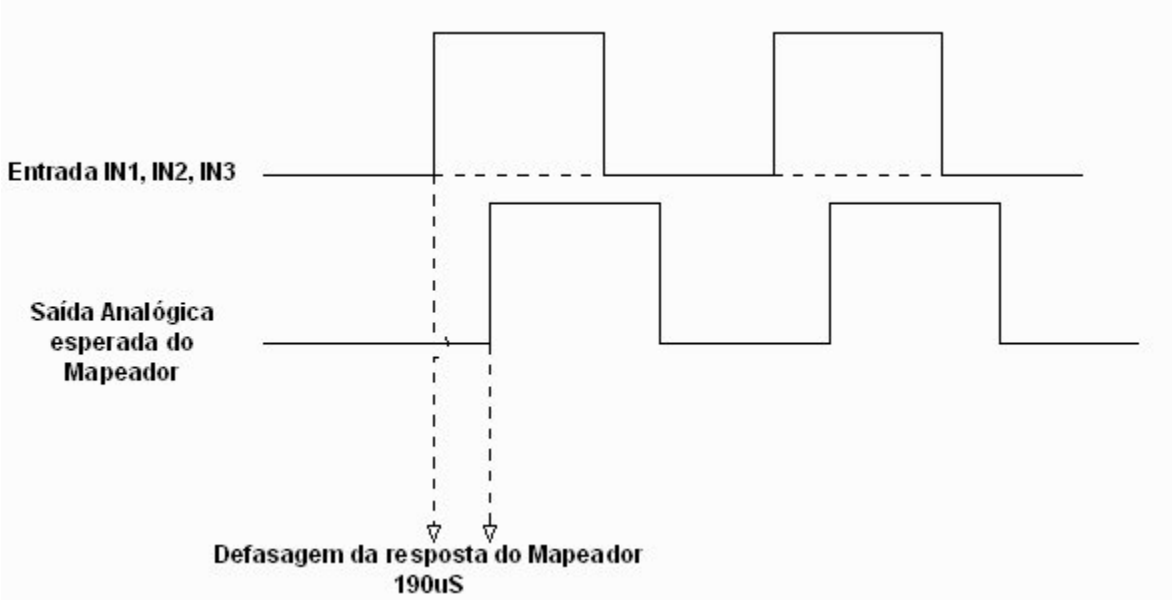


Figura 69 – Tempo de Resposta do Sistema

CAPÍTULO 7
CONCLUSÕES E TRABALHOS FUTUROS

7.1 Considerações Finais

Esta dissertação apresentou um estudo sobre a implementação eletrônica de um sistema de inferência fuzzy em um hardware baseado em uma arquitetura de mapeador analógico-digital.

Além da construção de um protótipo funcional microcontrolado que utiliza cartões do tipo SD, a pesquisa também focou na redução dos bits necessários para quantização através da avaliação de erros via software – Matlab, e na implementação de acumuladores para um futuro estudo da supervisão de uso das entradas, com o fim de auxiliar na redução do número de bits de quantização das mesmas.

Cabe ressaltar que é possível escolher no hardware as quantizações individuais para cada uma das entradas, variando de 5 a 8 bits, o que possibilita a implementação em um mesmo sistema de diversas combinações em relação à quantização das entradas.

Para viabilizar o estudo relativo à quantização, utilizou-se o próprio Matlab para gerar tabelas comparativas para análise. Deste modo, foi realizada uma análise para avaliar a possibilidade de redução na quantização das entradas de um sistema controlador de velocidade de exaustor.

Concluiu-se nesta análise que a redução de quantização de um sistema em *n* bits para um de quantização menor de *m* bits fica totalmente condicionado à faixa de tolerância de erro admissível na saída. Mesmo considerando-se uma faixa de erro elevado, provavelmente haverá uma faixa que se mostrará impeditiva em relação a esta diminuição de quantização. Porém, conforme foi apresentado no capítulo 6, esta faixa pode se tornar pequena, podendo ou não ser relevante ao processo controlado pela execução da aplicação.

Espera-se que o acumulador de incidências de valores de entrada venha a ser uma ferramenta auxiliar no processo de redução da quantização. Tal acumulador vai fornecer ao especialista dados reais, indicando se esses valores, impeditivos para redução dos bits, foram muitos, poucos ou não utilizados no processo real de execução da inferência.

Em relação ao tempo de resposta, esta plataforma se mostra constante no aspecto de trabalhar com 5 ou com 8 bits de quantização nas entradas, pois trabalha com mapeamento da memória. Por este motivo, espera-se um melhor desempenho em relação às plataformas microcontroladas que possuem a inferência realizada com cálculos na própria plataforma, em especial quando o mapeador estiver configurado para trabalhar com 8 bits. Esta conclusão se fez pelo fato das plataformas que fazem o processamento local, quando trabalham com as 3 entradas quantizadas em 8 bits, terão que realizar cálculos mais complexos, tornando o tempo de resposta maior do que o produzido neste mapeador. Esta observação considera o uso de processadores da mesma família, de 8 ou 16 bits, tanto para a plataforma mapeada como para de execução das inferências locais.

Pela flexibilidade na implementação de um sistema fuzzy no mundo real, como unidade de hardware de desenvolvimento para controladores, a plataforma é um sistema adequado e consistente, que serve como passo inicial para viabilizar testes em laboratório de aplicações que envolvem sistemas fuzzy, de um modo geral, e controladores fuzzy em especial.

7.2 Trabalhos Futuros

O foco da pesquisa foi a implementação de um protótipo funcional de uma arquitetura baseada em mapeador para um sistema fuzzy e as respectivas tecnologias envolvidas, mais especificamente, as memórias do tipo SD (*Secure Digital*), a interface USB e a utilização de microcontrolador.

Tendo em vista que existe um protótipo funcional, uma idéia de trabalho futuro seria uma comparação com outras arquiteturas de implementação em hardware, especialmente as que utilizam microcontroladores. Aspectos relativos ao desempenho, ou seja, ao tempo de inferência, seriam importantes.

Uma outra possibilidade é evoluir os estudos relativos à redução do número de bits utilizados na quantização pois tais estudos serviriam também para outras arquiteturas de implementação via hardware e podem gerar resultados importantes para a simplificação das implementações eletrônicas.

REFERÊNCIAS

YAMAKAWA, T. A Fuzzy Inference Engine in Nonlinear Analog Mode and Its Application to Fuzzy Logic Control. IEEE Trans. on Neural Networks, Vol.4, no. 3, pp. 496-522, May 1993.

AMARAL, J. F. M SANTINI, C. C., TANSCHEIT, R., VELLASCO, M. M. R., PACHECO, M. A. C., Towards Evolvable Analog Fuzzy Logic 11 7 Controllers, in Proc. of the NASA/DoD Conference on Evolvable Hardware, 2002.

AMARAL, J.F.M, "Síntese de Sistemas Fuzzy por Computação Evolucionária", Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, Fevereiro, 2003.

AMARAL, J. F. M. do et al. Intrinsic evolutionary design of analog building blocks for fuzzy logic controllers. In: NEDJAH, N.; MOURELLE, L. M. (Ed.). Evolutionary Machine Design: Methododology and applications. 1. ed. New York, USA: Nova Science Publisher, Inc, 2005.

BATURONE, S. S´anchez Solano, J. L. Huertas. CMOS Design of a Current-Mode Multiplier/ divider Circuit with Applications to Fuzzy Controllers. Analog Integrated Circuits and Signal Processing, Kluwer Academic publishers, Vol. 23, N. 3, pp. 199-210, June 2000.

BAUCHSPIESS, A., GOSMANN, H. L. (2002). Controle Fuzzy paraSistema de Nível de Líquidos In: CBA2002 - Congresso Brasileiro de Automática, Natal-RN.

BLICKLE, T.; THIELE, L. A mathematical analysis of tournament selection. In: Proceedings of the Sixth International Conference on Genetic Algorithms. San Francisco, CA: Morgan Kaufmann, 1996

CATANIA,V; RUSSO, M"A VLS1 fuzzy inference processor based on a discrete analog approach,". Fuzzy , 1994.

COLOMBANO, S.P; LOHN,J.D;, "Spatial autocatalytic dynamics: an approach to modeling prebiotic evolution," Proc. 1998 Intl. Conf. on Complex Systems, Nashua NH, 1998

DRIANKOV, D.; HELLENDOORN, H.; REINFRANK, M. An Introduction to Fuzzy Control, Springer-Verlag, 1996.

FABRO, J. A., Uma abordagem neuro-nebulosa para controle preditivo de processos multi-estágios. Tese de Doutorado, Programa de Pós-Graduação em Eng.Elétrica e Informática Industrial, CEFET-PR, Curitiba – PR, 2003

GOLDBERG, D. E., KORB, B., and DEB, K. , Messy genetic algorithms: Motivation, analysis, and first results. Technical Report TCGA Report No. 89002, The Clearinghouse for Genetic Algorithms, University of Alabama, Tuscaloosa ,1989

GUPTA, M.M.; QI, J. Theory of T-norms and Fuzzy Inference Methods. Fuzzy Sets and Systems no. 40, pp. 431-450, 1991

HARVEY, I., HUSBANDS, P., CLIFF, D., THOMPSON, A., JAKOBI, N., Evolutionary robotics: the Sussex approach, Robotics and Autonomous Systems, 20 (2-4), pp. 205-224, 1997.

HOLLAND, J. Adaptation In Natural and Artificial Systems. The University of Michigan Press, Ann Arbour, 1975

HIGUCHI, T.; IWATA, M.; KEYMEULEN, D.; SAKANASHI, H.; MURAKAWA, M.; KAJITANI, I.; TAKANASHI, E.; TODA, K.; SALAMI, M.; KAJIHARA, N.; OTSU, N. Real World Applications of Analog and Digital Evolvable Hardware. IEEE Transactions on Evolutionary Computation, Vol.3, no. 3, September 1999.

HUERTAS,J. L., S. S´ANCHEZ-SOLANO, BARRIGA A., BATURONE I. . Serial Architecture for Fuzzy Controllers: Hardware Implementation Using Analog/Digital VLSI Techniques. Proc. 2nd. Int. Conf. on Fuzzy Logic and Neural Networks, pp. 535-538, Iizuka, Jul. 1992.

LJUNG, L., System Identification - Theory For the User, 2nd ed, PTR Prentice Hall, 1999

JANG, J.-S.; HAN, K.-H.; KIM, J.-H. Face Detection using Quantum-inspired Evolutionary Algorithm. In: Proceedings of the Congress on Evolutionary Computation. Piscataway, NJ, USA: IEEE Press, 2004.

MAMDANI, E. H., ASSILIAN, S. An Experiment in Lingüistic Synthesis with a Fuzzy Logic Controller. International Journal of Man-Machine Studies, 7(1): 1-13, 1975.

PASSINO, K. M.; YURKOVICH, S., Fuzzy Control, Adison Wesley Longman.-1997

PEDRYCZ, W. Fuzzy Control and Fuzzy Systems. England, Research Studies Press, 1989.

SÁ, LEONARDO BRUNO DE; "Síntese de Filtros com Baixa Sensibilidade Utilizando Técnicas Evolutivas" Tese de Doutorado – Rio de Janeiro: UFRJ/COPPE, 2009.

SAMS**,** HOWARD W. and Co., Indianapolis, 1st edition. Yamamoto et al., 1994

SHAW, I. S.; SIMÕES, M.G., Controle e Modelagem Fuzzy. Editora Edgard Blücher Ltda – 1ª edição 1999

SOUZA, F. J. Modelos Neuro-fuzzy Hierárquicos. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, Abril 1999.

TANSCHEIT, R. Fundamentos de Lógica Fuzzy e Controle Fuzzy. Artigo do Curso de Lógica Fuzzy. Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, 2000.

THOMPSON, A. Silicon Evolution. In: Proceedings of Genetic Programming. [S.l.]: MIT Press, 1996. p. 444–452.

YAKOUT, M.A.; El-Masry, E.I.; Abdel-Fattah, A.I. Hardware realization of analog CMOS current-mode minimum circuit. Radio Science Conference, 1998

YAMAKAWA, T. A Fuzzy Inference Engine in Nonlinear Analog Mode and Its Application to Fuzzy Logic Control. IEEE Trans. on Neural Networks, Vol.4, no. 3, pp. 496-522, May 1993.

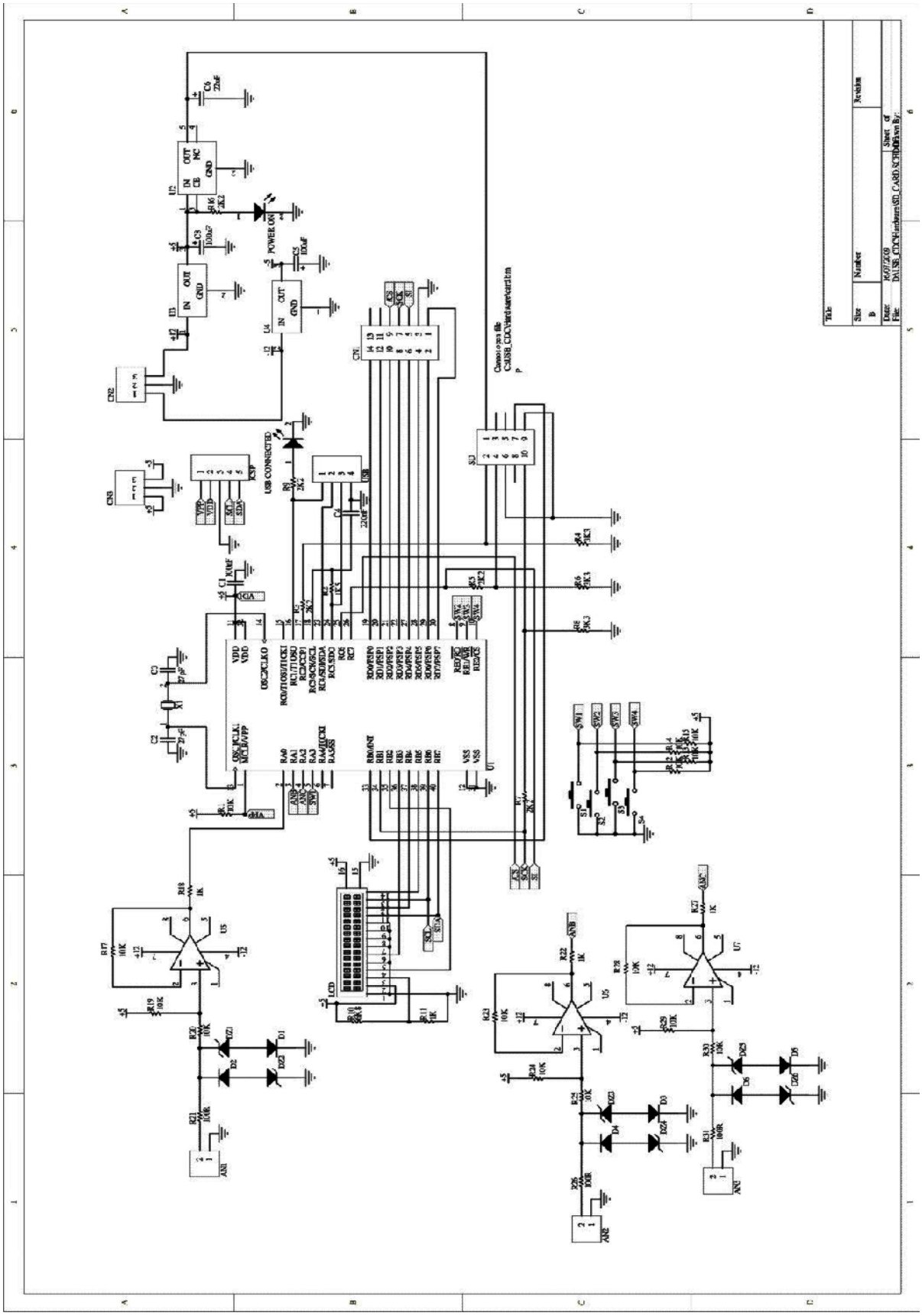
YAO, X.; HIGUCHI, T. Promises and Challenges of Evolvable Hardware. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Press, Piscataway, NJ, USA, v. 29, n. 1, p. 87–97, feb. 1997.

ZADEH, L. A, Fuzzy sets, Information and Control, vol. 8, pp. 338-353, 1965.

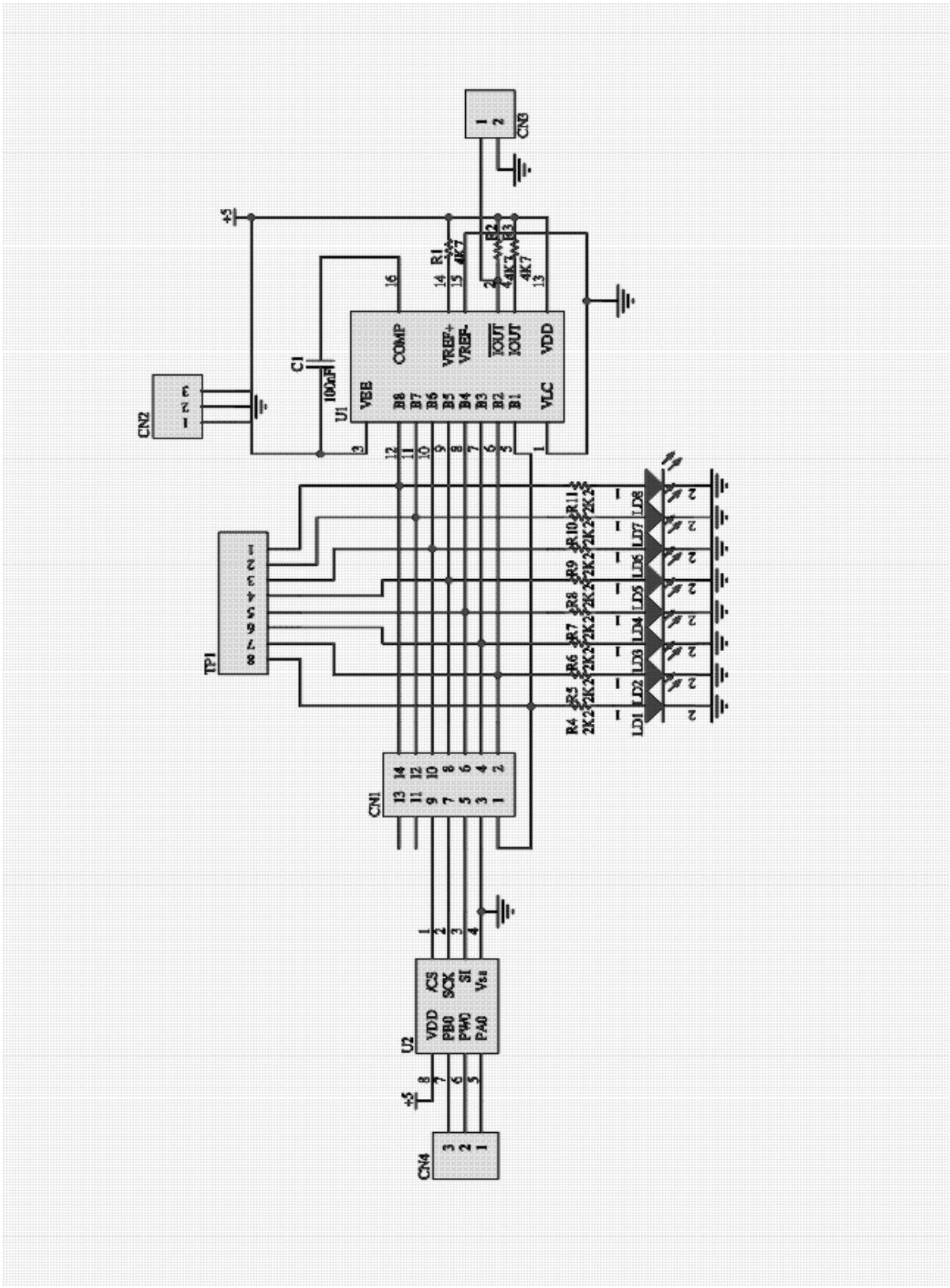
ZEBULUM, R. S.; SANTINI, C; M. VELLASCO; “A Reconfigurable Plataform for the Automatic Synthesis of AnalogCircuits” , Proceedings of the Second NASA DoD Workshop on Evolvable Hardware, pp.91-98, IEEE Computer press., July, 2000.

ZEBULUM, R. S.; PACHECO, M. A.; VELLASCO, M. M. B. R. Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms. USA: CRC Press LLC, 2002. 328 p. (The CRC Press international series on computacional intelligence).

APÊNDICE A – Esquemas Elétricos da Plataforma.



Esquema 1 - Esquema Elétrico da Unidade Central



Esquema 2 - Esquema Elétrico da Unidade Saída Sinal