UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE UM CONTROLADOR DE TEMPERATURA BASEADO EM LÓGICA FUZZY UTILIZANDO UM MICROCONTROLADOR

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

EDUARDO KLAUS BARG

BLUMENAU JUNHO/2002

2002/1-27

PROTÓTIPO DE UM CONTROLADOR DE TEMPERATURA BASEADO EM LÓGICA FUZZY UTILIZANDO UM MICROCONTROLADOR

EDUARDO KLAUS BARG

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Miguel Alexandre Wisintainer — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Miguel Alexandre Wisintainer

_____

Prof. Antônio Carlos Tavares

_____

Prof. Sérgio Stringari

SUMÁRIO

RESUMO

Este trabalho apresenta a especificação e implementação de um protótipo de *hardware* e *software*, utilizando uma arquitetura de microcontrolador de 8 bits, para implementar o processo de controle de temperatura de um ambiente, utilizando os conceitos de Lógica Fuzzy. Todos os dados deste processo serão disponibilizados em um *software* de supervisão. Como objetivo secundário é apresentado um estudo sobre a arquitetura de *hardware* utilizada e os conceitos gerais de Lógica Fuzzy.

ABSTRACT

This work shows the specification and implementation of a hardware and software prototype, using an 8 bit microcontroller architecture, to implement the temperature control process of an environment using the Fuzzy Logic concepts. All process data will be avaliable to a supervisory software. A secondary objective is show a study about the hardware architecture used and the general Fuzzy Logic concepts.

1 INTRODUÇÃO

Controles tradicionais de sistemas, neste caso, controle de temperatura, são em geral baseados em modelos matemáticos que descrevem o sistema de controle usando uma ou mais equações diferenciais que definem a resposta do sistema para suas entradas; tais sistemas são freqüentemente implementados pelo chamado controlador "PID" (proporcional-integral-derivativo). Tais controladores são produtos de décadas de desenvolvimento e trabalho teórico e são altamente eficazes.

Se controladores PID e outros sistemas de controle de temperatura tradicionais são tão bem desenvolvidos, por que se preocupar com Lógica Fuzzy. Somente porque em alguns casos ela tem alguma vantagem: em muitos casos, como por exemplo, sistemas de freio ABS ou no controle de guindastes para descarga de *containers*, o modelo matemático do processo pode não existir ou pode ser muito "caro" em termos de poder de processamento computacional e memória - e um sistema baseado em regras empíricas pode ser mais efetivo.

Operadores humanos são capazes de controlar processos bastante complexos, baseados em informações imprecisas ou aproximadas a respeito desses processos. A estratégia adotada pelos operadores humanos é também de natureza imprecisa e geralmente possível de ser expressa em termos lingüísticos.

A Teoria de Conjuntos Fuzzy (Zadeh, 1965) e os conceitos de Lógica Fuzzy (Zadeh, 1973) podem ser utilizados para traduzir em termos matemáticos a informação imprecisa expressa por um conjunto de regras lingüísticas. Se um operador humano for capaz de articular sua estratégia de ação como um conjunto de regras da forma SE ENTÃO, um algoritmo passível de ser implementado em computador pode ser construído, conforme demonstrado em Mamdani (1974).

Baseado na descrição acima, este trabalho visa implementar um controle de temperatura utilizando o microcontrolador 80C552 da Philips. Segundo Anlauf (1993), o microcontrolador, também chamado de “microcomputador em um só chip”, reúne num único chip vários sistemas independentes, como contadores, unidade central de processamento, memória para programa e dados, entradas e saídas analógicas, portas seriais, entre outros. Com isso diminuem o tempo e o custo requeridos para o desenvolvimento do projeto. Tais

características tornam o microcontrolador ideal para o projeto de sistemas dedicados e, sobretudo compactos.

1.1 OBJETIVOS

O trabalho tem como objetivo principal especificar e implementar um protótipo de *hardware*, utilizando uma arquitetura de microcontrolador de 8 bits, para realizar o processo de controle de temperatura de um ambiente, utilizando os conceitos de Lógica Fuzzy.

Os objetivos específicos do trabalho são:

a) realizar simulações do comportamento do processo;

b) disponibilização dos dados do processo em um software de supervisão;

c) envio de *setpoints* ao processo através do software de supervisão.

1.2 ORGANIZAÇÃO DO TEXTO

O capítulo 1 apresenta a introdução do trabalho contendo alguns conceitos fundamentais sobre microcontroladores e lógica fuzzy, bem como a apresentação dos objetivos e a organização do texto.

O capítulo 2 faz um breve estudo sobre a Lógica Fuzzy, descrevendo-se os seus fundamentos e destacando-se a definição de conjuntos fuzzy e a sua normalização. Apresentam-se também as principais operações com conjuntos fuzzy, e os sistemas de controle fuzzy. Finaliza-se o capítulo com um estudo de utilização e perspectivas desta lógica.

O capítulo3 introduz os microcontroladores da família MCS51 e aspectos inerentes ao chip 80C552 que será utilizado no protótipo e os elementos principais desta tecnologia para melhor entendimento do protótipo.

2 LÓGICA FUZZY

Aristóteles, filósofo grego (384 - 322 a.C.), foi o fundador da ciência da lógica, e estabeleceu um conjunto de regras rígidas para que conclusões pudessem ser aceitas como logicamente válidas. O emprego da lógica de Aristóteles levava a uma linha de raciocínio lógico baseado em premissas e conclusões. Como um exemplo: se é observado que "todo ser vivo é mortal" (premissa 1), a seguir é constatado que "João é um ser vivo" (premissa 2), como conclusão temos que "João é mortal".

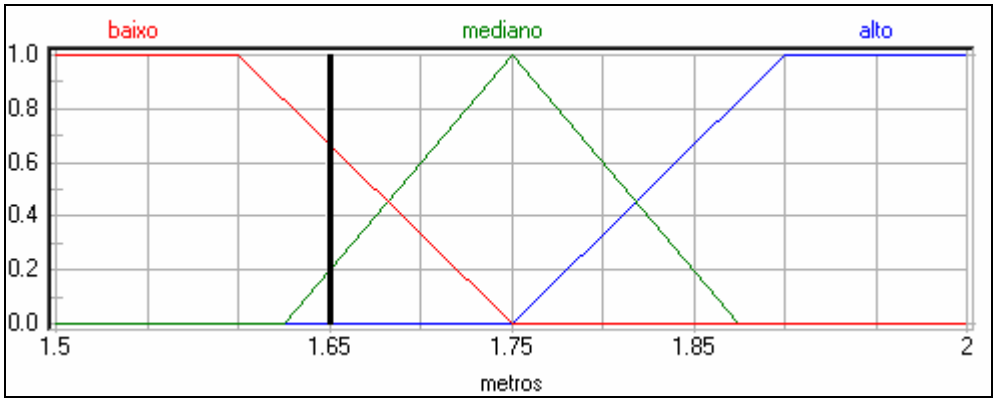
Desde então, a lógica Ocidental, assim chamada, tem sido binária, isto é, uma declaração é falsa ou verdadeira, não podendo ser ao mesmo tempo parcialmente verdadeira e parcialmente falsa. A lógica fuzzy viola estas suposições (Tarig, 2001).

A Lógica Fuzzy consiste em aproximar a decisão computacional da decisão humana, tornando as máquinas mais capacitadas a seu trabalho. Isto é feito de forma que a decisão de uma máquina não se resuma apenas a um "sim" ou um "não", mas também tenha decisões "abstratas", do tipo "um pouco mais", "talvez sim", e outras tantas variáveis que representem as decisões humanas. É um modo de interligar inerentemente processos analógicos que deslocam-se através de uma faixa contínua para um computador digital que podem ver coisas com valores numéricos bem definidos (valores discretos).

O conceito de conjunto Fuzzy foi introduzido, em 1965, por Lotfi A. Zadeh da Universidade da Califórnia, Berkeley (Zadeh, 1965). No meio da década de 60 do século XX, 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 o Professor Zadeh publicou em 1965 um artigo resumindo os conceitos dos conjuntos Fuzzy, revolucionando o assunto com a criação de sistemas Fuzzy.

Uma representação gráfica convencional de valores na lógica fuzzy é ilustrada na Figura 2.1, em que a altura das pessoas é representada na abscissa e três funções (baixo, mediano e alto) representam a classificação das pessoas quanto à altura.

FIGURA 2.1 – REPRESENTAÇÃO DOS VALORES NA LÓGICA FUZZY



Uma pessoa medindo 1,65 m de altura é considerada baixa, de acordo com o gráfico da Figura 2.1, mas está muito próxima de ser considerada de altura mediana. Entretanto, é possível observar que uma pessoa é totalmente baixa de 1,5 até 1,60 m (a curva “baixo” indica valor 1 e as demais valor 0 na faixa de valores indicada). A partir de 1,60 m a reta que define o valor baixo começa a decrescer, enquanto a reta que define a altura mediana começa a crescer.

Uma pessoa com 1,75 m é considerada de altura mediana e acima de 1,90 m é considerada alta. Essas medidas são totalmente imprecisas, variando de acordo com os conceitos de cada pessoa, região, cidade, país e outros. Em muitas situações porém, os valores inexatos são mais importantes e possuem significados mais expressivos do que os valores exatos.

2.1 FUNDAMENTOS DA LÓGICA FUZZY

A lógica fuzzy foi proposta por Lotfi A. Zadeh em 1965 como uma matemática que podia representar as incertezas do cotidiano (Cox, 1994); é basicamente uma linguagem que serve para descrever e analisar dependências imprecisas (Zadeh, 1973). Diante dos problemas da lógica booleana e dos recursos oferecidos pela lógica fuzzy, muitos pesquisadores passaram a utilizá-la como ferramenta para o desenvolvimento de sistemas inteligentes. Atualmente, há uma grande variedade de pesquisas envolvendo a lógica fuzzy.

Na teoria clássica dos conjuntos um elemento do universo de discurso (domínio) pertencente ou não ao referido conjunto, assume valores 0 ou 1 (Cox, 1994). Na teoria dos

conjuntos fuzzy existe um grau de pertinência de cada elemento que pode assumir qualquer valor dentro do intervalo [0,1].

O valor 1 representa completa pertinência e o valor 0 indica uma completa exclusão. Essa generalização aumenta significativamente o poder de expressão da função característica, onde esta função, diz o grau de pertinência μA(x) de um elemento x pertencente a um universo U com respeito a um conjunto A, onde A $\subset$ U. Quando μA(x)=0, representa nenhuma pertinência e μA(x)=1, representa pertinência total.

Para Rabuske (1995), os conjuntos difusos se contrapõem a assim chamada “lei da contradição”, onde valores variam simplesmente de verdadeiro para falso, de sim para não, permitindo assim a manutenção de proposições conflitantes.

2.2 OPERAÇÕES DOS CONJUNTOS FUZZY

Como nos conjuntos convencionais, existem operações especificamente definidas para combinar e modificar os conjuntos fuzzy (Cox, 1994). Estas funções são as ferramentas fundamentais da lógica fuzzy. A teoria originária dos conjuntos fuzzy foi fundamentada nos termos das três operações realizadas com conjuntos *(complemento, união e interseção*) que são equivalentes as operações da lógica booleana *(negação, ou*, e *e*).

Para esclarecer isto, temos o seguinte exemplo. Seja A (figura 2.2) um intervalo fuzzy entre 5 e 8, e B (figura 2.3) um número fuzzy em torno de 4. Temos as seguintes representações:

FIGURA 2.2 – REPRESENTAÇÃO GRÁFICA DE UM INTERVALO FUZZY

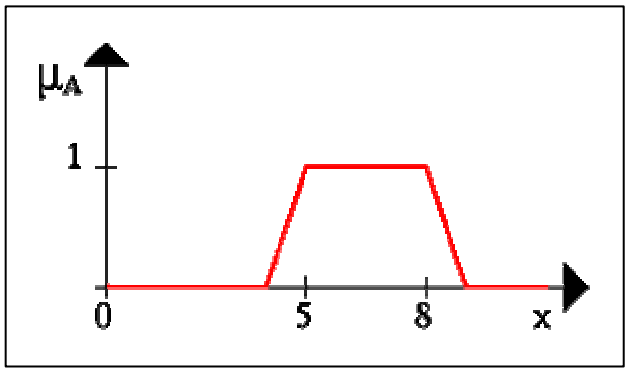
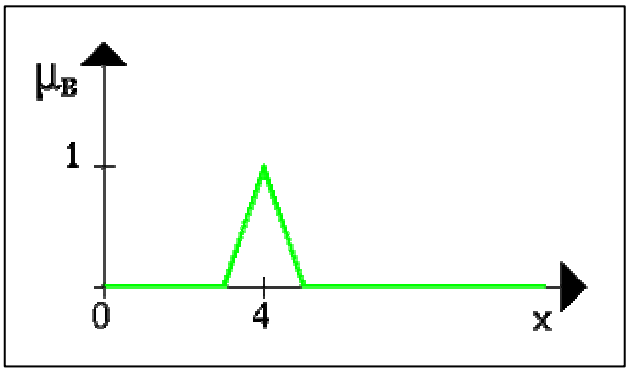


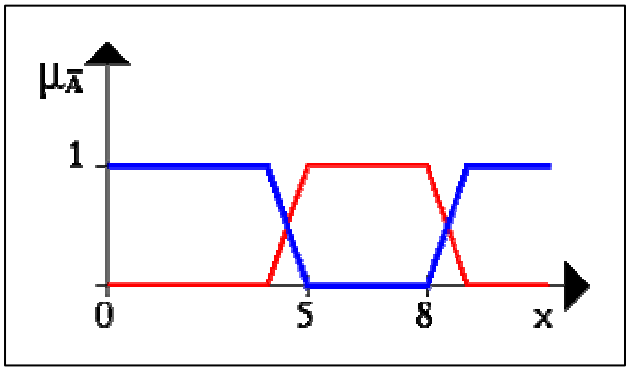
FIGURA 2.3 – REPRESENTAÇÃO GRÁFICA DE UM NÚMERO FUZZY



2.2.1 COMPLEMENTO

A operação complemento é utilizada para definir a função de pertinência oposta de um subconjunto, ou seja, o complemento do subconjunto A, definido como Ā, é formado pelos pontos opostos de A de dentro do intervalo [0, 1]. Essa operação, quando tratada nos extremos desse intervalo, é equivalente à operação “negação” da lógica booleana. A representação formal da operação complemento é descrita na equação: $\mu\bar{A}(x) = 1-\mu A(x)$. A sua representação gráfica é ilustrada na Figura 2.4.

FIGURA 2.4 – REPRESENTAÇÃO GRÁFICA DA OPERAÇÃO COMPLEMENTO

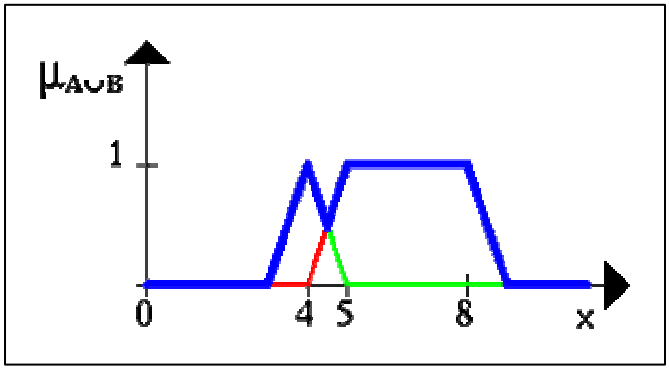


2.2.2 UNIÃO

A operação união é utilizada para associar dois subconjuntos, ou seja, a união do subconjunto A com B resulta em um subconjunto abrangendo os pontos máximos dos dois subconjuntos unidos. Essa operação, quando tratada nos extremos do intervalo [0, 1], é equivalente à operação “ou” da lógica booleana. A representação formal da operação união é

descrita na equação: $A \cup B \rightarrow \mu A(x) \cup \mu B(x) = \max(\mu A(x), \mu B(x))$. A sua representação gráfica é ilustrada na Figura 2.5.

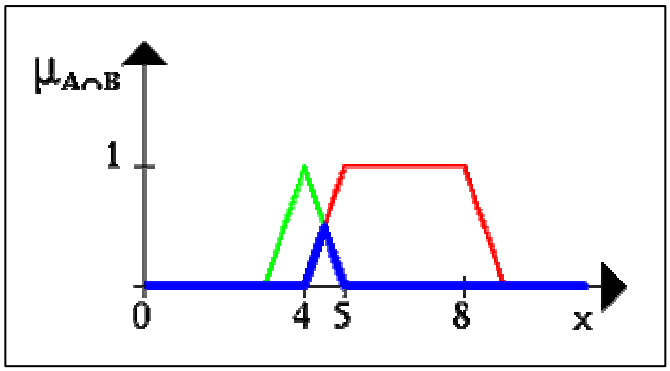
FIGURA 2.5 – REPRESENTAÇÃO GRÁFICA DA OPERAÇÃO UNIÃO



2.2.3 INTERSEÇÃO

A operação interseção é utilizada para definir a região comum entre dois subconjuntos, ou seja, a interseção do subconjunto A com B resulta em um subconjunto abrangendo os pontos que pertencem tanto ao subconjunto A quanto ao subconjunto B. Essa operação, quando tratada nos extremos do intervalo [0, 1], é equivalente à operação “e” da lógica booleana. A representação formal da operação interseção é descrita na seguinte equação: $A \cap B \rightarrow \mu A(x) \cap \mu B(x) = \min(\mu A(x), \mu B(x))$. A sua representação gráfica é ilustrada na Figura 2.6.

FIGURA 2.6 – REPRESENTAÇÃO GRÁFICA DA OPERAÇÃO INTERSEÇÃO



2.3 VARIÁVEIS LINGÜÍSTICAS

Para expressar conceito é muito comum o uso de elementos qualitativos ao invés de valores quantitativos. Elementos típicos incluem “mais ou menos”, “alto”, “não muito”, “médio”, entre outros.

Estas idéias são ditas pela definição de variável lingüística. Uma variável lingüística tem por característica assumir valores dentro de um conjunto de termos lingüísticos, ou seja, palavras ou frases (Tarig, 2001).

Segundo Pereira (1995), variáveis lingüísticas, são variáveis cujos valores são palavras em linguagem natural representadas em conjuntos difusos. Por exemplo, uma variável lingüística altura poderá assumir um dos membros do conjunto {muito alto, alto, médio, médio alto, baixo, muito baixo}. Para se atribuir um significado aos termos lingüísticos, associa-se a cada um deles um conjunto fuzzy definido sobre um universo de discurso comum.

A principal função das variáveis lingüísticas é fornecer uma maneira sistemática para uma caracterização aproximada de fenômenos complexos ou mal definidos. Em essência, a utilização do tipo de descrição lingüística empregada por seres humanos, e não de variáveis quantificadas, permite o tratamento de sistemas que são muito complexos para serem analisados através de termos matemáticos convencionais.

Para Pacheco (1991) o raciocínio humano é por natureza “aproximado”, e qualquer técnica de modela-lo diferentemente está desprezando a principal vantagem humana, a de tratar diretamente com conceitos inexatos. Daí a importância das variáveis lingüísticas.

2.4 EXPRESSÃO FUZZY DO CONHECIMENTO

Segundo Tarig (2001), uma das formas mais comuns de se expressar o conhecimento é por meio de regras do tipo condição-ação. Exemplificando, um conjunto de condições que descrevem uma parcela observável das saídas do processo, é associado a uma ação de controle que irá manter ou elevar o processo às condições de operações desejadas.

A idéia aqui é representar o conhecimento por meio de um conjunto de regras nas quais as condições são dadas a partir de um conjunto de termos lingüísticos associados a variáveis de entrada/saída do processo (as quais são entradas do controlador).

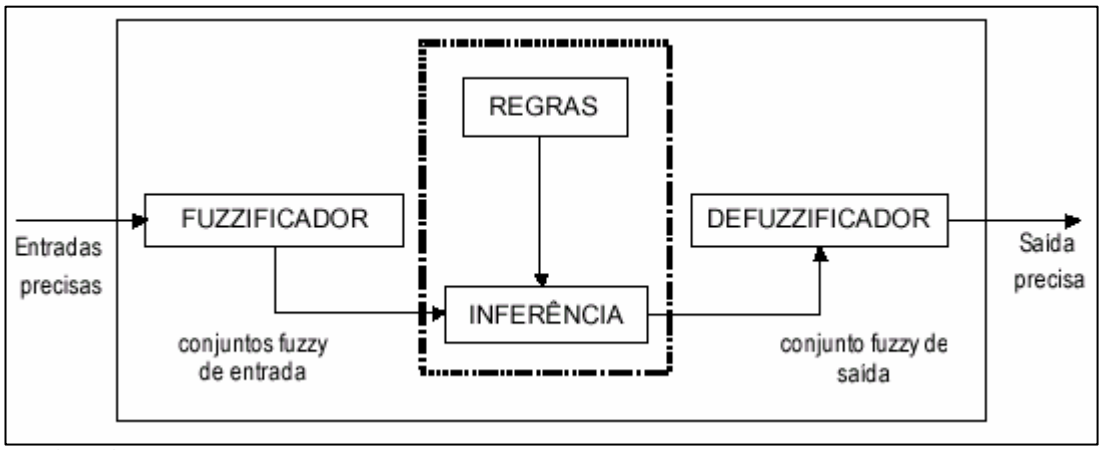
As ações de controle ou as saídas são expressas de modo similar para cada variável de controle (saída). Regras do tipo *se-então-senão* são freqüentemente chamadas de regras de controle fuzzy.

2.5 SISTEMA DE CONTROLE FUZZY

Segundo Welstead (1994), a combinação de conjuntos fuzzy definidos por variáveis lingüísticas de entrada e saída, junto com o conjunto de regras de controle fuzzy, que por sua vez, ligam um ou mais conjuntos fuzzy de entrada a um conjunto fuzzy de saída, é compreendido como um sistema de controle fuzzy.

O controle fuzzy não necessita da modelagem do processo, e sim, da modelagem das ações a partir de um conhecimento de um especialista. Essa é, portanto, uma abordagem diferente dos métodos convencionais de controle de processos, pois os mesmos são desenvolvidos via modelagem matemática dos processos de modo a derivar as ações de controle como função do estado do processo. A estrutura básica de um controlador fuzzy está ilustrada na Figura 2.7.

FIGURA 2.7 – SISTEMA DE CONTROLE FUZZY



Fonte: Tarig (2001)

O núcleo do controlador só analisa variáveis fuzzy. As informações têm que ser transformadas na forma fuzzy ou “fuzzificadas” (transformadas em conjuntos fuzzy). A interface de “fuzzificação” recebe os valores das variáveis de entrada (vindo dos sensores), faz um escalonamento para dimensionar os valores a universos discursos normalizados e “fuzzifica” os valores (transformando números em conjuntos fuzzy), para torná-los instâncias de variáveis lingüísticas (Tarig, 2001). As regras caracterizam as estratégias de controle e seus objetivos.

O procedimento de inferência atua sobre os dados fuzzy de entrada, juntamente com as regras, para inferir as ações de controle fuzzy, usando o operador de implicação fuzzy e as regras de inferência da lógica fuzzy. O “defuzzificador” atua sobre as ações de controle fuzzy inferidas, transformando-as em ações de controle não fuzzy, efetuando, em seguida, um escalonamento para compatibilizar os valores normalizados vindos do passo anterior com os valores dos universos de discursos reais das variáveis.

Deve-se determinar uma ação de controle não-fuzzy para ser enviada ao controlador logo após se inferir a ação de controle fuzzy. A ação de controle não-fuzzy escolhida deve ser a que represente melhor a decisão fuzzy. Não há nenhum procedimento sistemático para escolher a estratégia de “defuzzificação”.

2.6 INTERFACE COM O PROCESSO

Nesta seção considera-se a definição das variáveis de controle e métodos de “defuzzificação”, necessários para se estabelecer a conexão do controlador fuzzy com um processo não-fuzzy.

2.6.1 VARIÁVEIS DE CONTROLE

Em controladores fuzzy de caráter geral como o apresentado aqui, as variáveis de entrada são o erro, gerado a partir da diferença entre o sinal de referência e a saída do processo, e a variação do erro, normalmente gerada a partir da diferença entre o erro atual e o erro anterior. A variável de saída do controlador é a variação no controle**.**

A opção por uma saída incremental, ao invés de absoluta, é mais condizente com o raciocínio empregado por operadores humanos e, além disso, proporciona uma economia em termos do universo da saída.

Estabelecendo uma ligação com as seções anteriores, as variáveis fuzzy erro e variação do erro são subconjuntos fuzzy em seus respectivos universos.

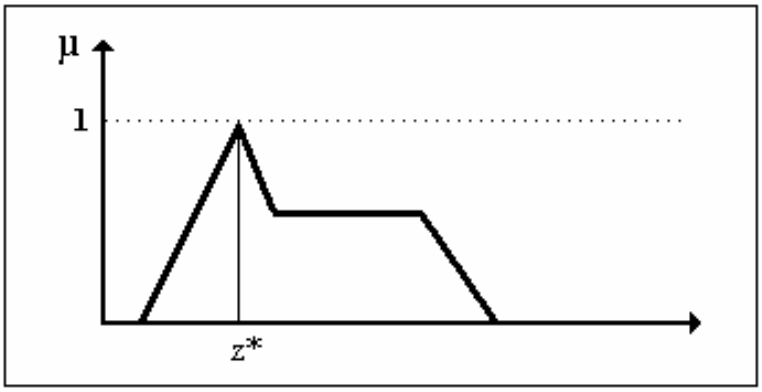
2.6.2 MÉTODOS DE DEFUZZIFICAÇÃO

Conforme Tarig (2001), a saída do controlador fuzzy é um subconjunto fuzzy do universo da saída. Como o processo requer um sinal não-fuzzy em sua entrada, deve-se fazer uma interpretação daquele conjunto fuzzy.

Para Ross (1995), a defuzzificação é o processo pelo qual um conjunto fuzzy tem a sua abrangência representada por um simples número. Segundo ele, existem pelo menos sete métodos que tem sido pesquisados e popularizados com o decorrer do tempo. Dentre eles destacam-se o princípio da maior pertinência, o método centróide, ou centro de gravidade, e a média da pertinência máxima.

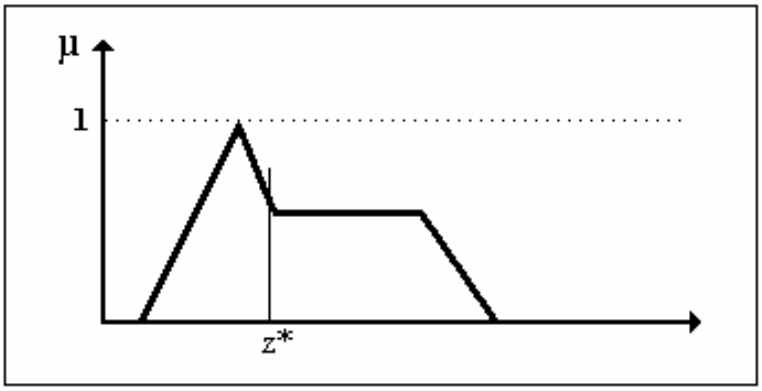
O princípio da maior pertinência, também conhecido como método da altura, limita-se ao pico da função. A figura 2.8 mostra o comportamento deste método.

FIGURA 2.8 – REPRESENTAÇÃO GRÁFICA DO MÉTODO DE DEFUZZIFICAÇÃO DA MAIOR PERTINÊNCIA



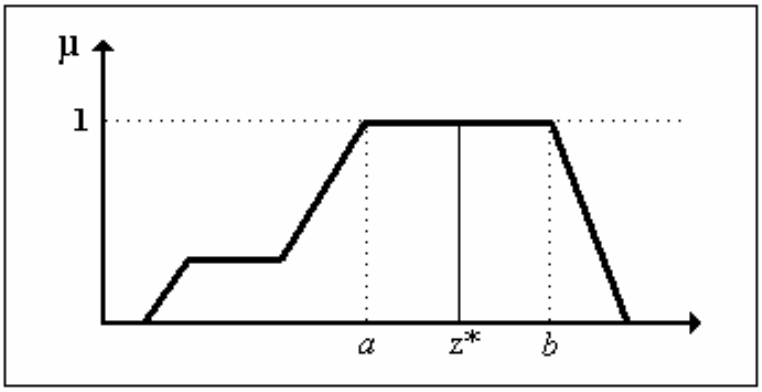
Já o método da centróide, também chamado de centro da gravidade, é o mais usado de todos os métodos de defuzzificação. Este método tem como saída o ponto que divide a área da função de pertinência em duas partes iguais. A figura 2.9 ilustra o método da centróide.

FIGURA 2.9 – REPRESENTAÇÃO GRÁFICA DO MÉTODO DE DEFUZZIFICAÇÃO DA CENTRÓIDE



E finalmente, o método da média da pertinência máxima, também chamada de média dos máximos, é quase idêntico ao primeiro método. Este parte do princípio de que a localização da maior pertinência pode não se limitar a um único elemento, mas sim, a diversos. O valor da defuzzificação é dado pela média de *a* e *b*, que são definidos na figura 2.10.

FIGURA 2.10 – REPRESENTAÇÃO GRÁFICA DO MÉTODO DE DEFUZZIFICAÇÃO DA MÉDIA DOS MÁXIMOS



Na realidade, com o Centro de Gravidade nunca se consegue obter os valores extremos do universo, devido a própria natureza do método. Isto pode dar origem a uma resposta mais lenta do que a obtida com a Média dos Máximos.

2.7 EXEMPLOS DE EMPREGO DE SISTEMAS FUZZY RECENTES

Em 1990, a lógica fuzzy era implementada em grande escala de aplicações eletrônicas para o lar no Japão, tais como refrigeradores, aspiradores de pó, lavadoras, secadores, panelas para cozinhar arroz e ar condicionados (Toshinori, 1994). Mas a aplicação da lógica fuzzy não se restringiu somente ao lar. Outras aplicações recentes tiveram a lógica fuzzy implementada, dentre elas podemos citar:

a) vídeo câmeras: quatro funções das vídeo câmeras são baseadas em princípios da lógica fuzzy, o ajuste automático do foco, exposição automática, balanceamento automático do branco e sistemas de estabilização de imagem. A técnica do autofoco utilizava capacidade aproximada de inteligência em regras fuzzy para controlar a velocidade do motor, melhorando a qualidade do foco e reduzindo seu tempo. A estabilização da imagem detectava suficientemente os movimentos indesejados causados pelos solavancos e tremores causados pelo movimento das mãos, corrigindo então, muitas das imagens falhas;
b) automobilística: o primeiro dispositivo de controle fuzzy em um carro foi vendido em 1991. Outros estudos incluíam a injeção de combustível e sistemas de transmissão e frenagem (ABS);
c) espaço aéreo: um número significativo de algoritmos difusos e aplicações para o controle de problemas aéreos tem sido implementados pela NASA nos últimos anos. Durante 1992 e 1993, artefatos experimentais voavam com dispositivos controladores de temperatura baseados na lógica fuzzy.

2.8 PERSPECTIVAS DA LÓGICA FUZZY

Diversas áreas estão sendo beneficiadas pela tecnologia decorrente da lógica Fuzzy. O controle de processos industriais foi a área pioneira, sendo as primeiras experiências datadas de 1975 quando foi demonstrado, que um controlador Fuzzy muito simples conseguiu controlar eficientemente uma máquina a vapor (Mamdani, 1974).

Nos últimos anos o potencial de manuseio de incertezas e de controle de sistemas complexos tornados possíveis pela lógica Fuzzy, estão sendo combinados com redes neurais artificiais, que por sua vez, possuem características de adaptação e aprendizagem. Estes controladores são conhecidos como neurofuzzy (Kosko, 1992).

Com certeza estes sistemas deverão proporcionar uma significativa contribuição para os sistemas de automação e controle do futuro, principalmente em controle de processos.

3 MICROCONTROLADORES

Este capítulo apresenta os conceitos e demais aspectos da arquitetura dos microcontroladores da família MCS51.

Existem muitas dúvidas quanto a diferença entre um microprocessador e um microcontrolador. Um microcontrolador é um componente que tem, num único chip, além de uma CPU, elementos tais como memórias ROM e RAM, temporizadores, contadores, canais de comunicação e conversores analógico-digitais (Silva Júnior, 1988).

Esta característica diferencia os sistemas baseados em microcontroladores daqueles baseados em microprocessadores, onde normalmente se utilizam vários componentes para implementar essas funções. Com isso, os microcontroladores permitem a implementação de sistemas mais compactos e baratos do que aqueles baseados em microprocessadores.

A Intel iniciou a produção do 8051 em 1981. Diversos fabricantes produzem microcontroladores da família 8051 (Intel, AMD, Atmel, Dallas, OKI, Matra, Philips, Siemens, SMC, SSI). Cada empresa procurou melhorar o desempenho do 8051, adicionando novos recursos ao mesmo, mas a arquitetura básica continua a mesma.

3.1 PRINCIPAIS CARACTERÍSTICAS

Citaremos a seguir as características básicas que formam o núcleo da família 8051:

d) freqüência de *clock* de 12 MHz, com algumas versões que alcançam 40 MHz;
e) até 64KB de memória de dados externa;
f) até 64KB de memória de programa, independente da anterior;
g) 128 bytes de RAM interna;
h) 4 portas bidirecionais de I/O;
i) 2 temporizadores /contadores de 16 bits;
j) 5 fontes de interrupção (dois *timers*, dois pinos externos e o canal de comunicação serial), com 2 níveis de prioridade;
k) oscilador de *clock* interno;
l) porta serial *full-duplex*.

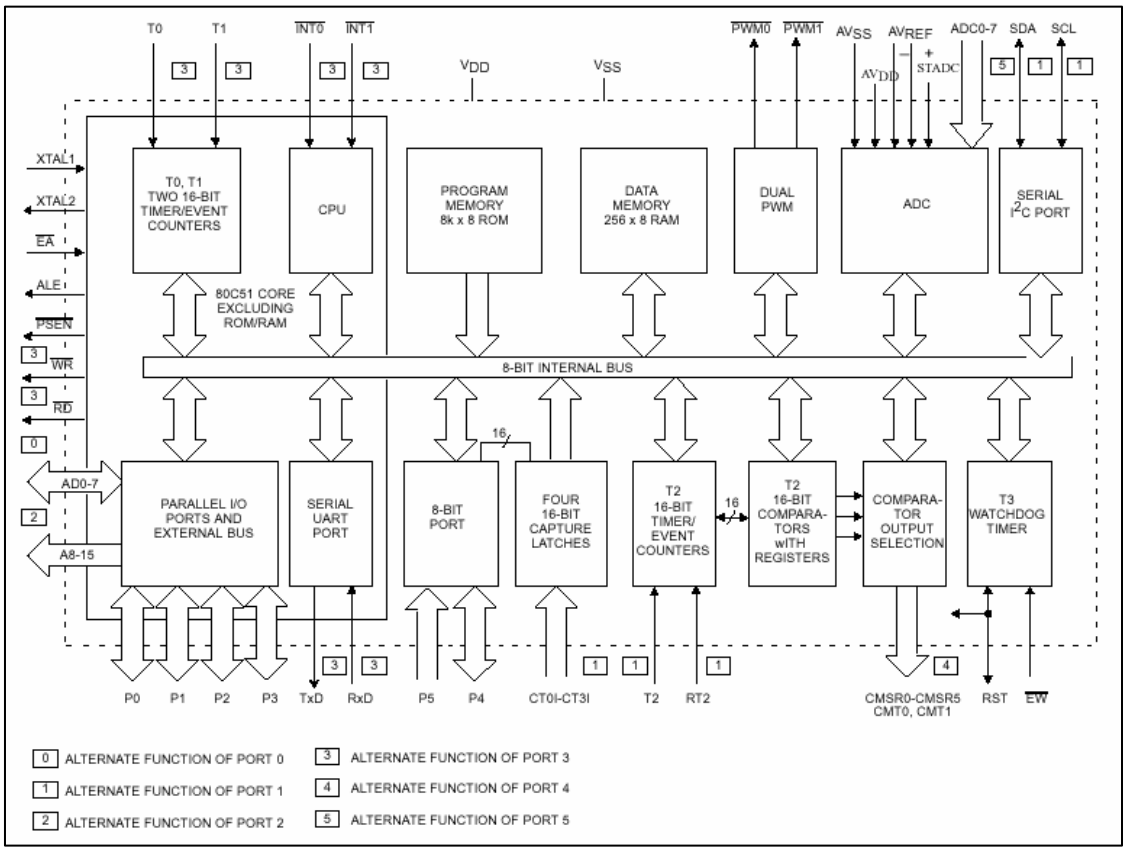
No caso deste trabalho será utilizado um variante do 8051, o 80C552 produzido pela Philips. O 80C552 possui as mesmas características básicas do 8051, e foi acrescido dos seguintes componentes:

a) um temporizador / contador de 16 bits adicional;
b) 256 bytes de RAM interna;
c) 8 entradas analógicas multiplexadas de 10 bits de resolução;
d) 2 saídas de PWM (*Pulse Width Modulation*), com 8 bits de resolução;
e) *watchdog timer* interno;
f) porta de I/O para interface serial do tipo I²C;
g) freqüência de clock de 16MHz.

3.2 DIAGRAMA DE BLOCOS INTERNO DO 80C552

A arquitetura do microcontrolador 80C552 está ilustrada na figura 3.1.

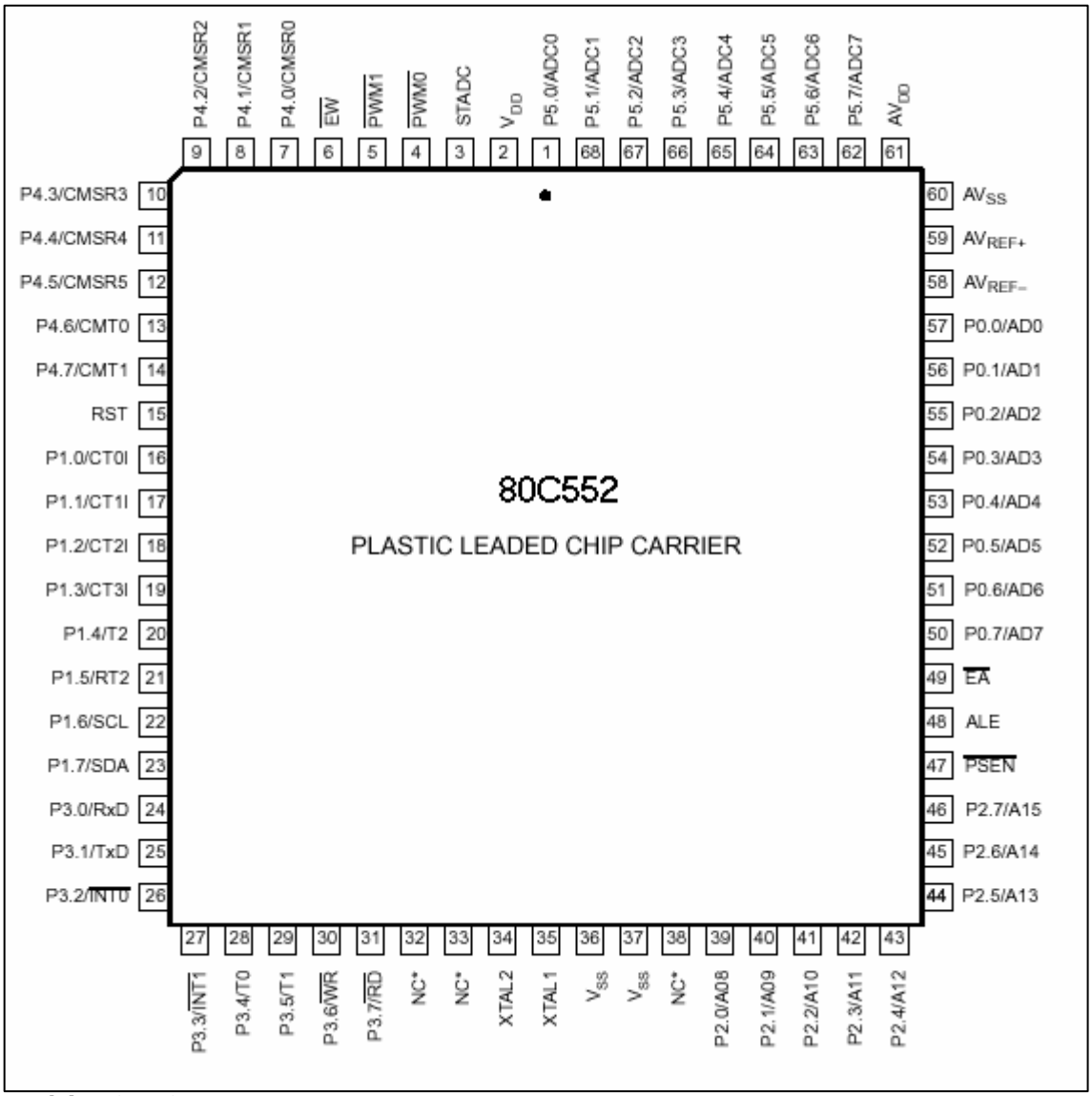
FIGURA 3.1 – DIAGRAMA DE BLOCOS DO 80C552



Fonte: Philips (1996)

A figura 3.2 exibe a pinagem de um microcontrolador 80C552 com encapsulamento PLCC (*Plastic Leaded Chip Carrier*).

FIGURA 3.2 – PINAGEM DO 80C552



Fonte: Philips (1996)

As portas P0 e P2, ficam comprometidas com o uso de memória externa, assim como os pinos P3.6 e P3.7. O sinal ALE (*Address Latch Enable*) permite fazer a demultiplexação de dados e endereços na porta P0. Através do sinal PSEN (*Program Storage Enable*), o microcontrolador informa o mundo externo se a operação em andamento é uma leitura de instrução (acesso à memória de programa) ou um acesso à memória de dados. Este sinal permite que o processador tenha duas regiões distintas de memória externa, uma para armazenar código e outra para dados. Ambas ocupam os endereços de 0 a FFFFH (64 kB),

num total de 128 kB. O pino EA é um sinal de entrada, através do qual o usuário escolhe se será utilizada a memória ROM interna ou se todo o programa será armazenado externamente.

3.3 DIFERENÇAS COM RELAÇÃO AO 8051

Tendo em vista que a arquitetura básica da família MCS-51 é bem conhecida, e já foi amplamente discutida em Klitzke (1999) e Silva Júnior (1990), será apresentado somente um estudo das características que diferem o microcontrolador 80C552 do 8051.

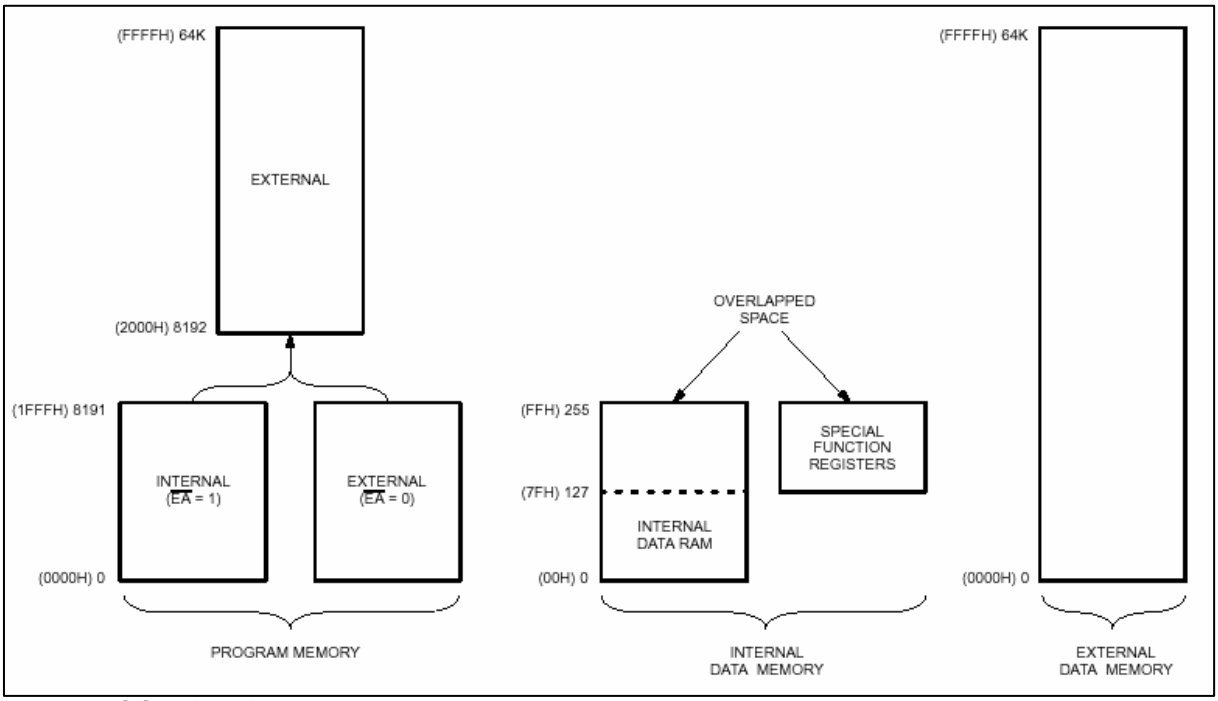
3.3.1 MEMÓRIA DE PROGRAMAS

O 80C552 possui 8K bytes de memória de programa *on-chip* que pode ser extendida até 64K bytes com memórias externas, como é mostrado na figura 3.3. O pino EA serve para informar ao microcontrolador onde buscar o programa a ser executado, seja na memória de programa interna ou em uma externa. Deve ser aterrado (*low*) ou ligado ao VCC (*high*) antes do *reset* do microcontrolador. Se durante o *reset* do microcontrolador, EA estiver em nível *low*, o microcontrolador executará o programa que estiver na memória de programa externa, caso contrário executará o programa na memória interna. Os endereços de memória de programas de 0003H à 0073H, são utilizados pelas rotinas de interrupção.

3.3.2 MEMÓRIA DE DADOS

A memória de dados interna é dividida em 3 setores: os 128 bytes mais baixos da RAM, os 128 bytes mais altos e a área de 128 bytes dos registradores de funções especiais (SFR's). Os 128 bytes mais baixos podem ser endereçados direta ou indiretamente. Já as posições de RAM de 128 a 255 e os SFR's, compartilham a mesma área de endereços, mas são acessados de formas diferentes. Os SFR's somente são diretamente endereçáveis, já os 128 bytes mais altos de RAM são acessados somente de forma indireta. Todos os outros aspectos da RAM interna são idênticos ao 8051.

FIGURA 3.3 – MAPA DE MEMÓRIA DO 80C552

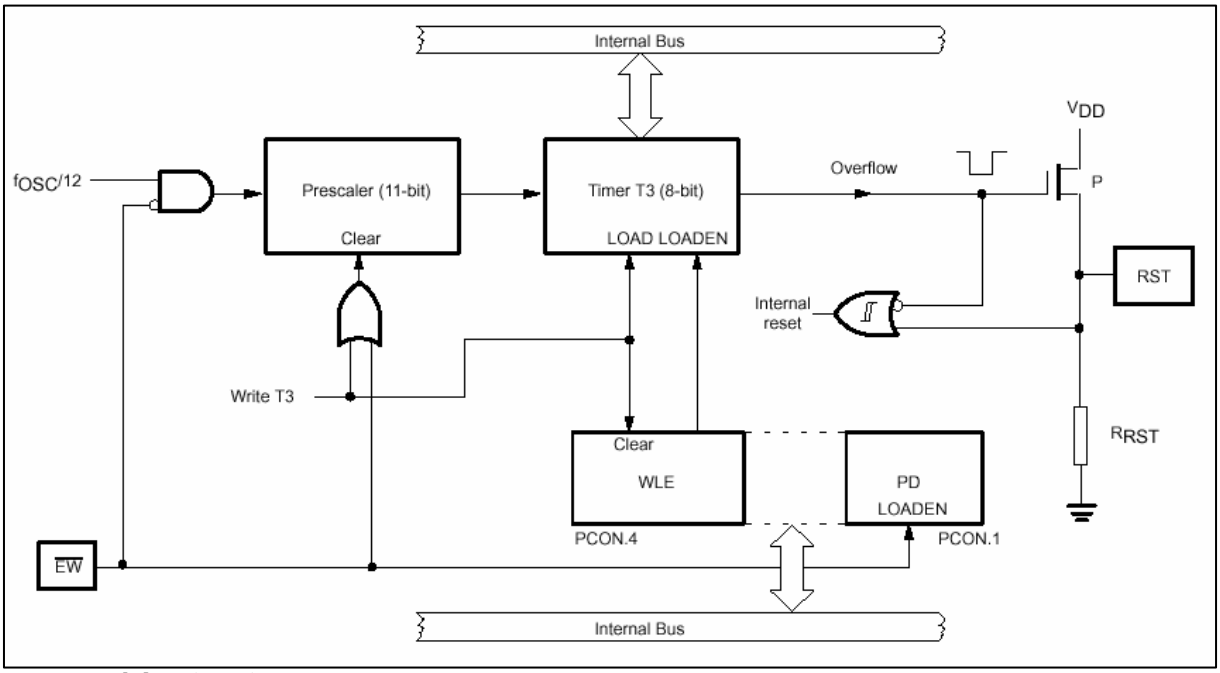


Fonte: Philips (1996)

3.3.3 O *WATCHDOG TIMER*

Além dos temporizadores padrão o 80C552 também incorpora um *watchdog timer*. *Watchdog* significa “cão de guarda”, analogia perfeita ao funcionamento deste dispositivo, pois sua tarefa consiste em supervisionar a CPU. O objetivo de um *watchdog timer* é resetar o microcontrolador se este entrar em um estado de erro (possivelmente causado por ruídos elétricos). Quando habilitado, o circuito de *watchdog* irá gerar um reset no microcontrolador, se o programa do usuário falhar em reinicializar o temporizador do *watchdog* dentro de um intervalo de tempo, conhecido como intervalo do *watchdog.* A figura 3.4 mostra o diagrama do circuito do *watchdog*.

FIGURA 3.4 – DIAGRAMA DO CIRCUITO DE WATCHDOG



Fonte: Philips (1996)

3.3.4 INTERFACE SERIAL I²C

Há aproximadamente 20 anos atrás a Philips, voltada a simplificar a comunicação digital entre dois dispositivos, desenvolveu um sistema bastante simples designado por I²C. Ele utiliza somente duas linhas de comunicação denominadas por: *serial data* SDA e *serial clock* SCL. As linhas SDA e SCL carregam somente informações digitais, e portanto operam dentro dos limites de 0 a 5 volts. Para que este sistema funcione corretamente, alguns protocolos bem definidos devem ser obedecidos, como por exemplo: o sinal de *start* (início da transmissão), o sinal de *stop* (final da transmissão), etc. A figura 3.5 ilustra a típica configuração de um barramento I²C.

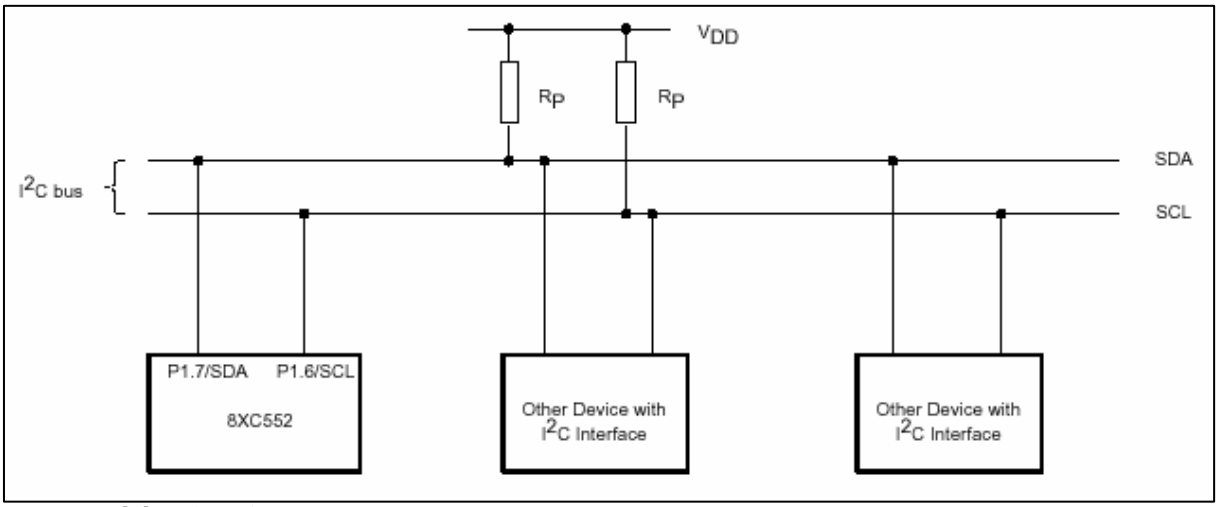
A seguir são apresentadas as noções básicas desta interface.

3.3.4.1 LINHA DE CLOCK

Este sinal é que fornece a cadência para que a transmissão serial seja entendida. Por transmissão serial entende-se que as informações serão enviadas uma após a outra - serialmente - atravéz da mesma linha digital. Nesta tarefa é a linha de *clock* que identifica

quando um dado (um bit) pode ser considerado válido - esta situação sempre é definida quando a linha de *clock* está *high* (nível lógico 1). Portanto, sempre que a linha de clock SCL estiver *high*, sabe-se que a linha SDA possui um dado válido (nível lógico 1 ou 0).

FIGURA 3.5 – BARRAMENTO I²C



Fonte: Philips (1996)

3.3.4.2 SINAL DE START

Para dar início a uma transmissão o seguinte protocolo deverá ser reconhecido: enquanto a linha SCL se mantém em nível lógico alto, a linha SDA deve passar de nível alto para baixo (transição). Esta é a *Start condition*.

3.3.4.3 ENDEREÇAMENTO

A interface I²C permite conexão com diversos periféricos que passam a ser identificados por um endereço. Para este propósito foram reservados 10 bits após o sinal de *start*. Somente ao circuito endereçado é que será dirigida a transferência de dados.

3.3.4.4 SINAL DE STOP

Para finalizar uma transmissão o seguinte protocolo deverá ser reconhecido: enquanto a linha SCL se mantém em nível lógico alto, a linha SDA deve passar de nível baixo para alto (transição). Esta é a *Stop condition*.

3.3.4.5 MASTER – SLAVE

Na interface serial I²C, podemos distinguir dois circuitos: a) o principal - encarregado de gerenciar o sistema - (normalmente o microprocessador) - designado por *master* (mestre), e b) o(s) secundário(s) que serão comandados por ele - designados por *slaves* (escravos).

O protocolo do I²C permite a transferência de dados bidirecional entre mestres e escravos, bem como, possui um barramento *Multimaster* (não há apenas um mestre central). Para tanto é feito o controle de transmissões simultâneas entre os mestres, sem que haja a corrupção dos dados no barramento.

3.3.4.6 TAXA DE TRANSFERÊNCIA

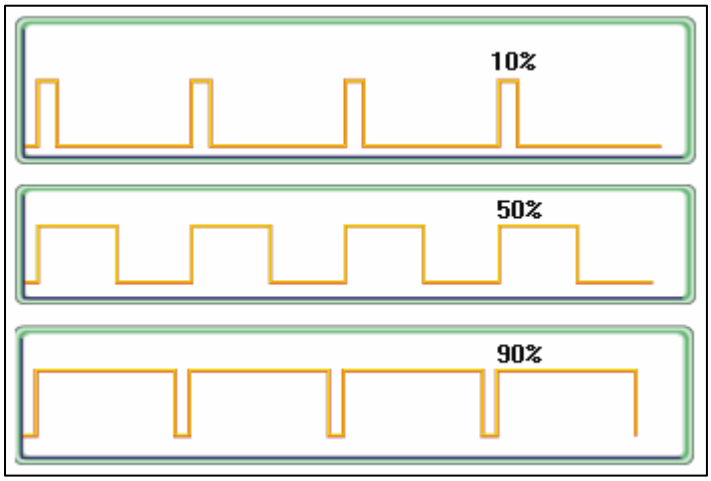
Uma das vantagens do padrão I²C é que ele não fixa a velocidade de transmissão (freqüência), pois ela será determinada pelo circuito *master* (transmissão do SCL).

3.3.5 SAÍDAS PWM

PWM (*pulse width modulation*), ou modulação por largura de pulso é uma técnica poderosa para controlar circuitos analógicos utilizando as saídas de microcontroladores. Controlando circuitos analógicos digitalmente pode-se reduzir drasticamente custos e consumo de energia. Por isso mais e mais microcontroladores passaram a incluir controladores de PWM *on-chip* para facilitar a implementação. O 80C552 possui 2 saídas de PWM.

Basicamente o PWM é uma forma de codificar digitalmente sinais analógicos (Barr, 2001). Através do uso de contadores de alta resolução o *duty cycle* (quantidade de tempo que o pulso está em nível *high* (1 lógico)) de uma onda quadrada é modulado para codificar um nível específico do sinal analógico. A figura 3.6 mostra primeiramente um *duty cycle* de 10%, em seguida mostra um *duty cycle* de 50% e depois um *duty cycle* de 90%. Estas três diferentes saídas de PWM codificam diferentes valores de sinais analógicos. Por exemplo, para uma fonte de 5V, um *duty cycle* de 10% resultaria em um sinal de 0.5V.

FIGURA 3.6 – REPRESENTAÇÃO DE SAÍDAS PWM

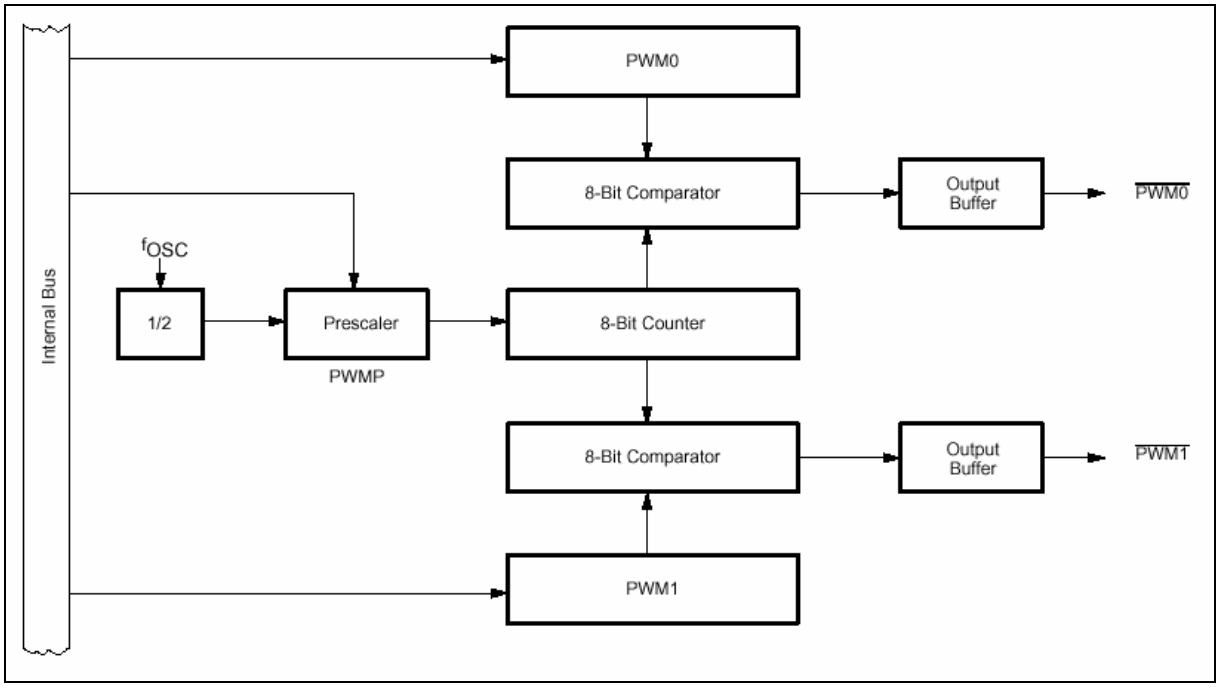


Fonte: Barr (2001)

No 80C552 a freqüência de repetição é definida por um registrador de pré-escala de 8 bits, chamado PWMP, que gera o *clock* para o contador. Ambos canais de PWM usam o mesmo registrador de pré-escala e o mesmo contador (Philips, 1996). O valor deste contador de 8 bits, é comparado ao conteúdo de dois registradores: PWM0 e PWM1. Se o conteúdo destes registradores for maior que o valor do contador, o pino de saída do microcontrolador correspondente, PWM0 ou PWM1 será *low* – 0 lógico. Se o conteúdo dos registradores for menor ou igual ao valor do contador, a saída será *high* – 1 lógico (Philips, 1996). A figura 3.7 apresenta o diagrama funcional das saídas de PWM do 80C552.

Segundo Barr (2001), uma das vantagens do PWM é que o sinal permanece digital por todo o caminho desde o processador até o sistema controlado, nenhuma conversão digital analógico é necessária. Mantendo-se o sinal digital, os efeitos de ruídos elétricos são minimizados, pois estas interferências só podem afetar o sistema se forem fortes o suficiente para mudar um 1 lógico para um 0 lógico, ou vice-versa.

FIGURA 3.7 – DIAGRAMA DAS SAÍDAS DE PWM DO 80C552



Fonte: Philips (1996)

3.3.6 CONVERSOR ANALÓGICO-DIGITAL

O objetivo funcional deste dispositivo é o de produzir um número binário proporcional ao valor analógico da tensão que se introduz na entrada.

3.3.6.1 RESOLUÇÃO

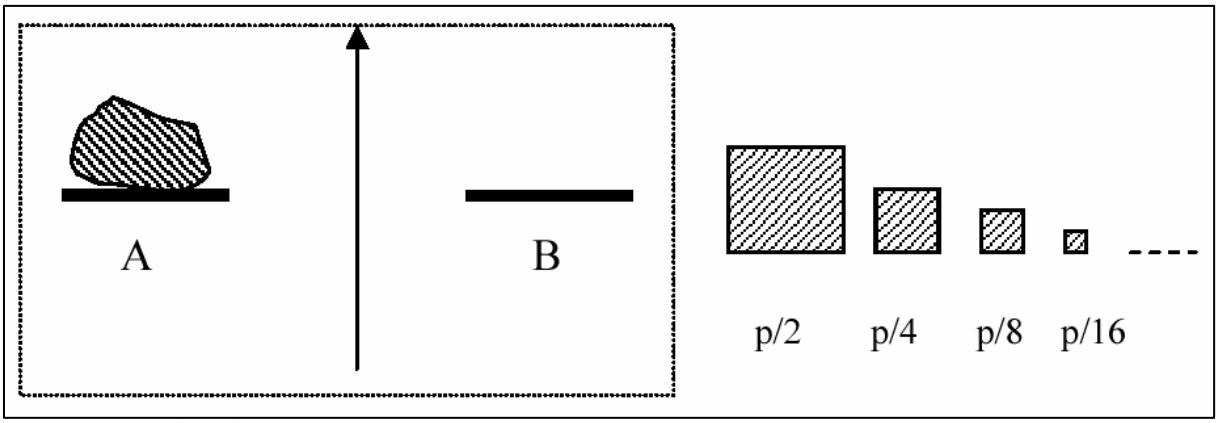
Também quanto à resolução é possível intuir que quanto maior o número binário que a saída pode produzir, maior será a resolução. O circuito de entradas analógicas do 80C552 consiste em 8 entradas analógicas multiplexadas (pode-se ler apenas uma de cada vez), com uma resolução de 10 bits (Philips, 1996). Isto faz com que a saída do ADC (*analog-digital converter*) do 80C552 possa alcançar o valor de $2^{10} – 1 = 1023$, tendo assim, uma resolução de uma parte em 1023.

3.3.6.2 MÉTODO DE CONVERSÃO

O 80C552 utiliza o método, ou arquitetura, de conversão conhecida como ADC de aproximações sucessivas (Philips, 1996). Sua implementação se baseia em uma unidade

lógica conhecida como SAR (*Successive Approximation Register*), que utiliza o mesmo algoritmo de pesagem das balanças do tipo Roberval. Nestas balanças, quando uma massa desconhecida é colocada num dos pratos, o operador começa por colocar no outro prato o maior peso-referência de que dispõe (Correia, 2001). Pode-se desde já imaginar que os pesos-referência obedecem a um escalonamento binário como se indica na figura 3.8.

FIGURA 3.8 – PESAGEM COM BALANÇA ROBERVAL E ESCALA BINÁRIA DE REFERÊNCIAS



Fonte: Correia (2001)

Segundo Correia (2001), o operador começa sempre colocando o maior peso da escala (que por conveniência atribuímos o valor p/2) no prato B e decide, observando o fiel, se ele é excessivo (o fiel desvia-se para o lado do prato B) ou se, pelo contrário, tem de adicionar em B o peso-referência seguinte, p/4.

Na primeira hipótese, deve-se retirar p/2 do prato B e atribuir ZERO ao dígito mais significativo do número que está a determinar. Na segunda hipótese o peso p/2 permanece em B e é atribuído UM a esse digito. Em qualquer dos casos o operador prossegue testando o efeito dos pesos e decidindo, com o mesmo critério, se o digito seguinte é ZERO ou UM.

O operador repete esta operação sucessivamente até chegar ao último peso da escala binária de que dispõe. Se dispuser de N elementos nesta escala o número binário que construiu tem também N dígitos.

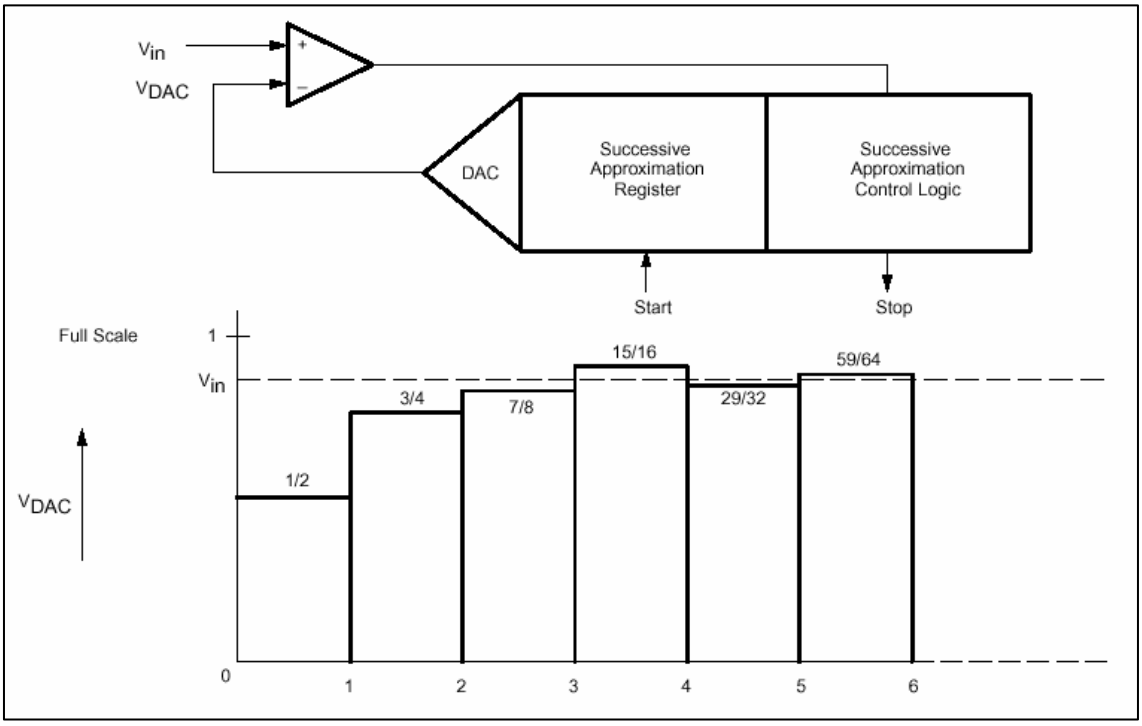
O ADC de aproximações sucessivas, cujos elementos são mostrados na figura 3.9, contém um conversor digital-analógico (DAC) que converte os valores do registrador de

aproximações sucessivas (SAR) em tensão (VDAC), que é comparada ao valor da tensão da entrada analógica (Vin) (Philips, 1996).

A lógica de controle de aproximações sucessivas primeiramente seta o bit mais significativo do SAR e reseta todos os outros (10 0000 0000B). A saída do DAC (50% da escala) é comparada a tensão da entrada analógica (Vin). Se a tensão Vin for maior que VDAC, o bit permanece setado, senão ele é resetado.

Agora a lógica de controle de aproximações sucessivas seta o próximo bit mais significativo (11 0000 0000B ou 01 0000 0000B, dependendo do resultado anterior) e VDAC é comparada novamente à Vin. Se Vin for maior que VDAC, o bit que está sendo testado permanece setado, senão o bit é resetado. Este processo se repete até que todos os 10 bits sejam testados, neste ponto o resultado da conversão se encontra no SAR.

FIGURA 3.9 – ADC DE APROXIMAÇÕES SUCESSIVAS



Fonte: Philips (1996)

4 DESENVOLVIMENTO DO PROTÓTIPO

Neste capitulo serão apresentados a especificação do protótipo de hardware e software e as ferramentas utilizadas.

Após a especificação, a implementação descreve a montagem do hardware do protótipo, o desenvolvimento do software e sua operacionalidade.

4.1 FERRAMENTAS UTILIZADAS

Para a especificação do esquema eletrônico e para a implementação da placa de circuito impresso do protótipo de hardware, foi utilizada a ferramenta Tango. O Tango possui todas as funções necessárias para o desenvolvimento de placas de circuito impresso, desde o editor de esquemas eletrônicos até a ferramenta de *layout* da placa propriamente dita (Altium, 1992).

A especificação do software de controle de temperatura baseado em lógica fuzzy foi realizada utilizando-se fluxogramas e a ferramenta FuzzyTech (Inform, 2001). O fluxograma é um método de especificação e documentação bastante conhecido e muito utilizado principalmente para especificar processos com execução seqüencial. Já o software FuzzyTech é especializado para o desenvolvimento de aplicações com lógica fuzzy, gerando uma ampla documentação do sistema desenvolvido.

O software do protótipo foi implementado na linguagem C, muito difundida nos projetos para microcontroladores. Foi utilizado o compilador da Keil Software, C51. Este compilador é específico para a família de microcontroladores MCS-51 e seus derivados, como é o caso do 80C552.

4.2 ESPECIFICAÇÃO DO HARDWARE

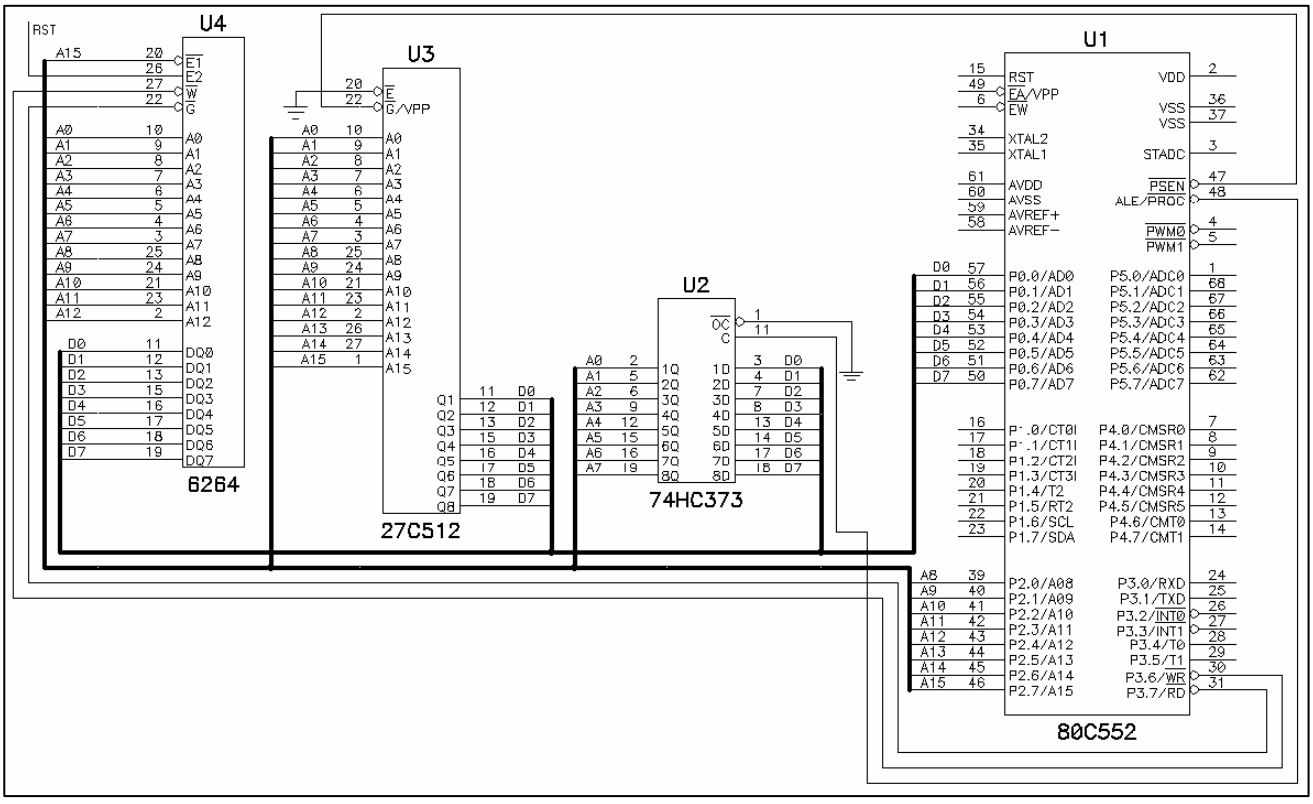
O primeiro passo tomado na especificação do hardware deste protótipo, foi a escolha de um processador, no caso um microcontrolador, que atendesse aos requisitos do projeto. Optou-se por utilizar o 80C552 da Philips, devido a algumas de suas características: este microcontrolador possui entradas analógicas e saídas de PWM *on-chip*. Tendo em vista que o projeto prevê o controle de temperatura de um ambiente, fica clara a necessidade de pelo

menos 1 entrada analógica para a leitura desta temperatura, neste protótipo, e uma saída para controlar proporcionalmente algum dispositivo capaz de atuar sobre esta temperatura.

4.2.1 MEMÓRIAS DE DADOS E PROGRAMA

O modelo do microcontrolador utilizado não possui memória de programa (ROM – *Read Only Memory*) interna, por este motivo fez-se necessária a utilização de uma memória do tipo EPROM (*Eraseable Programable Read Only Memory*) externa. O tamanho desta memória foi super dimensionado para este projeto para dar mais versatilidade ao protótipo, sendo utilizada a memória 27C512 de 64 Kbytes. Pelo mesmo motivo do super dimensionamento da memória de programa, optou-se também por incluir uma memória de dados (RAM – *Random Access Memory*) externa. Foi utilizado a memória 6264, que acrescentou 8 Kbytes de memória de dados ao sistema. A figura 4.1 apresenta a esquema eletrônico das conexões do microcontrolador com as memórias externas:

FIGURA 4.1 – CONEXÃO DO MICROCONTROLADOR E MEMÓRIAS RAM E EPROM



Na figura 4.1 pode-se identificar claramente os 3 barramentos que possibilitam a interface entre o microcontrolador e as memórias externas:

a) barramento de dados (D0..D7), local por onde trafegam os dados lidos e escritos nas memórias;
b) barramento de endereços (A0..A15), local onde a memória é endereçada, ou seja, no caso de uma escrita a memória, os dados contidos no barramento de dados serão armazenados no endereço apontado pelo barramento de endereços;
c) barramento de controle (PSEN, ALE, WR, RD), serve para indicar o tipo de operação (leitura ou escrita) que será realizada, e para habilitar o *chip* correto.

Como em toda família de microcontroladores MCS-51, o barramento de dados e endereços é multiplexado (os pinos de dados D0..D7 são usados também como A0..A7). Esta multiplexação é controlada pelo pino ALE (*Adress Latch Enable*) do microcontrolador e utilizando um *latch* para separar os sinais de dados dos sinais de endereços. No caso deste protótipo utilizou-se o *chip* 74HC373. O 74HC373 é um *latch* octal que possui dois pinos de controle independentes, o *latch enable* (C), e o *output enable* (OC). Neste caso o pino OC está diretamente ligado ao GND (nível lógico 0) para que as saídas do *chip* fiquem sempre habilitadas, já o pino *latch enable* (C), está conectado ao pino ALE do microcontrolador. Este pino em nível alto, indica que o controlador esta endereçando algum periférico, quando a entrada *latch enable* do 74HC373 sente este nível alto, faz com que o dado contido no pinos de entrada do *latch* (1D..7D), sejam transferidos à saída (1Q..7Q), como pode ser visto na figura 4.1, gerando assim, juntamente com os pinos A8..A15, um endereço de 16 bits.

A seleção da memória a ser acessada (RAM ou EPROM) é controlada pelo 80C552 através do pino PSEN (*Program Store Enable*) e dos pinos RD (*Read*) e WR (*Write*). Nível baixo (0 lógico) no pino PSEN indica que o microcontrolador deseja acessar a memória de programa para buscar instruções. Como a EPROM utilizada possui 64 Kbytes de tamanho, ela utiliza toda a faixa de endereçamento possível, indo de 0000H até FFFFH.

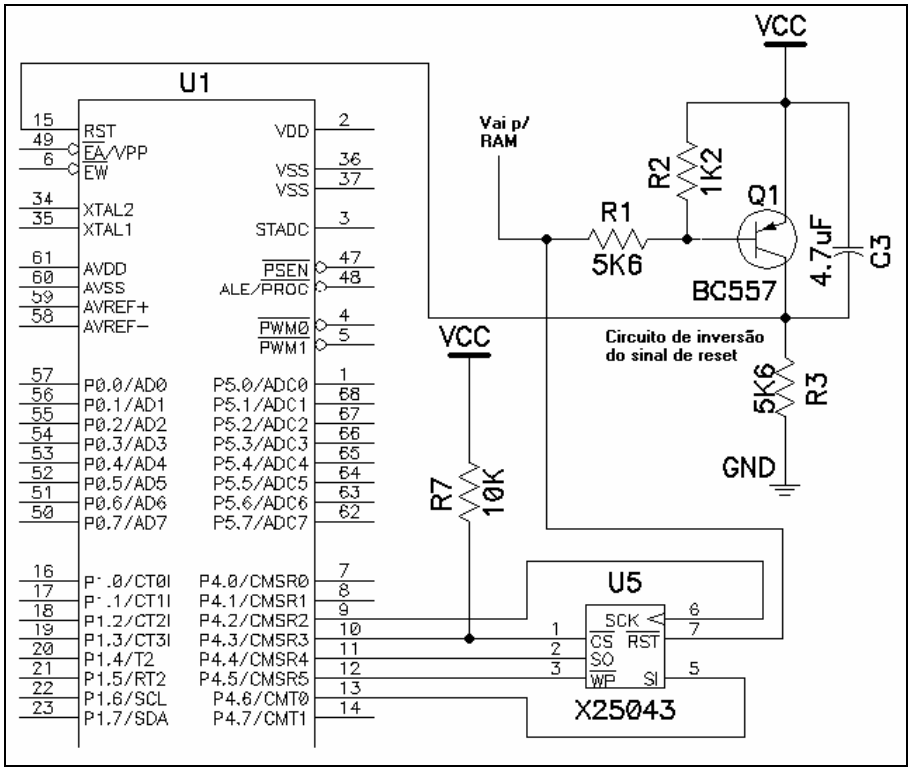
Para o acesso a RAM são utilizados os pinos de controle RD e WR. Nível baixo em RD indica que o controlador irá ler dados da RAM, e nível baixo em WR, indica uma escrita na memória de dados. Pode-se notar na figura 4.1, que o pino 20 (*enable1*) do *chip* 6264 está ligado à linha de endereço A15, como o *enable1* é ativo baixo, a faixa de endereços disponível para a RAM fica sendo de 0000H até 7FFFH. Foi convencionado, para este

protótipo, utilizar a faixa de endereços de 0000H até 1FFFH, tendo em vista que a RAM possui somente 8 Kbytes.

4.2.2 SISTEMÁTICA DE *RESET*

A figura 4.2 ilustra o circuito de *reset* do microcontrolador utilizando o chip de *watchdog* externo, o X25043 da XICOR.

FIGURA 4.2 – CIRCUITO DE RESET DO PROTÓTIPO



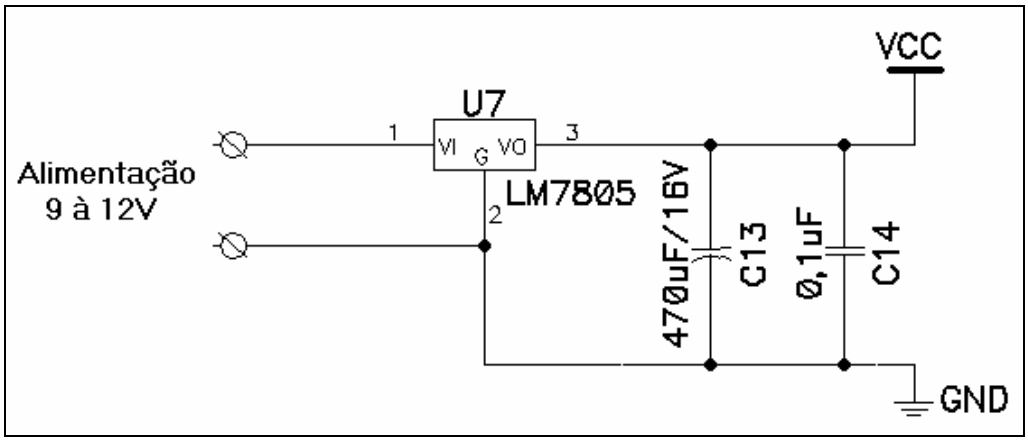
O *chip* X25043 foi escolhido, ao invés do *watchdog timer* interno do microcontrolador, para este protótipo, devido ao seu funcionamento simples e eficaz, e pelo fato de possuir além da função de *watchdog*, um controle de *reset* que detecta se a tensão de alimentação do sistema está fora da faixa de operação do controlador (por exemplo, durante o *power-up* do sistema), e mantém o mesmo "resetado" até que a tensão se normalize, e 512 bytes de E²PROM (*Electrically Eraseble Programmable Read Only Memory*), que futuramente poderão vir a ser utilizados para armazenar parâmetros de controle do protótipo, dando assim maior versatilidade ao sistema.

Como o pino de *reset* do X25043 é ativo baixo, e o 80C552 necessita de um pulso alto para ser “resetado” foi necessário adicionar um pequeno circuito de inversão do sinal do *watchdog*.

4.2.3 FONTE DE ALIMENTAÇÃO

Este protótipo necessita de uma tensão (VCC) de 5V para seu funcionamento. Para gerar esta tensão o protótipo deve receber uma tensão de 9 à 12V em seus bornes de alimentação. Para transformar esta tensão de entrada em 5V, é utilizado um componente simples e muito comum na eletrônica, o regulador de tensão 7805. Conforme a figura 4.3, a tensão injetada no pino 1 do 7805 (U7) é regulada para 5V e disponibilizada no pino 3. A figura 4.3 mostra também os capacitores C13 e C14, responsáveis pela filtragem de possíveis ruídos elétricos.

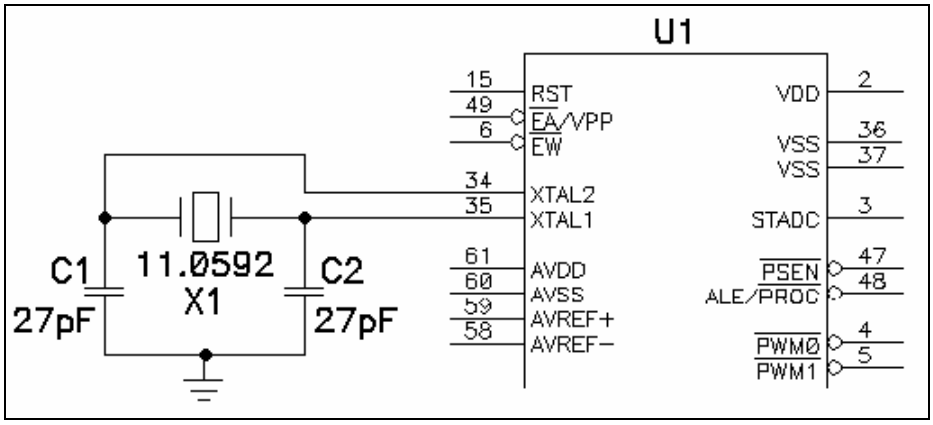
FIGURA 4.3 – FONTE DE ALIMENTAÇÃO DO PROTÓTIPO



4.2.4 CRISTAL OSCILADOR

O cristal é o componente responsável pela base de tempo para todos os processos internos do microcontrolador. A figura 4.4 ilustra a conexão de um cristal ao 80C552. O microcontrolador 80C552 permite um cristal de até 16 Mhz em seus pinos (34 e 36). Porém, quanto se utiliza comunicação serial, há a preferência a um cristal cujo valor facilite o cálculo da *baud rate* (taxa de transmissão). Neste caso foi utilizado um cristal de 11.0592 Mhz (X1).

FIGURA 4.4 – CRISTAL OSCILADOR CONECTADO AO 80C552



4.2.5 ENTRADA ANALÓGICA PARA TEMPERATURA

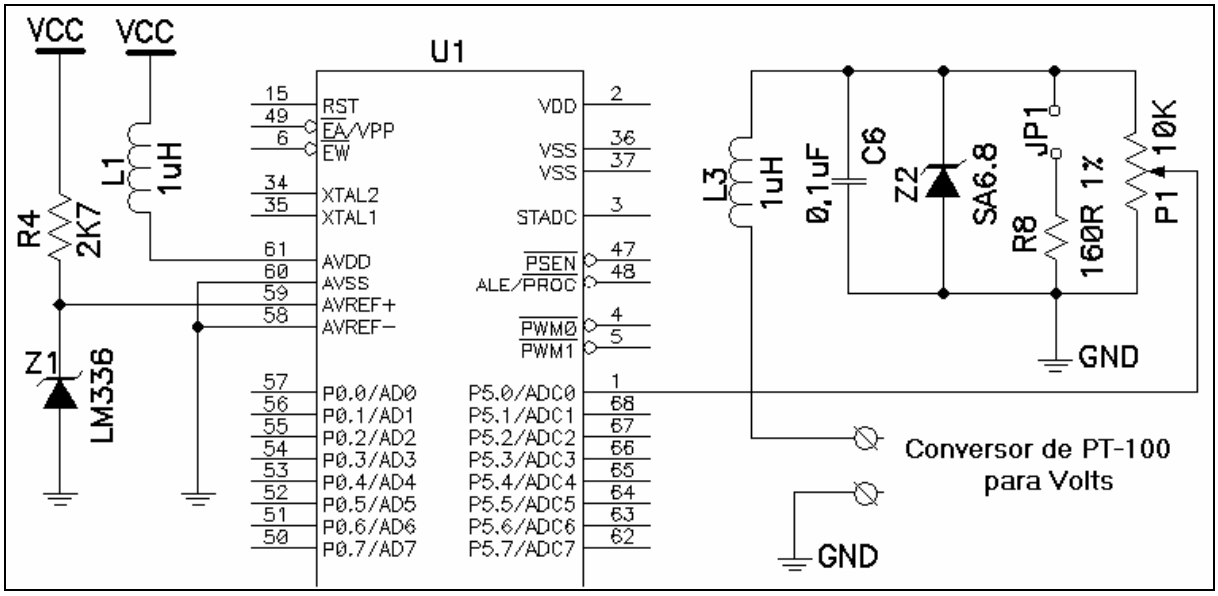
Como visto no capítulo 3, o microcontrolador 80C552 possui 8 entradas analógicas *on-chip*, uma destas entradas é utilizada no protótipo para realizar a aquisição do valor da temperatura do ambiente a ser controlado.

Para a leitura desta temperatura foi utilizado um sensor do tipo termoresistência, ou também conhecido como PT-100. O sensor PT-100 é uma resistência que varia linearmente com a temperatura a ela aplicada. Para 0 (zero) graus Celsius seu valor é de 100Ω (ohms) e, para 100 graus Celsius é de 138,5Ω (ohms).

A entrada analógica do microcontrolador é desenvolvida para leitura de tensões, não resistências, por este motivo é necessário utilizar um conversor de PT-100 para Volts. Neste protótipo foi utilizado um conversor fabricado pela Multitherm Sistemas e Automação Ltda. Este dispositivo converte a resistência equivalente a uma temperatura de 0 à 100°C, em uma tensão na faixa de 0 à 2,5V.

A figura 4.5 ilustra o circuito que realiza a leitura da temperatura (depois de convertida em Volts), mostrando todos os componentes discretos necessários ao condicionamento do sinal, bem como a geração das tensões de referência do conversor analógico – digital.

FIGURA 4.5 – CIRCUITO DE LEITURA DE TEMPERATURA DO AMBIENTE



Pode-se notar na figura 4.5 as conexões necessárias ao funcionamento do ADC (*Analog to Digital Converter*) do microcontrolador 80C552. Primeiramente a alimentação do conversor, que se dá através dos pinos 60 e 61 (AVSS e AVDD respectivamente). Se faz necessária também a geração de uma tensão de referência constante. Isso é conseguido através do uso de um diodo zenner, o LM336, que regula a tensão no pino 59 (AVREF+) do 80C552, em exatos 2,5V, e conectando-se o pino 58 (AVREF-) ao GND.

A leitura da temperatura propriamente dita, se dá no pino 1 (ADC0) do microcontrolador. Antes de chegar a este pino o sinal do PT-100, já convertido para Volts, passa por um circuito condicionador, que irá filtrar e regular o sinal.

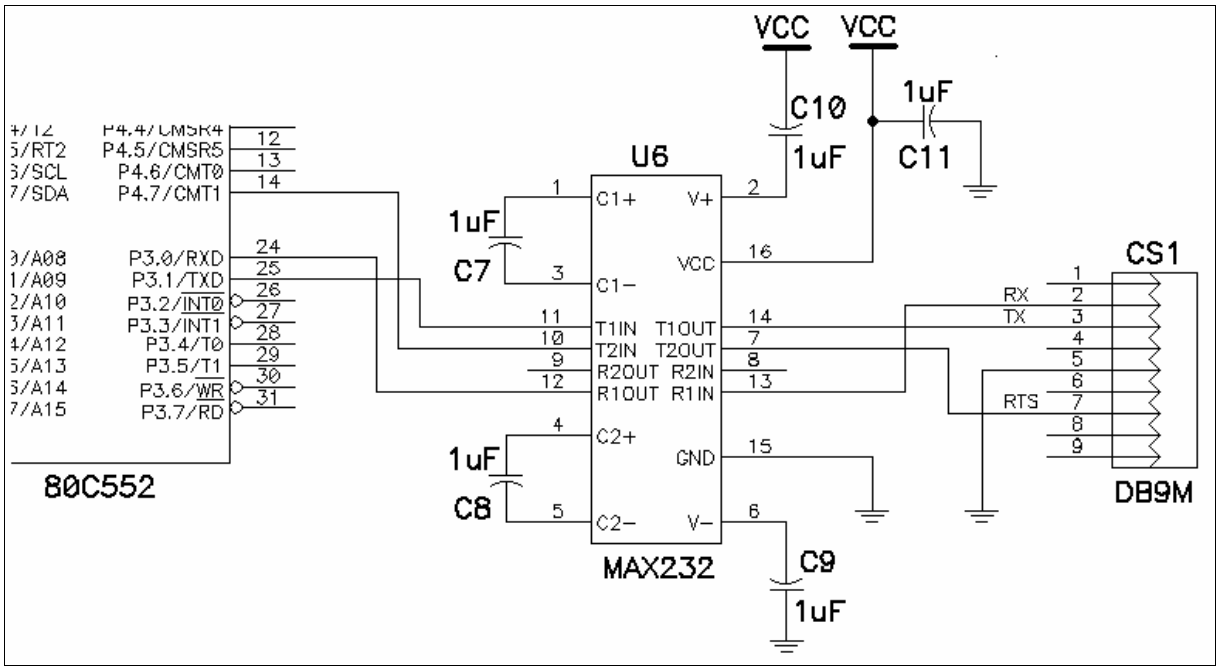
4.2.6 INTERFACE SERIAL

Um dos objetivos específicos deste trabalho, conforme visto no item 1.1, é a disponibilização dos dados, bem como, o envio de *setpoints* ao protótipo através de um software de supervisão. Para tanto se faz necessário o uso de algum meio de comunicação entre o protótipo e um computador.

Foi incluído no projeto uma interface serial padrão RS-232, que, ligada à UART (*Universal Assinchronous Receiver Transmiter*) do microcontrolador 80C552, que é compatível ao padrão da família MCS-51, faz a interface entre o protótipo e o software de supervisão que está rodando em um PC.

A figura 4.6 mostra o circuito da interface serial do protótipo:

FIGURA 4.6 – CIRCUITO DA INTERFACE SERIAL DO PROTÓTIPO



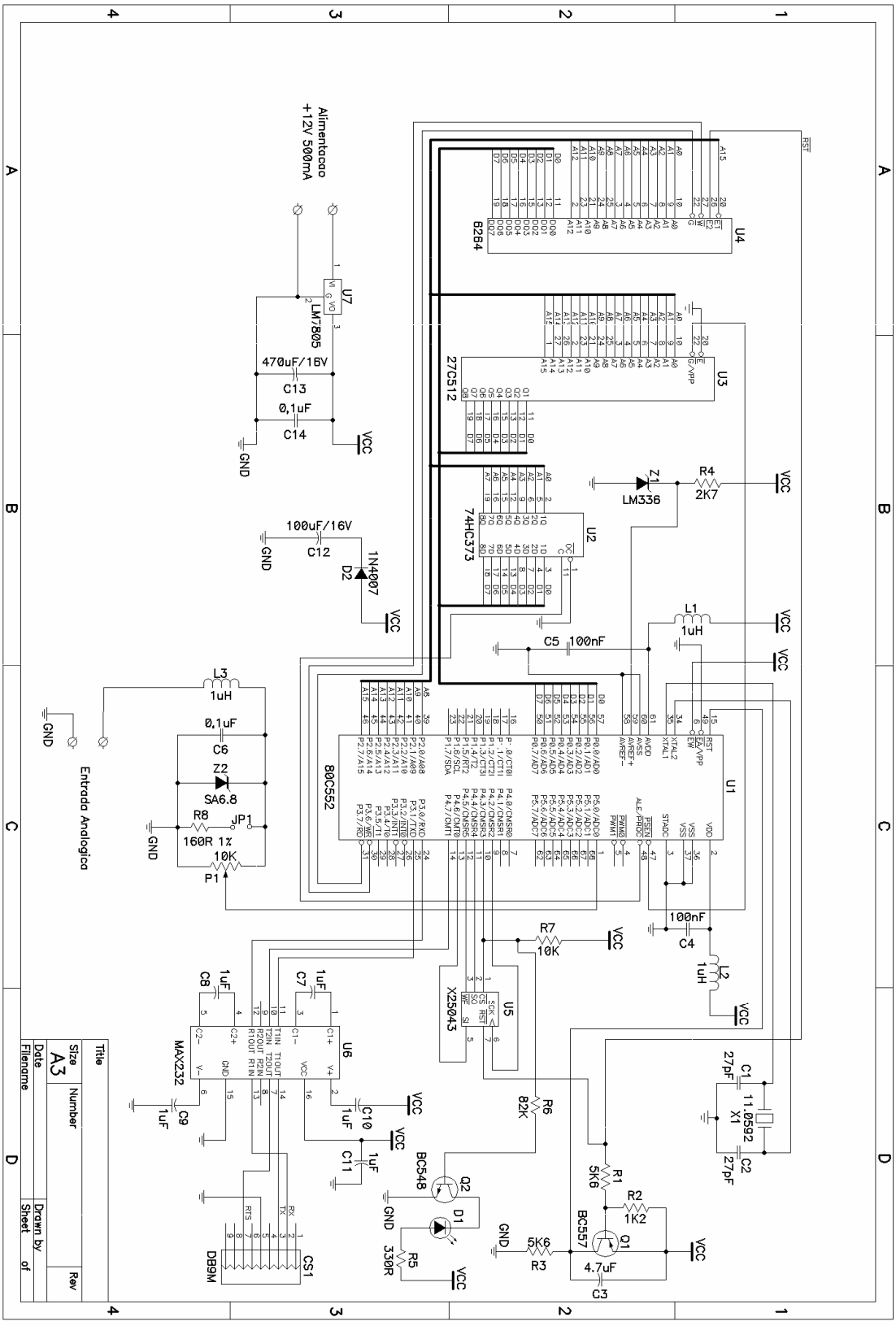
O *chip* MAX232 (U6) é um conversor duplo bidirecional de nível de tensão para a norma RS-232. Com apenas alguns capacitores é possível alimentar este *chip* com 5V e obter o padrão de sinais RS-232 em suas saídas de forma a possibilitar a implementação de uma interface serial com qualquer computador que possua uma porta serial RS232.

Mais informações sobre interface serial RS232 podem ser encontradas em Jordan (1994) e Axelson (1998).

4.2.7 ESQUEMA ELETRÔNICO COMPLETO

A figura 4.7 ilustra todo o esquema eletrônico do protótipo. Como citado anteriormente foi utilizada a ferramenta Tango para a geração deste esquema.

FIGURA 4.7 – ESQUEMA ELETRÔNICO COMPLETO DO PROTÓTIPO



4.3 ESPECIFICAÇÃO DO SOFTWARE

Nesta seção serão abordadas as especificações do *software* de controle Fuzzy, bem como, a especificação do *software* de supervisão e o protocolo de comunicação utilizado.

4.3.1 SOFTWARE DE CONTROLE FUZZY

Como mencionado anteriormente, a especificação do software do protótipo foi feita com a ferramenta FuzzyTech. Inicialmente, vê-se uma introdução ao princípio de funcionamento e estrutura do sistema.

4.3.1.1 ESTRUTURA DO SISTEMA

A temperatura do sistema é lida através da entrada analógica do protótipo. A diferença entre a temperatura lida e o *setpoint* do sistema, é chamada de “delta_temp” e é uma das entradas do sistema (medida em °C). O valor da temperatura é então subtraído do valor da temperatura lida anteriormente (é feita uma leitura por segundo), isto gera a variável “gradiente”, que é a segunda entrada do controlador fuzzy(esta entrada é medida em °C/s).

Cada uma das regras de inferência, do tipo **se..então**, irá analizar estas entradas e gerar uma saída apropriada. As saídas individuais de cada regra serão combinadas e posteriormente desfuzzificadas gerando assim a saída do sistema chamada de “saída_pwm”, que no caso deste sistema representa o percentual de potência aplicada sobre um agente de aquecimento do ambiente.

A figura 4.8 ilustra a estrutura do sistema de controle fuzzy implementado. Nela podemos observar as duas entradas (delta_temp e gradiente), o bloco de regras (RB1 – *Rule Block 1*), e a saída do sistema (saída_pwm).

FIGURA 4.8 – ESTRUTURA DO SISTEMA

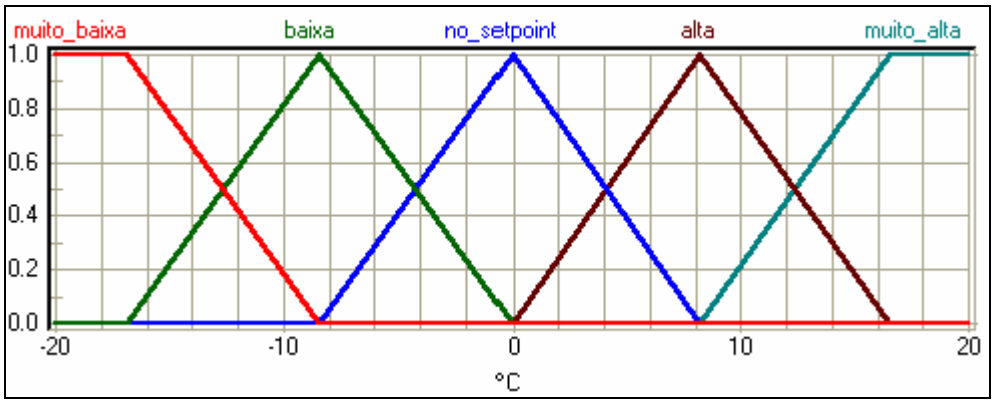
4.3.1.2 VARIÁVEIS DE ENTRADA

Aqui é apresentada a especificação das duas entradas do sistema, transformadas em conjuntos difusos.

Primeiramente a variável delta_temp. Esta entrada foi definida como podendo assumir valores de –20 à 20°C, representado, respectivamente, uma faixa de 20°C abaixo do *setpoint* até 20°C acima do *setpoint*. Fora desta faixa a saída será 100% aberta, para valores abaixo do *setpoint*, e 0% aberta para valores acima do *setpoint*.

A figura 4.9 ilustra o conjunto fuzzy gerado para esta entrada. Definiu-se que os valores lingüísticos possíveis para delta_temp serão: muito baixa, baixa, no *setpoint*, alta e muito alta.

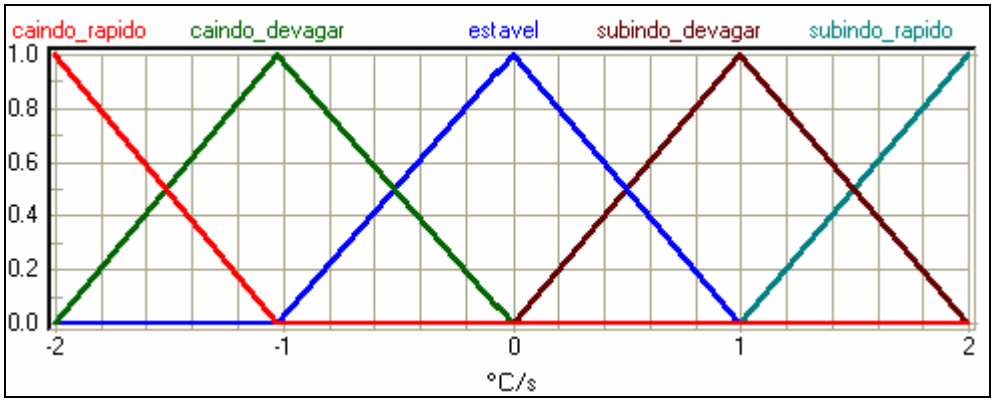
FIGURA 4.9 – VARIÁVEL DELTA_TEMP



A variável gradiente, por sua vez, poderá variar na faixa de –2 à 2°C/s. Valores negativos indicam que a temperatura está caindo e valores positivos indicam temperatura em elevação. Os valores lingüísticos definidos para gradiente são: caindo rápido, caindo devagar, estável, subindo devagar e subindo rápido.

A figura 4.10 ilustra o conjunto fuzzy gerado para a variável gradiente.

FIGURA 4.10 – VARIÁVEL GRADIENTE

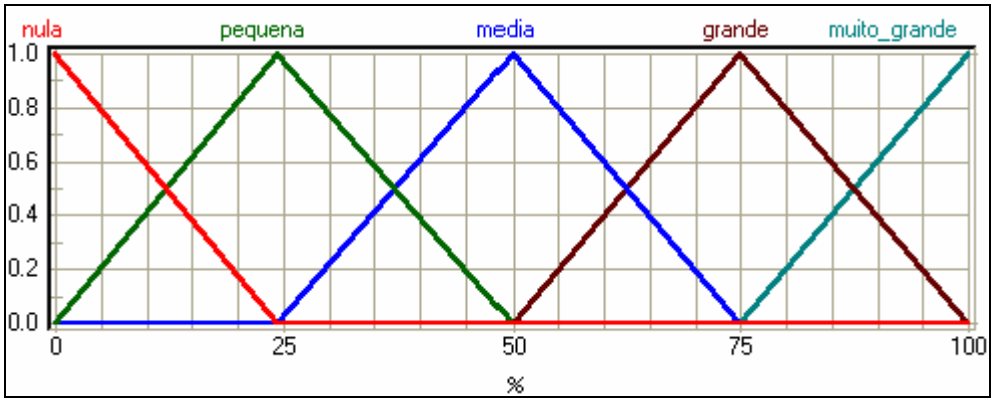


4.3.1.3 VARIÁVEL DE SAÍDA

A variável de saída do sistema, saída_pwm, representa a "potência" aplicada sobre um agente de aquecimento para possibilitar ao sistema atingir o *setpoint* desejado. Este dado está sendo representado em percentual (0 à 100%). Notaremos que, quando a temperatura do processo estiver acima do *setpoint*, o valor de saida_pwm tenderá a 0, pois este sistema prevê apenas o aquecimento de um ambiente.

Os valores lingüísticos previstos para a variável saída_pwm são: nula, pequena, media, grande e muito grande. A figura 4.11 mostra a representação da variável saída_pwm.

FIGURA 4.11 – VARIÁVEL SAÍDA_PWM



4.3.1.4 BLOCO DE REGRAS

A lógica fuzzy necessita de regras para definir seu comportamento. Estas regras definem as condições esperadas durante o processo, e que atitudes serão tomadas para cada

condição. Elas substituem as fórmulas matemáticas normalmente utilizadas. Estas regras devem cobrir todas as situações possíveis. Por este motivo o bloco de regras desta implementação possui 25 regras, que cobrem todas as combinações das 2 entradas.

As atitudes a serem tomadas foram definidas com base na própria experiência do autor em controle de processos, sem a necessidade de conhecimento aprofundado, e de um modelo matemático do sistema.

A figura 4.12 mostra as regras definidas dentro da ferramenta FuzzyTech.

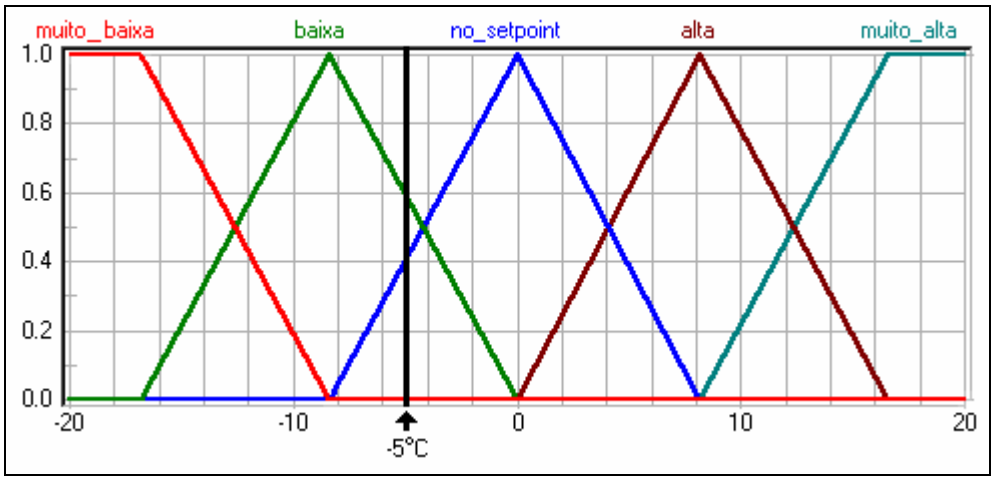
FIGURA 4.12 – BLOCO DE REGRAS

#	IF		THEN
	delta_temp	gradiente	saida_pwm
1	muito_baixa	caindo_rapido	muito_grande
2	baixa	caindo_rapido	muito_grande
3	no_setpoint	caindo_rapido	grande
4	alta	caindo_rapido	media
5	muito_alta	caindo_rapido	pequena
6	muito_baixa	caindo_devagar	muito_grande
7	baixa	caindo_devagar	grande
8	no_setpoint	caindo_devagar	media
9	alta	caindo_devagar	pequena
10	muito_alta	caindo_devagar	pequena
11	muito_baixa	estavel	muito_grande
12	baixa	estavel	grande
13	no_setpoint	estavel	nula
14	alta	estavel	nula
15	muito_alta	estavel	nula
16	muito_baixa	subindo_devagar	grande
17	baixa	subindo_devagar	media
18	no_setpoint	subindo_devagar	nula
19	alta	subindo_devagar	nula
20	muito_alta	subindo_devagar	nula
21	muito_baixa	subindo_rapido	media
22	baixa	subindo_rapido	pequena
23	no_setpoint	subindo_rapido	nula
24	alta	subindo_rapido	nula
25	muito_alta	subindo_rapido	nula

4.3.2 EXEMPLO DE FUNCIONAMENTO

Tomando por exemplo, uma situação onde o *setpoint* do processo é 60°C e a temperatura atual seja 55°C, isso faria com que a entrada delta_temp assumisse o valor –5. Supondo também que a temperatura está subindo a uma velocidade de 0,6°C/s, ou seja, entrada gradiente igual a 0,6. A figura 4.13 ilustra graficamente a variável delta_temp.

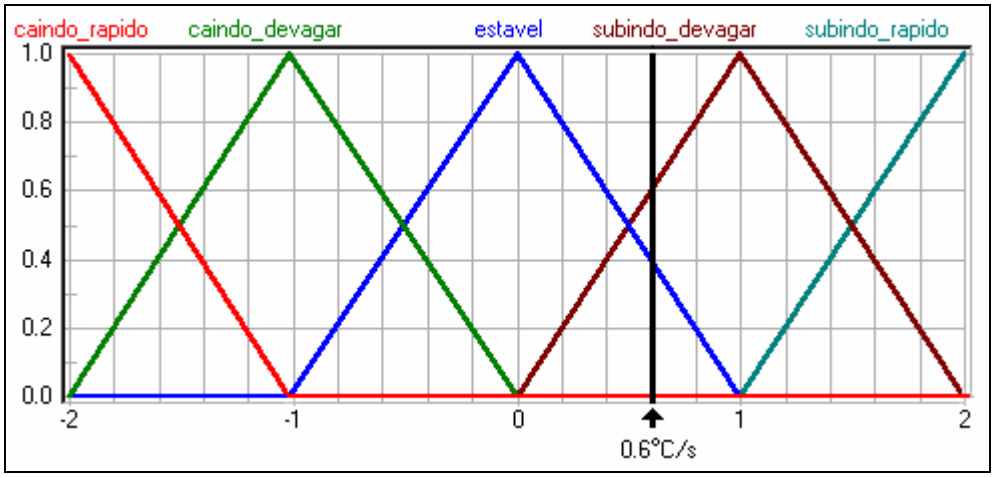
FIGURA 4.13 – VARIÁVEL DELTA_TEMP COM VALOR IGUAL A –5°C



Como se pode notar na figura 4.13, o valor –5°C pertence ao conjunto “baixa” com uma pertinência de 0.6 e ao conjunto “no setpoint”, com uma pertinência de 0.4.

A figura 4.14 ilustra graficamente a variável gradiente.

FIGURA 4.14 – VARIÁVEL GRADIENTE COM VALOR IGUAL A 0.6°C/S



O valor suposto para o exemplo, 0,6°C/s, pertence a dois conjuntos, “estável”, com uma pertinência de 0.4 e ao conjunto “subindo devagar” com uma pertinência de 0.6.

Esta situação ativará 4 das 25 regras definidas: 12, 13, 17 e 18. A figura 4.15 mostra estas regras.

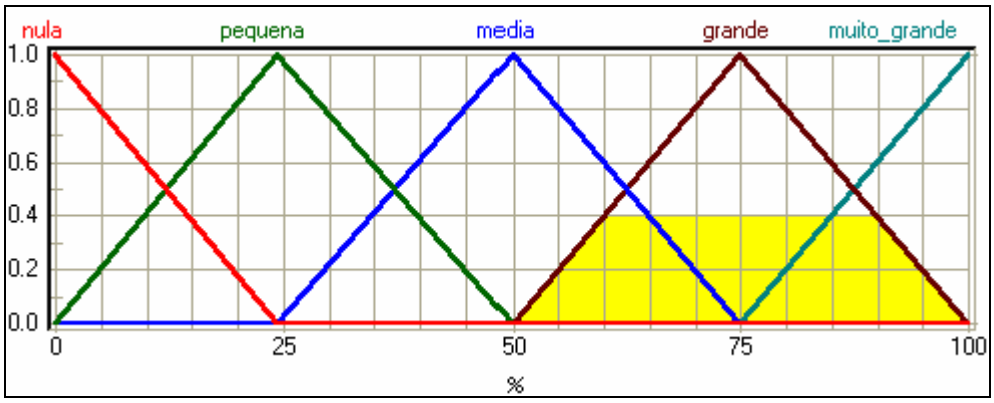
FIGURA 4.15 – REGRAS ATIVADAS NA SITUAÇÃO DE EXEMPLO

#	IF		THEN
	delta_temp	gradiente	saida_pwm
12	baixa	estavel	grande
13	no_setpoint	estavel	nula
17	baixa	subindo_devagar	media
18	no_setpoint	subindo_devagar	nula

Analisando, inicialmente, a regra 12, tem-se o conjunto “baixa” com pertinência 0.6, e o conjunto “estável“ com pertinência 0.4. Lembrando que estes conjuntos estão ligados pela operação AND, e lembrando também das operações em conjuntos fuzzy, vistas no capítulo 3, a operação AND equivale a uma interseção dos conjuntos. A interseção define que o resultado da operação deve ser o mínimo valor dos conjuntos, no nosso caso 0.4.

Então temos como saída da regra 12, uma atuação na variável saída_pwm de pertinência 0.4 no conjunto “grande”. A figura 4.16 demonstra graficamente este exemplo.

FIGURA 4.16 – SAÍDA DA REGRA 12 APLICADA À VARIÁVEL SAÍDA_PWM

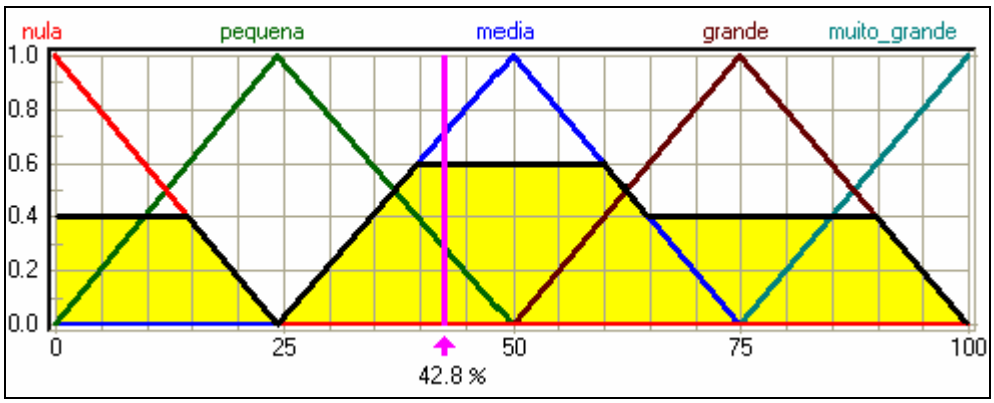


Depois de determinadas as saídas de cada regra, todas são combinadas para formar o chamado *Logical Sum*. As saídas são combinadas através da operação OR. O OR lógico

significa “pertencente a qualquer um dos conjuntos”. Segundo a teoria das operações com conjuntos Fuzzy, cada conjunto irá contribuir com o seu maior valor para formar o *Logical Sum.* Por exemplo, duas regras geram saídas para o conjunto “grande”, uma com pertinência 0.3 e outra com pertinência 0.5. Para fins de geração do *Logical Sum* será utilizado o valor 0.5.

A figura 4.17 ilustra a variável saída_pwm após todas as regras serem inferidas e realizar a desfuzzificação.

FIGURA 4.17 – VARIÁVEL SAÍDA_PWM DESFUZZIFICADA



O método de desfuzzificação utilizado aqui é o método da média dos máximos. O cálculo deste método funciona da seguinte maneira: utilizando os conjuntos “nula”, “media” e “grande”, deve-se obter o máximo valor que estes conjuntos podem representar. O valor máximo para “nula” é 0, para “média” é 50 e para “grande” o máximo é 75. Utiliza-se então este valor máximo e a pertinência de cada conjunto na formula do quadro 5.1.

QUADRO 5.1- FORMULA DA DESFUZZIFICAÇÃO MÉDIA DOS MÁXIMOS

Σ(valor máximo * pertinência)/ Σ(pertinência)

Utilizando os valores do exemplo temos: ((0*0.4)+(50*0.6)+(75*0.4)) / (0.4+0.6+0.4), com resultado igual a 42.8%.

4.3.3 SOFTWARE DE SUPERVISÃO

A função primordial do software de supervisão é a aquisição de dados e o envio de *setpoints* ao protótipo. Para tanto, foi desenvolvido um protocolo de comunicação serial, baseado no protocolo Modbus da Gould Electronics.

Este protocolo define o computador como sendo o *master* e o protótipo como sendo o *slave*. O *master* controla as transações realizadas durante a comunicação, requisitando e enviando dados ao protótipo. Para tanto, foram utilizadas 2 funções do protocolo original, a função de código 3, para leitura de registros e a função de código 6, para escrita de registros.

4.3.3.1 FUNÇÃO DE LEITURA DE REGISTROS (3)

O quadro 5.2 ilustra o formato da função 3 do protocolo Modbus. Esta função permite ao *master* ler registros do protótipo. O bloco é formado por 8 bytes, que possuem as seguintes funções:

a) endereço: identifica qual dos *slaves* será acessado. No caso deste projeto é fixo em 1 pelo fato de haver apenas um *slave*;
b) função: indica a função a ser executada, neste caso leitura (3);
c) registro inicial (2 bytes): indica o primeiro registro do *slave* a ser lido;
d) quantidade de registros (2 bytes): quantos registros a partir do registro inicial serão lidos com este comando;
e) CRC: *Cyclical Redundancy Check*.

QUADRO 5.2 – FORMATO DA FUNÇÃO 3

Endereço	Função	Registro Inicial (MSB)	Registro Inicial (LSB)	Quantidade Registros (MSB)	Quantidade Registros (LSB)	CRC (MSB)	CRC (LSB)

O quadro 5.3 mostra o formato da resposta do *slave* a um comando de leitura. Vamos supor que foi solicitada uma leitura de 2 a partir do registro 3. No caso deste exemplo o campo “Contador Bytes” assumiria o valor 4, pois cada registro é uma variável de 16 bits.

QUADRO 5.3 – FORMATO DA RESPOSTA DA FUNÇÃO 3

Endereço	Função	Contador Bytes	Registro 3 (MSB)	Registro 3 (LSB)	Registro 4 (MSB)	Registro 4 (LSB)	CRC (MSB)	CRC (LSB)

4.3.3.2 FUNÇÃO DE ESCRITA DE REGISTROS (6)

A função de código 6 permite ao *master* escrever em um dos registros (variáveis) do *slave*. Este comando é utilizado pelo software de supervisão para enviar o valor da temperatura programada (*setpoint*) ao protótipo.

O quadro 5.3 mostra o formato do comando de escrita do protocolo.

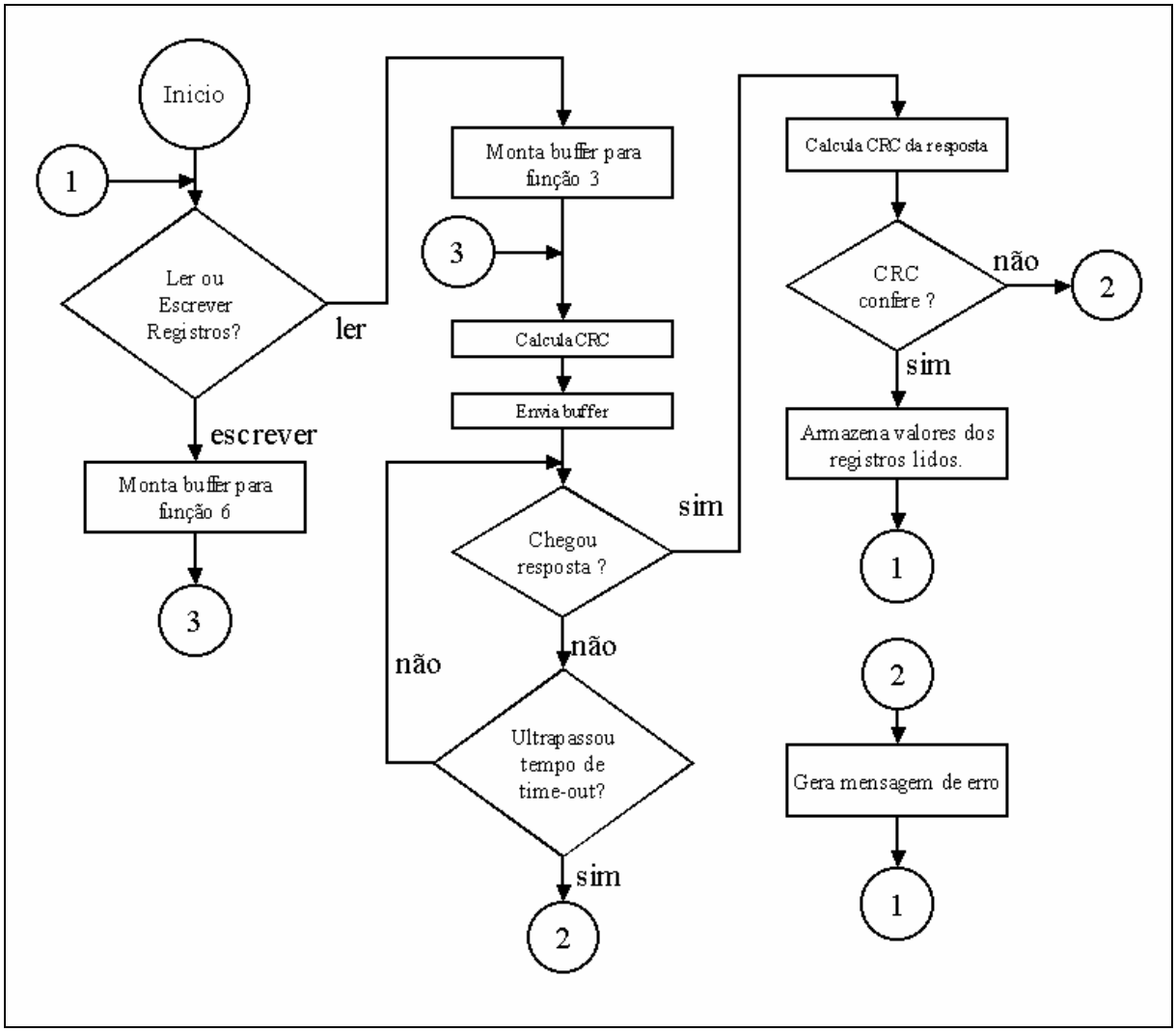
QUADRO 5.2 – FORMATO DA FUNÇÃO 6

Endereço	Função	Nº. Registro (MSB)	N°. Registro (LSB)	Dado (MSB)	Dado (LSB)	CRC (MSB)	CRC (LSB)

A resposta do *slave* é retornar este mesmo comando ao *master* indicando que a operação foi realizada com sucesso.

O fluxograma da figura 4.18 ilustra a seqüência de operação do protocolo de comunicação entre o software de supervisão e o protótipo.

FIGURA 4.18 – FLUXOGRAMA DO PROTOCOLO DE COMUNICAÇÃO



4.4 IMPLEMENTAÇÃO

Na parte de implementação alguns passos devem ser seguidos. O primeiro passo é a montagem do *hardware* do protótipo, pois sobre esta plataforma serão realizados os testes do *software*. Em seguida será apresentado o *software* de supervisão, que é a interface homem-máquina do protótipo, e permite visualizar o processo de controle de temperatura que esta sendo realizado pelo protótipo.

4.4.1 MONTAGEM DO PROTÓTIPO

O estágio final do desenvolvimento de um *hardware* é a geração de uma placa com todos os dispositivos especificados no esquema eletrônico.

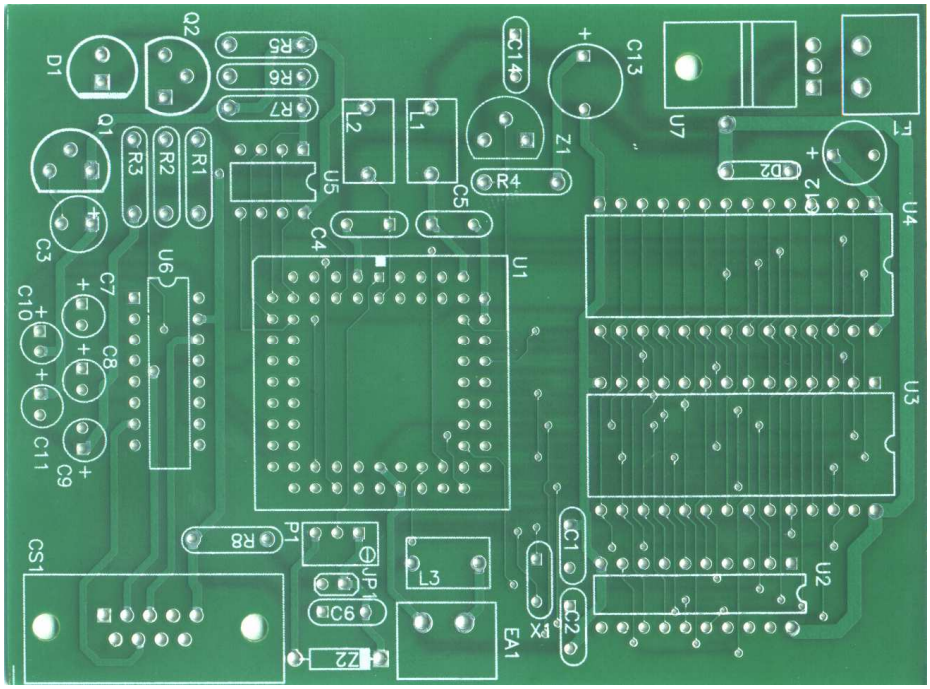
Existem várias formas para se realizar montagens de placas eletrônicas, a mais conhecida é a placa de circuito impresso (PCI), encontrada nos mais diversos tipos de aparelhos eletrônicos do mercado. Esta forma de montagem de placa é muito utilizada, pois além de suas trilhas estarem impressas na placa, os componentes eletrônicos são soldados nas mesmas, diminuindo bastante o risco de trilhas rompidas e componentes mal conectados(mau contato).

Quando se trata do desenvolvimento de um protótipo, algumas desvantagens são encontradas na placa de circuito impresso. Pelo fato de suas trilhas e componentes eletrônicos estarem presos à placa, existe muita dificuldade para eventuais alterações do projeto, além de sua confecção requerer tempo e materiais especiais.

Todo o *layout* da placa de circuito impresso do protótipo foi feito utilizando-se também a ferramenta Tango. Para tanto, importou-se o esquema eletrônico criado no módulo de esquemas do Tango para o módulo de *layout*, também conhecido como Tango PCB (*Printed Circuit Board*). Este, por sua vez, possui uma função de roteamento, que, após delimitado o tamanho e posição dos componentes na placa, gera as ligações(trilhas), automaticamente. Normalmente este processo, também chamado de *auto-route*, necessita de alguns ajustes manuais após concluído, mas é de grande auxilio, principalmente quando o tempo de implementação do projeto é um fator a ser considerado.

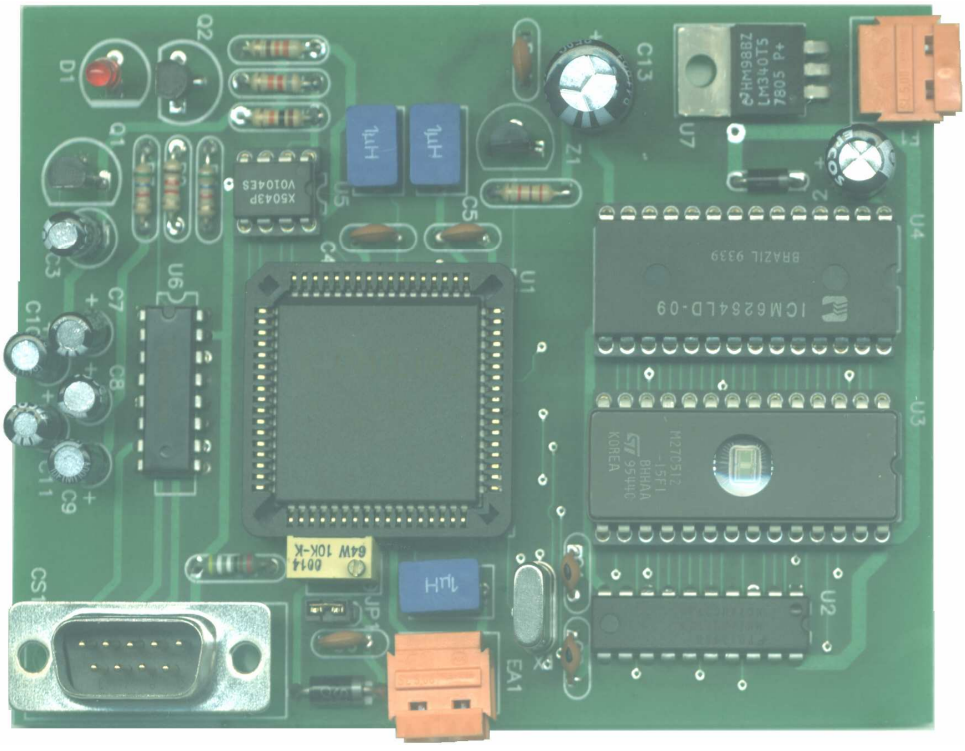
A figura 4.19 mostra a placa de circuito impresso do protótipo do lado dos componentes, ou seja nesta face da placa serão inseridos todos os componentes definidos na especificação. Esta placa possui trilhas nas duas faces, este tipo de placa é conhecido como dupla-face. Optou-se por utilizar uma placa dupla-face devido à complexidade do *layout*, uma placa de face simples teria que ser muito maior para comportar a mesma quantidade de ligações (trilhas) sem sobrepô-las (uso de muitos *jumpers*).

FIGURA 4.19 – PLACA DE CIRCUITO IMPRESSO(LADO COMPONENTES)



A figura 4.20 mostra o protótipo já montado com todos os componentes soldados à placa de circuito impresso.

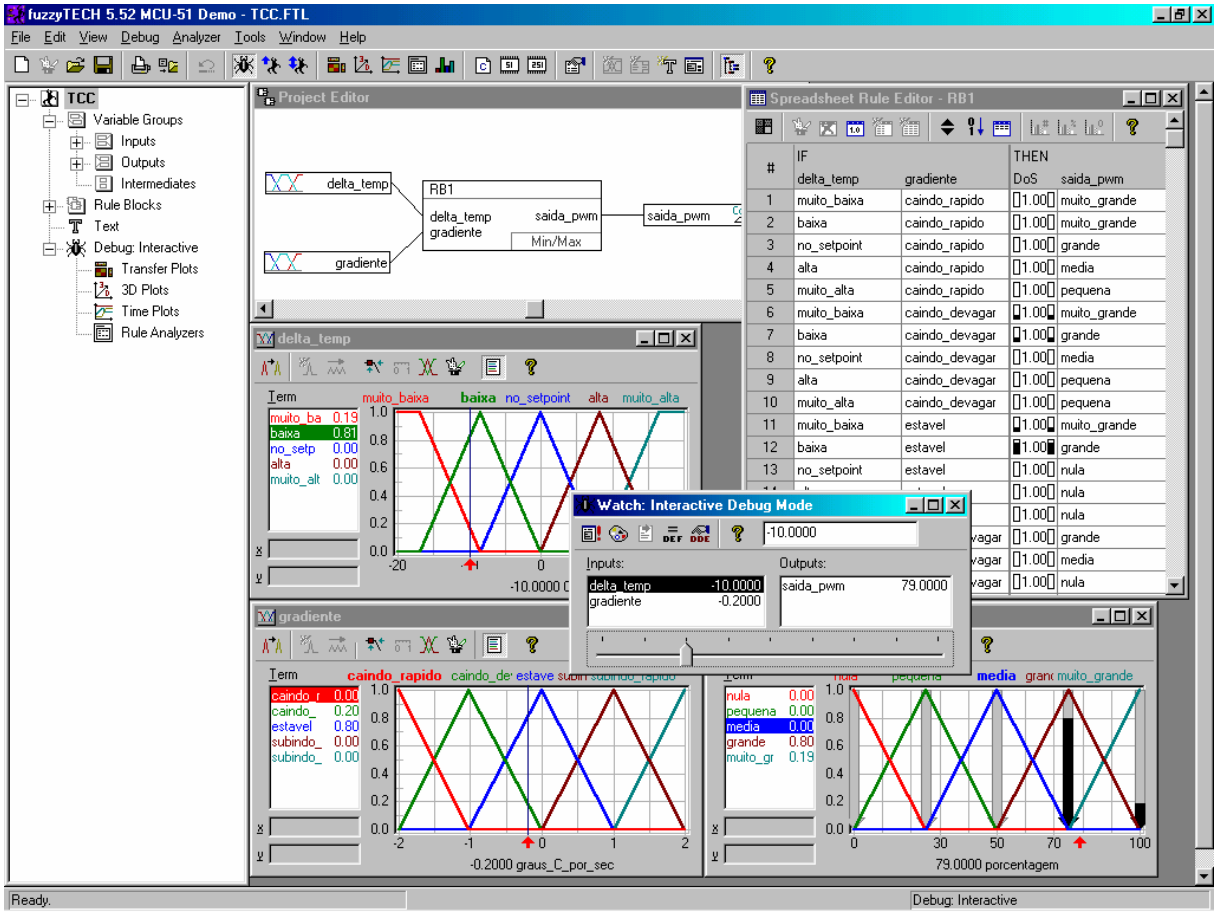
FIGURA 4.20 – PROTÓTIPO MONTADO



4.4.2 SOFTWARE DO PROTÓTIPO

Além da especificação, a implementação de parte do *software* do protótipo também foi criada utilizando a ferramenta FuzzyTech, uma vez que esta ferramenta possui uma função de geração de código a partir da especificação criada. Sendo assim toda parte de controle utilizando lógica fuzzy foi implementada automaticamente. A figura 4.21 mostra a interface da ferramenta FuzzyTech.

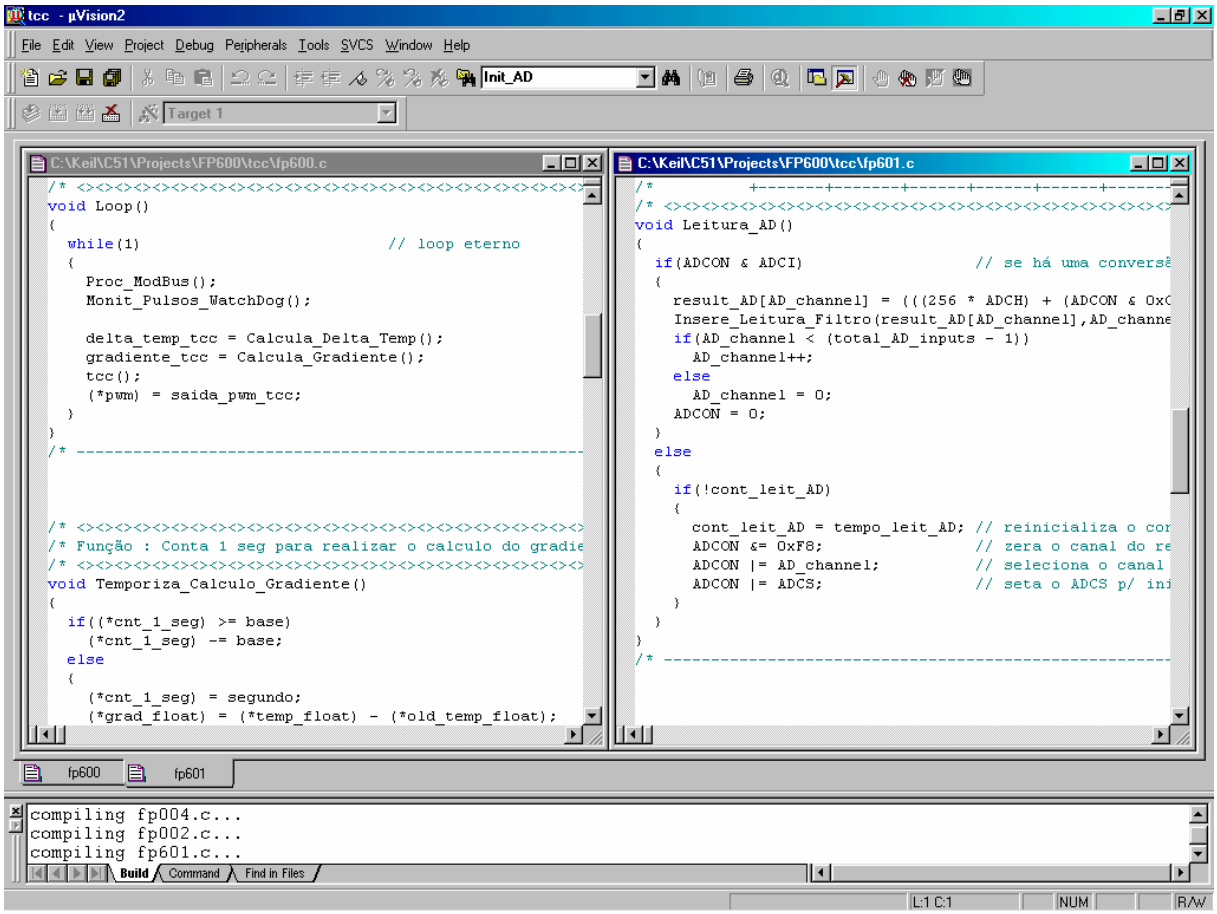
FIGURA 4.21 – FERRAMENTA FUZZYTECH DA INFORM



Esta ferramenta se mostrou muito eficiente para a criação rápida de rotinas de controle baseadas em lógica fuzzy. O programa apresenta várias características úteis. O modo de *debug* interativo, que pode ser visto no centro da figura 4.21, foi muito importante na fase de testes do protótipo, pois permitia comparar os resultados obtidos no protótipo com os gerados pela ferramenta FuzzyTech, e estes foram sempre os mesmos.

Já a parte de comunicação do protótipo bem como leitura da entrada analógica e interpretação dos resultados gerados pelo controle fuzzy foram implementados em linguagem C utilizando o compilador da Keil Software, o C51. A figura 4.22 mostra a interface deste compilador.

FIGURA 4.22 – COMPILADOR C51 DA KEIL



4.4.3 SOFTWARE DE SUPERVISÃO

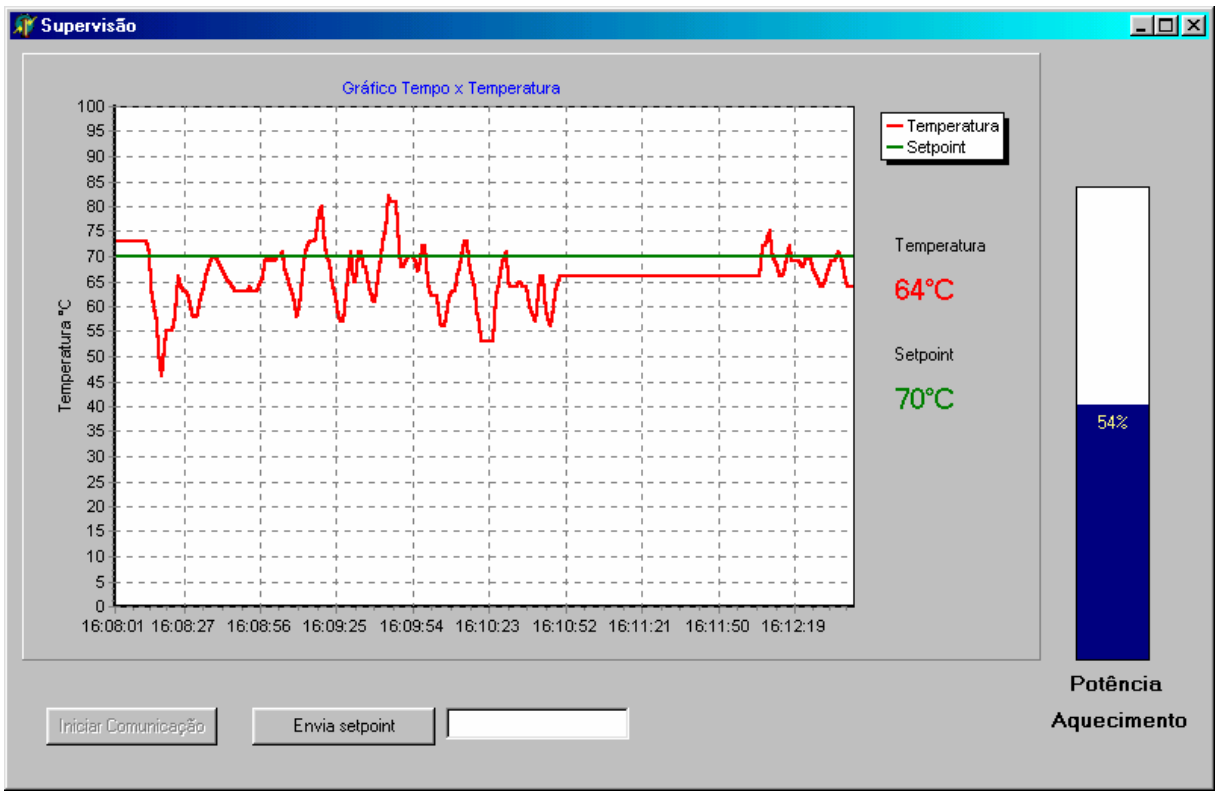
O *software* de supervisão foi implementado no ambiente Delphi 5, e tem por objetivo apresentar graficamente o processo de controle de temperatura realizado pelo protótipo. O ambiente de programação Delphi foi escolhido por ser um dos mais utilizados hoje para desenvolvimento de *software*, além da vasta documentação a seu respeito, e pelas facilidades

que oferece no desenvolvimento de aplicações que utilizam comunicação serial e gráficos. Este *software* consiste de apenas uma tela onde pode-se destacar os seguintes itens:

a) gráfico de tempo e temperatura do tipo linhas, onde são mostradas a temperatura atual lida no sensor e a temperatura programada (*setpoint*);
b) gráfico do tipo barra, mostrando o status de saída do processo, ou seja, o percentual de potência a ser aplicado no elemento de aquecimento;
c) botão de início de comunicação com o protótipo;
d) botão para o envio de *setpoints* ao protótipo.

A figura 4.23mostra a tela do *software* de supervisão.

FIGURA 4.23 – TELA DO SOFTWARE DE SUPERVISÃO



Ao ser iniciado, o programa de supervisão ainda não está coletando dados do protótipo, para tanto é necessário que o usuário pressione o botão “Iniciar Comunicação”.

Ao fazer isto o programa iniciará a comunicação serial com o protótipo. Esta comunicação está configurada para acontecer através da porta COM1, e a um *baudrate* de 19200. O padrão do programa é sempre utilizar a função 3 do protocolo (leitura de registros),

lendo as variáveis temperatura atual, temperatura programada e potência de aquecimento. O programa apenas utiliza a função 6 (escrita de registros), quando o operador preencher a caixa de texto ao lado do botão “Envia setpoint” com um novo valor para a temperatura programada e pressionar o botão. Este valor então é enviado ao protótipo e imediatamente utilizada nos cálculos de controle do processo.

4.4.4 TESTES E VALIDAÇÕES DO PROTÓTIPO

Para considerar alcançados os objetivos do trabalho, alguns itens foram validados:

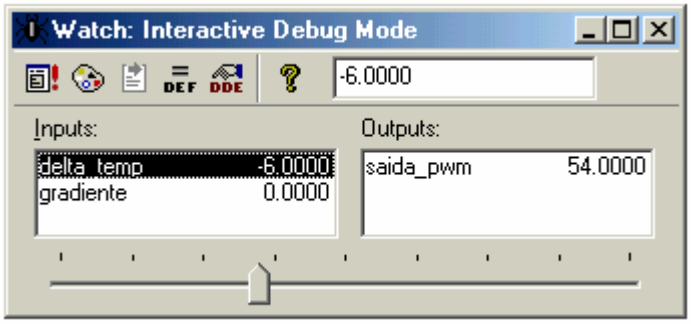
a) teste de funcionamento de todos os componentes do protótipo;
b) teste da comunicação serial (RS-232) entre o protótipo e o PC;
c) apresentação dos dados do processo no software de supervisão;
d) comparação entre os resultados obtidos com o protótipo e os resultados gerados pelo módulo de *debug* da ferramenta FuzzyTech.

As fases de testes a e b, foram realizadas utilizando-se ferramentas comuns ao ambiente da eletrônica, como multímetro e osciloscópio. O objetivo destes testes foi certificar o funcionamento dos componentes básicos do protótipo, como, a fonte de alimentação, o cristal oscilador da CPU, a interface da CPU com as memórias e a interface serial.

O passo seguinte foi testar o funcionamento do software de supervisão. Isso se deu através de uma comparação entre os valores de temperatura apresentados na tela do software e valores conhecidos que foram injetados na entrada analógica do protótipo.

Já o funcionamento do sistema de controle Fuzzy que é executado no protótipo, foi validado através da comparação entre a saída gerada pelo protótipo com determinados valores nas entradas, e a saída que é gerada pela ferramenta FuzzyTech no modo de *debug*, se inseridos os mesmos valores de entrada. A figura 4.24 ilustra este teste:

FIGURA 4.24 – MODO DE *DEBUG* DO SOFTWARE FUZZYTECH



Comparando o resultado do *degub* interativo do software FuzzyTech, visto na figura acima, com os valores lidos do protótipo, que podem ser vistos na figura 4.23, nota-se que, tendo entradas iguais, delta_temp igual -6°C e gradiente igual a 0°C/s, o valor da saída do controlador Fuzzy, tanto no *debug*, como no protótipo, também é o mesmo, 54%.

5 CONCLUSÃO

Os objetivos do trabalho foram atingidos, os quais eram implementar um protótipo de *hardware* baseado no microcontrolador 80C552, implementar um *software* para este protótipo que realizasse o controle de temperatura de um ambiente utilizando os conceitos da lógica fuzzy e disponibilizar estes dados em um *software* de supervisão, para que o processo pudesse ser visualizado graficamente.

Apesar disto o trabalho possui algumas limitações, as quais são citadas abaixo.

a) o protótipo esta programado para realizar apenas o aquecimento;
b) a porta de comunicação e *baudrate* são fixas, obrigando que se utilize a COM1 e *baudrate* de 19200;
c) não foi possível realizar um teste utilizando sensores e elementos de aquecimento reais, apenas simulações do comportamento do processo.

Obervou-se durante o trabalho que a lógica fuzzy possui características fundamentais para a solução de determinados tipos de problemas, principalmente aqueles relacionados com tomada de decisão sobre valores imprecisos, alem de ser bem adaptada para implementações de baixo custo baseadas em sensores, conversores A/D e microcontroladores.

Uma das dificuldades encontradas no uso desta tecnologia, foi a quantidade reduzida de trabalhos mais didáticos que utilizam os conceitos de lógica Fuzzy. Praticamente a pesquisa teórica se limitou a publicações extremamente técnicas, o que, inicialmente, dificulta a compreensão do funcionamento da lógica.

Também os microcontroladores da família MCS51, criados pela Intel a mais de 20 anos, mostram porque ainda são muito utilizados em projetos eletrônicos nas mais diversas áreas. A grande quantidade de instruções, a arquitetura, e sua filosofia de funcionamento fazem dos microcontroladores desta família uma referência em microcontroladores de 8 bits. Mais especificamente o derivado da família MCS51, 80C552, utilizado neste protótipo mostrou ser muito eficaz e versátil em projetos de baixo custo e que necessitem de periféricos como conversores A/D e saídas de PWM.

No que diz respeito ao hardware do protótipo, um dos problemas que surgiu, foi a dificuldade de se encontrar alguns componentes como o 80C552 e o X5043, em lojas

especializadas em componentes eletrônicos da região. A solução foi encomenda-los de distribuidores, que se encontram, em sua maioria, no estado de São Paulo. Há também o custo de fabricação da placa de circuito impresso, que deve ser levado em consideração em projetos nesta área.

5.1 EXTENSÕES

As sugestões para extensões e trabalhos futuros nesta área são:

a) criar mais uma saída no processo responsável por controlar o resfriamento;
b) implementar em conjunto um controlador PID e comparar suas performances;
c) implementar um controle de temperatura onde a velocidade de aquecimento e resfriamento possa ser programada (gradiente de °C/min) .

REFERÊNCIAS BIBLIOGRÁFICAS

ALTIUM INC. **Tango user´s guide.** San Diego: Altium, 1992.

ANLAUF, Dárcio Heinz. **Projeto de um coletor de dados baseado em microcontrolador.** 1993. 107 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

AXELSON, Jan. **Serial port complete:** programming and circuits for RS-232 and RS-485 links and networks. Madison: Lakeview Research, 1998.

BARR, Michael. **Introduction to pulse width modulation**, Maryland, ago. 2001. Disponível em: <http://www.embedded.com/story/OEG20010821S0096>. Acesso em 08 mai. 2002.

CORREIA, Carlos. **Conversores AD e DA.** Coimbra, 2001. Disponível em: <http://lei.fis.uc.pt/ppessoais/correia/Electronica-2001/ADC-DAC.pdf>. Acesso em 09 mai. 2002.

COX, Earl. **The fuzzy systems handbook.** New York: AP Professional, 1994.

INFORM GMBH. **FuzzyTech user's guide.** Aachen: Inform, 2001.

JORDAN, Larry. **Comunicações e redes com o PC.** Rio de Janeiro: Axcel Books, 1994.

KEIL SOFTWARE INC. **Keil user's guide.** Dallas: Keil, 2001.

KLITZKE, Marcelo. **Protótipo de hardware para aquisição e transmissão de imagens via padrão serial RS-485.** 1999. 77 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

KOSKO, Bart. **Neural networks and fuzzy systems.** New Jersey: Prentice Hall, 1992.

MAMDANI, E.H. Application of fuzzy algorithms for control of simple dynamic plant. **Proceedings of the IEEE (Control and Science)**, Piscataway, v.121, p. 298-316, 1974.

PACHECO, Roberto C. S. **Tratamento de imprecisão em sistemas especialistas.** 1991. 85 f. Dissertação (Mestrado em Engenharia de Produção) – Engenharia de Produção e Sistemas, UFSC, Florianópolis.

PEREIRA, Cledy Gonçalves. **Um sistema especialista com técnicas difusas para os limites da agência.** 1995. 91 f. Dissertação (Mestrado em Engenharia de Produção) – Engenharia de Produção e Sistemas, UFSC, Florianópolis.

PHILIPS SEMICONDUCTORS. **80C51 Family derivatives:** 8XC552/562 overview. Sunnyvale: Philips, 1996.

RABUSKE, Renato Antônio. **Inteligência artificial.** Florianópolis: UFSC, 1995.

ROSS, Timothy J. **Fuzzy logic with engineering applications.** New York: McGraw-Hill, 1995.

SILVA JÚNIOR, Vidal Pereira da Silva. **Microcontroladores.** São Paulo: Érica, 1988.

SILVA JÚNIOR, Vidal Pereira da Silva. **Microcontrolador 8051:** hardware e software. São Paulo: Érica, 1990.

TARIG, Ali Abdurrahman E. S. **Controle de um braço robótico utilizando uma abordagem de agente inteligente**. 2001. 98 f. Dissertação (Mestrado em Informática) – Coordenação Pós-Graduação em Informática, Universidade Federal da Paraíba, João Pessoa.

TOSHINORI, Munakata; JANI, Yashvant. **Comunications of the ACM:** fuzzy systems, New York, v.37, p. 69-76, 1994.

WELSTEAD, Stephen T. **Neural network and fuzzy logic applications in C/C++.** New York: Wiley, 1994.

ZADEH, Lotfi A. Fuzzy sets. **Information and control,** San Diego, v. 8, p. 338-353, 1965.

ZADEH, Lotfi A. Outline of a new approach to the analysis of complex systems and decision processes. **IEEE Transactions on Systems Man & Cybernetics**, Tampa, v.3, p. 28-44, 1973.