PAULO ROBERTO VIDAL

SISTEMA DE CONTROLE INDUSTRIAL MICROPROCESSADO

ASSIS
2008

SISTEMA DE CONTROLE INDUSTRIAL MICROPROCESSADO

PAULO ROBERTO VIDAL

Trabalho de Conclusão de Curso apresentado ao
Instituto Municipal de Ensino Superior de Assis,
Como requisito do Curso de Graduação, analisado
Pela seguinte comissão examinadora:

Orientadora:_____
Regina Fumie Eto

Analisador (1):_____
Domingos de Carvalho Villela Junior

Analisador (2):_____
José Augusto Fabri

ASSIS
2008

PAULO ROBERTO VIDAL

SISTEMA DE CONTROLE INDUSTRIAL MICROPROCESSADO

Monografia apresentada no Instituto Municipal de Ensino Superior de Assis ao cumprimento das exigências para a obtenção do título de Bacharel em Ciência da Computação sob a orientação da Prof.a. Ms.Regina Fumie Eto.

Área de Concentração: Ciências Exatas e da Terra.

ASSIS
2008

DEDICATÓRIA

Dedico este trabalho aos meus pais in memorian, a minha família e a minha esposa que foi sempre minha aliada para concretizar este sonho.

AGRADECIMENTOS

Primeiramente, À DEUS por me indicar o caminho e me capacitar a trilhá-lo.

A Prof.a. Regina Fumie Eto pela orientação e pelo constante estimulo transmitido durante este trabalho , que permitiu a realização deste.

Aos professores da Fundação Educacional do Município de Assis (FEMA), que ministram as disciplinas deste curso.

À minha família, principalmente meu pais hoje in memórian que me mostraram o caráter de um cidadão e o valor de um ser humano.
À minha esposa que sempre me deu forças, apoio e compreensão e aos meus filhos que completam nossa felicidade.

Aos meus amigos, pelo apoio, compreensão e ajuda.

RESUMO

O que se desejou investigar neste trabalho foram os resultados da integração do conceito de *software* livre atuando como um sistema supervisorio de automação, associado a um *hardware* que tem um microcontrolador como elemento principal realizando a função de uma interface com equipamentos e sensores primários através da comunicação serial. O sistema permite o controle de pequenas automações industriais controlando variáveis de processo ou mesmo no uso doméstico. Tal composição permite varias vantagens, como a redução do custo de implantação e manutenção quando comparado a um sistema profissional, desempenho com características precisas e seguras. As variáveis de controle aplicada nesta planta de automação serão supervisionadas ininterruptamente através do uso de ferramentas operacionais como , alarmes, relatórios e gráficos de tendências. As ferramentas de programação são gratuitas e os componentes que compõem o *hardware* são de fácil aquisição no mercado.

Palavras chaves: *Software* livre. Automação. *Hardware*. Microcontrolador.

ABSTRACT

The goals of this essay are to show the results of the integration of the free *software* concept, acting as a supervisory system for automation, associated in a *hardware* that has a microcontroller as a principal element, doing an interface function with equipment and primary sensors through the serial communication. The system allows the control of little industrials automation controlling changeable process or even the household use. This allows a lot of advantages, as costs reductions of introduction and maintenance when compared to a professional system, performance with accuracy and safe characteristics. The changeable controls applied in this projects will be always supervised through the use of tools as alarms, reports and tendency's graphic. The programming are free and the components that there are in the software are easy to find out.

Keywords: Free software. Automation. Hardware. Microcontroller.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE TERMOS TÉCNICOS

Bsp: Bits por segundo

CAD: Conversor Analógico Digital

CCP: *Comparação, Captura e PWM*

CCS: Compilador C para microcontrolador

Cisc: *Complex Instruction Set Computing*

DIP: *Dual InLine Package*

***Full Duplex*:** Modelo de transmissão

GPRs: Registradores de propósito geral

IDE: *Interface Development Environment*

mA: miliamperes

MV: Variavél Manipulada

***Off-Set*:** Desvio

On-Off: Liga - Desliga

Overshoot: Ganho excessivo

PIC: Família de microcontrolador

PID: Proporcional Integral Derivativo

PV: Variável de Processo

PWM: *Pulse-width modulation*

Range: Faixa de medição

Risc: *Reduced Instruction Set Computing*

RS-232: Protocolo de comunicação

RX: Recepção de dados

Set Point: Ponto Desejado

SFR: Registrador de uso especifico

Tags: Código

TTL: *Transistor-Transistor-Logic*

TX: Transmissão de Dados

Vca: Volts corrente alternada

Vdc: Volts corrente continua

SUMÁRIO

1 INTRODUÇÃO

A automação industrial tem sido um dos pilares para o crescimento fabril em qualquer país, pois existe uma constante necessidade de melhorias nas linhas de produção, envolvendo qualidade final do produto, produtividade e baixos custos. Em vários segmentos de mercado houve um aumento das empresas terceirizadas que operam para grandes investidores, com a obrigação e necessidade de manter a mesma a qualidade original. Paralelamente as empresas voltadas para o segmento de automação investem em desenvolvimento de sensores, *software* e componentes direcionados para as indústrias, aproveitando este momento de crescimento global. Assim esses componentes ganharam significativas melhoras, em qualidade, funções, compactação e custo.

As pequenas e/ou micro empresas também necessitam de pequenas automações em seu processo produtivo, mas que sejam robusta, precisa, confiável, proporcionem históricos do processo e principalmente baixo custo de implementação. Quando citado o caso de pequenas automações não se tem um numero pré-definido para determiná-lo, pois dependem de vários fatores como tipo de aplicação, segurança da lógica, local e tipo de operação. Porém aplicando o conceito apresentado neste trabalho, pode ser considerada uma pequena automação o controle e/ou indicação com até cinco variáveis.

1.1 JUSTIFICATIVAS E MOTIVAÇÕES

Apesar do crescimento da tecnologia voltada ao segmento de automação e instrumentação de processos industriais, ainda existem particularidades que a implementação com equipamentos industriais torna-se inviável comercialmente, como exemplo a produção de verduras em cultivo hidropônico, fornos de cerâmicas entre outras, onde necessitam de controle e/ou indicação de três ou quatro variáveis. Por esse motivo abre-se uma porta comercial para tecnologias com custo menor.

1.2 OBJETIVOS

Conforme citado no item 1.1 este trabalho busca o baixo custo em automação viabilizando a integração de software livre com sensores de preço reduzido,

originando um sistema de supervisão e interface lógica de controle, além de colocar em pratica os ensinos acadêmicos adquiridos em sala de aula.

1.3 ESTRUTURA DO TRABALHO

Capitulo 1: descreve introdução, justificativas e objetivos do trabalho.

Capitulo 2: descrevem princípios da automação, instrumentação e suas terminologias, ações de controle e sistema de supervisão.

Capitulo 3: descrevem os microcontroladores e suas principais características, em especial o PIC 16F877A.

Capitulo 4: descreve o projeto físico da planta, seus componentes e o funcionamento pratico.

Capitulo 5: descreve a programação do microcontrolador e as linguagens de programação empregadas para o sistema de gerenciamento da planta.

Capitulo 6: descreve a conclusão com os resultados alcançados e trabalhos futuros.

2 AUTOMAÇÃO DE PROCESSOS

2.1 INTRODUÇÃO

No texto [Eurogam,2008], automação é um conjunto das técnicas e dos sistemas de produção fabril baseado em máquinas com capacidade de executar tarefas previamente executadas pelo homem e de controlar seqüências de operações sem a intervenção humana. Baseado em aparelhos programáveis a automação industrial tem capacidade de operar quase independentemente do controle humano (como acontece nos domínios das telecomunicações, da aeronáutica e da astronáutica).

A automação incorpora diretamente a instrumentação de processos, pois são esses os elementos primários de campo que geram sinais para os equipamentos de automação realizar suas tarefas.

Desenvolvimento da automação: as primeiras iniciativas do homem para mecanizar atividades manuais ocorreram na pré-história. Invenções como a roda, o moinho movido por vento ou força animal e as rodas d'água demonstram a criatividade do homem para poupar esforço. A partir de 1870, também a energia elétrica passou a ser utilizada e a estimular indústrias como a do aço, a química e a de máquinas-ferramenta. Os sistemas inteiramente automáticos surgiram no início do século XX, passando a contar com computadores, servomecanismos e controladores microprocessados e programáveis. Os computadores são os alicerces de toda a tecnologia da automação contemporânea.

Nas ultimas décadas, a necessidade do aumento de produção para atender a crescente demanda aliada a um baixo custo e o desenvolvimento de novos produtos, propiciou o surgimento de um número cada vez maior de indústrias. Estas indústrias só puderam surgir entre outros fatores, devido a um Controle Automático de Processos Industriais, sem o qual a produção não seria de boa qualidade e mesmo alguns produtos não poderiam ser fabricados.

2.2 AUTOMAÇÃO

Noções preliminares: como em outros segmentos a terminologia como exposta no texto [Ecil,2008] é de suma importância para compreensão do contexto técnico. Cada sistema de automação compõe-se de cinco elementos básicos:

Acionamento: provê o sistema de energia para atingir determinado objetivo. É o caso dos motores elétricos, pistões hidráulicos etc.;

Sensoriamento: mede o desempenho do sistema de automação ou uma propriedade particular de algum de seus componentes. Exemplos: sensores de temperatura, posição e demais variáveis.

Controle: utiliza a informação dos sensores para regular o acionamento, para atingir um valor de *Set point*

Comparador ou **elemento de decisão**: compara os valores medidos com valores preestabelecidos e toma a decisão de quando atuar no sistema. Como exemplos, podemos citar os programas de computadores;

Programas: chamado de *software* contém informações de processo e permitem controlar as interações entre os diversos equipamentos através de instruções lógicas, seqüencialmente organizadas. Indicam ao controlador ou ao computador o que fazer.

2.3 INSTRUMENTAÇÃO INDUSTRIAL

Instrumentação pode ser definida [Amadeo,2002], como a ciência que se ocupa em desenvolver e aplicar técnicas de medição, indicação, registro e controle de processos de transformação visando a otimização da eficiência dos mesmos. Para uma melhor compreensão seguem as terminologias empregadas na instrumentação de processos.

Processo: pode ser explicado em como sendo as funções coletivas executadas no processo e pelo equipamento no qual uma variável é controlada. Então, podemos concluir que o termo “processo” engloba tudo aquilo que afeta a variável controlada. operação ou série de operações no qual o valor de uma quantidade ou condição é controlado. Inclui todas variáveis das funções que, direta ou indiretamente, afetam o valor da Variável Controlada.

Variáveis de Processos ou PV: são Fenômenos físicos que chamamos simplesmente variáveis, por exemplo: vazão, temperatura, pressão, nível, densidade, etc.

Variável Manipulada ou MV: variável sobre a qual o controlador atua para controlar o processo, como posição de uma válvula, tensão aplicada a uma resistência de aquecimento, etc.

Instrumentos: medem variáveis de processo.

Precisão: é o maior valor de erro estático, ao longo da faixa de medição. Pode ser expressa em unidade de grandeza física, porcentagem do valor medido, porcentagem do valor total da escala.

Tempo de Resposta: intervalo de tempo entre o instante em que um estímulo é submetido a uma variação brusca e o instante em que a resposta alcança seu valor final e nele permanece, dentro de limites especificados.

***Set point* :** valor desejado para manter a variável de processo

***Off Set* ou Desvio: d**esvio entre o valor desejado e o valor real da PV

De um modo geral os elementos de controle contêm:

Elemento Primário: componente que está em contato com a variável de processo e tem por função, transformá-la em uma grandeza mensurável por um mecanismo.

Transmissor: instrumento que mede uma determinada variável, e envia um sinal proporcional à distância, a um indicador, registrador, controlador, etc.

Elemento Final De Controle: dispositivo que está em contato direto com a variável manipulada, modificando-a em resposta a um sinal de comando.

2.3.1 Controle de Processos Industriais

Em [Matias,2002] é definido que controlar um processo industrial, não é basicamente manter os valores das variáveis de processo dentro de uma faixa aceitável de operação conveniente, mas ainda buscar dentro de cada faixa, o valor ótimo para cada variável. Para poder controlar automaticamente um processo é necessário saber do seu comportando e corrigi-lo se necessário, fornecendo ou retirando dele alguma forma de energia, como por exemplo: pressão ou calor. Essa atividade de medir e comparar grandezas são feitas por equipamentos ou instrumentos quer eletrônicos, pneumáticos, hidráulicos ou mecânicos. Para entender o conceito de controle, a figura 1, descreve de forma sucinta.

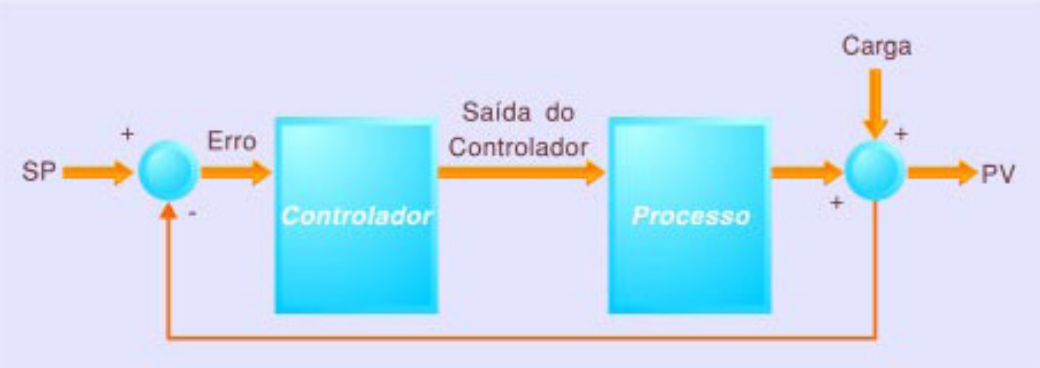

Figura 1. Representação em bloco do controle automático

2.3.2 Ação de Controle

Pode ser reversa ou direta. Define genericamente a atuação aplicada à MV na ocorrência de variações da PV. Uma ação de controle bem ajustada, permite obter um resultado satisfatório no controle das variáveis.

Encontrar os parâmetros da PID (proporcional, integral derivativo) atende critérios tais como:

- Conseguir variabilidade mínima em operação normal;
- Mínimo (ou nenhum) "*overshoot*" para mudanças de *Set point*;
- Atingir rapidamente o novo "S*et point*" em caso de mudança;
- Os itens acima refletem diretamente na qualidade final do produto, economia; de energia e de insumos, entre outros fatores.

2.3.3 Controle Proporcional - Integral - Derivativo

Para um controle atuar de forma satisfatória atuando nas variáveis de processo mantendo a estabilidade, são aplicadas determinadas funções matemáticas, que interpretam a entrada de sinal dos transmissores, efetua a comparação com o *Set point* e envia um sinal de saída para o elemento final de controle em função do erro encontrado. A velocidade, a amplitude e a antecipação da correção estão ligadas diretamente com as ações de controle PID.

De acordo com a descrição no texto [Novus,2008], de uma maneira bem simples, as ações de controle é a composição de três ações quase intuitivas, conforme resume a seguir:

P= Proporcional: a correção a ser aplicada ao processo deve crescer na proporção que cresce o erro entre o valor real e o desejado. (CORREÇÃO PROPORCIONAL AO ERRO). Ver figura 2

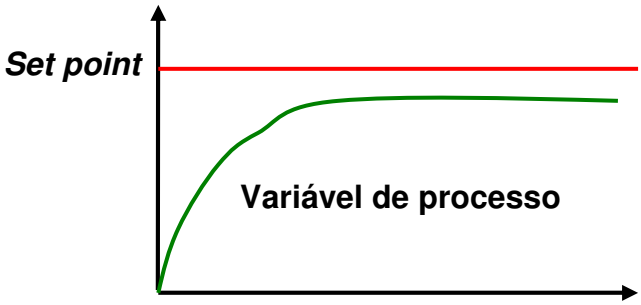

Figura 2. Representação da correção da ação proporcional

I= Integral: erros pequenos, mas que existem há muito tempo requerem correção mais intensa. (CORREÇÃO PROPORCIONAL AO PRODUTO ERRO x TEMPO). A integral não é, isoladamente, uma técnica de controle, pois não pode ser empregado separado de uma ação proporcional. A ação integral consiste em uma resposta na saída do controlador (MV) que é proporcional à amplitude e duração do desvio.

A ação integral tem o efeito de eliminar o desvio característico de um controle puramente proporcional. Para uma melhor compreensão, no gráfico da figura 3 é demonstrada a reação da variável em função da ação integral, sendo a linha vermelha o S*et point*. A adoção de um termo integral excessivamente atuante pode levar o processo à instabilidade.

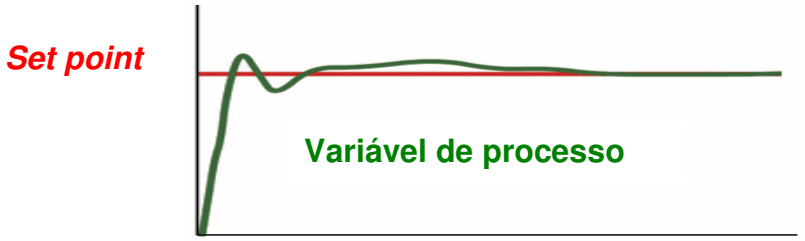

Figura 3. Representação da correção da ação proporcional + integral

D= Derivativa: o derivativo não é, isoladamente, uma técnica de controle, pois não pode ser empregado separado de uma ação proporcional. A ação derivativa consiste em uma resposta na saída do controlador (MV) que é proporcional à velocidade de variação do desvio.

A ação derivativa em [Novus,2008] tem o efeito de reduzir a velocidade das variações de PV, evitando que se eleve ou reduza muito rapidamente (CORREÇÃO PROPORCIONAL À TAXA DE VARIAÇÃO DO ERRO). O derivativo só atua quando há variação no erro. Se o processo está estável, seu efeito é nulo. Durante perturbações ou na partida do processo, quando o erro está variando, o derivativo sempre atua no sentido de atenuar as variações sendo, portanto sua principal função melhorar o desempenho do processo durante os transitórios.

Um pouco de matemática: uma das equações mais usual do PID é apresentada a seguir [Novus,2008]:

$$MV(t) = Kp \times \left[E(t) \quad + \quad Ki \times \int E(t)dt \quad + \quad Kd \times \frac{dE(t)}{dt} \right]$$

Onde Kp, Ki e Kd são os ganhos das parcelas P, I e D, e definem a intensidade de cada ação. Equipamentos PID de diferentes fabricantes implementam esta equação de diferentes maneiras. É usual a adoção do conceito de “Banda Proporcional” em substituição a Kp, “Tempo derivativo” em substituição a Kd e “Taxa Integral” ou “*Reset*” em substituição a Ki, ficando a equação da seguinte forma.

$$MV(t) = \frac{100}{Pb} \times \left[E(t) \quad + \quad Ir \times \int E(t)dt \quad + \quad Td \times \frac{dE(t)}{dt} \right]$$

Onde Pb, Ir e Td estão relacionados a Kp, Ki e Kd e serão individualmente abordados ao longo deste texto. Porém outras funções matemáticas podem ser usadas, originado o mesmo tipo de correção

2.3.4 Controle *On-Off*

Baseia-se na comparação do sinal fornecido pelo sensor com o sinal gerado a partir do *Set point* selecionado no controle. O comportamento da variável controlada equivale a uma oscilação próximo aos valores equivalentes aos comandos *On-Off* do controle. A figura 4, ilustra a resposta de um sistema sob controle *On-Off*, mostrando que a oscilação não é necessariamente senoidal [Coel,2008].

A linha vermelha indica o valor desejado da variável controlada; observe que a média não equivale necessariamente ao valor desejado. O controle O*n-Off*, evidentemente, não consegue manter a variável em um S*et point*. Este tipo de

controle é normalmente aplicado onde não necessita uma variável estabilizada, porém pode ser aplicada nos mais diversos controles de nível, pressão, temperatura.

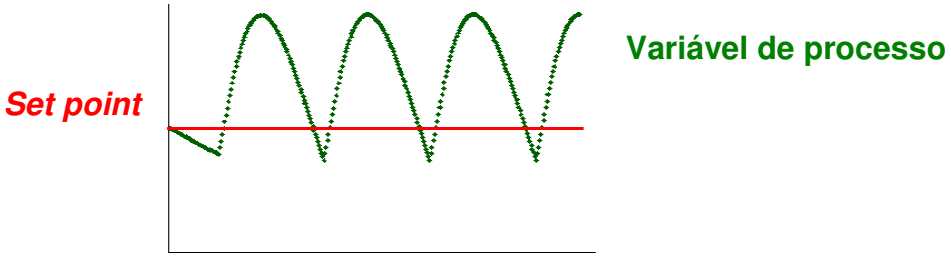

Figura 4. Representação de um controle *On-Off*

2.4 SISTEMAS DE SUPERVISÃO E CONTROLE

Um sistema de supervisão e controle [Unifebe,2008], é um conjunto definido por estações de operação ou também chamadas de cliente, que são representadas por microcomputadores, instrumentos de aquisição e/ou controle e um *software* de supervisão. Tem como funções fornecer informações sobre o processo de forma numérica e gráfica, iniciar e interromper os vários processos, armazenar as ocorrências geradas pelos equipamentos e pelo operador, emitir relatórios, realizar cálculos, controle estatístico entre outras funções.

Com todas essas informações disponibiliza aos operadores intervirem no processo quando necessário, através de uma interface gráfica amigável como ligar/ desligar bombas, abrir/fechar válvulas. Um sistema de supervisão é composto de vários aplicativos.

2.4.1 Base de Dados

A parte central do *software* de supervisão é a base de dados, onde são definidas todas as variáveis que serão utilizadas, ficando esta com função de concentrar e conectar todos os aplicativos disponíveis no sistema de supervisão.

2.4.2 Telas de Sinóticos

As telas de sinótico podem assumir vários formatos, sendo o desenvolvimento de telas muito particular para cada sistema e para cada configurador, mostrando de forma bastante simplificada uma visão geral do processo.

2.4.3 Tela de Alarmes

Tabela alfanumérica contendo os alarmes ativos, seus estados, reconhecidos ou não, a condição de alarme, crítico ou não, horário de ativação, reconhecimento e desativação.

2.4.4 Tela de Gráfico de Tendências

A tela de histórico tem grande funcionalidade na operação da planta, pois auxilia na parametrização das ações de controle, variações do processo com datas e horas, mostrando os gráficos *on-line* das variáveis e os valores já armazenados anteriormente [Unifebe,2008].

3 MICROCONTROLADOR

3.1 INTRODUÇÃO

O desenvolvimento dos circuitos integrados permitiu ser possível armazenar centenas de milhares de transistores num único *chip*. Um crescente aumento do nível de integração permitiu que os CI (circuitos integrados) tivessem simultaneamente processadores e periféricos. Surge então o primeiro *chip* contendo um microcomputador, designado por microcontrolador.

Basicamente um microcontrolador [Souza,2007] é um pequeno componente eletrônico, dotado de uma "inteligência" programável, utilizado no controle de processos lógicos. O controle de processos neste caso são os periféricos como: motor, relé, equipamentos industriais como inversores de freqüência, *nobreak*, sensores. São lógicos, pois as ações se baseiam em ações lógicas que devem ser executadas.

3.2 MICROCONTROLADOR DA FAMILA PIC

É a família de microcontroladores fabricados pela *Microchip Technology* utilizam a arquitetura *RISC*. Os mesmos podem funcionar com freqüências até 40 Mhz. São divididos em três grupos diferenciados pela capacidade de armazenamento da sua memória de programa: 12 *bits*, 14 *bits* e 16 *bits*. [*Microchip,*2008]

3.2.1 Arquitetura *RISC* e *CISC*:

Microcontroladores do tipo *CISC* (*Complex Instruction Set Computing*) possuem muitas instruções, portanto são mais fáceis de programar, uma vez que o programador sempre pode utilizar instruções mais elaboradas para realizar uma determinada tarefa, reduzindo o número de linhas do programa. Já os do tipo *RISC* (*Reduced Instruction Set Computing*) possuem um número reduzido de instruções, sendo mais difíceis de serem programados, pois uma tarefa mais complexa deve ser realizada por seqüências de instruções mais simples. Os microcontroladores *RISC*, no entanto, apresentam-se como menor custo e mais rápidos em aplicações gerais.

3.2.2 Memória

Os microcontroladores da família PIC possuem barramentos diferenciados para as memórias de programas e dados, conseqüentemente as memórias são totalmente separadas. Pode ser citado quatro tecnologias para esse fim.

RAM (*Random Access Memory*): os registradores com finalidades especificas (SFRs) e pelos registradores de propósito geral (GPRs), que são áreas da memória RAM do microcontrolador utilizadas diretamente no processamento, assim como a área para memória de dados. O acesso a esses registradores é feito pelo seu endereço de localização na memória RAM [Pereira,2002]. A *Microchip* implementou a filosofia de paginação, criando bancos de memória, cada um com 128 posições**.**

EPROM (*Erasable Programmable Read-only Memory*): alguns modelos de PIC possuem ainda uma terceira memória que também pode ser utilizada pelo usuário para guardar dados, sendo considerada uma memória não volátil, que consegue manter as informações mesmo sem alimentação.

***FLASH*:** os mais versáteis dispositivos da família PIC são os que possuem memória do tipo *flash.* Tais dispositivos permitem um mínimo de 1000 ciclos de gravação/apagamento. As maiores vantagens desse tipo de memória é sua ocupação mínima de espaço, seu baixo consumo de energia, sua alta resistência, sua durabilidade e segurança [Souza,2007].

3.2.3 Principais Registradores

Os registradores têm a função de guardar a configuração e o estado de funcionamento da maquina. Dentre os registradores SFR, podem ser destacados [Zanco,2005]:

STATUS – utilizado para armazenamento de *flags* matemáticos e do estado da CPU, além dos *bits* de seleção do banco de memória RAM.

INTCON – utilizado para controle de interrupções

PORTx – utilizado para a leitura ou escrita de informações nos pinos externos.

3.2.4 Recursos Avançados

PORTAS: esses registradores servem para configurar os pinos de entrada e saída [Zanco,2005]. Quando colocado “1” em um *bit* do registrador TRISA, o pino

relacionado a ele é configurado para entrada. Para configurar como saída o *bit* relacionado deve ser colocado em “0”.

PWM (*Pulse-width modulation*): o PWM é provavelmente o recurso mais utilizado, já que possibilita criarmos uma saída analógica; porque quando uma onda PWM passa por um filtro externo pré determinado, pode ser convertida em um sinal variável de 0 a 5 Vdc [Zanco,2005]. Este sinal pode ser aplicado a um módulo de potência para controle de periféricos como velocidade de motor, iluminação e similares. O ciclo ativo, conforme, é a parte do ciclo em que o sinal permanece em nível 1. O sinal de PWM atua diretamente no disparo dos tiristores como no caso pratico de um inversor de freqüência. A figura 5 demonstra um sinal de PWM e permite visualizar o seu conceito.

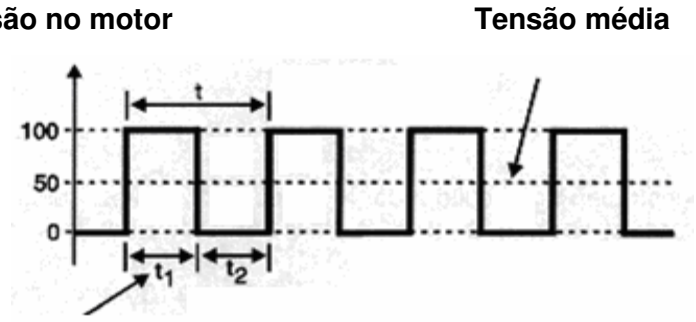

Figura 5. Representação da onda PWM

3.2.5 *USART (* Interface *Serial Universal síncrona / assíncrona):*

Este recurso de *hardware* dentro do microcontrolador, pois possibilita a comunicação com o mundo exterior de forma simples e eficiente. Com esse sistema é possível programar uma comunicação com um Computador, através do protocolo RS-232.

3.2.6 Interrupções

Interrupção é um evento externo [Pereira,2002] ao programa que provoca a parada de sua execução, a verificação e o tratamento do referido evento em seguida o retorno do programa ao ponto em que foi interrompido. As estruturas de interrupção são usadas para que a CPU, tome conhecimento de eventos de alta prioridade para o programa.

3.3 MICROCONTROLADOR PIC 16F877A

3.3.1 Descrição

A base do *hardware* [Souza,2005], empregado neste projeto é o microcontrolador PIC 16F877A. Este componente tem a função de realizar a interface entre os sinais analógicos de saída dos sensores de campo e sinais de saída digital de controle pelo sistema de supervisão. Este microcontrolador pertence a família de 8 *bits* e núcleo de 14 *bits* fabricado pela *Microchip Technology.*

3.3.2 Caracteristicas Principais

Possui memória *flash* de programa com 8192 palavras de 14 *bits*, memória RAM com 368 *bytes* e memória EEPROM com 256 *bytes*. Sua frequência de operação (*clock*) vai até 20MHz, resultando em uma velocidade de processamento de 5 *MIPS*. Seu conjunto de instruções *RISC* se compõe de 35 instruções.

O controle de saída tipo PWM [Souza,2005] faz parte do modulo CCP (*Capture,Compare* e PWM) do microcontrolador. Esse PIC possui dois canais de PWM (CCP1 e CCP2), cada um com uma resolução de 10 *bits* ou 1024 pontos. Pode funcionar com alimentação de 2V a 5,5V.

Complemento:

- Encapsulamento DIP (*Dual In Line Package*) com 40 pinos
- 05 conjuntos de portas de entrada e saída (total de 33 portas)
- (CAD) Conversor analógico-digital de 10 *bits* de resolução e 8 entradas
- Periférico de comunicação paralela e serial (USART e MSSP)
- 02 Módulos CCP (*Comparação, Captura e* PWM)
- 03 *Timers* (1 de 16 *bits* e 2 de 8 *bits*)
- *Watchdog timer*

3.3.3 Conversor Analógico Digital - Interno

O CAD utilizado no PIC 16F877A utiliza o método de aproximação sucessiva. O conversor interno é de 10 *bits*, dando um total de 1024 pontos. Todas as referências de sinais usados no programa são de 0 a 1024. O sensor LM35 usado nesta planta fornece uma saída analógica, proporcional a temperatura, portanto precisamos de um CAD para que o microcontrolador possa tratar esse sinal. A resolução é dado diretamente pelo seu numero de *bits* e pode ser expresso por:

Resolução = (V_{ref}) / 2^{n}

Onde Vref é uma tensão de referência que serve de fundo de escala para o sinal de saída do conversor e *n* o numero de *bits* do conversor. Cada um dos n de *bits* que compõem a informação digital representa uma parcela do valor da tensão analógica a ser convertida, de forma que a soma de todas as tensões formará o valor de entrada do CAD. Apenas os *bits* em 1 representam algum valor de tensão.

3.4 DEFINIÇÃO DO PIC NA MONTAGEM

Apesar do custo aproximado de R$ 20,00 e o seu concorrente o PIC 16F628A ter um valor em torno de R$ 5,00, suas vantagens para possiveis expansão como nesta aplicação, são maiores. Conforme já exposto o PIC 16F877A apresenta um encapsulamento de 40 pinos conforme figura 6, disponibilizando maior numero de entradas e saídas. Na tabela 1 segue um comparativo entre ambos.

Descrição	PIC 16F877A	PIC 16F628A
Portas de entrada e saída	33	16
Saídas PWM	02	01
Memória *flash*	8192 palavras	2048 palavras
Mémoria RAM	386 *Bytes*	224 *Bytes*

Tabela 1 – Caracteristicas do PIC 16F877A

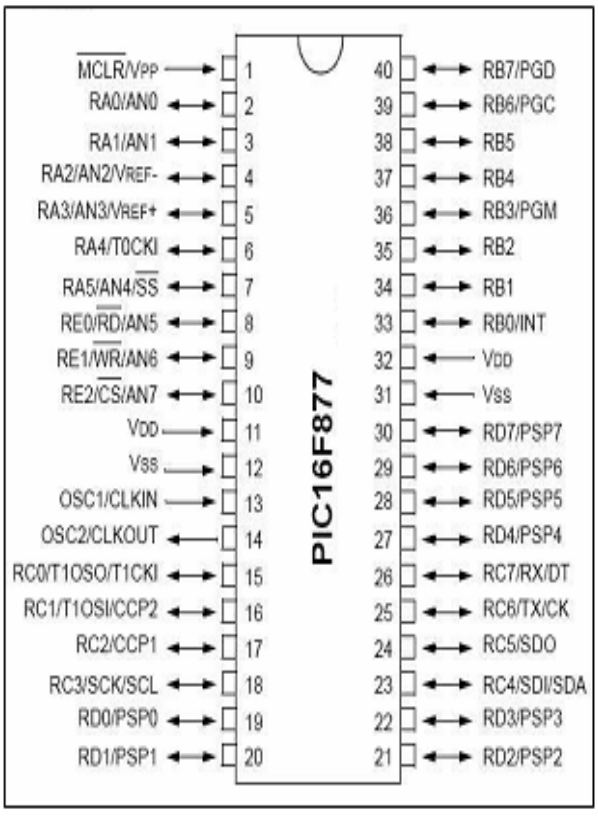

Figura 6. PIC 16F877A.

3.5 MPLAB – versão 5.70

O MPLAB IDE (*Interface Development Environment*), [*Microship*,2008], é um programa que tem a função de um gerenciador, para o desenvolvimento de projetos com a família PIC de microcontroladores. É distribuído gratuitamente pela *Microchip*, fabricante dos PIC's. O MPLAB integra num único ambiente o editor de programa fonte, o compilador, o simulador e quando conectado às ferramentas da *Microchip* também integra o gravador do PIC, o emulador etc.

O Compilador é o programa que converte o arquivo fonte em códigos de máquina. O Simulador é programa que simula o funcionamento da CPU , conforme o programa fonte que está sendo desenvolvido [*Microchip*,2008]. O Projeto no MPLAB é um conjunto de arquivos e informações que diz ao ambiente integrado qual o PIC que está sendo usada, freqüência de *clock*, a linguagem de programação usada, o *layout* das janelas etc. Este programa se integra ao ambiente *Windows*, permitindo cópia de arquivos, de textos de um aplicativo para outro de uma forma bem simplificada.

3.6 LINGUAGENS DE PROGRAMAÇÃO

3.6.1 Linguagem *Assembly*

A criação de programas para microcontroladores pode ser uma tarefa desgastante a medida que aumenta a complexidade da aplicação. A linguagem *Assembly* ainda é a mais usada para os microcontroladores. Consiste em uma forma de representação dos códigos de maquina usando mnemônicos, ou seja, abreviações de termos usuais que descrevem a operação efetuada pelo comando em código da maquina.

A conversão dos mnemônicos em códigos binários executáveis pela maquina é feita por um tipo de programa chamado *Assembler* (montador). A linguagem Assembly é de baixo nível, não possuindo nenhum comando, instrução ou função além daqueles definidos no conjunto de instruções do processador utilizado. Isto implica em serviços extras do programado para desenvolver rotinas e operações.

Porém existem as vantagens de usar essa linguagem:

- Eficiência devido à proximidade com o *hardware* da maquina;
- Devido à eficiência, esses programas são mais rápidos na execução.

3.6.2 Linguagem C – DEVC++ versão 4.9.9.2

Atualmente a maioria dos microcontroladores disponíveis no mercado, contam com compiladores de linguagem C para o desenvolvimento do *software* [Pereira,2007]. O desenvolvimento com a linguagem C, considerada de alto nível, tornou-se mais rápido devido às facilidades de programação oferecidas pela linguagem e também por sua portabilidade. Também devido a sua proximidade com o *hardware*, essa linguagem é extremamente eficiente. Além disso, a linguagem permite que o programador se preocupe mais com a programação da aplicação em si, já que o compilador assume para si as tarefas como controle e localização das variáveis, operações matemáticas e lógicas etc.

- Uso do compilador no MPLAB
- Funções especiais para os microcontroladores
- Biblioteca de funções da **CCS**

3.6.3 Definição de *Software* Livre

O *software* livre [FSF,2008] é uma questão para os usuários terem exercerem a liberdade de executar, copiar, distribuir, estudar, modificar e melhorar o *software*. Mais precisamente, ele se refere a quatro tipos de liberdade, para os usuários do *software*:

Liberdade 0: a liberdade de executar o programa, para qualquer propósito.

Liberdade 1: a liberdade de estudar como o programa funciona, e adaptá-lo às suas necessidades.

Liberdade 2: acesso ao código-fonte é um pré-requisito para esta. A liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo.

Liberdade 3: a liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos ao público, de modo que toda a comunidade se beneficie. Acesso ao código-fonte é um pré-requisito para esta. A liberdade de utilizar um programa significa a liberdade para as pessoas ou organização em utilizá-la em qualquer tipo de sistema de computador, para todo tipo de trabalho global. Para que essas liberdades sejam reais, elas devem ser irrevogáveis, desde que você faça nada errado; caso o desenvolvedor do *software* tenha o poder de revogar a licença, sem fazer nada para dar o seu motivo, o *software* não é livre.

4 DESCRIÇÃO DO PROJETO DA PLANTA

4.1) INTRODUÇÃO

Para demonstrar o aplicativo do controle de processo microprocessado, houve a necessidade da montagem de uma estrutura adequada, envolvendo a montagem propriamente dita, a instalação dos equipamentos elétricos, sensores, a interface com o microcontrolador e a programação. Os passos para obtenção do sucesso da aplicação se deu primeiramente a um escopo previamente montado conforme segue:

- Aquisição dos itens básicos e componentes eletrônicos;
- Testes com o compilador CCS para PIC
- Desenvolvimento da programação básica no PIC 16F877A;
- Desenvolvimento de programação do projeto principal
- Desenvolvimento do protocolo de comunicação ;
- Desenvolvimento da etapa de comunicação PIC / *Java*;
- Criação do Banco de Dados;
- Criação da interface do sistema supervisorio;
- Testes integrado ao sistema;

Todos os passos acima foram seguidos individualmente por vários testes de performace. O material empregado e a montagem de suporte físico seguem na seqüência:

- Placa eletrônica com microcontrolador, contendo conversor RS232/TTL, o PIC 16F877A, *leds* indicadores de *status*, cristal oscilador e demais componentes conforme figura 7.
- Motor / Bomba: bomba d'água, utilizado em automóveis;
- Aquecedor: resistência elétrica de 80 *Watts* x 220 Vac;
- Relé: componente eletromecânico para comutação da bomba d'água;
- Sensor de temperatura LM 35, montada em bainha de aço inox;
- Sensor de pressão instalado no tanque 2;
- Reservatório plástico transparente de 20 litros;
- Computador para sistema supervisorio;

- Cabos de comunicação, cabos elétricos, conectores;
- Bateria de 12 Vdc para alimentação do sistema bomba d'água.

Montagem: os dois reservatórios – tanque 1 e tanque 2, foram montados verticalmente em estrutura de madeira devidamente suportada. O motor fica alojada no tanque 1; o bombeamento, a conexão do transmissor de nível e a válvula de realimentação de água do tanque 2 para o tanque 1 se dá através de mangueira plástica de 5mm. A placa eletrônica principal tem alimentação de fonte de 12 Vdc x 250 mA, cabo de comunicação serial RS232, cabos de sinal para controle *On-Off* e saída PWM para o motor.

Para uma melhor visualização foi colocado corante azul na água.

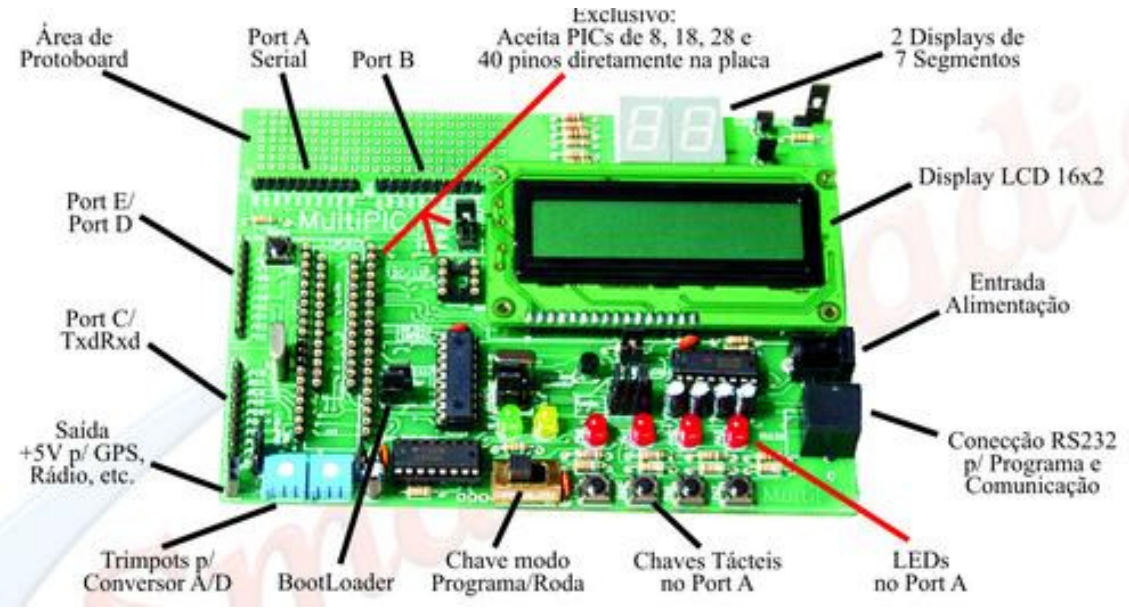

Figura 7. Placa eletrônica - Multipic

4.2 SENSOR DE TEMPERATURA LM35

O sensor LM35, conforme figura 8 é um sensor de precisão, fabricado pela *National Semiconductor* [*Datasheet*,2008]. As séries de circuitos integrados LM35 são sensores de temperatura de precisão, cuja tensão de saída é diretamente proporcional a temperatura em Graus Celsius (centígrados). O LM35 não requer qualquer calibração externa ou calibração para fornecer precisões típicas de ± 0,25 ℃ . Outras características:

Faixa de trabalho de -55 a +150 ℃;

Relação: fator Linear + 10,0 mV / °C ;

Precisão: 0,5 °C (a +25 °C);

Tensão de operação: a partir de 4 a 30 volts

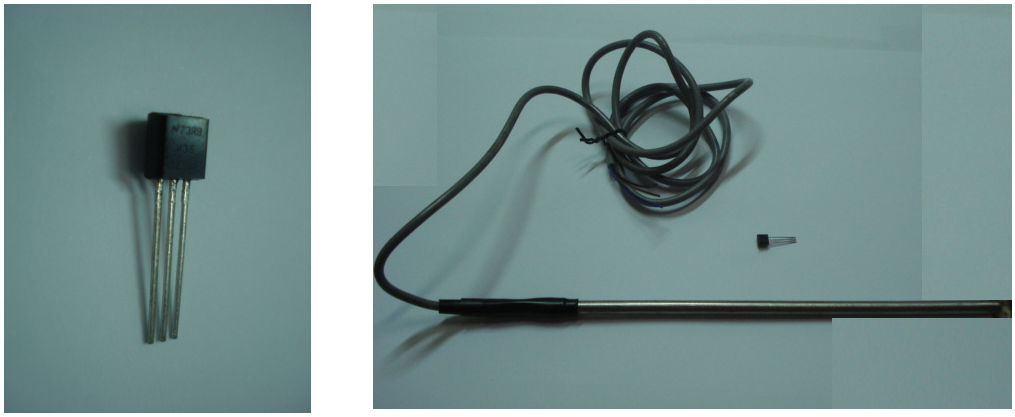

Figura 8 – Sensor LM 35

4.3 SENSOR DE PRESSÃO

Conforme o texto técnico em [Sun,2008], este sensor consiste em um do transdutor piezoresistivo, com compensação interna de temperatura. É um sensor de pressão projetado para uma ampla gama de aplicações, mas especialmente empregando em microcontrolador com CAD . Fornece sinal de forma precisa, alto nível de sinal analógico de tensão proporcional à diferença de pressão entre suas duas entradas ou seja, na figura 9, é possível observar as entradas de pressão, que estará conectada na parte inferior do tanque e a segunda entrada em contato com a pressão atmosférica, temos um sinal proporcional à pressão exercida pela coluna de água do tanque. Outras características:

- Sensor de membrana de silício com transdutor: piezoresistivo
- O range de pressão: 0 até 10 KPa,
- Saída de sinal: 0 ~ 5 Vdc.
- Compensação de temperatura: 0ºC até 85 ºC e *offse*t.

Modelo: MPXV5004DP

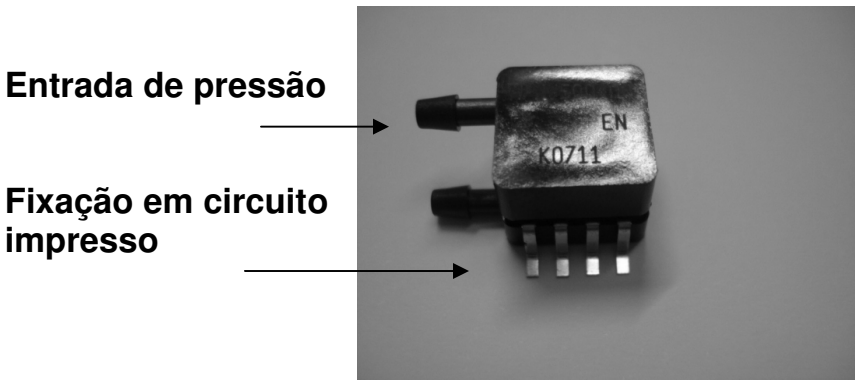

Figura 9 – Transmissor de pressão diferencial

4.4 *DRIVER* DE CORRENTE L298N

Este circuito integrado tem a finalidade de servir como drive de corrente, ou seja a partir da entrada de sinal, o mesmo consegue drenar corrente até 2,5 A e tensão de trabalho até 50 Vdc. Figura 10

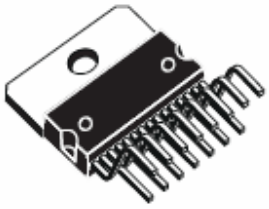

Figura 10 – L298N

Conforme a figura 11, o componente é constituído internamente de duas pontes de transistores (duas saídas) em configuração H, para chaveamento em altas freqüências – como o caso do sinal de PWM para o motor. Características importantes:

- Vs – Alimentação até 50 Vdc – pino 4
- Vss – Alimentação lógica de 5 ~ 7 Vdc – pino 9
- Vi, Ven – Tensão de entrada e habilitar 0,3 ~ 7 Vdc – pinos 5 e 6

- Out1, Out2 – Saída de sinal PWM – pino 2 e 3
- Ptot – potência total dissipada de 25 *Watts* -
- Top – Temperatura de operação da junção -25 ~ 130ºC

Nos teste pratico com a velocidade máxima do motor a corrente consumida é de 1,6 Amperes, sendo necessário a montagem com dissipador de calor.

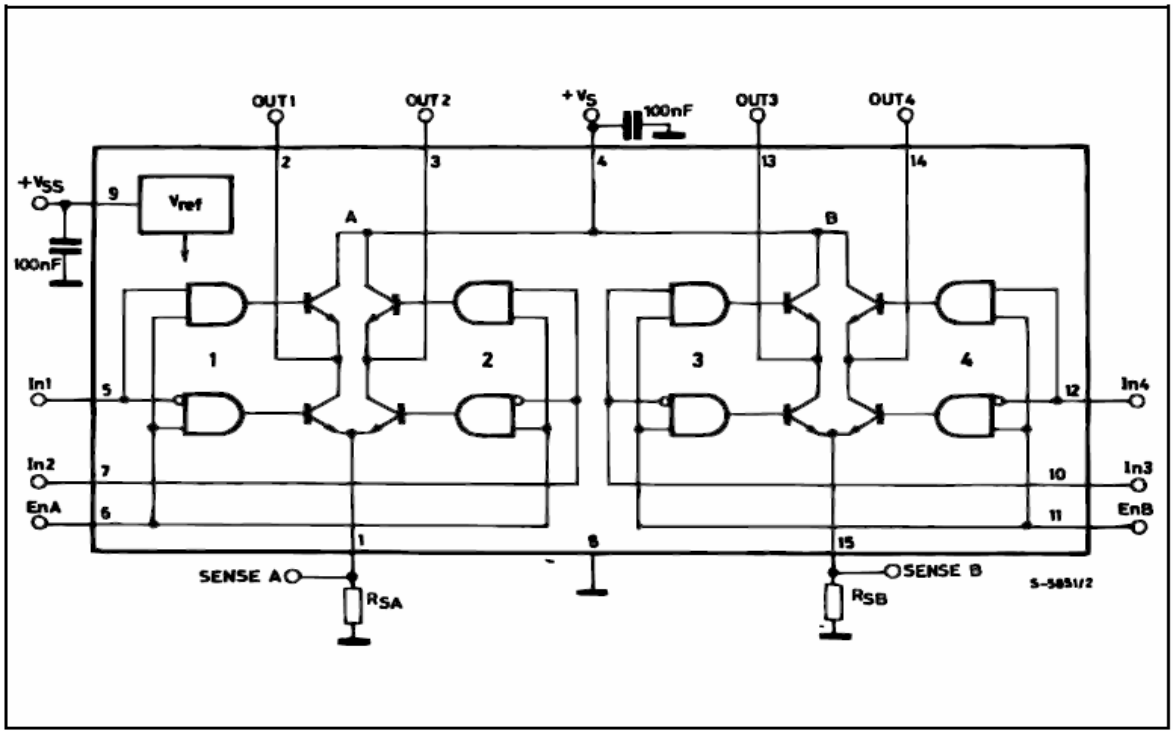

Figura 11 – Configuração interna do L298N

A segunda saída deste componente foi utilizado para acionamento do relé de 12 Vdc que faz a comutação do 220 Vac para a resistência elétrica no controle de temperatura *On-Off.*

4.5 COMUNICAÇÃO

Na transmissão o tamanho dos dados (intervalo entre cada *bit*) deve ser completamente padronizado em ambos os lados. Como essa comunicação trabalha com base em *bits*, essa velocidade é normalmente indicada em *bits* por segundo, ou bps. A velocidade de comunicação para esta aplicação está em 9600 bps ou seja uma transferência de 9600 dados por segundo.

TBIT= 1/Baud Rate

A maioria das mensagens digitais é mais longa que alguns poucos *bits*. Por não ser prático nem econômico transferir todos os *bits* de uma mensagem simultaneamente, a mensagem é quebrada em partes menores e transmitida seqüencialmente. A transmissão *bit*-serial converte a mensagem em um *bit* por vez através de um canal. Cada *bit* representa uma parte da mensagem. Os *bits* individuais são então rearranjados no destino para compor a mensagem original. Em geral, um canal irá passar apenas um *bit* por vez. A transmissão *bit*-serial é normalmente chamada de transmissão serial e foi o método de comunicação escolhido nesta aplicação.

Sempre é necessário um circuito de conversão de nível TTL/RS232. O circuito integrado mais comum para efetuar esta conversão é o MAX232 que possui alimentação TTL. O MAX 232 é um circuito integrado conversor de nível, que converte sinais TTL em RS232 e virse-versa – Figura 12. Ele fornece uma ótima rejeição de ruído e é mais robusto à descargas e curtos. Se o seu projeto for mais avançado, você deve utilizar um CI especializado para esta tarefa. No entanto, soluções especializadas são mais caras que as outras.

Figura 12 – Circuito integrado MAX 232

5 PROGRAMAÇÃO DO PIC

5.1 LÓGICA DO MICROCONTROLADOR

Um dos principais aspectos da programação é a otimização e eficiência do código gerado. Os compiladores possuem uma lista de comandos internos que não são diretamente traduzidos em código, cuja finalidade é de especificar determinados parâmetros no momento de compilar o código fonte e que são chamados de diretivas do compilador. Na figura 13, é mostrado a seqüência lógica do programa implementado no PIC.

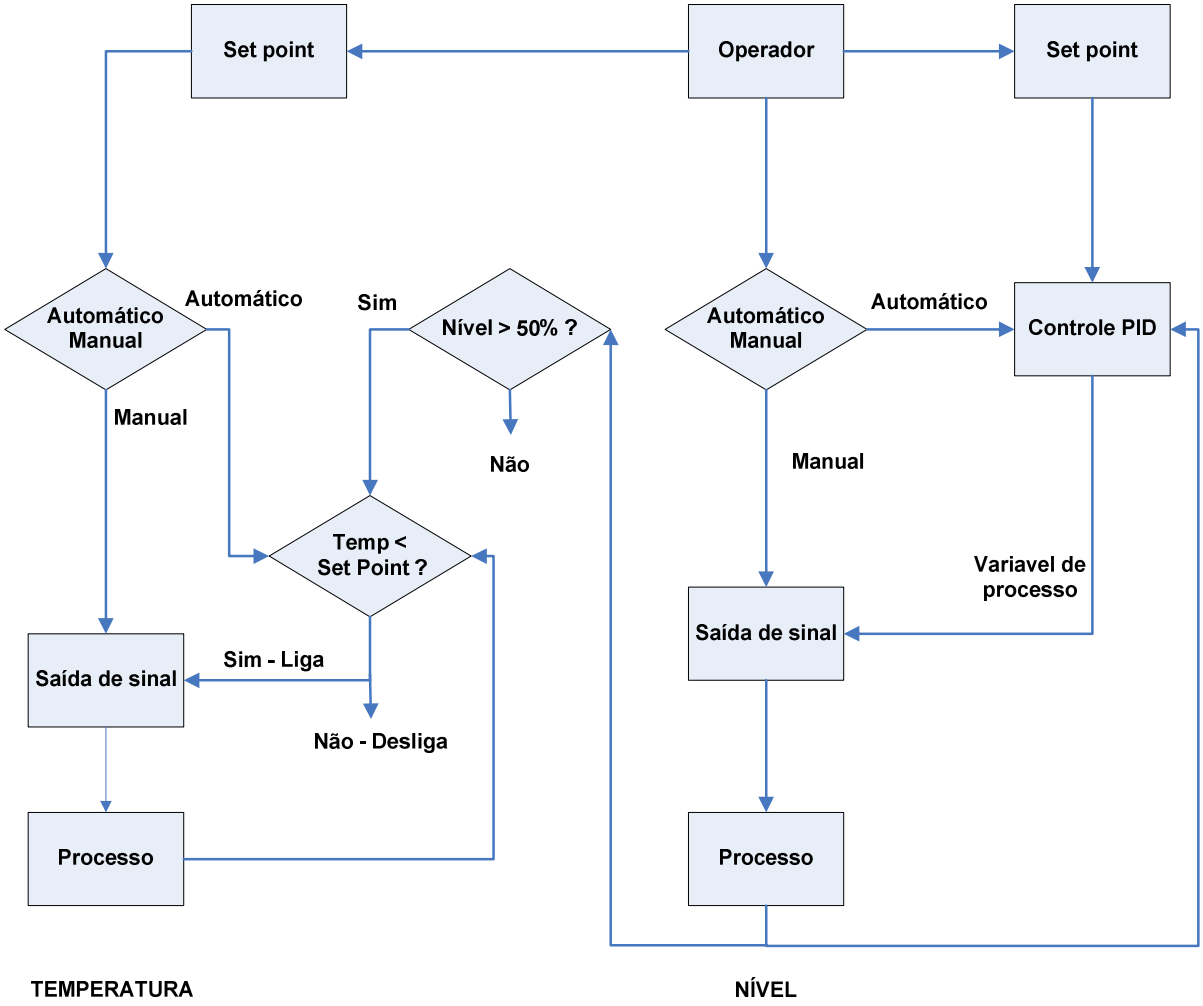

Figura 13 – Implementação Lógica no PIC

O compilador CCS (versão 4.057), possui uma grande quantidade de diretivas e que estão empregadas no cabeçalho do programa. A programação e a concepção

do controle lógico empregado neste trabalho e implementado no PIC 16F877A, seguem em varias etapas como: declaração das variáveis, configuração do CAD, implementação da lógica do controle PID e *On-Off*, tratamento de valores na EEPROM e comunicação serial. No apêndice A, encontra-se o arquivo fonte completo com demais comentários.

Para a realização dos primeiros testes de comunicação do PIC, foi de suma importância o uso dos programas *Hyperterminal* do *Windows* e a interface M2COMM, onde é possível verificar o protocolo de comunicação.

Foi empregado como nomes das variáveis de programação os termos mais utilizados no campo da instrumentação e automação, conforme os respectivos capítulos.

Cabeçalho

```
#include <16F877A.h>              // Define o microcontrolador a ser usado
#device ADC=10                    // Define o A/D para resolução de 10 bits
#use delay(clock=4000000)         // Define oscilador de 4MHZ
#fuses XT, NOWDT, PUT             // Define oscilador, não uso do Watdog time
#include <usart.c>                //  Diretiva de comunicação
#include <internal_eeprom.c>      // Diretiva da eeprom
```

5.1.1) Declaração das Variáveis

Todas as variáveis empregadas na lógica de controle foram implementadas procurando obedecer as nomenclaturas usadas em automação e instrumentação, facilitando assim a compreensão do programa conforme tabela 2.

Exemplos:

Variável	Descrição
sptemp, spnivel	*Set point* de temperatura, *Set point* do nível
pv, am,	Variável de processo, automático/manual
PID	Proporcional Integral Derivativo
mv, desvio	Variável manipulada, diferença entre SP - PV

Tabela 2 – Variáveis de Programação

5.1.2 Configuração do PWM

Sintaxe: SETUP_CCPX(modo) // modo é uma variável ou constante

setup_ccp1 (CCP_PWM) // Habilita saída PWM

setup_timer_2(T2_DIV_BY_4, 248, 1); //1 KHz – determina a freqüência do PWM

Sintaxe: SET_PWMX_DUTY( ) // Configura o ciclo ativo do modo CCP do PWM
set_pwm1_duty(0) // inicia o ciclo ativo em 0

//O sinal PWM inicia sempre em 0%, quando religado da energia elétrica.

5.1.3 Configuração da interrupção

```
enable_interrupts(GLOBAL);
enable_interrupts(int_timer0);      // Contagem de tempo
enable_interrupts(int_rda);         // Interrupção da comunicação
```

5.1.4 Configuração do CAD

Sintaxe: SETUP_ADC(0) // define o canal do CAD interno para o sinal de nível

```
set_ADC_channel(0);          // Habilita a primeira entrada analógica – PA0
delay_ms(5);
pvnivel = read_adc();        // recebe o valor do CAD referente ao nível
set_ADC_channel(1);          // Habilita a segunda entrada analógica - PA1
delay_ms(5);
pvtemp = read_adc();         // recebe o valor do CAD referente a temperatura
```

5.1.5 Lógica de controle PID

O calculo da correção atuando no elemento final de controle(motor).

//O erro atual é o resultado do valor do nível – o valor *Set point*

```
erro_atual  = pv - sp;
// Se o controle estiver em automático, a correção efetua a função calculaPID
if (am)
 {
  correcao = calculaPID(variacao, prop, integral);
   mv = mv + (correcao * acao);
  }
// Função CalculaPID
correcao = variacao * prop;      // correção da proporcional
soma_integral += (((1f * erro_atual * prop))/((60000f * 100) /
(integral * vel_scan)));         //correção da integral
  delay_ms(10);
```

5.1.6 Lógica de controle *On – Off:*

// Se a temperatura for maior que o *Set point* e o nível maior que 50% liga o contato S1 (resistência elétrica), senão ficará desligado

```
if ((pvtemp > sptemp) && pvnivel > 500) liga_s1;
   else desl_s1;
```

5.1.7 Manipulação da EEPROM

Em um controle de processo a continuidade operacional é de suma importância.

Como as variáveis em um PIC são armazenados na memória de acesso aleatório (RAM), a falta de alimentação elétrica pode ocasionar sérios problemas, fazendo com que os valores pré ajustados como ações de sintonia e parâmetros de controle se percam. Para manter a integridade do processo o PIC utiliza uma memória interna EEPROM que está sendo atualizada constantemente. Assim na falta de energia elétrica os últimos valores ficam armazenados, ou seja no restabelecimento da energia o controle volta a operação normal.

A EEPROM no entanto, muito mais lento do que RAM, e por isso não deve ser utilizada como uso geral. Pode levar vários milissegundos para escrever uma variável na EEPROM.

WRITE_EEPROM: Escreve um valor em um determinado endereço da memória interna.

Sintaxe: write_eeprom(13, valor) // escreva no endereço 13 o valor da variável.

```
prop = valor;
write_eeprom(15,prop);
integral = valor;
write_eeprom(16,integral);
```

5.1.8 Descrição do Protocolo

Sincronismo (modo assíncrono): a comunicação é feita em duas vias, como este modo não é sincronizado, essas duas vias são usadas para dados. Uma transmissão (TX) e uma para recepção (RX). Isto possibilita que as informações sejam enviadas e recebidas ao mesmo tempo (*Full Duplex*). O Sincronismo é feito pela velocidade de transmissão.

A comunicação entre microcontrolador e sistema supervisorio é o ponto critico para visualização *on-line* do funcionamento da planta. Existe somente um sincronismo de

tempo feito para a transmissão e recepção de cada *bit*. Esse sincronismo é alcançado através do *start bit* (@). Para a comunicação RS232 foi criado um protocolo de 8 *bits*, conforme a tabela 3 e complemento na tabela 4.

1º *Bit*	*Start bit*	@
2º *Bit*	Definição do sinal: Analógico ou Parâmetro	A ou P
3º e 4º *Bit*	Complemento 2º *Bit* – definição do sinal	XX
5º *Bit* ao 8º *Bit*	Valores	XXXX

Tabela 3 – Seqüência dos *bits*

Reconhecimento do protocolo

```
databuf[caracter] = rxreg;        // Recebe dados
caracter++;
if (databuf[0]!= '@')             //  Erro de transmissão, não recebeu start bit
{
 caracter = 0;                    //  Emite erro de comunicação
}
if (caracter == 8)                // Recebeu o start bit e somou 8 bit da transmissão ok
{
caracter = 0;
}
```

/*A função transmite envia dados pela comunicação serial. Antes, ela verifica se ainda existe algum dado para ser transmitido, e caso não tenha, um novo *byte* é colocado para a transmissão.*/

```
 usart_transmite("@A02");//pv controle 1
 converte(pvnivel);
 usart_transmite("@A12");//pv controle 2
 converte(pvtemp);
 usart_transmite("@P01");//velocidade do scan
 converte(vel_scan);
 usart_transmite("@P11");//auto/manual controle 1
```

Start Bit + Tipo de sinal	Aplicação	Indice	Range Microcontrolador
@A	saída do controle do nível	01	0 ~ 1023
@A	Indicação do nível	0 2	0 ~ 1023
@A	Indicação da temperatura	12	0 ~ 1023
@P	Velocidade do scan	01	0 ~ 250
@P	AM controle 1	11	0 ~ 1
@P	Ação controle 1	12	0 ~ 1
@P	SP controle 1	13	0 ~ 1023
@P	Prop controle 1	14	0 ~ 250
@P	Integral controle 1	15	0 ~ 250
@P	AM controle 2	21	0 ~ 1
@P	SP do controle 2	23	0~1023

Tabela 4 - Combinações do Protocolo

5.2 SISTEMA DE GERENCIAMENTO

A configuração do sistema de supervisão, foi desenvolvido em *Java*, tendo como interface gráfica a IDE *NetBeans* e banco de dados *PostgreSQL*. O *Java* terá a função de executar comunicação com o PIC, geração das interfaces gráficas e acesso através da porta serial das informações dos sensores.

5.2.1 *Javax.comm*

A API 3.0 é uma extensão padrão da plataforma *Java*, segundo [Sun,2008], que facilita o desenvolvimento independente de plataforma para comunicação, sendo aplicados em sistemas incorporados, dispositivos de ponto-de-venda, serviços financeiros, modems, exibição terminais, equipamentos e robótica. O *Java Communications* API (também conhecido como *javax.comm*) proporciona o acesso a aplicações RS-232 *hardware* (portas seriais) e de acesso limitado a IEEE-1284 (paralela portos).

Existem dois níveis importantes de classes na API de comunicação *Java*:
• classes de alto nível como *CommPortIdentifier* e *CommPort para* gerenciar o acesso e a apropriação de comunicação dos portos.
• Baixo nível de classes como *SerialPort ParallelPort* e proporcionar uma interface de comunicações físicas dos portos

```
// Parte do programa para abrir porta de comunicação
public void AbrirPorta() {
      Comunicacao comunicacao = new Comunicacao();
      comunicacao.start();

      Comando comando = new Comando();
      comando.start();
```

5.2.2 *Java* - Programação

Java é uma linguagem de programação desenvolvida pela *Sun Microsystems*. Considerada uma linguagem para criar programas seguros, robustos, orientados a objetos, *multithread* [Naughton,1996]. Realiza a façanha de independência de plataforma, compilando em uma representação intermediaria chamada *bytecode Java*, que pode ser interpretada em qualquer sistema que tenha um *runtime* de *Java* apropriado. A biblioteca de classes *Java* também fornece modelo unificado de protocolos da Internet, interface gráfica. Varias classes foram criadas conforme apêndice B

Programação aplicada ao projeto:

```
/ // Acesso ao Banco de Dados

public class Banco {

   private Connection connection;
   private String teste_endereco;
   private boolean conectado;

   public void Conetar_Banco(){

      String url = "jdbc:postgresql://localhost:5432/dados_db?charSet=LATIN1";
      try {
           Class.forName("org.postgresql.Driver");
           connection = DriverManager.getConnection(url,"usuario", "senha");
           conectado = true;
          }

// Insere Parametro para controle manual

public void setNivelManual() throws Exception{
      String query = "INSERT INTO comandos VALUES ('@P110000')";
      PreparedStatement ps = null;
      ps = banco.getConexao().prepareStatement(query);
      ps.executeUpdate();
      ps.close();
```

```
// Atualiza valores referente ao nível
public void Atualizar_Dado(DadosVO dado, char tipo, int indice)throws Exception{

    String query;
    PreparedStatement ps = null;
    switch(tipo){

       case 'A':

          switch(indice){

             case 1:
                query = "UPDATE dados SET MVNivel = ?";
                ps = banco.getConexao().prepareStatement(query);
             break;
```

5.2.3 *NetBeans* – versão 6.0.1

O *NetBeans* [Gonçalves,2006], é um ambiente de desenvolvimento, IDE, *open-source* escrito totalmente em *Java*. É uma ambiente que permite escrever, compilar e *debugar*. Possui um editor amigável de telas, conhecida como AWT *(Abstract Window Toolkit*) *que* fornece um conjunto de classes independentes de plataforma específicas para operações gráficas. Permite ainda integração com bancos de dados e diversos *plug-ins* de diversos tipos que estendem a capacidade de programa.

O *NetBeans* está implementado como a interface gráfica, onde será visualizado de forma *on-line* as variáveis de processo e os comandos necessários para operação da planta. Para um perfeito desenvolvimento e operação da planta, a parte gráfica também é de suma importância, pois é nesta tela que o operador terá as informações desejadas e posterior correção se necessário. As interfaces desenvolvidas estão na seção 5.3. Pode ser verificado que o propósito original de mostrar o fluxograma de processo não foi possível devido aos poucos recursos na parte de desenhos.

5.2.4 PostgreSql – versão 8.2.5-1

O *PostgreSQL* [*Sourceforge,*2008] é um SGBDO (Sistema Gerenciador de Banco de Dados Objeto-Relacional). São oferecidas muitas funcionalidades modernas, como: comandos complexos, chaves estrangeiras, gatilhos, integridade transacional controle de simultaneidade multiversão. Além disso, o *PostgreSQL* pode ser

estendido de muitas maneiras como, por exemplo, adicionando novos tipos de dado, funções, operadores, funções de agregação, métodos de índice, linguagens procedurais. O *PostgreSQL* terá a função de armazenar as variáveis de processo para acesso posterior pela conexão de Banco de Dados no *Java*

5.3 DESCRIÇÃO DA OPERAÇÃO DA PLANTA

Na figura 14 é possível uma visão da implantação elétrica na planta

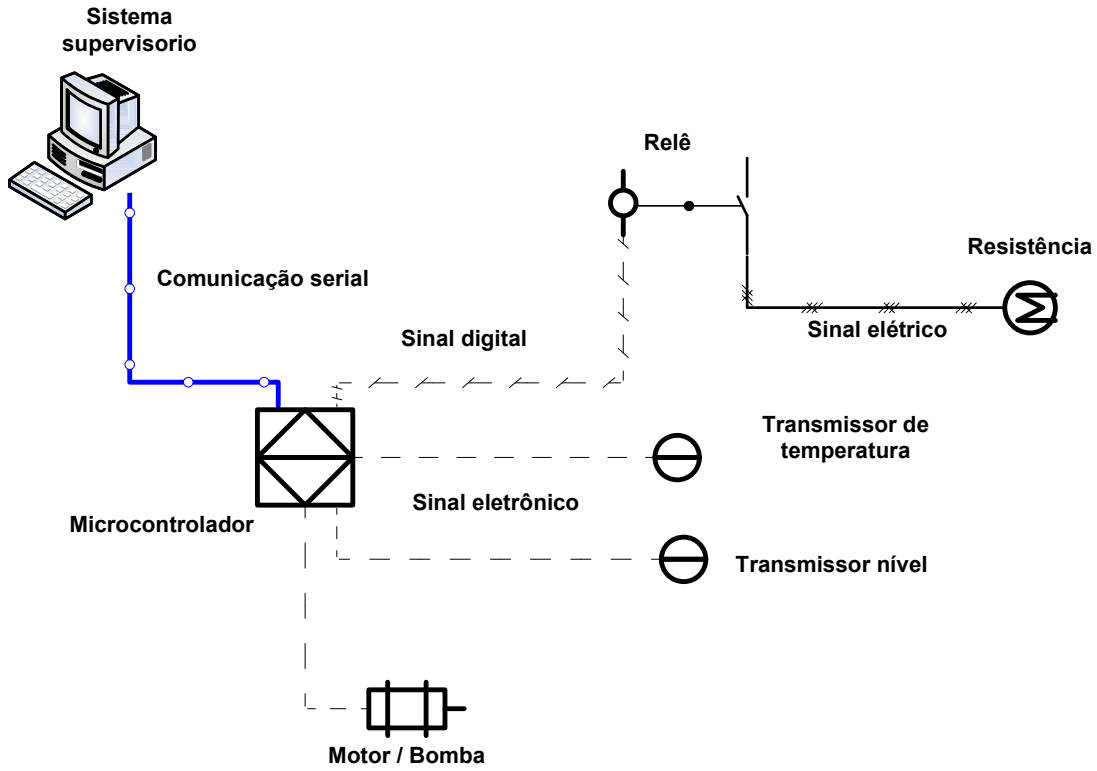

Figura 14 – Diagrama elétrico da planta

O objetivo da planta é manter estabilizadas as variáveis de controle como temperatura e nível através do controle automático implantado no sistema de supervisão. A planta consta de 02 reservatórios plásticos com capacidade de 20 litros cada. O tanque 2(parte inferior) irá abastecer o tanque 1 (parte superior) através da bomba elétrica. Ainda no tanque 1, foi instalado uma válvula manual para reposição de água no tanque 2, ficando assim um circuito de água. No tanque 01 está montado um sensor de temperatura (LM35) como elemento primário da medição de temperatura e um sensor de pressão para a medição do nível.

Com a malha de controle do nível em automático, o motor será ligado, variando a rotação para atingir o *Set point* do tanque 1. A válvula manual está aberta e a realimentação do nível se dá pelo transmissor LT-01. Quando o nível de água no tanque 1 estiver acima da posição física do aquecedor (50%) e o controle de temperatura em automático, a resistência será energizada a fim de transferir calor para a água, aumentando a temperatura até atingir o *Set point* da temperatura. Quando ultrapassar o valor do *Set point* a resistência é desligada.

Através da tela principal do sistema supervisorio – figura 15 o operador tem a possibilidade de passar o controle de nível em manual, para alterar a rotação da bomba ou ainda ligar/desligar o controle de temperatura. As figuras 16/17/18 são complementos como telas operacionais.

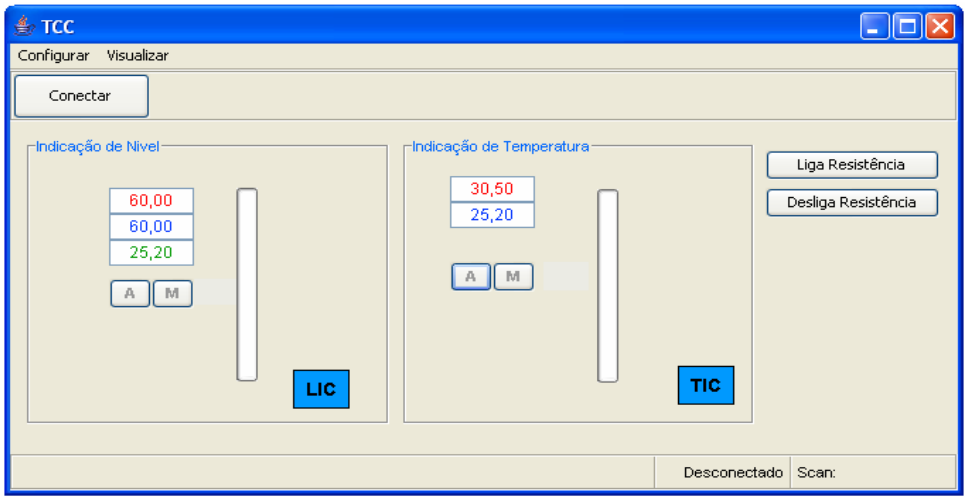

Figura 15 – Tela principal do sistema supervisorio

A figura 16, mostra a tela de ajuste dos alarmes de nível e das ações de controle.

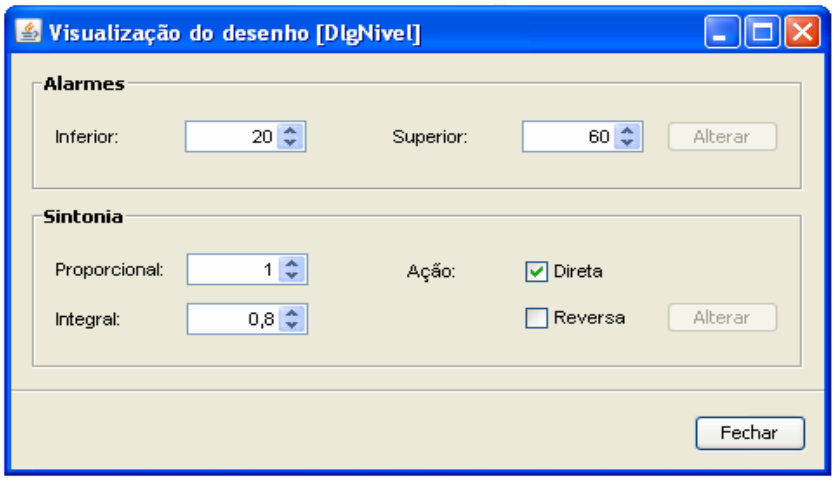

Figura 16 – Tela de ajustes de alarmes e ações de controle do nível

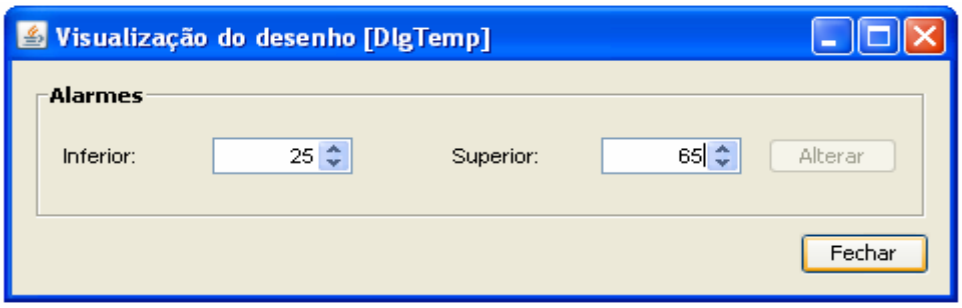

Figura 17 – Ajustes de ajuste dos alarmes do controle de temperatura

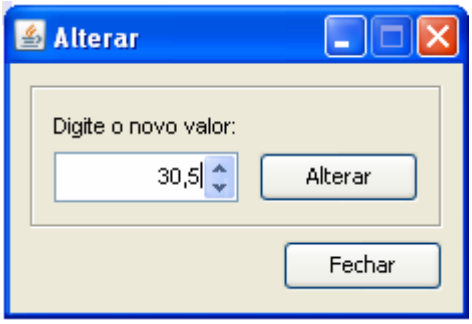

Figura 18 – Tela de ajustes do *Set point*

O gerenciamento consta com uma tela de tendência, onde serão disponibilizadas as informações de controle de forma *on-line* através de gráficos, conforme figura 17.

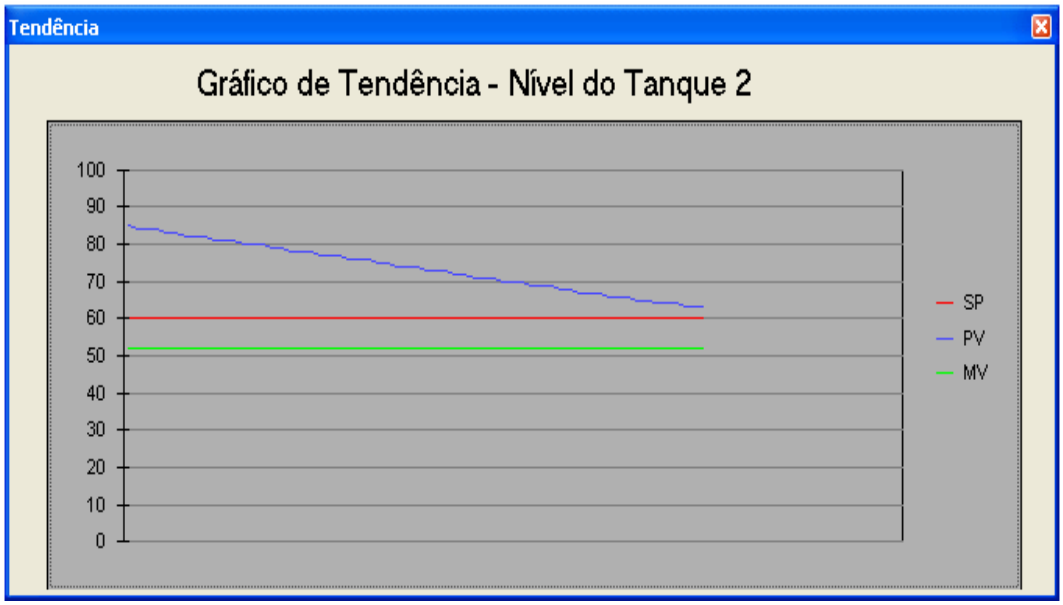

Figura 19 – Sugestão do Gráfico de tendência

5.3.1 Condições de Segurança

Para maior segurança dos equipamentos com o mesmo conceito de uma planta industrial, algumas considerações de segurança estão implantadas:

- A resistência só será ligado em automático se o nível atingir um valor pré determinado (50%) ; para evitar a queima por falta de água;
- Alarmes de nível e temperatura alta e baixo.
- Com a perda de comunicação serial, a planta continua em funcionamento

Na tabela 5, está designada a função de cada *Tag,* conforme instrumentação norma mundial da ISA (*Standart International of America*) conforme [Bega,2003].

Tag	Descrição
LT-01	Transmissor de nível
LIC-01	Controlador, indicador de nível
TIC-01	Controlador, indicador de temperatura

Tabela 5. *Tags* e sua respectiva descrição

6 CONCLUSÃO

Durante o desenvolvimento deste trabalho procurou-se em todos os aspectos se aproximar da realidade dos conceitos e aplicações da automação de processo industrial, desde os nomes das variáveis de programação, lógicas e tipos de controle empregados e as interfaces de trabalho para o usuário. Dentro deste propósito o sistema foi testado por um período de vinte e quatro horas ininterruptamente, onde foram executados vários testes de performace.

6.1 RESULTADOS ALCANÇADOS

Durante este período de testes não houveram falhas na lógica implementada, erros de comunicação ou na interface do supervisorio, demonstrando assim uma segurança operacional para o sistema. **Após a finalização dos testes práticos, conclui-se que o objetivo original deste trabalho foi alcançado**. A integração do microcontrolador com as linguagens de programação com o conceito de *software* livre apresentam uma alternativa confiável e custo baixo para uso de pequenas automações. Se a lógica desenvolvida precisar de um algoritmo rápido é recomendado o uso da linguagem *Assembly*, mas se a lógica tiver funções complexas e exigir menos tempo para desenvolve-las, a linguagem C é recomendada.

Não pode deixar de ser ressaltado o desenvolvimento acadêmico adquirido por função das pesquisas necessárias em um contexto geral: programação, comunicação serial e sobre o *hardware* empregado

6.2 TRABALHOS FUTUROS

Este trabalho pode ser continuado com a finalidade de integrar outras linguagens programação dentro dos conceitos iniciais.

- Uso dos comparadores e operação com *Display* de Cristal Liquido no microcontrolador;
- Uso de IDE, que permita criação de fluxogramas de processo, pois o *NetBeans* não possui esse recurso;
- Integração com o *JfreeChart* para geração de gráficos *on-line*;
- Substituição da comunicação serial por transmissor e receptor sem fio.

Bibliografia:

[1] **O que é Automação Industrial?.** Eurogam Automação Industrial, Estado do Paraná - disponível em < www.eurogam.com.br >
Acesso em: 20 maio 2008.

[2] **Controladores PID Microprocessados**. Ecil S/A Sistemas e Controle , Estado de S.Paulo – disponível em < www.help-temperatura.com.br/html / interesse / micro. HTML www.ecil >
Acesso em: 20 maio 2008.

[3] AMADEO, José Carlos. Instrumentação Industrial. **Mecatrônica Atual**, Nº 05, Agosto/2002, p.19-21. Editora Saber.

[4] MATIAS, Juliano. Teoria d Controle PID. **Mecatrônica Atual**, Nº 03, Abril/2002, p.18-25. Editora Saber.

[5] **Controle PID Básico**. Novus Produtos Eletrônicos Ltda, Estado de S.Paulo – disponível em < www.novus.com.br/downloads/Arquivos/artigopidbasico.pdf > Acessso em: 28 maio 2008.

[6] **Manual do controlador HW4200**. Coel Controles Elétricos Ltda, Estado S.Paulo – disponível em < www.coel.com.br/pdf/m_HW4200_r4.pdf >
Acesso em: 02 junho 2008

[7] **Sistema de Supervisão**. Biblioteca acadêmica da Fundação Educacional de Brusque, Estado de Santa Catarina- disponível em < www.unifebe.edu.br/ftp/ sistemas_de_informacao/cervi/Aula%2028-03.ppt#273,18, classificação de telas >
Acesso em: 04 junho 2008

[8] SOUZA, David José. **Desbravando o PIC**, 12º Edição. S.Paulo, Editora Érica Ltda, 2007.

[9] **MPLAB**. Disponibel em: <www1.microchip.com/downloads/en/DeviceDoc/39582b.pdf>
Acesso em: 18 maio 2008

[10] PEREIRA, Fabio. **Microcontroladores PIC Técnicas avançadas**, 2º edição, S.Paulo, Editora Érica, 2002.

[11] ZANCO, Wagner da Silva. **Microcontroladores uma abordagem prática e objetiva**, 1º edição. S.Paulo, Editora Érica, 2005.

[12] ZANCO, Wagner da Silva. **Microcontroladores uma abordagem prática e objetiva**, 1º edição. S.Paulo, Editora Érica, 2005.

[13] SOUZA, David José; LAVINIA, Nicolas Cesar. **Conectando o PIC 16F877A, recursos avançados**, 3º Edição. S.Paulo, Editora Érica, 2005.

[14] PEREIRA, Fabio. **Microcontroladores PIC - Programação em C**, 7º edição. S.Paulo, Editora Érica, 2007.

[15] **Software Free**. Disponível em: <www.fsf.org/licensing/essays/free-sw.html>
Acesso em 05 maio 2008.

[16] **DataSheet**. Disponível em: <www.national.com/mpf/LM/LM35.html>
Acesso em: 06 maio 2008.

[17] **DataSheet** . Disponível em:
<www.datasheetcatalog.org/datasheet/motorola/MPX5700GS.pdfdatasheet/motorola/MPX5700GS.pdf>.
Acesso em 01 maio 2008

[18] **Datasheet**. Disponivel em:
<www.datasheetcatalog.org/datasheet/texasinstruments/max232.pdf >
Acesso em 28 maio 2008.

[19] **Java Communications** , Disponível em:
<WWW.java.sun.com/products/javacomm/índex.jsp>
Acesso em 28 maio 2008.

[20] BEGA, Egidio Alberto; DELMÉE, Gerard Jean; COHN, Pedro Estefano; BULGARELLI, Roberval; KOCH, Ricardo; FINKEL, Vitor Schimidt. **Instrumentação Industrial**,1º Edição. Estado R. Janeiro, Editora Interciência, 2003.

[21] GONÇALVES, Edson. **Dominando NetBeans**, 1º Edição, Estado R.Janeiro. Editora Ciência Moderna, 2006.

[22] NAUGHTON, Patrick. **Dominando o Java**, 1º Edição, Estado S.Paulo. Editora Makron Books, 1996.

[23] **PostegrSQL Documentação**. Disponível em < www.sourceforge. net/forum/fórum .php? forum_id=680508 >. Acesso em 05 maio 2008