CAPÍTULO I

INTRODUÇÃO

As inovações constantes na tecnologia de informação, juntamente com o desenvolvimento das redes de comunicação de dados, permitiram o aprimoramento de novos recursos educacionais como a Educação à Distância (EAD). Apesar de existirem metodologias pedagógicas abordando a transmissão e técnicas de ensino de conteúdos teóricos via www (Internet), poucos recursos foram desenvolvidos em relação às aulas práticas tais como experimentos de laboratório, particularmente na área de sistemas de controle. Neste sentido a elaboração de sistemas que possam contemplar práticas de laboratório a distância são bem vindas.

Este trabalho tem como proposta a implementação de um sistema simples e eficiente que permita a realização de experiências remotas via Internet, de aulas práticas relativas a malhas de controle de processos contínuos. Os usuários poderão realizar remotamente experiências reais em controle de processos utilizando plantas em escala reduzida (ou bancadas didáticas), tais como sistemas de nível, vazão, pressão, temperatura, posição, etc.

A relevância desta proposta se deve a alguns fatores descritos a seguir. A aquisição de determinadas quantidades de bancadas didáticas para laboratórios em disciplinas de sistemas de controle em universidades, escolas técnicas, etc., envolve custos geralmente superiores aos orçamentos disponíveis em muitas instituições de ensino. Isto restringe a aquisição da quantidade e da variedade dos equipamentos necessários para realizações de experiências práticas nas disciplinas correlacionadas. Restrições na quantidade de equipamentos podem gerar um número elevado e inadequado de alunos por bancadas nas salas de aulas práticas, ou podem levar a um número grande de turmas alocadas a uma sala específica, dificultando desta maneira o seu gerenciamento.

Essas questões podem ser contornadas através da utilização da Internet na realização de experiências reais executadas à distância. Através de microcomputadores ligados em rede, os usuários podem realizar experiências utilizando um mínimo de bancadas de laboratório. Além da utilização de microcomputadores da própria sala do laboratório em questão, é fácil empregar computadores de outras salas da instituição ou até de computadores externos a mesma, desde que ligados a uma rede de comunicação de dados.

A definição dos parâmetros que definem as características de resposta de um dado sistema de controle e a visualização das grandezas físicas controladas é facilmente configurada através de telas de *interface* com o usuário. É simples também incluir imagens de câmeras digitais para visualização real dos processos controlados.

Esta abordagem pode levar a uma série de benefícios tais como redução de custos, otimização na utilização de equipamentos e salas de aula, possibilidades de ensino à distância, educação continuada e outras facilidades no suporte a disciplinas que necessitem de aulas práticas em laboratórios, em particular em sistemas de controle.

Questões de disponibilidade de utilização podem ser resolvidas através de acesso seqüencial ou por agendamento prévio. Um procedimento automatizado para esta questão não será o foco deste trabalho. Atualmente é utilizada uma lista de controle com filas de espera e uma pessoa é responsável por gerenciar e liberar o acesso, permitindo a realização da experiência no devido tempo programado.

Este trabalho está dividido em outros seis capítulos. O Capítulo 2 faz uma revisão da bibliografia existente na área. No Capítulo 3, será apresentada uma revisão conceitual sobre malhas de controle de processos contínuos. O Capítulo 4 contém a formulação da proposta a ser desenvolvida nesta dissertação. O Capítulo 5 apresenta os desenvolvimentos realizados na implementação do sistema proposto. Os resultados obtidos estão descritos no Capítulo 6. E finalmente, no Capítulo 7, estão as conclusões da dissertação e propostas para trabalhos futuros.

CAPÍTULO II

RESENHA BIBLIOGRÁFICA

A expansão das redes de comunicação de dados está permitindo que a troca de informações entre pessoas, empresas, instituições, etc. ocorra de maneira cada vez mais rápida, simples e com custos reduzidos. A Internet vem alterando a maneira como as pessoas se comunicam e interagem. Nesse contexto onde diversas aplicações são vislumbradas, pode-se citar a educação à distância. O desenvolvimento das tecnologias de informações tem incrementado significativamente os meios de transmissão, divulgação e aprendizagem de várias áreas do saber humano. Com a expansão da Internet novas modalidades de ensino à distância passam a ser viáveis e começam a delinear novos contornos e práticas.

Nos últimos anos tem ocorrido um aumento nas publicações de trabalhos e artigos sobre educação à distância em diversos campos. As referências bibliográficas de interesse para esta dissertação estão focadas em publicações que utilizam recursos de redes de comunicação de dados aplicadas no ensino de engenharia em áreas que abordem sistemas de controle, instrumentação, automação, etc.

Novaes (1994) abordou o ensino de engenharia em geral através da utilização de redes de computadores apresentando um sistema interativo considerando o que facilitaria o sucesso em integrar a sala de aula tradicional com os recursos multimídia possibilitados pelas inovações das tecnologias de informações.

Arpaia et al. (1997) apresentaram um laboratório virtual de instrumentação eletrônica utilizando programas residentes em um servidor WEB agindo como uma interface gráfica entre a janela do cliente e o software local onde funcionava a experiência.

Bhandari e Shor (1998) desenvolveram um software de aprendizagem a distância que permitia o usuário realizar experiências remotamente no laboratório da Universidade de

Engenharia de OREGON. Este software permitia o acesso através da WEB com um browser básico instalado e a implementação do controle de um robô via Internet.

Salzmann et al. (1999) mostraram um laboratório virtual para realização de experimentos na área de educação em engenharia. Propuseram o uso eficiente da largura de banda do sistema com uma qualidade aceitável de serviço para a experimentação remota com a Internet.

Poindexter e Heck (1999) apresentaram um tutorial sobre como criar *sites* especializados para educação através da Internet, mostrando alguns modelos, exemplos e aplicações específicas de laboratórios remotos.

Um laboratório experimental via Internet, utilizando o sistema operacional Linux foi relatado em Jochheim e Rohrig (1999). A idéia principal deste laboratório remoto foi utilizar a Word Wide Web como plataforma de comunicação e o Web Browser como sua interface, onde a Internet provê a plataforma para transmissão de informações, e o Web Browser é o próprio ambiente para rodar o software cliente.

Dixon et al. (2001) utilizam hardwares e softwares comerciais para implementar um sistema de controle para laboratórios educacionais via Internet onde é necessário a uniformidade e a padronização dos equipamentos . A solução do laboratório de controle foi baseado no MatLab com *Toolbox* de tempo real sendo executado em ambiente Linux.

Rodrigo (2002) apresentou um laboratório remoto de eletrônica de potência pela Internet que possibilita o controle do acionamento de motores elétricos através de conversores estáticos de energia, onde utilizou um servidor com o sistema Real Time Linux realizando o controle dosistema.

Silva et al. (2002) mostraram como mídias e tecnologias eletrônicas podem ser utilizadas no aprendizado de sistemas de controle, apresentaram um equipamento real, no caso, um servo mecanismo.

Nitin et al. (2002) empregaram os softwares NetMeeting e Simulink para realizar experimentos remotos em um sistema de pêndulo invertido. O experimento era gerado em um computador e carregado em outro utilizando o protocolo FTP para realizar funções remotas.

Henry (2002) apresentou um sistema para experiências de controle utilizando uma rede de computadores Macintosh executando o software LabView. Os dados resultantes eram analisados através do Excel por meio de arquivos transferidos via FTP.

Um sistema de tanque duplo controlado pela Internet foi realizado por Ramakrishnan et al. (2002) utilizando-se de protocolo TCP, controladores PID e Lógica Fuzzy. Uma câmera e o software NetMeeting foram utilizados para realizar sessões de vídeo conferência das experiências efetuadas.

Snachés et al. (2004) utilizaram linguagem Java, o padrão XML e o software Simulink para controlar remotamente via *Web* um sistema de pêndulo invertido, utilizando o ambiente novell.

Nessa dissertação não será considerado o uso de sistemas industriais como controladores lógico programáveis (CLPs), controladores de malhas (Single-Loops, Multi-Loops), sistemas digitais de controle distribuídos (SDCDs), instrumentações inteligentes (Field-Bus), etc. Estes sistemas já incorporam além de funções específicas de controle, outras que permitem a comunicação de dados com redes industriais e até mesmo com redes comerciais como a Internet. Também possibilitam integrar e monitorar eficientemente informações através de softwares de supervisão, executados em microcomputadores pessoais, ligados à rede de comunicação. Atualmente estes sistemas permitem o controle de processos ou plantas complexas à distância, mas os custos resultantes dos sistemas de hardware e software necessários tendem a ser elevados. Assim, a utilização de hardwares e softwares comerciais geralmente conduz a soluções com custos menores.

Na maioria das referências bibliográficas consultadas os equipamentos utilizados (controladores, sistemas de aquisição de dados, câmeras digitais, plantas, processos, etc.) são sofisticados e apresentam custos que podem ser considerados altos para a maioria das instituições educacionais brasileiras. Em muitos trabalhos os processos são dedicados (alguns

de fabricação própria), e consequentemente os softwares desenvolvidos tendem a ser também específicos.

Neste contexto foi possível constatar que faltam alternativas genéricas e de custos reduzidos para sistemas de experiências remotas em malhas de controle de processos contínuos, visando o ensino à distância.

CAPÍTULO III

FUNDAMENTOS SOBRE MALHAS DE CONTROLE

As malhas de controle de processos contínuos são amplamente utilizadas em vários sistemas industriais. Como exemplo pode-se citar: controle de tensão e freqüência em geração de energia elétrica; controle de temperatura em caldeiras a vapor ou fornos; controle de nível em reservatórios; controle de rotação de motores elétricos; controle automático de navegação em veículos, aviões, navios; etc. Exemplos de equipamentos utilizados em controles automáticos de processos contínuos são os controladores de malha. Estes controladores, dependendo do número de malhas de controle implementadas e de outras funções de processamento, podem ser classificados em *Single-Loop*, *Multi-Loops*, SDCD (Sistemas Digitais de Controle Distribuído), Controladores Lógicos Programáveis (CLP), Computadores de Processo, etc.

A necessidade de controle automático ou de malhas de controle contínuas não está relacionada apenas com o interesse da automação em si, mas sim ligada às questões mais fundamentais, como estabilizar sistemas instáveis, rejeitar distúrbios, perturbações ou variações em um processo, planta ou então modificar a dinâmica de um sistema. Uma das idéias fundamentais neste contexto é o conceito de realimentação (*feedback*). Basicamente um sinal de informação relacionado com a grandeza a ser controlada, como a de uma temperatura, pressão, velocidade, tensão, fluxo etc., é somado (subtraído) a um valor de referência desejado para a mesma. O erro resultante dessa operação é processado (atenuado/amplificado, filtrado, etc.) objetivando excitar a entrada do processo sob controle. Em malhas de controle contínuas utilizam-se realimentações negativas com a finalidade de estabilizar sistemas, rejeitar distúrbios e/ou modificar dinâmicas, daí o sinal de realimentação ser subtraído da referência desejada. A Figura 3.1 ilustra um diagrama de blocos de um sistema de controle contínuo típico. O bloco *G* representa o processo (planta) a ser controlado, sendo *u* a entrada de excitação do mesmo, *y* é a grandeza a ser controlada, e *d* eventuais distúrbios do sistema. O bloco *H* representa a transdução e a instrumentação relacionada com a grandeza controlada, e

algumas vezes incorporam também um compensador. A entrada *r* define o valor desejado de *y*, sendo *e* o erro (subtração) entre essas duas variáveis, *F* é um fator de escala e/ou um filtro. O bloco *C* simboliza o controlador/compensador necessário para impor determinadas características na malha de controle como estabilidade, tempo de acomodação, ultrapassagem máxima, etc.

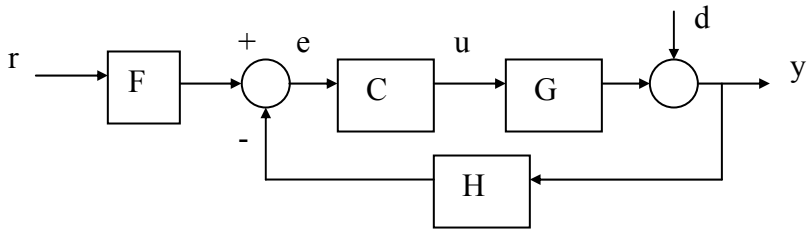

Figura 3.1 - Elementos constituintes de um sistema de controle contínuo.

Visando fornecer um exemplo de um processo, o desenho da Figura 3.2 ilustra um reservatório onde é necessário manter um nível específico de um determinado líquido. O sistema possui uma válvula que controla o fluxo de entrada (q_i) do líquido no reservatório. Através de uma tubulação de escape o fluxo de saída (q_o) escoa do sistema. Geralmente este fluxo não é controlado e pode variar de maneira casual representando um distúrbio ou uma perturbação no sistema. A diferença entre os fluxos de entrada e o de saída vai determinar o nível (h) no tanque, definindo se a coluna de líquido vai aumentar, diminuir ou permanecer constante.

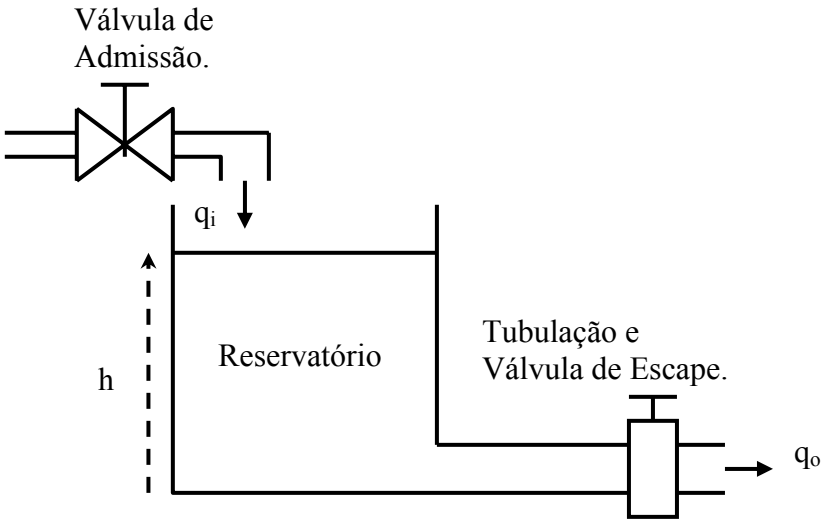

Figura 3.2 - Sistema de nível típico.

A medição de nível pode ser realizada através de vários tipos de transdutores, entre eles uma bóia cuja haste movimenta o cursor de um potenciômetro linear, onde conforme a

variação do nível do líquido ocasiona-se uma variação da resistência do mesmo. Com os extremos do potenciômetro conectado a uma fonte de tensão (V), o seu cursor vai apresentar uma voltagem (v_h) proporcional ao nível (h = y) do reservatório, como ilustrado na Figura 3.3 e modelado pelas expressões em (3.1). Geralmente a válvula que controla o fluxo de entrada de líquido no reservatório é acionada eletricamente através de um sinal de comando (*u*), e apresenta uma ação linearmente proporcional conforme indicado pela equação (3.2).

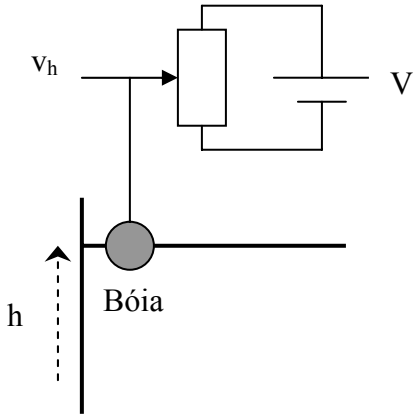

Figura 3.3 - Exemplo de medição de nível.

Onde, matematicamente, tem-se:

$$v_h = K_1.h \;\; ; \quad y = h = K_2.v_h; \qquad (3.1)$$

$$q_i = K_3.u. \qquad (3.2)$$

É possível obter um modelo matemático aproximado que represente a planta de nível (Ogata, 2003). As variáveis ou grandezas físicas mais freqüentes para descrever o comportamento de sistemas hidráulicos são: nível; volume; fluxo; pressão. Os elementos básicos mais comuns destes sistemas são a resistência e a capacidade hidráulica. As relações que expressam os fenômenos básicos envolvidos em tais sistemas são representadas através de algumas equações bem conhecidas em hidráulica e estão descritas a seguir.

- Resistência hidráulica *R*: Relaciona a vazão *q* de um fluído em uma tubulação de um reservatório com um dado desnível *h*, ou seja: *R=dh/dq*. Se o escoamento for laminar (vazão não turbulenta) a relação é linear, ou seja: *q(t)=h(t)/R*. Analogamente a sistemas elétricos, este parâmetro indica a oposição (resistência) a passagem de um fluído (corrente).

- Capacidade hidráulica *C*: Relaciona a variação da vazão Δq de um fluído pela taxa de variação do seu nível *h* no tempo *t* em um reservatório: $C=\Delta q(t)/(dh(t)/dt)$. Este parâmetro pode ser interpretado como sendo a variação no volume de um líquido armazenado em tanque, necessária para causar uma variação unitária em seu nível. A variação de vazão pode ser convencionada como $\Delta q(t) = q_i(t) - q_o(t)$ e usando as definições de resistência e capacidade hidráulica pode-se escrever a equação diferencial:

$$C\frac{dh(t)}{dt} = q_i(t) - q_o(t) = q_i(t) - \frac{h(t)}{R} \qquad (3.3)$$

Usando a transformada de Laplace tem-se:

$$sRCh(s) + h(s) = Rq_i(s) \quad (3.4)$$

A relação $h(s)/q_i(s)$ é a chamada função de transferência (3.5) do sistema, e relaciona a variação de nível do reservatório em função da vazão de entrada. Associando esta função às expressões (3.1), (3.2) e atribuindo valores de *R*, *C*, K_1, K_2, K_3 pode-se obter os valores numéricos correspondentes às funções de modelagem y(.)/u(.) ou h(.)/q_i(.). Os modelos e parâmetros associados a estas funções também podem ser obtidos por técnicas de identificação de sistemas (Aguirre, 2000).

$$\mathrm{g(s)} = \frac{h(s)}{q_i(s)} = \frac{R}{RCs+1} \qquad (3.5)$$

Tomando o exemplo do sistema de nível, a idéia básica de uma malha de controle consiste em usar o valor do erro do sistema, segundo as expressões em (3.6), objetivando ajustar o fluxo de entrada do fluído no reservatório para manter a coluna do líquido em um valor constante. Preferencialmente o mais próximo possível daquele definido pela entrada de referência também conhecida como *set-point*. A função do controlador será responsável por impor determinadas características de resposta. Em malhas de controle contínuo os tipos de entradas de referência mais empregadas são degrau e rampa. Uma entrada tipo degrau é aquela que quando o sistema é ligado mantém sempre um valor constante. Já uma entrada tipo rampa tem seu valor inicial baixo ou nulo e vai crescendo até atingir um valor nominal pré-estabelecido.

$$e = F.r - H.y \;\; ; \quad u = C.e. \qquad (3.6)$$

Em sistemas de controle de processos são desejadas algumas características de respostas dinâmicas pré-estabelecidas. Estas características usualmente estão relacionadas a entradas de referência do tipo degrau. A Figura 3.4 ilustra um gráfico de resposta típica, onde a variável controlada (y) do sistema pode oscilar de forma amortecida até atingir um valor estável. É desejável que este valor seja o mais próximo possível do dado da referência de entrada (r), ou seja, que o erro da malha tenda a um valor dentro de uma determinada faixa especificada ou que seja nulo. O tempo que o sistema leva para atingir esta faixa de erro e permanecer dentro da mesma é conhecido como tempo de acomodação (T_a). A resposta ainda pode apresentar um máximo pico (M_P) igual ou inferior a um valor pré-determinado. Eventualmente este valor pode não ultrapassar o valor da referência e nesta condição a resposta é chamada de superamortecida. Uma malha de controle não deve apresentar reposta oscilatória (permanente ou crescente) que ultrapasse o valor da faixa de erro especificada, pois isto caracteriza um sistema de controle instável.

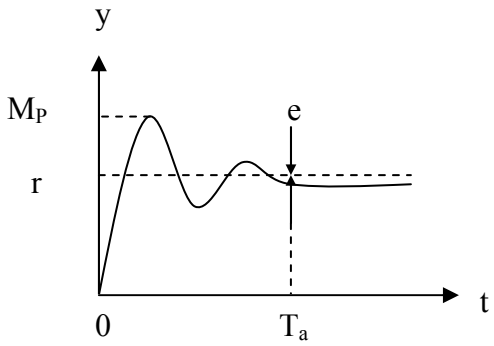

Figura 3.4 - Resposta típica de uma malha de controle para entrada degrau.

As finalidades dos controladores ou compensadores nas malhas de controle são de estabilizar sistemas instáveis, alterar características de respostas dinâmicas e atenuar eventuais distúrbios ou alterações de parâmetros que um processo, planta ou sistema físico possa apresentar. Os controladores ou compensadores mais usuais são os PID (proporcional-integral-derivativo). A função dos compensadores é processar (amplificar/atenuar, filtrar, etc.) o sinal do erro (e) de uma malha de controle, fornecendo um sinal (u) que sirva para efetuar as correções necessárias no sistema. A equação (3.5) representa a função básica de um controlador PID contínuo no tempo, e a expressão (3.6) em tempo discreto. Os parâmetros *Kp*, *Ki* e *Kd* são conhecidos como ganho proporcional, integral e derivativo respectivamente.

Nas implementações em tempo discreto o valor de *T* define a taxa de amostragem ou de varredura da malha de controle. Se esse valor for bem menor (por exemplo, dez vezes) que a menor constante de tempo do processo (ou planta) a ser controlado, o comportamento de um controlador digital é bem semelhante ao de um compensador analógico.

$$u(t) = Kp.e(t) + Ki\int e(t).dt + Kd\,\frac{de(t)}{dt} \qquad (3.5)$$

$$u(k) = Kp.e(k) + Ki\sum e(k).T + Kd\,\frac{\Delta e(k)}{T} \qquad (3.6)$$

Uma outra forma de representar a equação ou modelo de um controlador é no domínio da freqüência. Tomando como base a equação (3.5) pode-se escrever a expressão (3.7), onde o operador *s* é conhecido como operador de Laplace (o nome domínio da freqüência é devido a relação $s = jW$, onde $W = 2\pi f$). A relação entre o sinal de saída (u) do controlador e a entrada (e) é a função de transferência do controlador.

$$\frac{u(s)}{e(s)} = Kp + \frac{Ki}{s} + sKd \qquad (3.7)$$

Nos controladores analógicos são utilizados resistores, capacitores, e amplificadores operacionais para implementar as funções de compensação. Nos controladores digitais (Single-Loop; CLP, SDCD, etc.) a estrutura de *hardware* típica é a exemplificada na Figura 3.5. Uma unidade processadora realiza as funções de subtração, compensação e outras necessárias a implementações de malhas de controle. Circuitos conversores analógicos-digitais (A/D) e digitais-analógicos (D/A) realizam a *interface* do processador com a instrumentação do processo controlado. Em registros ou em posições de memórias da unidade processadora são armazenados os parâmetros ou ganhos das funções de compensação, cujo processamento está incluído no software que acompanha os equipamentos de controle.

Figura 3.5 - Estrutura dos controladores digitais de malhas de controle.

Sintonizar uma malha de controle significa ajustar os parâmetros de controladores/compensadores de maneira que a mesma seja estável e que sejam estabelecidas determinadas especificações no domínio do tempo ou da freqüência. Estas especificações determinam as respostas dinâmicas desejadas e outras solicitações de controle necessárias. Existem diversas alternativas ou procedimentos para sintonizar malhas de controle. A mais simples, mas que demanda maiores esforços, consiste em alterar por tentativa e erro os parâmetros de controladores/compensadores até que as especificações sejam atingidas. Uma alternativa consiste nas regras empíricas de Ziegler-Nichols (Ogata, 2003). Estas regras são utilizadas em algumas situações práticas, mas exigem ajustes posteriores, devido a não apresentarem precisão nas estimativas dos ganhos de sintonia.

Existem outros métodos gráficos e/ou analíticos mais precisos na estimação dos ganhos dos controladores. Geralmente estes métodos necessitam de alguns dados do processo a ser controlado. Essas informações podem ser apresentadas na forma de gráficos ou funções. Para um sistema de nível como visto anteriormente, um exemplo de função de transferência seria o modelo dado através da função (3.3). Com as informações do modelo do processo, é possível utilizar expressões como as (3.8) conhecidas da teoria de controle (Phillips e Harbor, 1996), para calcular os ganhos de um controlador PID. As especificações consistem na freqüência de cruzamento de ganho (Wcg) e na margem de fase (MF). Geralmente quanto maior a margem de fase, menor será o máximo pico da resposta temporal da grandeza controlada. E quanto maior a freqüência de cruzamento de ganho, menor será o tempo de acomodação da resposta dinâmica do sistema.

$$\theta = -180^{o} + MF - \angle G(jWcg)H(jWcg) \Rightarrow \begin{cases} Kp = \dfrac{\cos\theta}{\left|G(jWcg)H(jWcg)\right|} \\ KdWcg - \dfrac{Ki}{Wcg} = \dfrac{\sin\theta}{\left|G(jWcg)H(jWcg)\right|} \end{cases} \qquad (3.8)$$

Como exemplo seja a função G(s) = y(s)/u(s) = 22/(160s + 1). Admitindo que as especificações desejadas sejam margem de fase (MF) próxima de 85^0 e freqüência de cruzamento de ganho (Wcg) em torno de 1 [rd/s]. Usando uma calculadora que opera com números complexos vem G(1j) = 0,0009-0,1375j ou /G(1j)/≈ 0,1375 e /_G1j)≈ -90.

Empregando as expressões de (3.8) vem: $\theta = -5^0$; Kp = 7,23; Ki = 0,63 (para Kd = 0). A Figura 3.6 ilustra a resposta em freqüência da malha de controle compensada. A Figura 3.7 mostra a resposta temporal da malha de controle para uma entrada de referência em degrau unitário. A dinâmica estabelecida para o sistema está adequada, apresentando um máximo pico menor que 5% e um tempo de acomodação em torno de 30 [seg.]. A obtenção das curvas de módulo e fase pela freqüência pode ser obtida matematicamente pela substituição s = jW. As funções resultantes G(jW) ou C(jW)G(jW) passam a ser números complexos, de onde é possível obter os valores dos módulos e fases em função da freqüência (W), e montar os gráficos resultantes. Existem *softwares* que realizam estas operações e gráficos de maneira bem simples e direta. Entre eles pode-se citar o MatLab/Simulink ([21]-[23]), cujos os comando listados a seguir podem ser utilizados para a obtenção das figuras citadas, facilitando a análise e a sintonia de malhas de controle em geral.

```
Ng = 22; Dg = [160  1]; <Enter>
Nc = [7.23   0.63]; Dc = [1  0]; <Enter>
[No,Do] = series(Nc,Dc,Ng,Dg); <Enter>
bode(No,Do) <Enter>
[Nm,Dm] = cloop(No,Do); <Enter>
step(Nm,Dm) <Enter>
```

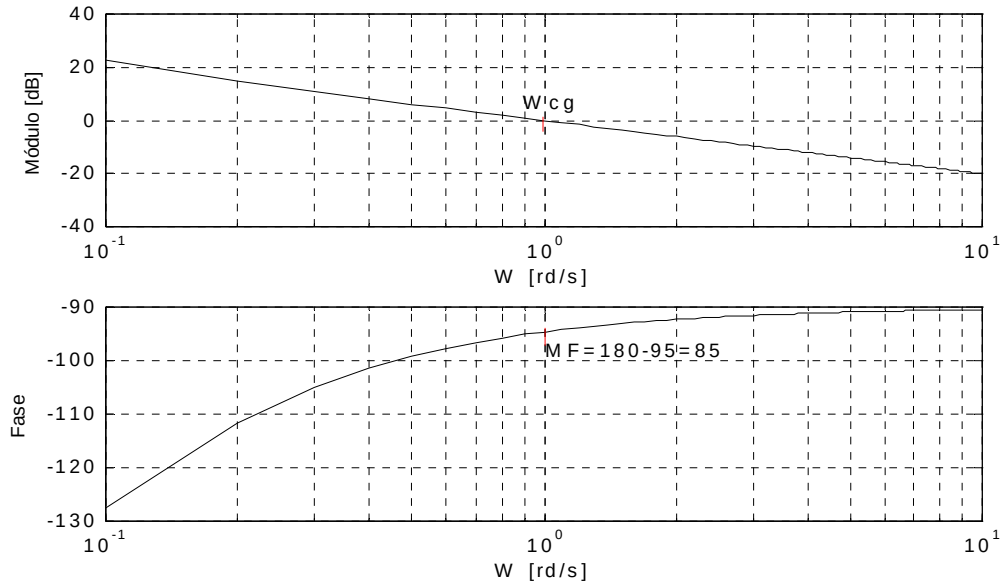

Figura 3.6 - Reposta em freqüência C(jW)G(jW) do sistema compensado.

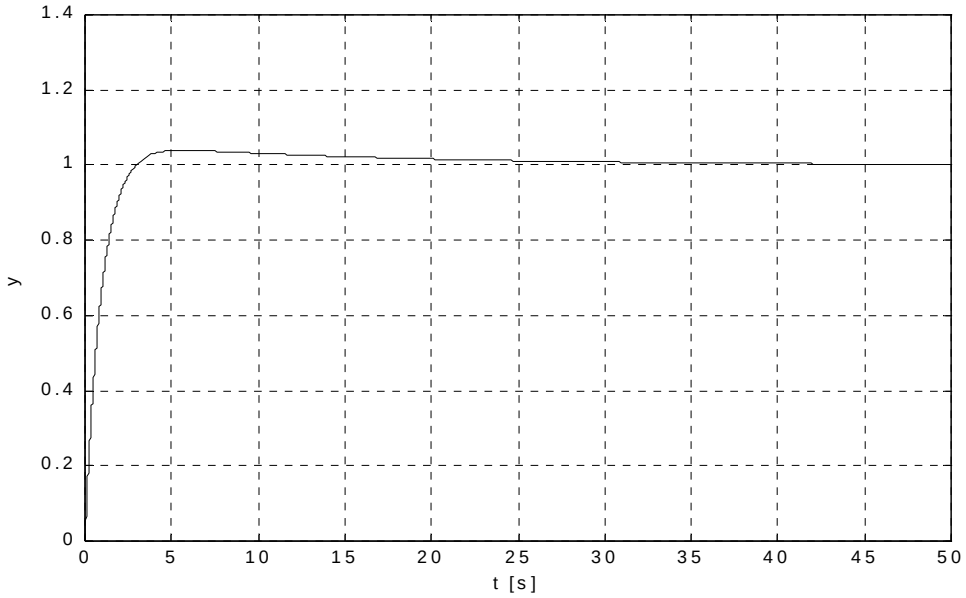

Figura 3.7 - Reposta temporal da malha de controle.

CAPÍTULO IV

PROPOSTA DE UM SISTEMA VIA INTERNET PARA EXPERIÊNCAS DE LABORATÓRIO DE CONTROLE DE PROCESSOS CONTÍNUOS

A Universidade Federal de Itajubá possui um laboratório de controle de processos industriais contínuos (LCPIC) que contêm seis plantas ou kits didáticos para experiências em sistemas de controle de processos contínuos. Estes equipamentos permitem efetuar experiências em controle de processos de nível, vazão, pressão, transferência, temperatura, posição e velocidade. Cada equipamento está associado a uma bancada do laboratório que possui um microcomputador com uma placa de aquisição de dados. O laboratório também possui uma unidade remota de entradas e saídas (I/O) analógicas e digitais de um CLP. As malhas de controle podem ser realizadas através dos microcomputadores (com interfaces via placas de aquisição de dados) ou através do CLP (via interface remota de I/O). Cada bancada comporta uma equipe de três alunos e está associada a uma única planta. Assim, somente uma equipe por vez pode realizar uma experiência distinta por aula. Ou seja, uma turma com seis equipes, totalizando dezoito alunos, não pode realizar a mesma experiência por aula de laboratório. É necessário um rodízio entre as equipes nas diferentes bancadas da sala de aula prática. Além disso, este laboratório é utilizado em média por 160 alunos em cada semestre letivo. Isto gera um número significativo de turmas práticas de laboratório e o conseqüente gerenciamento de todas as experiências programadas para todas as equipes de alunos. Uma solução seria a aquisição de mais cinco conjuntos de plantas ou kits didáticos similares aos existentes, de modo a ter seis bancadas completas que permitissem a realização de seis experiências iguais, simultaneamente, por aula prática. Uma outra solução consiste em desenvolver um sistema de baixo custo que permita acessar uma determinada experiência remotamente em períodos de horários diversificados, disponibilizados a cada semana segundo a aula programada.

Esta dissertação propõe implementar um sistema de controle à distância via Internet com a finalidade de realizar experiências práticas remotas aplicadas às plantas citadas, mas que sirva também a outros processos diferentes e que apresente custo menor que outras soluções disponíveis. O objetivo é que o usuário possa acessar uma determinada experiência prática de uma localização remota e obter dados reais a partir de um "Web-Browser", conforme ilustrado na Figura 4.1. Outro objetivo é que este trabalho possa ser utilizado e até reproduzido por outras instituições de ensino, empresas e indústrias que necessitem de treinamento em sistemas de controle, particularmente em malhas de controle de processos contínuos.

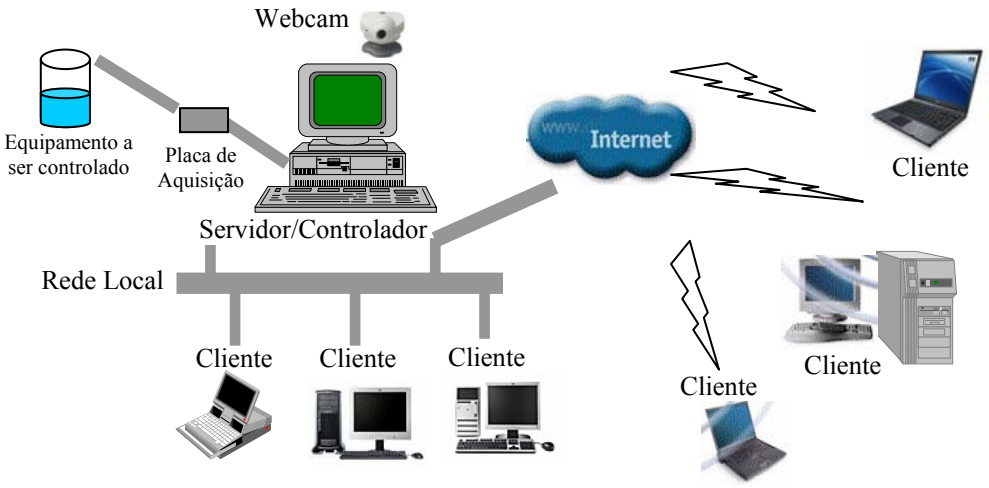

Figura 4.1 - Estrutura do Sistema de Controle à Distância via Internet.

Para realizar a implementação serão utilizados recursos de ambiente de rede de comunicação de dados cliente/servidor conectados à Internet, permitindo interações com o mundo físico, garantindo que os resultados obtidos sejam os mesmos que se obteria presencialmente no laboratório. Em princípio, os experimentos podem ser realizados a qualquer hora e de qualquer lugar, desde que se tenha acesso à Internet, o conjunto controlador/servidor e o processo a ser controlado estejam acionados. O próprio microcomputador que serve como servidor também realizará a tarefa de controlador do processo em questão.

A caracterização da experimentação remota será definida conforme os itens citados abaixo:

1. Informar no servidor os parâmetros de controle, referência da grandeza controlada, etc.;
2. Definir os experimentos, dentre os disponíveis, e a sua interação com o sistema;
3. Definir a natureza das informações recebidas, sua interação com o usuário e o programa de controle (através de gráficos para visualização dos resultados numéricos obtidos e geração de imagens reais do processo controlado);
4. Definir o procedimento de agendamento (com capacidade de gerenciamento das solicitações dos usuários) para as experiências disponibilizadas.

O diferencial deste trabalho em relação à maioria dos outros citados na referência bibliográfica, é que nesta proposta o microcomputador que realiza a transferência de informações da planta/processo para a rede de comunicação de dados, atua também como controlador do sistema. Além da vantagem de poder economizar a utilização de um controlador específico (Single-Loop, CLP, etc.), também é possível implementar através deste microcomputador algoritmos especiais como controles adaptativos, controle fuzzy e outros, incluindo funções típicas como PI, PD, PID, etc. A plataforma Windows será preferencialmente considerada devido ao motivo de ainda ser amplamente utilizada em muitos sistemas comerciais e pessoais, mesmo não possuindo características eficazes para aplicações em tempo real.

A IHM (*interface homem máquina)* entre o microcomputador e a planta a ser controlada será realizada através de placas ou sistemas de aquisição de dados comerciais de baixo custo. As funções de controle e transmissão de dados podem ser facilmente realizadas através de linguagens de programação de propósitos gerais como C++, Delphi e outras. Ou então por meio de softwares bem conhecidos e disseminados nas instituições de ensino (para propósitos de aquisição de dados e controle), tais como o MatLab/Simulink e o LabView. Os programas do sistema servirão para implementar os algoritmos de controle em tempo real, mesmo que com determinadas limitações devido ao sistema operacional a ser utilizado. Assim como as comunicações de dados entre o microcomputador controlador/servidor local (acoplado ao processo controlado) e os outros microcomputadores (remotos ou não) da rede, denominados “clientes”.

A Figura 4.2 exemplifica uma planta em escala reduzida de um sistema de nível que foi utilizado no controle remoto de uma das experiências testadas neste trabalho. O processo é montado em uma bancada didática que contém dois tanques de acrílico para armazenar água. O mesmo possui um transdutor de nível ultra-sônico e um atuador motorizado que realiza a circulação de água entre os tanques. Os sinais do transdutor e do atuador são ligados a uma placa ou sistema de aquisição de dados ligado ao microcomputador local. A Figura 4.3 mostra outro exemplo de processo de um sistema de controle de posição e/ou velocidade angular. O sistema consta de um módulo (*kit*) didático que possui uma haste metálica (representando o braço de um robô, um misturador, etc.) acionada por um servo-motor. Um transdutor de posição angular composto por um servo-potênciometro e um transdutor de velocidade constituído por um tacômetro, fornecem os sinais das grandezas básicas do processo. Estes dispositivos são ligados a placa ou sistema de aquisição de dados do computador controlador/servidor, que desta maneira pode implementar as funções de controle e estabelecer a comunicação de informações com a rede (Intranet/Internet), e assim realizar as experiências de controle a distância. A Figura 4.4 mostra a foto de um modelo (PCL-812PG da Advantech) de placa de aquisição de dados a ser conectada em um "slot" livre do computador local.

Na execução das experiências de controle algumas informações como os parâmetros de sintonia do controlador e valores de referências devem ser enviados do(s) micro(s) remoto(s) ao microcomputador local. Também os valores numéricos da variável(s) controlada(s) devem ser transmitidos do micro local ao(s) remoto(s) para visualização, assim como as imagens capturadas pela *webcam* do sistema. Isto é realizado através de telas (janelas) que serão descritas durante o trabalho com a denominação "servidor" e "cliente".

Figura 4.2 - Exemplo de planta de nível.

Figura 4.3 - Exemplo de processo de posicionamento angular.

Figura 4.4 - Exemplo de placa de aquisição de dados.

A questão sobre o sistema operacional (SO) a ser utilizado será considerada a seguir. O SO é responsável por gerenciar e organizar os recursos computacionais fornecendo uma interface mínima para os usuários. O SO deve atender aos requisitos de tempo de um sistema computacional possibilitando um desempenho otimizado dos seus recursos. Escalonamentos são efetuados objetivando atender as aplicações com requisitos de tempo real, embora conforme as aplicações existam limitações no que diz respeito a estes requisitos. Devido ao motivo de ter que dividir a atenção da unidade central de processamento (UCP) com todas as tarefas compartilhadas, vários sistemas operacionais (e suas versões) foram e são constantemente aperfeiçoados.

O sistema operacional Unix provê a funcionalidade de acesso remoto e recursos de rede/multitarefa. O problema é que SO multitarefa diminui a eficiência das tarefas em tempo real porque realiza rotinas de divisão do tempo de acesso da UCP entre os processos concorrentes da máquina.

O QNX é uma solução para aplicações específicas de tempo real. Ele é pequeno, inclui relógios e temporizadores, sendo considerado ideal para aplicações embarcadas de tempo real. Ele pode ser escalonado em tamanhos muito pequenos, prover multitarefas, “threads”, prioridade em agendamentos, e permitir uma rápida troca de contexto apresentando assim todos os requisitos essenciais para um sistema de tempo real efetivo.

O “Real Time Linux” (RTLinux) pode ser uma alternativa viável dentro do contexto de software livre, propondo suportar tarefas com restrições temporais críticas. Outros sistemas operacionais abertos e de larga difusão no meio acadêmico, como o Linux, estão emergindo cada vez mais neste mercado com grande potencial de aplicações genéricas.

É importante considerar as tendências tecnológicas, onde ainda há uma cultura predominante do sistema operacional Windows. Entre as versões atuais pode-se citar o Windows XP que é bem estável, de instalação e configuração fácil.

Visando acessar e controlar o sistema proposto, foi adotado e desenvolvido um conjunto de programas, onde optou-se por utilizar o ambiente Windows XP por ser comumente empregado em boa parte das plataformas de microcomputadores pessoais de uso geral, embora ele não garanta condições robustas de tempo real (uma vez que se forem

disparados outros processos em *foreground* pelo Windows XP, poderão ocorrer atrasos no processamento de certas tarefas), o que é o caso, uma vez que ele atua também como servidor. Porém, em muitas aplicações essas variações são pequenas e podem ser desconsideradas, porque os requisitos de tempo dos processos a serem tratados não são críticos. E nas plantas citadas esses atrasos não implicarão em degradação do desempenho dos processos sob controle. O microcomputador controlador/servidor ficará dedicado a executar o algoritmo de controle definido e transmitir os dados resultantes ao computador cliente, não gastando, deste modo, tempo adicional em outras tarefas.

Muitas linguagens de programação, em geral, não possuem características adequadas no sentido do escalonamento de tempo real. Há versões da linguagem JAVA no contexto da Internet para programação em tempo real (Real-Time Java). Houve uma melhora no desempenho das páginas no formato “HTML” quando passou-se a desenvolver programas em JAVA que é uma linguagem orientada a objetos, portável e que utiliza “object-web” (objetos utilizados para programação em ambientes de “internet”), dando grande flexibilidade para as páginas “Web”, podendo-se acessar dados remotamente através de criação de “Applets Java” (tipo de código direcionado à programação Web).

Inicialmente para o desenvolvimento dos programas de controle “cliente” e “servidor” foi utilizada a linguagem de programação Borland C++ Builder 6, que tem recursos adequados para redes cliente/servidor. Foram analisadas também linguagens de programação e formatação para ambiente "Web" tais como "Applet Java" e "HTML", mas optou-se por utilizar apenas a linguagem orientada a objetos C++ Builder.

No que diz respeito aos aspectos da interface gráfica com o usuário para o sistema proposto, algumas características na programação do sistema são necessárias:

- Que tenha parte visual, baseada em “botões” com informações sobre seus significados;
- Desenvolvida em modo gráfico para visualização de resultados;
- Visualização de experimentos por animação gráfica com as variáveis geradas no sistema real;
- Visualização de imagens captadas através da *webcam*.

O ajuste dos parâmetros do algoritmo do controlador a ser utilizado, por vezes, necessita de uma série de etapas que, dependendo da complexidade do processo, passando pela modelagem do sistema dinâmico a ser controlado e a simulação da malha de controle resultante. Muito deste trabalho é realizado com o auxílio de ferramentas de simulação.

Existem pacotes de softwares comerciais muito conhecidos e utilizados como o MatLab e o LabView, os quais possuem funções e bibliotecas específicas adequadas para comandar sistemas de aquisições em aplicações de tempo real e serão considerados em relação ao desenvolvimento utilizando uma linguagem de propósito geral como C++.

Os protocolos de comunicação a serem utilizados para comunicação entre os programas do "servidor" e do "cliente" são baseados em padrões aceitos universalmente que são:

- Protocolo "TCP/IP" que permite a transmissão de dados;
- Protocolo "HTTP" que permite a troca de informações;

Para a câmera digital (webcam) será empregado o protocolo "HTTP", com páginas "HTML" permitindo uma visualização em tempo real no "Browser" padrão como Internet Explorer ou Netscape do computador do cliente. São visualizadas as mesmas telas monitoradas e controladas localmente.

Os computadores são conectados à rede local LAN "Intranet" (rede de computadores interno) utilizando a tecnologia "Ethernet". Para a independência da plataforma de rede serão utilizados os protocolos "TCP/IP", permitindo acesso de qualquer local. Para a segurança das redes pode-se ter um *firewall*, autenticação padrão do sistema operacional com senhas de acessos e antivírus nos microcomputadores utilizados.

CAPÍTULO V

DESENVOLVIMENTO DO SISTEMA

Na primeira etapa de desenvolvimento do sistema proposto definiu-se o hardware e o sistema operacional a serem utilizados. Depois veio a fase de análise e implementação dos primeiros protótipos dos programas aplicativos (Figura 5.1). Inicialmente desenvolveu-se o software de controle "cliente" e "servidor" na linguagem Borland C++ Builder 6.

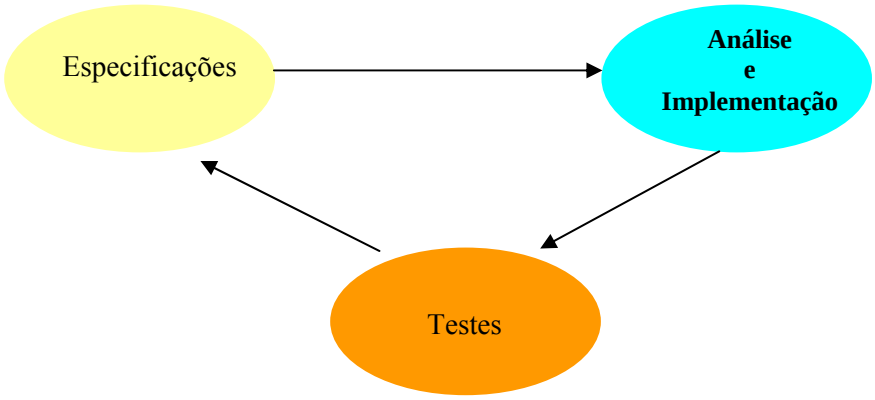

Figura 5.1 – Etapas de desenvolvimento do sistema.

As funções básicas do software estão divididas em dois módulos independentes denominados "servidor" e "cliente". Dentre os seus recursos destaca-se o acesso ao sistema através da Intranet/Internet. O servidor pode comunicar-se com aplicações clientes que estejam executando na mesma máquina (execução local) ou em máquinas diferentes. Neste último caso as máquinas devem estar interligadas em uma rede de computadores. Com isso, o servidor poderá ser acessado através da rede via protocolo HTTP e TCP/IP, onde o usuário irá acessar a bancada de experiências disponibilizada.

Por questões de segurança geralmente o sistema operacional Windows nas versões NT, 2000 e XP não permite o acesso direto a hardwares, em linguagens genéricas de programação, especificamente o acesso de portas de periféricos do computador. Para acessar as mesmas é necessário o uso de *Drivers*, que nada mais são que arquivos de programas do SO que fazem a *interface* com um periférico. Alguns arquivos de *drivers* tem as extensões

DLL, *SYS*, *DRV*, *OCX*, *VxD*, etc., conforme ilustrado no diagrama da Figura 5.2. Em algumas aplicações é possível utilizar rotinas ou funções programadas em linguagem de máquina (*assembly*) como alternativa ao uso de *Drivers* (que tendem a ser mais portáveis).

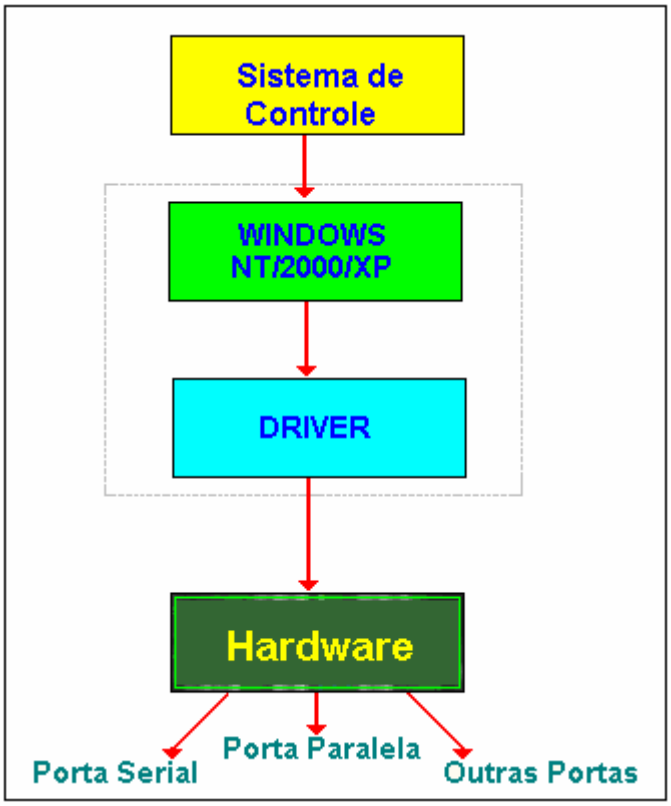

Figura 5.2 - Diagrama estrutural de um *driver*.

5.1 Protótipo 1

Inicialmente a interface gráfica do sistema proposto foi implementada em um módulo servidor representado pela Figura 5.3. O mesmo possui botões de acesso e uma área gráfica onde é visualizado o gráfico *Y* x *t*, sendo a grandeza *Y* os valores da grandeza controlada no tempo. O dado *R* é a referência de entrada (*set point*) da malha de controle. Os parâmetros *Kp*, *Ki* e *Kd* são respectivamente os ganhos proporcional, integral e derivativo do algoritmo de uma função de controle PID. O dado *N* é a quantidade de amostras a se processar, e *T* é o tempo de varredura da malha de controle. O botão *Exec. Local* (Executar Localmente) permite executar todo o processo de controle diretamente no servidor, tornando o sistema independente do cliente. Este recurso serve de teste de funcionamento da malha de controle

no local (laboratório) onde está alocado o processo a ser controlado. O botão *Lib. Cliente* (Libera Cliente) faz com que o acesso ao cliente seja liberado, permitindo o envio de parâmetros ao módulo servidor via rede. Como também o acionamento da malha de controle pelo servidor para se realizar a experiência em questão. Os dados resultantes são enviados para o módulo cliente que os recebe e monta uma tela de supervisão correspondente.

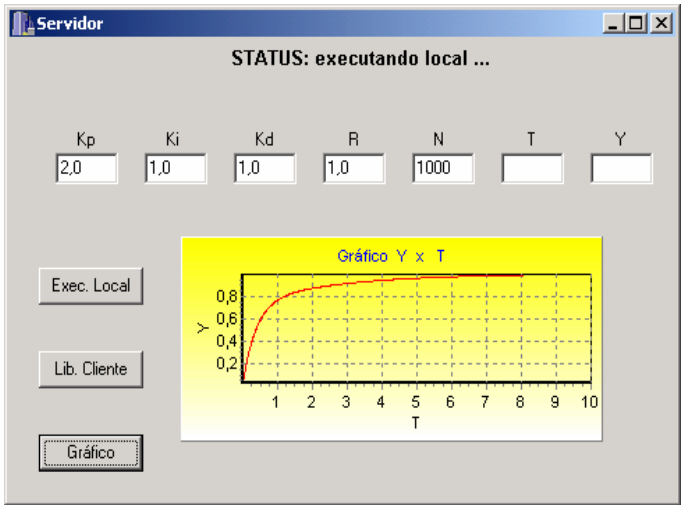

Figura 5.3 - Interface gráfica do módulo servidor.

A Figura 5.4 representa a tela denominada “cliente”. Neste módulo é configurado o endereço IP do servidor (que aguarda o envio dos parâmetros e dados) após a opção de escolha no servidor, através do acionamento do botão *Lib. Cliente* (Libera Cliente).

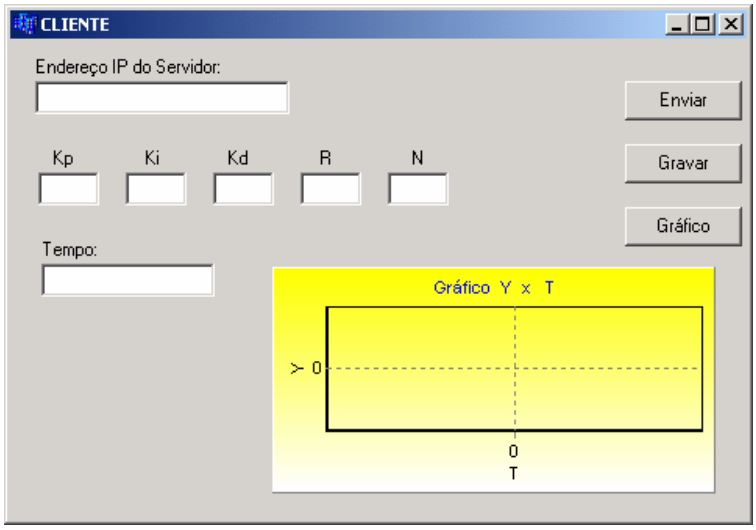

Figura 5.4 - Interface gráfica do módulo cliente.

O módulo cliente conecta-se então ao servidor e passa a enviar e receber dados reais da experiência à distância. Com este recurso o usuário acessa experiências de controle de processos de qualquer lugar, sendo visualizadas as mesmas telas de informações obtidas e armazenadas no servidor. Além dos botões de acesso e dos campos de leitura dos parâmetros, o módulo cliente também apresenta uma área gráfica no qual serão visualizados gráficos *Y* x *t* com os dados reais recebidos do servidor. Caso haja algum problema com o servidor, impossibilitando o seu uso, ou caso ainda não esteja liberado o acesso pelo servidor, será emitido uma notificação de acesso não liberado tornando os módulos completamente integrados, disponibilizando os dados *on-line* assim que o sistema volte a estar acessível. Os programas fontes dos módulos "servidor" e "cliente" são mostrados no Anexo I. Na parte do código referente ao acesso das interfaces da placa de aquisição de dados inicialmente utilizada (modelo PCL-812PG da Advantech), especificamente os canais de entrada e saída analógicas, utilizou rotinas programadas em linguagem de máquina através de diretivas do tipo *asm*. A Listagem 5.1 ilustra o procedimento de leitura do conversor A/D da placa de aquisição de dados. O acesso do timer/counter do microcomputador (para determinar ou estabelecer o tempo de varredura da malha de controle) também teve parte do seu código escrito em *assembly*. A Listagem 5.2 mostra o código correspondente.

Listagem 5.1 - Rotina de leitura do conversor A/D.

```
EndIO=EndBase+12;
asm mov al,0;
asm mov dx,EndIO;
asm out dx,al;   /* Inicia conversao do A/D. */
do {
    EndIO=EndBase+5;
    asm mov dx,EndIO;
    asm in al,dx;              /* Ler bits MSB do A/D. */
       asm mov dadoMSB,al;
       dado = dadoMSB & 0x10;       /* Mascara do bit de conversao. */
     } while (dado != 0);          /* Teste de final de conversao. */
EndIO=EndBase+4;
asm mov dx,EndIO;
asm in al,dx;                  /* Ler bits LSB do A/D. */
asm mov dadoLSB,al;
dado = (dadoMSB << 8) + dadoLSB;   /* Dado em decimal. */
```

Listagem 5.2 - Rotina de leitura do timer.

```
asm mov al,128;     /* T/C2 modo leitura. */
asm out 43h,al;
asm in al,42h;      /* Ler parte LSB. */
asm mov x1,al;
asm in al,42h;      /* Ler parte MSB. */
asm mov x2,al;

JI=256*x2+x1;       /* Converte em 16 bits. */
if (JI < 0) te=65536.0+JI;
    else te=JI;     /* Testa se positivo. */
te=65535.0-te;
te=te/1.19318e6;    /* Tempo de varredura. */
```

A parte do programa referente à implementação do algoritmo PID, foi desenvolvida utilizando equações numéricas correspondentes às funções proporcional, integral e derivativa desejadas. Outros algoritmos de controle (adaptativos, Fuzzy, etc.) podem ser desenvolvidos também. A Listagem 5.3 mostra o trecho de código referente a implementação do algoritmo PID.

Listagem 5.3 - Rotina do algoritmo PID.

```
e = r-y;                    /* Erro da Malha. */
p = kp * e;                 /* Parte Proporcional. */
i = i + ki * e * te;        /* Parte Integral. */
d = (kd*e – Id)/(0.1*kd);   /* Parte Derivativa. */
Id = Id + d*te;
u = p+i+d;                  /* Lei de Controle. */
```

5.2 Protótipo 2

Durante o desenvolvimento do software e na fase de testes, foram constatadas algumas limitações nos recursos visuais da interface gráfica com o usuário (GUI) apresentada pela linguagem utilizada (Borland C++ Builder 6). Não foi alcançada uma interface gráfica agradável, deixando-se a desejar na parte visual e gráfica do sistema.

Foi levantada uma questão referente a portabilidade e a legibilidade do código do algoritmo de controle, que foi realizado, uma parte, em linguagem de máquina e dependia especificamente do modelo da placa de aquisição de dados utilizada.

Durante o desenvolvimento deste projeto foram adquiridos outros sistemas de aquisição de dados mais acessíveis que os inicialmente utilizados. A diferença de preços destes itens foi de aproximadamente 1/7 por unidade. Outra vantagem é que o sistema é conectado ao microcomputador controlador através de uma *interface* USB padrão, não sendo necessário conectar a um *slot* de expansão interno do computador. Além disso, este sistema tem disponível *drivers* para algumas linguagens de programação com características visuais elaboradas e com softwares comerciais de aquisição de dados e controle como o LabView.

Foi elaborado então um outro software usando recursos das bibliotecas de Instrumentação Virtual (VI) do LabView, deixando o sistema com recursos gráficos mais interessantes e também mais portátil (no caso de mudança do sistema de aquisição de dados basta selecionar o *driver* correspondente, não sendo necessário mudar os programas aplicativos). A *interface* com o sistema disponibilizará botões de controle, tela gráfica e um sinóptico representativo do processo, deixando uma aparência mais agradável e amigável para o usuário. A seguir serão mostradas as novas implementações realizadas no sistema proposto.

Serão apresentadas as implementações dos subsistemas “servidor” e “cliente” utilizando agora o software LabView. O conceito utilizado é o mesmo da implementação citada anteriormente e desenvolvida através do compilador Borland C++ Builder 6.

O tutorial do LabView fornece alguns exemplos que demonstram a utilização de recursos de relógio RTC de controle de tempo real e conexão em rede utilizando o protocolo TCP/IP. Estes recursos estão disponíveis em ambiente de programação gráfica altamente amigável que satisfaz a maioria das necessidades básicas de desenvolvimentos de aplicações genéricas [18]-[20].

As funções básicas do software seguem a mesma linha de raciocínio da versão anterior, sendo divididas nos dois módulos independentes, denominados Servidor e Cliente (Figura 5.5).

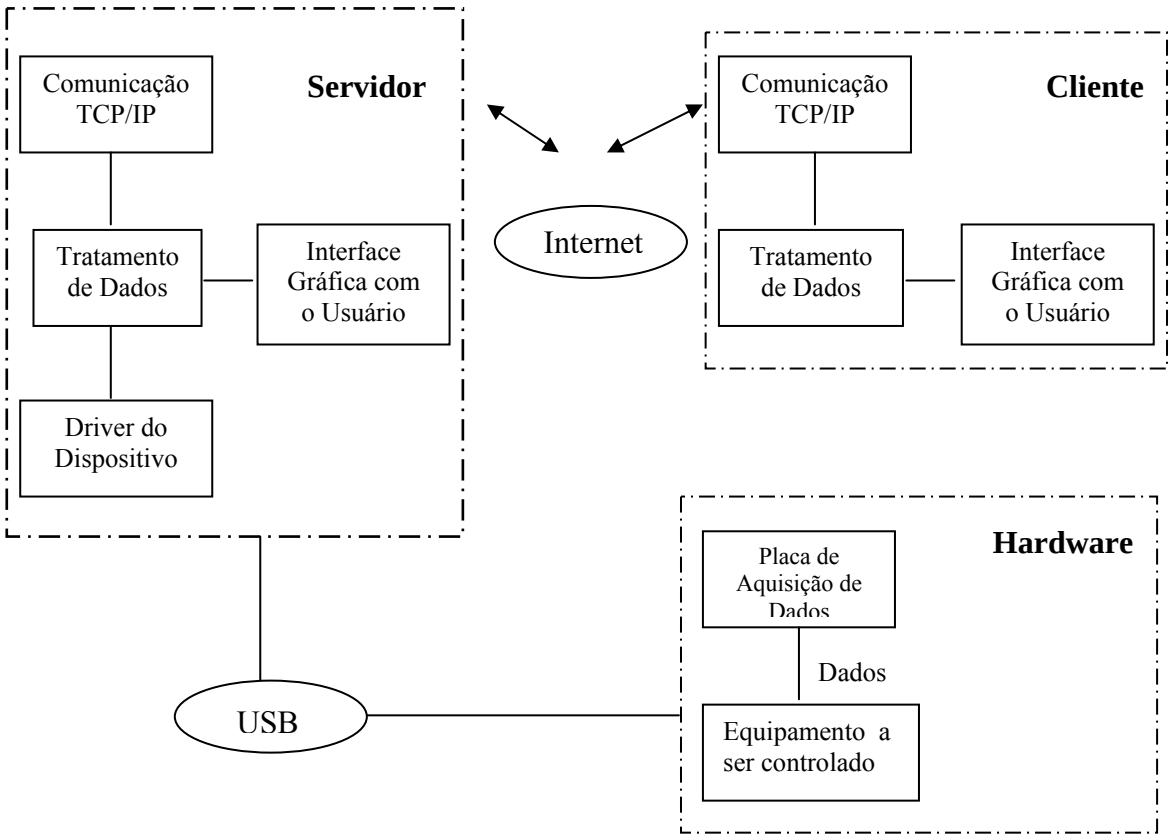

Figura 5.5 - Estrutura do sistema desenvolvido.

O módulo denominado "Comunicação TCP/IP", disponível tanto no servidor como no cliente, é responsável por estabelecer a comunicação via "socket" [1] entre o usuário e o servidor onde está sendo realizado o experimento. Este módulo envia as informações configuradas pelo cliente e retorna os dados da experiência controlada pelo servidor.

O módulo "Tratamento de Dados" existente tanto no servidor como no cliente tem como função a codificação e decodificação das informações geradas e dos dados enviados pelo cliente para serem tratados pelo *driver* da placa de aquisição de dados, e também para a comunicação via TCP/IP.

A "Interface Gráfica com o Usuário" tem o propósito de permitir o monitoramento da experiência on-line e o diagnóstico do sistema como um todo. A *interface* gráfica com o

[1] Protocolo para troca de dados entre computadores via protocolo TCP/IP.

usuário é importante, pois é por meio dela que ocorrerão todas as tarefas de visualização dos dados da experiência remota.

O *hardware* (sistema de aquisição de dados) se comunica com o computador por meio de uma porta USB. O programa servidor dispõe de um módulo para comunicação com o *driver* USB possibilitando o acesso ao sistema de aquisição, deixando o sistema mais portátil no caso da eventual troca do mesmo, bastando selecionar o *driver* correspondente ao novo modelo.

A foto da Figura 5.6 ilustra os equipamentos utilizados pelo sistema atual proposto. O processo a ser controlado é de uma planta didática de um sistema de nível, mas poderia ser qualquer outro sistema. Na figura é fácil visualizar também o sistema de aquisição de dados (modelo NI USB-6008 da National Instruments), o microcomputador "Controlador/Servidor" e a *webcam*.

A foto da Figura 5.7 ilustra a tela do computador "Cliente", onde são visualizados os gráficos de experimentos, uma representação sinóptica simples do processo e a imagem real da planta transmitida pela *webcam*.

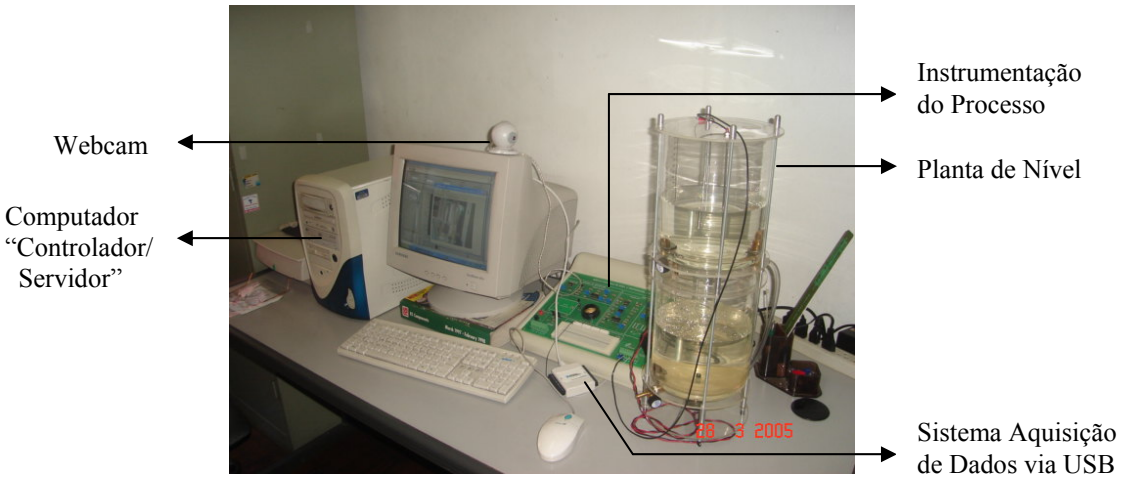

Figura 5.6 - Ilustração do sistema "Servidor".

Figura 5.7 - Ilustração do sistema “Cliente”.

Os parâmetros da experiência real são apresentados na tela do programa “Cliente” e ao serem ajustados funcionam como se o usuário estivesse atuando diretamente sobre o experimento, ou seja, a dinâmica do processo controlado pode ser modificada. Para cada ação do usuário um retorno visual é apresentado, o que propicia uma representação fiel do estado do sistema que está sendo ensaiado.

Na Figura 5.8 é mostrada a tela de *interface* com o usuário do módulo “Servidor”. Definiu-se um formato que permitisse uma comunicação clara com o usuário e cuja operação fosse intuitiva. A *interface* possui botões de acesso para entradas dos parâmetros do sistema, uma área gráfica onde é visualizado o valor de referência da malha controladora e a resposta da grandeza controlada no tempo. Há também uma representação sinóptica simples do processo supervisionado. Neste exemplo a grandeza a ser controlada é a coluna de líquido de uma planta de nível.

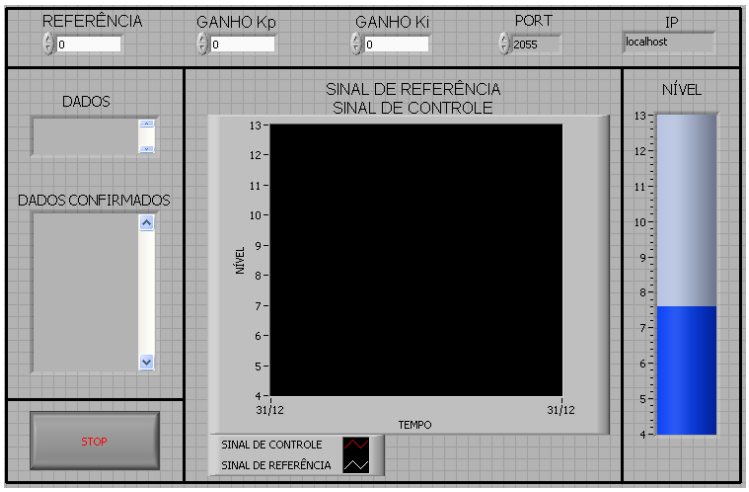

Figura 5.8 - Interface Front Panel [2] do módulo Servidor.

[2] Tela de interação entre usuário e o programa desenvolvido no LabView.

A Figura 5.9 mostra a programação funcional do módulo Servidor. O desenvolvimento do programa é realizado através de diagrama em blocos funcionais do LabView. Os blocos "DAQ Assistant" são os *drivers* de comunicação do sistema de aquisição de dados. Neles são definidos os canais das entradas e saídas analógicas utilizadas e o modelo do sistema de aquisição empregado. O bloco com essa denominação a esquerda da Figura 5.9 representa a entrada analógica que recebe o dado do transdutor do processo, e o bloco a direita da figura simboliza a saída analógica que fornece o sinal de comando do atuador da planta. Na eventualidade de se trocar o modelo do sistema de aquisição de dados, basta re-configurar estes blocos sem a necessidade de modificações adicionais nos outros blocos funcionais do programa. As funções dos outros blocos serão explanadas a seguir.

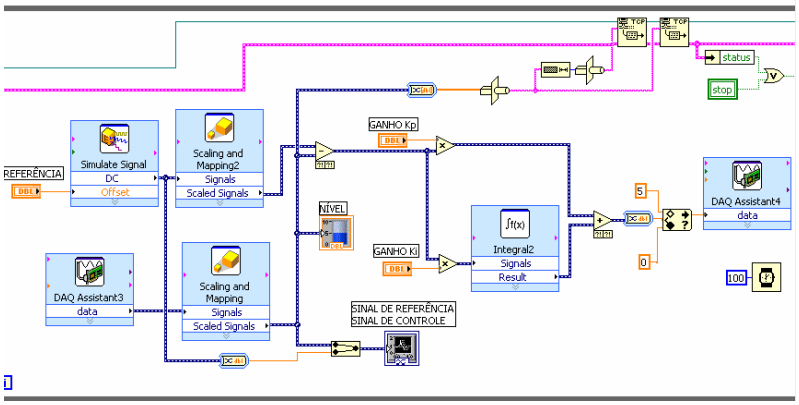

Figura 5.9 - Diagrama em blocos do módulo "Servidor"

Os blocos denominados "REFERÊNCIA" e "Simulate Signal" operam com a finalidade de receber valores numéricos da referência de entrada da malha de controle, estando relacionado com o botão e o respectivo campo da tela de supervisão da Figura 5.8. Os blocos "Scaling and Mapping" sevem para escalonar valores, por exemplo, convertendo o dado do sinal do transdutor do processo em um valor correspondente em unidade de engenharia da grandeza controlada. A função com as etiquetas "SINAL DE REFERÊNCIA E SINAL DE CONTROLE" gera os gráficos correspondentes na tela gráfica do módulo servidor. A função denotada como "NÍVEL" realiza uma representação sinóptica simples do processo. O bloco de subtração (-) calcula o erro da malha de controle, ou seja, a diferença entre o valor da referência desejada e a informação da grandeza física realimentada (conforme explicado no Capítulo 3). Os blocos "GANHO Kp" e "GANHO Ki" são as entradas de dados dos parâmetros da função do controlador (neste exemplo um PI), cujos valores são

multiplicados com o valor do erro da malha de controle através das respectivas funções de multiplicação (x). O bloco “Integral” realiza a integração do erro e o valor resultante é somado através da função soma (+) com a parte proporcional obtendo-se a lei de controle desejada. Se for necessária uma ação derivativa ou outras leis (ou estratégias) de controle, isto pode ser facilmente programado utilizando-se blocos de funções aritméticas básicas, de integração, etc.

O tempo de varredura da malha de controle é imposto pelo bloco definido por um ícone de um relógio (neste sistema foi ajustado um tempo de 0,1 segundos). O botão “Stop” serve para finalizar a execução do módulo. Finalmente a comunicação dos dados do módulo “Servidor” com a Internet é realizada através dos blocos de cor amarela no canto superior direito da Figura 5.9. Na *interface Front Panel* do módulo servidor têm-se os correspondentes botões e campos para a definição da porta (PORT) e do endereço IP do sistema.

O módulo cliente foi desenvolvido de maneira semelhante ao módulo servidor. A Figura 5.10 ilustra a tela do mesmo, onde é possível notar a mesma área gráfica para visualização dos gráficos dos experimentos realizados, um sinóptico simplificado do processo e uma outra área para dados da malha de controle, além dos botões e campos para definição da porta (PORT) e endereço IP do módulo servidor.

Em tempo de execução o módulo cliente conecta-se ao servidor que passa a enviar e receber os dados reais da experiência de controle realizada à distância. Com este recurso, o usuário acessa experiências de qualquer lugar, sendo visualizadas as mesmas telas de informações obtidas e armazenadas no servidor.

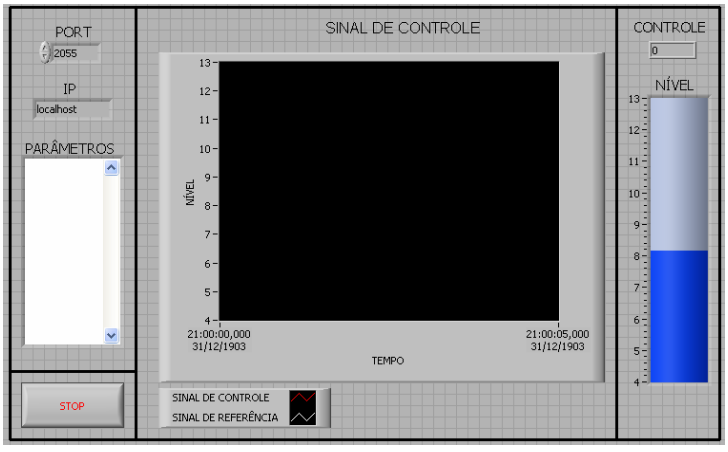

Figura 5.10 – Interface do módulo “Cliente”.

A Figura 5.11 mostra a programação funcional do módulo “Cliente”. Os blocos dentro do retângulo superior da figura realizam a comunicação de dados entre os parâmetros de controle dos módulos. Os blocos do retângulo inferior efetuam a recepção dos dados relativos a experiências executadas no módulo “Controlador/Servidor”. Essas funções fazem parte da biblioteca do software LabView e são bem fáceis e simples de se programar e utilizar.

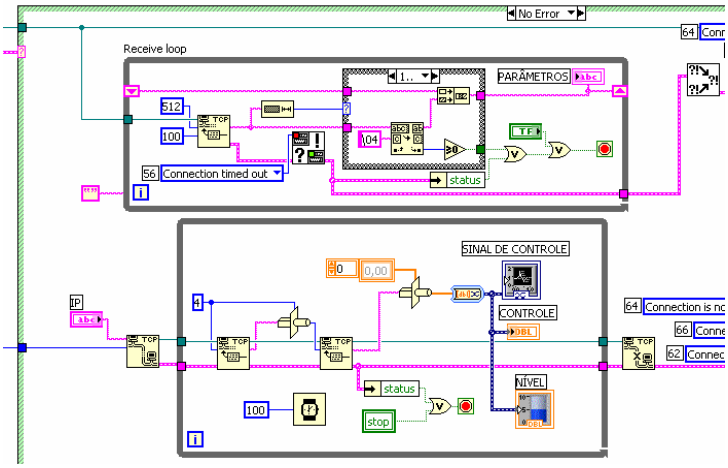

Figura 5.11 - Diagrama em blocos do módulo “Cliente”.

O módulo de captura de imagens da *webcam* do computador “Controlador/Servidor” e a transmissão das mesmas ao computador “Cliente” é realizada conforme descrito a seguir. A conexão com o servidor é feita através da porta 80. Esta porta é usada pelo servidor para que se tenha a garantia de que a conexão da rede seja atendida com eficiência.A Figura 5.12 mostra a tela do módulo em questão. Para a conexão com o “servidor” se faz necessário, também a configuração do endereço IP do servidor de imagens (juntamente com a porta de acesso). A Figura 5.13 ilustra a tela de recepção de imagens no computador cliente. Os programas fontes do módulo WebCam são mostrados no Anexo II e alguns trechos dos programas estão ilustrados nas Listagens 5.4 e 5.5.

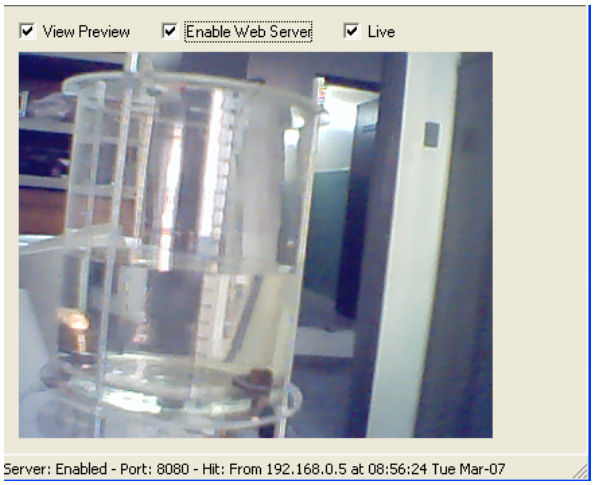

Figura 5.12 - Módulo WebCam servidor de imagens.

Figura 5.13 - Módulo WebCam-Cliente.

Listagem 5.4 – Programação do módulo WebCam – Cliente.

```
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Button1: TButton;
    WebBrowser: TWebBrowser;   // Acesso ao Navegador
    Endereco: TEdit;                  // Endereço IP do Servidor de Imagem
```

```
    Label1: TLabel;
    Button2: TButton;
    CheckBox1: TCheckBox;        // Botão de Tempo Real
    Timer1: TTimer;                    // Tempo do microcomputador
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);

begin
 WebBrowser.Navigate( Endereco.Text );   // Acesso ao Navegador através do Parâmetro do
                                                    endereço IP do Servidor
end;

procedure TForm1.Button2Click(Sender: TObject);

begin
 Close;
end;

procedure TForm1.Timer1Timer(Sender: TObject);   // Rotina que atualiza a imagem no
```

```
                                                    navegador
begin
 if checkbox1.Checked then                  // Botão de tempo real ativado
   button1click(sender);                    //  Exibe a imagem em tempo real
end;

end.
```

Listagem 5.5 – Programação do módulo WebCam – Servidor

```
#include "webcam.h"
//---------------------------------------------------------------------------
// Execução externa
// Através da diretiva #pragma, o C++Builder pode definir diretivas que podem ser passadas ao compilador. Se o compilador não identificar o nome da diretiva, ele simplesmente ignora #pragma sem emitir qualquer aviso ou mensagem de ERRO. A sintaxe de pragma é #pragma nome_da_diretiva.

#pragma package(smart_init)
#pragma link "VLCommonDisplay"
#pragma link "VLDSCapture"
#pragma link "VLDSImageDisplay"
#pragma link "VLDSVideoLogger"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
```

O procedimento para a utilização do sistema está detalhado a seguir:

1- O usuário fará uma requisição para utilização do laboratório remoto agendando por e-mail ou seguindo-se de uma programação de horários pré-estabelecida.

2- Um tempo antes do horário estabelecido, o sistema "Controlador/Servidor" é ligado e fica à espera da solicitação do início de um ensaio pelo usuário. A planta a ser controlada também deve estar ligada esperando ser acionada pelo sistema de aquisição de dados.

3- Os parâmetros de controle, endereços de portas e de IP devem ser estabelecidos.

4- Os ensaios são então realizados e o usuário obtém os dados e imagens correspondentes remotamente através do computador "Cliente". Os gráficos e imagens capturadas podem ser gravados para utilização posterior.

5- Após o período de tempo total programado para a realização da experiência, ou mediante mensagem de final de experiência, o usuário deixa de ter acesso ao sistema.

6- O sistema fica então disponível para utilização por outro usuário seguindo-se a programação de horários estabelecidos.

No próximo capítulo serão mostrados os resultados obtidos com a versão atual do sistema proposto. Para exemplificar e testar os programas desenvolvidos será empregada uma planta de nível. Outros processos podem ser igualmente controlados. Eventuais alterações e outras implementações podem ser realizadas com algumas modificações nos programas (servidor e cliente) desenvolvidos neste capítulo.

CAPÍTULO VI

RESULTADOS OBTIDOS

Com o objetivo de testar o sistema proposto foram realizados ensaios com uma planta didática de nível. Os resultados obtidos serão mostrados a seguir. O cálculo do ganho proporcional (Kp) e do ganho integral (Ki) da malha de controle com compensação PI foi efetuado de modo similar ao mostrado no Capítulo 3. Para o modelo da planta empregada foram estimados os ganhos Kp = 6 e Ki = 0,1. O valor de ensaio da referência de entrada da malha de controle foi de 8 [cm]. A Figura 6.1 mostra o resultado correspondente obtido no ensaio efetuado. O gráfico em linha cinza indica a referência de entrada e a linha vermelha sobreposta mostra a grandeza controlada. A figura ilustra uma tela capturada depois de decorrido um determinado tempo suficiente para a resposta estabilizar, ficando o processo em regime permanente. O gráfico é dinâmico e mostra as variações da grandeza controlada em tempo real, com amostras atualizadas a cada décimo de segundo. O sinóptico simplificado indica em cor azul o nível controlado.

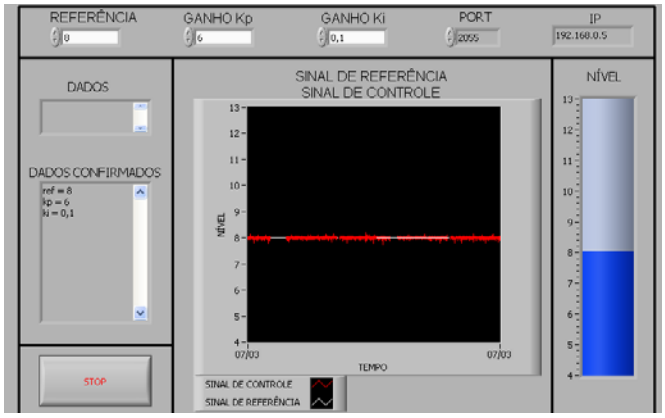

Figura 6.1 - Resultados obtidos na tela de supervisão do computador “Controlador/Servidor”.

A Figura 6.2 mostra os dados recebidos no computador “Cliente” proveniente das informações obtidas no computador “Servidor”, assim como as imagens capturadas pela

webcam e transmitidas também via Internet. A figura ilustra uma tela capturada do módulo cliente, que em tempo efetivo mostra dinamicamente no gráfico as variações da grandeza controlada. Os dados atualizados no gráfico, no sinóptico e na tela de imagem da câmera digital podem conter atrasos de propagação da rede de comunicação de dados, mas para efeito de observação dos resultados obtidos na experiência remota, estes não atrapalham a visualização efetiva dos ensaios realizados na prática.

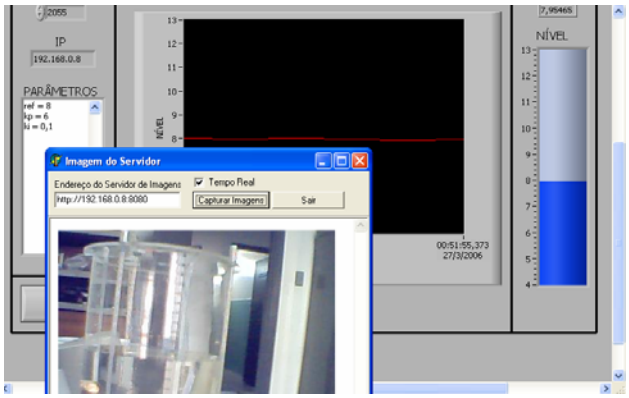

Figura 6.2 - Resultados recebidos no computador “Cliente”.

É fácil notar que os resultados obtidos foram bem adequados. O sistema mostrou ser de simples utilização e de funcionamento bem efetivo. Foram realizados outros ensaios com valores distintos de referência de entrada e parâmetros (ganhos) de controle. Os resultados foram igualmente adequados aos propósitos deste trabalho. O sistema proposto está sendo atualmente implementado em praticamente todas as plantas (vazão, pressão, temperatura e posição) citadas do Laboratório de Controle de Processos Contínuos Industriais da Universidade Federal de Itajubá. Os resultados desta dissertação estarão disponíveis para qualquer instituição de ensino que deseje reproduzi-los, e esta talvez seja a maior contribuição deste trabalho, ou seja, viabilizar um sistema prático, simples e de custo mais reduzido que as implementações convencionais com múltiplas bancadas iguais.

CAPÍTULO VII

CONCLUSÃO

Este trabalho teve como objetivo implementar um sistema simples e eficaz, que pudesse efetuar experiências de malhas de controle à distância, utilizando recursos de comunicação de dados, como a Internet. A finalidade é possibilitar a execução remota de experiências reais de sistemas de controle tais como plantas didáticas de nível, vazão, posição, temperatura, etc.

Foram utilizados placas ou sistemas de aquisição de dados comerciais de baixo custo para realizar a *interface* entre o microcomputador pessoal atuando como servidor/controlador e a planta a ser controlada. As funções de controle e transmissão de dados a computadores remotos foram implementadas através de linguagens de programação de propósitos gerais, ou por meio de softwares bem conhecidos e disseminados nas instituições de ensino para propósitos de aquisição de dados e controle.

O sistema proposto também incorpora uma câmera digital que permite a observação visual das experiências realizadas remotamente. A vantagem do sistema é que uma única bancada de laboratório pode ser compartilhada a distância por vários usuários independentemente das suas localizações físicas.

O sistema proposto mostrou ser de simples utilização e de funcionamento bem efetivo. Foram realizados ensaios em condições diversas e os resultados obtidos foram adequados aos propósitos deste trabalho. A utilização do sistema para outras plantas (de vazão, pressão, temperatura, posição, etc.), pode ser realizada sem grandes modificações nos programas desenvolvidos, bastando ajustar as unidades das grandezas físicas controladas e eventuais fatores de escala nas representações gráficas.

Os resultados desta dissertação estarão disponíveis para qualquer instituição de ensino que deseje reproduzi-los, e esta talvez seja a maior contribuição deste trabalho, ou seja, viabilizar um sistema prático, simples e de custo mais reduzido que as implementações convencionais com múltiplas bancadas iguais.

Não existe a pretensão que um laboratório remoto possa substituir um laboratório presencial, mas sim servir de ferramenta adicional e complementar de apoio ao ensino tecnológico nas áreas de sistemas de controle.

Para trabalhos futuros pode-se citar:

→ A implementação de estratégias de controle mais elaboradas como controladores com lógica Fuzzy, adaptativos e outros.

→ A incorporação de textos digitais com material teórico para efetivar cursos a distâncias com metodologias de Educação à Distância, proporcionando um aprendizado integral e eficiente;

→ Aprimoramento dos métodos de agendamento atuais para um método automatizado, podendo ser gerenciado via software, possibilitando maior segurança e agilidade.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Aguirre, L. A., “Introdução à Identificação de Sistemas”, Ed. UFMG, 2000.

[2] Arpaia, P, Baccigalupi, A, Cennamo, F, Daponte, P, "A remote measurement laboratory for educational experiments", *Measurement*, v. 21, n. 4, 1997, pp. 157-169.

[3] Bhandari, A, Shor, M, "Access to an Instructional Control Laboratory Experiment through the World Wide Web", *Proc. ACC´98*, pp. 1319-1325, 1998.

[4] Dixon, W., E., Dawson, D., M., Costic, B. T., “Toward the standardization of a MATLAB-based control system laboratory experience for undergraduate students“ in Proc. ACC´01, Arlington, pp. 1161-1166, 2001.

[5] Henry, J. “Virtual attendance in real engineering labs” Proc. IEEE, pp. 61-65, 2002.

[6] Jochheim, A., Rohrig, C., “The virtual lab for teleoperated control of real experiments,” in Proc. 38th Conf. Decision & Control, Phoenix, pp. 819-824, 1999.

[7] Nitin, S., Ognjen, K., Frank, L., “Internet-basead education control systems lab using NetMeeting” in Proc. IEEE Transactions on Education, pp. 145-151, 2002.

[8] Novaes, A, G, "Ensino à distância na Engenharia: Contornos e perspectivas", *Revista do Depto de Eng. de Produção da Universidade Federal de São Carlos*, v. 1, n. 3, 1994.

[9] Ogata, K. “Engenharia de Controle Moderno”, 4ª Ed., Prentice Hall, 2003.

[10] Phillips, C. Harbor, R. D. “Feedback Control Systems”, Prentice Hall, 1996.

[11] Poindexter, S, E, Heck, B, S, "Using the Web in your Courses: What Can You Do? What Should You Do? ", *IEEE Control Systems Magazine*, v. 19, n. 1, 1999, pp. 83-92.

[12] Ramakrishnan V. et al. “Development of web-basead control experiment for a couple tank apparatus”, Proceedings of the American Control Conference, pp. 4409-4413, 2002.

[13] Rodrigo, M, F, "Laboratório Remoto de Eletrônica de Potência" *Trabalho de diploma, Universidade Federal do Rio de Janeiro – Depto de Eletrotécnica*, p. 133, 2002.

[14] Rohrig, C., Jochheim, A., “The virtual lab for controlling real experiments via Internet“ in Proc. IEEE Int. Symp. Computer Aided Control System Design, pp. 279-284, 1999.

[15] Salzmann, C, Latchman, H, Gillet, D, Crisalle, O, "Virtual Laboratories and Real-time Experiments in Engineering Education", *ICEE’99*, 1999.

[16] Silva, O, F, Sampaio, O, A, K, Alves J, B, M, Barreiros, J, A, L, "Mídias e tecnologias instrucionais para o ensino/aprendizado de sistemas de controle", *Anais do XIV Congresso Brasileiro de Automática* (*CD ROM),* 2002.

[17] Snachés J. et al., “A Java/Matlab-based environment for remote control system laboratories illustrated with an inverted pendulum” in Proc. IEEE Transactions on Education, pp. 321-329, 2004.

[18] LabVIEW User Manual, National Instruments, 2003.

[19] LabVIEW Real-Time Communication Wizard, National Instruments, 2004.

[20] LabVIEW Real-Time Module User Manual, National Instruments, 2004.

[21] MatLab User Manual,

[22] Simulink and ModelSim Tutorial

[23] Simulink User's Guide

ANEXO I

```
/*************************************************************************/
/*                          Programa:  Cliente                           */
/*                          Borland C++ Builder 6,                       */
/*                          Helenice Cristina Ferreira                   */
/*                                                                       */
/*************************************************************************/

//---------------------------------------------------------------------
// Includes

#include <vcl.h>
#pragma hdrstop
#include <time.h>          // include de controle de tempo
#include <stdio.h>         // include de entrada/saida
#include <dos.h>           // include de recursos do sistema operacional
#include <conio.h>          // include de uso de teclado e monitor
#include <stdlib.h>        // include de diversas utilidades
#include <math.h>          //include de funções matemáticas
#include "Unit1_cli.h"

//---------------------------------------------------------------------
// Execução externa
// Através da diretiva #pragma, o C++Builder pode definir diretivas que podem ser passadas ao compilador. Se o compilador não identificar o nome da diretiva, ele simplesmente ignora #pragma sem emitir qualquer aviso ou mensagem de ERRO. A sintaxe de pragma é #pragma nome_da_diretiva.

#pragma package(smart_init)
#pragma resource "*.dfm"
```

```
//---------------------------------------------------------------------------
// Declaração de vetores e variáveis

int lv[10],yv[10];

long int startticks;      // variável de controle de tempo
unsigned char X0,X1,X2;
float DT;

TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

//---------------------------------------------------------------------------


void __fastcall TForm1::Button1Click(TObject *Sender)
{
  float kp, ki, kd, r;     // variáveis do controlador PID
  int k;
  String buffer;

  NMUDP1->RemoteHost = LabeledEdit1->Text;   // Leitura do endereço IP
  NMUDP1->LocalPort = 5000;       // definição da Porta utilizada localmente
  NMUDP1->RemotePort = 5000;    // definição da Porta utilizada remotamente

  kp = StrToFloat(LabeledEdit2->Text);       // leitura do parâmetro kp
  ki = StrToFloat(LabeledEdit4->Text);       // leitura do parâmetro ki
  kd = StrToFloat(LabeledEdit6->Text);       // leitura do parâmetro kd
  r  = StrToFloat(LabeledEdit5->Text);       // leitura do parâmetro r
  k  = StrToInt(LabeledEdit7->Text);         // leitura do parâmetro k
```

```
 buffer = FormatFloat("0.000",kp)+FormatFloat("0.000",ki)+       // conversão no formato
                                                                  //decimal do parâmetros kp, kd, ki, r e k
        FormatFloat("0.000",kd)+FormatFloat("0.000",r)+
        FormatFloat("00000",k*1.0);

 NMUDP1->SendBuffer(buffer.c_str(),30,30);       // envio dos parâmetros ao servidor
 startticks = GetTickCount();                    // contagem do tempo real
}
//---------------------------------------------------------------------------

// Função de Recepção dos parâmetros na porta especificada e cálculo do tempo

void __fastcall TForm1::NMUDP1DataReceived(TComponent *Sender,
    int NumberBytes, AnsiString FromIP, int Port)
{

  LabeledEdit3->Text = FloatToStr(DT);

  LabeledEdit3->Text = FloatToStr((GetTickCount()-startticks)*0.055)+" ms"; // cálculo do
tempo
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
for (int i=1; i<=10; i++)
  {
  lv[i]=rand();
  yv[i]=rand();
  }

  for (int i=1; i<=10; i++)
  {
```

```
  Series1->AddXY(lv[i],yv[i],"",clRed);      // montagem do gráfico, de barras, para o cliente
   }

}
//---------------------------------------------------------------------

/*************************************************************************/
/*                                  Programa   Servidor                  */
/*                                 Borland C++ Builder 6,                */
/*                                Helenice Cristina Ferreira             */
/*                                                                       */
/*************************************************************************/


//---------------------------------------------------------------------
// Includes

#include <vcl.h>
#include <math.h>     //include de funções matemáticas
#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------
// Execução externa
// Através da diretiva #pragma, o C++Builder pode definir diretivas que podem ser passadas ao compilador. Se o compilador não identificar o nome da diretiva, ele simplesmente ignora #pragma sem emitir qualquer aviso ou mensagem de ERRO. A sintaxe de pragma é #pragma nome_da_diretiva.

#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
```

```
//---------------------------------------------------------------------------
// Declaração de vetores e variáveis

float tv[1000],yv[1000];
bool local;
int dado, dadoLSB, dadoMSB, EndBase, EndIO, JI, x1, x2; // variáveis da placa de aquisição
                                                          de dados

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
  : TForm(Owner)
{
}
//---------------------------------------------------------------------------

/* Inicializacao de interfaces (dados do fabricante da PCL-812PG). */
 EndBase=0x230;     /* Endereço base da placa de aquisição de dados. */

 EndIO=EndBase+11;
 asm mov al,1;
 asm mov dx,EndIO;
 asm out dx,al;            /* Habilita conversão por software do A/D. */

 EndIO=EndBase+10;
 asm mov al,10;
 asm mov dx,EndIO;
 asm out dx,al;            /* Seleção do canal 10 - pino 1 do conector. */

 EndIO=EndBase+9;
 asm mov al,0;
 asm mov dx,EndIO;
 asm out dx,al;                  /* Ganho unitário na entrada do A/D. */

 dado = 0;                       /* Valor inicial do D/A. */
```

```
  dadoMSB = dado >> 8;              /* Deslocar oito vezes para a direita. */
  dadoMSB = dadoMSB & 15;           /* Pegar apenas os primeiros 4 bits. */
  dadoLSB = dado & 255;             /* Pegar os 8 bits da esquerda. */

  EndIO=EndBase+4;
  asm mov al,dadoLSB;
  asm mov dx,EndIO;
  asm out dx,al;            /* Escrever nos registros do D/A. */

  EndIO=EndBase+5;
  asm mov al,dadoMSB;
  asm mov dx,EndIO;
  asm out dx,al;            /* Escrever nos registros do D/A. */

void calculo(float kp, float ki, float kd, float r, int k)
{
   float e,d,Id,p,u,i,t,te=0.01;

   y=i=Id=t=0;

   asm cli;                  /* Desabilita interrupção. */

   asm in al,61h;           /* Lê status do Gate2 do Timer do micro. */
   asm or al,01h;           /* Mascara bits. */
   asm out 61h,al;          /* Ativa Gate2. */

   do
   {
     asm mov al,180;              /* T/C2, Modo 2, 16 bits. */
     asm out 43h,al;              /* Programando Timer. */
     asm mov al,00h               /* Contagem máxima. */
     asm out 42h,al;             /* LSB. */
     asm out 42h,al;             /* MSB. */
```

```
 EndIO=EndBase+12;
 asm mov al,0;
 asm mov dx,EndIO;
 asm out dx,al;              /* Inicia conversão do A/D. */

 do {
    EndIO=EndBase+5;
    asm mov dx,EndIO;
    asm in al,dx;            /* Lê bits MSB do A/D. */
       asm mov dadoMSB,al;

       dado = dadoMSB & 0x10;        /* Mascara do bit de conversão. */
     } while (dado != 0);            /* Teste de final de conversão. */

 EndIO=EndBase+4;
 asm mov dx,EndIO;
 asm in al,dx;                     /* Lê bits LSB do A/D. */
 asm mov dadoLSB,al;

 dado = (dadoMSB << 8) + dadoLSB;   /* Dado em decimal. */

 y = 10.0 * dado / 4095.0 - 5.0;          /* Valor em tensão. */

tv[k] = t;
yv[k] = y;

e = r-y;                    /* Erro da Malha. */
p = kp * e;                 /* Parte Proporcional. */
i = i + ki * e * te;        /* Parte Integral. */
d = (kd*e – Id)/(0.1*kd);  /* Parte Derivativa. */
Id = Id + d*te;
u = p+i+d;                  /* Lei de Controle. */

 if ( u < 0.0 ) u = 0.0;        /* Limites. */
```

```
    if ( u > 5.0 ) u = 5.0;

    dado = 4095.0 * u / 5.0;            /* Valor de tensão em decimal inteiro. */

    dadoMSB = dado >> 8;                /* Deslocar oito vezes para a direita. */
    dadoMSB = dadoMSB & 15;             /* Pegar apenas os primeiros 4 bits. */
    dadoLSB = dado & 255;               /* Pegar os 8 bits da esquerda. */

    EndIO=EndBase+4;
    asm mov al,dadoLSB;
    asm mov dx,EndIO;
    asm out dx,al;                      /* Escrever nos registros do D/A. */

    EndIO=EndBase+5;
    asm mov al,dadoMSB;
    asm mov dx,EndIO;
    asm out dx,al;                      /* Escrever nos registros do D/A. */

   asm mov al,128;                      /* T/C2 modo leitura. */
   asm out 43h,al;
   asm in al,42h;                       /* Ler parte LSB. */
   asm mov x1,al;
   asm in al,42h;                       /* Ler parte MSB. */
   asm mov x2,al;

    JI=256*x2+x1;                       /* Converte em 16 bits. */
    if (JI < 0) te=65536.0+JI;
        else te=JI;                     /* Testa se positivo. */
    te=65535.0-te;
    te=te/1.19318e6;                    /* Tempo de varredura. */

  t = t + te;

} while (!kbhit());
```

```
  asm in al,61h;                    /* Lê status do Gate2 do Timer do micro. */
  asm and al,0FEh;                 /* Mascara bits. */
  asm out 61h,al;                   /* Inibe Gate2. */

  asm sti;                     /* Habilita interrupção. */

}


void __fastcall TForm1::NMUDP1DataReceived(TComponent *Sender,
     int NumberBytes, AnsiString FromIP, int Port)
{
   char buffer[50], temp[6];
   int n,x,k;
   float kp,ki,kd,r;

   NMUDP1->RemoteHost = FromIP;
   NMUDP1->RemotePort = Port;

   NMUDP1->ReadBuffer(buffer,50,n);
   buffer[n]=0;

   if (!local)
   {
    Label1->Caption = "STATUS: Recebendo ...";              // Status do servidor //
    for (x=0;x<=4;x++) temp[x]=buffer[x];
    temp[5]=0;
    kp = StrToFloat(temp);

    for (x=5;x<=9;x++) temp[x-5]=buffer[x];
    temp[5]=0;
    ki = StrToFloat(temp);
```

```
    for (x=10;x<=14;x++) temp[x-10]=buffer[x];
    temp[5]=0;
    kd = StrToFloat(temp);

    for (x=15;x<=19;x++) temp[x-15]=buffer[x];
    temp[5]=0;
    r = StrToFloat(temp);

    for (x=20;x<=24;x++) temp[x-20]=buffer[x];
    temp[5]=0;
    k = StrToInt(temp);

    KP->Text = FormatFloat("0.000",kp);
    KI->Text = FormatFloat("0.000",ki);
    KD->Text = FormatFloat("0.000",kd);
    R->Text = FormatFloat("0.000",r);
    K->Text = IntToStr(k);

    calculo(kp, ki, kd, r, k);              // função para calcular o PID

    NMUDP1->SendBuffer(buffer,n+1,n+1);
    Button2->Enabled = true;
    Label1->Caption = "STATUS: executando remoto ...";  // Status do servidor //
   }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  KP->Visible = false;
  KI->Visible = false;
  KD->Visible = false;
  R->Visible = false;
  K->Visible = false;
  T->Visible = false;
```

```
  Y->Visible = false;
  Button2->Enabled = false;
  local = true;
  Label1->Caption = "STATUS: executando local ...";       // Status do servidor //
  NMUDP1->LocalPort = 5000;                               // Liberação da porta usada
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  local = true;
  Button2->Enabled = false;
  KP->Visible = true;
  KI->Visible = true;
  KD->Visible = true;
  R->Visible = true;
  K->Visible = true;
  T->Visible = true;
  Y->Visible = true;
  KP->SetFocus();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
  for (int i=1; i<=1000; i++)
     Series1->AddXY(tv[i],yv[i],"",clRed);            // Valores dos eixos p/ montagem do
                                                        Gráfico de colunas
}
//---------------------------------------------------------------------------

void __fastcall TForm1::KExit(TObject *Sender)
{
  calculo(StrToFloat(KP->Text),StrToFloat(KI->Text), StrToFloat(KD->Text),
```

```
        StrToFloat(R->Text), StrToInt(K->Text));
  Button2->Enabled = true;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
  local = false;
  Label1->Caption = "STATUS: excutando remoto ...";  // Status do servidor //
  KP->Visible = true;            // libera a variável  KP
  KI->Visible = true;            // libera a variável   KI
  KD->Visible = true;            // libera a variável   KD
  R->Visible = true;             // libera a variável   R
  K->Visible = true;             // libera a variável   K
  T->Visible = true;             // libera a variável   T
  Y->Visible = true;             // libera a variável   Y
}
```

ANEXO II

```
/*********************************************************************/
/*                Programa de Captura de Imagem  - Módulo Cliente     */
/*                                Delphi  7                           */
/*                          Helenice Cristina Ferreira                */
/*                                                                    */
/*********************************************************************/

unit CAMERA1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, OleCtrls, SHDocVw, ExtCtrls;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Button1: TButton;
    WebBrowser: TWebBrowser;   // Acesso ao Navegador
    Endereco: TEdit;                  // Endereço IP do Servidor de Imagem
    Label1: TLabel;
    Button2: TButton;
    CheckBox1: TCheckBox;        // Botão de Tempo Real
    Timer1: TTimer;                 // Tempo do microcomputador
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
```

```
 private
   { Private declarations }
 public
   { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);

begin
 WebBrowser.Navigate( Endereco.Text );   // Acesso ao Navegador através do Parâmetro do
                                            endereço IP do Servidor
end;

procedure TForm1.Button2Click(Sender: TObject);

begin
 Close;
end;

procedure TForm1.Timer1Timer(Sender: TObject);   // Rotina que atualiza a imagem no
                                                    navegador
begin
 if checkbox1.Checked then                 // Botão de tempo real ativado
   button1click(sender);                   //  Exibe a imagem em tempo real
end;

end.
```

```
/*************************************************************************/
/*                  Programa de Captura de Imagem - Módulo Servidor      */
/*                              Borland C++ Builder 6,                   */
/*                          Helenice Cristina Ferreira                   */
/*                                                                       */
/*************************************************************************/

#include <vcl.h>
#pragma hdrstop

#include "webcam.h"
//---------------------------------------------------------------------------
// Execução externa
// Através da diretiva #pragma, o C++Builder pode definir diretivas que podem ser passadas ao compilador. Se o compilador não identificar o nome da diretiva, ele simplesmente ignora #pragma sem emitir qualquer aviso ou mensagem de ERRO. A sintaxe de pragma é #pragma nome_da_diretiva.

#pragma package(smart_init)
#pragma link "VLCommonDisplay"
#pragma link "VLDSCapture"
#pragma link "VLDSImageDisplay"
#pragma link "VLDSVideoLogger"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
```

```
/*************************************************************************/
/*                          Transmissão de Imagens                       */
/*                          Módulo Servidor                              */
/*                          Borland C++ Builder 6,                       */
/*                          Helenice Cristina Ferreira                   */
/*                                                                       */
/*************************************************************************/
#ifndef webcamH
#define webcamH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include "VLCommonDisplay.h"
#include "VLDSCapture.h"
#include "VLDSImageDisplay.h"
#include "VLDSVideoLogger.h"
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published    :    // Componentes de gerenciamento da Câmera
    TVLDSImageDisplay *VLDSImageDisplay1;
    TVLDSVideoLogger *VLDSVideoLogger1;
    TVLDSCapture *VLDSCapture1;
private:    // Declaração
public:
    __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

```
/*************************************************************************/
/*                                                                       */
/*                  Configurações da Webcam  - Módulo Servidor           */
/*                              Janeiro 2004                             */
/*                          Helenice Cristina Ferreira                   */
/*                                                                       */
/*************************************************************************/
object Form1: TForm1
  Left = 192
  Top = 107
  Width = 696
  Height = 480
  Caption = 'Form1'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object VLDSImageDisplay1: TVLDSImageDisplay        // Formato da Imagem
    Left = 96
    Top = 88
    Width = 353
    Height = 297
    Align = alCustom
    InputPin.SourcePin = VLDSCapture1.OutputPin          // Forma de captura da Imagem
  end
  object VLDSVideoLogger1: TVLDSVideoLogger
    FileName = 'C:\videolab.avi'
    VideoCompression.Enabled = False        // Compressão de Vídeo desativada
    VideoCompression.Compressions = <
```

```
    item
      KeyFrameRate = 15
      WindowSize = 10
      PFramesPerKeyFrame = 0
      Data = (
        '@device:cm:{33D9A760-90C8-11D0-BD43-00A0C911CE86}\cvid'
        'Cinepak Codec by Radius')
    end>
  AudioCompression.Enabled = False
  AudioCompression.Compressions = <
    item
      Data = (
        '@device:cm:{33D9A761-90C8-11D0-BD43-00A0C911CE86}\1PCM'
        '')
    end>
  Left = 480
  Top = 104
 end
 object VLDSCapture1: TVLDSCapture     //Captura de Imagens
  VideoPreview.Enabled = False         // Desabilita parâmetros de visualização
  VideoPreview.FullScreen = False      // Desabilita parâmetros de Tela Cheia
  AudioPreview.Enabled = False         // Desabilita parâmetros de Áudio
  VideoCaptureDevice.AlternativeDevices = <>
  VideoCaptureDevice.Data = (
   '@device:cm:{860BB310-5D01-11D0-BD3B-00A0C911CE86}\AIPTEK PenCam'
   'AIPTEK PenCam')                    //Configura a WebCam
  AudioCaptureDevice.AlternativeDevices = <>
  OutputPin.SinkPins = (
   VLDSImageDisplay1.InputPin)
  Left = 408
  Top = 96
 end
end
```

```
/*************************************************************************/
/*                    Transmissão/ Recepção de Imagens                   */
/*                      Configurações de Compilação                      */
/*                      Módulos Servidor e Cliente                       */
/*                            Janeiro 2004                               */
/*                      Helenice Cristina Ferreira                       */
/*                                                                       */
/*************************************************************************/
<?xml version='1.0' encoding='utf-8' ?>
<!-- C++Builder XML Project -->
<PROJECT>
  <MACROS>
    <VERSION value="BCB.06.00"/>
    <PROJECT value="webcam2.exe"/>
    <OBJFILES value="webcam2.obj webcam.obj"/>
    <RESFILES value="webcam2.res"/>
    <IDLFILES value=""/>
    <IDLGENFILES value=""/>
    <DEFFILE value=""/>
    <RESDEPEN value="$(RESFILES) webcam.dfm"/>
    <LIBFILES value=""/>
    <LIBRARIES value=""/>
    <SPARELIBS value="vcl.lib rtl.lib VideoLabPkgCB6.lib OpenWirePkgCB6.lib
      SignalLabBasicPkgCB6.lib AudioLabBasicPkgCB6.lib"/>
    <PACKAGES value="vcl.bpi rtl.bpi dbrtl.bpi adortl.bpi vcldb.bpi vclx.bpi bdertl.bpi
      vcldbx.bpi ibxpress.bpi dsnap.bpi cds.bpi bdecds.bpi qrpt.bpi teeui.bpi
      teedb.bpi tee.bpi dss.bpi teeqr.bpi visualclx.bpi visualdbclx.bpi
      dsnapcrba.bpi dsnapcon.bpi bcbsmp.bpi vclie.bpi xmlrtl.bpi inet.bpi
      inetdbbde.bpi inetdbxpress.bpi inetdb.bpi nmfast.bpi webdsnap.bpi
      bcbie.bpi websnap.bpi soaprtl.bpi dclocx.bpi dbexpress.bpi dbxcds.bpi
      indy.bpi bcb2kaxserver.bpi OpenWirePkgCB6.bpi SignalLabBasicPkgCB6.bpi
      AudioLabBasicPkgCB6.bpi VideoLabPkgCB6.bpi SignalLabScopePkgCB6.bpi"/>
    <PATHCPP value=".;"/>
    <PATHPAS value=".;"/>
```

```
  <PATHRC value=".;"/>
  <PATHASM value=".;"/>
  <DEBUGLIBPATH value="$(BCB)\lib\debug"/>
  <RELEASELIBPATH value="$(BCB)\lib\release"/>
  <LINKER value="ilink32"/>
  <USERDEFINES value="_DEBUG"/>
  <SYSDEFINES value="_RTLDLL;NO_STRICT;USEPACKAGES"/>
  <MAINSOURCE value="webcam2.cpp"/>
  <INCLUDEPATH value="&quot;C:\Arquivos de
programas\Borland\CBuilder6\Projects&quot;;&quot;D:\Helenice\video
lab&quot;;$(BCB)\include;$(BCB)\include\vcl;$(BCB)\LabPacks;$(BCB)\OpenWire"/>
  <LIBPATH value="&quot;C:\Arquivos de
programas\Borland\CBuilder6\Projects&quot;;&quot;D:\Helenice\video
lab&quot;;$(BCB)\Projects\Lib;$(BCB)\lib\obj;$(BCB)\lib;$(BCB)\LabPacks"/>
  <WARNINGS value="-w-par"/>
  <OTHERFILES value=""/>
 </MACROS>
 <OPTIONS>
  <IDLCFLAGS value="-I&quot;C:\Arquivos de
programas\Borland\CBuilder6\Projects&quot;
    -I&quot;D:\Helenice\video lab&quot; -I$(BCB)\include -I$(BCB)\include\vcl
    -I$(BCB)\LabPacks -I$(BCB)\OpenWire -src_suffix cpp -D_DEBUG -boa"/>
  <CFLAG1 value="-Od -H=$(BCB)\lib\vcl60.csm -Hc -Vx -Ve -X- -r- -a8 -b- -k -y -v -vi- -
c
    -tW -tWM"/>
  <PFLAGS value="-$YD -$W -$O- -$A8 -v -JPHNE -M"/>
  <RFLAGS value=""/>
  <AFLAGS value="/mx /w2 /zd"/>
  <LFLAGS value="-D&quot;&quot; -aa -Tpe -x -Gn -v"/>
  <OTHERFILES value=""/>
 </OPTIONS>
 <LINKER>
  <ALLOBJ value="c0w32.obj $(PACKAGES) Memmgr.Lib sysinit.obj $(OBJFILES)"/>
  <ALLRES value="$(RESFILES)"/>
```

```
  <ALLLIB value="$(LIBFILES) $(LIBRARIES) import32.lib cp32mti.lib"/>
  <OTHERFILES value=""/>
 </LINKER>
 <FILELIST>
    <FILE FILENAME="webcam2.res" FORMNAME="" UNITNAME="webcam2.res"
CONTAINERID="ResTool" DESIGNCLASS="" LOCALCOMMAND=""/>
    <FILE FILENAME="webcam2.cpp" FORMNAME="" UNITNAME="webcam2"
CONTAINERID="CCompiler" DESIGNCLASS="" LOCALCOMMAND=""/>
    <FILE FILENAME="webcam.cpp" FORMNAME="Form1" UNITNAME="webcam"
CONTAINERID="CCompiler" DESIGNCLASS="" LOCALCOMMAND=""/>
 </FILELIST>
 <BUILDTOOLS>
 </BUILDTOOLS>

 <IDEOPTIONS>
[Version Info]
IncludeVerInfo=0
AutoIncBuild=0
MajorVer=1
MinorVer=0
Release=0
Build=0
Debug=0
PreRelease=0
Special=0
Private=0
DLL=0
Locale=1046
CodePage=1252

[Version Info Keys]
CompanyName=
FileDescription=
FileVersion=1.0.0.0
```

InternalName=
LegalCopyright=
LegalTrademarks=
OriginalFilename=
ProductName=
ProductVersion=1.0.0.0
Comments=

[Excluded Packages]
c:\arquivos de programas\borland\cbuilder6\Bin\dclite60.bpl=Borland Integrated Translation Environment

[HistoryLists\hlIncludePath]
Count=2
Item0=$(BCB)\include;$(BCB)\include\vcl;$(BCB)\LabPacks;$(BCB)\OpenWire
Item1=$(BCB)\include;$(BCB)\include\vcl

[HistoryLists\hlLibraryPath]
Count=1
Item0=$(BCB)\lib\obj;$(BCB)\lib

[HistoryLists\hlDebugSourcePath]
Count=1
Item0=$(BCB)\source\vcl

[HistoryLists\hlConditionals]
Count=1
Item0=_DEBUG

[Debugging]
DebugSourceDirs=$(BCB)\source\vcl

[Parameters]
RunParams=

```
Launcher=
UseLauncher=0
DebugCWD=
HostApplication=
RemoteHost=
RemotePath=
RemoteLauncher=
RemoteCWD=
RemoteDebug=0

[Compiler]
ShowInfoMsgs=0
LinkDebugVcl=0
LinkCGLIB=0

[CORBA]
AddServerUnit=1
AddClientUnit=1
PrecompiledHeaders=1
  </IDEOPTIONS>
</PROJECT>
```