UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

Integração de Ferramentas de Automação Direcionadas a Aplicações de Telerobótica - Implementação de um Sistema de Supervisão e Controle num Sistema Teleoperado

Autor: **Sérgio Guanaes Cosso**

Orientador: **João Maurício Rosário**

02/02

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETOS MECÂNICOS

Integração de Ferramentas de Automação Direcionadas a Aplicações de Telerobótica - Implementação de um Sistema de Supervisão e Controle num Sistema Teleoperado

Autor: **Sérgio Guanaes Cosso**
Orientador: **João Maurício Rosário**

Curso: Engenharia Mecânica
Área de Concentração: Departamento de Projeto Mecânico

Dissertação de mestrado acadêmico apresentado à comissão de Pós Graduação da Faculdade de Engenharia Mecânica, como requisito para a obtenção do título de Mestre em Engenharia Mecânica.

Campinas, 2002.
S.P. - Brasil.

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

C823i

Cosso, Sérgio Guanaes
Integração de ferramentas de automação direcionadas a aplicações de telerobótica – implementação de um sistema de supervisão e controle num sistema teleoperado / Sérgio Guanaes Cosso.--Campinas, SP: [s.n.], 2002.

Orientador: João Maurício Rosário.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Robótica. 2. Automação. 3. Internet (Redes de computação). I. Moreno Júnior, Armando Lopes. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETOS MECÂNICOS

DISSERTAÇÃO DE MESTRADO ACADÊMICO

Integração de Ferramentas de Automação Direcionadas a Aplicações de Telerobótica - Implementação de um Sistema de Supervisão e Controle num Sistema Teleoperado

Autor: **Sérgio Guanaes Cosso**
Orientadores: **João Maurício Rosário**

Prof. Dr. João Maurício Rosário - Presidente
Universidade Estadual de Campinas – UNICAMP

Profª. Drª. Kátia Lucchesi Cavalca Dedini
Universidade Estadual de Campinas – UNICAMP

Prof. Dr. Oswaldo Horikawa
Escola Politécnica da Universidade de São Paulo - USP

Campinas, 22 de fevereiro de 2002

Dedicatória

Dedico este trabalho:

À minha esposa Daniela pela sua compreensão e carinho, à minha filha Mariana, irmãos e à minha querida mãe.

Agradecimentos

Ao meu orientador, Professor.Dr. João Maurício Rosário, pelo apoio e confiança que teve comigo durante todo trabalho.

Aos amigos desta Instituição e funcionários pelo apoio e amizade.

Em especial a Professora Silvia Palma Sampaio Ciccu e aos amigos Almiro Franco da Silveira Junior e Marcelo Georgini.

Resumo

COSSO, Sérgio Guanaes, *Integração de Ferramentas de Automação Direcionadas a Aplicações de Telerobótica - Implementação de um Sistema de Supervisão e Controle num Sistema Teleoperado*, Campinas,: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2002. 134 p. Dissertação (Mestrado).

Este trabalho, dentro da área de conhecimento da Telerobótica, está fundamentado na implementação de um sistema de controle e supervisão à distância com a visualização do ambiente a ser controlado, no qual um usuário pode conectar-se ao sistema em qualquer parte do mundo, com a utilização da Internet como meio de comunicação. Nele são abordados os seguintes pontos: supervisão de sistemas, modo de funcionamento de sistemas supervisionados, modos de comunicação destes sistemas, problemas de controle a distância via internet, transmissão de imagem em redes de comunicação, etc. O objetivo é estudar a sistemática de implementação de sistemas supervisórios em redes de comunicação com a criação de um sistema de controle flexível em que exista uma margem de confiabilidade operacional junto aos sistemas a serem controlados e verificar as contribuições obtidas com a utilização destes sistemas. Os resultados da pesquisa foram: a partir do momento em que foram bem definidas as especificações relativas a arquitetura de supervisão e controle o problema se tornou genérico, o que possibilitou a integração de diferentes dispositivos automatizados com partes operativas diferentes. Quanto ao sistema de envio de imagens do ambiente controlado para o usuário, este se mostrou perfeitamente compatível para auxílio no controle de sistemas à distância.

Palavras Chave

Telerobótica, Teleoperação, Automação, SCADA, Controle Supervisório.

Abstract

COSSO, Sérgio Guanaes, *Integration of Automation Tools Managed to Telerobotic Applications Implementation of a Control Supervision System in a Teleoperated System*, Campinas,: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2002, 134 p. Dissertação (Mestrado).

This dissertation, in the area telerobotic, it is based on the implementation of a system of control and supervision, with the image received from the environment at the operational system, which the client can connect to anywhere through the of the internet. The following features are pointed out: supervision of systems, working mode of supervisory systems, their communication system, problems of distance control through the internet, image transmission in communication network. The objective is to study the systematic of implementation of supervisory systems in communication network, with the development of a flexive system where we can find a reliable operation on systems to be controlled and check the obtained contributions with the use of these systems. The research results were: from the moment when it was well defined all the specifications related to the architecture of supervision and control, the problem became generic which made it possible to integrate the different automatous devices. The system to image trasmission was adequate to this project.

Key Words

Telerobotics,Teleoperation, Automation, SCADA, Supervisory Control.

Sumário

Lista de Figuras

Lista de Tabelas

Nomenclatura

CGI – Common Gateway Interface

CLP – Controlador Lógico Programável

CP – Controladores Programáveis

CPU – Central Processing Unit

DNS – Domain Name System

E/S – Entrada/Saída

HTML – Hytertext Markup Language

HTTP – Hypertext Transfer Protocol

IHM – Interface Homem Máquina

I/O – Input/Output

IP – Internet Protocol

IPX – Internet Packet Exchange

LAN – Local Area Network Rede de Alcance Local

NetBEUI – Netbios Enhanced User Interface

PC – Personal Computer

PID – Proporcional Integral Derivativo

PLC – Programmable Logic Controller

RTP – Real Time Transport Protocol

RTU – Remote Terminal Units

SCADA – Supervisory Control and Data Acquisitions

SCM – Sistema Controlado e Monitorado

SPX – Sequenced Packet Exchange

TCP/IP – Transport Controle Protocol/Internet Protocol

URL – Uniform Resourse Locador

UWA – University of Western Austrália

WAN – Wide Area Network Rede de Alcance Remoto

WWW – World Wide Web

ISO – International Standards Organization

RIA – Associação das Industrias de Robótica

Capítulo 1

Introdução

A Revolução Industrial impulsionou e forneceu um novo marco na história humana, transformando o universo das organizações e a sociedade. A economia deixou de ter uma base artesanal e manufatureira para se firmar na produção industrial e mecanizada.

Desde a revolução industrial a interação homem-máquina tem sido um grande tópico de estudo.

O ser humano começa a viver uma nova era, na qual a competitividade industrial se torna um alto alvo das organizações mundiais, e também uma questão de sobrevivência.

Para aumentar a produtividade e qualidade, em âmbito geral, por volta dos anos 50 do século XX inicia-se a utilização de robôs industriais onde suas principais aplicações estão voltadas para o auxílio no trabalho do homem, em tarefas repetitivas, perigosas ou insalubres. Aliado à informática, a utilização de robôs visa a melhoria da produtividade e qualidade dos produtos e serviços.

Com a crescente automação e conseqüente aumento dos dispositivos eletro-eletrônicos e eletromecânicosdentre outros, surge a necessidade de interligação destes componentes, um monitoramento e controle destes dispositivos, propiciando ao sistema uma maior confiabilidade. A preocupação com a interação homem-máquina está voltada para uma relação de todo o sistema, como se o sistema industrial estivesse ligado a uma rede em que todos os dispositivos pudessem se comunicar. Para que a interação de todo sistema automatizado ocorresse

de forma eficiente e com isso propiciasse um efetivo controle e supervisão, surgiu a necessidade de evolução dos antigos métodos de supervisão, os quais ofereciam o monitoramento dos dispositivos de uma forma rudimentar, em que os equipamentos eram ligados a lâmpadas que apenas informavam o "status" atual através de sinais luminosos.

Nesta evolução tecnológica surge o primeiro Computador Pessoal (Personal Computer) desenvolvido pela IBM. Os computadores evoluíram e se tornaram fatores indispensáveis na implementação de sistemas automatizados.

Com a necessidade de troca de informações entre os computadores, há o surgimento das redes de computadores, dentro das quais é possível ter acesso a um dado que está fisicamente localizado distante, ou seja, em locais os mais remotos. Quanto à tecnologia de redes não podemos considerá-la nova, recente, pois ela existe desde a época dos primeiros computadores pessoais (PCs). Entretanto, novas padronizações e tecnologias permitiram que computadores pudessem se comunicar melhor a um custo menor.

Na corrida tecnológica outros dispositivos oferecem, pela sua própria evolução, uma grande contribuição para a sistemática da automação, dentre os quais podemos destacar os controladores lógicos programáveis, sensores, atuadores e robôs, sendo que estes últimos hoje se encontram com uma tecnologia mais aberta no sentido de propiciar uma fácil integração aos sistemas automatizados.

O impacto da automação tem sido de grande importância para os dias de hoje. Esta busca pela interação homem-máquina desenvolveu sistemas altamente sofisticados que podemos, por exemplo, encontrar em aviões, navios, controle de plantas industriais, redes de comunicação, avanços que contribuíram significativamente para a melhoria de vida do ser humano.

O grau de automação desenvolveu-se a tal ponto que, antigamente, nos sistemas que apresentavam um controle de malha fechada, em que existia uma supremacia manual do controle gerado pelo homem, passaram agora a um controle muito mais automatizado, resultando que o operador humano está se tornando somente um monitor e um supervisor da automação e, com isto, as possibilidades de falhas primárias diminuem de maneira acentuada, tornando os sistemas mais seguros.

A Internet é a grande rede mundial de computadores, que caminha em constante evolução desde a sua criação na década de 60.

É inegável que o desenvolvimento da Internet está causando uma grande mudança na vida e hábito das pessoas, e podemos comparar esta mudança social com a mesma ocorrida no período da Revolução Industrial.

Grandes fatores significativos foram introduzidos na comunicação via Internet desde sua criação, entre os quais podemos destacar o aumento da capacidade de envio e recebimento de dados, novas tecnologias para transmissão da comunicação e a introdução de novas linguagens de programação voltadas para o ambiente Web, em que se ressalta a linguagem de programação denominada Java.

A adição desta linguagem - Java - nas Redes de Computadores possibilitou a expansão de novas tecnologias com o uso na rede. Atualmente a linguagem Java está deixando de ser uma linguagem apenas voltada para a Internet e sendo utilizada como uma linguagem de programação "comum", comparada ao C++ ou Delphi, por exemplo. Os trabalhos de controle de robô à distância e os softwares de supervisão estão mais e mais integrados com a tecnologia da linguagem Java.

Com o desenvolvimento da tecnologia direcionada a aplicações via Web, o homem começa a se utilizar destas inovações tecnológicas para desenvolver sistemas voltados para a teleoperação com baixo custo, em que usa a Internet para alcançar estes objetivos. A teleoperação é uma grande área que está se destacando na era da informática, tanto na área da robótica conhecida como telerobótica, quanto no controle de satélites, e até mesmo na área médica com a realização de telecirurgias.

Esta dissertação aborda o contexto da teleoperação e descreve a integração de ferramentas de automação direcionadas às suas aplicações em Telerobótica via Internet, em que é implementado um sistema de supervisão para o controle de sistemas eletromecânicos que utilizam a tecnologia da linguagem de programação Java, sistema de supervisão, transmissão de imagens via Internet em tempo real e redes de comunicação.

1.1 **Objetivos do Trabalho**

Este trabalho tem como objetivo a implementação de Sistemas Supervisórios Industriais em aplicações direcionadas ao uso das tecnologias atuais voltadas à Internet e redes de comunicação, permitindo que metodologias e tecnologias sejam estudadas e sirvam de base para um enriquecimento tecnológico e bibliográfico, possibilitando o uso desta sistemática em outras aplicações. O sistema implementado deverá possuir grande flexibilidade e confiabilidade. Isto permitirá a integração de diferentes dispositivos automatizados junto ao sistema, ou seja, através da especificação funcional da parte de comando, diferentes dispositivos poderão ser integrados.

No que concerne a aspectos relacionados à confiabilidade e segurança do sistema, tratando-se de aplicações de teleoperação, a arquitetura de supervisão e controle será descentralizada, permitindo, no caso de ocorrer falhas na conexão via internet ou problemas associados ao sistema operacional do computador conectado ao dispositivo controlado, que os diferentes dispositivos automatizados integrados realizem suas tarefas em modo manual, independentes do sistema de comunicação.

O controle dos manipuladores robóticos e o magazine de montagem devem ser feitos a uma distância remota com auxílio de um navegador Web, no qual o usuário poderá obter e/ou enviar informações sobre tarefas a serem executadas pelos sistemas em questão. Para tornar o ambiente de controle mais "real" foi instalada uma câmera de captura de vídeo do tipo "Webcam" que possibilita a transmissão de imagens do sistema controlado em "tempo real". Sendo assim todo sistema pode ser visto, monitorado e comandado, tanto em uma rede Intranet, como também na rede mundial de computadores, a Internet.

1.2 **Delineamento do Trabalho**

Esta dissertação de mestrado está dividida na seguinte forma:

- Capítulo 1 – Introdução: são feitas considerações sobre a automação, seus benefícios, ferramentas que auxiliaram no processo de implementação de sistemas automatizados e uma apresentação geral do trabalho;

- Capítulo 2 – Revisão de Conceitos Utilizados em Teleoperação e Telerobótica: neste capítulo existe a descrição de sistemas de teleoperação e telerobótica, dos aspectos importantes que compõem estes sistemas, é feita uma revisão bibliográfica na área de telerobótica e alguns exemplos do uso da telerobótica nos dias atuais;

- Capítulo 3 – Sistemas de Supervisão e Controle em Automação: são apresentados conceitos gerais, evolução dos sistemas quanto ao grau de automação com o emprego de sistemáticas de supervisão, descrição dos principais componentes dos sistemas de supervisão e controle, descrição de alguns sistemas implementados em trabalhos utilizando softwares de supervisão e a descrição mais específica das características do software utilizado neste trabalho;

- Capítulo 4 – Principais Elementos de Integração Utilizados num Sistema Supervisório: são descritos os elementos que compõem um sistema de supervisão e controle e o tipo de arquitetura de comunicação utilizado para o funcionamento do sistema em uma rede Ethernet e, na seqüência, é feita a implementação da comunicação dos componentes;

- Capítulo 5 – Aplicações Utilizando Sistemas Supervisórios e Telerobótica: este capítulo descreve a implementação e modo de funcionamento de três dispositivos junto ao sistema de supervisão e controle via internet que são um magazine de montagem de peças por seleção de cores e dois manipuladores robóticos;

- Capítulo 6 – Conclusões e Perspectivas Futuras.

Capítulo 2

Revisão de Conceitos Utilizados em Teleoperação e Telerobótica

Para fornecer um maior conhecimento sobre teleoperação e telerobótica, assim como sistemas robóticos, este capítulo apresenta revisão e conceitos sobre os temas pertinentes aos assuntos em questão e relaciona trabalhos envolvendo esta área.

A teleoperação é definida como o controle contínuo e direto de um teleoperador.

Teleoperador é uma máquina que estende a sensibilidade e/ou capacidade de manipular da pessoa que o controla, de um local remoto. Um teleoperador necessariamente inclui sensores artificiais de ambiente e canais de comunicação para fornecer ou receber do operador (humano). Em adição, o teleoperador pode incluir braços e mãos artificiais ou outros dispositivos para aplicar forças e efetuar trabalhos mecânicos no ambiente.

A Telerobótica é uma forma avançada de teleoperação, é o procedimento pelo qual um operador supervisiona robô(s) através da intermediação de um computador, isto é, o operador localizado fora do ambiente operacional do robô e com a intermediação de um computador tem a possibilidade de controle e monitoramento dos dispositivos deste sistema, com o qual este operador pode interagir enviando tarefas a serem cumpridas. O robô controlado executa as tarefas baseado em informações recebidas do operador (humano) e tem a possibilidade de enviar sinais (dados) para informar ao sistema de controle os parâmetros necessários para obter um controle eficiente.

Convém notar que o termo "controle supervisório" é comumente usado para se referir à supervisão humana de algum sistema semi-autonômo (um avião, uma indústria química ou geração de energia, etc) e não leva em consideração a distância de separação entre o sistema e o operador(es). O termo "telerobótica" comumente se refere a um controle supervisório de um sistema que está distante do operador.

À medida que os sensores, atuadores e computadores dos telerobôs os tornam mais eficientes, juntamente com a determinação da sua capacidade de operação, a extensão com supervisor humano se tornará melhor.

A World Wide Web (WWW) foi planejada como uma distribuidora de informações de documentos técnicos armazenados em grandes sistemas de informações computacionais. Com a evolução da Internet, novas tecnologias foram surgindo e sendo aplicadas nas mais variadas situações. Diante desta evolução surge a utilização da Internet como meio de controle remoto de robôs. No item a seguir são mencionados alguns trabalhos relevantes na área de telerobótica.

2.1 Revisão Bibliográfica

Em setembro de 1994 pela primeira vez foi conectado um robô industrial denominado ASEA IRB-6 na Internet.

Esta conexão foi estabelecida através de um servidor WEB da Universidade do Oeste da Austrália (University of Western Austrália – UWA) de modo que todas as pessoas com acesso a WEB podiam receber informações textuais e gráficas do robô controlado. A garra deste dispositivo robótico era utilizada para manipular os blocos de madeira dispostos à mesa. Em Taylor (1995) a comunicação Internet-Robô está exemplificada na figura 2.1.

Somente quatro semanas depois, Ken Goldberg da Universidade da Califórnia, Berkeley, conectou um robô modelo Scara na Internet (Taylor 1995). Estes foram os primeiros dispositivos físicos conectados na WEB.

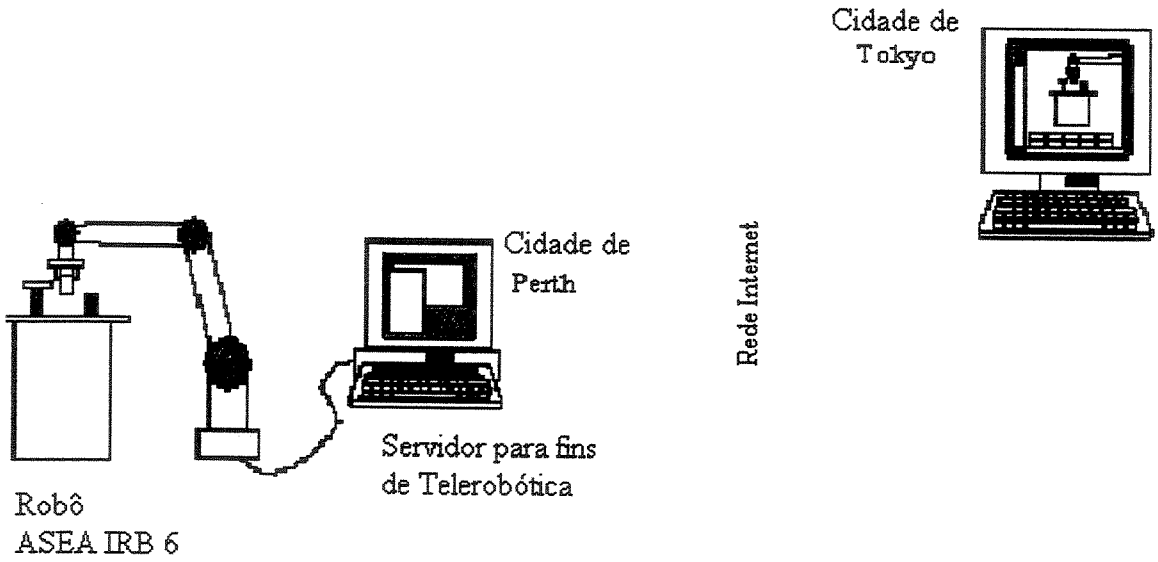


Figura 2.1- Controle Via Internet do Manipulador Robótico IRB-6
(Taylor, 1995)

O IRB-6 foi trocado em agosto de 1996 por um robô ABB IRB 1400. O software foi reescrito para trabalhar com o novo robô e os ganhos de conhecimentos provindos da primeira versão foram incorporados. Taylor (1997) faz uma análise sobre o novo sistema internet-robô utilizado, no qual faz um esquema de funcionamento (figura 2.2) de um robô conectado remotamente (via Internet) com uma interface CGI[1].

No sistema utilizado, o usuário do robô envia um pedido de movimento preenchendo um campo em HTML[2] criado na página Web ou clicando sobre a imagem que aparece no campo de trabalho. O navegador Web ou Browser (Netscape™ ou Microsoft® Internet Explorer) do usuário submete os detalhes do formulário como um pedido ao CGI (*Common Gateway Interface*) para um servidor Web, que recebe o pedido e lança um certificado do CGI para realizar o movimento requisitado. Uma vez terminado o movimento, novas imagens são enviadas para o ambiente de

[1] CGI (*Common Gateway Interface*) – também conhecido como Script CGI é um método utilizado na execução de programas da Web com base na entrada de um navegador da Web.
[2] HTML (HyperText Markup Language - Linguagem de Formação de Hipertexto) - é uma linguagem voltada para criação de páginas para internet, destinada a descrever a estrutura de um documento, e não a sua apresentação real.

trabalho e um novo formulário com a última imagem e posição do robô retorna via CGI para o usuário. Na página Web está criado um Applet[3] desenvolvido na linguagem de programação Java[4] operacional que mostra a configuração atual do robô, a qual permite que usuários simulem os movimentos antes de enviar os comandos ao robô.

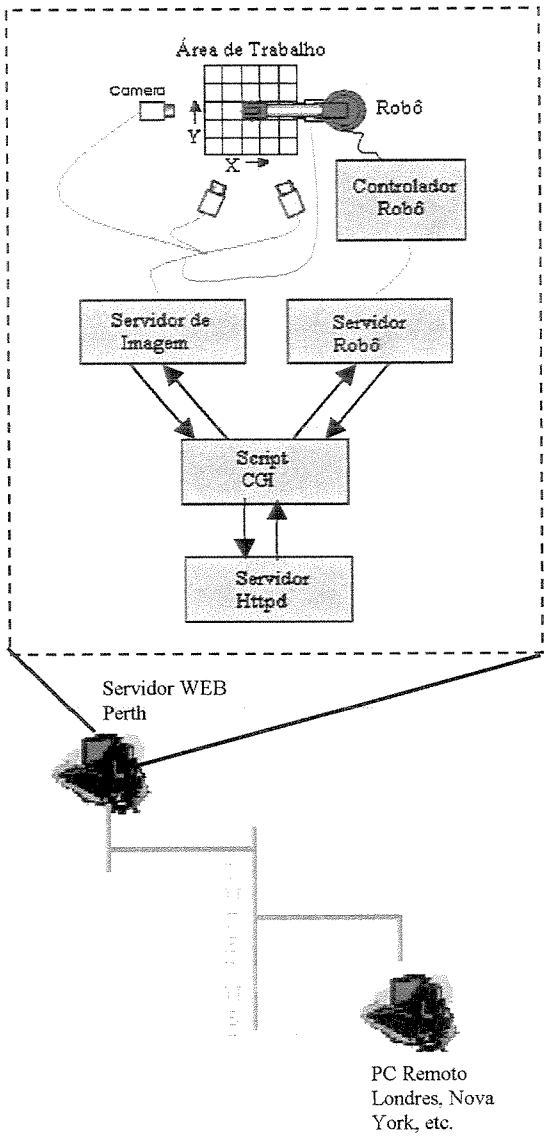


Figura 2.2 – Sistema de Telerobótica (Taylor,1997)

[3] Applets – são programas desenvolvidos com a linguagem Java que são executados em navegadores de internet (mais detalhes no anexo 1).

[4] Java é uma linguagem de programação muito conveniente para o desenvolvimento de software que funcione em conjunto com a Internet (mais detalhes anexo 1).

Em Otsuda (1996) é feita a avaliação de um Formulário/CGI ou como é mais conhecido, "script CGI". Este recurso é utilizado em inúmeros trabalhos de telerobótica. O "Formulário" (*Form*) é um elemento da linguagem HTML que permite ao usuário fornecer dados a uma aplicação de hiperdocumento, através do preenchimento de campos de um formulário. Esses dados são enviados para um servidor HTTP[5], no qual serão interpretados e processados por um programa CGI. Um CGI é colocado em um servidor WWW para realizar a interface deste com programas externos. Assim, o CGI atua como um "portão" (*Gateway*) entre o servidor WWW e bancos de dados, programas locais ou geradores de documentos. O funcionamento do CGI dá-se da seguinte forma - primeiramente o cliente WWW (navegador Web) solicita uma URL ao servidor WWW. A URL[6] solicitada é referente a uma CGI. Portanto, o servidor WWW executa o CGI solicitado, que trabalha interagindo com outras aplicações do sistema, recuperando dados destas aplicações e retornando o resultado ao servidor WWW. O servidor WWW, por sua vez, envia os dados para o cliente WWW, que formata o resultado e o apresenta ao usuário. A figura 2.3 ilustra o funcionamento do CGI.

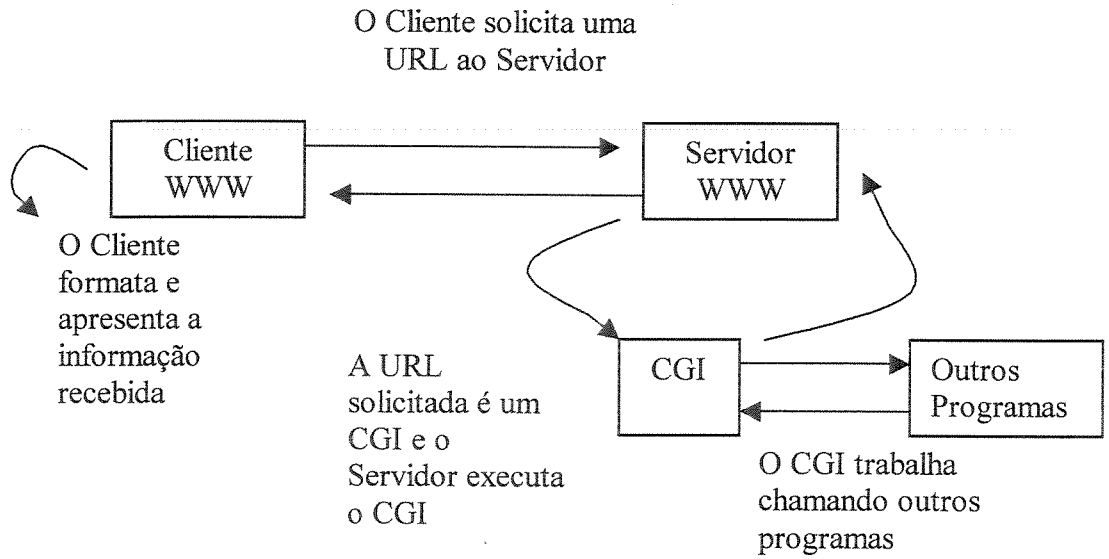


Figura 2.3 – Funcionamento do CGI (Otsuka,1996)

[5] HTTP (Hypertext Transfer Protocol) – protocolo de comunicação voltado para aplicações via internet.
[6] URL (Uniform Resource Locator) – é um ponteiro que indica uma informação específica disponível na internet.

Monteiro et al (1997) descreve uma estrutura computacional que pode "rodar" em todos os computadores conectados na base IP network, para estudo das técnicas de teleoperação. É discutida a teleoperação em um robô móvel através de um computador remoto.

No trabalho de Paulos (1997) foi criado um browser para ambientes remotos, chamado Mechanical Gaze, que permite a múltiplos usuários controlar ativamente um robô (manipulador) com seis graus de liberdade, com o objetivo da exploração de um ambiente remoto real. Seu ambiente inicial foi uma coleção de exibições de um museu, de onde os usuários podiam visualizar as peças em várias posições, orientações e níveis de resoluções.

Brady, et al (1998) apresenta em seu trabalho uma nova metodologia baseada no controle de robôs via internet, nos casos em que ocorre o atraso na comunicação.

Suzuki, et al (1999) descreve um sistema de interface para teleoperação de vários robôs usando o sistema WWW. Ele discute como um fluxo de comandos operacionais efetivamente enviados, através de uma interface com homem, pode comandar vários robôs na realização de tarefas de manutenção. Baseado nesta discussão, um sistema utilizando a tecnologia WWW é construído em uma Estação de Trabalho que possui um servidor WWW. Quando um cliente WWW especifica uma determinada tarefa para ser realizada pelo robô, o servidor WWW convoca um programa de interface CGI que faz com que esta interação homem-robô seja executada, e o operador (homem) pode acompanhar a execução desta tarefa à distância.

Em Dalton (2000) é feita uma abordagem sobre o uso do CGI, na qual a combinação do HTML com CGI possibilita execução de processos em ambientes remotos. Este é um dos métodos utilizados para controle do manipulador robótico da Universidade do Oeste da Austrália (UWA) e muitos outros sistemas Web telerobô, como o projeto Mercury de Goldberg (1995), dentre outros. O CGI produz um teor dinâmico a cada HTTP requerido. A combinação do CGI com HTML possui limitações. A cada requisição do usuário um novo processo deve ser iniciado o que causa latência entre esta requisição e a resposta do sistema. Apesar do HTML suportar algumas coisas simples, ele não é uma ferramenta de interface com usuário e os estilos de interações que podem ser suportados são muito limitados. Estas limitações podem ser superadas através da utilização da linguagem Java. A linguagem de programação Java capacita um cliente para ter controle "em cima" de uma conexão de rede de comunicação e ter uma completa

interface funcional com o usuário, mesmo com a utilização da tecnologia dos navegadores Web (os navegadores "rodam" programa Java como Applets, ver mais detalhes no anexo 1).

No trabalho de Tourino (2000) encontra-se a descrição de um robô móvel autônomo controlado remotamente via internet utilizando o sistema operacional Linux como plataforma de controle e comunicação. A locomoção do robô é feita por motores de passo controlados por um "driver' de potência e programas em linguagem C. O controle é utilizado através de uma interface Java/CGI (*Common Gateway Interface*) que envia os comandos do usuário para o sistema de navegação do robô, neste caso via rádio, tendo uma realimentação através de um sistema de captura de vídeo baseado em uma câmera do tipo Webcam.

Álvares (2000) apresenta o desenvolvimento de um sistema para controlar remotamente um robô industrial com seis graus de liberdade modelo IRB 2000 da Asea Brown Boveri (ABB) utilizando a Internet como via de controle. A arquitetura do sistema é do tipo Cliente/Servidor e será detalhada no capítulo 4. O sistema apresenta a seguinte sistemática: o controlador do robô ABB tem incorporado um sistema de controle remoto via interface serial RS 232 baseado em 42 funções. O protocolo de comunicação utilizado é o TCP/IP (*Transport Controle Protocol/ Internet Protocol*). O serviço foi disponibilizado via WWW, que pode ser acessado através de um navegador Web (Netscape ou Explorer), no qual o usuário pode remotamente enviar os programas ao robô teleoperado através de programas baseados no CGI e HTML. O usuário pode através de uma "janela" criada na página Web receber imagens do robô, permitindo a visualização deste.

2.2 Revisão dos Conceitos Aplicados a Telerobótica

2.2.1 Sistemas Automatizados

A telerobótica pode ser vista como uma grande sofisticação de sistema automatizados que tem sua capacidade de controle estendida através dos "links" de comunicação. Como foi visto a teleoperação requer uma maneira de interligação entre o homem e a máquina. Esta divisão nos sistemas automatizados pode ser de dois tipos:

- Parte operativa
- Parte de controle ou comando

A **parte operativa** é a responsável pela execução das operações físicas do sistema. Ela é composta pelos manipuladores robóticos, atuadores, sensores, bombas etc. No sistema desenvolvido neste trabalho a parte operativa é constituída pelos manipuladores robóticos (Robix e Robô ABB) e uma estação de montagem que atualmente está em fase de implementação junto à plataforma PIPEFA[7].

A **parte de controle** ou comando é a responsável pelo controle do sistema. Toda funcionalidade no lado de supervisão e controle e de troca de informações do sistema com o meio exterior são provenientes desta parte. De uma maneira geral, é o equipamento que recebe as informações provenientes do operador, ou do processo a ser controlado, emitindo ordens à parte Operativa. A figura 2.4 ilustra as partes dos sistemas automatizados.

[7] PIPEFA - Plataforma Industrial para Pesquisa, Ensino e Formação em Automação, ver mais detalhes no capítulo 5.

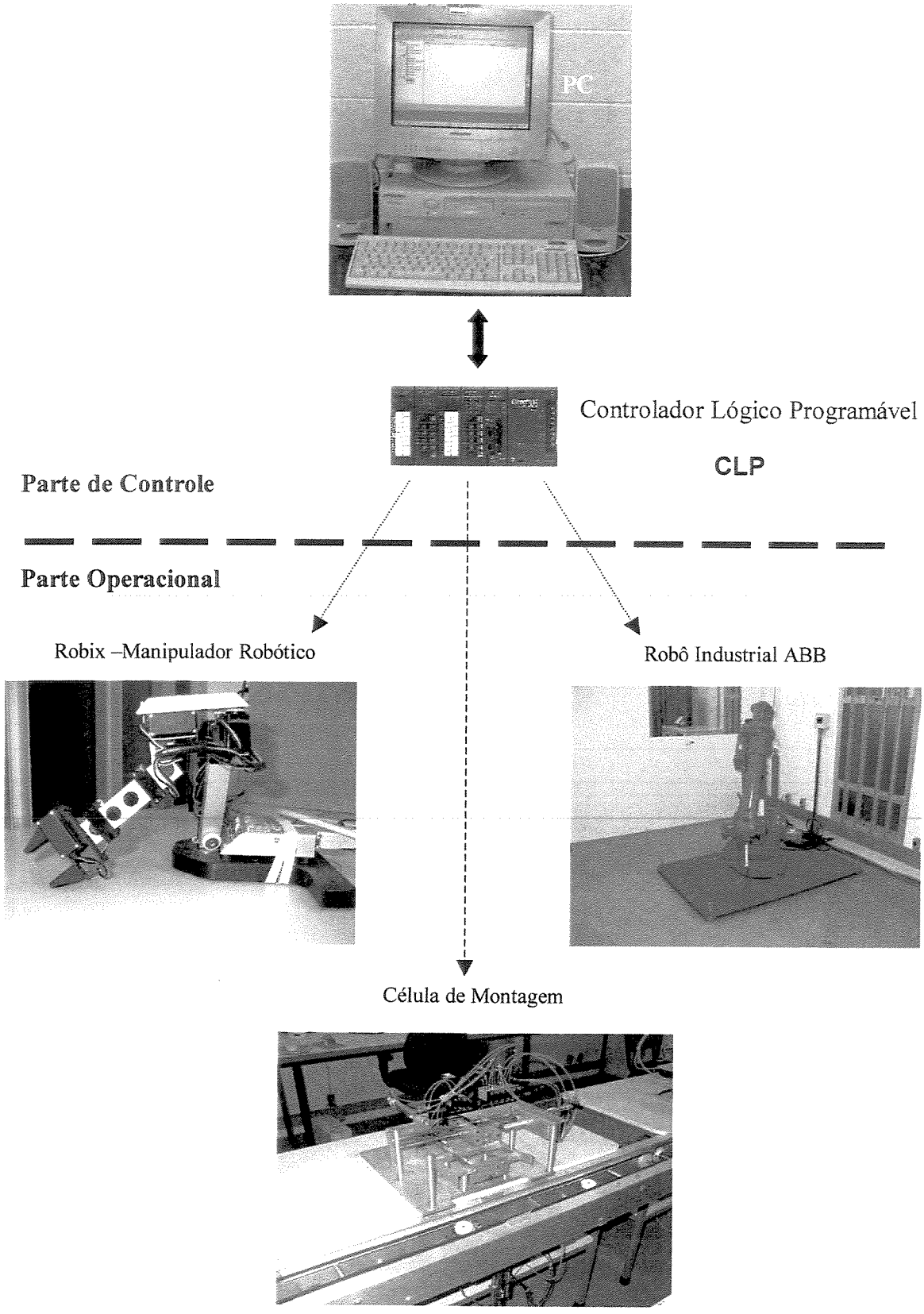


Figura 2.4 – Partes dos Sistemas Automatizados

2.2.2 **Sistemas Robóticos**

A robótica e automação são duas tecnologias intimamente relacionadas. Num contexto industrial podemos definir a automação como uma tecnologia que se ocupa do uso de sistemas mecânicos, eletrônicos e à base de computadores na operação e controle da produção. Como exemplo tem-se máquinas de montagens mecanizadas, sistemas de controle de realimentação, máquinas operatrizes dotadas de comandos numéricos e robôs (Groover, 1988).

A definição de um robô industrial dada pela Associação das Indústrias de Robótica (RIA) é a seguinte:

"Um robô industrial é um manipulador reprogramável, multifuncional, projetado para mover materiais, peças, ferramentas ou dispositivos especiais em movimentos variáveis programados para a realização de uma variedade de tarefas".

Os robôs, graças ao seu sistema lógico ou informático, podem ser reprogramados e utilizados em uma grande variedade de tarefas, sendo que a reprogramação não é o fator mais importante na versatilidade desejada, e sim a adaptação às variações no seu ambiente de trabalho, mediante um sistema adequado de percepção e tratamento de informação.

Os manipuladores possuem um número de articulações para atingir os movimentos necessários para a realização das tarefas. O número de articulações em um braço do robô é também referenciado como grau de liberdade. Quando o movimento relativo ocorre em um único eixo, a articulação tem 1 grau de liberdade. Quando o movimento é por mais de um eixo, as articulações têm 2 graus de liberdade. A maioria dos robôs tem entre 4 a 6 graus de liberdade; a figura 2.5 exemplifica um manipulador robótico com 5 graus de liberdade.

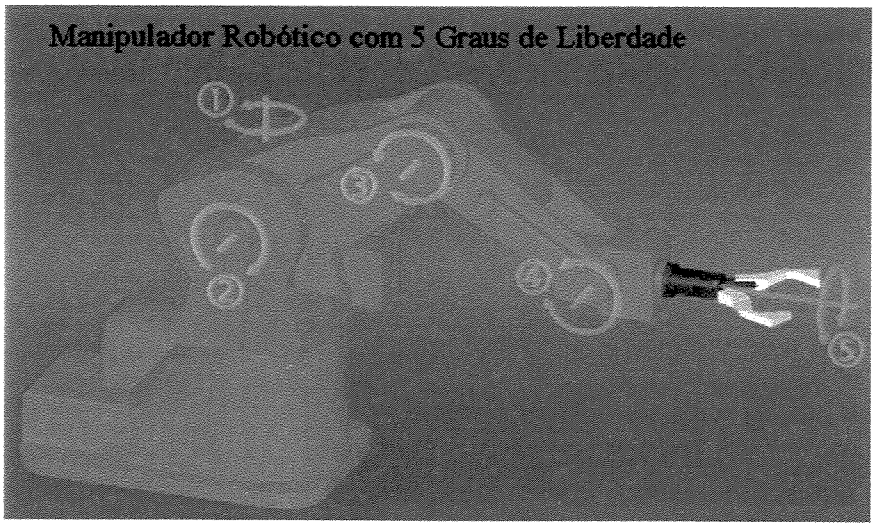


Figura 2.5 – Exemplo de um Manipulador Robótico com 5 Graus de Liberdade

2.2.3 **Modo de Operação dos Sistemas Robóticos**

2.2.3.a Modo Master-Slave – tradicionalmente utilizado em operações de intervenção, nos sistemas mais atuais foi acrescido da visualização, pelo computador, do manipulador no espaço de tarefas conforme mostrado na figura 2.6. Este dispositivo é constituído por um simulador físico que tem a geometria e os sensores idênticos ao robô original ("Master") que permite a aprendizagem de trajetórias a partir do movimento das diferentes juntas do robô original, ("Slave") favorecendo a gravação de pontos intermediários de uma trajetória.

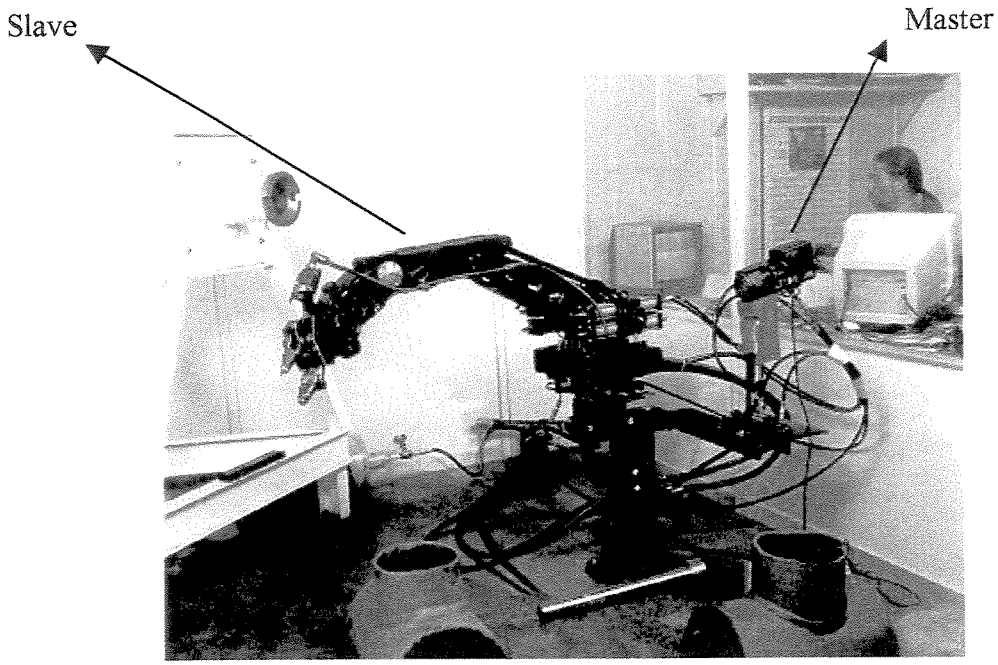


Figura 2.6 - Sistema de Controle - Master-slave - Manipulador Kraft-Grips

2.2.3.b Modo Microcomputador (figura 2.7) – permite o comando do manipulador através de teclas dedicadas do computador. Neste modo, é possível o movimento individual das juntas (modo angular) ou o movimento em direções (modo cartesiano). Este módulo é muito utilizado para o controle com envio de tarefas pré-estabelecidas.

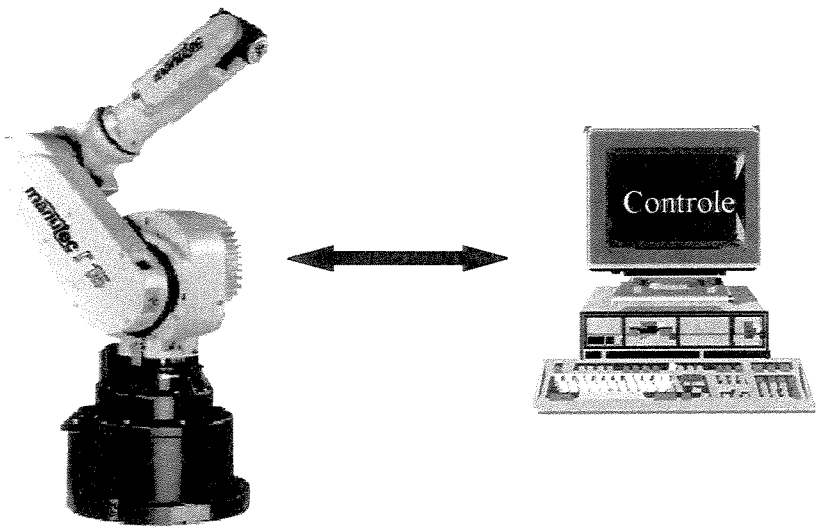


Figura 2.7 – Sistema de Controle por Microcomputador – Manutec

2.2.3.c Modo Mouse Espacial – segundo Nogueira (2000) mouse espacial é um sistema que gera sinais de deslocamentos em coordenadas cartesianas no espaço tridimensional enviados a um microcomputador, a fim de controlar a posição e orientação do manipulador.

2.3 Exemplos de Aplicações em Telerobótica

A Telerobótica tem avançado em inúmeras áreas atualmente como na exploração espacial com auxílio de robôs móveis (ver figura 2.8) através de um "link" de comunicação via satélite; no controle de manipuladores robóticos (ver figura 2.9) à distância e, também, na medicina ou Telemedicina, mais especificamente na área da Telecirurgia (ver figura 2.10) vem sendo utilizada esta tecnologia para realização de cirurgias. É possível hoje, com a evolução dessa técnica, acessar locais os mais remotos e realizar cirurugias de alta precisão.

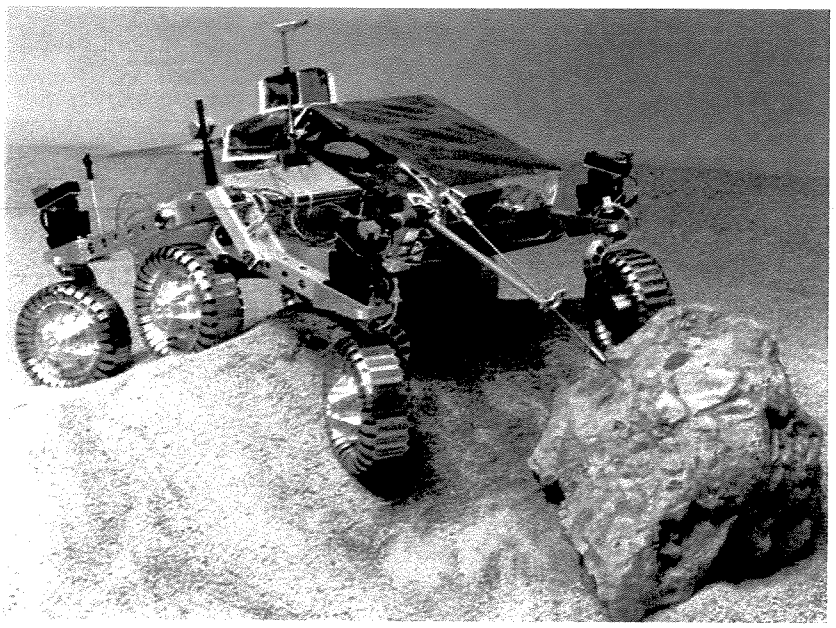


Figura 2.8 – Telerobótica Aplicada em Robótica Móvel (RockyIV – Nasa)

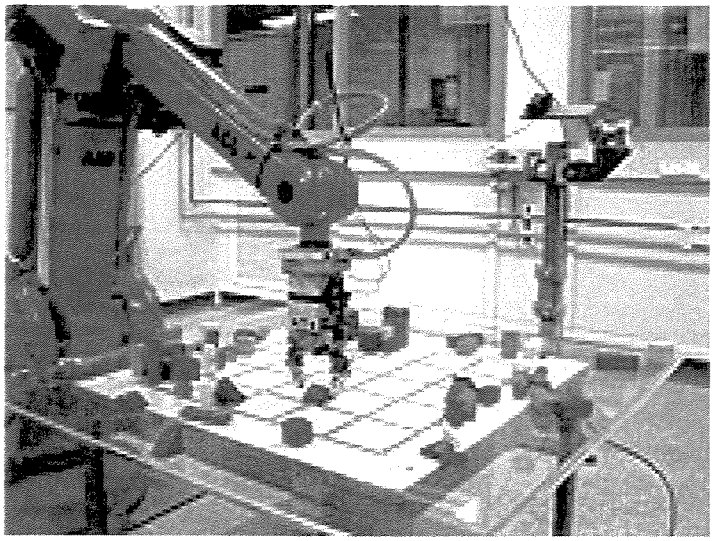


Figura 2.9 – Telerobótica Aplicada em Manipuladores Robóticos (Sistema AWU)

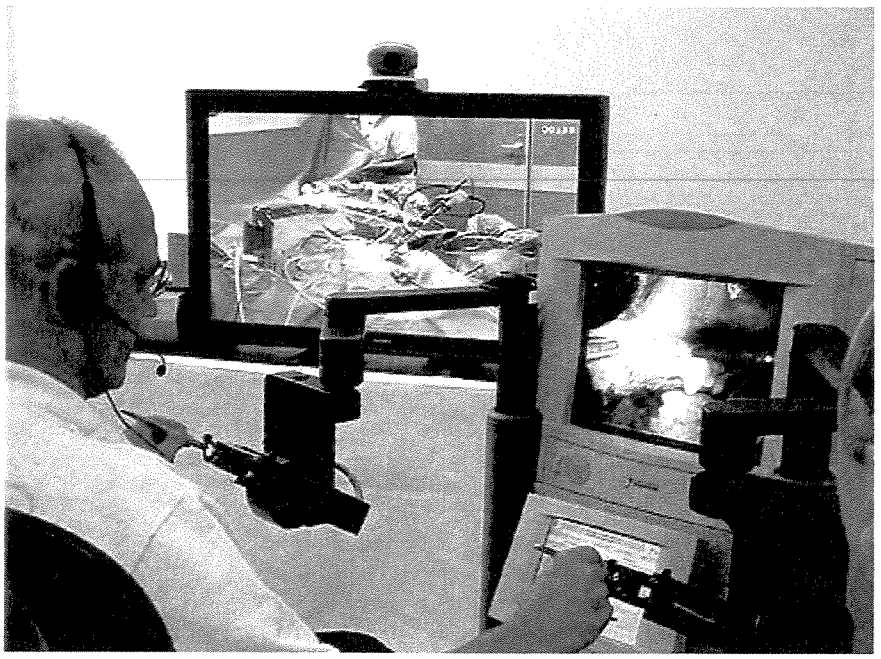


Figura 2.10 – Telerobótica Aplicada na Telemedicina (Sistema Master-Slave)

2.4 Considerações Finais

Como foi visto, os sistemas telerobóticos necessitam de trocas de dados entre os ambientes, de controle e operacional. Surge, então, a necessidade de uma conexão do dispositivo a ser controlado com um sistema que possibilite o envio e/ou recebimento de dados com maior rapidez e segurança.

Nota-se que na telerobótica o recebimento de imagens pelo controlador do sistema (homem) tem sido muito usado e quanto maior for a sofisticação destes sistemas de transmissão de imagem com a utilização da Internet, novas aplicações poderão ser desenvolvidas com o emprego desta tecnologia.

Segundo Sheridan (1993), alguns fatores somados com o tempo de espera em controles feitos de maneira remota podem ocasionar um problema de instabilidade no sistema, o qual pode ser eliminado com a implementação de um controle supervisório.

Seguindo a proposta de Sheridan (1993) o sistema de controle e supervisão via Internet foi implementado com auxilio de um software de supervisão e controle, para que no caso de um problema de falha do sistema de controle, ou seja, no caso da comunicação do computador pessoal (PC) ficar impossibilitada, o operador pode obter o controle de forma manual com auxílio de um controlador lógico.

No capítulo a seguir serão feitas descrições gerais sobre sistemas de Supervisão e Controle e suas funcionalidades, com exemplos de algumas aplicações.

Capítulo 3

Sistemas de Supervisão e Controle em Automação

Segundo Aihara e Cosso (2001) um Sistema de Supervisão e Controle é responsável pelo monitoramento de variáveis de controle dos sistemas Automatizados, como também pela integração entre estes sistemas e os sistemas hierarquicamente superiores responsáveis por um gerenciamento mais global como, por exemplo, nas indústrias os Sistemas de Gerenciamento da Produção.

Atualmente os Sistemas Supervisórios podem ser definidos como uma interface homem-máquina (IHM) amigável, e possuem grandes recursos tecnológicos que permitem a supervisão e/ou o controle de sistemas automatizados.

Os primeiros sistemas SCADA (*Supervisory Control and Data Acquisition*), basicamente telemétricos, permitiam informar periodicamente o estado corrente do processo industrial, monitorando sinais representativos de medidas e estados de dispositivos, através de um painel de lâmpadas e indicadores sem que houvesse qualquer interface de aplicação com o operador.

Com a evolução tecnológica, os computadores assumiram um papel de gestão na aquisição e tratamento de dados, permitindo a sua visualização num "écran" e a geração de funções de controle complexas.

Atualmente os sistemas SCADA utilizam tecnologias de computação e comunicação para automatizar o monitoramento e controle dos Sistemas Automatizados, efetuando o recolhimento dos dados em ambientes complexos, eventualmente dispersos geograficamente, e os

respectivos sistemas apresentam uma visualização de modo amigável para o utilizador, com recurso Interface Homem-Máquina (IHM) altamente sofisticado.

Estes sistemas revelam-se de crucial importância na estrutura de gestão de Sistemas, e, por isso, deixaram de ser vistos como mera ferramenta operacional ou de engenharia, e passaram a ser vistos como uma importante fonte de informação e controle.

Num ambiente industrial cada vez mais complexo e competitivo, os fatores relacionados com a disponibilidade e segurança da informação assumem elevada relevância tornando-se necessário garantir que a informação está disponível e segura quando necessária, independentemente da localização geográfica. Torna-se, portanto, necessário implementar mecanismos de acessibilidade, mecanismos de segurança e mecanismos de tolerância à falhas.

Os sistemas SCADA melhoram a eficiência do processo de monitoração e controle, disponibilizando em tempo útil o estado atual do sistema através de um conjunto de previsões gráficas e relatórios, de modo a permitir a tomada de decisões operacionais apropriadas, quer automaticamente, quer por iniciativa do próprio operador.

3.1 Características dos Sistemas Supervisórios

Dentre as principais características necessárias para atender os requisitos para um sistema de supervisão e controle destacam-se:

- Inferface amigável com o operador, ou seja, o sistema de supervisão tem que propiciar ao operador uma facilidade de visualização gráfica e operação do sistema;
- Geração automática de relatórios – Controle Estatístico do Sistema;
- Histórico de Tendências (acompanhamento das variáveis controladas);
- Facilidade para interação com outros aplicativos (software);
- Acesso automático a banco de dados;
- Acesso compartilhado e Remoto;

- Conexão em rede e através de "modem" ou rádio;

- Gerenciamento das condições de alarmes.

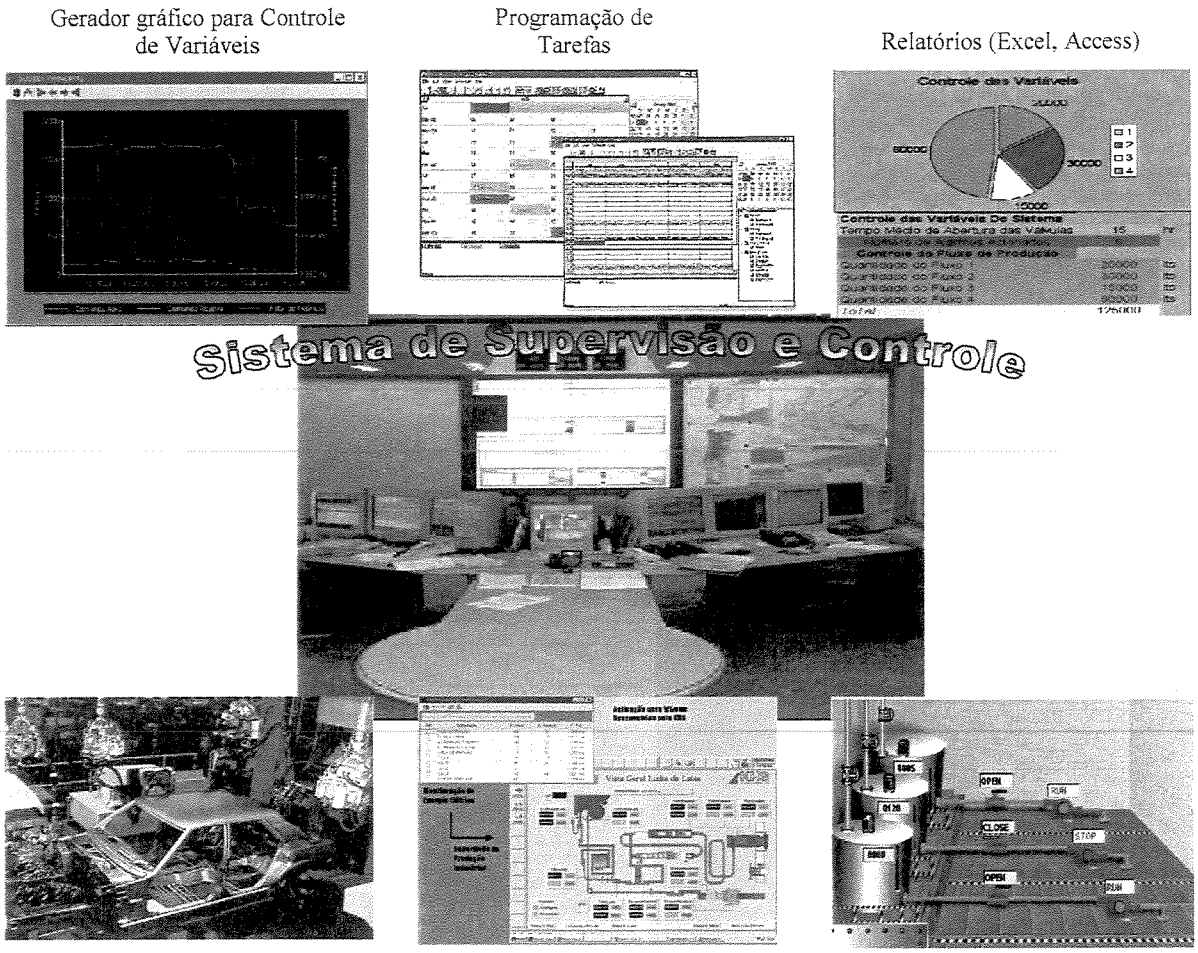


Figura 3.1 – Principais Características de um Sistema de Supervisório

Alguns Supervisórios existentes no Mercado: Wizcon [1], Ifix [2], Intouch [3], Elipse [4], Cimplicity [5], dentre outros.

Os sistemas SCADA cobrem um mercado cada vez mais vasto, podendo ser encontrados em diversas áreas tais como indústria de celulose, petrolíferas, hidroelétricas, têxtil, metalúrgica,

automotiva, eletrônica entre outras. Os sistemas de Supervisão e Controle estão, cada vez mais, sendo empregados em automações prediais, área esta conhecida como Domótica. Na figura 3.2, abaixo, tem-se uma visão de um hospital totalmente automatizado em que são empregadas as técnicas de controle e supervisão nas quais são utilizadas como suporte à Telemedicina (Telecirurgia e Telediagnóstico. Esse assunto será tratado no capítulo 5.

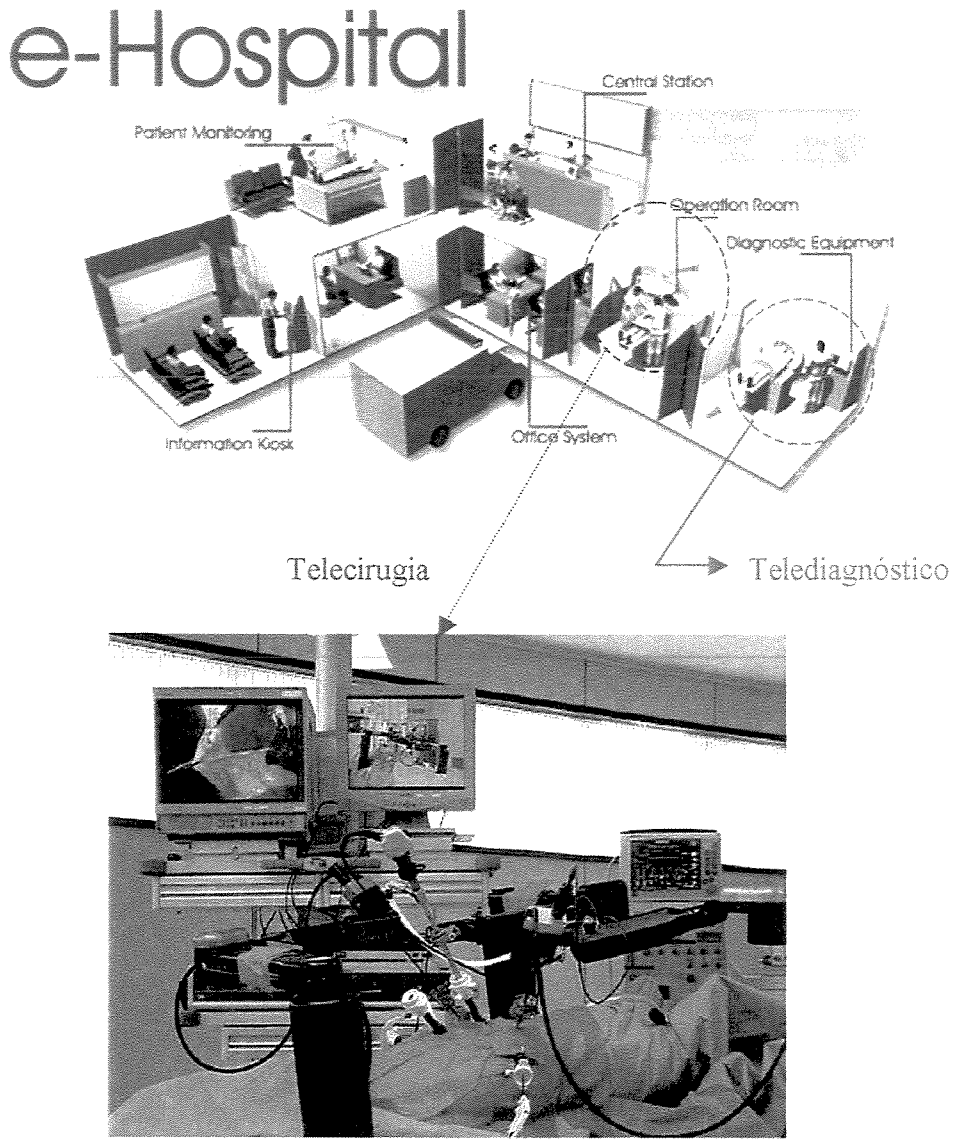


Figura 3.2 – Hospital Totalmente Automatizado e Integrado

3.2 Analogia entre Controle Supervisório Humano e Controle Supervisório

Controle supervisório significa a mesma coisa que controle supervisório humano, o termo "homem" está subentendido.

O termo é derivado da analogia fechada entre a interação do superior com membros subordinados em uma organização humana e a interação destas pessoas com subsistemas "inteligentes" automatizados. O supervisor do grupo dá diretivas que são entendidas e traduzidas dentro de ações detalhadas pelas pessoas subordinadas. Em turnos, subordinados coletam informações detalhadas sobre resultados e apresentam estes resultados de forma simplificada para o supervisor, que deve inferir o estado do sistema e tomar decisões para uma melhor ação. A inteligência dos subordinados determina o grau de complexidade de envovimento do seu supervisor no processo. Automação de subsistemas semi-inteligentes permite que a mesma espécie de interação ocorra entre o supervisor humano e o processo mediado por computador (Ferrell 1967, Sheridan 1984).

No "*strictest sense*" o controle supervisório significa que um ou mais operadores humanos estão intermitentemente programando e continuamente recebendo informações provindas do computador que fecha um "loop" de controle através de atuadores artificiais e sensores para o processo controlado ou tarefa envolvida.

Em "*less strict sense*" o controle supervisório significa que um ou mais operadores (humanos) estão continuamente programando e recebendo informações provindas de um computador o qual está interconectado com atuadores artificiais e sensores para o processo controlado ou tarefa envolvida.

Em ambas as definições, o computador transforma informações do homem para o processo controlado e do processo controlado para o homem, mas somente segundo a definição "*strict*" o computador faz necessariamente um "loop" fechado de controle que exclui o homem, fazendo desta maneira o computador um controlador autônomo para algumas variáveis, pelo menos por algum espaço de tempo. O acompanhamento da figura caracteriza o controle supervisório em relação aos extremos do controle manual ao controle totalmente automático. Comum para os

cinco diagramas homem-máquina são mostradores e interfaces de controle com o operador (homem), e sensores e atuadores com um processo controlado ou ambiente de tarefa.

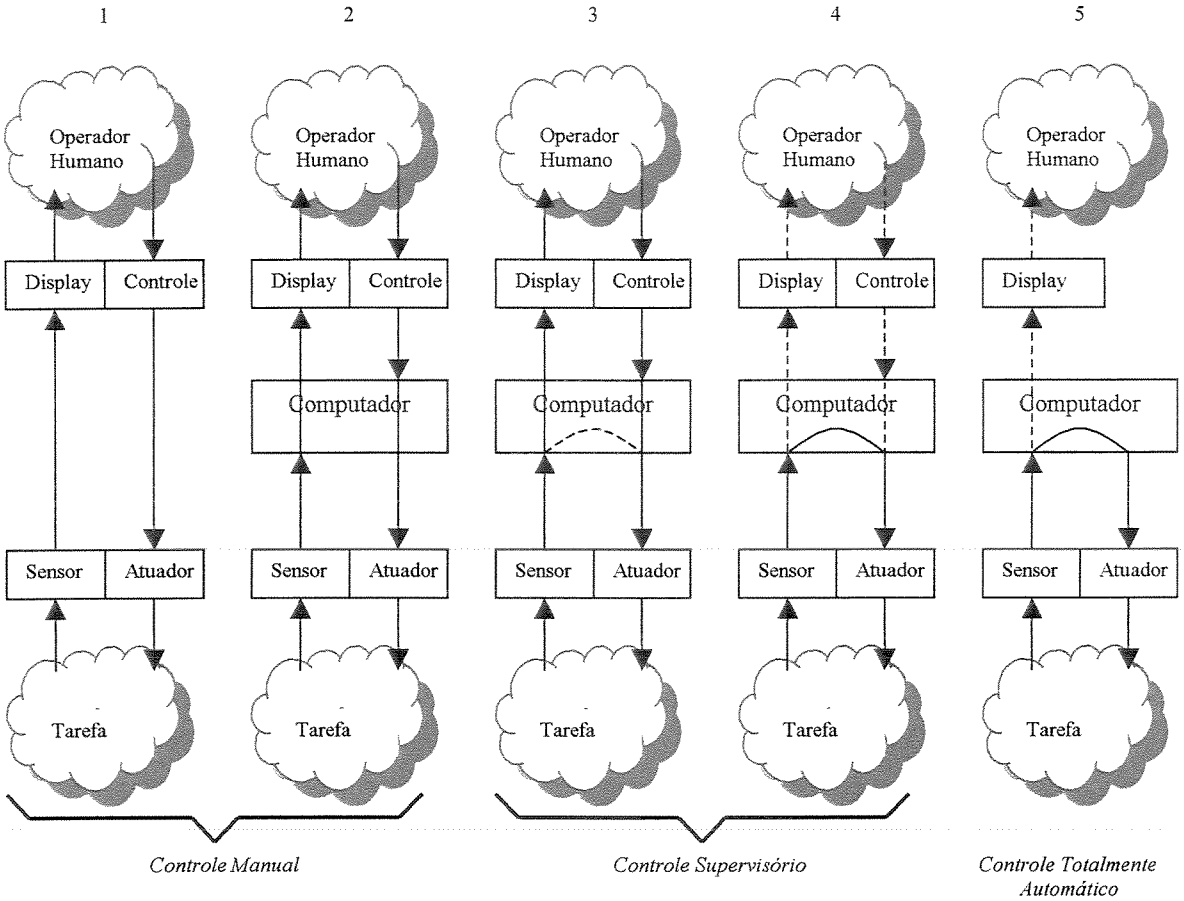


Figura 3.3 – "Spectrun" de Modelos de Controle (As linhas tracejadas indicam que os "loops" menores são fechados através do computador e os "loops" maiores através do homem. As linhas contínuas indicam que os "loops" maiores são fechados através do computador e os "loops" menores através do homem).

O sistema ❶ representa um controle manual convencional (nenhum computador adicionado). No sistema ❷ é introduzida significante mudança com a adição ou transformação com auxilio do computador, em cada ou em ambos os lados, ou seja, na aquisição de dados provindos dos sensores, como no lado dos atuadores. Isto corresponde a menor definição de controle supervisório. Note, de qualquer modo, que em ambos os sistemas (❶ e ❷) todas as decisões de controle dependem do operador humano. Se o homem pára, o controle pára.

Quando a fração do controle de um menor (sistema ❸) ou um maior (sistema ❹) está acompanhado diretamente pelo controle de um "loop" fechado com o auxílio de um computador e exclusivo do homem, este é o controle supervisório no "strict sense". Uma vez que o sistema de controle está montado, essencialmente todo o controle é automático (sistema ❺), isto é, o operador (humano) pode observar mas não pode influenciar no processo (outra maneira de sistemática). O sistema não está distante de um controle supervisório.

Os diagramas estão ordenados de acordo com o respectivo grau de automação.

As formas estritas (sistemas ❸ e ❹) e não estritas (sistema ❷) de controle supervisório podem parecer as mesmas para o supervisor, desde que ele sempre veja e atue através do computador e, portanto pode não saber se o computador está atuando em um "loop" aberto ou fechado, maneira em que se teria um melhor comportamento do sistema.

O computador pode funcionar predominantemente sobre o transporte de informação ou, do lado do próprio controle, para implementar o comando do supervisório (geralmente faz algumas partes da totalidade e deixa outras partes para o homem, ou fornece alguns controles de compensação para facilitar o controle das tarefas pelo homem). O computador pode também funcionar predominantemente sobre o lado de monitoramento (para integrar e interpretar a entrada de informação provinda de "baixo", ou para servir como um "sistema esperto" instruindo o supervisor sobre o que fazer na próxima etapa). Geralmente ele faz alguma de cada.

Dentro da forma "*strict*" de controle supervisório (sistema ❸ e ❹), o operador humano (supervisor) programa especificando para o computador metas, objetivos, "tradeoff", restrições físicas, modelos e procedimentos lógicos "se-então-faça" (*if-then-else*). Esta especificação é usual e convenientemente colocada em um nível mais elevado de linguagem "natural" – em termos desejados para trocas no processo ou sistema controlado sendo mais rápido do que em termos de controle de sinais.

Uma vez o supervisor "rodando" o controle no computador, o computador executa o programa armazenado e atua sobre novas informações provindas de sensores, independentemente do homem, ao menos por um período curto de tempo. O homem pode permanecer como um supervisor ou pode de tempo em tempo assumir diretamente o controle (isto é chamado de

"*traded control*") ou pode influenciar como supervisor com o respectivo controle de algumas variáveis e controlar diretamente com auxílio de outras variáveis ("*shared control*").

3.3 **Revisão Bibliográfica de Sistemas de Supervisão e Controle.**

Em Fayan (dez 1992) são feitos estudos para especificação de um Supervisor de Controle para um robô industrial; é descrita uma estrutura completa do supervisor de controle que está sendo desenvolvido (figura 3.4) que, a partir de uma interface homem-máquina amigável, por um software de visualização do robô e de seu ambiente de trabalho possibilita a geração de arquivos de trajetórias que posteriormente são transmitidos à estrutura de controle propriamente dita.

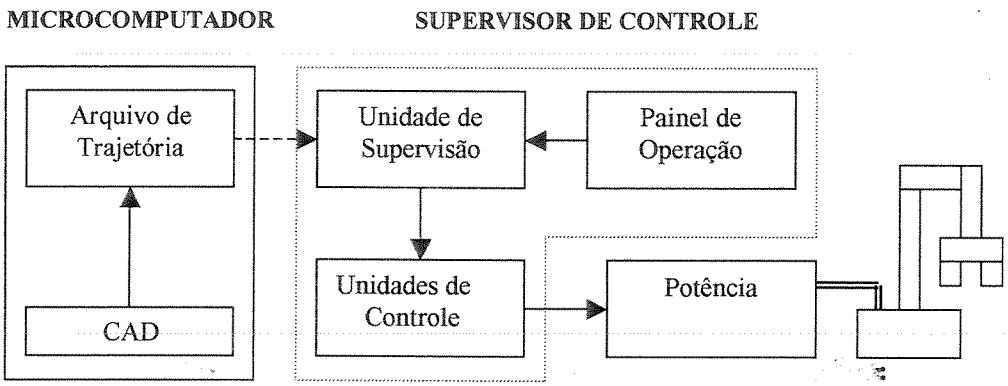


Figura 3.4 – Estrutura do Supervisor de Controle

Para a elaboração da estrutura utilizada, foram abordados alguns aspectos essenciais na execução das aplicações especificadas para o supervisor, dentre os quais tem-se:

- Definição da arquitetura de controle;
- Geração de trajetórias;
- Modelagem de manipuladores;
- Algoritmos para modelagem cinemática inversa.

No respectivo trabalho, para elaboração dos algoritmos de geração de trajetória utilizando o modelo de cinemática inversa foi utilizada a arquitetura de um robô industrial Manutec r3.

Outros aspectos salientados na tese de Fayan estão ligados a considerações feitas quanto à estruturação de uma arquitetura para um supervisor de controle envolvendo a parte de controle de manipuladores a saber:

a) O movimento da estrutura mecânica se realiza através de movimentos de rotação e translação de suas juntas que devem ser controladas simultaneamente e cujo acoplamento dinâmico dificulta o controle independente das mesmas;

b) O comportamento dinâmico da estrutura articulada, fortemente não linear e dependente das condições operativas, deve ser levado em consideração na estratégia de controle escolhida, sendo desejável o acesso a elementos do sistema tais como parâmetros de controle, modelos, interface com o meio exterior, calibração, etc, levando assim a uma perfeita adequação do robô às suas aplicações;

c) A trajetória desejada é definida pelas posições velocidade, aceleração e orientação do atuador, tornando-se necessário então, efetuar transformações de coordenadas com tempos bem definidos e grandes complexidades de cálculos.

O trabalho de Fayan tem continuidade em Dias (junho 1993) e seu objetivo é fazer uma análise prática do desempenho de uma arquitetura distribuída baseada na técnica ATGS (que hierarquiza o controle de movimentos de robôs em dois níveis), aplicada à supervisão e controle de robôs industriais. O robô industrial utilizado foi o mesmo de Fayan.

No trabalho de Martins (1998) destaca-se o estudo embasado na automação relativa ao planejamento do processo, monitoração de máquinas e equipamentos, fluxo de materiais, supervisão do sistema e redes de comunicação. É feita uma distinção entre os termos sistema de supervisão e controle de processos. O relato acrescenta que é possível, mas não eficiente desenvolver estas duas atividades na mesma unidade. O controle de processos, por definição, é o responsável pelo funcionamento do processo, o qual normalmente é contínuo. A função do sistema de supervisão, contudo, está ligada à transição de estado dos processos individuais, como

a coordenação entre estes processos; então se conclui que o sistema de supervisão tem uma visão discreta do sistema, enquanto o controle de processo tem uma visão contínua do sistema.

Um sistema supervisório industrial com a utilização do software Interact (v5.21-fevereiro/99) foi implementado por Georgini (1999). Algumas características destacam-se abaixo:

- Aplicativo de 32 bits;
- Ambiente de execução se dá na plataforma operacional Windows® 95 ou NT;
- Drivers para comunicação com diferentes CLPs;
- Gerenciador de alarmes;
- Módulo para animação gráfica;
- NBIOS – NetBios Network Driver, que permite a conexão do software em rede;
- IMD – Interact Modem Driver, que permite acesso ao software através de Modem.

Neste trabalho são descritos os elementos necessários para implementação de sistemas automatizados de produção. O trabalho de Georgini envolve um estudo, desde a fase preliminar à implementação dos elementos necessários a esta automação até a fase de execução da parte de comando e operativa. Este estudo está fundamentado na plataforma Industrial para Pesquisa, Ensino e Formação em Automação – PIPEFA (ver figura 5.9 no capítulo 5). A PIPEFA foi desenvolvida para ensino, pesquisa e formação tecnológica na área de automação industrial, para proporcionar o desenvolvimento de aplicações em automação que possibilitem melhoria de desempenho, redução de custos, flexibilidade de operação e aumento da qualidade em um sistema de manufatura.

No artigo de Blanc (1999) verifica-se a implementação de um software SCADA, onde se relata a sistemática da arquitetura de controle do processo. Neste trabalho o software de supervisão incorporado foi o Wizcon, um módulo do Wizfactory®, em uma planta industrial de Resfriamento e Ventilação (CV – Cooling & Ventilation). Deste trabalho está centrado na própria implementação do software supervisório Wizcon, sua arquitetura, funcionalidade e da sua real importância para a supervisão de sistemas, no qual se obtêm uma ampla interface gráfica e a possibilidade de configurar as ferramentas para um melhor desenvolvimento de aplicações altamente sofisticadas. A arquitetura utilizada é mostrada na figura 3.5 abaixo.

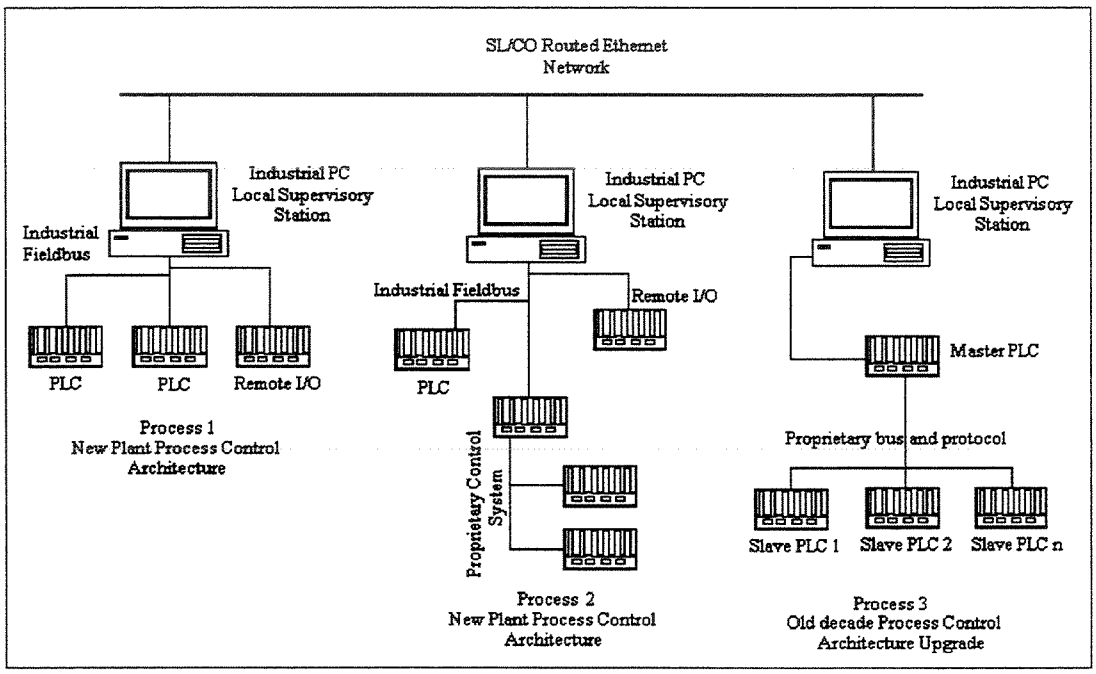


Figura 3.5 – Arquitetura do Software Supervisório para um Processo de Supervisão (Blanc,1999)

3.4 **Visão Geral de um Sistema Supervisório Industrial**

Neste trabalho procurou-se selecionar um software de Supervisão e Controle com características funcionais semelhantes a outros softwares de supervisão existentes no mercado, boa qualidade na interface gráfica e adequado às novas tecnologias de mercado.

Outros fatores que influenciaram na escolha foram o custo, pois existe uma chave para "desenvolvimento educacional" e a facilidade de integração do software com outros aplicativos computacionais, por ser um software que é executado no sistema operacional Windows® e aproveita todos os recursos gerados por este sistema. Esse fato permite a interação do sistema de Supervisão com varios aplicativos destinados a plataforma Windows® inclusive, segundo Blanc (1999), permite a interação com o software Matlab® que é largamente utilizado em engenharia. O software já é desenvolvido para geração de aplicativos para Internet, com o uso da linguagem Java, o que o torna uma plataforma poderosa para transmissão via WEB. Por apresentar uma documentação propicia um treinamento de novas equipes de trabalho que poderão adequar ou até mesmo sofisticar o sistema com as novas tecnologias que irão surgir.

O **WizFactory** é um conjunto de softwares para PC, totalmente integrados e de alta eficiência em aplicações industriais e prediais (domótica) no mundo todo, e que combina o controle discreto de processos e o SCADA com a Internet. O WizFactory desenvolve sistemas de controle com o WizPLC e o WizDCS, visualiza e fornece as informações para usuários autorizados com o Wizcon para Windows e Internet. Com conectividade total entre a Internet, a visualização e os níveis de controle, o Wizfactory simplifica a hierarquia da automação, resultando em menores custos de desenvolvimento e manutenção.

Dentro do software Wizfactory encontra-se uma subdivisão em módulos de acordo com suas funcionalidades (figura 3.6):

- Wizcon
- WizPLC
- WizReport
- WizSheduler

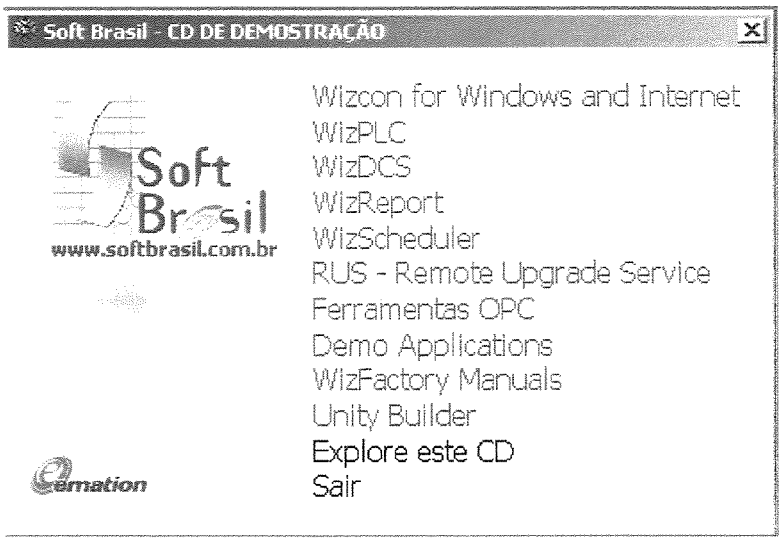


Figura 3.6 – Módulos de instalação do Software Supervisório

Nos itens a seguir serão apresentados os módulos mencionados acima.

WizReport - módulo que facilita a geração de relatórios de produção em vários formatos. Com ele pode-se obter a criação de relatórios complexos em minutos (ver figura 3.7), o registro de dados, agendamento para emissão de relatórios, cálculos personalizados desenvolvidos em Visual Basic. Pode ser conectado a qualquer componente do WizFactory ou pode operar independentemente de outros sistemas de plantas existentes.

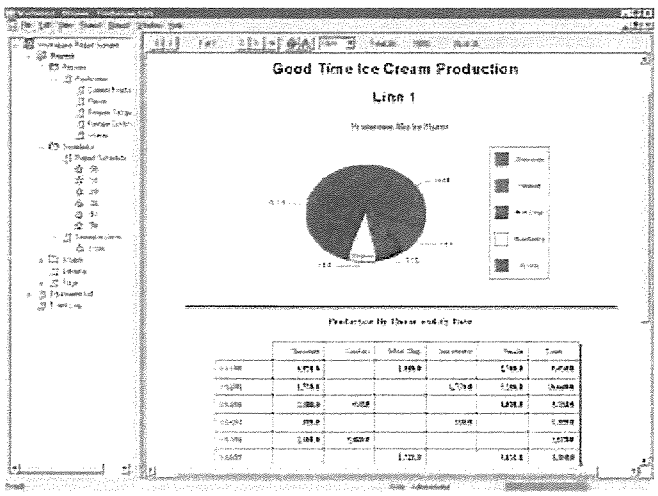


Figura 3.7 - Planilha Gerada pelo Software Supervisório

WizScheduler – utilizado para planejamento, programação e execução de diferentes tarefas baseadas na hora e data. Com a utilização deste módulo pode-se maximizar a energia e reduzir os custos.

WizPIC – é um software de controle para PC que permite desenvolver programas de lógica e de controle discreto. Tem suporte para os principais protocolos fieldbus (para mais detalhes ver anexo II), permite o desenvolvimento de funções de blocos definidas e bibliotecas. É um poderoso aliado do Wizcon; a sua introdução possibilita um melhoramento e apoio à lógica desenvolvida para os sistemas.

Wizcon – é um sistema supervisório de controle e aquisição de dados (SCADA) usado como ferramenta para desenvolvimento de aplicações que permite aos integradores de sistemas criarem sofisticadas aplicações para automações industriais, entre outras. O Wizcon aproveita todas as capacidades do sistema operacional Windows® (preemptivo e multitarefa) e um mecanismo event-driven que garante excelente performance e integridade de dados. Ele também utiliza interface de janelas a fim de garantir limpa e eficiente visualização dos seus componentes. O Wizcon se comunica com CLPs, instrumentos de medida e outros periféricos.

3.5 **Características do Supervisório Wizcon**

Neste módulo encontram-se as características principais de todo o sistema, assim pode-se dizer que os outros módulos servem de apoio a este.

Observar que o Wizcon tem as propriedades para trabalhar em uma rede de comunicação com os seguintes protocolos: NetBEUI, TCP/IP, IPX/SPX que serão abordados no capítulo seguinte.

O funcionamento do software se dá através de uma interação entre este e uma chave de autorização – Hardkey (figura 3.8) – a qual limita ou não o uso do aplicativo (ver tabela 3.1).

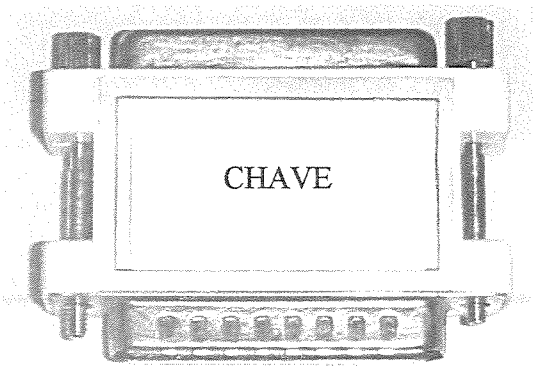


Figura 3.8 - Hardkey

Chave (Plug)	Código	Drivers	Empacotamento
Chave desenvolvimento	DEV	Só vem com os padrões do Wizcon	Empacota
Run Time	RT	Padrões	Não empacota
Evaluation	EVAL	Todos os drivers	Não empacota (funciona por 2h)
Engenharia	ENG	Todos os drivers	Empacota (funciona por 2h)

Tabela 3.1- Informações da Chave (HardKey)

Como mostrado na tabela 3.1, dizemos que uma chave empacota, quando ela gera o arquivo de empacotamento w2pack, este arquivo é o que empacota a aplicação para que ela possa rodar no Run Time. O modo Run Time é um modo no qual não se tem acesso à configuração do Wizcon, ou seja, não é permitido criar gráficos, imagens, Tags. O primeiro ícone da barra de atalhos do Wizcon não existe, portanto não se consegue configurar nada, é um modo apenas para utilizar o software.

Existe outro modo denominado "Modo de Desenvolvimento", cujo ambiente permite criar telas gráficas, isto é, permite a utilização das janelas em que se elaboram os Layouts do sistema monitorado e controlado criando aplicações com animações gráficas (dinâmicas).

O Wizcon possui um modo de atalho (figura 3.9) que possibilita a abertura do modo de apresentação (*Wizcon Application Studio*) do Wizcon; é mostrada na figura 3.10.

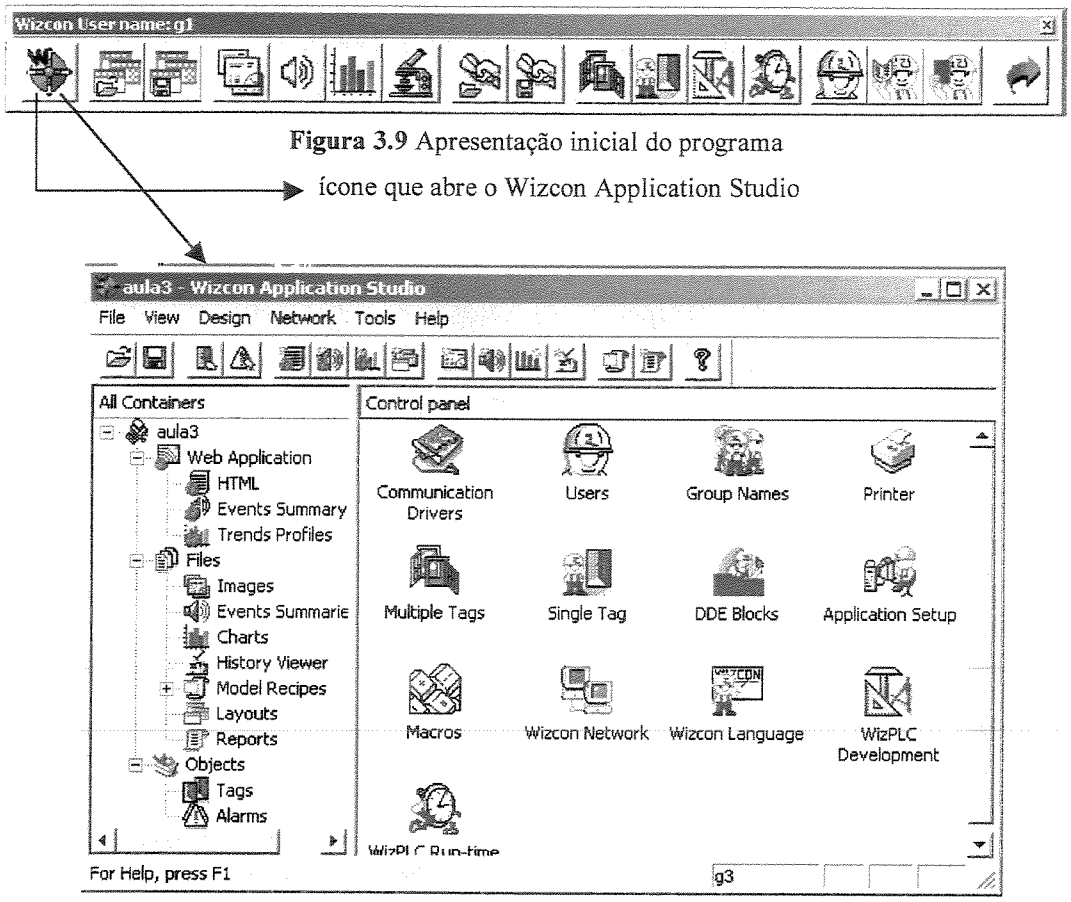


Figura 3.9 Apresentação inicial do programa

Figura 3.10 Módulo de Apresentação do Software de Supervisão

• Animação Gráfica (*Images*): Images é uma janela (figura 3.11) utilizada para desenhar o Layout da planta do supervisório, ou seja, desenhar o ambiente de contato com o operador, desde animação até botões de ação. O ambiente gráfico do Wizcon utiliza figuras vetoriais, permitindo ao engenheiro de aplicação mudar de escala sem a deformação do desenho. Também permite importar figuras nos principais formatos (JPEG, BMP, VIM, ILS) figuras de aplicativos do tipo CAD, como por exemplo, o conhecido Autocad.

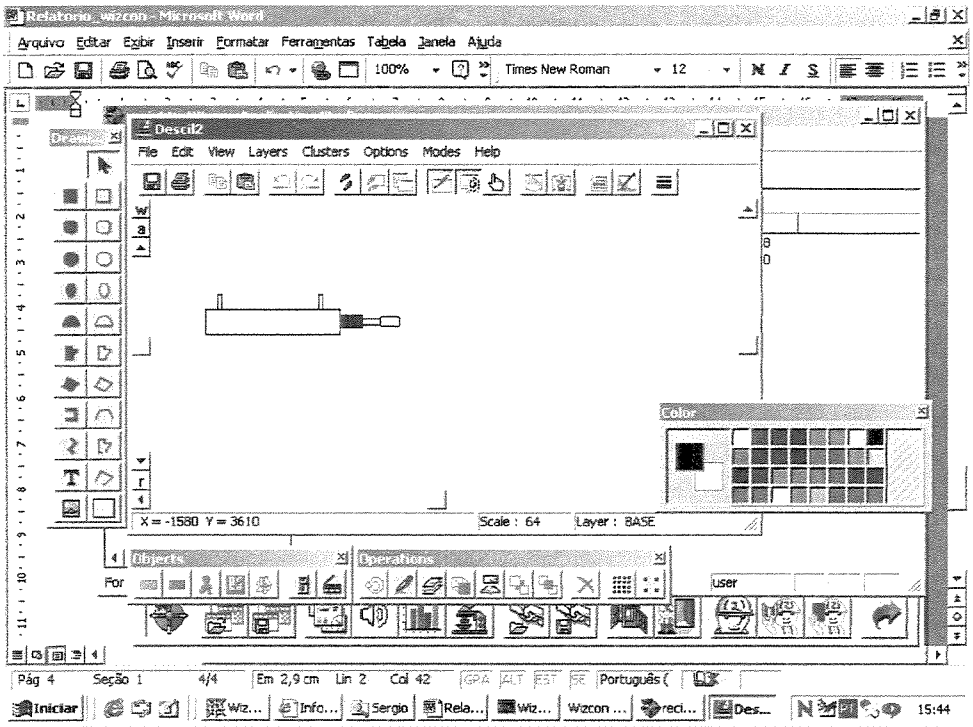


Figura 3.11 - Janela do Editor Gráfico

Integra esta janela um banco de dados Cluster Library, onde existem ferramentas que permitem a construção fácil e rápida de Painéis de Operação e Gerenciamento compostos por Ferramentas de Entrada e Saída, tais como: Bargraphs, Dispositivos Hidráulicos, Reservatórios, Push Buttons, Displays Numéricos, Teclas de Funções, entre outros. As ferramentas são configuradas para enviar e/ou receber dados (discretos ou analógicos) do PLC. A figura 3.12 mostra um Layout criado no qual está simulado um ambiente real.

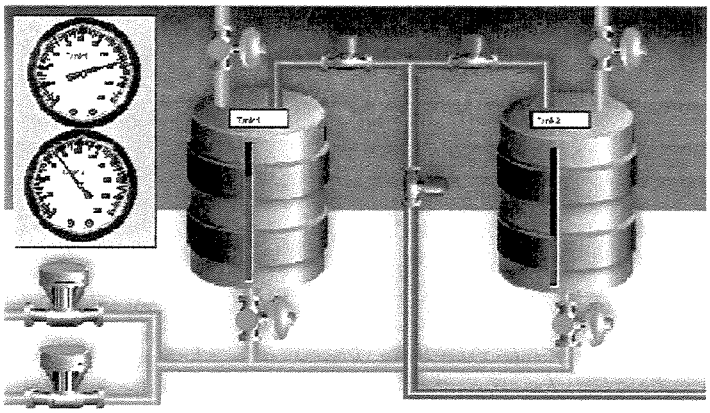


Figura 3.12 - Layout do Sistema com Aplicação Dinâmica

• Relatórios (*Report*) – o Wizcon através desta janela pode gerar relatórios otimizados para necessidades específicas de cada planta, em um formato livre com textos e valores calculados (field), baseado em arquivos de histórico gravados durante o processo. O "Field" é um campo para configuração de funções matemáticas e intervalos de valores em que se deseja obter das variáveis selecionadas. Quando um relatório (report) está pronto, ele pode ser gerado pelo "prompt" de comandos do sistema operacional, por Macros ou pela Linguagem Wizcon, uma linguagem específica do supervisório. Permite que o relatório gerado seja gravado em arquivo e/ou impresso.

• Alarmes (*Alarms*) : nesta janela, mostrada na figura 3.13, é permitida a criação de alarmes para o controle do sistema. Alarmes são mensagens de sistema definidas pelo engenheiro de aplicação para alertar o operador em relação a alguma situação específica. No Wizcon, podemos definir até 65000 alarmes, independente da chave que está sendo utilizada. Cada alarme é definido de maneira independente, portanto, pode-se ter 65000 alarmes associados ao mesmo Tag. A seguir são citadas algumas características dos alarmes:

- definidos por condições algébricas;
- podem ser mostrado com textos específicos;
- os alarmes podem ser enviados para impressoras;
- podem ser configurados para tocar sirenes reais;
- na ativação do alarme a "zona" em que está acontecendo o evento pode ser mostrada;
- quando ativados enviam e-mails e mensagens para celulares (via WAP); esta função encontra-se na última versão do Wizcon.

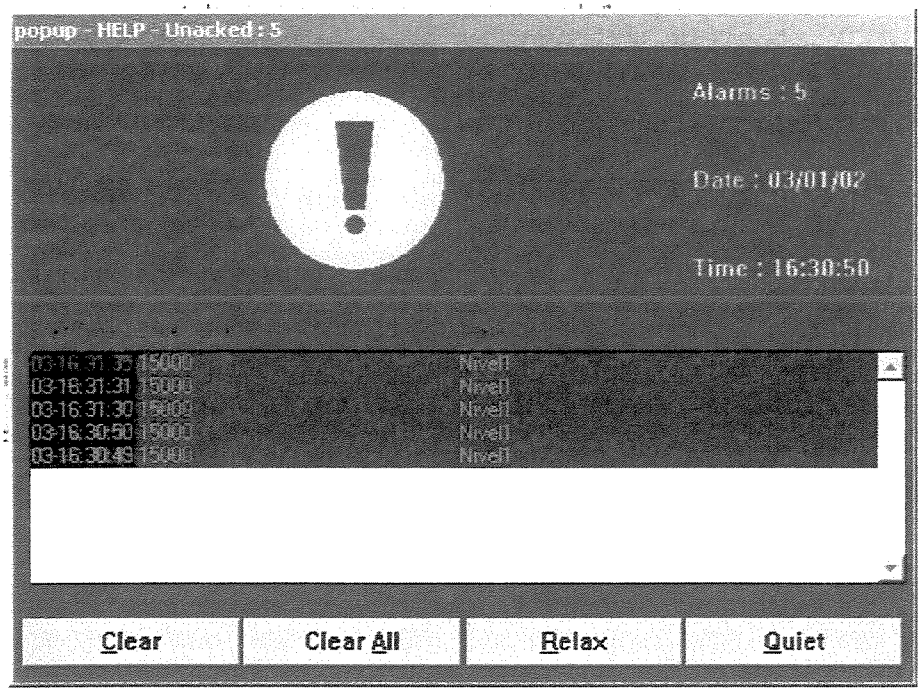
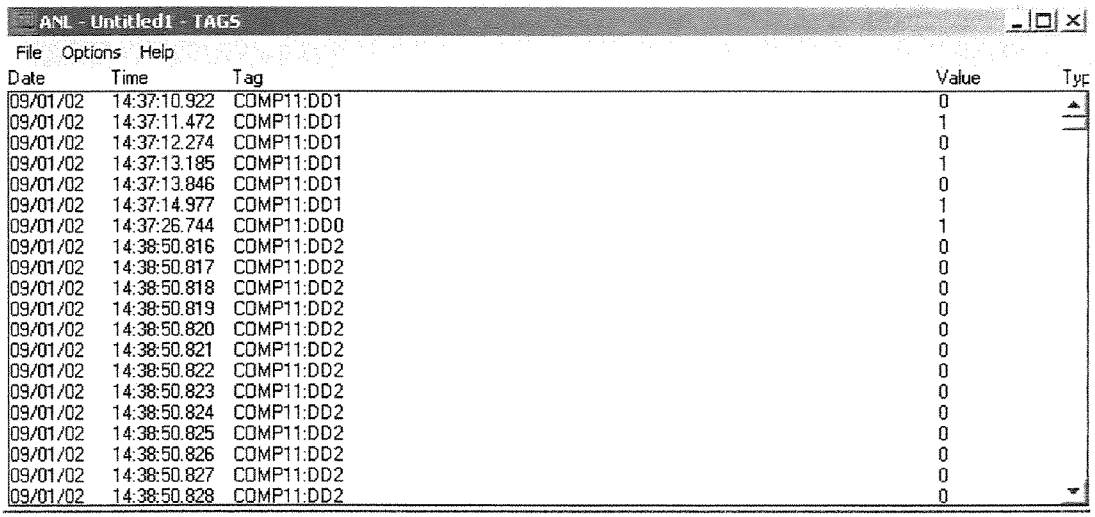


Figura 3.13 - Alarme

• Visualizador de Histórico (*History Viewer*) : com esta função pode-se ter a leitura das variáveis de estado do sistema. Permite fazer uma pré-seleção das variáveis que se deseja colocar no histórico. Esta função está mostrada na figura 3.14.



Date	Time	Tag	Value	Typ
09/01/02	14:37:10.922	COMP11:DD1	0	
09/01/02	14:37:11.472	COMP11:DD1	1	
09/01/02	14:37:12.274	COMP11:DD1	0	
09/01/02	14:37:13.185	COMP11:DD1	1	
09/01/02	14:37:13.846	COMP11:DD1	0	
09/01/02	14:37:14.977	COMP11:DD1	1	
09/01/02	14:37:26.744	COMP11:DD0	1	
09/01/02	14:38:50.816	COMP11:DD2	0	
09/01/02	14:38:50.817	COMP11:DD2	0	
09/01/02	14:38:50.818	COMP11:DD2	0	
09/01/02	14:38:50.819	COMP11:DD2	0	
09/01/02	14:38:50.820	COMP11:DD2	0	
09/01/02	14:38:50.821	COMP11:DD2	0	
09/01/02	14:38:50.822	COMP11:DD2	0	
09/01/02	14:38:50.823	COMP11:DD2	0	
09/01/02	14:38:50.824	COMP11:DD2	0	
09/01/02	14:38:50.825	COMP11:DD2	0	
09/01/02	14:38:50.826	COMP11:DD2	0	
09/01/02	14:38:50.827	COMP11:DD2	0	
09/01/02	14:38:50.828	COMP11:DD2	0	

Figura 3.14 - Visualizador de Histórico

• Sumário de Alarmes (*Events Summaries*): esta janela (figura 3.15) é utilizada para visualização de alarmes. Através dela, pode-se reconhecer alarmes, visualizar alarmes históricos gravados em disco e muitas outras funções. Nesta janela pode-se acompanhar as informações sobre os alarmes gerados desde o início do projeto.Uma função agregada neste sumário e muito importante para soluções industriais é quando o alarme se encontra em vermelho, significando que o problema ainda não foi solucionado. Nesta janela é possível selecionar os alarmes por:

- data e horário em que o alarme ocorreu;
- a "zona" em que aconteceu o alarme;
- grupo a que pertence o alarme;
- nível de severidade do alarme;
- descrição da condição de alarme (vermelho indica que a condição não foi sanada);
- data e horário em que o alarme tornou-se inativo;
- data e horário em que foi reconhecido;
- nome do operador que reconheceu o alarme;
- valor do alarme (no momento em que ocorreu);
- o nome da estação onde está sendo gerado o alarme;
- texto de informação sobre o alarme.

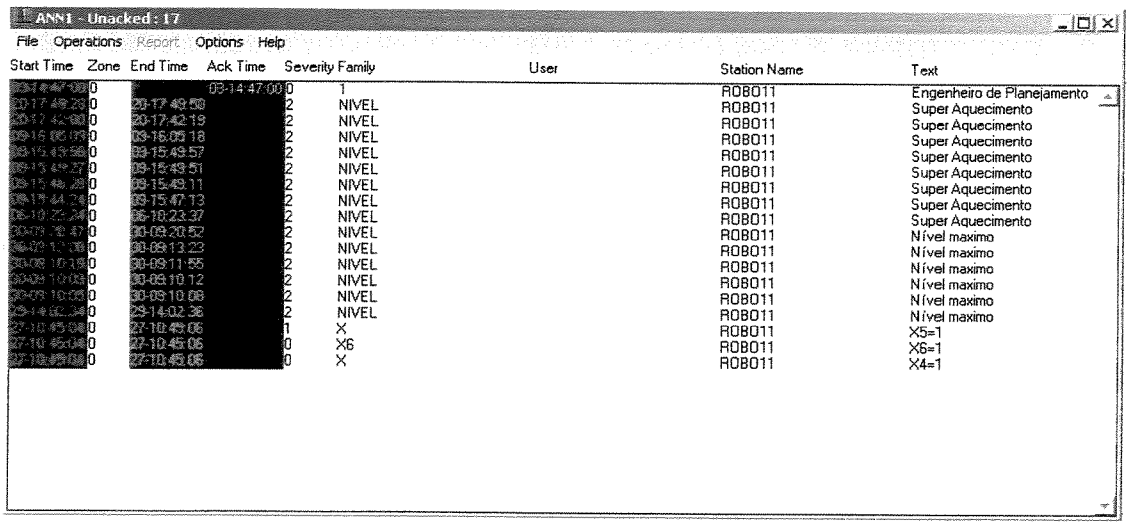


Figura 3.15– Sumário de Alarmes

• Programação interna Wizcon (**Wizcon Language**): nesta propriedade o Wizcon fornece uma lógica paralela para aumentar o poder de funcionalidade do sistema. É uma ferramenta utilizada para fazer cálculos avançados e executar operações automaticamente, de acordo com lógicas pré-estabelecidas pelo programador. Esta propriedade é muito utilizada, pois melhora a eficiência do sistema e aumenta as possibilidades de expansibilidade do sistema. Nesta propriedade é encontrada uma janela (figura 3.16) para a edição de uma programação que é feita em módulos lógicos e um programa para a compilação é adicionado ao sistema através de um aplicativo especialmente criado para gerenciar adições de programas executáveis para o Wizcon.

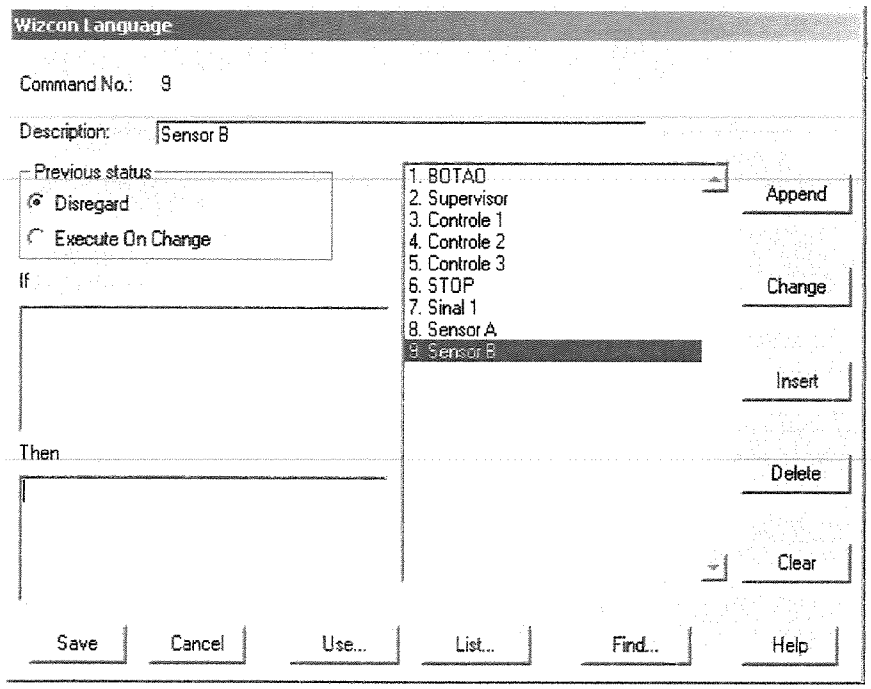


Figura 3.16 – Ambiente de Programação

• Gerador de Gráficos on Line (*Chart*): através desta ferramenta pode-se acompanhar gráficamente como mostrado na figura 3.17 as mudanças das variáveis de controle de maneira on-line, ou seja, em tempo real.

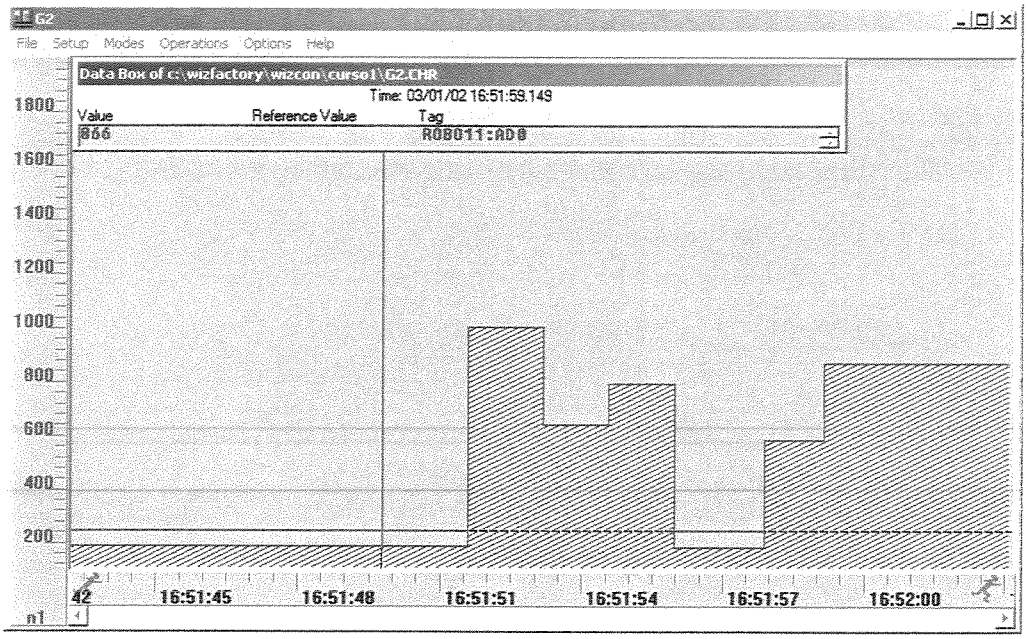


Figura 3.17 – Gerador Gráfico de Variáveis do Sistema

• Localização de Unidades com o Supervisor (Wizcon Network) – através deste aplicativo o Wizcon fornece um banco de dados sobre as Estações de Trabalho que estão rodando o aplicativo Wizcon naquele momento. Estas estações podem interagir possibilitando que uma estação remota possa acessar drivers (Tags) da outra Estação.

• Drivers de Comunicação (Communication Drivers) – são utilizados para fazer a comunicação do Wizcon com o equipamento desejado. O Wizcon oferece vários tipos de drivers para os mais variados tipos de PLC, placas controladoras etc, sendo que cada driver possui uma configuração específica que deve ser observada cuidadosamente.

Neste capítulo foram delineados temas como significado de supervisório, campos de aplicação, grau de automação, descrição dos recursos do Software Wizfactory com ênfase no módulo Wizcon.

No próximo capítulo, serão apresentados os principais elementos de integração utilizados em um sistema supervisório para possibilitar o controle e monitoramento dos sistemas automatizados em uma rede de comunicação.

Capítulo 4

Principais Elementos de Integração Utilizados em um Sistema Supervisório

O sistema Supervisório ou SCADA, como é conhecido no ambiente industrial, opera de acordo com vários fatores tanto relativos a software como a hardware. A presença de um ambiente de comunicação entre elementos de controle e monitoração é atualmente uma tendência que traz benefícios e sofisticações ao sistema, possibilitando a integração de todo sistema automatizado. Para permitir que se estabeleça uma comunicação do sistema de supervisão e controle com as outras partes, possibilitando uma troca dinâmica de dados entre a parte operacional e o comando, uma série de elementos são necessários. Este capítulo será dividido em dois itens: o primeiro (4.1) aborda os principais elementos para um sistema de controle e supervisão e no item subseqüente (4.2) será mostrada a arquitetura de todo sistema utilizado neste trabalho com a implementação dos elementos para comunicação e controle juntamente com o sistema para envio de imagens em tempo real. A figura 4.1 descreve toda a arquitetura do sistema de comunicação utilizado neste trabalho.

4.1 **Principais Elementos para Integração em um Ambiente de Controle e Supervisão via rede Local ou Internet**

Neste item serão abordados os principais componentes para integração de um sistema moderno de controle e supervisão com acesso remoto. A figura 4.1 mostra de maneira exemplificativa os componentes necessários para a integração de todo nosso sistema de controle e

monitoração. Entre os elementos principais temos estações remotas, protocolos, estações de supervisão, servidores, etc.

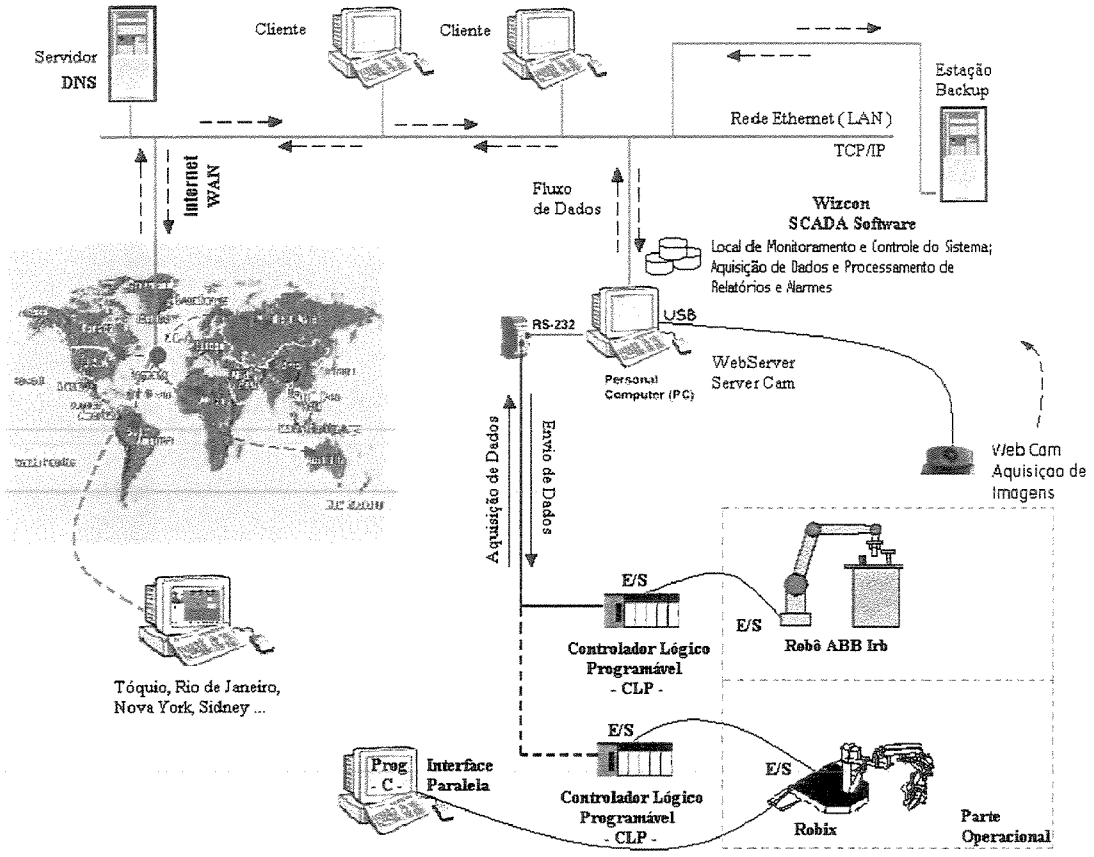


Figura 4.1 – Arquitetura para Controle e Supervisão via Internet

4.1.1 **Estações Remotas**

O processo de controle e aquisição de dados inicia-se nas estações remotas, CLPs (*Controlador Lógico Programável*) e RTUs (*Remote Terminal Units*), com a leitura dos valores atuais dos dispositivos que lhes são devidamente associados.

Os CLPs e os RTUs são pequenos computadores através dos quais as estações centrais de monitoração se comunicam com os dispositivos existentes nas instalações controladas e monitoradas.

Os PLCs tem como principal vantagem a facilidade de programação e controle de dispositivos através das Entradas/Saídas (E/S) ou Input/Output (I/O). Por outro lado, os RTUs possuem boa capacidade de comunicação, incluindo comunicação via rádio, estando especialmente indicados para situação adversa onde a comunicação é difícil.

Atualmente, nota-se uma convergência no sentido de reunir as melhores características destes dois equipamentos - a facilidade de programação e o controle dos CLPs e as capacidades de comunicação dos RTUs.

Na maioria dos sistemas SCADA a leitura das variáveis de estado é associadas a Tags, que representam o endereço lógico das entradas e saídas dos componentes conectados ao sistema, como, por exemplo, determinada entrada de um CLP denominada "X0", para o sistema supervisório, significa um Tag, ou seja, um endereçamento lógico de memória, pois com o devido endereçamento, o protocolo de comunicação permitirá o envio e recebimento de dados do dispositivo em questão com o computador (Sistema Supervisório).

No item a seguir atentaremos para algumas definições e características dos CLPs para um melhor entendimento do leitor. Este trabalho utiliza o auxílio deste dispositivo para o controle e supervisão dos sistemas implementados.

4.1.1.a **Controlador Lógico Programável – CLP**

O PLC (Programmable Logic Controller) também conhecido no Brasil como CLP (Controlador Lógica Programável) ou CP (Controladores Programáveis) é um dispositivo físico, eletrônico que possui uma memória interna programável, capaz de armazenar seqüências de instruções lógicas binárias além de outros comandos. Possui dispositivo para conectar–se com outros equipamentos externos o que permite o recebimento e/ou envio de variáveis de entrada ou variáveis de saída respectivamente.

As variáveis de entrada são sinais externos recebidos pelo CLP que podem ser oriundos de fontes pertencentes ao processo controlado ou de comandos gerados pelo operador. Tais sinais são gerados por dispositivos como sensores, chaves ou botoeiras, transdutores, dentre outros.

As variáveis de saída são os dispositivos controlados por cada ponto de saída do CLP. Tais pontos poderão servir para intervenção direta no processo controlado por acionamento próprio, ou também poderão servir para sinalização de estado em painel sinótico. Podem ser utilizados como exemplo de variáveis de saída, para o acionamento de válvulas, solenóides, displays, chaves e até mesmo sinais para outros CLPs, dentre outros. Na figura 4.2, a aplicação de um CLP.

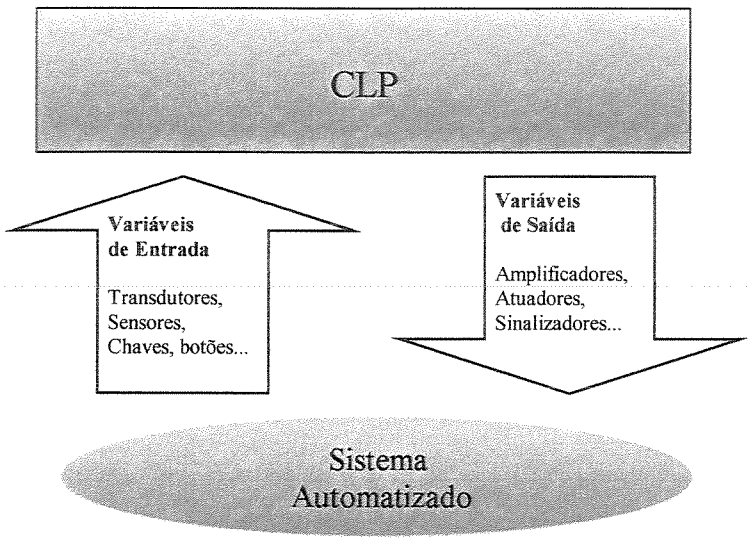


Figura 4.2 - Aplicação Genérica do Controlador Lógico Programável
(Georgini, 2000)

Historicamente podemos dizer que os primeiros CLPs surgiram em 1968 quando a Divisão Hydromatic da GM determinou os critérios para projeto do CLP, sendo que o primeiro dispositivo a atender às especificações foi desenvolvido pela Gould Modicon em 1969. Em 1971, ocorreram as primeiras utilizações fora da indústria automobilística. Em 1975, foi introduzido o controlador PID. Até 1977 eram construídos com componentes eletrônicos discretos; a partir desta época estes passaram a ser substituídos por microprocessadores, sendo daí em diante aceito industrialmente.

O princípio de funcionamento fundamental do CLP é a execução dentro de uma CPU de um programa (Firmware), desenvolvido pelo fabricante, que realiza sistematicamente ações de

leitura das variáveis de entrada através do módulo de entrada do CLP, em conjunto executa um programa armazenado e desenvolvido pelo usuário, sendo que este programa é destinado para controle e monitoração de tarefas específicas, no qual, através de uma lógica, faz ou não intervenções nas variáveis de saída através do módulo de saída do CLP. A figura 4.3 mostra de forma simplificada, o que foi exposto.

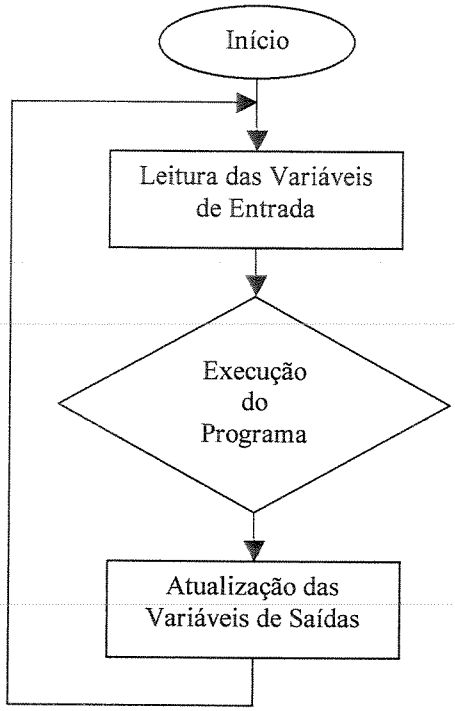


Figura 4.3 – Funcionamento do CLP

- Os Controladores Lógicos Programáveis possuem a seguinte arquitetura básica:

    1. Unidade Central de Processamento (CPU- *Central Processing Unit*): integra esta unidade o processador (microprocessador, processador dedicado ou microcontrolador), o sistema de memória (ROM e RAM) e os circuitos internos;

    2. Fonte de Alimentação: parte responsável pelo fornecimento de tensão de alimentação fornecida à CPU e aos circuitos / módulos de E/S (entrada e saída);

3. Circuitos de E/S (*Input/Output* – I/O): circuitos para o recebimento ou envio de sinais. Podem ser discretos (sinais digitais, contatos normalmente abertos ou fechados) ou sinais analógicos.

4. Base: é responsável por proporcionar conexão mecânica e elétrica entre a CPU, os módulos de entrada e/ou saída e a fonte de alimentação. Possui o barramento de comunicação entre eles, no qual os sinais de dados, endereço, controle e tensão de alimentação estão presentes.

A figura 4.4 mostra o CLP do fabricante Koyo.

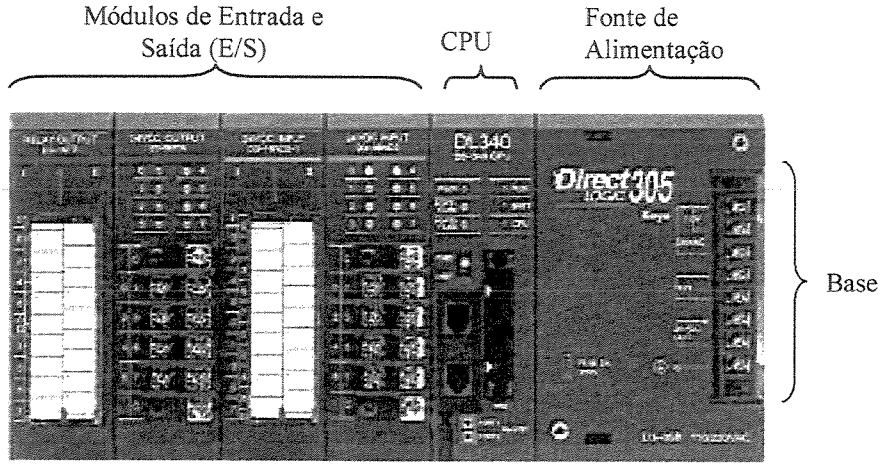


Figura 4.4 – Exemplo de um CLP

Para a existência da comunicação do CLP com o PC é necessário que dois aspectos sejam observados tanto em relação ao hardware como em relação ao software.

Quando se fala em **hardware** de comunicação, normalmente as CPUs possuem pelo menos uma porta de comunicação serial que pode ser conectada a dispositivos externos. A quantidade pode variar de acordo com o modelo e fabricante de CLPs. Através das portas de comunicação pode-se estabelecer uma conexão entre o usuário e o controlador ou mesmo interligar a outras CLPs. Nesta porta pode-se interligar um computador (PC) ou um dispositivo portátil de programação que, quando utilizados, permitem que se façam autodiagnóstico, programação ou reprogramação de instruções, alterações de configurações do dispositivo, monitoração, gravação e apagamento da memória.

A comunicação entre o computador (PC) e a CLP é feita por meio da porta serial (RS-232) e com a utilização de cabos apropriados. Todavia algumas CLPs utilizam o padrão RS-422 e necessitam de conversor RS 232/RS422 (ver figura 4.5) para conexão. Existe ainda CLPs que utilizam padrão próprio e necessitam de interface dedicada instalada no PC.

A porta serial RS232 padrão EIA (*Eletronic Industries Association*) é utilizada para transmissão de dados até 15m. É encontrada nas portas seriais dos PCs. Tem-se também a porta RS 422 também padrão EIA na qual a transmissão de dados é feita a distâncias maiores (até 1200m) maior velocidade e, por algumas de suas características, possui maior imunidade a ruídos. O tipo de comunicação é *full-duplex* (pode enviar e receber dados simultaneamente). Uma outra conexão similar ao padrão RS422 é a RS485 na qual os receptores têm proteções e capacidades maiores. A comunicação é *half-duplex* (pode apenas enviar ou receber dados em um mesmo instante).

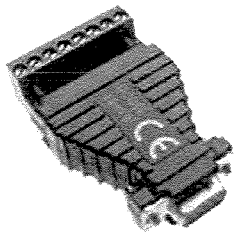


Figura 4.5 – Conversor RS 232/RS422 ou RS485

Quando se fale de **software**, estamos falando em protocolo de comunicação, o qual determina a forma de transmissão dos dados, ou seja, ele é responsável pelo sinal de comunicação entre a CLP e o PC. Os protocolos são fornecidos pelos fabricantes que geralmente adotam Protocolos de Comunicação próprios, os quais normalmente são conhecidos como "Protocolo Proprietário". Alguns são "abertos" (o usuário pode ter acesso ao formato da transmissão de dados utilizado, podendo desenvolver seus próprios programas de comunicação) e outros o fabricante não fornece nenhum dado sobre o seu Protocolo Proprietário. Entretanto existe a possibilidade de certos CLPs suportarem protocolos padrões como por exemplo Modbus, o que

permite a comunicação com dispositivos e softwares adotador por outros fabricantes, além de permitir a conexão em rede.

As linguagens de programação desenvolvidas para os CLPs podem ser representadas de três formas:

- Redes de contato (Ladder): similar aos esquemas elétricos de relés e contatores;
- Blocos funcionais; similares aos esquemas elétricos de circuitos digitais (AND, OR, XOR, etc.);
- Lista de instruções: é a programação diretamente apoiada nas funções lógicas binárias e se assemelha bastante aos programas escritos em linguagem assembler.

A mais utilizada é a linguagem Ladder (figura 4.6), que está presente praticamente em todos os CLPs disponíveis no mercado. Esta linguagem baseia-se nas redes de contatos dos esquemas elétricos, sendo por isso, uma linguagem gráfica de fácil manipulação e, mesmo existindo diferenças entre os fabricantes quanto às representações das instruções, são facilmente assimiladas pelo usuário.

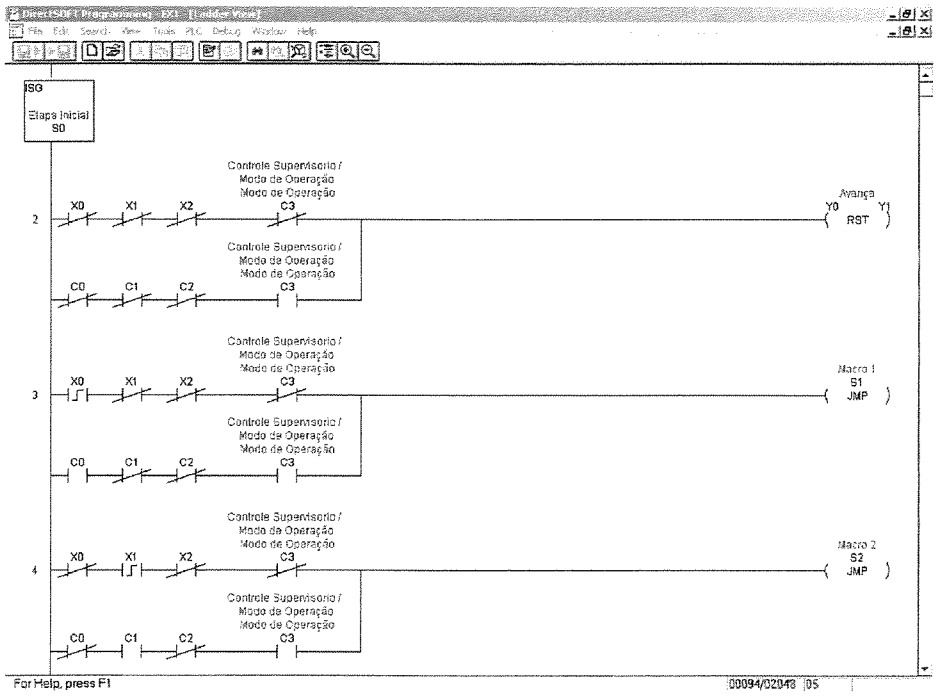


Figura 4.6 – Programa em Ladder (DirectSoft)

Na linguagem Ladder cada contato ao assumir dois estados (fechado ou aberto), representa uma variável "booleana", ou seja, uma variável que assume dois estados; verdadeiro ou falso. Como mostrado na figura 4.6 o símbolo -[ ]- significa contato normalmente aberto, cuja variável associada é 1. No símbolo -[/]- contato normalmente fechado, o estado da variável associado é zero. Cada linha "Ladder" permite programar desde funções binárias até funções digitais complexas.

4.1.2 **Dispositivos Para Auxílio no Controle de Sistemas**

Os Sensores e Atuadores são dispositivos conectados aos equipamentos monitorados e/ou controlados pelos sistemas SCADA.

Os Sensores são utilizados para aquisição de dados informativos dos sistemas controlados, os quais convertem parâmetros físicos, tais como presença física de um corpo, níveis de água, temperatura, etc. para sinais analógicos e digitais para as estações remotas.

Os Atuadores são utilizados, como o próprio nome indica, para atuar no sistema. Os atuadores podem ser considerados como dispositivos de saída das unidades remotas.

4.1.3 **Redes de Comunicação.**

O aprimoramento da tecnologia empregada nos computadores e na comunicação influenciou profundamente o modo como são organizados os sistemas computacionais e, portanto, os sistemas de automação. O conceito de "centro de computação", uma sala com um computador enorme ao qual as pessoas levam seus trabalhos para serem processados, está obsoleto. Foi substituído por um outro em que existe um grande número de computadores autônomos independentes, porém interconectados. Esses sistemas são chamados de Redes de Computadores (Tanenboum, 1997).

Neste tópico será descrita a arquitetura das redes de equipamentos e computadores que podem compor um sistema de automação.

Conforme o número de controladores programáveis (computadores aplicados em automação), os tipos de sistemas existentes podem ser classificados em **concentrados** ou **distribuídos**.

Por sistemas **concentrados** compreende-se um computador gerenciando um processo construído por unidades remotas, sendo todo processamento realizado numa única máquina. A informação percorre uma direção vertical através de uma estrutura hierarquizada.

Nos sistemas **distribuídos** o gerenciamento da informação, assim como o controle da automação, é realizado por máquinas alocadas ao longo da planta. As remotas deixam de ser executoras para assumirem participação no processamento. Os terminais também são "inteligentes". Dessa forma, sistemas distribuídos (*distributed systems*) e redes de computadores (*network*) requerem uma cuidadosa análise para não serem confundidos.

Quando se fala em sistemas distribuídos, existe um conjunto de máquinas autônomas invisíveis ao operador. Para este, a presença de diversos processadores é transparente, tudo se realiza automaticamente pelo sistema, constituído por um conjunto de softwares instalados na planta.

A alocação de trabalhos e/ou tarefas aos processadores de arquivo para discos, a transferência de arquivos de onde eles são gerados até onde serão utilizados, assim como outras funções do sistema, são automáticas.

Em redes de computadores, os operadores deverão declaradamente efetuar o login na máquina, mover arquivos em geral e manipular pessoalmente o gerenciamento da rede. Nos dois casos, tanto em sistemas distribuídos como em redes, existem itens em comum - o processamento ocorre em vários pontos do sistema e a informação e a movimentação de arquivos se realizam nas direções vertical e horizontal através dos controladores e/ ou computadores.

Nota-se, então, que a diferença entre os sistemas está no modo de participação do operador- - nas redes o usuário coordena a realização das tarefas que nos sistemas distribuídos é automática.

Conclui-se que em relação ao sistema físico (hardware) a rede e os sistemas distribuídos são iguais. A diferença está residindo no sistema lógico operacional (software). Nos sistemas distribuídos é instalado um conjunto de softwares conferindo à planta confiabilidade, coerência e transparência, operando automaticamente.

As vantagens dos sistemas distribuídos e/ou redes de computadores podem ser exemplificadas nos sistemas de automação industrial, os quais possuem um grande número de CLPs e interfaces homem-máquina operando em locais remotos. Com a finalidade de aumentar a confiabilidade do sistema automatizado verifica-se que a integração entre os CLPs contribui para isso e as redundâncias, através de backup, garantem o sucesso desejado (Moraes, 2001).

4.1.4 **Protocolos**

Com o surgimento das redes de computadores, as soluções eram na maioria das vezes proprietários, isto é, uma determinada tecnologia só era suportada por seu fabricante. Dessa forma, um mesmo fabricante era responsável por construir praticamente tudo na rede. Mas estas situações geravam problemas de custos, expansibilidade dos sistemas com dificuldade de integração destes sistemas. O mercado começou a almejar uma rede de comunicação "inteligente" algo que possibilitasse a total integração entre estes sistemas de comunicação, ou seja, um sistema integrado que "fala-se entre si", uma tecnologia de comunicação mais padronizada.

Então em 1977, a International Standards Organization (ISO) desenvolveu um modelo de referência, para interconexão aberta num sentido mais universal, cuja intitulação foi Open Systems Interconnection (OSI), para que os fabricantes pudessem criar protocolos a partir desse modelo.

O modelo de protocolo OSI é um modelo de sete camadas, apresentadas na figura 4.7.

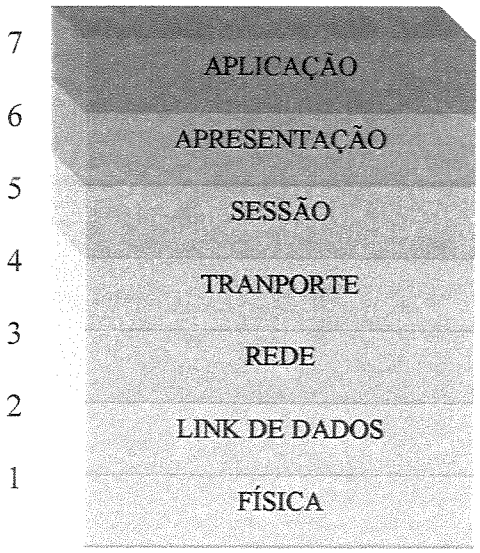


Figura 4.7 - Modelo OSI de Protocolos

Camada 1 - Física (*Physical*): recebe os quadros pela camada de transmissão de dados e os transforma em sinais compatíveis com o meio no qual os dados deverão se transmitidos. Nesta camada, por exemplo, existe a transferência de bits.

Camada 2 – Link de dados (*data link*): também conhecida como enlace, assegura que o conteúdo da mensagem gerado em sua origem seja exatamente igual ao conteúdo no local de destino. Para tal essa camada possui uma sub-rotina, muitas vezes constituída por algoritmo especial, que permite a detecção de erros de comunicação, em nível da camada física, mantendo a integridade da mensagem. Esta camada é geradora de um bit de paridade, ou um conjunto de bits extras, que possui a função de proteção. Ela cria números seqüenciais do lado da transmissão e do lado da recepção para sua devida validação.

Camada 3 – Rede (*Network*): cuida do tráfego e "roteamento" (encaminhamento) dos dados, este "roteamento" é baseado em fatores condicionantes como, por exemplo, tráfego de dados e prioridades.

Camada 4 – Transporte (*transport*): controla a transferência dos dados e transmissões; isto é executado pelo protocolo utilizado.

Camada 5 – Sessão (*session*): permite que duas aplicações em computadores diferentes estabeleçam uma sessão de comunicação.

Camada 6 – Apresentação (*presentation*): também conhecida como camada de tradução, converte o formato do dado recebido pela camada de aplicação em um formato comum a ser usado na transmissão desse dado, ou seja, um formato entendido pelo protocolo usado.

Camada 7 – Aplicação (*application*): esta camada faz o interfaceamento entre o protocolo de comunicação e o aplicativo que pediu ou receberá a informação através da rede.

Dentre os principais protocolos utilizados atualmente podem ser destacados:

- TCP/IP;
- NETBUI;
- IPX/SPX;

• **TCP/IP** é, na realidade, um conjunto de protocolos. Os mais conhecidos dão o nome a este conjunto: TCP (*Transmission Control Protocol*, Protocolo de Controle da Transmissão) e IP (*Internet Protocol*), o protocolo TCP/IP é atualmente o mais utilizado em redes locais. Isso se deve basicamente à popularização da Internet, a rede mundial de computadores, já que esse protocolo foi criado para ser usado na Internet. Mesmo os sistemas operacionais de redes, que no passado só utilizavam o seu protocolo proprietário (como o Windows® com o seu NetBEUI e o Netware com seu IPX/SPX), hoje suportam o protocolo TCP/IP.

Segundo Torres (2000) umas das grandes vantagens do TCP/IP em relação a outros protocolos existentes é que ele é "roteável", isto é, foi pensado em redes grandes e de longa distância, em que pode haver vários caminhos para o dado atingir o computador receptor. Outro fato que tornou o TCP/IP popular é que ele possui arquitetura aberta e qualquer fabricante pode adotar a sua própria versão do TCP/IP em seu sistema operacional, sem a necessidade de pagamento de direitos autorais a ninguém. Com isso, todo o fabricante de sistemas operacionais

acabou adotando o TCP/IP, transformando-o em um protocolo universal, possibilitando que todos os sistemas possam comunicar-se entre si sem dificuldade.

Como foi dito anteriormente, o protocolo TCP/IP é "roteável", isto é, ele foi criado pensando-se na interligação de diversas redes – na quais podem se ter diversos caminhos ligando o transmissor e o receptor -, culminando na rede mundial que hoje conhecemos por Internet. Por isso ele utiliza um esquema de endereçamento lógico chamado endereçamento IP. Esse endereço permite identificar o dispositivo e a rede à qual ele pertence, ou seja, cada computador deve ter um número que o identifique em função da rede que ele ocupa na rede mundial de computadore,s a Internet.

A arquitetura do TCP/IP está mostrada na figura 4.8. Como pode se observar, o protocolo TCP/IP é um protocolo de quatro camadas.

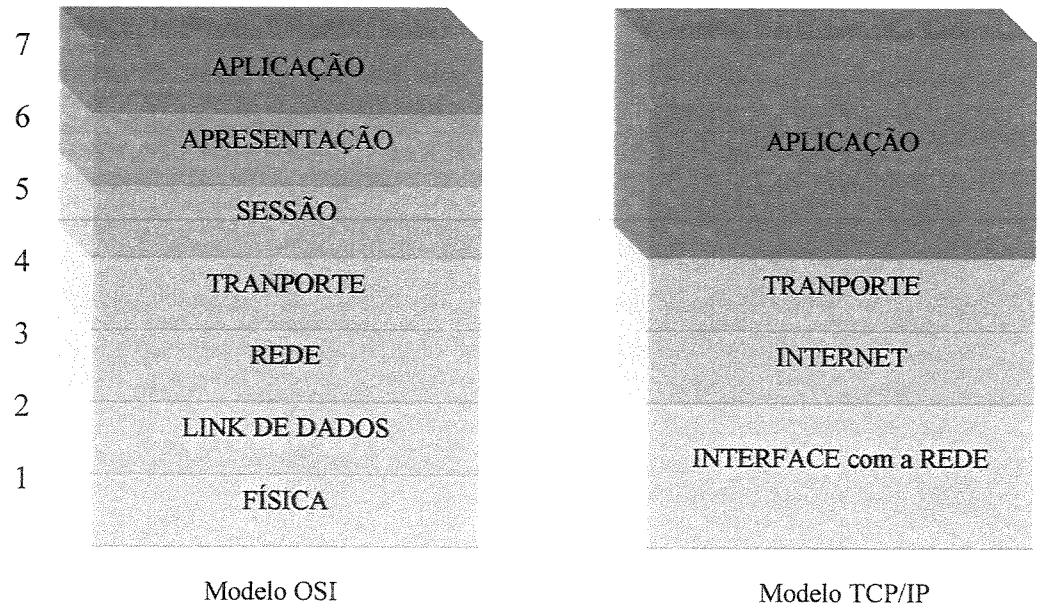


Figura 4.8 – Arquitetura do TCP/IP

A camada de aplicação do modelo TCP/IP equivale às camadas 5,6 e 7 do modelo OSI e faz comunicação entre os aplicativos e o protocolo de transporte. Nesta camada existem vários protocolos que a compõe. Dentro desta camada podemos destacar o HTTP (*Hypertext Transfer*

Protocol). Quando se acessa um endereço "www" em seu navegador Web para visualizar uma página na Internet, o navegador comunicar-se-á com a camada de aplicação do TCP/IP, sendo atendido pelo protocolo HTTP (Torres, 2000). O protocolo HTTP utiliza sempre a porta 80.

Outro protocolo de aplicação encontrado é conhecido como DNS (*Domain Name System*). Como foi visto, toda a máquina em uma rede TCP/IP possui um endereço IP. Sabe-se que os endereços IP não são tão fáceis de ser guardados quanto os nomes, para as pessoas. Por isso, foi criado o sistema DNS, que permite dar nome a endereços IP, ou seja, seria uma espécie de apelido que facilita a localização de máquinas por nós, humanos. Quando se digita um endereço de um site, digitamos endereços como www.fem.unicamp.br/robot11 e, na verdade, os endereços são uma conversão para a forma nominal de um endereço IP (é muito mais fácil guardar o endereço nominal www.fem.unicamp.br/robot11 do que um endereço IP da máquina como o número 210.212.321.9 , por exemplo). Quando se entra com o endereço em um navegador Web (Explorer ou Netscape), o navegador se comunica com um servidor DNS, que é responsável por descobrir o endereço IP do nome entrado, permitindo que a conexão seja efetuada. Cada rede local TCP/IP precisa ter, ao menos, um servidor DNS como mostrado na figura 4.1. Todos os pedidos por conversão de nomes em endereços IP ou vice-versa são enviados a este servidor.

• **NetBEUI** (*Netbios Enhanced User Interface*) é um protocolo proprietário da Microsoft®, que acompanha todos os sistemas operacionais e produtos de redes, como o Windows 9x/ME, Windows NT/2000, LAN Manager, LAN Server, etc. Foi criado originalmente pela IBM, na época em que a IBM® e a Microsoft® possuíam uma parceria para a produção de sistemas operacionais e software (1985).

• **IPX/SPX** é o protocolo proprietário criado pela Novell® para o seu sistema operacional Netware, baseado no protocolo XNS (*Xerox Network Systems*). O funcionamento deste protocolo possui semelhança com o TCP/IP. O IPX (*Internet Packet Exchange*) é um protocolo que opera na camada de rede, equivalendo ao IP, enquanto que o SPX (*Sequenced Packet Exchange*) opera na camada de transporte, equivalente ao TCP.

4.1.5 **Estações de Monitoramento SCADA**

As Estações de Monitoramento Central são as unidades principais dos sistemas SCADA, sendo responsáveis pelo recebimento de informações geradas por estações remotas e por agir em conformidade com eventos ocorridos. Podem estar centralizadas em uma única estação de trabalho (PC) ou distribuídas por uma rede de computadores de modo a permitir a partilha de informações provenientes do sistema SCADA. Para aumentar o grau de segurança e confiabilidade, pode se configurar ao sistema uma estação de backup. Esta configuração consiste em duas estações SCADA idênticas, ambas conectadas aos mesmos CLPs (vide figura 4.1), mas uma estação roda como mestre, coletando os dados, enquanto a outra permanece em "*stand-by*". Quando a estação mestra pára de funcionar, a estação backup assume, comunicando com os CLPs e com as outras estações da rede. Depois de recuperada a estação mestre, a estação backup volta para "*stand-by*" e atualiza os dados históricos na mestra.

4.2 **Integração dos Componentes e Arquitetura do Sistema**

4.2.1 **Seleção do Sistema de Supervisão**

Em se tratando de sistemas automatizados complexos, surge a necessidade da criação de uma interface de maneira a facilitar o controle e monitoramento do sistema.

Pensando nisto adotou-se o controle feito com auxílio de um sistema de supervisão e controle (software).

O primeiro passo para desenvolvimento do projeto foi planejamento do supervisório a ser adotado. Um dos fatores que propiciou a implementação do software Wizcon (v 7.61) foi por ser um sistema sofisticado e adequado às novas tecnologias de mercado. Outros fatores foram o custo e recursos que, bem integrados, possibilitam que esta tecnologia desenvolvida permita ao sistema uma grande flexibilidade.

4.2.2 **Comunicação do Sistema de Supervisão**

A sistemática de implementação envolveu estudos na área de sistema de supervisão, controladores lógicos, sistemas operacionais, linguagens de programação, redes, etc.

O computador é configurado para trabalhar com um nome e número de uma estação de trabalho (Wizcon), e esta configuração é feita internamente no sistema supervisório e deve ser realizada com o nome diferente do adotado na configuração da máquina na rede, pois a nova configuração adotada no sistema supervisório é gerada somente para que este sistema trabalhe no endereço (IP) da máquina. As outras estações que estão com o supervisor instalado, quando devidamente configuradas e rodando em conjunto o sistema supervisor, são automaticamente reconhecidas o que possibilita a troca de informações entre elas.

Em nosso caso, o computador conectado diretamente ao sistema a ser controlado é adotado como Master, ou seja, o principal na rede de supervisão. O software de supervisão adotado foi Wizcon como mencionado anteriormente e descrito no capítulo anterior. O software de supervisão se comunica com a CLP, da marca Koyo (ver figura 4.9) através da saída serial RS 232 e com a utilização (através de uma seleção) do protocolo de comunicação TEXAS INSTRUMENTS SERIES 405. O protocolo selecionado para a comunicação incluirá, dentre outros itens, a velocidade de comunicação, porta de comunicação, porta sinal conectado, tempo de scan, etc.

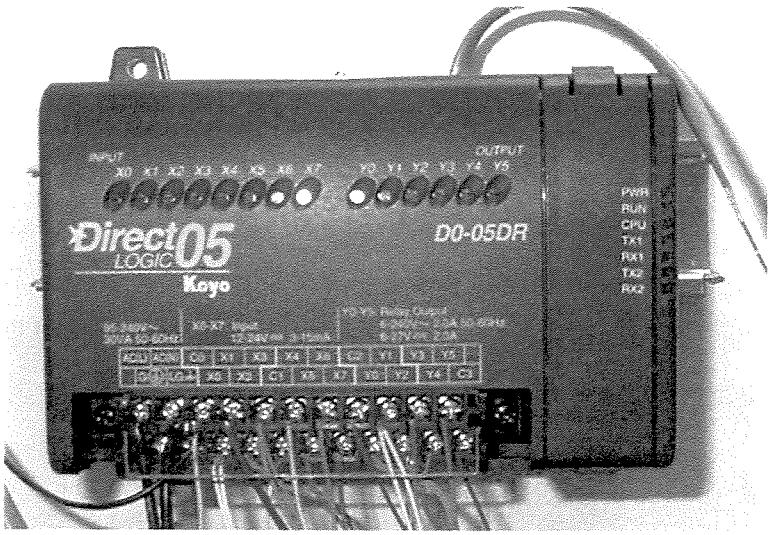


Figura 4.9 – CLP Koyo

O CLP da marca Koyo modelo Direct Logic D0-05DR têm as seguintes propriedades apresentadas na tabela 4.1.

Portas Seriais	02 (RS-232C)	
	Porta 1	**Porta 2**
Baud Rate	9600 bps (fixo)	300-38400bps (configurável)
Protocolo DirectNet	Slave	Master / Slave
Protocolo MODBUS	Slave	Master / Slave
Comunicação ASCII	Não	possui

Tabela 4.1 – Característica Koyo modelo D0-05DR

Na configuração de comunicação entre a CLP e o supervisório (PC-Wizcon), estabeleceu-se uma taxa de transmissão (*Braud Rate*) de 19200 bps. A taxa de transmissão é que determina a velocidade, expressa em bits por segundo (bps), da transmissão de dados. O padrão de comunicação adotado no supervisório deve ser o mesmo que o adotado para a CPU do CLP. Com a determinação do protocolo de comunicação existe a necessidade de criar blocos de endereçamento de memória para as respectivas variáveis de memória do CLP. Através dos blocos pode-se obter a configuração principal das entradas, saídas, temporizador etc. do CLP. Os blocos são obtidos através de uma documentação existente no Wizcon, as VPIs contêm uma referência de documentação para configurar com o protocolo selecionado o formato correto de endereço para as variáveis da CLP (*Address Format*). Neste caso, para o protocolo Texas a referência é a VPI, que irá fornecer toda a documentação necessária para configurar os blocos de endereço; as informações contidas nesta VPI se encontram no anexo III. Na tabela 4.2 são mostrados alguns blocos criados para a comunicação.

Endereçamento dos Blocos	
Entrada (Xn)	0140400
Saída (Yn)	0140500
Memória Interna (Cn)	0140600
Temporizador (Tn)	0141100
Temporizador em modo Analógico	0100000

Tabela 4.2 – Blocos de Comunicação

Com a configuração dos blocos pode-se configurar as variáveis do sistema, os "Tags", que estabelecem a comunicação com a CLP, endereços estes que estão atrelados aos seus respectivos blocos, ou seja, através do bloco de entrada configuram-se os formatos de endereços para todas as variáveis de entrada do CLP. Os Tags podem ser: "CLP" ou "Memory".

Os "Tags CLPs" significam que se originam dos blocos de comunicação descritos acima e se comunicam diretamente com a CLP. Os "Memory" (Dummy para o Wizcon) são aqueles que residem localmente no sistema supervisório, são próprios dele, criados para desempenhar funções em que substituem os Tags da CLP, podem ser utilizados, por exemplo, para realizar simulações de entradas e saídas sem a necessidade de uma CLP conectada ao sistema ou também ser usados como variáveis auxiliares no controle e supervisão.

Para verificar se está correta a comunicação e se o sistema está atualizando as variáveis de entrada e/ou saída existe um aplicativo no Wizcon que fornece estas informações – Multiple Tags.

A comunicação entre o Sistema Controlado e Monitorado (SCM) e o sistema de supervisão (software) como mostrado acima utiliza uma CLP, pois esta introduz no sistema uma maior segurança; no caso de uma falha do sistema operacional que esta "rodando" junto com o software de supervisão a CLP deixará o sistema operante até que a tarefa designada seja cumprida. A figura 4.1 ilustra a comunicação entre os sistemas.

4.2.3 **Arquitetura Cliente – Servidor (Client-Server)**

A arquitetura Cliente Servidor é um auxílio usado para o monitoramento e controle de sistemas localizados em diferentes locais. Segundo Blanc (2000) a base de dados do processo pode ser trocada sob a funcionalidade do Wizcon-DDE (Dynamic Data Exchange), entre as Estações de supervisão sobre a conexão Ethernet TCP-IP a qual está aberta adequadamente para filas de arquivos TCP-IP. Por outro lado, as Estações Supervisórias locais estão interconectadas através da rede Ethernet TCP-IP para a estação Wizcon-Server, mas não necessariamente são utilizadas como uma estação de supervisão. Nestes caminhos todos os dados locais, com eventos em tempo real, diversos "status" pré-definidos e tendências, também como relatórios diários, podem facilmente ser armazenados em um histórico de dados.

Além disso, a estação server (Master) implementada com o software Web Server, - em nosso caso foi utilizado o Apache Server [6] - também permite usuários monitorarem e/ou enviarem tarefas para o robô através de um Web Browser standard (Explorer ou Netscape). O Web Server é um software que gerencia as páginas em uma rede TCP-IP. Quando um navegador Web requisita uma página, ele envia à rede as informações referentes à página que ele deseja abrir. O Web Server recebe estas informações e envia a página ao navegador web (Web Browser) que está requisitando. Neste caso, se o protocolo que está sendo utilizado é HTTP que se localiza dentro da camada do protocolo TCP-IP, e este modo de comunicação utiliza-se da porta 80 do protocolo TCP-IP, o Servidor Web deve estar configurado para utilizar esta porta de comunicação.

É muito importante que os softwares navegadores Web possuam o Java Virtual Machine (JVM) habilitado em suas configurações internas, para que os aplicativos em Java (Applets) possam ser executados por estes navegadores Web (Browsers). Desse modo, a estação Wizcon Server está "rodando" um Web Server com as ferramentas para a execução da aplicação em qualquer localização, ou seja, em uma arquitetura WAN ou LAN, sem a necessidade que no computador do usuário ou cliente esteja instalado um sistema SCADA. A figura 4.1 mostra o funcionamento simplificado do navegador Web requisitando dados de um sistema SCADA conectado ao SCM. A figura abaixo mostra aplicação da Telerobótica com a utilização da Internet como meio de comunicação baseada na arquitetura Cliente Servidor, utilizando o protocolo HTTP (Hypertext Transfer Protocol), no qual um cliente ou usuário faz a requisição

através de um navegador Web com a interface de comunicação robô utilizando um sistema SCADA.

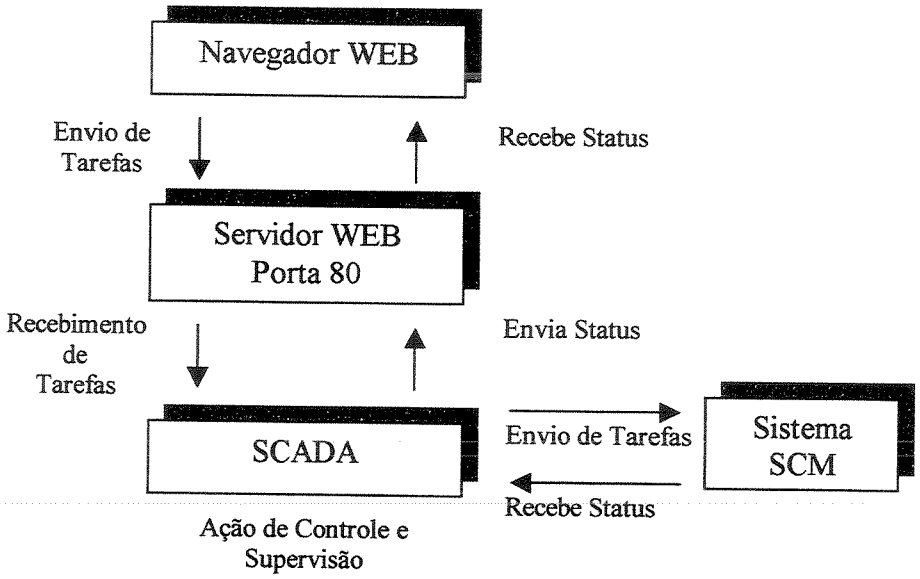


Figura 4.10 – Teleoperação utilizando a Arquitetura Cliente/Servidor

4.2.4 **Interface com o usuário**

Através da área gráfica do sistema pode-se criar um layout bem elaborado dos dispositivos e informações necessárias para se obter maior funcionalidade do sistema. Este Layout pode ser configurado segundo perfis de usuários. Isto significa que determinado usuário quando se inscreve e digita uma senha, automaticamente poderá ter acesso limitado ou não a certos controles ou visualizações

Com o layout elaborado gera-se uma página html na qual os "mini aplicativos" são escritos em Java (Applet).

Através da visualização do Supervisório via navegador Web (browser) como mostrado nas figuras 5.17 e 5.21 no capítulo 5, o usuário poderá estar enviando tarefas a serem cumpridas,

como também durante todo processo estará recebendo informações de monitoramento ("status") destas tarefas.

No Layout existe uma chave que pode ou não acionar o supervisório; caso esta chave esteja desligada no módulo automático nenhuma tarefa poderá ser enviada, o usuário no módulo automático apenas tem "status" das variáveis (monitorando) e não de controle do sistema SCM.

4.2.5 **Sistema de aquisição de imagens**

O sistema de aquisição de imagens está conectado a uma câmera do tipo WebCam USB modelo Intel Pro Pc (ver figura 4.11) da fabricante internacional Intel.

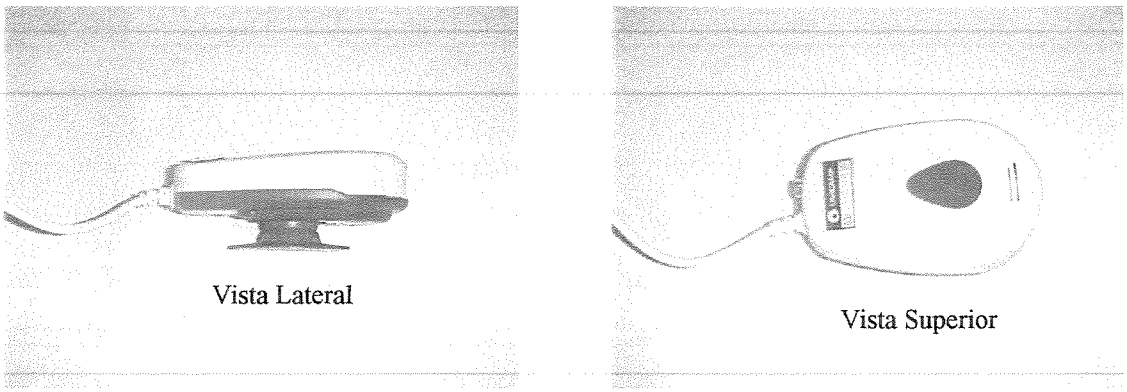


Figura 4.11 – Câmera WebCam (Intel Pro PC)

A sigla universal Serial Bus (USB) é um padrão recente já consolidado e usado largamente na conexão de dispositivos com computadores. A versão 1.0, que é o padrão de conexão da câmera utilizada, é suportada pela maioria dos computadores, facilmente adaptada em outros, e possui uma taxa de transferência de dados na ordem de 12 Mbps (mega bytes por segundo). Uma nova versão, a 2.0, mais rápida, está surgindo com velocidade de transmissão de até 480 Mbps. Esta nova versão suporta a antiga USB 1.0 e vai introduzir uma significativa melhora na velocidade de captura de imagens em tempo real, acarretando uma grande diminuição no tempo de retardo das transmissões via Internet.

As imagens estão sendo transmitidas em tempo real, através de uma seqüência de imagens JPEG, que utiliza taxa de compressão muito baixa quando comparada com MPEG ou Real Vídeo (estes dois últimos são mecanismos utilizados em transmissão de vídeo). A principal vantagem da sistemática utilizada é que se pode enviar as imagens de vídeo utilizando o mecanismo server-push do Servidor HTTP diretamente para o cliente WWW, como o Microsoft® Internet Explorer ou Netscape™, sem a necessidade de se utilizar um software especial ou um *plugin* para receber o formato de vídeo.

Para o funcionamento desta sistemática toda uma programação (ver apêndice A) deve ser criada na Página Web (HTML – [7]) com auxílio de algum editor para esta funcionalidade. A Web página deve conter uma programação para que o software de captura de imagens, "Server WebCam" que faz a fução de server-push, possa interagir com a página, para que o usuário através do navegador Web receba de forma correta e constante as imagens providas do sistema de controle e supervisão.

A página Web principal quando requisitada carrega simultaneamente duas páginas HTML (Web) na qual uma contém o "layout" do sistema de supervisão e controle e a outra página é responsável pelo controle das imagens. Esta sistemática de separação das páginas foi desenvolvida para possibilitar aos sistemas (captura de imagens e supervisório) uma independencia funcional, gerando uma maior segurança e confiabilidade ao painel de controle.

Capítulo 5

Aplicações Utilizando Sistemas Supervisórios e Telerobótica

Nos capítulos anteriores foi dada ênfase às ferramentas de integração de sistemas automatizados direcionadas a aplicações de Supervisão e Controle e Telerobótica; neste capítulo serão implementadas algumas aplicações direcionadas as áreas de Automação Industrial e Telerobótica, que permitirão a validação experimental de conceitos apresentados anteriormente.

Com esta finalidade utilizaremos um software de Supervisão Industrial, que permitiu integrarmos dispositivos automatizados com partes operativas diferentes, a partir da especificação funcional das aplicações propostas, tornando o problema em estudo genérico, fato que teve como resultado a possibilidade de integração de diferentes dispositivos automatizados.

Assim são apresentados exemplos constituídos de partes operativas diferentes, mas todos baseados na mesma arquitetura de controle e comunicação. Os sistemas implementados são os seguintes:

- Implementação de um Magazine de montagem por seleção de cores junto à plataforma PIPEFA;
- Aplicação exemplificativa direcionada a Telecirurgia com auxílio de um manipulador robótico em que o controle e monitoramento são realizados à distância;
- Implementação de um sistema de supervisão e controle em um robô industrial.

5.1 **Implementação de um Magazine de montagem por seleção de cores**

Neste item será apresentada a estrutura da Plataforma Industrial para Pesquisa, Ensino e Formação em Automação (PIPEFA), no qual será dada ênfase à implementação de uma nova linha de montagem que futuramente estará integrando a PIPEFA.

O objetivo desta implementação é fornecer à Plataforma PIPEFA uma nova célula de montagem por seleção de cores, no qual usuário através de uma interface gráfica terá acesso ao controle desta célula, bem como à informações em tempo real dos parâmetros seqüenciais de montagem.

A Plataforma PIPEFA foi um projeto desenvolvido para ensino e pesquisa e formação tecnológica na área de automação industrial, proporcionando o desenvolvimento de aplicações em automação que possibilitem melhoria de desempenho, redução de custos, flexibilidade de operação e aumento da qualidade em um sistema de manufatura. É composta por componentes industriais, tais como sensores magnéticos, sensores ópticos, CLP, cilindros e eletro-válvulas pneumáticas e leitor código de barra, simulando uma situação real.

A plataforma realiza operações típicas de Sistemas Automatizados de Produção, como carregamento, descarregamento, operações de montagem e desmontagem, e inspeção.

O projeto PIPEFA mostrado na figura 5.1 estabeleceu um acordo de cooperação entre o Laboratório de Automação Integrada e Robótica (LAR) da Unicamp, O Instituto de Automação da Fundação CTI e o LIISI (Laboratorie d'Inngénierie Integrée dês Systèmes Industriels - França) de acordo com Rosário (1999). Esse acordo permitiu o aprofundamento de conhecimentos, realizações no ponto de vista material e sua aplicação em Sistemas Automatizados de Produção, dentro dos objetivos de geração de uma massa crítica de pesquisadores no domínio de Automação Integrada.

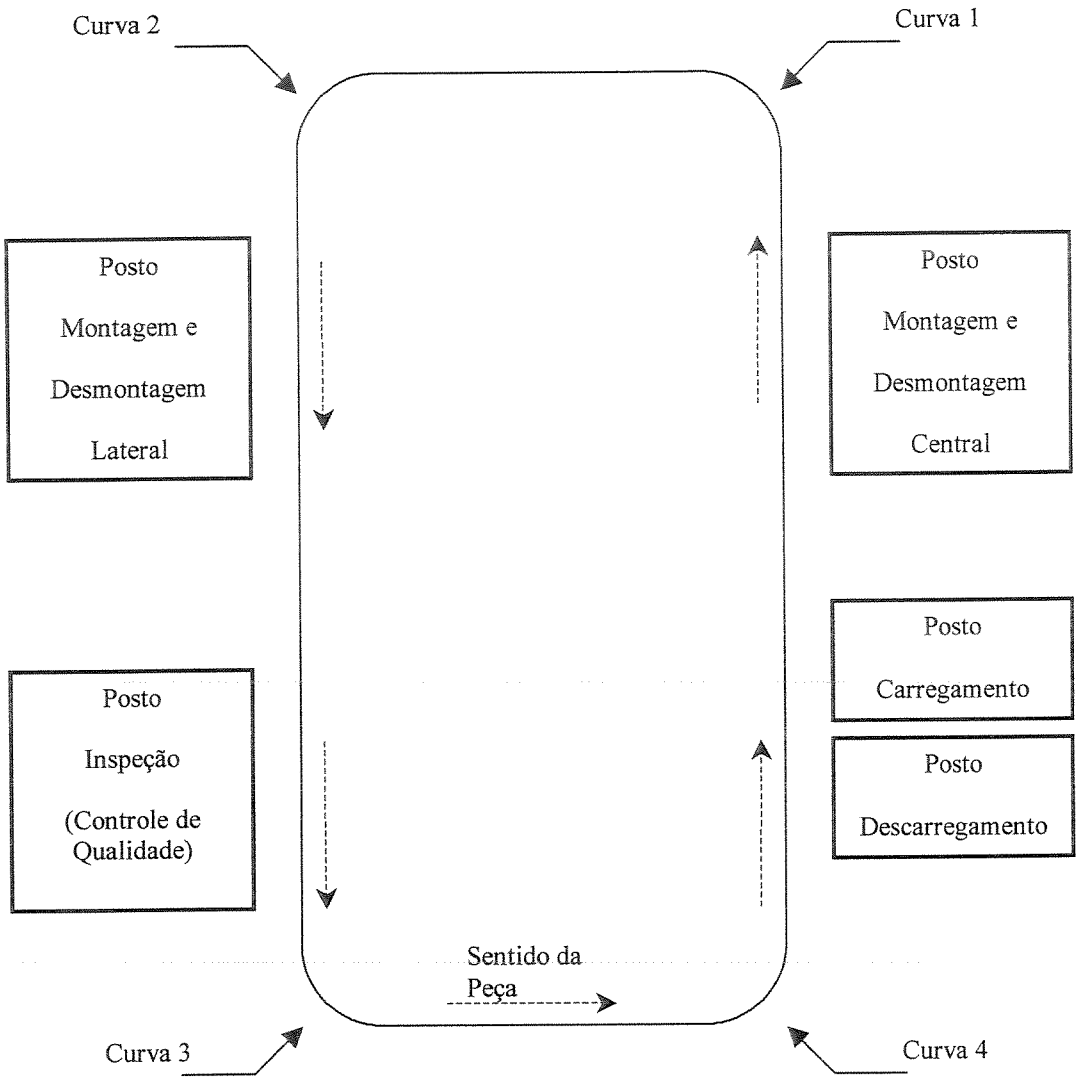


Figura 5.1 – Estrutura Básica PIPEFA

5.1.1 **Descrição da PIPEFA**

• Posto de Carregamento

O posto de carregamento, apresentado na figura 5.2, armazena as placas e as fornece à esteira transportadora para a confecção do produto, por sistema mecânico (com acionamento pneumático) que as posiciona sobre um elevador (sistema de transferência). Possui verificação da qualidade de placas-base armazenadas, que indica se o número mínimo foi atingido.

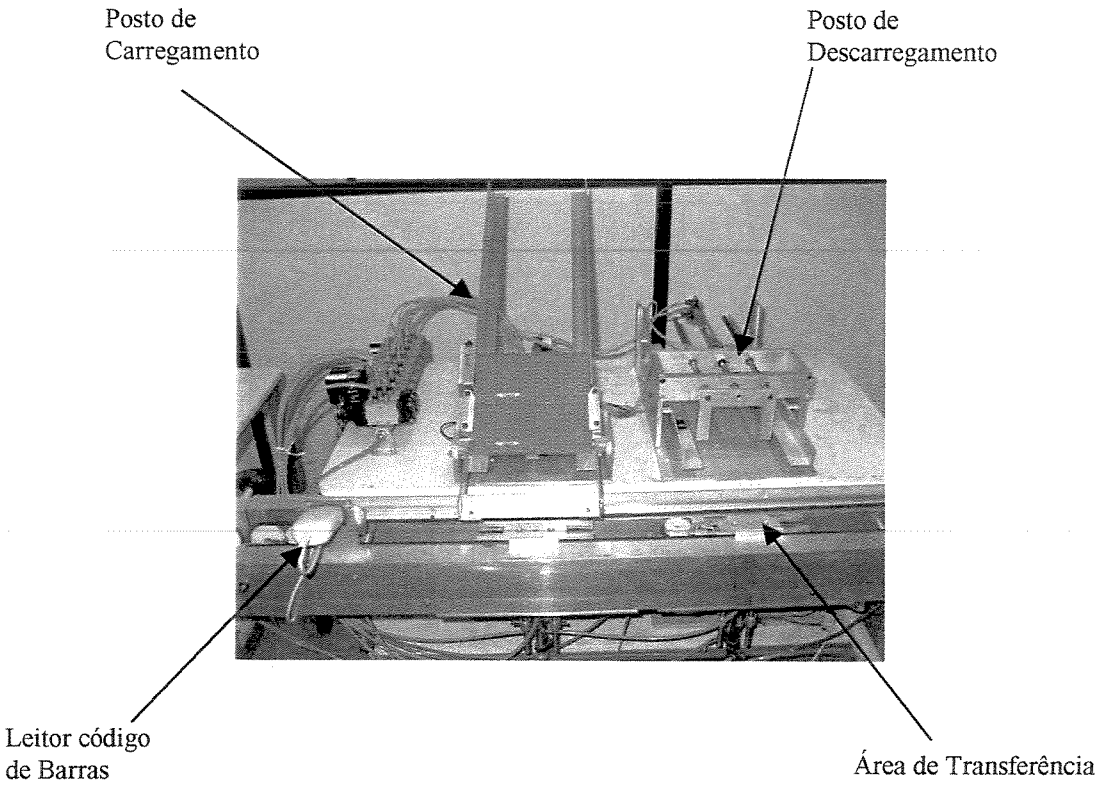


Figura 5.2 - Posto de Carregamento

• Leitor de Código de Barras

Trata-se de um leitor de Código de Barras comercial, conforme apresentado na figura 5.2, com saída serial (RS-232) conectado diretamente em uma das portas seriais do CLP. Faz leitura do código de barras existente nas placas bases, definindo o produto a ser confeccionado. O código lido é enviado ao Sistema de Transferência, que determinará a utilização dos Postos de Montagem Central e/ou Lateral.

• Posto de Montagem/Desmontagem Central

Este posto realiza as operações de montagem e desmontagem de cubos na posição central das placas-base, nos dois níveis possíveis, conforme apresentado na figura 5.3.

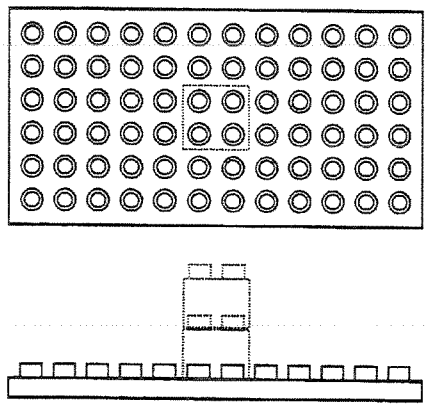


Figura 5. 3 – Possibilidades de Operação no Posto de Montagem/Desmontagem Central

Através de sistema mecânico (com acionamento pneumático) recebe a placa fornecida pelo elevador (sistema de transferência), posiciona-a na área de atuação do sistema de montagem/desmontagem e executa as operações necessárias, devolvendo-a ao elevador que a deposita novamente na esteira transportadora. Possui verificação da quantidade de cubos para montagem, que indica se o número mínimo foi atingido.

A operação deste posto está apresentada na figura 5.4.

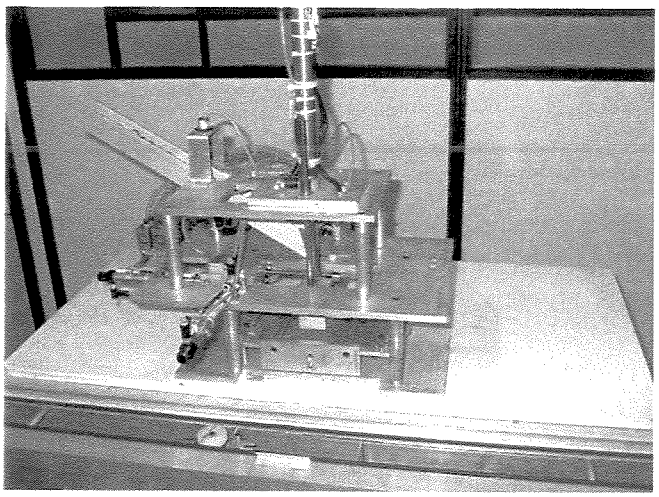


Figura 5.4 – Posto de Montagem/Desmontagem Central

• Posto de Montagem/Desmontagem Lateral

Este posto realiza as operações de montagem e desmontagem de cubos nas posições laterais (à direita e à esquerda) das placas-base, nos dois níveis possíveis, conforme apresentado na figura 5.5. Através de sistema mecânico (com acionamento pneumático) recebe as placas-base fornecida pelo elevador (sistema de transferência), posiciona-a na área de atuação do sistema de montagem/desmontagem e executa as operações necessárias, retornando ao elevador que a deposita novamente na esteira transportadora. Possui verificação da quantidade de cubos para montagem, que indica se o número mínimo foi atingido. A operação deste Posto é mostrada na figura 5.6.

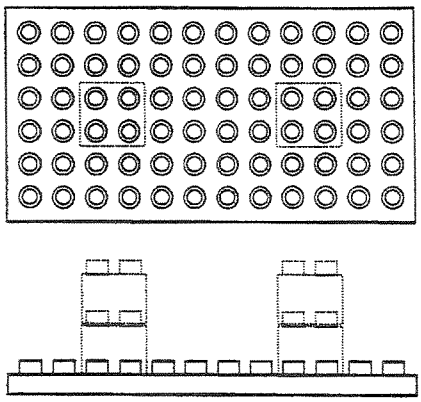


Figura 5.5 – Possibilidades de Operações no Posto de Montagem/Desmontagem Lateral

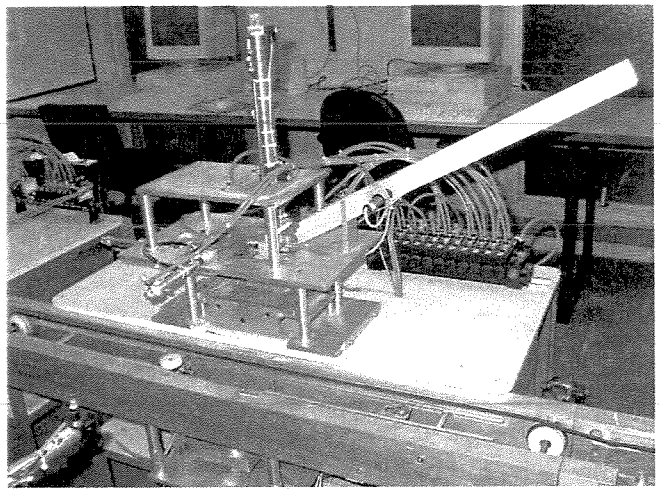


Figura 5.6 – Posto de Montagem/Desmontagem Lateral

- Posto de Inspeção

Este posto pode ser visualizado na figura 5.7. Nele se realiza a inspeção do produto confeccionado, o resultado é enviado ao Sistema de Transferência. Através de sistema mecânico (com acionamento Pneumático) recebe o produto confeccionado (placas-base e respectivos cubos) fornecido pelo elevador (sistema de transferência), posiciona-o na área de atuação do sistema de inspeção e, através de sensores ópticos, verifica a montagem (existência) de cubos em cada posição possível. Finalizada a inspeção, retorna o produto confeccionado ao elevador que o deposita novamente na esteira transportadora.

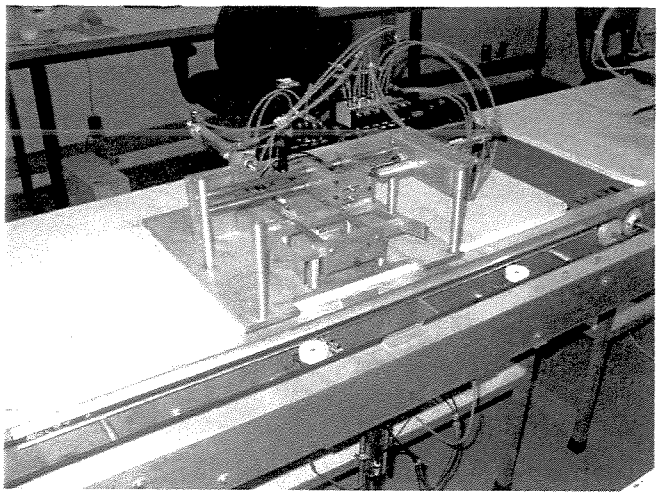


Figura 5.7 – Posto de Inspeção

- Posto de Descarregamento

Através de sistema mecânico (com acionamento pneumático), o Posto de Descarregamento recebe o produto confeccionado fornecido pelo elevador (sistema de transferência), para ser rejeitado (produto "não conforme" e "irreparável") ou descarregado (produto 100% conforme), sendo armazenado no próprio Posto. Caso o produto tenha sido classificado como "não conforme", mas com possibilidade de ser reparado, o Posto de Descarregamento não é acessado e o produto é transportado diretamente ao Posto de Montagem e Desmontagem Central e/ou Lateral, passando posteriormente por uma nova inspeção. Possui verificação da quantidade de produtos armazenados, que indica se o número máximo foi atingido. Este posto é representado pela figura 5.8.

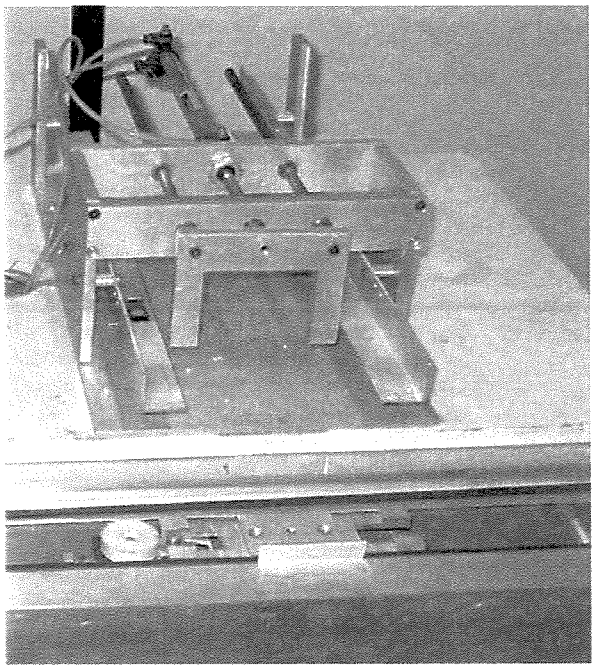


Figura 5.8 – Posto de Descarregamento

• Sistema de Tranferência

O Sistema de Transferência é composto por:

- Esteira transportadora dividida em 4 seguimentos acionados individualmente por motores de corrente contínua;
- Elevadores da Base, presentes em todos os Postos da PIPEFA, fazem a interface física entre o Sistema de Transferência e cada um dos Postos. São acionados por cilindros pneumáticos. O elevador da Base do Posto de Montagem/Desmontagem Lateral possui também um cilindro pneumático rotativo, para rotação das placas-base de 180°. Todos estes cilindros pneumáticos possuem sensores magnéticos de posicionamento (avançado e recuado / direita e esquerda);
- Posicionadores da Base, presentes no Leitor de Código de Barras e em todos os Postos da PIPEFA, exceto no de Carregamento, funcionam como 'stop' para as placas-base, posicionando-a corretamente para atuação dos Elevadores ou do Leitor de Código de Barras. Também são utilizados para impedir o avanço das placas-base se o Posto seguinte estiver ocupado. São acionados por pequenos cilindros pneumáticos e não possuem sensores;

- Verificadores de Base, presentes no leitor de Código de Barras, nas curvas da esteira Transportadora e em todos os Postos da PIPEFA, exceto no carregamento. São sensores magnéticos que indicam a presença das placas-base, sendo atuados por um pequeno disco metálico existente na parte inferior das placas-base.

Além de controlar cada um destes componentes, o sistema de Transferência gerencia todo o fluxo da PIPEFA. Inicia a operação de cada Posto definindo as operações que devem ser executadas. Também controla o Leitor de Código de Barras e faz a classificação do produto confeccionado, determinando seu destino final. A figura 5.9 apresenta o sistema de Transferência.

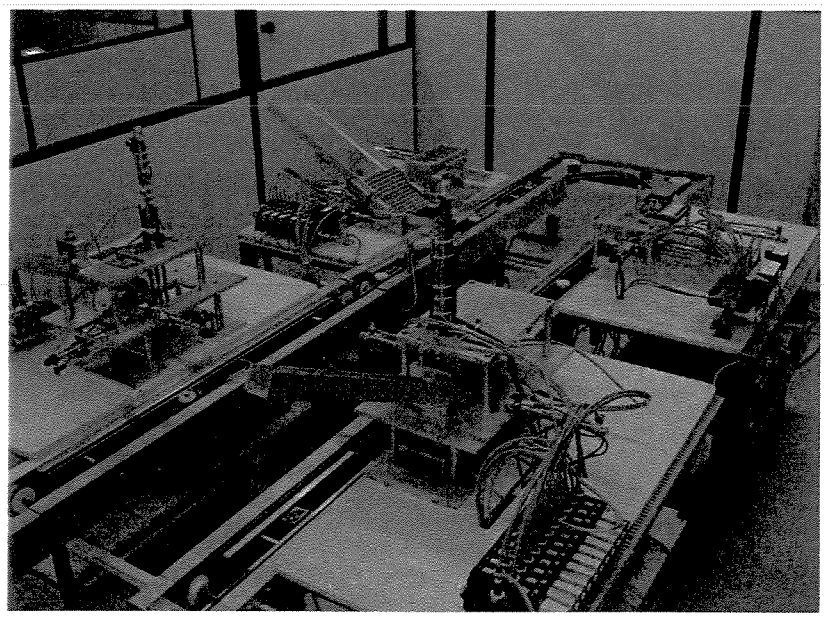
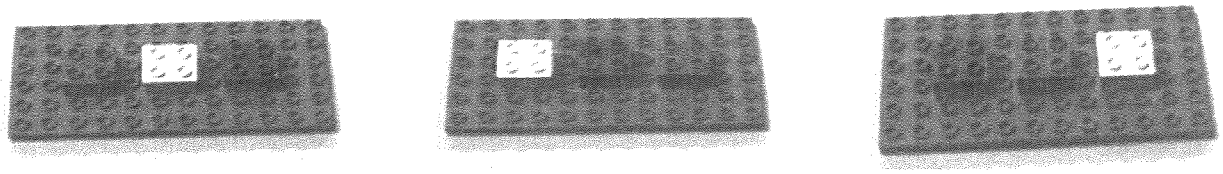


Figura 5.9 – Plataforma PIPEFA (Sistema de Transferência)

5.1.2 **Descrição da Célula de Montagem com Seleção de Cores**

A célula que está sendo integrada a PIPEFA deverá ser capaz de montar três produtos diferentes, que estão mostrados na figura 5.10, em que será obedecida uma seqüência de cores pré-estabelecida na qual se dará a diferenciação dos produtos.



Produto 1 **Produto 2** **Produto 3**

Figura 5. 10 – Seleção dos Produtos

Para isso será necessário um controle sobre esta célula para possibilitar a seleção do produto a ser montado.

A célula de montagem mostrada na figura 5.11 possui sensores e atuadores que possibilitam a leitura e atualização dos dados fornecidos ao CLP. A unidade lógica de controle está baseada no CLP que contém programas (ver apêndice B) gravados em sua memória para controlar as funções lógicas da célula de montagem.

Para um melhor controle da célula de montagem necessita-se de uma central de comando que permita a visualização dos sinais provindos da célula de montagem, a escolha do produto a ser montado e que exista um comando capaz de iniciar a montagem ou interromper o processo antes da montagem iniciar; este painel de comando pode ser visualizado na figura 5.12.

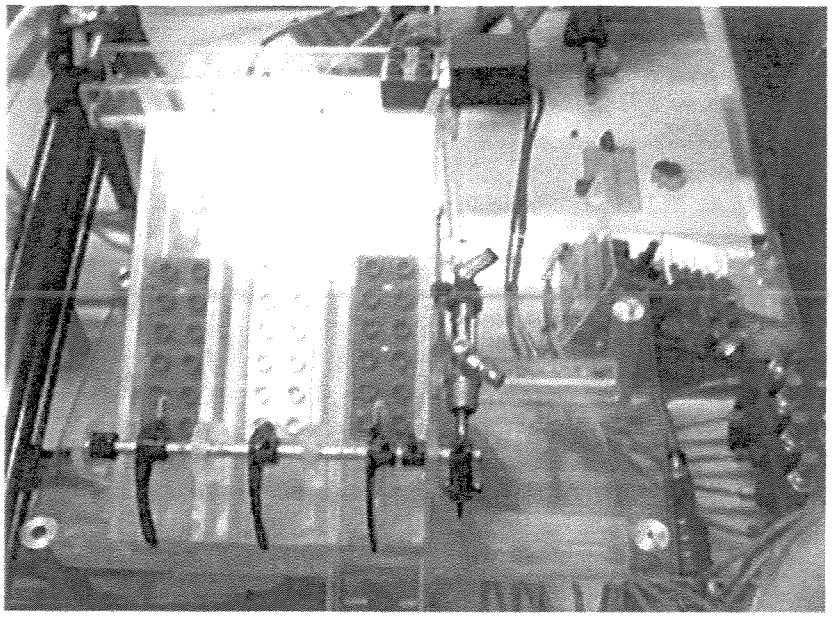


Figura 5.11 – Célula de Montagem por Cores

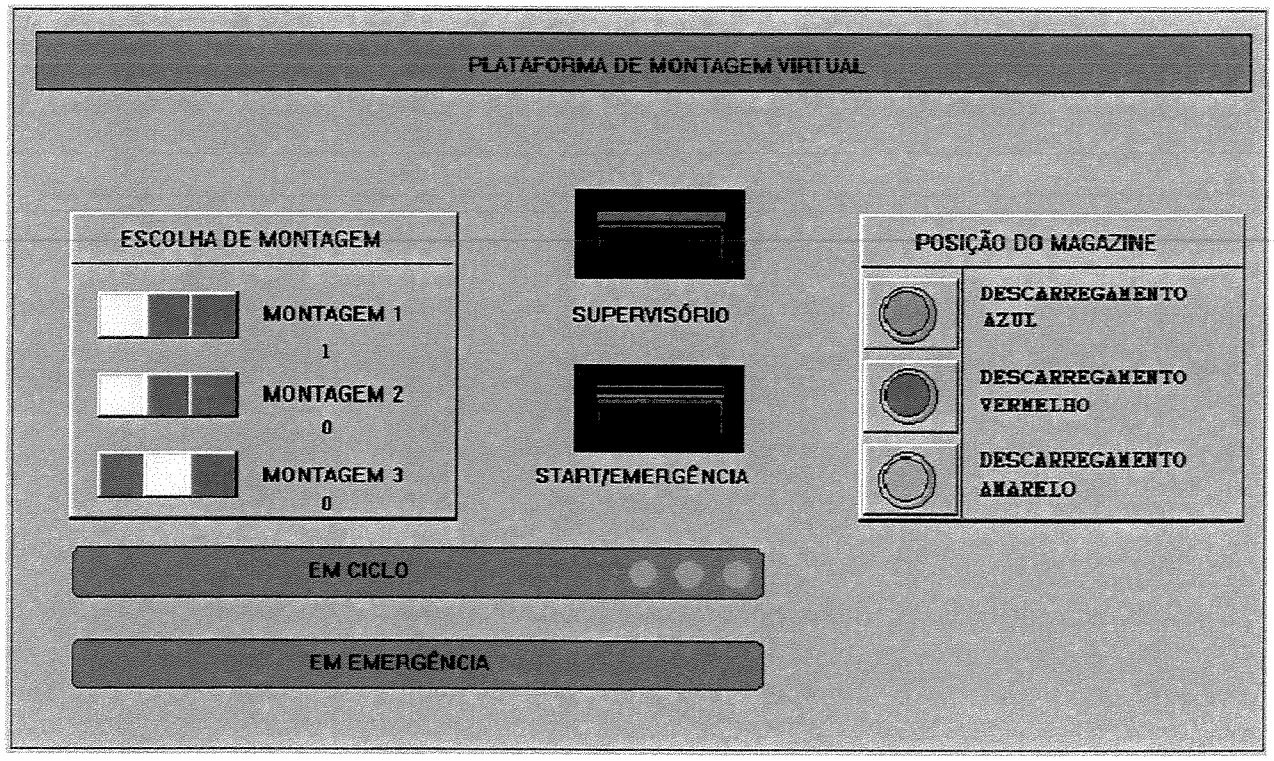


Figura 5.12 - Sistema de controle e Supervisão da Célula de Montagem

Após a conclusão da implementação da nova linha de montagem, a Plataforma PIPEFA ampliará o número de produtos em sua linha de montagem e, juntamente com a nova tecnologia empregada na elaboração da nova célula - a PLATAFORMA PIPEFA - poderá ser futuramente controlada e monitorada à distância.

5.2 **Aplicação Direcionada a Telecirurgia**

Telecirurgia é a utilização de recursos de informática, telecomunicações e mecatrônicos para a transmissão remota de dados biomédicos e para o controle de equipamentos biomédicos à distância.

Na telecirurgia a cirurgia é efetuada com acompanhamento de equipe médica localizada remotamente e que, através de um interfaceamento visual, recebe imagens em fluxo contínuo, isto é, as imagens chegam em tempo real. A telecirurgia pode ser considerada como um ramo da telemedicina.

A Telemedicina surgiu em 1960 da necessidade de atendimento dos astronautas na corrida espacial. Devido à impossibilidade de envio de médicos especialistas e equipamentos médicos com os astronautas, a NASA obrigou-se a desenvolver técnicas e equipamentos para o atendimento à distância de quaisquer problemas de saúde e para a monitoração dos sinais vitais de cada astronauta. Outro grande impulsionador da Telemedicina foi a necessidade de atendimento de prisioneiros nas penitenciárias. As distâncias destas penitenciárias dos centros clínicos, aliadas ao risco no atendimento dos criminosos, criou uma resistência das instituições de saúde para este tipo de serviço.

Na telecirurgia, como foi mencionado acima, é necessário o uso da telecomunicação e de equipamentos que possibilitem a transmissão em tempo real e de forma interativa dos sinais de vídeo e áudio. A partir desta informação utilizou-se o sistema implementado, parte de controle, com auxílio de elemento robótico, para demonstrar algumas das ferramentas necessárias para uso da telecirurgia e, com a introdução de um microcomputador industrial, assegura-se uma maior confiabilidade ao sistema.

5.2.1 **Descrição do Elemento Robótico**

Este elemento robótico pode ser apresentado através do ROBIX^{TM} que é um kit didático para ensino e aprendizagem de robótica e que pode ser montado em diversas configurações. Neste trabalho, adotou-se uma configuração que se assemelha a um braço mecânico, ou manipulador robótico, com cinco Graus de Liberdade.

Sua programação é dada através de software próprio. A interface de programação é de estilo industrial, sendo operada por teclas ou mouse. Ou seja, os comandos podem ser digitados ou operados por aprendizagem. Possui recursos de debug (execução passo-a-passo e janela de mensagens de erros) e de visualização de status do robô e dos seus sensores. O editor de programas possui todos os recursos de um editor de texto profissional (cortar, colar, salvar, salvar como, etc.) com janelas que ajudam na sua operação.

O Device Driver, tem a função de fazer a interligação, em nível de hardware, da Interface de Programação com o Módulo de Controle (Adapter). Também permite a utilização de linguagem de alto nível para comunicação (Basic, C) e o programa RCS-6, C e QBASIC, sendo que nestas duas últimas é necessária a inclusão de bibliotecas. Permite também que cada servomotor opere segundo parâmetros estabelecidos pelo usuário como aceleração, desaceleração, posição inicial, etc., interpolando linearmente os movimentos dos servomotores.

A figura 5.13 apresenta o ROBIX^{TM} na configuração utilizada, em que cinco servomotores são responsáveis pela movimentação dos cinco Graus de Liberdade, enquanto um servomotor comanda a abertura e fechamento da garra.

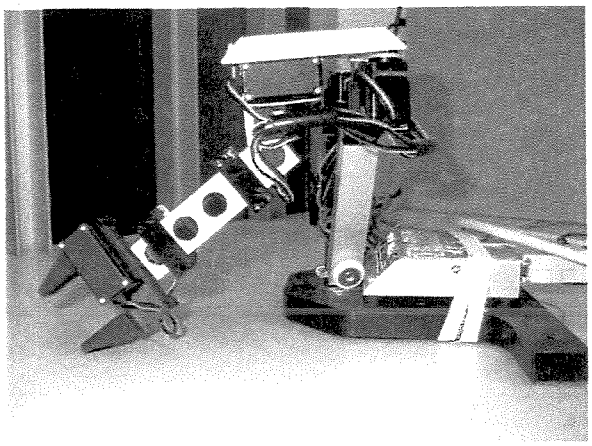


Figura 5.13 – Robix™

Desta forma teremos a interface de programação que cria uma janela de edição de programas de controle e o device driver sendo instalado na memória para ser ativado automaticamente pela interface de programação, sempre que for necessária a comunicação com o Módulo Controlador.

Todos os arquivos apresentados são carregados e executados pelo arquivo rbx.bat. A tela inicial de programação é apresentada na figura 5.14:

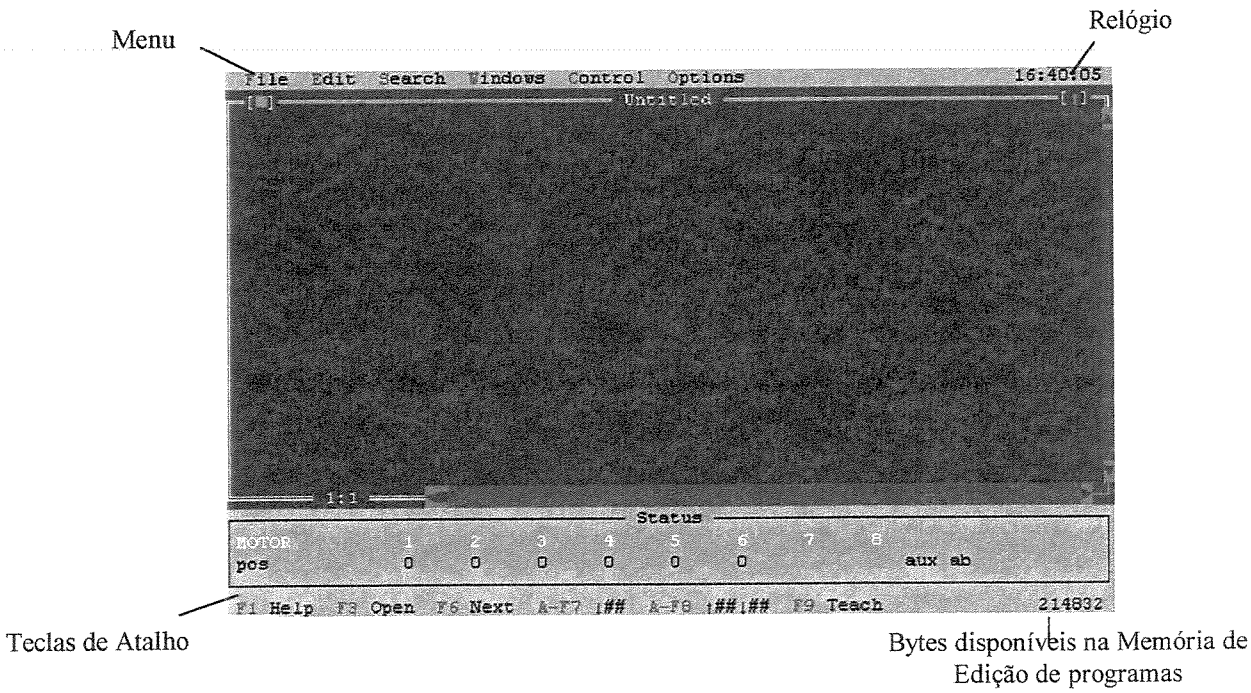


Figura 5.14 - Tela inicial da interface de programação do ROBIX™

A programação do manipulador robótico pode também ser realizada conforme comentado anteriormente, através de aprendizagem em que cada grau de liberdade é operado individualmente (Teach In), como ilustrado na figura 5.15.

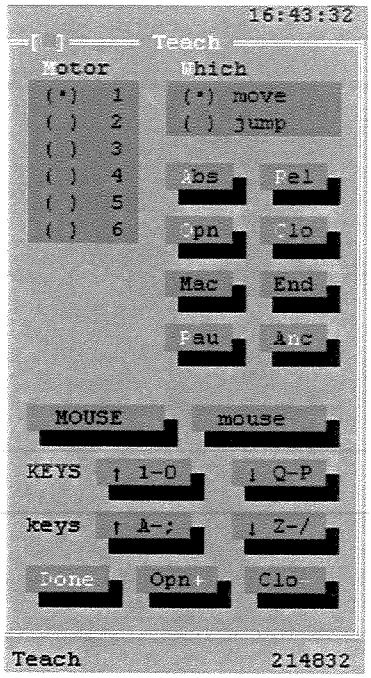


Figura 5.15 - Tela Teach de programação do ROBIX

A janela apresentada na figura 5.15 é o "painel de controle" do ROBIX^{TM}. Com esta janela aberta é possível mover cada junta do robô, individualmente ou conjuntamente, e gravar as posições assumidas para ir criando um programa de controle.

Desta forma, o teclado e/ou o mouse são transformados num "Teach Pendant", uma caixa de programação de posições para o robô, sendo possível mover os servomotores de cada junta em duas velocidades: a rápida e a lenta.

Os principais parâmetros utilizados para o comando dos servomotores são:

- pos: posição corrente
- maxspd: máxima velocidade durante um movimento (move)
- accel: aceleração durante um movimento (move)

- decel: aceleração durante um movimento (move)
- minpos: min posição depois de um movimento (move ou jump)
- maxpos: max posição depois de um movimento (move ou jump)
- initpos: posição depois de iniciar ou movimentar todos os servomotores para a posição inicial
- power: liga / desliga os servomotores
- invert: inverte as coordenadas dos servomotores
- p0pos: posição 0 (físico).

A figura 5.16 esquematiza o funcionamento da bancada para simulação Telecirúrgica. Foram configuradas no Robix trajetórias para simulação de uma intervenção cirúrgica.

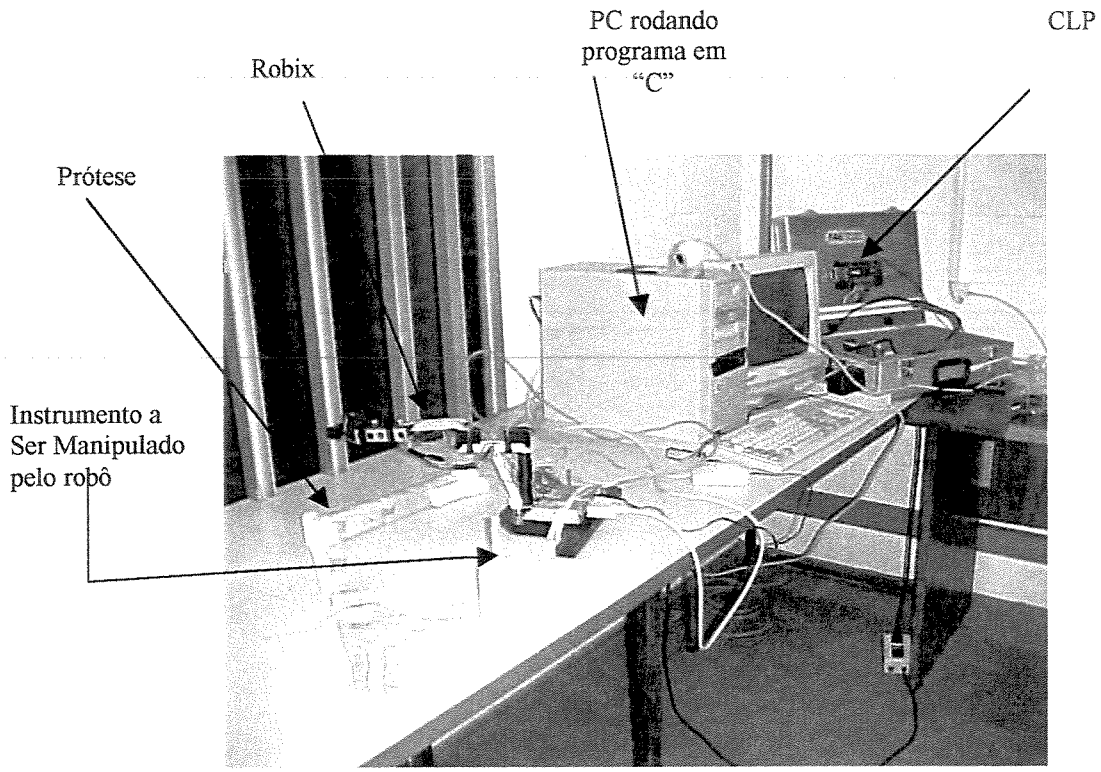


Figura 5.16 – Bancada de Simulação Experimental Voltada à Telecirurgia

Com um sistema via Internet mostrado na figura 5.17, o usuário pode receber imagens do local onde está acontecendo a intervenção cirúrgica e controlar certas funções para uma maior funcionalidade do sistema.

5.2.2 "Modus Operandi" do Sistema

A arquitetura de comunicação está descrita no capítulo 4 e mostrada na figura 4.1. O sistema funciona com o auxílio do programa (tela da figura 5.15) do Robix em que foram criadas trajetórias que simulam uma intervenção cirúrgica; o usuário tem uma seqüência pré-definida de trajetórias, as quais são enviadas através de um navegador Web, que pode ser visualizado na figura 5.17. As trajetórias estão estabelecidas da seguinte maneira - na trajetória A, o robô desloca-se até o instrumento de auxílio na intervenção e com auxílio das garras segura o instrumento e, seqüencialmente, toma uma posição determinada de espera para aguardar o recebimento da nova trajetória. A trajetória (B), seguinte, é propriamente a que simula uma intervenção cirúrgica. Após o cumprimento correto da trajetória B, o robô volta à posição anterior para aguardar o envio da nova trajetória, que seria a "C", e, recebendo esta trajetória, vai em direção ao recipiente onde deposita o instrumento utilizado para depois voltar à posição inicial.

Quando o usuário envia as trajetórias com o auxílio de um navegador Web, a interface de controle se comunica com as E/S do Robix através de um CLP, no qual está sendo executado um programa (ver apêndice C) para o estabelecimento das funções lógicas necessárias à execução das tarefas. O interfaceamento que possibilita a comunicação do Robix com o CLP é feito em um programa em "C" (ver anexo III) que está paralelamente rodando junto ao programa de controle do Robix, ambos no mesmo computador (ver figura 4.1).

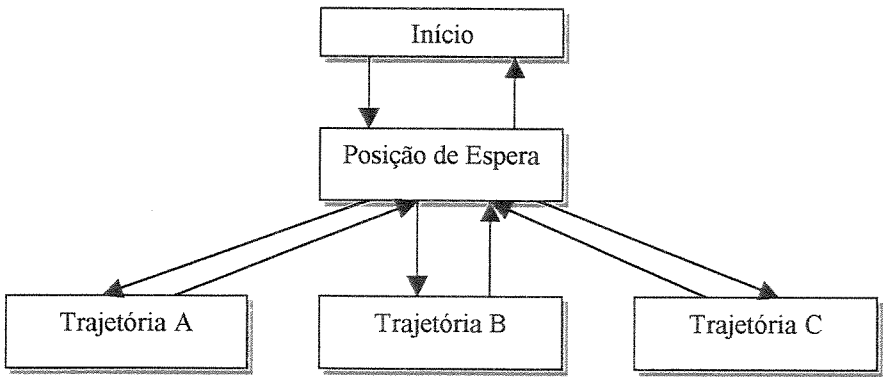


Figura 5.17 – Estruturação das Trajetórias

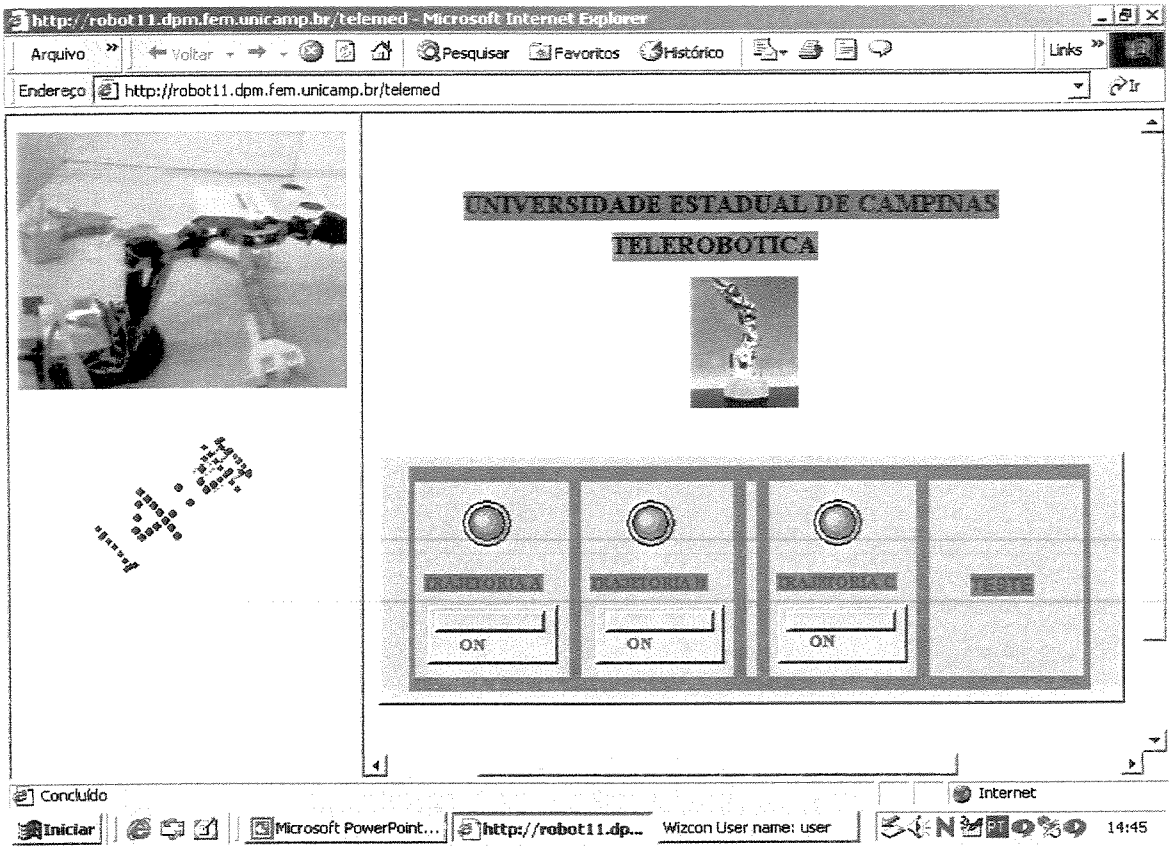


Figura 5.18 – Telemedicina Aplicada Via Internet

5.3 **Implementação de um sistema de supervisão e controle em um robô industrial**

A figura 5.19 mostra o robô industrial Asea IRB 1400 que possui seis graus de liberdade, no qual a comunicação com o controle supervisório está estabelecida da seguinte forma: o sistema supervisório, através da comunicação RS232, envia ao e/ou recebe do CLP os parâmetros requisitados pelo usuário através de um navegador Web (figura 5.23), no CLP através de uma lógica de programação similar à utilizada no sistema anterior na simulação de telecirurgia, ou seja, é estabelecida uma transmissão de dados para as E/S do robô que, através destas informações, executa as tarefas que estão pré-programadas. A estrutura das tarefas se encontra na figura 5.22. Os dados são transmitidos ao robô, para sua unidade de controle, memória e processamento que será responsável pela execução e controle das tarefas ligadas ao robô. Este, após o término das tarefas envia sinais para indicar a finalização destas tarefas e, assim, o usuário obtém as informações necessárias para o controle e monitoramento do sistema.

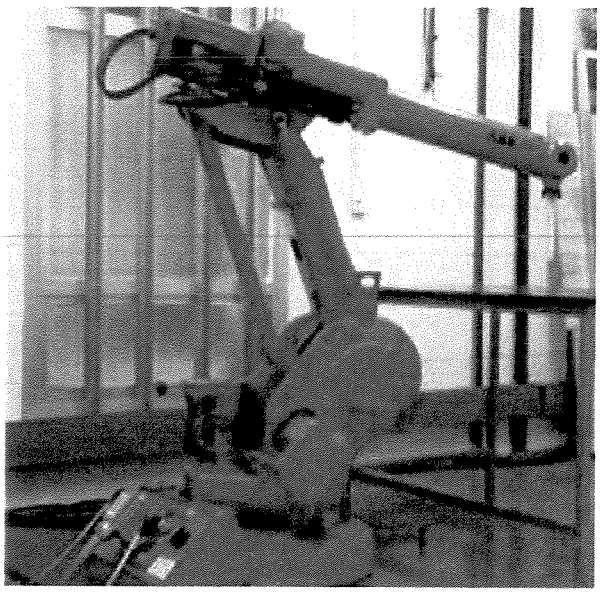


Figura 5.19 – Robô ASEA Irb 1400

O sistema pode ser visto, tanto em uma rede interna como via Internet, com o uso de um navegador Web. O usuário faz a requisição do painel de controle via navegador Web e a imagem do painel pode ser vista no computador, conforme a figura 5.23. O sistema de arquitetura na rede e transmissão de imagens está descrito no capítulo 4.

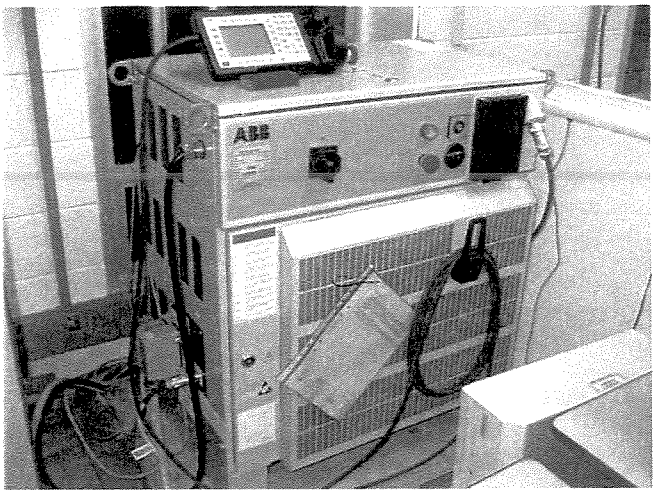


Figura 5.20 – Unidade de Controle, Memória e Processamento do Robô ABB

Para auxílio na criação das tarefas foi utilizado o Teach InBox (figura 5.21), dispositivo de auxílio de controle do robô, que serve para inserir a programação (ver apêndice D), tarefas para o robô e, subseqüentemente, envia para a unidade de processamento e memória mostrada na figura 5.20.

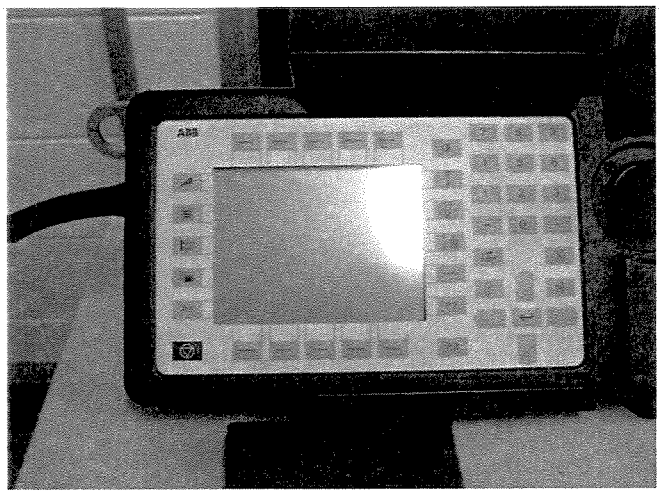


Figura 5.21 – Teach Inbox

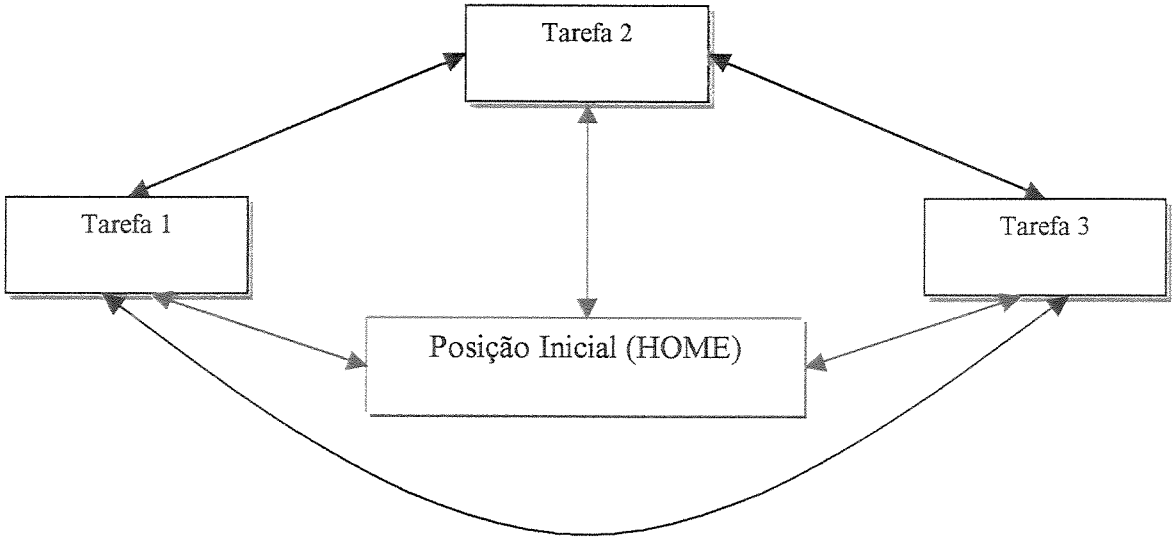


Figura 5.22 – Estrutura das Tarefas

(tarefa 4 = tarefa 1 → tarefa 2 → tarefa 3 → Posição Inicial)

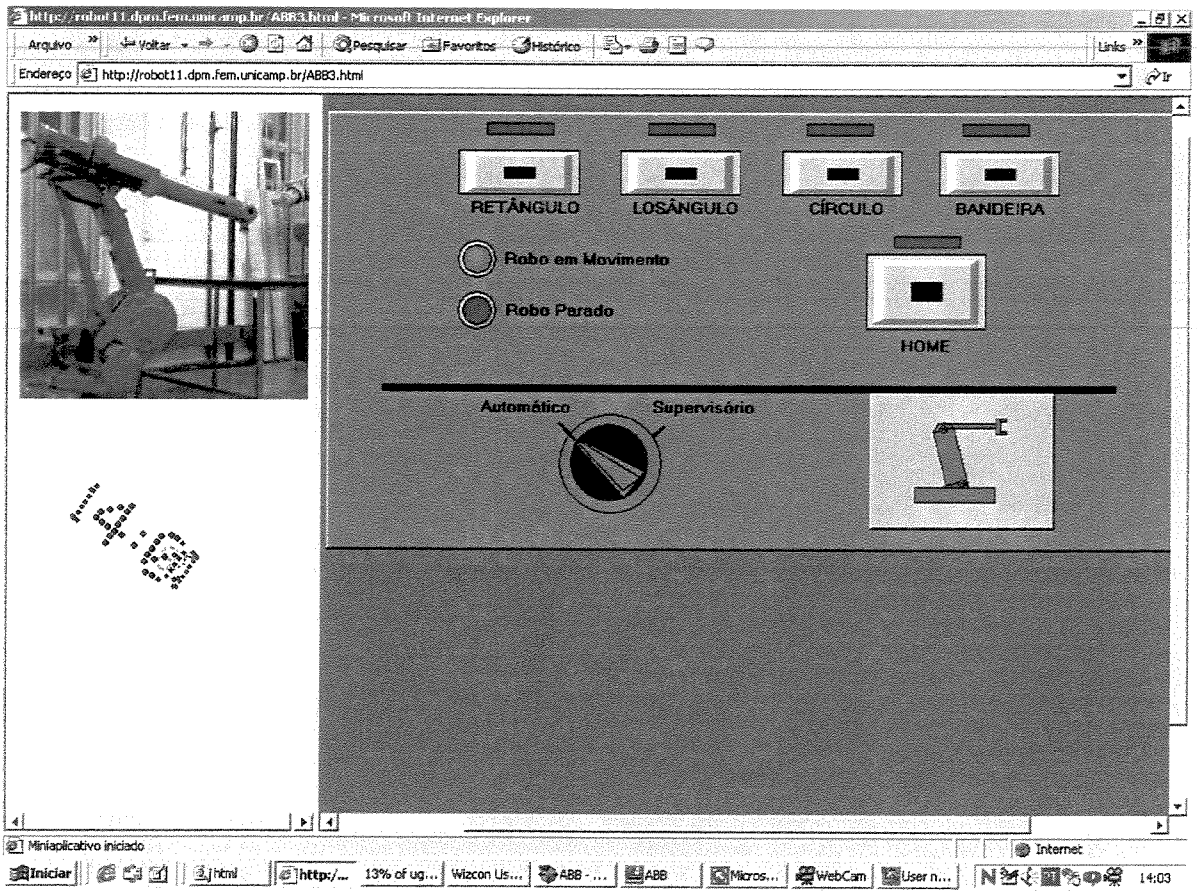


Figura 5.23 – Layout do Sistema de Supervisão na Web Página

Capítulo 6

Conclusões e Perspectivas Futuras

A presente dissertação de mestrado apresentou sistemáticas para implementação de um sistema de supervisão e controle em que usuários de qualquer parte do mundo pudessem se conectar a este sistema e obter informações em tempo real com possibilidade de controle e, teve como objetivo, o estudo e desenvolvimento de ferramentas para possibilitar esta implementação.

A rede Internet apresenta uma largura de banda bastante heterogênea em que as taxas podem variar de 10Kbps (acesso modem) até 10 Mbps em redes locais e tudo está ligado a uma dependência ao tipo de conexão estabelecida na Internet e a um tráfego de dados. Esta limitação de largura de banda pode influenciar em aplicações em tempo real para captura de vídeo, apresentando, no entanto, certa restrição quanto a transmissão das imagens. Para vencer esta restrição é necessário utilizar sistema de compressão de dados e conexão de grande velocidade na Internet.

Outra limitação é o atraso (*delay*) relativo ao protocolo TCP (Otsuka,1996), pois os pacotes enviados não estão necessariamente na mesma ordem dos pacotes recebidos pelos clientes, o que não é apropriado para aplicações em tempo real. Esta restrição pode ser resolvida com a adição de algum grau de autonomia para a aplicação, no caso um Robô, de tal forma a contornar situações de emergência.

No presente trabalho foram estabelecidas duas soluções para contornar estes aspectos: a primeira, através de um controlador lógico, para o caso de uma ocorrência de falha do sistema,

tanto em relação à conexão com o usuário, como em relação à uma falha no próprio sistema operacional do computador (Master) ligado diretamente ao Robô. Neste caso, o controlador lógico executa a tarefa enviada colocando a seguir o Robô em uma situação de proteção. A segunda solução foi, em caso de emergência no sistema, programa-se uma parada deste e o subseqüente o envio de alarmes aos engenheiros de aplicação.

Assim, este estudo serve de suporte na área da Telerobótica, mostrando que o sistema de supervisão pode ser uma grande ferramenta para contornar as instabilidades do sistema controlado remotamente.

Como foi descrito anteriormente, o protocolo TCP não garante uma aplicação em tempo real, pois existe o problema de atraso. Uma alternativa que poderia ser utilizada em trabalhos futuros para melhorar esta comunicação seria a implementação do protocolo RTP (*Real Time Transport Protocol*) para aplicações em tempo real.

Em futuros trabalhos poderiam, ainda, estudar a viabilidade da implementação do sistema robix para alimentação automática dos postos de alimentação da Plataforma PIPEFA e o controle da mesma em uma rede Ethernet, como sugerido no trabalho de tese de Georgini (1999). A arquitetura do controle e supervisão desenvolvida no presente trabalho permite esta possibilidade. No sistema de controle do robô ABB poderiam ser implementados novas interfaces de comunicação (áudio, chat) para aumentar ainda mais os recursos de controle e supervisão do robô, criando conjuntamente um ambiente educacional.

Novos trabalhos podem ser desenvolvidos abrangendo as áreas de telemedicina, telerobótica e domótica, com grandes benefícios para os seres humanos.

Referências Bibliográficas

Aihara, C. K. Cosso, S. G. Saramago, M. A. P. Rosário, J. M. Desenvolvimento de Aplicativos para Monitoramento de Variáveis de Controle de Processos Industriais. In: *Aplicon 2001*, EEUSP São Carlos, Julho 2001.

Álvares, A. J., Paulinyi, L. F. A. Telerobótica: teleoperação do robô ABB IRB 2000 na www. In: Congresso de Engenharia Mecânica Norte-Nordeste-CONEN, 2000, Natal. *Anais...* Natal: Associação Brasileira de Ciências Mecânicas, 2000. p.250-258.(cd-rom).

Blanc, D. *Distribuited Control Software for high Performance Control Loop Algorithm.* In: International Conference Acclerator and Larg Experimental Physics trol Systems, 1999, Trieste, Italy. European Organization for Nuclear Research, pp 48-50.

Brady, Kelvin., Tarn, Tzyh-Jong. Internet-based remote teleoperation In: International Conference on Robotics and Automation, 1998, Leuven. *Proceedings of the...* Piscataway: IEEE, 1998. v.1, p. 65-70.

Dalton, B. Taylor, K. Distributed Robotics over the Internet. In: *IEEE Robotics & Automation Magazine*. June 2000, pp. 22-27.

Dias, Carlos Henrique. *Implementação de um Supervisor de Controle para Robôs Industriais.* Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1993, 149 p. Tese (mestrado).

Fayan, Benedito Luis. Estudo e Especificação de um Supervisor de Controle para um Robô Industrial. Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1992, 132 p. Tese (mestrado).

Ferrel, W. R. Sheridan, T. B. Supervisory Controlof Remote Manipulation. *IEEE Spectrum 4*, 1967, n°10, October: 81-88.

Georgini, João Marcelo. *Elementos para Estruturação e Implementação de Sistemas Automatizados de Produção*. Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1999. 209 p. Tese (Mestrado).

Georgini, João Marcelo. *Automação Aplicada – Descrição e Implementação de Sistemas Seqüenciais com PLCs*. São Paulo: Érica, 2000.

Goldberg, K. Mascha, K. Gter, M. Rothenberg, N. Sutter, C. and Wiegley, J. Desktop teleoperation via the word wide web. In: Proc. IEEE Int. Conf. Robotics and Automation, May 1995, pp. 213-219.

Groover, Mikell P. *Automation, Production Systems, and Computer Integrated Manufacturing* USA: Prentice-Hall International, Inc., 1987, 808 p.

Lemay, L., Candenhead, R. *Sams teach yourself Java 2 in 21 days*. 2.ed. Sams Publishing, 1999. 661p.

Martins, Maurício Pires. *Estruturação da Parte de Comando de um Sistema Automatizado de Produção com Ênfase na Implementação de um Sistema de Supervisão*. Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1998, 158 p. Tese (Doutorado).

Moraes, C. C., Castrucci, P. L.. *Engenharia de Automação Industrial*. Rio de Janeiro: LTC, 2001, 295 p.

Monteiro, F. et al. Teleoperating a mobile robot a solution based on java language. In: International Symposium on Industrial Eletronics , 1997, Guimaraes. *Proceedings of the...* Piscataway: IEEE, 1997. v.1, p. 263-267.

Nogueira, Reinaldo Gonçalves.*Controle de Posição e Orientação de um Manipulador através de um Mouse Espacial.* Campinas: Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas, 1995, 77 p. Tese (Mestrado).

Otsuka, J., *Fatores determinantes na efetividade de ferramentas de comunicação mediada por computador no ensino a distância*, 1996. Disponível pela Internet http://penta.ufrgs.br:/pesquisa/joice/joice_ti.html.

Paulo, E. Canny, J. *A World Wide Web Telerobotic Remote Environment browser.* http://www.cs.berkeley.edu/paulos/papers/www4/. Janeiro 1997. Forth International World Wide Web Conference.

Rosário, J. M. Fauré, J. M. Um exemplo de Coopération Franco-Brésilienne pour la Formation et la Recherche em Automatisation Industrielle. In: Troisème Conferense Internatinale Sur L'Automatisation Industrielle, *Third Internatoinal Conference on Industrial Automation.* Montreal, junho, 1999.

Sheridan, Thomas. B. *Telerobotics, automation, and human supervisory control.* Massachusetts: The Mit Press,1992, 393p.

Sheridan, T. B. Supervisory control of remote manipulators, vehicles and dynamic processes. In: Rouse, W. B., *Advances in Man-Machine Systems Research*, 1984, vol. 1, JAI Press.

Sheridan, Thomas. B. Space teleoperation though time delay review and prognosis. *IEEE Transactions on Robotics and Automation*, 1993. p.592-606.

Suzuki, Tsuyoshi., et al. Multi-robot teleoperation system utilizing the internet. *Advance Robotics*, v1, n.8, p 781-797, set 1999.

Tanenbaum, Andrew. S. *Redes de Computadores*. São Paulo: Campus Ltda, 1997, 952 p.

Taylor,K., Dalton,B. Australlia's Telerobot on the Web. *In: International Symposium on Industrial Robots*, Singapore, oct 1995. Disponível pela internet http://telerobot.mech.uwa.edu.au/robot/singapor.htm.

Taylor, K. , Dalton, B. Issues in internet telerobotics. In: International Conference on Field and Service Robotics, 1997, Canberra. Disponível pela internet http://telerobot.mech.uwa.edu.au/ROBOT/anupaper.htm.

Torres, Gabriel. Redes de Computadores, Curso Completo. Rio de Janeiro: Axcel Books, 2001, 664 p.

Tourino, S. R. G. , Álvares, J. A. Desenvolvimento de um robô móvel autônomo teleoperado via internet In: Congresso de Engenharia Mecânica Norte-Nordeste - CONEN, 2000, Natal. *Anais...* Natal: Associação Brasileira de Ciências Mecânicas, 2000. p267-274.(cd-rom)

Fontes Eletrônicas

[1] Emation - http://www.emation.com/

[2] Intellution - http://www.intellution.com/

[3] Wonderware - http://www.wonderware.com/

[4] Elipse - http://www.elipse.com.br/

[5] GE Fanuc http://www.gefanuc.com/

[6] Apache - http://www.apache.org

[7] HTML – http://www.w3.org

[8] Sun – http://www.java.sun.com

Anexo I

Java e Java Applet

Java é uma linguagem de programação muito conveniente para o desenvolvimento de software que funcione em conjunto com a Internet. Ela também é uma linguagem de programação orientada a objetos que utiliza uma metodologia que está se tornando cada vez mais útil no mundo do design de software. Além disso, ela é uma linguagem multiplataforma, o que significa que seus programas podem ser criados para executar do mesmo modo no Microsoft Windows, Apple Macintosh e na maioria das versões de UNIX, incluindo a Solaris. A figura a seguir mostra o resultado de um sistema dependente de plataforma: vários programas executáveis devem ser produzidos para os vários sistemas.

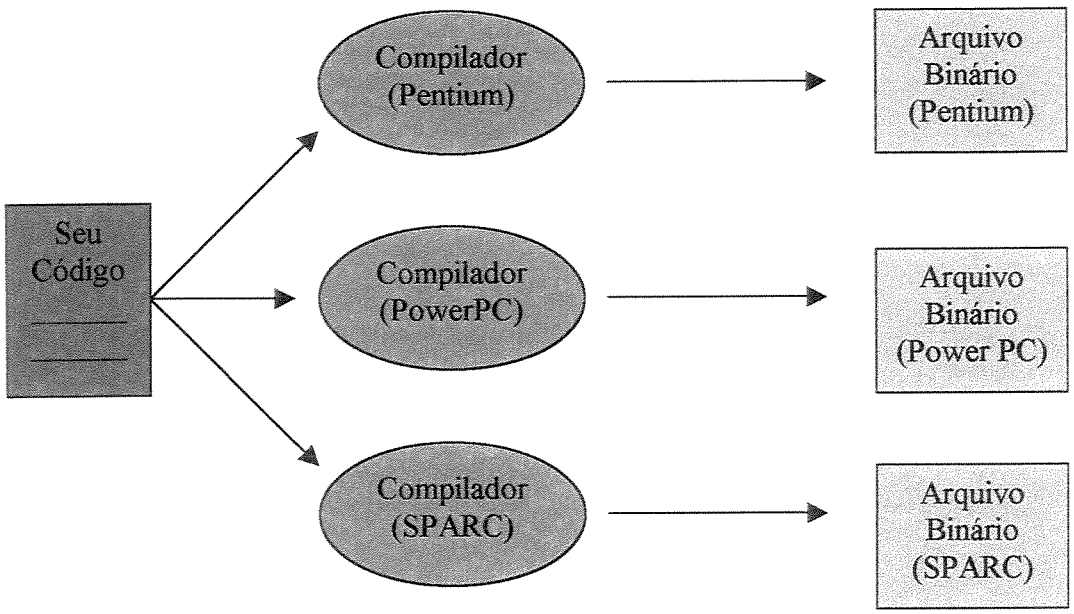


Figura A 1.1 - Programas tradicionais compilados (Java)

Os programas Java atingem essa independência através da utilização de uma máquina virtual – uma espécie de computador dentro de outro. A máquina virtual pega os programas Java compilados e converte suas instruções em comandos que um sistema operacional possa manipular. O mesmo programa compilado, que existe em um formato chamado *bytecode* (*bytecode* é a versão da máquina virtual Java do código de máquina, as instruções que ela entende diretamente), pode ser executado em qualquer plataforma e sistema operacional que possua uma máquina virtual Java.

A máquina virtual também é conhecida como *interpretador Java* ou *runtime Java.*

A linguagem Java vai além da área de trabalho, sendo executada em dispositivos como televisões, relógios de pulso e telefones celulares. A linguagem Java está mais próxima das linguagens de programação populares, como C, C++, Visual Basic e Delphi, do que de uma linguagem de descrição de páginas, como HTML, ou de uma simples linguagem de scripts, como a JavaScript.

A linguagem Java é mais conhecida por sua capacidade de executar em páginas da World Wide Web. Os navegadores Netscape Navigator e Microsoft Internet Explorer podem carregar por download um programa Java a partir de uma página da Web e executá-la localmente no sistema de um usuário

Esses programas, que são chamados de *applets*, aparecem em uma página de Web de maneira semelhante às imagens. Ao contrário das imagens, os applets podem ser interativos – pegando a entrada do usuário, respondendo a ela e apresentando conteúdo mutante (fonte Lemay,1999 e [8]).

Anexo II

Fieldbus

Para uma rede aplicada à interligação de elementos ao nível de chão-de-fábrica (CLPs, válvulas, indicadores dedicados, sensores, transdutores, atuadores, etc) é utilizada a denominação genérica de "barramento de campo", ou Fieldbus.

O termo fieldbus descreve uma rede de comunicação digital que veio substituir o sistema de sinal analógico 4 - 20mA existente ainda hoje nas indústrias (e muito difundido devido a sua imunidade a interferências eletromagnéticas, apesar de sua tecnologia ultrapassada desenvolvida na década de 60).

O fieldbus pode ser definido como uma rede digital, bidirecional (de acesso compartilhado), multiponto e serial, utilizado para interligar os dispositivos primários de automação (dispositivos de campo) a um sistema integrado de automação e controle de processos. Cada dispositivo de campo pode possuir uma "inteligência" (microprocessado), o que o torna capaz de executar funções simples em si mesmo, tais como diagnóstico, controle e funções de manutenção, além de possibilitar a comunicação entre dispositivos de campo (não apenas entre o engenheiro e o dispositivo de campo). Em outras palavras, o fieldbus veio para substituir o controle centralizado pelo distribuído.

Portanto o fieldbus é muito mais que um mero substituto do padrão analógico 4 - 20mA, pois promove a melhora de qualidade, a redução de custos e o aumento de eficiência.

Vale ressaltar que o fieldbus baseia-se no modelo ISO/OSI e que os níveis implementados são o 1,2 e 7. O modelo é reduzido para atender aos requisitos de tempo de resposta.

Anexo III

VPIS - WIZCON

Manufacturer, Country:

Texas Instruments , USA

PLC / Protocol:

TI-435, Koyo-205 Direct Net

Version:

File Name, Size, Date, Latest Version, Description:

VPIT4 DOC 15-07-97

VPIT4 COM 19,625 21-08-95 4:20a; driver for DOS

(not all of the below described functions are available for the DOS driver)

VPIT4 DLL 64,084 23-03-97 5:30a; driver for OS/2

AOS_2T4 EXE 3,789 25-08-95 4:20a; mini-application for OS/2

VPIWNT4 DLL 52,736 13-07-97 7:00a; dirver for Windows 95 and NT

AWN_5T4 EXE 4,897 11-07-97 5:30a; Mini application for Windows

Formato de Endereço (*Address Format*)**:**

Analog Gates : NNVVVVV

NN - Station address (hex, valid 0..FF)

VVVVV - V-memory address (octal, valid 0..77777)

Digital Gates : NNVVVVVB

NN - Station address (hex, valid 0..FF)

VVVVV - V-memory address (octal, valid 0..77777)

B - Bit number (hex, valid 0..F)

Informações dos Blocos

Blocks:

Blocks Read and Write:

An analog (without the B field) address is used for the first address in the block.

The maximum length of the block is 128 words.

Fonte: (Documentação do Software Wizcon v 7.61)

Anexo IV

Programa em C (Telecirurgia)

```
/* file cirurgia.c, build with makecdem.bat, uses cirurgia.h,
cirurgia.inc */
/* Sample command line interpreter for scripting language.
/* Demonstrates use of many functions from rbx?.lib's */

#include "cirurgia.h"
#include "Program.h"

   /* increments for jog commands */
#define LARGE_INC 30
#define SMALL_INC  3

static char acCommandBuf[RBX_COMMAND_LEN_MAX+1];

   /* get the literals tables; this is a convenient C technique
*/
   /*  to  keep  literals  to  in  an  easy-to-edit  table,  while
*/
   /*  using  the  "names"  of  the  literals  in  the  code.
*/
   /* The technique is general and is not unique to this program;
*/
   /*  it  is  useful  during  preparation  of  multi-lingual  apps.
*/

enum
{
   #define MAC(a,b) a,
   #include "cirurgia.inc"
   #undef MAC
};
```

```
char *aszLits[] =
{
   #define MAC(a,b) b,
   #include "cirurgia.inc"
   #undef MAC
};


/* wrapper around rbxSaveCfg call to save configuration file */
static void SaveConfig(char *szFspec)
{
   switch(rbxSaveCfg(szFspec))
   {
      case 0:
  printf("%s: %s\n", aszLits[CfgSavedLit],
     aszLits[DefaultCfgFnameLit]);
  return;

      case 1: goto bad_open;
      case 2: goto bad_close_write;
   }

   bad_close_write:
      unlink(szFspec);

   bad_open:
      printf("%s: %s\n", aszLits[FileWriteErrLit], szFspec);
}

/* noise to make when no action */
void NoActionBeep(void)
{
   sound(100);
   delay(200);
   nosound();
}


void OutOfRangeBeep(void)
{
   sound(2000);
   delay(200);
   nosound();
}


void EnterBeep(void)
```

```
{
   sound(2000);
   delay(200);
   nosound();
}


void ErrorBeep(void)
{
   sound(2000);
   delay(200);
   nosound();
}

void Stat(void)
{
   short sEBase = rbxEnumBaseR();
   short si, sStat;

   static short *psMotorsParam = NULL; /* for clarity; statics
are anyway */

      /*   could   also   just   declare   the   array   with
RBX_MOTOR_PARAM_COUNT_MAX */
   if (!psMotorsParam)
      psMotorsParam = malloc(sizeof(short) * rbxMotorCount);

   printf("%s  ", rbxMotorTtl);

   for (si = sEBase; si < rbxMotorCount + sEBase; si++)
      printf("%*d", RBX_READING_FMT_LEN+1, si);
   printf("\n");

   for (si = sEBase; si < rbxMotorParamCount + sEBase; si++)
   {
      short si1;

      printf("%s  ", rbxMotorParamTtl[si-sEBase /* it's a C-array
*/]);
      rbxMotorsParamR(si, psMotorsParam);
      for  (si1  =  0  /*  not  sEBase,  it's  a  C-ary  */;  si1  <
rbxMotorCount; si1++)
  printf("%*d", RBX_READING_FMT_LEN+1, psMotorsParam[si1]);
      printf("\n");
   }
   printf("\n");
```

```
#if 0
   /* alternate, less "automatic" approach */
   printf("%s\n", aszLits[ShowTitleLit]);

   for (si = sEbase; si < rbxParameterCount + sEBase; si++)


   for (si = sEBase; si < rbxMotorCount + sEBase; si++)
   {
      short sPos, sMn, sMx, sInit, sAcc, sDec, sSpd, sInvrt,
sPwr, sPin;

      sPos  = rbxPosR(si);
      sInit = rbxInitPosR(si);
      sMn   = rbxMinPosR(si);
      sMx   = rbxMaxPosR(si);
      sAcc  = rbxAccelR(si);
      sDec  = rbxDecelR(si);
      sSpd  = rbxMaxSpdR(si);
      sInvrt= rbxInvertR(si);
      sPwr  = rbxPowerR(si);
      sPin  = rbxPinnedR(si);

      printf(" %2d %5d %4d %4d %4d   %d  %5d %5d %5d %5d %5d\n",
  si, sPos, sSpd, sAcc, sDec, sPwr, sMn, sMx, sInit, sInvrt,
sPin);
   }
   printf("\n");
#endif


   sStat = rbxStatusR();

   printf("statuses:\n"
      "EnumBase:  %d     "
      "Compiling: %c     "
      "Running:   %c     "
      "Ready:     %c\n"
      "Continued: %c     "
      "CmdWasTxt: %c     "
      "AuxA:      %c     "
      "AuxB:      %c\n",

      rbxEnumBaseR(),
      (sStat & RBX_COMPILING_STAT) ? 'X' : '-',
      (sStat & RBX_RUNNING_STAT) ? 'X' : '-',
      (sStat & RBX_READY_STAT) ? 'X' : '-',
```

```
      (sStat & RBX_CONTINUED_STAT) ? 'X' : '-',
      (sStat & RBX_CMD_WAS_TEXT_STAT) ? 'X' : '-',
      (sStat & RBX_AUX_A_STAT) ? 'X' : '-',
      (sStat & RBX_AUX_B_STAT) ? 'X' : '-'
   );

   printf("\n");

   /* read switches and adc's */
   {
      short asAdc[8];
      short asSwitch[8];
      short si;

      printf(rbxSensorTtl);
      rbxAdcsR(asAdc);
      for (si = 0; si < rbxAdcCount; si++)
  printf("%6d", asAdc[si]);

      rbxSwitchesR(asSwitch);
      printf("  ");
      for (si = 0; si < rbxSwitchCount; si++)
  printf("%c", asSwitch[si] ? 'X' : '-');
   }

   printf("\n\n");
}


/* Jog a motor.  Returns 0 if character not recognized; 1 if
processed; */
short Jog(int iCh)
{
   //printf("jogging");          //LeoSoft
   short siM, sInc;
   char *pc;

   if (!iCh)
   {
      iCh = getch();
      goto extended;
   }

   iCh = (char)toupper(iCh);

   if ((pc = strchr(aszLits[JogLargePlusLit], iCh)) != NULL)
   {
```

```
    siM = (short)(pc - aszLits[JogLargePlusLit]);
    sInc = LARGE_INC;
    goto movit;
}

if ((pc = strchr(aszLits[JogLargeMinusLit], iCh)) != NULL)
{
    siM = (short)(pc - aszLits[JogLargeMinusLit]);
    sInc = -LARGE_INC;
    goto movit;
}

if ((pc = strchr(aszLits[JogSmallPlusLit], iCh)) != NULL)
{
    siM = (short)(pc - aszLits[JogSmallPlusLit]);
    sInc = SMALL_INC;
    goto movit;
}

if ((pc = strchr(aszLits[JogSmallMinusLit], iCh)) != NULL)
{
    siM = (short)(pc - aszLits[JogSmallMinusLit]);
    sInc = -SMALL_INC;
    goto movit;
}

return 0;

movit:
if (siM > rbxMotorCount - 1)
    return 0;

rbxJumpRel(siM + rbxEnumBaseR(), sInc);
printf("jogging");              //LeoSoft

if (rbxPinnedR(siM))
    OutOfRangeBeep();

return 1;

extended:

if (iCh >= 16 && iCh <= 24)
{
    short sPwr;

    siM = iCh - 16;
```

```
      if (siM > rbxMotorCount - 1)
  return 0;

  /* toggle power */
      rbxPowerW(siM, !rbxPowerR(siM));

      printf("%s %d: %s %d\n",
  rbxMotorTtl, siM+1,
  rbxMotorParamTtl[rbxPowerMotorParamNdx], sPwr);
      return 1;
   }

   return 0;
}


void ReportError(void)
{
   char *tkn;

   if (rbxCmdWasTextR())
   {
      printf("%s\n%*c\n", rbxLastCmdR(), rbxErrColR(), '^');
   }

   printf("%s", rbxErrMsgR());
   tkn = rbxErrTokenR();
   if (*tkn)
      printf(" --  \"%s\"", tkn);
   printf("\n");
}


static void DirectControl(void)
{
   printf("%s\n\n", aszLits[DirectPromptLit]);

   for (;;)
   {
      int ch;
      switch(ch = getch())
      {
   case '^':  Stat(); break;
   case '\x1b':   goto done; /* escape */
   default:
      if (!Jog(ch))
      {
```

```
            NoActionBeep();
            printf("%s\n", aszLits[DirectPromptLit]);
         }
         break;
          }
      }
      done:
          ;
}


static void Help(void)
{
   short sEBase = rbxEnumBaseR();
   short si, scItems;

      /* macros */
   scItems = rbxMacroCountR();

   printf("\n\n%s:\n", aszLits[DrvMacrosTitleLit]);
   for (si = sEBase; si < scItems + sEBase; si++)
      printf("%-20s", rbxMacroNameR(si));

   if (!scItems)
      printf(aszLits[NoMacrosLit]);

      /* resident primitive commands */
   scItems = rbxPrimitiveCount;

   printf("\n\n%s:\n", aszLits[DrvPrimitivesTitleLit]);

   for (si = sEBase; si < scItems + sEBase; si++)
      printf("%-20s", rbxPrimitiveNameR(si));


      /* resident keywords */
   scItems = rbxKeywordCount;
   printf("\n\n%s:\n", aszLits[DrvKeywordsTitleLit]);

   for (si = sEBase; si < scItems + sEBase; si++)
      printf("%-20s", rbxKeywordR(si));

      /* console commands */
   printf("\n\n%s:\n", aszLits[ConCommandsTitleLit]);
   for (si = CommandBeginsLit + 1; si < CommandEndsLit; si++)
      printf("%-20s", aszLits[si]);
```

```
   done:
   printf("\n\n");
}


static void GetCommands(void)
{
   char  acXBuf[RBX_COMMAND_LEN_MAX+1];      /* extra buffer for
strtok'ing */
   char *npos;

   printf("\n%s\n\n", aszLits[TypeHelpForHelpLit]);

   for (;;)
   {
      char *pcPrompt;
      ushort usStat;
      char *pcTkn;
      short scTkns;

      usStat = rbxStatusR();

      if (usStat & RBX_COMPILING_STAT)
      {
  if (usStat & RBX_CONTINUED_STAT)
     pcPrompt = aszLits[CompPlusPromptLit];
  else
     pcPrompt = aszLits[CompPromptLit];
  printf("%3d %s ", rbxPartialMacroSizeR(), pcPrompt);
      }
      else
      {
  if (usStat & RBX_CONTINUED_STAT)
     pcPrompt = aszLits[ExecPlusPromptLit];
  else
     pcPrompt = aszLits[ExecPromptLit];
  printf("%s ", pcPrompt);
      }

      fgets(acCommandBuf, sizeof(acCommandBuf), stdin);
      if ((npos = strchr(acCommandBuf, '\n')) != NULL)
  *npos = '\0';

      strcpy(acXBuf, acCommandBuf);

      scTkns = 0;
      if ((pcTkn = strtok(acXBuf, " ")) != NULL)
```

```
        {
   ++scTkns;
   while (strtok(NULL, " "))
      ++scTkns;
        }

        /* note that there is no explicit check for a blank line:
*/
        /* blank lines fall through to the rbx call. */

        if (scTkns == 1)
        {
   if (!stricmp(pcTkn, aszLits[ExitLit]))
      return;

   if (!stricmp(pcTkn, aszLits[SaveConfigLit]))
   {
      SaveConfig(aszLits[DefaultCfgFnameLit]);
      continue;
   }

   if (!stricmp(pcTkn, aszLits[StatusLit]))
   {
      Stat();
      continue;
   }

   if (!stricmp(pcTkn, aszLits[MemleftLit]))
   {
      printf("%d\n", rbxMemleftR());
      continue;
   }

   if (!stricmp(pcTkn, aszLits[HelpLit]))
   {
      Help();
      continue;
   }

   if (!stricmp(pcTkn, aszLits[DirectLit]))
   {
      DirectControl();
      continue;
   }
        }

        rbxCmd(acCommandBuf);
```

```
      if (rbxErrNo)
      {
  ReportError();
  ErrorBeep();
      }
      continue;
   }
}


void main(void)
{
   short lsensor1;                          //Program Estado do
Sensor 1
   short lsensor2;                         // Program Estado do
Sensor 2
   short lsensor3;                         // Program Estado do
Sensor 3
   short lsensor4;                         // Program Estado do
Sensor 4
   short lsensor5;                         // Program Estado do
Sensor 5
   short lsensor6;                         // Program Estado do
Sensor 6
   short lsensor7;                         // Program Estado do
Sensor 7

   short sInitErr;
   switch (sInitErr = rbxAttach())
   {
      case 0:
  break;

      case 1:
  printf("%s\n", aszLits[DrvNotInstalledLit]);
  exit(1);

      default:
  printf("rbx_Attach error [%d], not recognized", sInitErr);
  exit(1);
   }


   printf("%s:          %s...",          aszLits[CfgLoadingLit],
aszLits[DefaultCfgFnameLit]);
   switch (rbxLoadCfg(aszLits[DefaultCfgFnameLit]))
```

```
    {
       case 1:
    printf("%s: %s\n", aszLits[FileLoadErrLit],
         aszLits[DefaultCfgFnameLit]);
    printf("%s: %d\n", aszLits[LineLit], rbxErrLine);
    ReportError();
    break;

       case 2:
    printf("%s: %s\n", aszLits[CfgNotFoundLit],
         aszLits[DefaultCfgFnameLit]);
    break;

       default: /* went ok */
    printf("%s\n", aszLits[DoneLit]);
    break;
     }

    rbxRestart(); /* reprogam Adapter in case power has been
interrupted */

    printf("%s: %s...",
       aszLits[StartupLoadingLit],
aszLits[DefaultStartupFnameLit]);
    switch (rbxLoad(aszLits[DefaultStartupFnameLit]))
    {
       case 1:
   printf("%s: %s\n", aszLits[FileLoadErrLit],
        aszLits[DefaultStartupFnameLit]);
   printf("%s: %d\n", aszLits[LineLit], rbxErrLine);
   ReportError();
   break;

       case 2:
   printf("%s: %s\n", aszLits[StartupNotFoundLit],
        aszLits[DefaultStartupFnameLit]);
   break;

       default: /* went ok */
   printf("%s\n", aszLits[DoneLit]);
   break;
    }

    while(!kbhit())                  // Program leitura dos Sensores e
comparacao
    {
```

```
   lsensor1=rbxSwitchR(1);          //Program Leitura do sensor 1
(trajetoria1)
   lsensor2=rbxSwitchR(2);          //Program Leitura do sensor 2
(trajetoria2)
   lsensor3=rbxSwitchR(3);          //Program Leitura do sensor 3
(trajetoria3)
   lsensor4=rbxSwitchR(4);       //Program Leitura do sensor 4
   lsensor5=rbxSwitchR(5);       //Program Leitura do sensor 5
   lsensor6=rbxSwitchR(6);       //Program Leitura do sensor 6
   lsensor7=rbxSwitchR(7);          //Program Leitura do sensor 7
(inicio)

   if (lsensor1==1 && lsensor3==0 && lsensor5==0 && lsensor7==0)
   {
    printf ("trajetoria 1");
    rbxLoad("traj1.RBX");     //trajetoria 1
    }

   if (lsensor1==0 && lsensor3==1 && lsensor5==0 && lsensor7==0)
   {
    printf ("trajetoria 2");
    rbxLoad("traj2.RBX");     //trajetoria 2
    }

   if (lsensor1==0 && lsensor3==0 && lsensor5==1 && lsensor7==0)
   {
    printf ("trajetoria 3");
    rbxLoad("traj3.RBX");     //trajetoria 3
    }

   if (lsensor1==1 && lsensor3==0 && lsensor5==0 && lsensor7==0)
   {
    printf ("inicio");
    rbxLoad("traj0.RBX");     //inicio
    }
   }
  }
 }
```

Apêndice A

Programação WEBCAM - Video

```
<html>

   <applet code="javacamera.class"
codebase="http://143.106.9.80:2047/" align="MIDDLE"
    width="250" height="250">
      <param name="FILE" value="http://143.106.9.80:2047/cgi-
bin/video.vod">
    </applet>
<head>
   <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
   <meta name="GENERATOR" content="Microsoft FrontPage 4.0">
   <title>Supervisão - Java Script - Hora e Data</title>
   <base target="_self">
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<title>Clock3D.class (2)</title>
</head>

<body>
<p align="center"><applet code="Clock3D.class" align="left"
        width="200" height="200">
        <param name="12hour" value="0">
        <param name="a1" value="12500">
    <param name="bgcolor" value="#FFFFFF">
        <param name="color" value="#000000">
        <param name="fps" value="20">
        <param name="irotx" value="0">
        <param name="iroty" value="0">
        <param name="irotz" value="00">
        <param name="pixangle" value="5">
        <param name="pixd" value="25">
```

```
        <param name="radius" value="20">
        <param name="rotx" value="0">
        <param name="roty" value="-4">
        <param name="rotz" value="0.401">
        <param name="style" value="1">
    Your browser is without java support</applet>
</p>
</body>
</html>
```

Programação Homepage - Controle

```
<HTML>
<HEAD>
   <TITLE>Wizcon Application</TITLE>
</HEAD>
<BODY>

<!--==================
= Visualizer applet =
====================-->
<CENTER>
        <APPLET  ARCHIVE="w4ivsl200.jar,w4ireq200.jar"
         CODE="wizcon/visualizer/Visualizer.class"
         WIDTH=1024 HEIGHT=688>
        <PARAM NAME=filebase VALUE="Pictures">
        <PARAM NAME=file     VALUE="ABB.wnp">
        <PARAM NAME=InetStudioBase   VALUE="InetStudio">
        <PARAM NAME=ALPopupBase   VALUE="AnnPrf">
        <PARAM NAME=cabbase  VALUE="wizcon.cab">
        <strong>You need a Java-enabled browser to view this
applet.</strong><BR>
        </APPLET>
</CENTER>


</BODY>
</HTML>
```

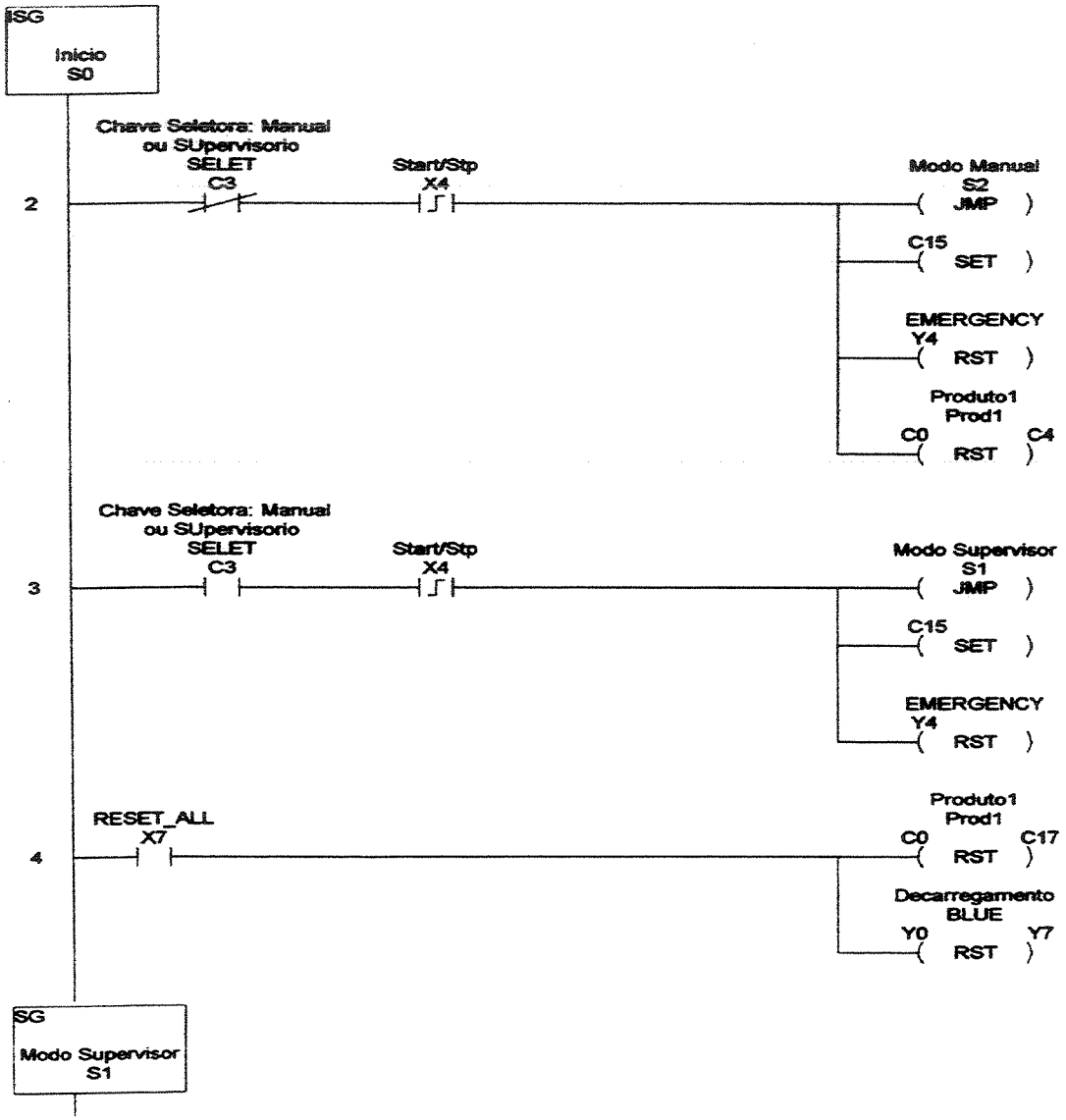
Distribuição da Homepage

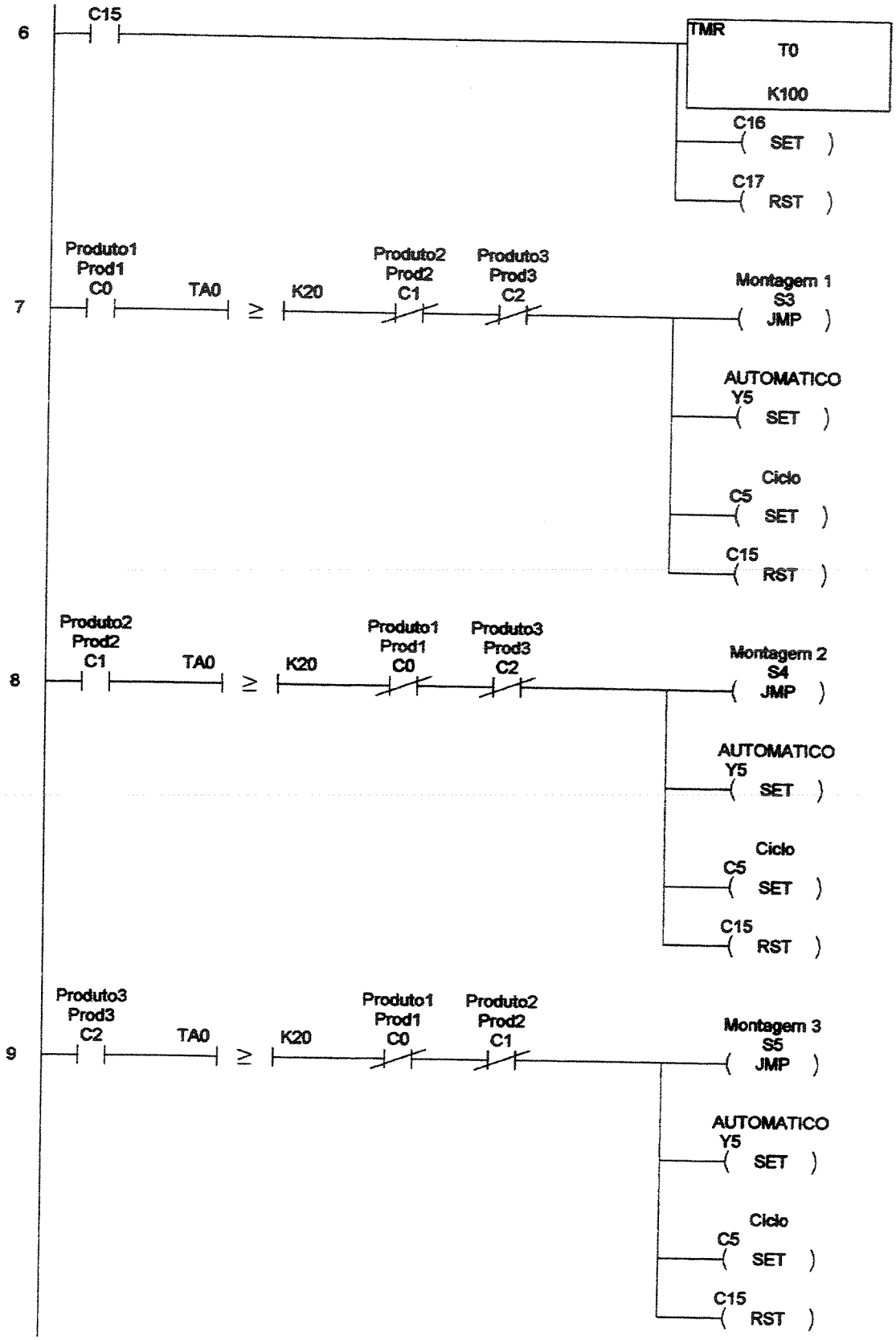
```
<frameset cols="30%,70%">
<frame src="pagcam6">
<frame src="ABB2.html">
<noframes>
<body bgcolor="#ffffcc">
```

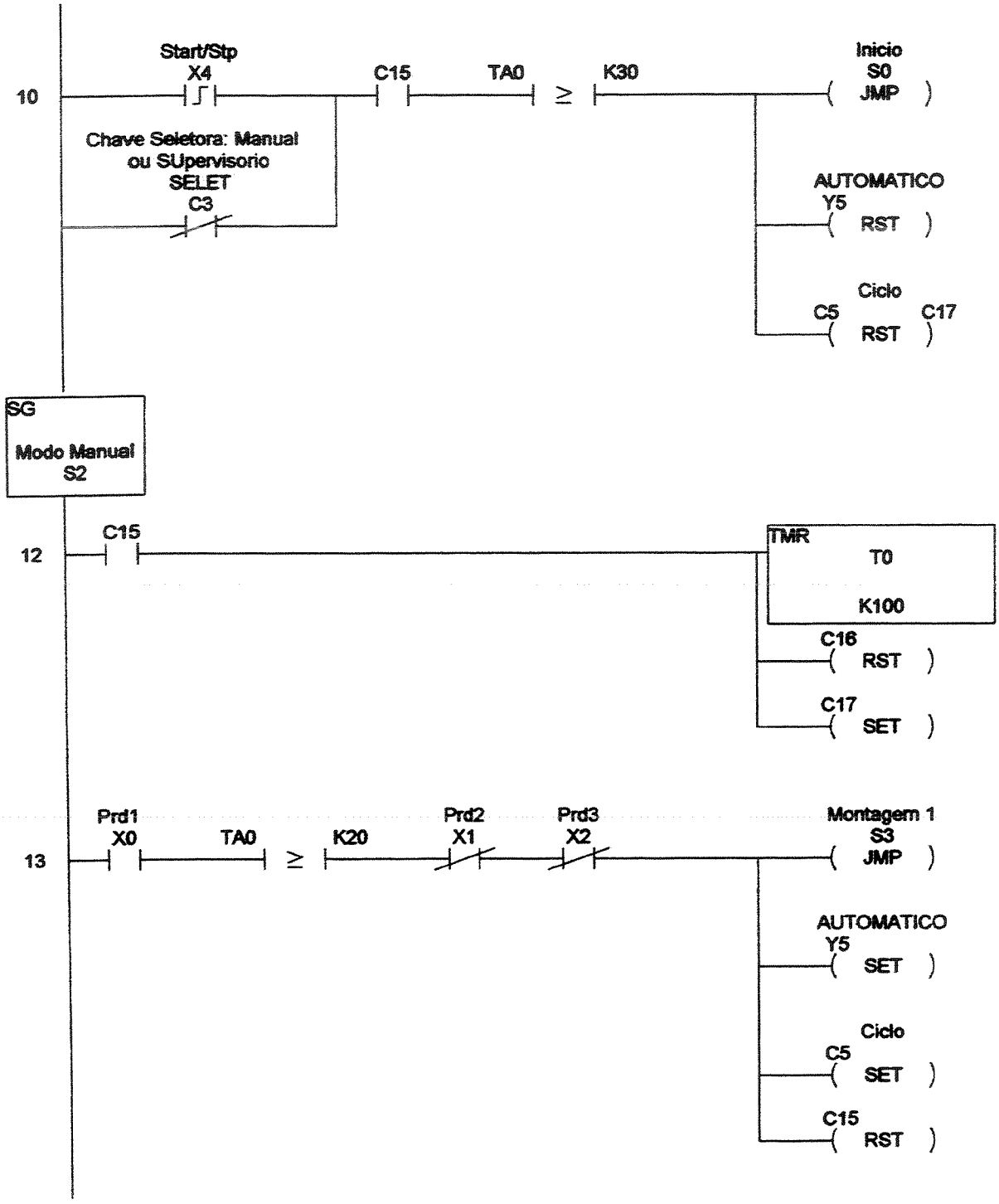
```
</body>
</noframes>
</frameset>

<frameset>

</frameset>
```

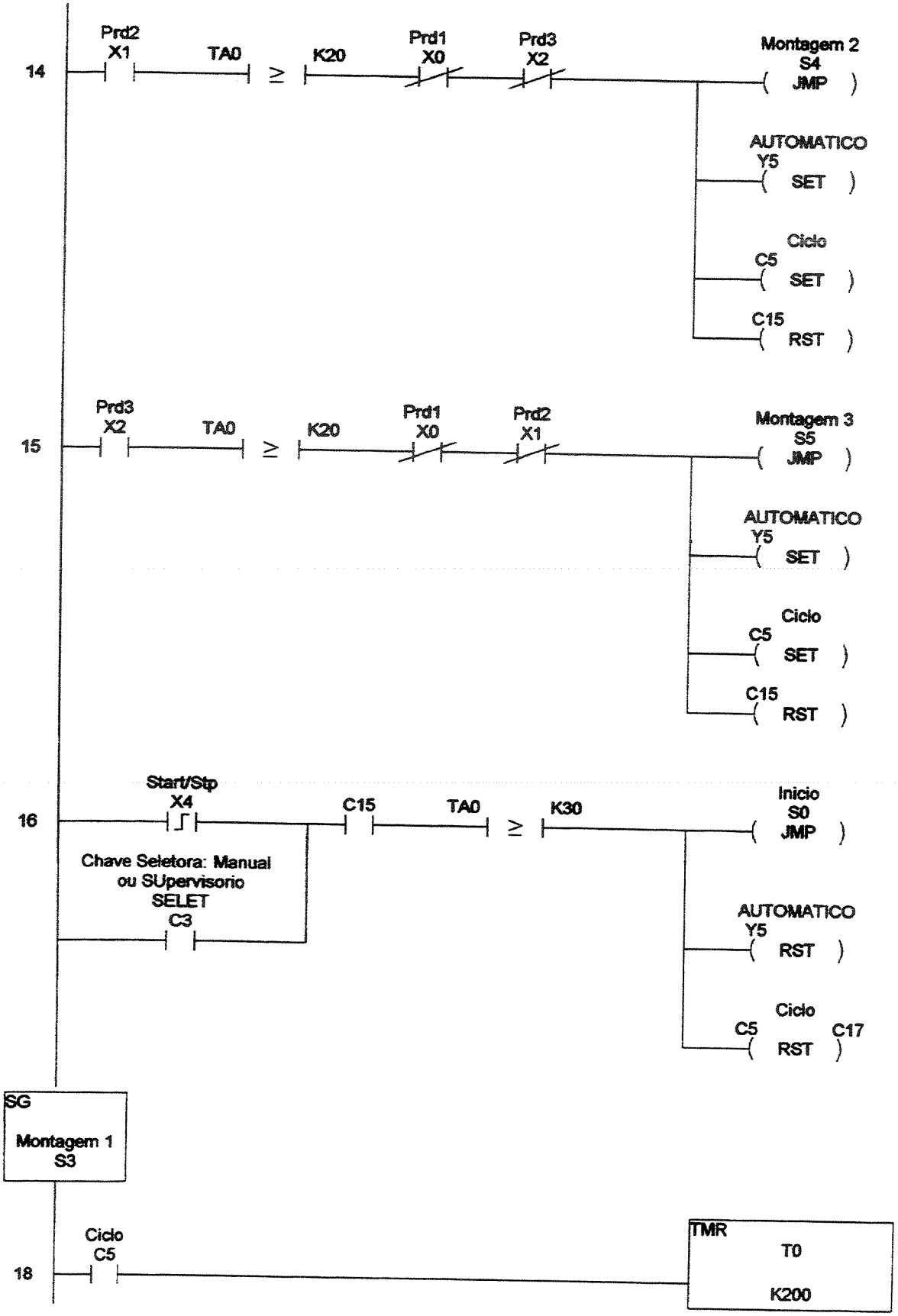
Apêndice B

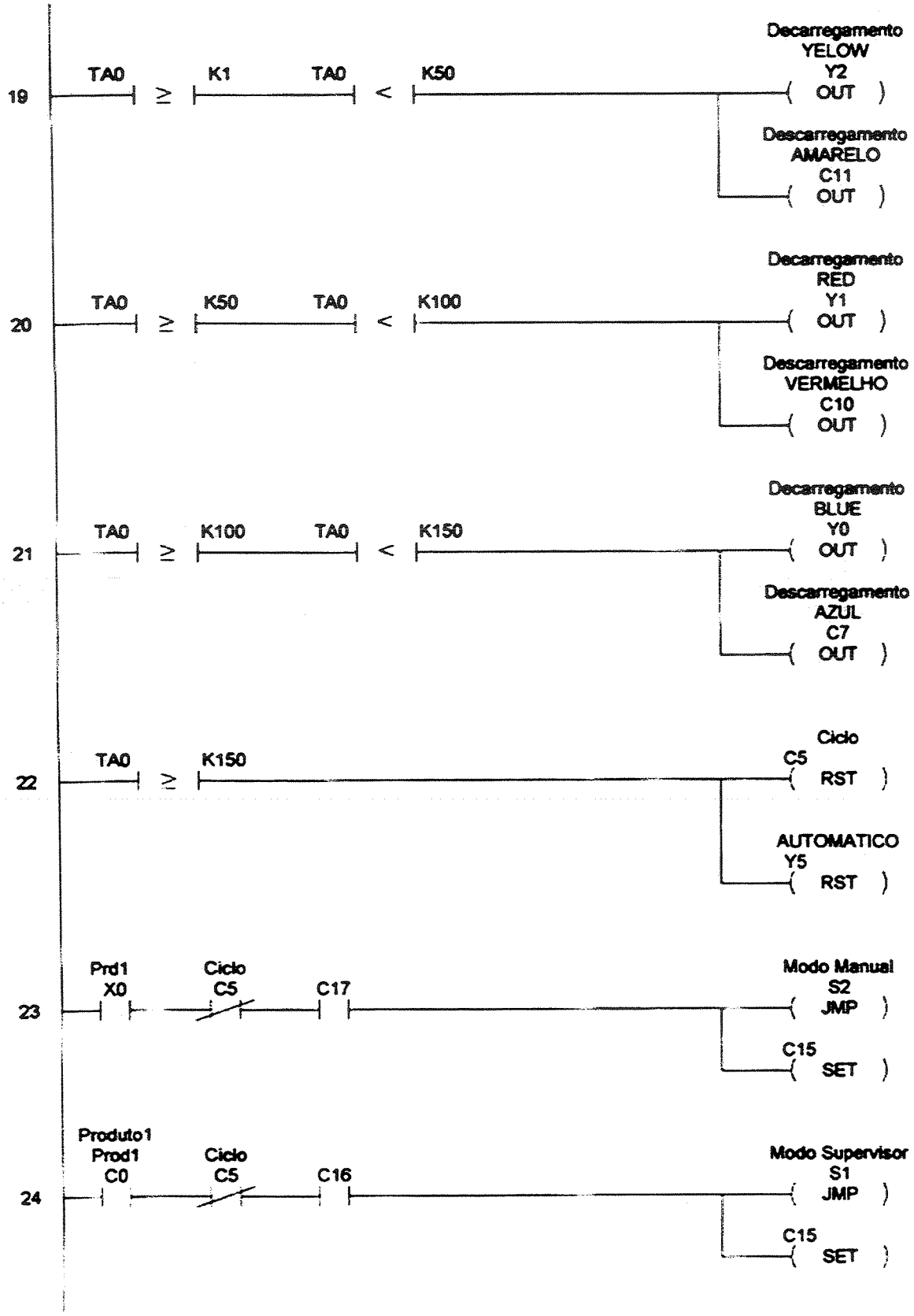
Programação Ladder (Célula de Montagem)

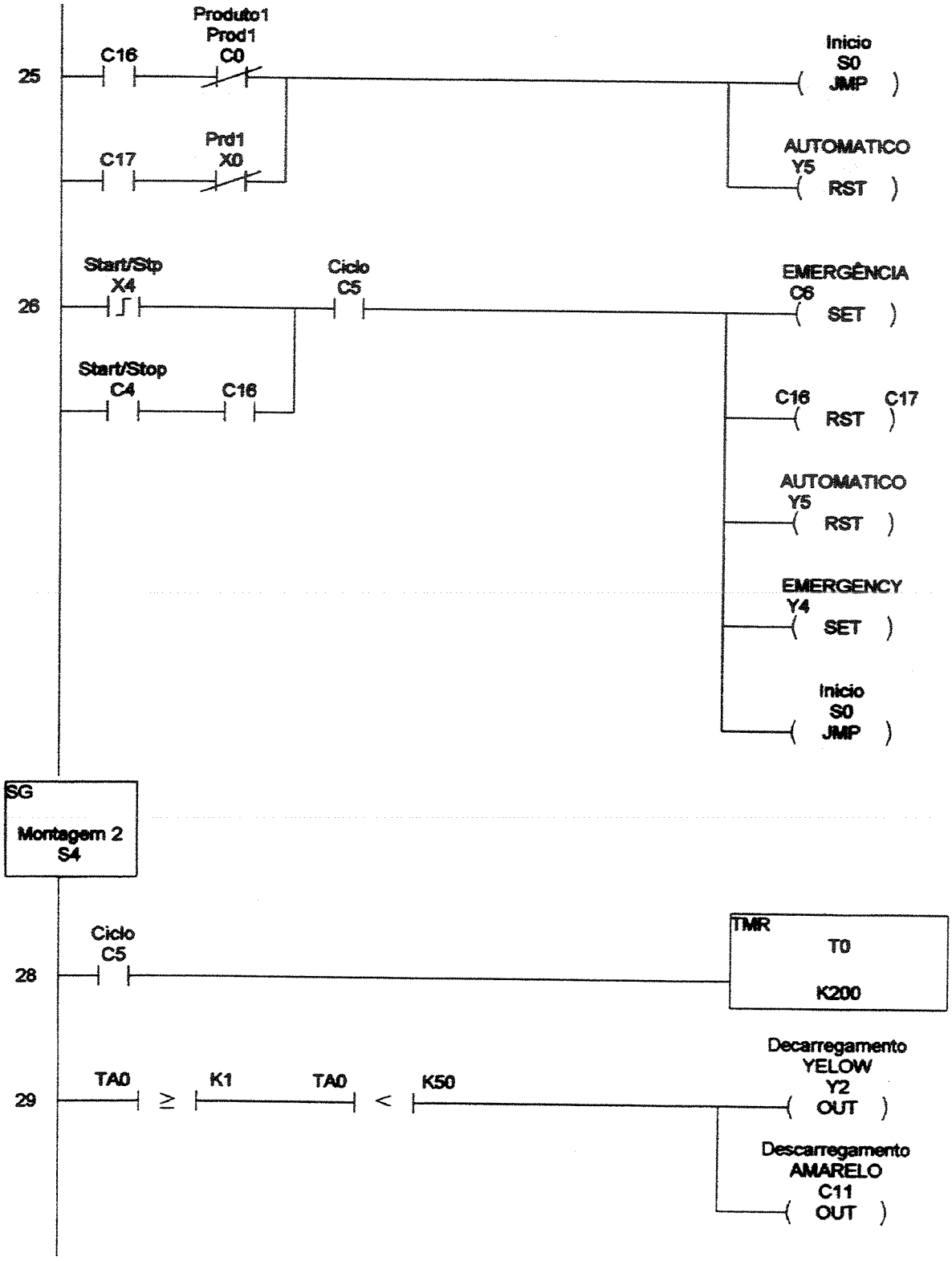


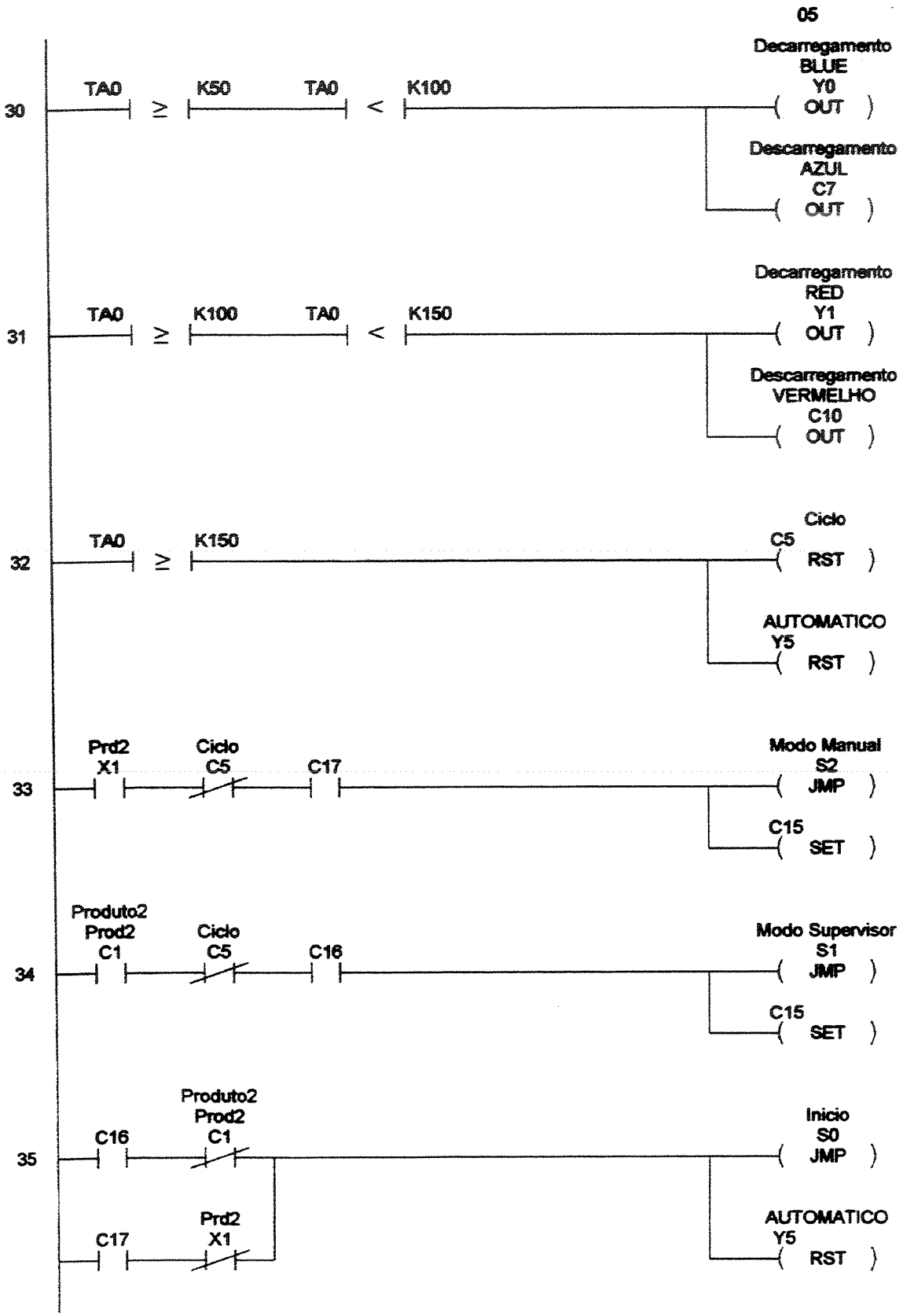

6
C15
TMR
T0
K100
C16
SET
C17
RST
7
Produto1
Prod1
C0
TA0
≥
K20
Produto2
Prod2
C1
Produto3
Prod3
C2
Montagem 1
S3
JMP
AUTOMATICO
Y5
SET
Ciclo
C5
SET
C15
RST
8
Produto2
Prod2
C1
TA0
≥
K20
Produto1
Prod1
C0
Produto3
Prod3
C2
Montagem 2
S4
JMP
AUTOMATICO
Y5
SET
Ciclo
C5
SET
C15
RST
9
Produto3
Prod3
C2
TA0
≥
K20
Produto1
Prod1
C0
Produto2
Prod2
C1
Montagem 3
S5
JMP
AUTOMATICO
Y5
SET
Ciclo
C5
SET
C15
RST

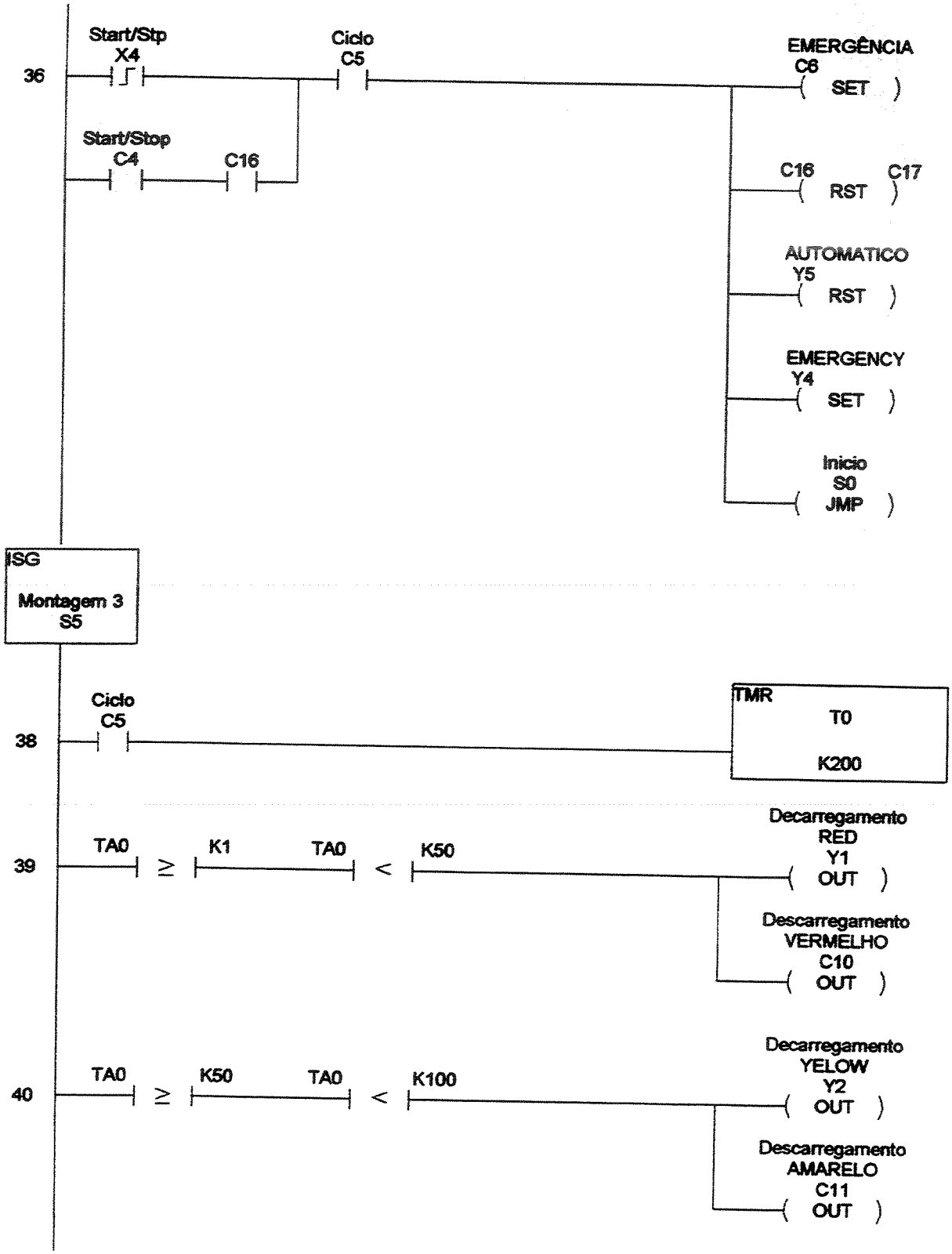

10
Start/Stp
X4
C15
TA0
≥
K30
Inicio
S0
JMP
Chave Seletora: Manual
ou SUpervisorio
SELET
C3
AUTOMATICO
Y5
RST
Ciclo
C5
C17
RST
SG
Modo Manual
S2
12
C15
TMR
T0
K100
C16
RST
C17
SET
13
Prd1
X0
TA0
≥
K20
Prd2
X1
Prd3
X2
Montagem 1
S3
JMP
AUTOMATICO
Y5
SET
Ciclo
C5
SET
C15
RST

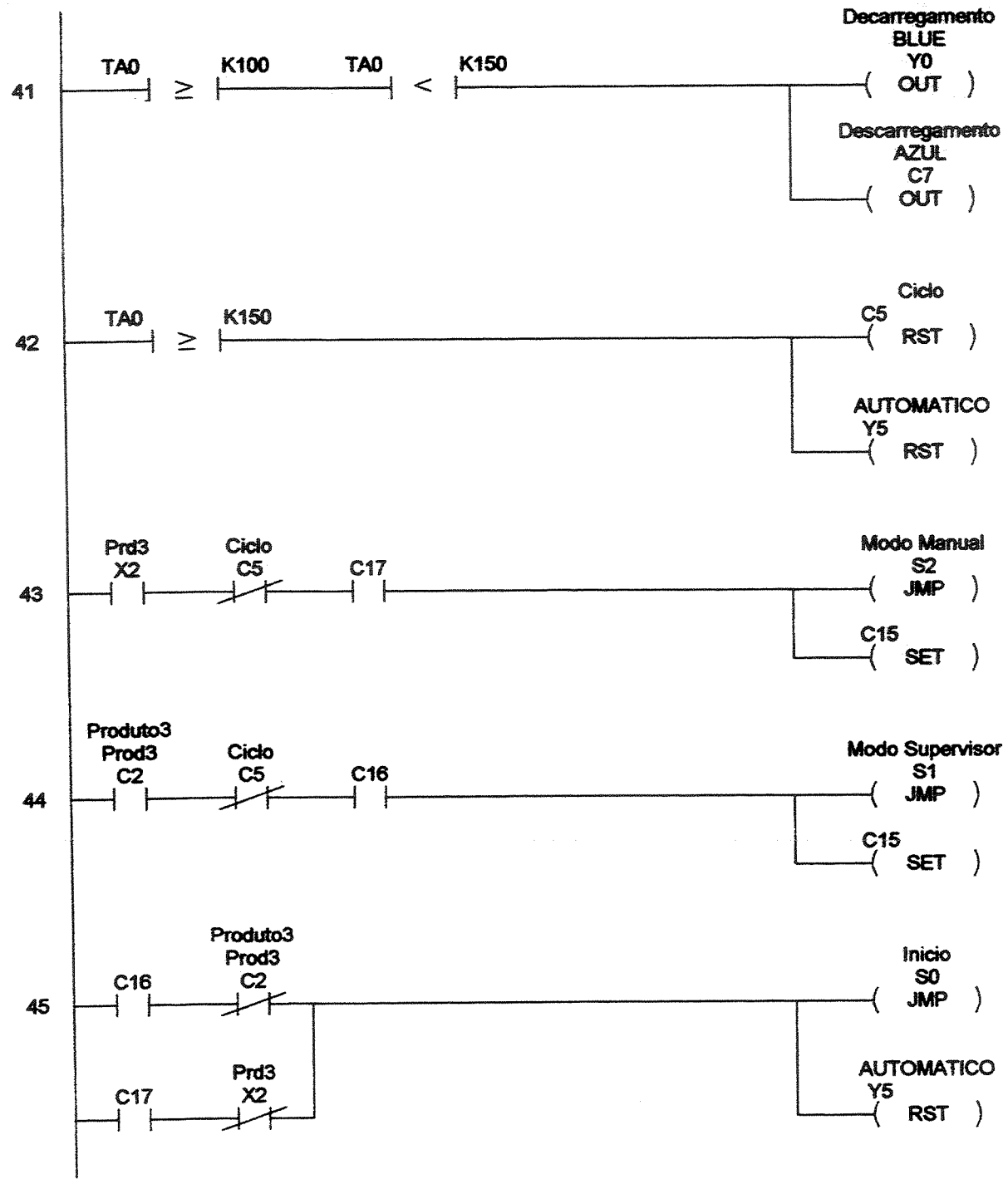

14
Prd2
X1
TA0
≥
K20
Prd1
X0
Prd3
X2
Montagem 2
S4
JMP
AUTOMATICO
Y5
SET
Ciclo
C5
SET
C15
RST
15
Prd3
X2
TA0
≥
K20
Prd1
X0
Prd2
X1
Montagem 3
S5
JMP
AUTOMATICO
Y5
SET
Ciclo
C5
SET
C15
RST
16
Start/Stp
X4
C15
TA0
≥
K30
Inicio
S0
JMP
Chave Seletora: Manual
ou SUpervisorio
SELET
C3
AUTOMATICO
Y5
RST
Ciclo
C5
RST
C17
SG
Montagem 1
S3
18
Ciclo
C5
TMR
T0
K200

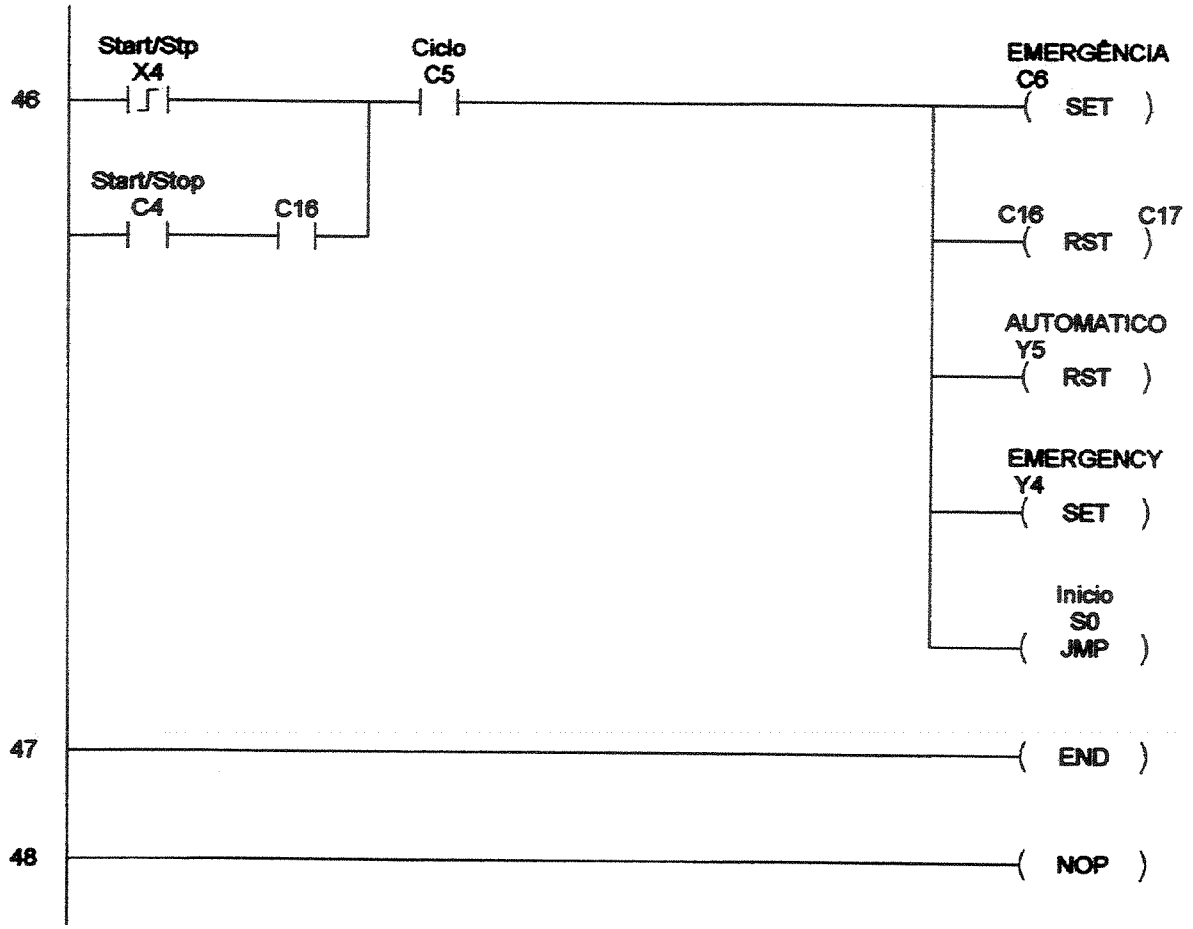

19
TA0 ≥ K1 TA0 < K50
Decarregamento YELOW Y2 OUT
Descarregamento AMARELO C11 OUT
20
TA0 ≥ K50 TA0 < K100
Decarregamento RED Y1 OUT
Descarregamento VERMELHO C10 OUT
21
TA0 ≥ K100 TA0 < K150
Decarregamento BLUE Y0 OUT
Descarregamento AZUL C7 OUT
22
TA0 ≥ K150
Ciclo C5 RST
AUTOMATICO Y5 RST
23
Prd1 X0 Ciclo C5 C17
Modo Manual S2 JMP
C15 SET
24
Produto1 Prod1 C0 Ciclo C5 C16
Modo Supervisor S1 JMP
C15 SET


Produto1
Prod1
C0
C16
25
Inicio
S0
JMP
C17
Prd1
X0
AUTOMATICO
Y5
RST
Start/Stp
X4
Ciclo
C5
EMERGÊNCIA
C6
SET
Start/Stop
C4
C16
C16
RST
C17
AUTOMATICO
Y5
RST
EMERGENCY
Y4
SET
Inicio
S0
JMP
SG
Montagem 2
S4
Ciclo
C5
28
TMR
T0
K200
TA0
≥
K1
TA0
<
K50
29
Decarregamento
YELOW
Y2
OUT
Descarregamento
AMARELO
C11
OUT


05
Decarregamento
BLUE
Y0
OUT
Descarregamento
AZUL
C7
OUT
Decarregamento
RED
Y1
OUT
Descarregamento
VERMELHO
C10
OUT
Ciclo
C5
RST
AUTOMATICO
Y5
RST
Modo Manual
S2
JMP
C15
SET
Modo Supervisor
S1
JMP
C15
SET
Inicio
S0
JMP
AUTOMATICO
Y5
RST
30
TA0
≥
K50
TA0
<
K100
31
TA0
≥
K100
TA0
<
K150
32
TA0
≥
K150
33
Prd2
X1
Ciclo
C5
C17
34
Produto2
Prod2
C1
Ciclo
C5
C16
35
C16
Produto2
Prod2
C1
C17
Prd2
X1

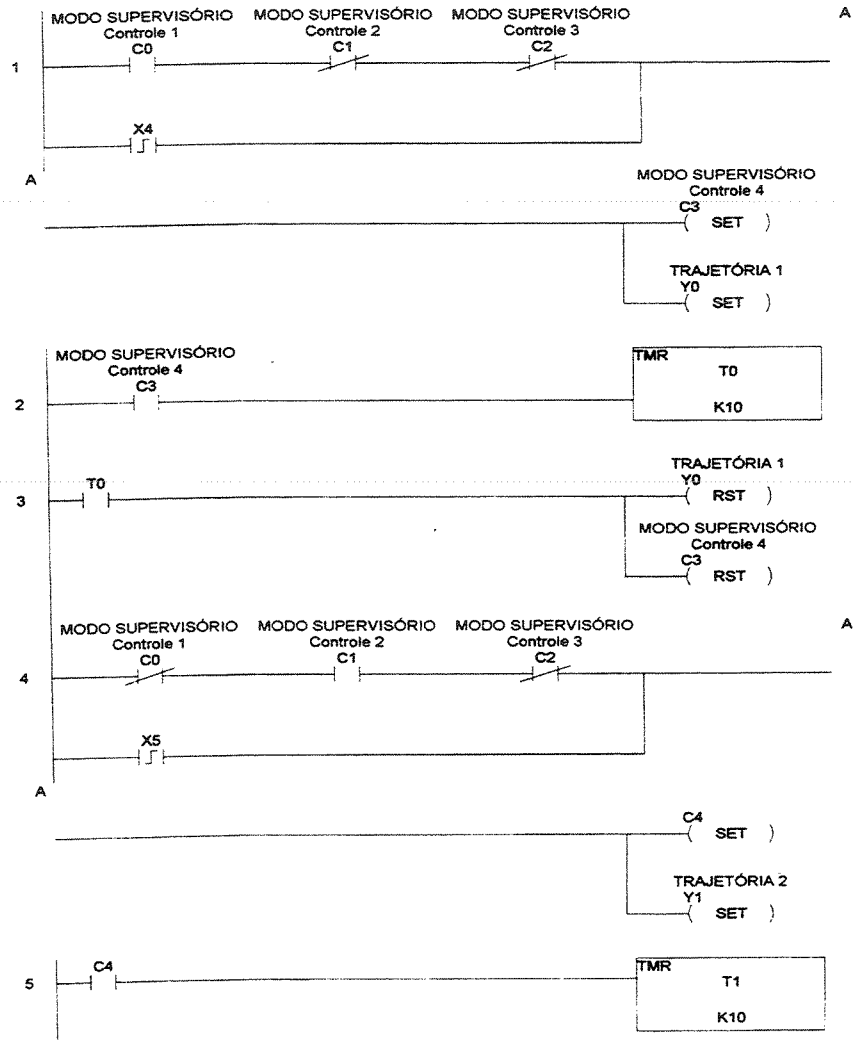

36
Start/Stp
X4
Ciclo
C5
EMERGÊNCIA
C6
SET
Start/Stop
C4
C16
C16
RST
C17
AUTOMATICO
Y5
RST
EMERGENCY
Y4
SET
Inicio
S0
JMP
ISG
Montagem 3
S5
38
Ciclo
C5
TMR
T0
K200
39
TA0
≥
K1
TA0
<
K50
Decarregamento
RED
Y1
OUT
Descarregamento
VERMELHO
C10
OUT
40
TA0
≥
K50
TA0
<
K100
Decarregamento
YELOW
Y2
OUT
Descarregamento
AMARELO
C11
OUT

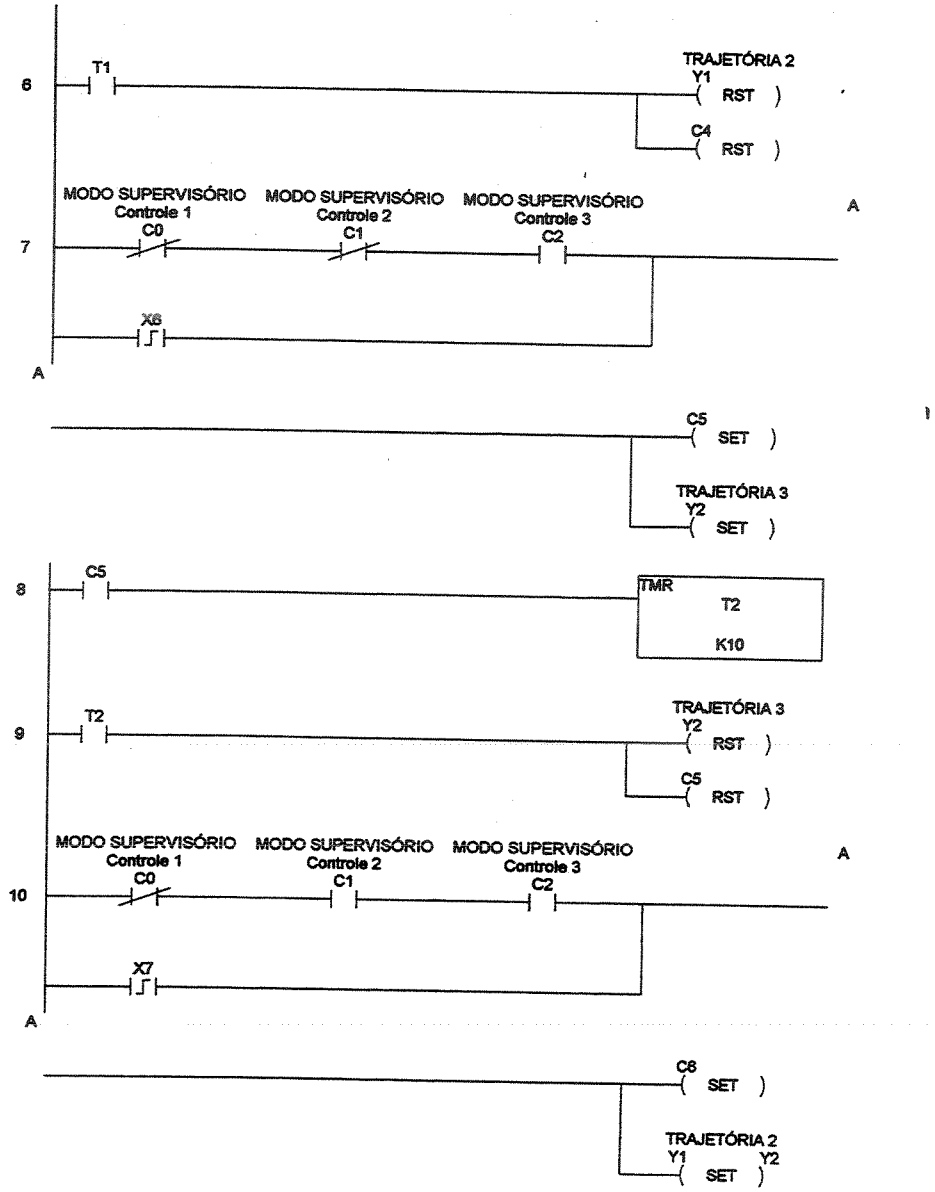

41
TA0
≥
K100
TA0
<
K150
Decarregamento
BLUE
Y0
OUT
Descarregamento
AZUL
C7
OUT
42
TA0
≥
K150
Ciclo
C5
RST
AUTOMATICO
Y5
RST
43
Prd3
X2
Ciclo
C5
C17
Modo Manual
S2
JMP
C15
SET
44
Produto3
Prod3
C2
Ciclo
C5
C16
Modo Supervisor
S1
JMP
C15
SET
45
C16
Produto3
Prod3
C2
Inicio
S0
JMP
C17
Prd3
X2
AUTOMATICO
Y5
RST

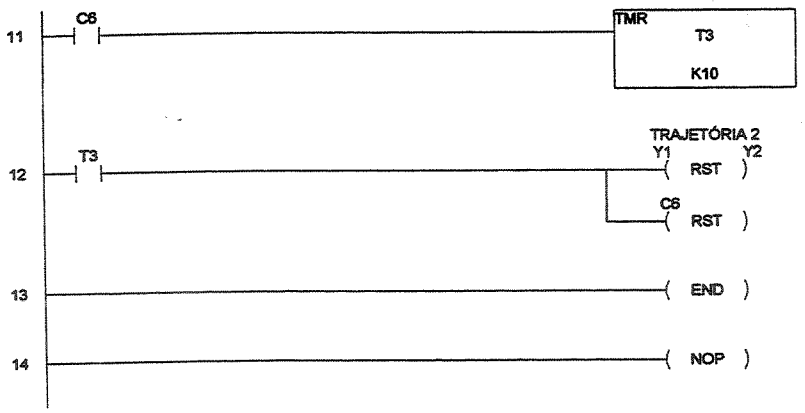

46
Start/Stp
X4
Ciclo
C5
EMERGÊNCIA
C6
SET
Start/Stop
C4
C16
C16
RST
C17
AUTOMATICO
Y5
RST
EMERGENCY
Y4
SET
Inicio
S0
JMP
47
END
48
NOP

Apêndice C

Programação Ladder (Telecirurgia)




6
T1
TRAJETÓRIA 2
Y1
RST
C4
RST
7
MODO SUPERVISÓRIO
Controle 1
C0
MODO SUPERVISÓRIO
Controle 2
C1
MODO SUPERVISÓRIO
Controle 3
C2
A
X6
A
C5
SET
TRAJETÓRIA 3
Y2
SET
8
C5
TMR
T2
K10
9
T2
TRAJETÓRIA 3
Y2
RST
C5
RST
MODO SUPERVISÓRIO
Controle 1
C0
MODO SUPERVISÓRIO
Controle 2
C1
MODO SUPERVISÓRIO
Controle 3
C2
A
10
X7
A
C6
SET
TRAJETÓRIA 2
Y1
Y2
SET


11
C6
TMR
T3
K10
12
T3
TRAJETÓRIA 2
Y1
Y2
RST
C6
RST
13
END
14
NOP

Apêndice D

Programação de Tarefas (Robô ABB)

```
%%%
  VERSION:1
  LANGUAGE:ENGLISH
%%%

MODULE <ID>
  CONST robtarget p50:=[[60.32,729.16,1202.08],[0.158683,-
0.72532,0.658322,0.123871],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget paprox7:=[[539.65,583.25,1354.69],[0.297063,-
0.418515,0.857163,0.043247],[0,-1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p40:=[[539.66,583.25,1354.69],[0.297062,-
0.418509,0.857167,0.043248],[0,-1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p30:=[[754.55,-117.34,656.99],[0.001638,-
0.011618,0.99993,0.001459],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p19:=[[754.55,-117.32,657],[0.001615,-
0.011666,0.99993,0.001471],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p18:=[[757.78,-55.21,660.34],[0.001634,-
0.011652,0.99993,0.001482],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p17:=[[789.5,3.8,656.36],[0.001671,-
0.011632,0.99993,0.00147],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget paprox6:=[[789.5,3.78,674.37],[0.001652,-
0.011694,0.999929,0.00146],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p16:=[[779.5,5.77,678.37],[0.001679,-
0.01167,0.99993,0.001457],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p15:=[[779.49,5.78,657.38],[0.001656,-
0.011733,0.999929,0.001457],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p14:=[[750.02,-48.3,658.59],[0.00169,-
0.011732,0.999929,0.001437],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget paprox5:=[[789.49,3.78,674.37],[0.001649,-
0.011697,0.999929,0.001458],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p13:=[[745.04,-114.3,659.45],[0.001712,-
0.01175,0.999929,0.001427],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p12:=[[764.98,-119.6,660.46],[0.001721,-
0.011812,0.999928,0.001418],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
  CONST robtarget p11:=[[833.07,-63.85,662.5],[0.001707,-
0.011916,0.999927,0.001427],[-1,0,- 1,0],
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p20:=[[754.55,-117.36,681.98],[0.001632,-
0.011618,0.99993,0.001448],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p10:=[[771.77,6.63,659.96],[0.001736,-
0.011905,0.999927,0.001429],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p9:=[[707.62,-54.54,657.14],[0.001766,-
0.011871,0.999927,0.001431],[-1,0,-1,0],
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget paprox4:=[[745.06,-114.31,676.45],[0.00174,-
0.01174,0.999929,0.001434],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p8:=[[865.9,-56.66,661.83],[0.001833,-
0.011895,0.999927,0.001441],[-1,0,-1,0],
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p7:=[[770.68,93.67,662.13],[0.001816,-
0.011949,0.999926,0.001456],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p6:=[[675.34,-56.06,660.29],[0.001855,-
0.011953,0.999926,0.001461],[-1,0,-1,0],
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p5:=[[769.96,-205.7,659.46],[0.001932,-
0.011905,0.999927,0.001429],[-1,0,-1,0].
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget paprox3:=[[707.65,-54.55,682.17],[0.001811,-
0.011855,0.999927,0.001426],[-1,0,-1,0],
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p4:=[[895.57,-236.8,662.8],[0.001945,-
0.011958,0.999926,0.001419],[-1,0,-1,0],
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p3:=[[896.85,123.72,661.96],[0.001972,-
0.011961,0.999926,0.001427],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p2:=[[645.55,124.28,660.08],[0.002002,-
0.011902,0.999926,0.001446],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget p1:=[[644.07,-235.84,659.58],[0.002033,-
0.011862,0.999927,0.00145],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget paprox2:=[[769.97,-205.72,683.49],[0.001937,-
0.011936,0.999926,0.001415],[-1,0,-1,0],
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget paprox1:=[[644.09,-235.85,684.6],[0.002068,-
0.011857,0.999927,0.001445],[-1,0,-1,0],
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget phome:=[[60.32,729.15,1202.07],[0.158679,-
0.725316,0.658327,0.123872],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  PERS tooldata Bic:=[TRUE,[[0.500839,-
0.574904,226.276],[1,0,0,0]],[0.25,[85,0,65],[1,0,0,0],0.01,0.01,0.01]];

  PROC Circulo()
    MoveJ paprox3,v400,z50,Bic;
    MoveL p9,v200,fine,Bic;
    MoveC p10,p11,v200,z5,Bic;
    MoveC p12,p9,v200,z5,Bic;
    MoveJ paprox3,v100,z50,Bic;
    MoveJ paprox4,v100,z50,Bic;
    MoveL p13,v400,fine,Bic;
    MoveC p14,p15,v100,z10,Bic;
    MoveL p16,v100,fine,Bic;
    MoveJ paprox5,v100,fine,Bic;
```

```
  MoveL p17,v100,fine,Bic;
  MoveC p18,p19,v100,fine,Bic;
  MoveL p20,v400,fine,Bic;
ENDPROC

PROC Losango()
  MoveJ paprox2,v400,z50,Bic;
  MoveL p5,v200,fine,Bic;
  MoveL p6,v200,fine,Bic;
  MoveL p7,v200,fine,Bic;
  MoveL p8,v200,fine,Bic;
  MoveL p5,v200,fine,Bic;
  MoveJ paprox2,v400,z50,Bic;
ENDPROC

PROC Retangulo()
  MoveL paprox1,v400,z10,Bic;
  MoveL p1,v200,fine,Bic;
  MoveL p2,v200,fine,Bic;
  MoveL p3,v200,fine,Bic;
  MoveL p4,v200,fine,Bic;
  MoveL p1,v200,fine,Bic;
  MoveJ paprox1,v400,z50,Bic;
ENDPROC

  PROC main()
  IF DI10_1=1 THEN
    Set DO10_6;
    Retangulo;
    Reset DO10_6;
  ELSE
    IF DI10_2=1 THEN
      Set DO10_6;
      Losango;
      Reset DO10_6;
    ELSE
      IF DI10_3=1 THEN
        Set DO10_6;
        Circulo;
        Reset DO10_6;
      ELSE
        IF DI10_4=1 THEN
          Set DO10_6;
          MoveJ phome,v400,z60,Bic;
          Reset DO10_6;
        ELSE
          IF DI10_5=1 THEN
            Set DO10_6;
            Losango;
            Circulo;
            Retangulo;
            Reset DO10_6;
          ELSE
            RETURN;
          ENDIF
        ENDIF
      ENDIF
```

```
      ENDIF
    ENDIF
  ENDPROC
ENDMODULE
```