MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Desenvolvimento e Implementação de uma Interface de Comunicação para Controlador Robótico Industrial

por

Fernando Mariano Bayer

Dissertação para obtenção do Título de
Mestre em Engenharia

Porto Alegre, abril de 2004

Desenvolvimento e Implementação de uma Interface de Comunicação

para Controlador Robótico Industrial

por

Fernando Mariano Bayer

Engenheiro Mecânico

Dissertação submetida ao Corpo Docente do Programa de Pós-Graduação em Engenharia Mecânica, PROMEC, da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como parte dos requisitos necessários para a obtenção do Título de

Mestre em Engenharia

Área de Concentração: Processos de Fabricação

Orientador: Prof. Dr. Flávio José Lorini

Comissão de Avaliação:

Prof. Dr. Eduardo André Perondi, UFRGS

Prof. Dr. José Antonio Esmerio Mazzaferro, UFRGS

Prof. Dr. Luiz Carlos Gertz, ULBRA

Prof. Dr. Jun Sérgio Ono

Coordenador do PROMEC

Porto Alegre, 26 de Abril de 2004.

À Carmen, minha eterna namorada.

AGRADECIMENTOS

Agradeço aos meus pais, por todo o apoio desde o início desta jornada.

A Carmen, minha eterna namorada, pelo carinho, compreensão e por compartilhar comigo minhas aflições e alegrias.

Ao professor Lorini, pela orientação, paciência e amizade.

Aos colegas, professores e funcionários do PROMEC, pela amizade e companheirismo.

A CAPES, pelo apoio financeiro.

Enfim, agradeço a Deus, pois sem ele nada disso seria possível.

RESUMO

O presente trabalho trata do desenvolvimento de um sistema para comunicação bidirecional entre um robô *ABB IRB1400*, originalmente desprovido de uma interface de comunicação de dados, e um microcomputador PC padrão. Para a implementação utilizou-se a porta paralela do PC e uma Placa E/S Digital para sinais discretos disponível no controlador do robô. Devido à diferença de características elétricas entre as interfaces utilizadas, foi necessário projetar um dispositivo que permitisse o ajuste dos níveis de tensão entre os sinais.

O sistema foi elaborado visando sua utilização por programas desenvolvidos pelos usuários em ambiente *Windows*, sendo disponibilizadas rotinas para envio e recebimento de dados através de um protocolo próprio. Na plataforma PC as rotinas estão encapsuladas em um arquivo compilado no formato *DLL* (*Dynamic Link Library*). No controlador do robô as rotinas foram criadas em linguagem *ABB RAPID*. O programa desenvolvido pelo usuário é responsável por todo o processamento das informações, que são então enviadas através do sistema de comunicação a um outro programa específico sendo executado no controlador do robô, o qual interpreta os dados e ativa as tarefas correspondentes.

Os resultados obtidos foram satisfatórios, sendo a velocidade de transmissão limitada pela velocidade da Placa E/S do robô. Utilizando-se uma placa *ABB DSQC 223*, atingiram-se taxas de transmissão da ordem de 12,5 *bytes*/s para envio e 6,1 *bytes*/s para o recebimento de informações a partir do PC. O sistema demonstrou ser uma alternativa viável para o controle do robô através de um microcomputador PC, apresentando boa confiabilidade, baixo custo e facilidade de implementação.

Abstract

Development and Implementation of a Communication Interface for an Industrial Robotic Controller

The present work presents the development of a system for bidirectional communication between an industrial robot, originally unprovided of a data communication interface, and a standard Personal Computer. For the implementation the PC's parallel port and a Digital I/O Board available in the robot's controller it was used. Because of the difference of electric characteristics between the used interfaces, it was necessary to project a device that allowed voltage level adjustments between the signals. The system was elaborated to be implemented in users developed programs in the Windows environment, giving routines for send and receive data through a proprietary protocol. In the PC's platform the routines are encapsulated in an archive compiled in DLL (Dynamic Link Library) format. In the robot's controller the routines had been developed in ABB RAPID. The program developed for the user is responsible for all the processing of the information, that are then sent by the communication system to another specific program running in the robot's controller, which interprets the data and activeted the corresponding tasks. Satisfactory results, were obtained being the transmission speed limited by the robot's Digital I/O Board. Using an ABB DSQC 223 board, transmission speeds of up to 12,5 bytes/s for sending and 6,1 bytes/s for receiving of information from the PC were reached. The system demonstrated to be a viable alternative for the robot control through a microcomputer PC, presenting a good trustworthiness, low cost and easiness of implementation.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE SÍMBOLOS:

ABB	*Asea Brown Boveri*
ASCII	*American Standard Code for Information Interchange*
CAN	*Control Area Network*
CNC	*Comando Numérico Computadorizado*
CSMA/CD	*Carrier Sense Multiple Access with Collision Detection*
DIN	*Deutsches Institut für Normung*
DLL	*Dinamic Link Library*
GPFAI	Grupo de Projeto, Fabricação e Automação Industrial
HART	*Highway Addressable Remote Transducer*
IEC	*International Electrothecnical Commission*
IEEE	*Institute for Electrical & Electronic Engineers*
ISA	*Instrument, Systems and Automation Society*
ISO	*International Standarts Organization*
ISP	*Interoperable Systems Project*
LAN	*Local Area Network*
MAP	*Manufacturing Automation Protocol*
OSI	*Open Systems Interconnect*
PC	*Personal Computer*
SPP	*Standart Parallel Port*
TCP/IP	*Transmission Control Protocol / Internet Protocol*
TOP	*Technical Office Protocol*
UFRGS	Universidade Federal do Rio Grande do Sul
WAN	*Wide Area Network*

1. Introdução

A utilização industrial da robótica iniciou-se na década de 1960, tendo como principal objetivo atender às grandes exigências da produção em massa. Sua principal aplicação foi à automação de processos que necessitavam de movimentação complexa, alta repetitividade e principalmente apresentassem problemas relacionados à saúde ocupacional.

Com o decorrer do tempo a demanda de produção mudou da produção em altas escalas para a produção de lotes menores de produtos diversificados, e junto a isto, iniciou-se uma crescente automação das linhas de produção. O cenário atual caracteriza-se por linhas de produção altamente automatizadas e integradas em redes de chão-de-fábrica, gerando produtos diversificados e de grande complexidade.

Neste cenário, uma máquina que não permite a comunicação com sistemas automatizados de controle da produção perde grande parte de seu potencial de integração.

Os robôs industriais têm como característica uma grande robustez mecânica, permitindo uma longa vida útil de trabalho confiável. Porém a grande velocidade de avanço da eletrônica e o advento de novas tecnologias de comunicação, têm tornado seus controladores obsoletos em tempos menores devido à evolução do *hardware* e à própria necessidade de interfaceamento com novos equipamentos integrados ao ambiente de trabalho. Até pouco tempo um robô não necessitava mais do que um *drive* de disquete para carregar um programa, o qual seria repetido indefinidamente durante o ciclo de produção. Atualmente os programas devem ser substituídos freqüentemente devido às mudanças do produto.

Caracteriza-se assim um cenário em que em algumas situações dispõe-se de máquinas em boas condições de produção, porém impossibilitadas de obter uma integração eficiente com outros equipamentos devido à limitação de seus controladores .

Este trabalho tem como objetivo desenvolver um sistema alternativo de comunicação entre um microcomputador PC padrão e um robô industrial. Este sistema pretende ser uma solução de baixo custo de implementação, confiabilidade e flexibilidade de utilização, mantendo uma velocidade de transmissão que permita o controle em tempo real. A implementação compreende o desenvolvimento de dispositivos de *hardware* e do *software* aplicativo, de modo a viabilizar a comunicação entre a porta paralela de um PC e a Placa E/S de sinais digitais disponível no controlador do robô.

2. Revisão Bibliográfica

2.1. Histórico dos sistemas de controle

No início do século XX os sistemas de controle eram geralmente localizados próximos ao processo. A razão para este controle localizado era simples, os métodos de controle eram basicamente manuais, onde equipamentos mecânicos indicavam ao operador o estado do sistema e este respondia através controles de ajuste manual. Para a operação de plantas complexas era necessário um grande esforço de coordenação humana, não possibilitando uma forma de controle centralizado e tendo poucos recursos para otimização.

O surgimento de controles automáticos tornou-se possível a partir do desenvolvimento de métodos e padrões analógicos de transmissão de dados, sendo um dos primeiros o padrão pneumático 3-15 *psi*, trabalhando em conjunto com processos instrumentados mecanicamente. O padrão pneumático foi então amplamente substituído pelo padrão elétrico 4-20 mA, o qual realiza as mesmas tarefas, porém de modo mais eficiente, sendo um padrão utilizado até os dias atuais [Murugeshan, 2003].

Na década de 1960, a implementação mais comum de um sistema de controle em qualquer processo industrial era através de um controlador central, conectado diretamente a uma série equipamentos de entrada e saída através de cabos. Este controlador estimava uma variável de saída baseando-se em um valor de controle, nos valores das variáveis do processo e nos parâmetros de seu sistema de controle (ganho, integral e/ou derivada no tempo). Em alguns anos a tendência passou a ser centralizar estes controladores em salas de controle. A comunicação entre os sensores, atuadores e o controlador desenvolvia-se de forma analógica basicamente por dois meios [Mahalik, 2003]:

- **Sinais elétricos**, padrão 4 a 20 mA: onde os dados eram transmitidos fazendo-se variar a corrente do circuito do sensor de forma proporcional à variação do valor da variável de controle.

- **Sinais pneumáticos**, padrão 3 a 15 psig: neste sistema o valor da variável era proporcional à pressão na linha de comunicação.

Com o desenvolvimento da eletrônica e, principalmente, de dispositivos semicondutores, os sistemas analógicos puramente pneumáticos tornaram-se obsoletos. No início dos anos 1980 os sinais elétricos analógicos provindos do chão-de-fábrica passaram a ser conectados à condicionadores de sinais, onde era feita a conversão analógica/digital, sendo então processados por microprocessadores digitais. Os dados resultantes eram enviados para um conversor digital/analógico, sendo o sinal analógico utilizado para o controle de atuadores com posicionadores eletro-pneumáticos. Esta prática dominou o cenário fabril por mais de 10 anos, mantendo-se o conceito de sala de controle centralizada. No final da década de 1980, surgiram no mercado os transmissores "inteligentes", utilizando a transmissão de sinais digitais sobrepostos aos sinais analógicos, permitindo sistemas de diagnose remota, para esta aplicação destacou-se o protocolo *HART (Highway Addressable Remote Transducer)*[Mahalik, 2003].

Com o desenvolvimento de transmissores capazes de manipular toda a estratégia de controle usando blocos de funções integradas ao software e por comunicação ponto-a-ponto com equipamentos similares eliminou-se a necessidade do processamento centralizado das informações. Através da rede de automação da planta, controles foram levados para o chão-de-fábrica, mais próximos ao processo, trazendo de volta a estratégia de controle descentralizado. Em adição a crescente economia de escala, a automação com redes ajudou a operação da planta, reduzindo a necessidade de mão-de-obra, reduzindo as paradas de produção e aumentando significantemente a confiabilidade.

Uma importante vantagem da comunicação digital é que uma grande quantidade de informação pode trafegar em apenas um único par de cabos, ao contrário dos sistemas analógicos, que necessitam de um par para cada transmissor. Com o advento dos Sistemas de Controle Distribuído e dos Controladores Lógicos Programáveis, houve uma migração para o uso de painéis locais, próximos ao processo, e com o uso da comunicação digital, todos os dados de supervisão e controle de milhares de instrumentos podem ser transmitidos através de uma única rede de chão-de-fábrica [Murugeshan, 2003].

A Tabela 2.1 apresenta um breve histórico do desenvolvimento das redes de chão-de-fábrica.

Tabela 2.1 - Histórico do desenvolvimento das redes para automação industrial.

Ano	Evento
1969	Publicação do *Recommended Standarts RS 232* pela *Electronic Industry Association (EIA).*
Década 1970	Desenvolvimento dos Microprocessadores e Sistemas de Controle Distribuído
1980	Publicação da especificação para redes *Ethernet*, pela Xerox, Digital e Intel.
1982	Início do desenvolvimento do *MAP* (*Manufacturing Automation Protocol*), apoiado principalmente pela *General Motors Company*.
1984	Publicação do *Open Systems Interconnect (OSI)* como *ISO 7498,* formando as bases para redes de automação industrial.
1985	Início do desenvolvimento do *TOP* (*Technical Office Protoco*l), em inciativa liderada pela *BOEING*.
1985	*Instrument, Systems and Automation Society (ISA), SP50 Committee* e *International Electrotechnical Commission (IEC)* iniciaram a elaboração do *Fieldbus Standart 61158*.
1990	Formado o *Interoperable Systems Project (ISP)* para desenvolver tecnologias para *Fieldbus*.
1994	Primeiras redes *Fieldbus* testadas nos Estados Unidos.

Os protocolos mais usados atualmente em redes de automação industrial são [Murugeshan, 2003]:

- *Foudation Fieldbus:* Estabelecida em 1994, é mantida pela *Fieldbus Fundation* que é uma organização composta por cerca de 140 membros. Fazem parte os principais fornecedores e usuários de sistemas de controle e automação. Foi projetado para suportar aplicações de alta confiabilidade e é uma rede interoperável (detalhes na seção 2.7) com padrão aberto baseada nas sete camadas do Modelo *OSI* (*Open Systems Interconnect*, descrito na seção 2.4) ;

- *Profibus:* Estabelecida com um Padrão Nacional Alemão (*DIN* 19245) em 1989, sendo certificada como um Padrão Europeu *EN* 50170 em 1996 e como Padrão Internacional *IE* 61158 em 2000. Suporta uma série de novas tecnologias permitindo um amplo uso em aplicações industriais.

- *ControlNet* : Desenvolvida pela *Allen-Bradley*, representa o estado-da-arte em redes de controle, atendendo às demandas de controle em tempo real e aplicações com altas taxas de transmissão. Todos os equipamentos em uma rede *ControlNet* podem ser configurados para ter

acesso direto a dados em redes *Foundation Fieldbus* através de um *ControlNet-to-Foundation Fieldbus Linking Device.*

- *WorldFIP:* Apresenta conformidade com o Padrão Europeu *EN* 50170, sendo uma tecnologia preferencial para todas as aplicações do setor público e de utilidades para a industria na União Européia.

- *Modbus*: Desenvolvido pela *Modicon* em 1979, é uma estrutura de mensagens que estabelece comunicação mestre/escravo ou cliente/servidor entre dispositivos inteligentes. É um padrão de protocolo de rede aberto e amplamente utilizado em ambientes de manufatura.

- *Interbus:* Desenvolvido para a troca de dados de processo entre sensores e atuadores em ambiente industrial. Devido a seu procedimento de transmissão e sua topologia em anel, oferece características como fácil manipulação e instalação, e todos os pré-requisitos para o uso de fibras-óticas.

- *CAN (Controller Area Network)*: Primariamente desenvolvido para aplicações automotivas, tem conformidade com o padrão ISO 11898 para comunicações seriais. Seu protocolo é normalmente implementado por *hardware*. Apesar de servir principalmente para aplicações automotivas e para máquina móveis, também tem se difundido amplamente em aplicações de automação industrial.

-*LonWorks*: Desenvolvida pela *Echelon Corporation* para prover soluções de controle para edifícios, indústrias, transporte e automação doméstica. Cada nó de uma rede *LonWorks* possui um dispositivo próprio para o controle do protocolo de comunicação e para realizar as tarefas de controle. Dispositivos como sensores de proximidade, chaves, relés e controladores de motores podem servir como nós da rede.

2.2. Modelo de Hierarquia de um sistema de Automação Industrial

Existem vários níveis de automação a serem considerados em uma planta de manufatura. Estes níveis podem ser classificados de diferentes formas, como o modelo de quatro níveis proposto por Boucher [1996], e que apresenta a seguinte hierarquia:

1. nível de Máquina;
2. nível de Linha de Produção ou Célula de Trabalho;
3. nível de Chão-de-fábrica;

4. nível da Planta.

As atribuições de cada nível deste modelo de hierarquia são ilustradas na Tabela 2.2.

Tabela 2.2 – Modelo de hierarquia de uma rede de Automação Industrial

Nível Hierárquico	Atribuições Comuns
Nível 4 **Planta**	- Processamento de Pedidos de Clientes; - Compras; - Planejamento de Produção; - Contabilidade.
Nível 3 **Chão-de-fábrica**	- Gerenciamento de Materiais; - Controle de Qualidade; - Planejamento da Produção.
Nível 2 **Célula de Trabalho ou Linha de Produção**	- Manipulação de Materiais; - Sequenciamento da Produção; - Inspeção e Controle Estatístico do Processo.
Nível 1 **Máquina**	- Máquinas CNC; - Robôs; - Controladores Programáveis.

Os objetivos da Automação do Nível de Máquina (Nível 1) são assegurar que as operações da máquina correspondam à seqüência de operações planejada. Tipicamente a seqüência de operações que deve ser seguida é prescrita no programa residente na máquina e não há decisões a serem tomadas. Um exemplo deste nível é o controle de máquinas *CNC* e robôs Industriais.

O problema da Automação na Linha de Produção ou Célula de Trabalho (Nível 2) é tomar decisões locais para coordenar as atividades de vários equipamentos. Típicas decisões neste nível consistem em controlar o movimento de componentes entre as máquinas da célula de manufatura e verificar dados de inspeção durante o processo, eliminando componentes fora da especificação.

No Nível de Chão-de-fábrica (Nível 3), o problema a ser resolvido é coordenar as atividades entre diversas células de manufatura ou linhas de produção. Um exemplo é o transporte de matéria prima do depósito para a linha de produção, e o transporte do produto acabado para o estoque. Coordenar estes sistemas de transporte, que normalmente são compartilhados pelas linhas de produção, é um problema para a Automação de Chão-de-fábrica, pois se situa em um nível acima de qualquer célula de trabalho.

Automação no Nível de Planta (nível 4) é menos voltado ao dia-a-dia da fábrica e mais aos objetivos de negócio da empresa, sendo encarregado de automatizar o processo de tomada de decisões global da empresa, ou ao menos, dar suporte a este processo através de sistemas de informação. Um exemplo é o planejamento do uso da capacidade de produção de modo a atender a demanda dos clientes em um período de um ano. O resultado deste plano é a definição de quantas unidades serão produzidas em cada mês do período analisado.

A Figura 2.1 apresenta um diagrama hierárquico de uma planta industrial.

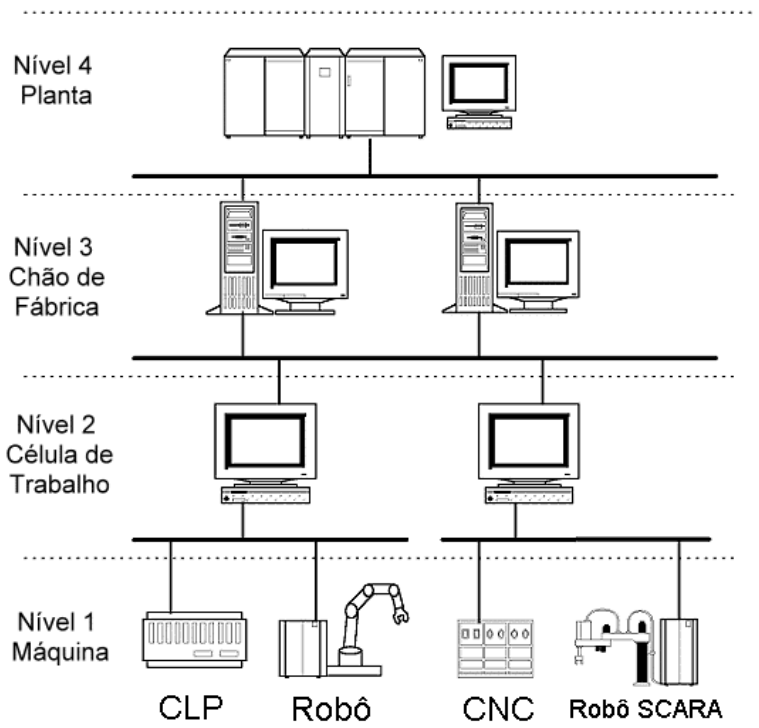


Figura 2.1 – Modelo de hierarquia de uma rede de Automação Industrial

A hierarquia de controle ilustrada acima começa no nível mais alto, onde são considerados os aspectos mais corporativos, e segue para os níveis mais baixos, responsáveis por questões operacionais. Este paradigma leva a modelagem do problema de manufatura como uma hierarquia de decisões, onde cada nível impõe restrições ao nível imediatamente inferior. O objetivo é passar cada decisão de controle para o nível mais baixo possível na hierarquia. A integração completa de todos estes níveis de processos de decisão, suportados por um sistema de informações computadorizado, é normalmente chamada Manufatura Integrada por Computador (*CIM*).

2.3. Redes Locais

Provavelmente o problema mais complicado na automação de chão-de-fábrica seja o de interconectar máquinas e processos, principalmente devido à tendência histórica de cada fabricante de utilizar seu próprio protocolo de comunicação. Porém com o crescimento da demanda pela integração de máquinas em sistemas coordenados de produção, criou-se a necessidade de um meio comum sobre o qual os controladores dos equipamentos pudessem comunicar-se entre si e com computadores em níveis superiores.

Os computadores em uma fábrica comunicam-se entre si através de uma rede local (*LAN – Local Área Network*), definida pela norma ISSO 11898, sendo normalmente implementada em uma área limitada e de propriedade de uma certa organização [Buchanan, 2003]. Este sistema de comunicação privado que permite a comunicação entre diversos dispositivos integrados na rede, a distâncias que variam de poucos metros até vários quilômetros. Os equipamentos conectados a rede incluem, entre outros, computadores, controladores lógicos programáveis, máquinas *CNC*, robôs industriais e dispositivos coletores de dados.

2.3.1. Topologias de Redes

Fisicamente uma rede de comunicações não possui uma configuração hierárquica como a descrita no Capítulo 2.2. Existem três configurações, ou topologias, usadas em redes locais [Boucher, 1996]:

1 – **Estrela**: ilustrada na Figura 2.2, consiste de uma topologia onde existe um computador central de controle ao qual todas as estações dos usuários estão conectadas. Todas as mensagens entre os diversos dispositivos da rede devem passar por este computador central, que age como um gerente de tráfego e controla o fluxo de comunicação entre os dispositivos. Esta configuração apresenta como inconveniente técnico o fato de um problema no computador central tornar toda a rede inoperante.

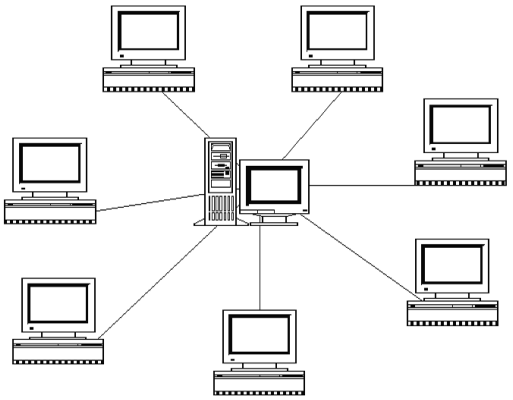


Figura 2.2 – Rede com topologia em Estrela

2 – **Anel**: nesta topologia, apresentada na figura 2.3, as estações estão conectadas em um anel contínuo. Não há um computador central ou um controle hierárquico da rede. Cada estação tem seu endereço único, recebe mensagens da estação imediatamente anterior e, caso esta mensagem não seja endereçada a ela, retransmite para a próxima estação, até atingir o destinatário.

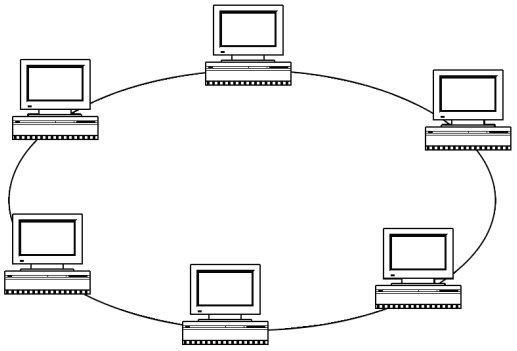


Figura 2.3 – Rede com topologia em Anel

3 – **Barramento**: é a topologia mais comum em aplicações de chão-de-fábrica. Como ilustrado na figura 2.4 é usada uma linha tronco onde são ligadas todas as estações. Quando uma estação envia uma mensagem, esta é recebida por todas as estações da rede, porém a única estação que irá copiá-la é a estação para a qual a mensagem está endereçada. Esta topologia é adotada em dois importantes padrões de rede, *IEEE 802.3* (*Ethernet*) e *IEEE 802.*4 (*Token-Bus*). Sendo também amplamente utilizada em redes proprietárias de vários fabricantes.

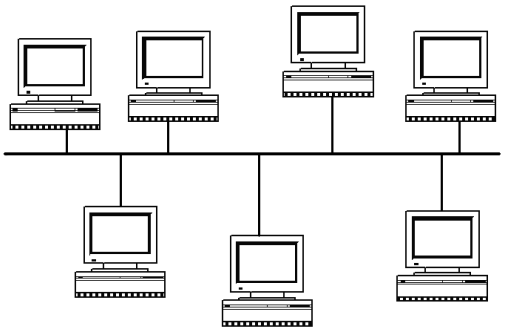


Figura 2.4 – Rede com topologia em Barramento

2.3.2. Método de Acesso

Uma propriedade importante que diferencia os diversos tipos de rede é a maneira como a transmissão é controlada, ou seja, o sistema que disciplina o acesso a transmissão na rede pelas diversas estações. Um exemplo é a rede *Ethernet*, que utiliza a técnica chamada *Carrier Sense Multiple Access with Collision Detection (CSMA/CD)* [Boucher,1996]. Neste método quando uma estação necessita transmitir uma mensagem, esta verifica se a linha está livre para transmissão e, em caso afirmativo, envia a mensagem contendo o endereço da estação destino. Caso duas estações tentem enviar mensagens ao mesmo tempo existirá uma colisão e os dados serão corrompidos. Quando uma colisão é detectada as estações entram em uma rotina de recuperação, que inclui a espera por um período de tempo aleatório até a nova tentativa de transmissão. Como é possível haver várias colisões consecutivas, não há como prever o tempo necessário para o envio de uma mensagem. Esta incerteza é um fator crítico em sistemas de automação, principalmente em aplicações de controle em tempo real, sendo a principal desvantagem das redes *Ethernet* em aplicações industriais.

Outro método amplamente utilizado para o controle de transmissão é *Passagem de Permissão*, ou *Token*. Sendo implementado nas normas *IEEE 802.5* (*Token Ring*) e *IEEE 802.4* (*Token Bus*) [Boucher, 1996]. Neste sistema apenas uma estação é autorizada a transmitir a cada momento, sendo esta permissão controlada através da passagem de um sinal de controle de estação para estação. Quando uma estação recebe o *Token* ela pode transmitir mensagens por um determinado período de tempo, e então deve passá-lo para a próxima estação da seqüência. Utilizando este sistema se evita o problema da colisão na transmissão, e permite-se que o tempo máximo para o envio de uma mensagem seja previamente calculado. Por esta razão o controle por *Token*, ou variações deste sistema, tem sido amplamente utilizados em aplicações onde controle em tempo real é envolvido.

2.4. O Modelo de Referência *OSI*

Como um esforço para encorajar a padronização no desenvolvimento de redes de comunicação, o *International Standarts Organization (ISO)* estabeleceu um modelo de referência que descreve como as redes devem ser desenvolvidas em termos de camadas de responsabilidade. Este modelo é conhecido como o modelo *OSI (Open System Interconnect)*, e tem o propósito de dividir o problema da comunicação em uma série de subproblemas.

O modelo *OSI* é estruturado em sete camadas, assim definidas [Mahalik, 2003]:

1. Camada Física;
2. Camada de Enlace de Dados;
3. Camada de Rede;
4. Camada de Transporte;
5. Camada de Sessão;
6. Camada de Apresentação;
7. Camada de Aplicação.

A figura 2.5 apresenta duas estações de uma rede, e a relação entre as sete camadas. Cada camada interage somente com a imediatamente superior e a imediatamente inferior. Para uma mesma rede, os serviços de cada camada, em cada estação, devem ser idênticos.

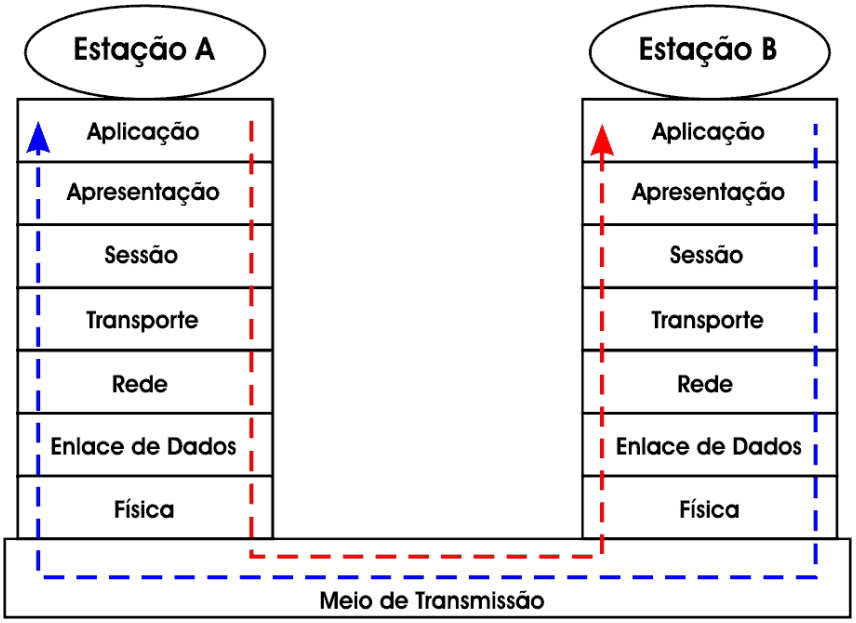


Figura 2.5 – Comunicação entre estações segundo o modelo *OSI* [Boucher,1996]

1. **Camada Física**: é o nível mais próximo do hardware. Tem a função fundamental de assegurar o formato físico dos dados adequado ao seu transporte através de sinais elétricos (digitais, analógicos, conforme o meio de transmissão). Em termos de recepção retira os sinais da rede no formato em que foram transmitidos e converte-os em *bits*. Em geral, neste nível cabe definir questões como: tipos de cabo a utilizar, os conectores, os níveis de tensão, as freqüências necessárias para codificar os *bits* e a taxa de transferência.

2. **Camada de Enlace de Dados**: é responsável pela estruturação de dados com vista ao seu envio pela rede, cabendo também um certo nível de controle de transmissão, identificando eventuais sinais de erros de recepção, e assegurando a requisição para o reenvio de um bloco que não tenha chegado em boas condições quanto à recepção. É responsável de igual modo pela estruturação do formato de bits provenientes da rede, para um formato que possa ser entendido por níveis superiores.

3. **Camada de Rede**: é responsável por estabelecer conexão e assegurar o correto encaminhamento da informação, de acordo com o destinatário ou nó da rede a quem se dirige à comunicação.

4. **Camada de Transporte**: supervisiona o serviço da rede, averiguando se o fluxo da comunicação está decorrendo adequadamente entre o emissor e receptor. Prepara os dados para serem transmitidos pela rede, subdivide-os em segmentos e acrescenta testes de verificação de controle de erros que acompanham os dados.

5. **Camada de Sessão**: gerencia a sessão de comunicação entre os nós envolvidos, assegurando que se inicie, que ocorra e que termine de forma conveniente, cabendo a este nível manter as partes de uma mensagem recebida até que esteja completa e pronta a ser remetida ao nível de apresentação.

6. **Camada de Apresentação**: tem a função genérica de providenciar para que os dois computadores envolvidos na comunicação se entendam entre si, com relação ao formato dos dados transmitidos; ao nível da emissão assegura que os dados sejam convertidos para um formato adequado ao receptor. Pode também proceder ao envio e compactação dos dados, ao nível da recepção converte-os para um formato inteligível descompactando-os e decodificando se for necessário.

7. **Camada de Aplicação**: é o nível mais próximo do usuário, as funções do nível de aplicação são as seguintes: ao nível de emissão, converte a informação do usuário ou de um formato digital adequado a ser passado ao nível seguinte com vista à sua transmissão e acrescenta uma indicação

relativa ao setor. Ao nível da recepção faz o inverso, converte os dados recebidos de uma transmissão num formato inteligível para aplicação do usuário.

É importante reiterar que o modelo *OSI* tem como objetivo apenas ser uma referência, servindo como guia no desenvolvimento do sistema de comunicação, não sendo necessária à implementação de todas as camadas em uma rede. Por exemplo, quando todos os equipamentos estão em uma mesma rede e utilizando os mesmos protocolos de apresentação, caracteriza-se uma situação em que não há necessidade da implementação das camadas de *Rede, Transporte, Sessão e Apresentação*. Estas camadas são mais importantes em redes amplas, onde o roteamento entre os nós é mais complexo, e diferentes formas de encriptação de dados são usadas [Boucher, 1996].

2.5 O protocolo HART (Highway Addressable Remote Transducer).

O *HART* foi um dos primeiros protocolos de comunicação digital a ser desenvolvido para uso industrial e utiliza-se da técnica de chaveamento por variação de freqüência baseada no padrão de comunicação *Bell 202* [Mahalik, 2003]. A comunicação digital é feita superpondo um sinal modulado em freqüência de 1mA pico a pico, ao sinal de corrente 4-20 mA como mostrado na Figura 2.6. Este sinal usa duas freqüências predefinidas de 1.200 Hz e 2.400 Hz representando os dígitos “1” e “0”, respectivamente. Os sinais de freqüência apresentam uma forma de onda senoidal, de média zero, portanto ao ser superposto ao sinal analógico de 4-20 mA não afetam seu valor. Com este sistema consegue-se comunicação bidirecional de 1.200 bps sem interferir no sinal analógico, mantendo-se a compatibilidade e utilizando-se o cabeamento do equipamento analógico existentes. A comunicação pode ser feita em 2 modos:

- Mestre/Escravo: o dispositivo mestre solicita um valor, o qual é então enviado pelo escravo. Nesta configuração consegue-se um ciclo de leitura típico de 500 ms, ou seja, duas leituras por segundo.

- Modo *Burst*: só possível em conexões ponto a ponto, onde um dos dispositivos é configurado para enviar de forma autônoma e periódica o valor da variável. Obtêm-se uma taxa típica de quatro leituras por segundo.

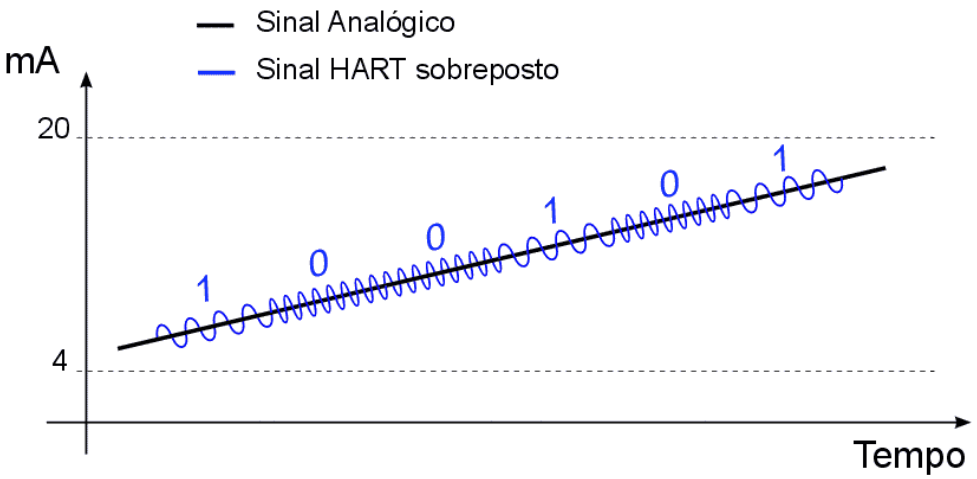


Figura 2.6 – Sinal digital *HART* sobreposto a um sinal analógico padrão.

A introdução do protocolo *HART* permitiu o aumento da flexibilidade e da performance dos instrumentos em campo, possibilitando a implementação de funções avançadas tais como diagnóstico remoto, aquisição de dados, configuração de valores de controle e calibração dos instrumentos, sem interferir na transmissão da variável de processo principal.

Alguns benefícios do Protocolo *HART*:

- permite comunicação analógica e digital na mesma linha;

- suporta múltiplos mestres em comunicação digital;

- permite redes de topologia multiponto;

- utiliza uma estrutura de mensagens comum em todos os instrumentos;

- provê uma estrutura de mensagem flexível.

Sua grande desvantagem reside nas baixas taxas de transmissão.

Devido a sua grande flexibilidade e simplicidade de implementação, uso e manutenção, o Protocolo *HART* vem se desenvolvendo em conjunto com a tecnologia *Fieldbus*, sendo largamente utilizado em uma grande variedade de aplicações. Atualmente estima-se que este seja o sistema de comunicação com a maior base instalada no mundo, em sistema de automação industrial [Mahalik,2003].

2.6. O Protocolo *MAP (Manufacturing Automation Protocol)*

Um dos principais problemas na implementação de redes é que os vários equipamentos da instalação nem sempre são capazes de se comunicar entre si. Mesmo que todos utilizem pulsos elétricos representando dados, o formato e interpretação destes pulsos difere de um equipamento para o outro. É necessário então um conjunto de procedimentos formalizados que possa ser seguido por cada elemento da rede, permitindo que os dados sejam transmitidos de forma padronizada. Este conjunto de padrões é chamado protocolo de comunicação [Groover, 1987].

No início da década de 80, um grande número de organizações se uniram para o desenvolvimento de um protocolo de comunicação chamado *MAP* (*Manufacturing Automation Protocol*), que é um conjunto de padrões desenhado para a utilização em redes locais de fábrica. Este protocolo foi amplamente adotado por industrias de manufatura devido principalmente ao esforço da *General Motors Corporation* em promover compatibilidade entre os equipamentos em suas plantas. A *GMC*, percebendo a necessidade de que equipamentos de fornecedores diferentes fossem capazes de comunicar-se entre si, iniciou um esforço para o desenvolvimento e adoção do protocolo *MAP*, exigindo que todos os fornecedores também satisfizessem esse padrão. Esta iniciativa acelerou a implementação deste protocolo em redes de manufatura.

Com referência aos padrões operacionais, o protocolo *MAP* usa barramento como topologia de rede, esquema de acesso através de *Token Bus*, transmissão em *broadband* (onde uma mensagem enviada por um computador chega igualmente a todas as máquinas conectadas na rede) e a uma taxa de 10 *Mbps*. A definição do protocolo segue o modelo de camadas *OSI* (*Open System Interconnection*).

Paralelamente à iniciativa *MAP*, e também nos Estados Unidos, um consórcio liderado pela *BOEING* lançou a iniciativa *TOP (Technical Office Protocol)*, com o mesmo objetivo do *MAP*, mas especialmente voltado para as áreas técnicas e administrativas da empresa.

Evidentemente as opções para as diferentes camadas do modelo *OSI* refletem os diferentes requisitos das aplicações *MAP* e das aplicações *TOP*. O *TOP* fornece essencialmente serviços relacionados com a transferência e interpretação de documentos de texto, de gráficos e técnicos. São exemplos desses serviços, o acesso e transferência de arquivos remotos, o acesso a terminais remotos, o correio eletrônico, a troca de documentos gráficos e de texto [Groover, 1987].

Atualmente, a iniciativa *TOP* está ligada à iniciativa *MAP*, inserindo-se no esforço global de normalização das redes de comunicação de dados.

2.7. O Padrão *FieldBus*

A comunicação digital e as tecnologias de processamento de sinais permitem inúmeros métodos diferentes para se obter uma comunicação robusta, com isso cada fabricante investiu em seu próprio protocolo para transmitir e processar dados. Essa situação leva muitas vezes a impossibilidade de obter comunicação direta entre equipamentos de diferentes fabricantes. Ao realizar-se uma expansão ou atualização nas instalações, o usuário fica obrigado a comprar sempre equipamentos do fornecedor original da planta. Para evitar estes problemas, na metade da década de 1980 iniciaram-se esforços por parte da *ISA* e do *IEC*, com o objetivo de desenvolver um protocolo de comunicação padrão com o qual sistemas e equipamentos de diferentes fornecedores pudessem se comunicar em uma rede comum chamada *Fieldbus* [Yuan, 2003]. Esta característica, chamada interoperabilidade, é uma das mais importantes dos padrões abertos *Fieldbus*.

O padrão *Fieldbus* define as seguintes camadas do modelo *OSI*:

Física (Camada 1): nesta camada são definidos dois padrões de barramento, H1 para baixas e H2 para altas velocidades de transmissão.

- H1: velocidades de até 31,25 kbps; de 2 a 32 dispositivos conectados; possibilidade de alimentação através do próprio barramento; cabeamento por par trançado.

- H2: velocidades entre 1 Mbps e 2,5 Mbps; até 127 dispositivos conectados; possibilidade de alimentação através do barramento, dependendo da configuração deste; cabeamento por par trançado blindado.

Comunicação (Camadas 2 e 7): dois tipos de mensagens são usadas:

- cíclicas, ou agendadas, usadas para o envio regular e periódico de informações de controle. Baseado no agendamento o Mestre da conexão emite a permissão para que o dispositivo envie uma mensagem.

- acíclicas: tipicamente usada para enviar comandos especiais tais como informações de diagnóstico ou alarmes.

Usuário (Camada 8): implementada acima da camada de Aplicação (figura 2.7), fornecendo estratégias de controle usando Blocos de Função. Algumas funções padronizadas são Entrada Analógica (*AI*), Saída Analógica (*AO*), Entrada Discreta (*DI*) e controles *PID*, entre outros. Para facilitar a interoperabilidade cada dispositivo *Fieldbus* possui uma Descrição de Dispositivo (*DD*) que inclui a descrição de todas as variáveis e procedimentos operacionais que podem ser necessárias.

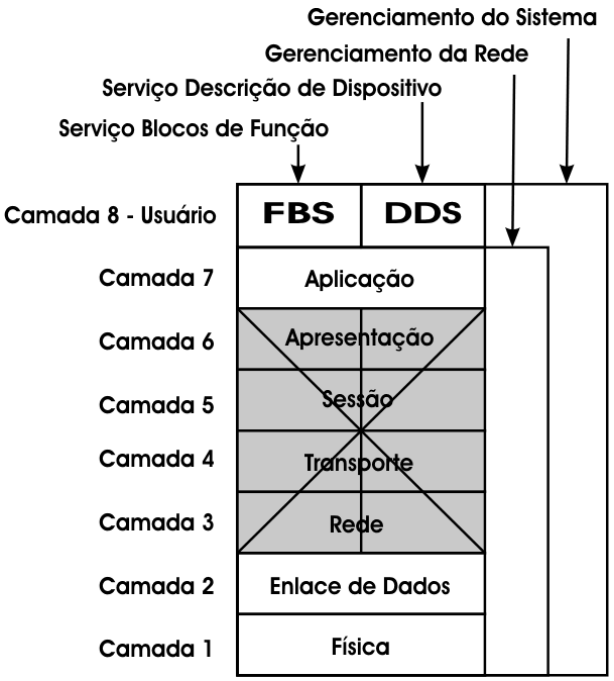


Figura 2.7 – Modelo OSI para o protocolo Fieldbus [Murugeshan, 2003].

Este padrão unificado deveria integrar a enorme gama de sistemas de instrumentação e controle com interfaces que permitissem operar vários dispositivos simultaneamente com um protocolo de comunicação comum. O objetivo mostrou-se muito complexo, não tendo conseguido a adesão esperada, ao menos nos primeiros 10 anos, e como muitos fornecedores receberam um protocolo de comunicação aberta com uma ameaça, continuaram a oferecer seus produtos com protocolos proprietários. Na prática, enquanto não se alcançam sistemas abertos completos e eficientes, o que vem ocorrendo é que os vários fabricantes têm aprendido a conviver mutuamente, criando produtos para que seus sistemas possam conversar com os sistemas concorrentes [Costa, 1995].

2.8. Redes *Internet* e *Ethernet* na Automação Industrial

Atualmente a Internet faz parte do dia a dia da indústria, não mais como uma simples fonte de pesquisa ou uma ferramenta acadêmica, mas sim como um instrumento de trabalho. Sua utilização vai

desde a simples correspondência entre os usuários através *do e-mail*, até sofisticados *sites* de comércio eletrônico e sistemas corporativos completos [Raji, 2003].

Essa difusão da *Internet* foi possível graças à padronização no modo em que dois computadores se comunicam. O protocolo *TCP/IP (Transmission Control Protocol / Internet Protocol)*, junto com a necessidade crescente da comunicação global, foi o grande responsável pela explosão da *Internet* nos últimos anos.

Com a padronização do protocolo *TCP/IP* na *Internet*, seu uso estendeu-se e passou a ser utilizado dentro da rede interna das empresas, na forma de *Intranet*, e recentemente na automação industrial, geralmente sobre uma estrutura de rede *Ethernet*.

O padrão de redes *Ethernet* foi desenvolvido originalmente pela *Xerox* com o objetivo de atender a automação de escritórios. Posteriormente, uma coalizão incluindo a *Digital Equipament Corporation e a Intel* acrescentou valor ao padrão ampliando a tecnologia. Devido a sua alta flexibilidade de leiaute e sua interoperabilidade com grande parte das redes de automação de escritórios, vários fornecedores de equipamentos de automação industrial estão criando soluções utilizando este padrão de comunicação em rede [Costa,1995].

Atualmente a Ethernet domina 80% do mercado de automação de escritórios e redes locais, e apresenta características de rápida inicialização, sistema de conexão simples, alta disponibilidade, facilidade de expansão e boa performance de comunicação [Thompson, 2003]. Tecnologias para integração completa da empresa também estão disponíveis e bem desenvolvidas, com possibilidade de interface com Redes de Área Ampla (*WAN –Wide Area Network*), como a *Internet*. Com o grande mercado atual a *Ethernet* demonstra ser uma tecnologia consolidada, garantindo segurança de investimento e constante atualização tecnológica, como a nova *Gigabit Ethernet* e as opções de comunicação sem-fio.

A especificação da rede *Ethernet* define apenas as duas primeiras camadas do modelo *OSI:*

Física (camada 1):

- Topologia tipo barramento, podendo-se optar por estrela com o uso de dispositivos tais como *hubs*.

- Cabeamento: atualmente o cabo mais utilizado é o par trançado *10baseT*, porém na especificação original o cabo coaxial *10base5* era definido como padrão. Podem ser usados ainda cabos coaxiais *10base2* ou mesmo fibra óptica.

Enlace de Dados (camada 2):

- Utiliza o protocolo de acesso ao meio *CSMA-CD*.
- Endereço único de 48-bit para os dispositivos da rede.
- Suporta endereçamento de mensagem individual, em grupos ou global.

As camadas superiores são normalmente implementadas através do protocolo *TCP/IP*, que é um conjunto de processos que atuam em cooperação, possibilitando que diferentes plataformas computacionais se comuniquem através de uma arquitetura de protocolos. O *TCP/IP* implementa as camadas 3 e 4 do modelo *OSI*, enquanto as redes tratam da troca de dados entre as estações, o *TCP/IP* trata dos pacotes de dados de forma independente. Os níveis superiores são implementados por diversos protocolos tais como *HTTP (Hipertext Transfer Protocol*), *FTP (File Transfer Protocol*) e *SMTP (Send Mail Transfer Protocol)* e mesmo protocolos específicos de cada fornecedor [Costa, 1995], conforme apresentado na figura 2.8.

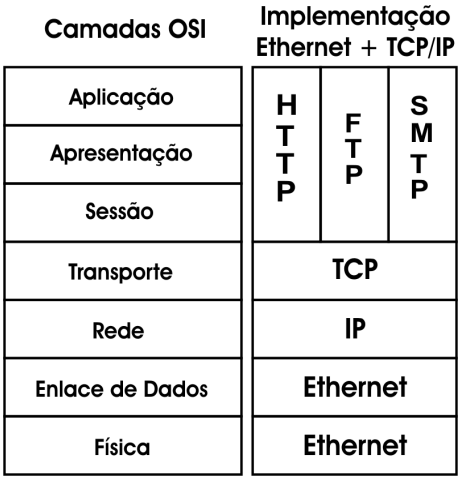


Figura 2.8 – Modelo OSI de uma rede Ethernet sob TCP/IP [Chantal, 2003]

Neste contexto, não surpreende o desenvolvimento da *Ethernet* como um solução de baixo custo para a indústria. As primeiras tentativas esbarraram no fato de que na concepção original a banda de transmissão é compartilhada e não dedicada, utilizando-se de *CSMA-CD*. Mesmo se dois *hubs* são conectados, continua-se tendo apenas um domínio de colisão, com todo o tráfego compartilhado, pois um *hub* é simplesmente um dispositivo que retransmite uma mensagem recebida em uma porta para todas as outras. Quando se acrescenta nós a rede, o tráfego cresce e com isso as chances de ser haverem colisões durante a comunicação, o que reduz a velocidade de comunicação. Nesta configuração torna-se impossível prever qual o tempo necessário para a transmissão de uma mensagem. Além disso não há nenhum conceito de prioridade de comunicação, portanto não se pode

diferenciar dados de baixa prioridade dos dados essenciais de controle, podendo por exemplo uma solicitação de impressão atrasar a transmissão de uma mensagem de controle importante.

Atualmente, com a introdução de novos dispositivos de controle, alguns destes problemas foram minimizados. Os primeiros foram os dispositivos chamados *bridges* que permitiram a conexão entre redes mantendo-se os domínios de colisão separados. O principal avanço entretanto foi a introdução dos *switches* em 1993 [Thompson, 2003]. Estes dispositivos combinam a multi-conectividade dos *hubs* com o roteamento seletivo de pacotes dos *bridges.*

Apesar do barateamento da tecnologia de redes *Ethernet* para automação de escritórios, esta ainda precisa ser adaptada para a sua utilização em ambiente industrial. Como resultado iniciou-se o desenvolvimento da *Ethernet* Industrial com o objetivo de oferecer uma estrutura de comunicação rápida, determinística e coesa, compatível com severo ambiente de chão-de-fábrica. Nos níveis mais altos desta estrutura os usuários podem utilizar protocolos já conhecidos, como *ModBus* e *ControlNet*, permitindo a comunicação entre equipamentos já existentes. Alguns fornecedores já apresentam soluções proprietárias, baseadas em protocolo *TCP/IP*. A adoção destes padrões proprietários porém leva a problemas de interoperabilidade, restringindo o usuário aos equipamentos de um mesmo fornecedor. Atualmente a *Ethernet* Industrial continua a ser um padrão em desenvolvimento, sendo formada a *Industrial Ethernet Association* com o objetivo de definir as especificações e padrões para o novo sistema.

Uma das principais características necessárias à *Ethernet* Industrial é a capacidade de diferenciar entre aplicações de tempo real, e garantir uma alta prioridade para estas informações. Isto significa ter vários níveis de serviços, garantindo que aplicações críticas não sejam afetadas por tráfego de baixa prioridade. Adicionalmente, a tecnologia deve ser estável, estar amplamente disponível e integrada com redes corporativas já existentes. Segurança, gerenciamento de rede e transmissões de longa distância também são características importantes.

Várias tecnologias estão atualmente em desenvolvimento, visando melhorar a velocidade e a tolerância a falhas. O padrão *Fast Ethernet* (*IEEE 802.3u*) opera a 100 *Mbps*, ou 200 *Mbps* bidirecional, ou seja, dez vezes mais rápida que o padrão original de 10 *Mbps*. No futuro, o padrão *Gigabit Ethernet* (*IEEE 802.3z*) deverá oferecer velocidade suficiente para eliminar grande parte dos

gargalos de rede. O padrão *IEEE 802.12* adiciona redundância à rede, facilitando a recuperação automática de falhas.

3. Desenvolvimento e Implementação

3.1. Especificação do Sistema

O objetivo deste trabalho é implementar um sistema de comunicação entre um robô industrial, no caso um *ABB IRB1400* equipado com um controlador *ABB S4*, e um PC padrão. Este sistema apresenta como características um baixo custo de implementação, confiabilidade e flexibilidade de utilização, mantendo uma velocidade de transmissão que permita o controle em tempo real.

Através de uma análise do problema e de suas possíveis soluções, concebeu-se uma definição inicial das características do sistema, considerando-se os seguintes aspectos:

- **Interface de comunicação do controlador**: a interface disponível é uma Placa E/S Digital *ABB DSQC 223*, a qual apresenta 16 canais de entrada e 16 de saída, destinada originalmente ao controle de dispositivos integrados à célula do robô, através de sinais discretos.

- **Interface de comunicação do PC**: as interfaces usuais de um PC padrão são a interface serial padrão *RS232* e a porta paralela. Optou-se pela utilização da porta paralela por disponibilizar os dados em um formato que pode ser transferido ao robô sem necessidade de manipulação, sendo que a pequena distância entre o robô e o PC torna o maior número de cabos necessários um fator irrelevante.

- **Implementação no PC**: com o objetivo de obter um sistema flexível, optou-se por desenvolver, para o *software* de interface, rotinas de comunicação e controle compiladas em um formato *DLL (Dinamic Link Library)*, o qual permite a integração do sistema à praticamente qualquer programa desenvolvido para a plataforma *Win32*. A programação foi realizada utilizando-se o ambiente de desenvolvimento *Dev-Pascal* em conjunto com o compilador *FreePascal*. Estas ferramentas são gratuitas e multiplataforma, possibilitando assim que o código seja portado para outros sistemas operacionais e também adaptado para outros modelos de robôs.

Em certas versões do *Windows*, notadamente as versões NT, 2000 e XP, por questões de segurança, não é possível acessar a porta paralela diretamente a partir de um programa. Para contornar esta limitação utilizou-se a biblioteca de rotinas *io.dll*, desenvolvida por Fred Bulback e fornecida gratuitamente através da Internet no site *http://www.geekhideout.com/iodll.shtml.*

- **Implementação no Controlador do Robô**: a implementação das rotinas no controlador deve ser feita através de programação na linguagem nativa do mesmo, no caso, em *ABB RAPID*. Estas

rotinas devem ser introduzidas no código fonte dos programas em que se deseja implementar a comunicação.

A comunicação implementada é bidirecional, onde o PC e o robô tem igual capacidade de envio de dados. O funcionamento, porém, dá-se no modo Cliente/Servidor, tendo o PC como Cliente. Nesta configuração o fluxo padrão de dados é do PC para o robô, a transmissão no sentido inverso só ocorre sob solicitação do Cliente.

O *Programa Cliente* desenvolvido pelo usuário recebe as informações externas, que podem ser provenientes tanto de um arquivo, de uma conexão de rede ou de outro dispositivo de entrada, e após processá-las decide quais tarefas devem ser realizadas pelo robô. O programa chama rotinas disponíveis na *Camada de Aplicação*, informando a ação a ser realizada, bem como os dados e parâmetros necessários. Estas informações são repassadas para a *Camada de Enlace de Dados*, a qual verifica os sinais da *Camada Física*, aguardando o sinal de "pronto para a recepção" do robô. Os dados são convertidos para o formato de *bytes* e enviados seqüencialmente para a *Camada Física*, este envio incluí também todo o procedimento de sincronismo de transmissão. Cada *byte* é então transferido para a porta paralela, estando disponível para o circuito externo. Os sinais da porta paralela são recebidos pelo circuito da Placa E/S do robô, através de um dispositivo de *hardware* especialmente desenvolvido para esta aplicação, denominado *Módulo IRBCom*, o qual faz os ajustes de tensão necessários.

Um diagrama simplificado do fluxo de informação no sistema é apresentado na Figura 3.1.

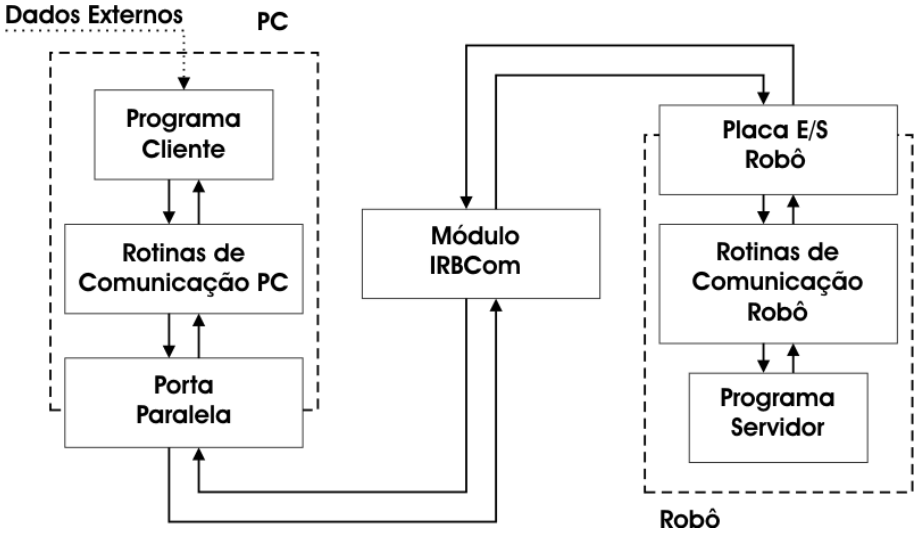


Figura 3.1 – Fluxo de informação em um sistema de controle.

No controlador do robô a *Camada de Enlace de Dados* do programa residente recebe a seqüência de *bytes* através da Placa E/S, recompondo os dados originais, os quais são repassados à *Camada de Aplicação* ou diretamente *ao Programa Servidor*, que analisa os dados recebidos, acionando os comandos necessários para executar a tarefa prevista.

A transmissão no sentido *Robô/PC* ocorre somente quando o *Programa Cliente* solicita uma informação ao *Programa Servidor*, que realiza a transmissão utilizando-se um procedimento semelhante.

Considerando-se o modelo de hierarquia de rede apresentada no capítulo 2.2 , o sistema integra-se no Nível de Máquina, ficando subordinado ao controlador principal da célula de trabalho, conforme apresentado na figura 3.2. Nesta configuração o sistema deve possuir um PC dedicado, o qual comunica-se com o controlador da célula através de uma rede local, conexão serial ou outro meio disponível. Em casos de células pequenas, o próprio PC do Sistema IRBCom pode assumir as funções de controlador principal.

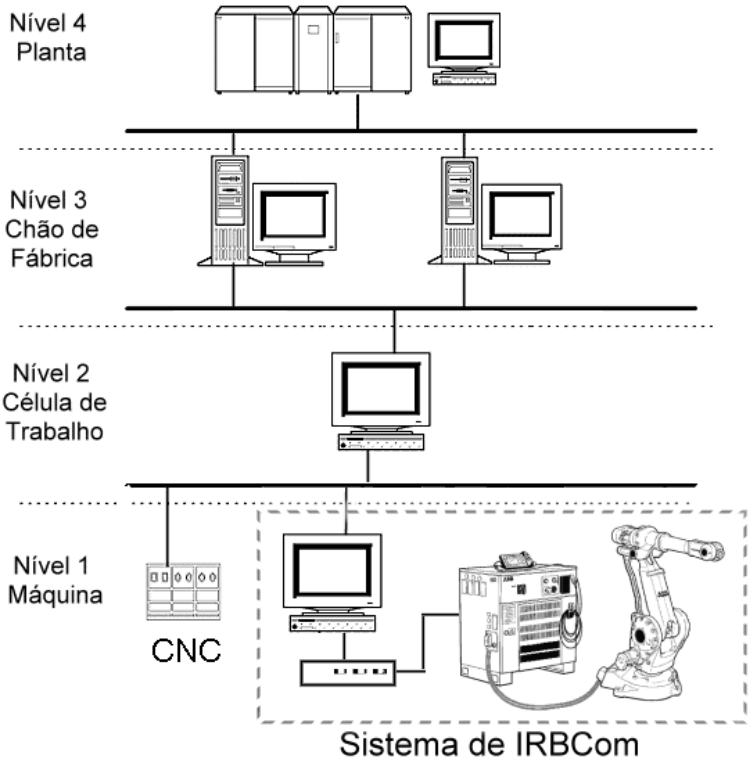


Figura 3.2 – Sistema IRBCom inserido no modelo hierárquico

Analisando-se o problema segundo o modelo *OSI*, chegou-se ao modelo proposto na figura 3.3. Este modelo estrutura-se em três camadas:

- **Física**: Interfaces de comunicação dos dispositivos, especificação de tensões e conectores;

- **Enlace de Dados**: Rotinas para formatação dos dados, controle e sincronismo da comunicação;

- **Aplicação**: Rotinas de controle e envio de dados disponíveis na interface do usuário.

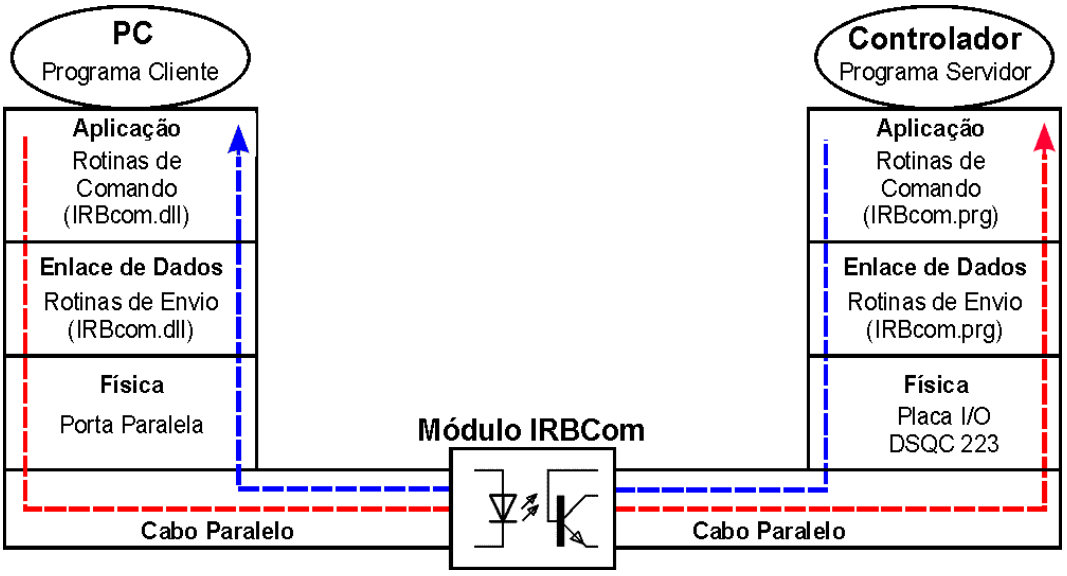


Figura 3.3 – Definição do sistema segundo o modelo de camadas *OSI*

3.2. Camadas Físicas

A *Camada Física* de um equipamento, segundo o modelo *OSI* [Groover,1987], é responsável por enviar os dados no formato de sinais elétricos na rede e receber os sinais enviados por outros dispositivos.

3.2.1 Camada Física do PC

A interface de comunicação utilizada no PC é a porta paralela (Apêndice II), em modo *SPP (Standart Parallel Port)*. Originalmente desenvolvida pela *IBM* para o uso com impressoras, esta interface é composta por um conector com 25 pinos sendo: oito sinais de dados, cinco sinais de *status*, quatro sinais de controle e oito linhas de terra.

Destas, foram utilizadas:

- 8 linhas de *Dados* para transmissão;

- 1 linha de *Controle* para o sinal de sincronismo do PC;

- 4 linhas de *Estado* para o recebimento de dados;

- 1 linha de *Estado* para o recebimento do sinal de sincronismo do robô.

As principais características dos sinais da porta paralela a serem observadas no projeto são:

- tensão de saída maior que 2,0V para o nível lógico “1”;

- tensão de saída menor que 0,8V para o nível lógico “0”;

- não possui proteção contra sobre-tensão ou sobre-corrente, podendo causar a queima do circuito integrado na placa-mãe do PC;

- linhas de entrada com resistores *pull-up*, ou seja, normalmente em nível “1”.

- hardware com tempo de varredura de 1µs;

- conector *DB25*.

A função de cada sinal da porta paralela, bem como o seu sinal correspondente na Placa E/S do controlador do robô, são indicadas na figura 3.4.

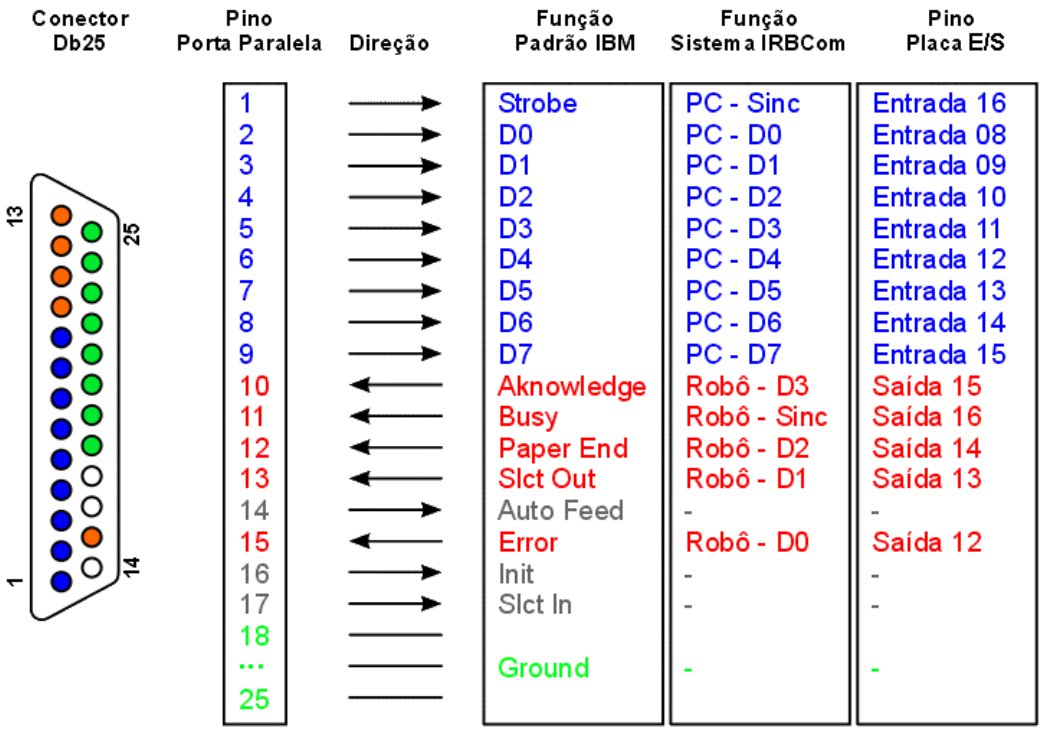


Figura 3.4 – Conexões do sistema, por função.

Para a transmissão no sentido do PC para o robô, tem-se o sinal de Controle *Strobe*, pino 1 do conector, utilizado como sinal de sincronismo do PC, sendo recebido pelo controlador do robô através do sinal de entrada 16 da Placa E/S. Os sinais dos pinos 2 a 9, são utilizados para o envio de dados, sendo conectados às entradas 8 a 15 da Placa E/S.

Para a transmissão de sinais pelo robô, o sinal de sincronismo, enviado através do sinal de saída 16 da Placa E/S, é recebido pelo PC através do sinal *Busy*, pino 11 da porta paralela. Os sinais de dados são provenientes das linhas de saída 12 a 15 são conectados respectivamente aos pinos 15, 13, 12 e 10 da porta paralela.

As conexões dos sinais apresentadas não são feitas diretamente entre as duas interfaces, pois para garantir os níveis de tensão corretos e a segurança de funcionamento os sinais são interfaceados pelo *Módulo IRBCom*.

3.2.2 Camada Física do Controlador do Robô

O controlador *ABB S4* (Apêndice II) utilizado não possui uma placa específica para comunicação de dados instalada, possuindo apenas uma Placa E/S digital de 16 canais de entrada e 16 canais de saída, modelo *ABB Digital I/O Board DSQC223*.

A função original desta placa é a recepção e envio de sinais de controle discreto para acessórios e dispositivos integrados à célula do robô, não sendo desenvolvida para comunicação de dados. Porém durante os testes iniciais deste trabalho demonstrou-se possível sua utilização para este fim, apesar de sua baixa velocidade de varredura dos sinais de entrada implicar em uma baixa taxa de transmissão.

Suas principais características físicas e operacionais são:

- tensão acima de 15V para o nível lógico “1”;
- tensão abaixo de 5V para o nível lógico “0”;
- corrente limitada internamente a 20mA por canal;
- canais isolados opticamente;
- tempo de varredura de 5ms;

- conector: borneira padrão industrial, porém foi introduzido um conector DB25 para facilitar a conexão do sistema;

- saída de tensão 24V para utilização externa.

As entradas digitais da placa são nomeadas por padrão como *di1* a *di16,* e as saídas como *do1* a *do16,* podendo-se acessar o valor de cada entrada individual diretamente no código do programa. Tanto as entradas quanto as saídas da placa podem ser configuradas para trabalhar em grupo. Neste modo tem-se a vantagem de não ser necessário ler o valor de cada entrada e realizar a conversão dos *bits* individuais em um valor numérico, pois esse procedimento é feito pelo próprio circuito da placa. Do mesmo modo não é necessária a conversão em *bits* dos sinais de saída. Neste trabalho foram configurados dois grupos:

-*gi1*: grupo formado pelos sinais de entrada *di8* até *di15*, utilizado para o recebimento de dados;

-*go1*: grupo formado pelos sinais de saída *do12* até *do15*, utilizado para o envio de dados.

Realizada a configuração inicial, para viabilizar a escrita e leitura destas portas basta acessar as variáveis de mesmo nome no código do programa.

3.2.3. Módulo IRBCom

Neste trabalho, devido à incompatibilidade dos sinais elétricos entre as duas interfaces utilizadas, foi necessário projetar um dispositivo visando ajustar os níveis de tensões e garantir a segurança de funcionamento na comunicação entre os equipamentos.

Este dispositivo é formado de uma série de acopladores óticos de alta sensibilidade modelo *4N33* (Apêndice III), os quais permitem a transmissão dos dados sem nenhuma conexão elétrica entre a porta paralela do PC e a Placa E/S do controlador do robô. Este sistema permite total segurança de operação, garantindo isolamento galvânico entre as duas interfaces para tensões da ordem de até 1,5kV.

As características da porta paralela possibilitam evitar o uso de uma fonte externa de energia, isso devido aos sinais de saída terem corrente suficiente para acionar diretamente os acopladores óticos, e também ao fato das linhas de entrada possuírem resistores *pull-up*, ou seja, resistores internos conectados ao pólo positivo da fonte de tensão, de modo a que as entradas tenham um nível “1” quando em aberto. Com isso o controle do nível da entrada pode ser feito pelo simples aterramento do

pino correspondente, o que o traz para o nível “0”. Na seção do circuito conectado ao controlador do robô é utilizada a tensão de 24V disponível na interface da própria placa.

A figura 3.5 apresenta o circuito de acoplamento dos sinais de uma linha de dados da porta paralela para os níveis necessários ao acionamento da linha correspondente na Placa E/S, incluindo o acionamento do *led* de controle do painel frontal.

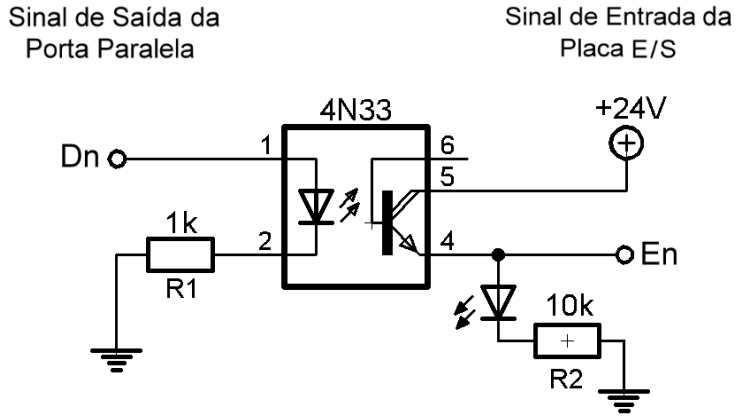


Figura 3.5 - Acoplamento óptico de um sinal entre a porta paralela e a Placa E/S

O circuito funciona de modo que quando a linha *Dn* recebe um sinal de nível lógico “1”, sua tensão eleva-se para um nível maior do que 2,0 V. Este nível de tensão excita o *LED* do acoplador ótico *4N33,* fazendo com que o seu fototransistor interno conduza, com isso elevando a tensão da linha *En* para um valor superior a 15 V. Esta tensão é então reconhecida como nível lógico “1” pela a Placa E/S. Quando a linha *Dn* recebe um sinal de nível lógico “0” a tensão mantém-se abaixo de 0,8 V, mantendo o acoplador ótico sem conduzir, e a tensão na linha *En* da Placa E/S abaixo de 5 V, o que é reconhecido como nível “0”. Os resistores do circuito têm a função de limitar as correntes a valores seguros. No *Módulo IRBCom* tem-se um circuito igual a este para cada linha de sinal enviado pelo *PC*, totalizando 9 circuitos independentes (Apêndice IV).

A figura 3.6 ilustra o circuito para o ajuste dos níveis de voltagem das linhas de dados enviados pela Placa E/S para o *PC*. Neste caso há uma inversão no nível do sinal durante a transmissão, visando aproveitar-se das características da porta paralela para simplificar o circuito. Esta inversão é corrigida nas rotinas da *Camada de Enlace de Dados*.

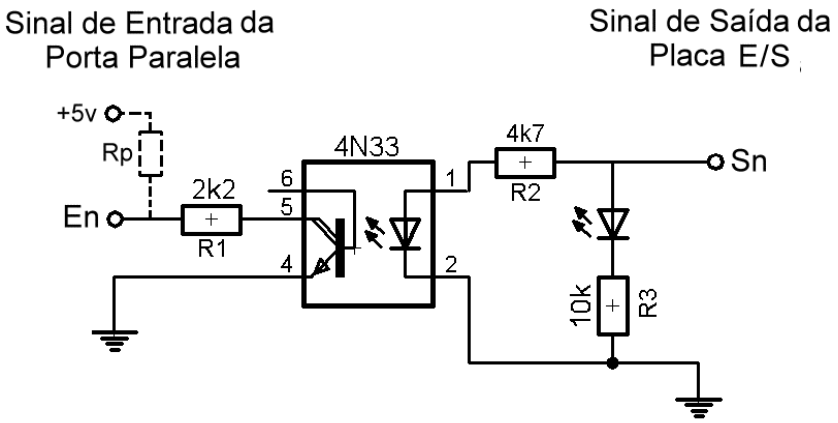


Figura 3.6 - Acoplamento óptico de um sinal entre a Placa E/S e a porta paralela

O funcionamento do circuito dá-se da seguinte maneira: quando a linha *Sn* da Placa E/S recebe um sinal de nível lógico "1" sua tensão sobe para um valor acima de 15 V. Esta tensão excita o emissor do acoplador ótico, provocando a saturação de seu fototransistor interno e com isso a redução da tensão da linha *En* para um valor inferior a 0,8 V. Tem-se então a entrada *En* sendo reconhecida como nível lógico "0". No caso da linha *Sn* receber um sinal "0", sua tensão ficará abaixo de 5 V, o acoplador ótico permanecerá em corte, e a linha *En* da porta paralela terá sua tensão elevada para mais de 2,4 V através de resistor *pull-up* interno, ou seja, ficará no nível lógico "1". O resistor *Rp* indicado no circuito é interno à porta paralela, sendo representado apenas para facilitar a compreensão do funcionamento. Os demais resistores têm a função de limitar as correntes do circuito. O *Módulo IRBCom* possui cinco circuitos iguais a este, um para cada linha de sinal enviado do controlador do robô para o PC (Apêndice IV).

Para a montagem final dos circuitos foram elaboradas placas de circuito impresso (Apêndice IV), sendo as mesmas acondicionadas em um gabinete com conectores externos.

O painel frontal do gabinete, figura 3.7, possui *leds* indicadores do nível de sinal presente em cada uma das linhas de comunicação. Estes estão agrupados por função, de modo a facilitar a monitoração do funcionamento do sistema.

Figura 3.7 – Painel frontal do módulo IRBCom

A figura 3.8 apresenta uma vista posterior do gabinete aberto, onde se visualiza a disposição dos componentes.

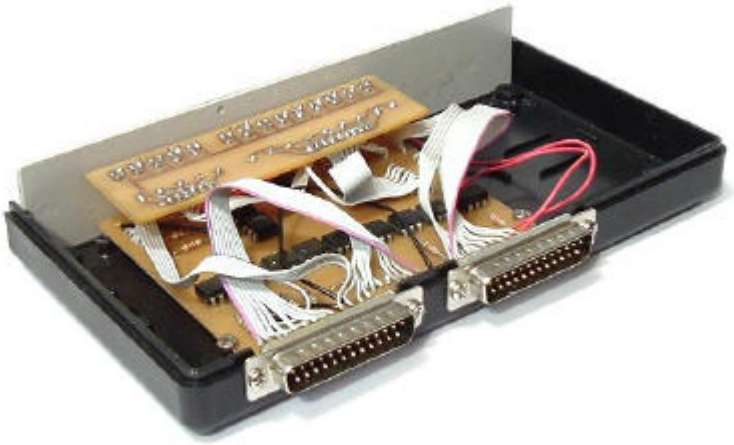


Figura 3.8 – Vista posterior do gabinete aberto

As conexões do sistema são feitas através de cabos paralelos de 25 vias com conectores DB25. No *Módulo IRBCom* utilizam-se dois conectores DB25 macho, figura 3.9, um para conexão com o robô e outro para a conexão ao PC, pois esta configuração impossibilita a conexão errônea do cabo proveniente do robô diretamente na porta paralela do PC, o que causaria danos a mesma.

Figura 3.9 – Conectores externos do *Módulo IRBCom*

3.3. Camada de Enlace de Dados

Por definição, a *Camada de Enlace de Dados* é responsável pela estruturação de dados com vista ao seu envio pela rede, cabendo também um certo nível de controle de transmissão. É responsável de igual modo pela transição do formato de *bits* provenientes da rede, para um formato que possa ser entendido por níveis superiores.

Esta camada constitui-se de rotinas pré-definidas para a transmissão e recebimento de dados, tendo implementação equivalente tanto no módulo do PC quanto no módulo do controlador do robô.

No módulo do PC estas rotinas estão encapsuladas em um arquivo chamado *IRBCom.dll*, o qual é compilado no formato *Dinamic Link Library (DLL)* para a plataforma *Win32*. Este formato de arquivo permite a sua utilização por qualquer programa desenvolvido para o sistema operacional *Microsoft Windows*.

Para o robô as rotinas foram escritas na linguagem nativa do controlador (*ABB RAPID*) e devem ser incluídas no código fonte do programa que pretende usar o sistema de comunicação.

3.3.1. Sincronismo de Transmissão

Devido à baixa velocidade de processamento do controlador do robô, que pode bloquear a comunicação durante a execução do programa, foi necessário implementar sinais de sincronismo para garantir a máxima taxa de transmissão e evitar conflitos. Com um procedimento de sincronização viabiliza-se a transmissão, permitindo que esta seja interrompida, e posteriormente retomada, por qualquer um dos dispositivos sem perda de dados.

O sincronismo da transmissão, representado na Figura 3.10, ocorre na seguinte seqüência:

- receptor ativa seu sinal de sincronismo, indicando estar livre para receber dados;
- dados são disponibilizados nas linhas de saída;
- emissor ativa seu sinal de sincronismo, indicando que os dados estão prontos para leitura;
- receptor lê os dados;
- receptor desativa o sinal de sincronismo, indicando a conclusão da leitura;
- emissor desativa o sinal de sincronismo indicam conclusão da transmissão.

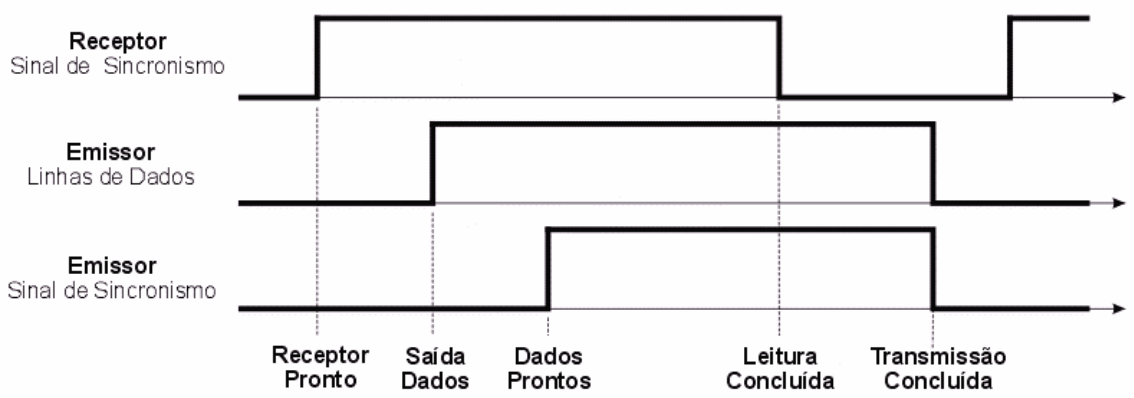


Figura 3.10 – Método de sincronismo na transmissão

3.3.2. Rotinas e tipos de dados

A implementação de rotinas dedicadas viabiliza o protocolo de comunicação para os diversos tipos de informações.

A *Camada de Enlace* de dados define três categorias de dados, a saber:

- *Byte*: número natural, transmitido no formato de 8 *bits*;

- Inteiro: número inteiro, transmitido no formato de 16 *bits*;

- Real: número real, com 16 *bits* para a parte inteira e 8 *bits* para a parte fracionária.

Para cada categoria de dado implementou-se uma rotina específica de envio, as quais estão indicadas na Tabela 3.1.

Tabela 3.1 – Rotinas da Camada de *Enlace de Dados*.

Emissor	Formato	Faixa	Precisão	Descrição	Receptor
SendByte	8 *bits*	0...255	1	Envia um *byte* de dados do PC para o controlador do robô.	***ReceiveByte***
SendInteger	16 *bits*	-32.768 ... 32.768	1	Envia o valor de um número inteiro o PC para o controlador do robô.	***ReceiveIntege r***
SendReal	24 *bits*	-32.768 ... 32.768	1 / 256 ≅ 0,0039	Envia um valor real o PC para o controlador do robô.	***ReceiveReal***

A rotina *SendByte,* em conjunto com *ReceiveByte,* é responsável pelo envio de 1 *byte* de dados do emissor para o receptor, implementando todo o procedimento de sincronismo de transmissão. As demais rotinas desenvolvidas apenas formatam os dados durante o envio ou recebimento.

Um exemplo é o envio de um número inteiro pela rotina *SendInteger*, onde o valor é enviado no formato 16 *bits*, sendo 15 *bits* de dados e 1 *bit* de sinal. A mensagem é decomposta em dois *bytes*, os quais são enviados em seqüência utilizando *SendByte* (figura 3.11). No receptor a rotina *ReceiveInteger* recebe estes dois *bytes* através de *ReceiveByte*, e então recupera o valor do número inteiro.

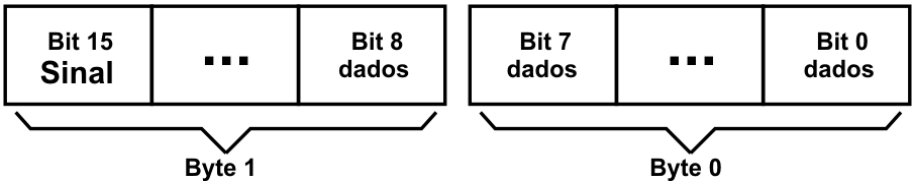


Figura 3.11 – Forma de envio de um valor 16 *bits* pela rotina *SendInteger* do PC

No caso de envio de mensagens do controlador do robô para o PC o procedimento é o mesmo, porém como existem apenas quatro linhas de dados a rotina *SendByte* envia o dado em dois pacotes de quatro *bits* (figura 3.12).

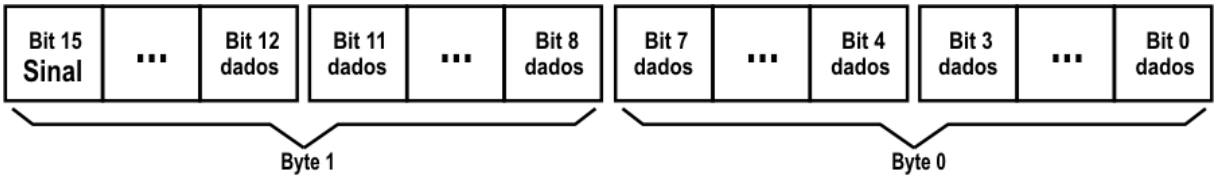


Figura 3.12 – Forma de envio de um valor 16 *bits* pela rotina *SendInteger* do Robô

A otimização na escolha dos tipos de dados a serem utilizados na transmissão das informações é um detalhe importante devido à limitada velocidade de transmissão.

Outros tipos de dados podem ser compostos por rotinas em níveis superiores, utilizando estes três tipos padrões. Por exemplo, o envio de caracteres pode ser feito através da sua posição na tabela *ASCII*, o qual é um valor de 8 *bits*, sendo o valor enviado por uma chamada a *SendByte*, e posteriormente decodificado por uma rotina de recepção. Do mesmo modo um valor real de 32 *bits* pode ser desmembrado em dois segmentos de 16 *bits* enviados por *SendInteger*, tendo-se uma rotina de recepção capaz de receber estes dois valores e recuperar o valor inicial da informação.

3.3.3. Detecção e correção de erros

Na definição do modelo OSI, a *Camada de Enlace de Dados* é responsável também pela detecção e correção de erros de transmissão. Durante os testes, porém, o sistema demonstrou ser robusto, apresentando sinais com níveis de tensão e tempos de duração elevados, gerando uma transmissão bastante estável.

Por este motivo, e também devido à baixa taxa de transmissão, optou-se por não implementar um sistema de verificação da integridade das informações durante a transmissão. Este sistema implicaria no envio de dados adicionais que reduziriam ainda mais a taxa de transmissão do sistema. Existe porém a possibilidade de implementação desta função pelo usuário, em rotinas de nível mais alto.

Para realizar uma verificação na integridade física das linhas de dados foi implementada na *Camada de Aplicação* a rotina *Check*, detalhada no capítulo 3.4.2, que realiza testes de transmissão verificando se todas as linhas de transmissão estão operacionais. Cabe ao *Programa Cliente* solicitar esta verificação quando for necessário.

Erros no sincronismo da transmissão e na execução do *Programa Servidor* ao serem detectados são tratados de modo a informar as camadas superiores sua ocorrência e qual sua natureza. As rotinas implementadas utilizam o formato de funções, ou seja, após completar a sua operação retornam um valor. Este valor de retorno indica o código de erro da função, conforme apresentados na tabela 3.2.

Tabela 3.2 – Códigos de erros das funções

Valor de retorno	**Descrição**
0	Rotina executada com sucesso.
1	Sem resposta do dispositivo.
2	Problema na execução da tarefa.

Ao ocorrer um erro na execução de um comando no *Programa Servidor*, este erro será capturado por uma função de tratamento de erros. Nesta situação o programa é interrompido e um sinal de erro de execução enviado para o PC através da ativação da linha de transmissão *Robô-D3*. Ao detectar este sinal as rotinas da Camada *de Enlace de Dados* do *PC* também serão abortadas, retornando o valor do respectivo código de erro.

A ação corretiva a ser tomada frente a cada tipo de erro varia conforme o tipo de aplicação que está sendo executada, portanto a responsabilidade de receber o código de retorno e tomar as devidas ações corretivas fica a cargo dos *Programas Cliente* e *Servidor* desenvolvidos pelo usuário.

3.4. Camada de Aplicação

A *Camada de Aplicação* é o nível mais próximo do usuário, implementando rotinas de interface entre o sistema de comunicação e o programa cliente desenvolvido pelo usuário.

Trabalhando-se com as rotinas padrão da *Camada de Enlace de Dados* é possível realizar a comunicação de forma efetiva. Porém, visando facilitar a utilização, principalmente para as tarefas mais comuns, decidiu-se acrescentar uma Camada de Aplicação, onde constam rotinas pré-configuradas e que satisfazem grande parte das necessidades básicas de um sistema de controle.

3.4.1. Protocolo de transmissão

O protocolo de transmissão de dados é orientado a *byte*, sendo formado basicamente de um *byte* de comando seguido de uma série de *bytes* de dados (figura 3.13). O número de *bytes* de dados

é variável e definido pelo número e tipo dos parâmetros necessários para cada comando. A transmissão dos dados é feita chamando-se as rotinas da camada de Enlace de Dados ou rotinas específicas desenvolvidas nos próprios programas.

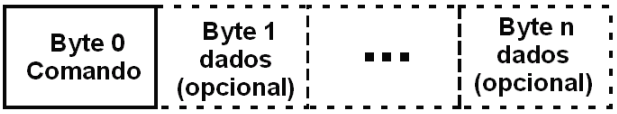


Figura 3.13 – Protocolo de transmissão

Como exemplo para o comando Move, que move a posição do efetuador do Robô para uma posição relativa a um ponto zero previamente definido, este comando admite três parâmetros, os quais são as coordenadas x, y e z, todas enviadas no formato de um número real de três *bytes*, Figura 3.14.

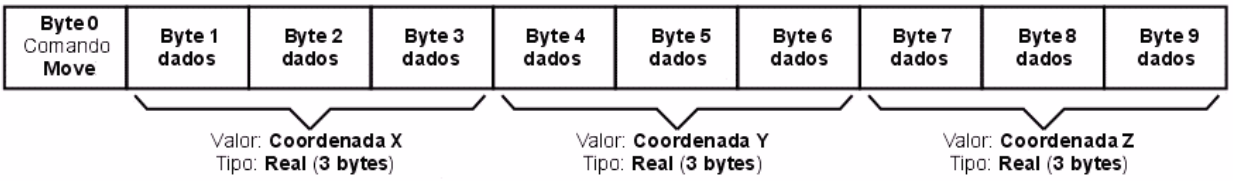


Figura 3.14. Exemplo de envio do comando *Move*

Devido às limitações quanto à velocidade de transmissão dos dados, o protocolo de comunicação é relativamente simples, não incluindo cabeçalhos ou qualquer outro dado referente ao comando ou ao conteúdo da transmissão. Com isso é necessário que cada rotina esteja programada para receber seus parâmetros em número e tipo corretos.

3.4.2. Rotinas da Camada de Aplicação

A tabela 3.3 apresenta as rotinas implementadas na *Camada de Aplicação*. Estas rotinas encontram-se compiladas no arquivo *irbcom.dll*, junto às rotinas da *Camada de Enlace de Dados*.

Tabela 3.3 – Rotinas da Camada de *Aplicação*

Emissor	Comando	Parâmetros	Descrição
DoHome	24	-	Configura a posição atual do efetuador como zero do sistema de coordenadas dos comandos *Move* enviados pelo PC

SetSpeed	23	1 Inteiro (2 *bytes)*	Configura a velocidade de movimento para os comandos *Move* enviados pelo PC.
SetZone	22	1 *byte*	Configura o raio da zona de passagem para movimentos *fly-by*.
SetDO	20	1 *byte*	Ativa a saída digital da Placa E/S do controlador.
ResetDO	21	1 *byte*	Desativa uma saída digital da Placa E/S do controlador.
Move	10	3 Reais (9 *bytes*)	Move linearmente o efetuador para a posição indicada pelos valores *x*, *y* e *z* enviados pelo *PC*.
Check	11	1 *byte*	Realiza a checagem física das linhas de transmissão.

A seguir são descritas em detalhes cada uma das rotinas implementadas na camada de aplicação:

Check: realiza a verificação das linhas de transmissão de dados. Este procedimento é feito através do envio de uma seqüência de *bytes* a partir do PC. O controlador de robô lê o valor enviado e o remete de volta ao PC, o qual confere o valor recebido, e caso seja diferente do enviado, acusa o erro de transmissão. É recomendável realizar esta verificação no início da operação do programa.

DoHome: permite a definição, pelo operador do robô, da posição de referência do sistema de coordenadas para os comandos *Move*. A seqüência de tarefas dessa rotina está assim definida:

- envia uma mensagem para o painel de controle do robô, ou *Teach Pendant*, solicitando que o efetuador seja posicionado no ponto de referência;

- libera o robô para manipulação manual pelo operador;

- recebida a confirmação do operador, retoma o controle e salva a posição atual do robô como um ponto *PRef*, que é utilizado como referência pelos comandos *Move* posteriores.

Esta rotina deve ser chamada no início do programa para permitir o posicionamento do robô dentro do ambiente de trabalho.

Move: exige três parâmetros no formato de números reais. Estes parâmetros são as coordenadas do ponto final do movimento do robô, relativos ao ponto de referência configurado pela rotina *DoHome*. O movimento é realizado através de um comando *RAPID MoveL*, o qual move a ferramenta de forma linear da posição atual até a posição final indicada. Este comando aceita ainda alguns parâmetros:

- velocidade linear: velocidade de deslocamento do efetuador, pode ser definida pela rotina *SetSpeed*;

- zona de passagem: é uma região sobre a qual a trajetória muda de direção, porém sem reduzir a velocidade de movimento. Pode ser definida pela rotina *SetZone*.

- */Conc*: permite que o controlador seja liberado para realizar outras tarefas, no caso receber novos comandos, antes que a movimentação seja completada, otimizando a velocidade de resposta do sistema. Este parâmetro é utilizado como padrão.

SetSpeed: esta rotina exige um parâmetro no formato de um número inteiro positivo. Este valor é utilizado para definir a velocidade linear de movimento, em mm/s, para os comandos *Move*. O valor inicial para a velocidade é definido como 200 mm/s, mas pode ser alterado durante o desenvolvimento do *Programa Servidor* pelo usuário final do sistema.

SetZone: exige um parâmetro no formato de um número inteiro positivo, o qual define o raio da *zona de passagem*, em milimetros, para os comandos *Move*. O valor padrão é definido como 10 mm.

SetDO: exige como parâmetro um numero inteiro entre 1 e 11 (as saídas 12 a 16 são utilizadas pelo sistema de comunicação). Coloca a condição da saída digital indicada da Placa E/S no estado alto, ou seja, ligado.

ResetDO: Recebe como parâmetro um número inteiro positivo de 1 a 11. Coloca o estado da saída digital indicada da Placa E/S no estado baixo, ou seja, desligada.

Todas as rotinas são implementadas na forma de funções, tendo como valor de retorno um código de erro da execução, conforme apresentado no capítulo 3.3.3.

3.5. O Programa Cliente rodando no PC

O *Programa Cliente* deve ser desenvolvido pelo usuário final, sendo de responsabilidade deste a lógica do sistema e a tomada de decisões, em cada aplicação específica. A programação pode ser feita em qualquer linguagem que possua um compilador para a plataforma *Win32* e permita acessar rotinas externas contidas em arquivos *DLL*. Atualmente a grande maioria das linguagens de programação, tais como *C++, Pascal, Delphi e Visual Basic*, possuem essas características.

O aplicativo deve ser desenvolvido em conjunto com o *Programa Servidor* a ser executado no controlador do robô, pois estes dois programas devem compartilhar o mesmo protocolo de comandos

e seus respectivos parâmetros. Isso é especialmente importante devido ao protocolo de transmissão não implementar um cabeçalho com informações sobre o número de dados a ser transmitido.

O programa utilizará as rotinas disponíveis na *Camada de Aplicação* para realizar a comunicação com o controlador do robô, porém não fica limitado a elas, podendo implementar rotinas mais complexas ou mais específicas para um dado fim. Esta composição permite a otimizar a comunicação, reduzindo a quantidade de informação a ser transmitida. Um exemplo de uma tarefa simples como pegar um objeto em um ponto de carga e transportá-lo para um outro ponto de descarga. Transmitindo-se todos os comandos a partir do *PC*, teríamos a seguinte seqüência:

- *Move*: aproximar o atuador do objeto;
- *SetSpeed*: reduzir a velocidade do movimento
- *Move*: atingir o ponto de captura do objeto;
- *SetDO*: acionar o atuador;
- *Move*: afastar o objeto do ponto de captura;
- *SetSpeed*: retornar a velocidade de manipulação;
- *Move*: levar o objeto para as proximidades do ponto de descarga;
- *SetSpeed*: reduzir a velocidade do movimento;
- *Move*: atingir o ponto de descarga;
- *ReSetDO*: desligar o atuador, liberando o objeto;
- *Move*: afastar o atuador do ponto de captura;
- *SetSpeed*: retornar a velocidade de manipulação;

Esta seqüência exigiria a transmissão de 76 *bytes*. Porém como a seqüência de movimentos e comandos é repetitiva, pode-se otimizá-la criando duas novas rotinas a um nível superior, *PegarObjeto* e *LargarObjeto* as quais receberiam como parâmetros as coordenadas do ponto de carga e descarga, respectivamente, utilizando-se das rotinas da *Camada de Enlace de Dados* para o envio das informações. Para a identificação de cada uma destas rotinas seria atribuído um valor de comando, e a mesma definição deve ser implementada também no *Programa Servidor* do controlador do robô. A seqüência então ficaria:

- *SendByte*: envia o valor do comando “PegarObjeto”;
- *SendReal*: envia o valor da coordenada x do ponto de carga;
- *SendReal*: envia o valor da coordenada y do ponto de carga;

- *SendReal*: envia o valor da coordenada z do ponto de carga;

- *SendByte*: envia o valor do comando “LargarObjeto”;

- *SendReal*: envia o valor da coordenada x do ponto de descarga;

- *SendReal*: envia o valor da coordenada y do ponto de descarga;

- *SendReal*: envia o valor da coordenada z do ponto de descarga;

Com isso pode-se reduzir o número de *bytes* enviados para apenas 20, o que melhora consideravelmente a performance do sistema visto que o robô precisa interromper as tarefas para realizar a comunicação.

3.6. O programa Servidor rodando no Controlador do Robô

Em uma aplicação típica, o programa principal executado no controlador do robô tem como elemento central um laço que faz a leitura do comando, o identifica e chama a rotina solicitada, esta rotina então fica responsável por receber seus parâmetros e executar a tarefa para a qual foi programada. Concluída a tarefa, o programa retorna ao laço central e chama *ReceiveByte* para receber o próximo comando, caso o PC não tenha nenhum comando a ser enviado, o controlador permanece aguardando. Do mesmo modo, no intervalo de tempo em que a tarefa está sendo executada, o controlador não é capaz de receber outros comandos, nesta situação o PC verifica que o controlador está indisponível e mantém-se aguardando.

Um exemplo é o comando *Move,* que tem seu ciclo através da seguinte seqüência:

- controlador informa estar livre para receber comandos, chamando *ReceiveByte*;

- *Programa Cliente* no *PC* chama a rotina *Move* da *Camada de Aplicação*, indicando também os parâmetros necessários;

- a rotina *Move* utiliza *SendByte* para enviar o valor 10, que é o código do comando;

- o *Programa Servidor* do controlador do robô recebe o *byte* e reconhece o comando, acionando a rotina *Move,* disponível na sua *Camada de Aplicação*;

- rotina chamada solicita a leitura dos parâmetros necessários, chamando *ReceiveReal* três vezes para receber os valores de x, y e z;

- rotina realiza a tarefa utilizando os parâmetros recebidos, acionando o comando *MoveL* para mover o efetuador para a posição x, y, z;

- terminada a tarefa retorna ao laço principal, onde o controlador solicita um novo comando chamando *ReceiveByte.*

Para permitir a utilização do sistema de comunicação *IRBCom* é necessário acrescentar ao código do programa as rotinas da camada de *Enlace de Dados* e as rotinas da *Camada de Aplicaçã*o que forem necessárias. Estas rotinas estão escritas na linguagem nativa do robô e devem ser simplesmente copiadas para dentro do programa.

O próprio arquivo *IRBCom.prg* onde se encontram as rotinas do sistema já é implementado no formato de um programa servidor completo, podendo ser utilizado diretamente caso tenha-se o objetivo de utilizar somente as rotinas disponíveis na camada de *Aplicação*, ou mesmo, como base para o desenvolvimento de programas mais elaborados.

4. PROCEDIMENTO EXPERIMENTAL

Para realizar a avaliação experimental do sistema foram desenvolvidos programas específicos, os quais simulam uma aplicação real na sua execução.

O programa Traçador Gráfico simula tarefas de seguimento de trajetória, típicas de aplicações do tipo soldagem e pintura. Este tipo de tarefa exige a manipulação de uma grande quantidade de dados, respectivos aos pontos da trajetória, previamente armazenados.

Outra aplicação comum dos robôs industriais é a manipulação de objetos, tais como carga e descarga de máquinas e paletização de embalagens. Para simular este tipo de aplicação foi desenvolvido um programa Manipulador de Blocos, contando também com um módulo de Operação Remota, onde se comprova a possibilidade de controle do robô através de redes locais via protocolo *TCP/IP*.

4.1. Traçador Gráfico

Este programa simula aplicações do tipo soldagem e pintura, onde o principal objetivo é que a ferramenta acoplada ao robô siga uma trajetória predeterminada em uma velocidade estabelecida. Além disso, a necessidade de transmissão de uma grande quantidade de dados, referentes aos pontos da trajetória, torna esta aplicação ideal para analisar a confiabilidade e a velocidade da comunicação do sistema.

Os desenhos a serem traçados pelo sistema devem ser elaborados em um programa de geração de desenhos vetoriais, tais como programas de CAD e depois exportados para um arquivo no padrão *HPGL (Hewlett Packard Graphics Language).* Este padrão foi desenvolvido pela *Hewlett-Packard* como uma linguagem para impressão de desenhos vetoriais, principalmente para a utilização com *plotters*. O arquivo HPGL utiliza o formato texto, o que facilita sua manipulação, e cada linha é constituída de dois elementos: posição da pena e coordenadas do ponto. A posição da pena é indicada por dois caracteres, os quais podem ser “PU” para indicar que a caneta deve estar levantada durante o movimento, ou seja, não traçando, e “PD” indicando que a caneta deve estar em contato com o papel durante seu deslocamento. Os valores das coordenadas indicam o próximo ponto para o qual a caneta deve ser movida, sendo apresentados em milésimos de polegada.

Para definir o ponto de referência do desenho utiliza-se a rotina *DoHome* padrão da camada de *Aplicação*. Para o controle da posição da caneta foram desenvolvidas três rotinas específicas, *SobePena*, *DescePena e MoveXY*, com o objetivo de otimizar o fluxo de informações.

O programa cliente rodando no PC foi desenvolvido em *Borland Delphi*, e sua principal função é ler os dados e comandos de um arquivo, interpretá-los e então enviar as tarefas e seus parâmetros para o robô através do sistema IRBCom. Durante a execução os comandos enviados são listados na tela, juntamente com uma simulação do desenho sendo traçado (figura 4.1).

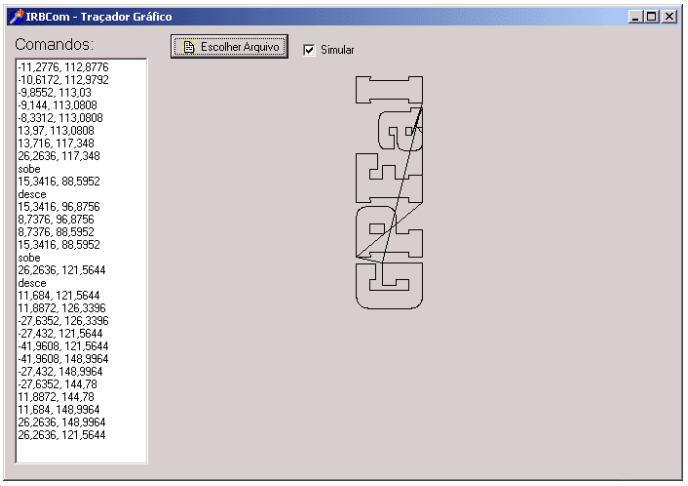


Figura 4.1 – Tela principal do programa Cliente do sistema Traçador Gráfico.

Para a realização do experimento foi acoplada uma caneta ao flange do robô, permitindo que este desenhe sobre uma folha de papel. Na figura 4.2 vemos o robô durante o traçado, e a figura 4.3 apresenta o traçado concluído.

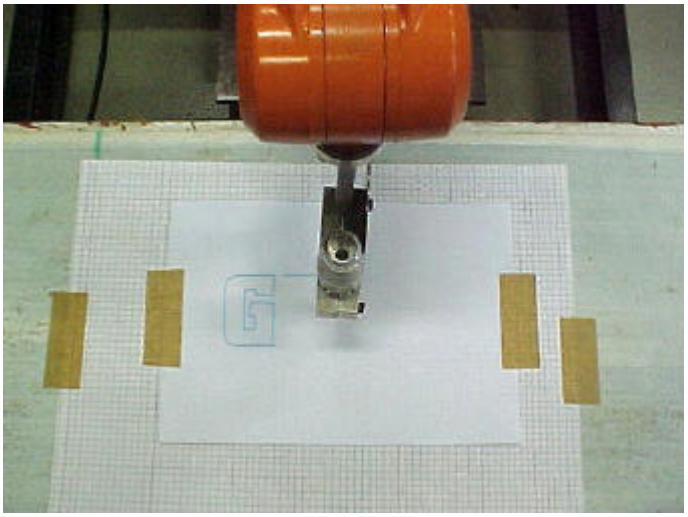


Figura 4.2 – Robô durante a operação de traçado

Figura 4.3 – Traçado concluído

Inicialmente o operador deve escolher o arquivo onde se encontram os dados do desenho a ser traçado. O primeiro passo do programa, após a indicação do arquivo, é definir o ponto de referência, o que é feito enviando um comando *DoHome*, o qual libera os controles manuais do robô para o operador, permitindo o posicionamento inicial da caneta. O próximo passo é a leitura e interpretação dos dados contidos no arquivo. As rotinas de controle da posição da pena foram implementadas diretamente no programa, sendo definido o valor 11 para o comando *DescePena* e o valor 12 para o comando *SobePena*. Basicamente a função destas rotinas é enviar ao robô o valor do comando apropriado utilizando-se a rotina *SendByte* disponível na *Camada de Enlace de Dados*.

Excluindo os comandos de subida e descida, todos os movimentos da caneta dão-se em apenas um plano, podendo portanto ser definidos por apenas duas coordenadas. Visando se aproveitar desta particularidade para otimizar a comunicação, desenvolveu-se um comando específico *MoveXY*, identificado pelo valor 15. Com esta configuração se consegue reduzir em até 30% o fluxo de dados necessário durante a execução do traçado, pois substitui-se um comando *Move* formado por 10 *bytes* por um *MoveXY* que utiliza apenas 7 *bytes*.

As coordenadas lidas do arquivo são enviadas a uma rotina *MoveXY*, onde são transformadas de milésimos de polegada para milímetros e após são enviadas como parâmetros em um comando *MoveXY* . A figura 4.4 apresenta o fluxograma do laço principal deste programa.

O *Programa Servidor* rodando no Controlador do Robô apresenta um laço onde lê um comando (*byte*) e decide a ação a ser tomada, conforme o fluxograma apresentado na Figura 4.5. Neste programa, além das rotinas padrão, foram implementadas também as rotinas correspondentes

aos comandos *SobePena*, *DescePena e MoveXY*, as quais são acionadas através dos comandos 12, 11 e 15, respectivamente. O código principal recebe os comandos, através de uma chamada a rotina *ReceiveByte*, e então os interpreta, chamando as rotinas correspondentes.

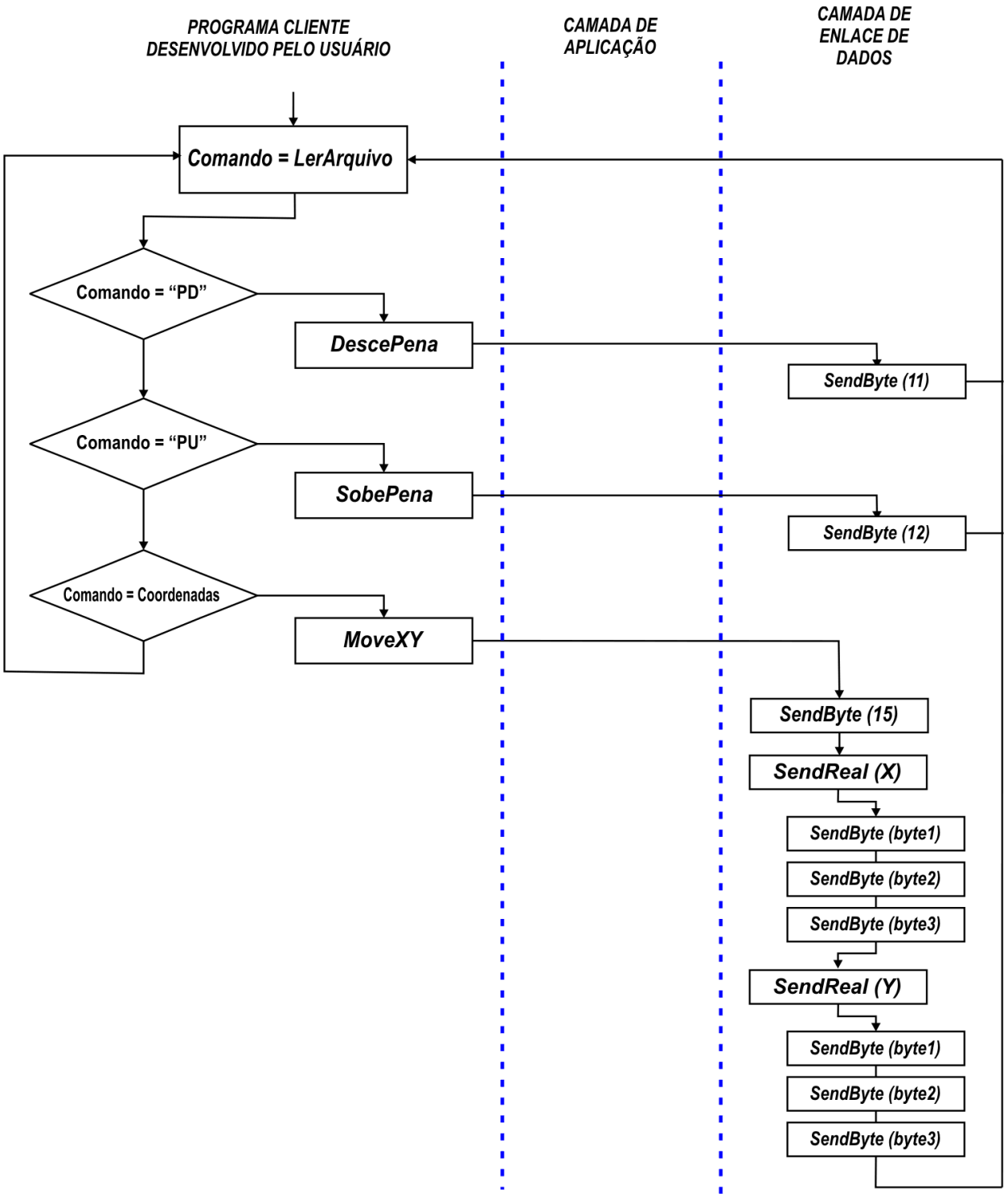


Figura 4.4 – Fluxograma do *Programa Cliente* para o sistema Traçador Gráfico

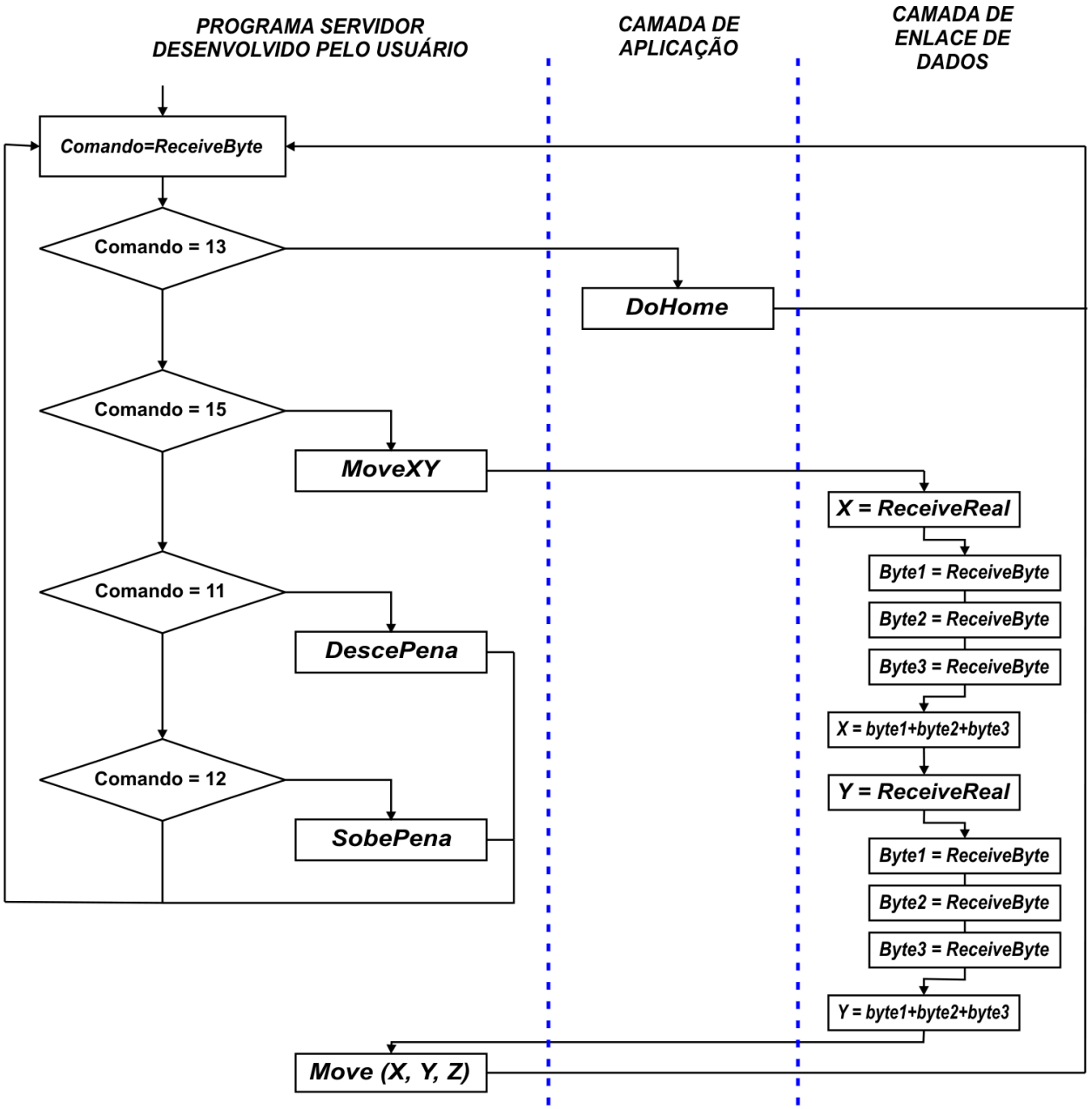


Figura 4.5 – Fluxograma do *Programa Servidor* para o Traçador Gráfico

4.2. Manipulador de Blocos com Operação Remota

Para a simulação de uma célula de manipulação de objetos em uma aplicação de carga e descarga, foi desenvolvido um sistema manipulador de blocos. A base do sistema é uma instalação didática utilizada no Laboratório de Robótica do GPFAI, onde o robô utiliza uma garra com acionamento pneumático para manipular blocos de espuma (figura 4.6). O sistema permite também a operação remota através de uma rede local *TCP/IP*.

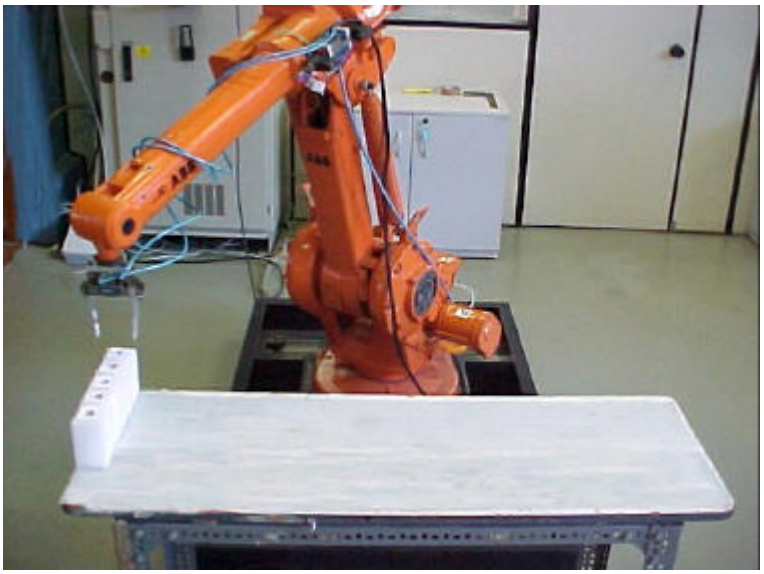


Figura 4.6 – Configuração da instalação didática utilizada.

4.2.1. Sistema de controle local

O Programa Manipulador de Blocos, apresenta uma interface gráfica na tela do PC onde se encontra uma representação da mesa com os respectivos blocos numerados (figura 4.7) e campos para ajuste das variáveis do processo, tais como altura e velocidade de transporte. Ao iniciar o programa todos os blocos estão alinhados no canto superior esquerdo da mesa (figura 4.8) e o operador deve indicar a referência do sistema, que é o ponto de carga do primeiro bloco. A partir do referenciamento, para realizar a manipulação de um bloco basta clicar com o mouse sobre sua representação na tela, e na seqüência indicar os pontos da trajetória desejada e, para iniciar a movimentação, pressionar o botão direito do mouse sobre a posição final desejada para o bloco.

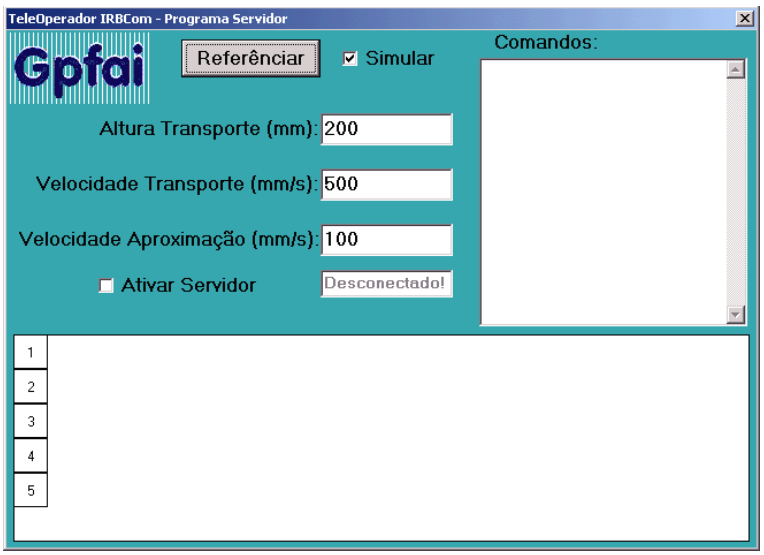


Figura 4.7 – Tela principal do programa *Cliente IRBCom* do sistema *TeleOperador*

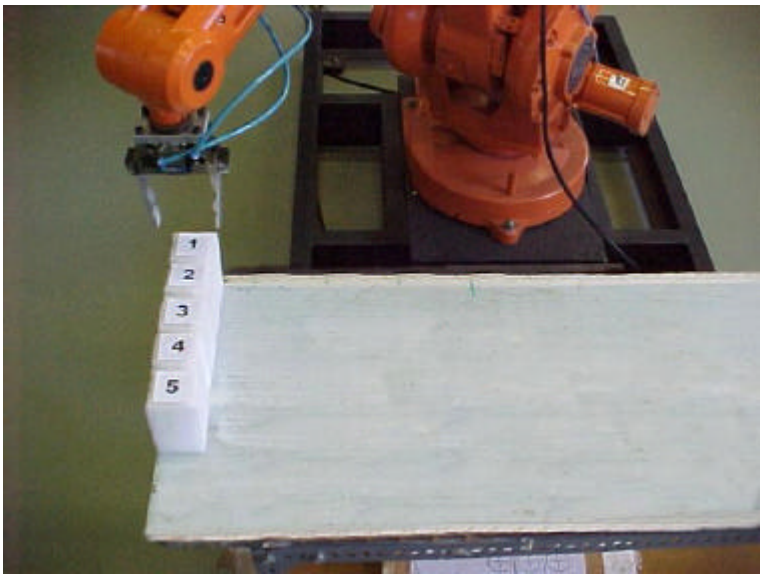


Figura 4.8 – Configuração inicial dos blocos sobre a mesa

A posição de cada bloco é armazenada na memória do programa, permitindo a localização e representação gráfica na tela do computador. O *Programa Cliente* é responsável por interpretar os dados de entrada do mouse e transformá-los em tarefas a serem executadas pelo robô. Esta interpretação consiste em identificar o bloco a ser manipulado, indicado ao se pressionar o botão do mouse, transformar as coordenadas dos pontos indicados na tela para coordenadas reais, salvá-los como uma seqüência de pontos da trajetória e enviar a seqüência de comandos para realizar a movimentação solicitada.

A figura 4.9 apresenta a tela do *Programa Cliente* durante a operação de manipulação do bloco número 5, e na figura 4.10 observa-se a configuração real dos blocos nesse mesmo instante.

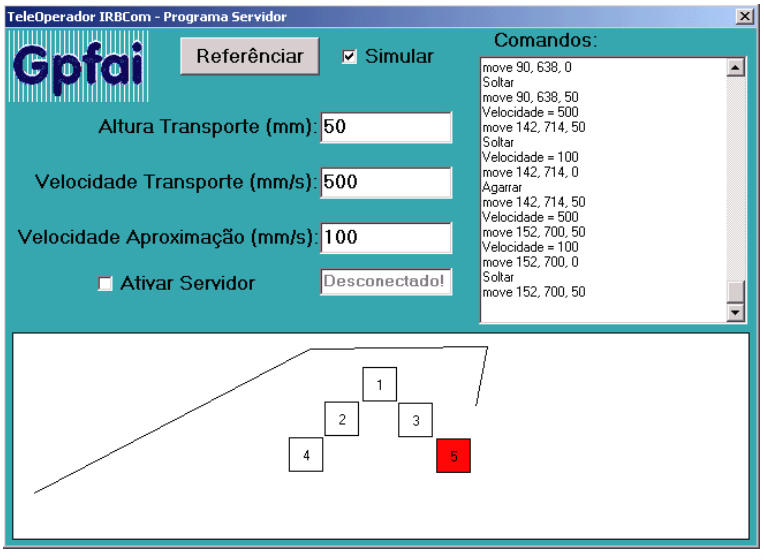


Figura 4.9 – Tela do Programa Servidor durante manipulação do bloco 5

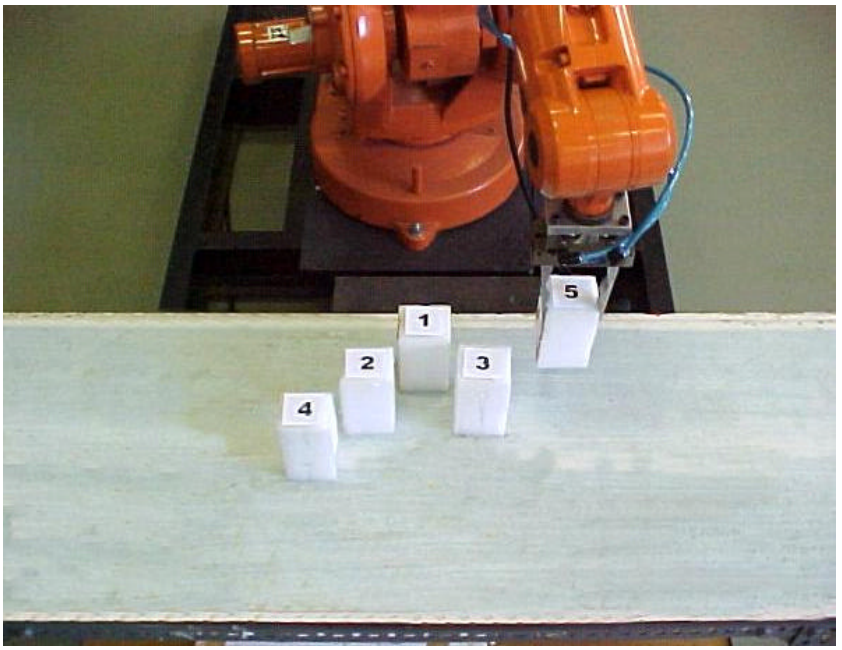


Figura 4.10 – Configuração real dos blocos durante a operação

O programa *Cliente IRBCom* acessa o sistema de comunicação através de rotinas padrão da *Camada de Aplicação*, não utilizando nenhuma rotina especial. A transmissão da seqüência de comandos não foi otimizada pois o objetivo principal é verificar a confiabilidade e velocidade de transmissão. Para a movimentação de um bloco é enviada a seguinte seqüência de comandos:

- *Move*: aproximar o atuador do bloco;
- *SetSpeed*: reduzir a velocidade do movimento
- *Move*: atingir o ponto de captura do bloco;
- *SetDO*: acionar o atuador;
- *Move*: afastar o objeto do ponto de captura;
- *SetSpeed*: retornar a velocidade de manipulação;
- Seqüência de comandos *Move*, movimentando o bloco através da trajetória indicada, até as proximidades do ponto de descarga;
- *SetSpeed*: reduzir a velocidade do movimento;
- *Move*: atingir o ponto de descarga;
- *ReSetDO*: desligar o atuador, liberando o bloco;
- *Move*: afastar o atuador do ponto de captura;
- *SetSpeed*: retornar a velocidade de manipulação;

A seqüência de comandos enviados é informada na tela principal do programa, bem como a posição atual dos blocos.

Como programa Servidor, rodando no controlador do Robô, é utilizado o próprio arquivo *IRBCom.prg*, o qual implementa as rotinas de comunicação e controle básicas do sistema.

4.2.2. Sistema de operação remota

Este sistema permite a operação do Manipulador de Blocos a partir de um segundo computador. Para isso os dois computadores deverão estar conectados a *internet*, ou a uma mesma rede local.

O programa *Cliente IRBCom* local tem capacidade de atuar também como um programa servidor no domínio de uma rede *TCP/IP* a qual o computador esteja conectado. Quando o modo *servidor* está ativado, o programa desabilita sua interface gráfica, passando a responder apenas aos comandos recebidos pela rede através de sua porta de comunicação número 2004. Estes comandos devem ser enviados em um protocolo próprio por um segundo programa *Cliente Remoto.*

O programa *Cliente Remoto* possui uma interface gráfica semelhante ao programa *Cliente IRBCom*, apresentada na Figura 4.11, tendo-se o mesmo modo de funcionamento e permitindo o mesmo tipo de operações do programa principal.

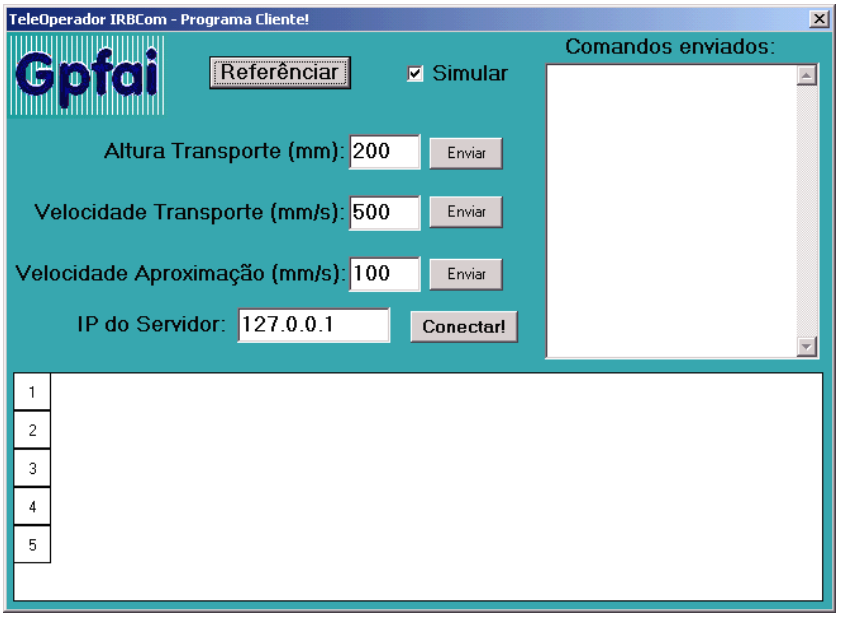


Figura 4.11 – Tela do programa *Cliente Remoto* do Manipulador de Blocos

O protocolo de comunicação é simples, os dados são enviados para o programa principal sem nenhum processamento, ou seja, são enviados apenas os dados da interface gráfica. Estes dados são

codificados em uma seqüência de caracteres que é enviada através da rede para o programa principal, este a decodifica e realiza o processamento seguindo o mesmo procedimento dos comandos recebidos da interface gráfica local.

Para permitir um retorno visual da operação do robô é utilizada uma câmera de vídeo que envia imagens do computador local para o operador remoto. Para essa tarefa é utilizado o programa *Microsoft NetMeeting*, por ser programa simples e presente em todos os sistema *Windows*.

5. Discussões

Como referência para comparação do desempenho do sistema pode-se considerar a interface de comunicação serial original do fabricante do robô. Para a implementação desta, porém, é necessária a aquisição de itens de *hardware* e *software* proprietários, os quais têm um alto custo. Esta interface apresenta a possibilidade de comunicação através de portas seriais nos padrões *RS232* ou *RS485*, permitindo taxas de transmissão de 19.200 *bps* e 38.400 *bps*, respectivamente e permite, além do controle, o envio de programas diretamente para memória do robô.

A interface desenvolvida neste trabalho é bem mais simples, apresentando como objetivo principal apenas a troca de dados de controle entre o PC e o robô, não apresentando a possibilidade de envio e ativação de programas. Isto devido principalmente ao módulo principal do sistema ser executado em nível de usuário, ao contrário do sistema original que possui o *software* rodando diretamente no sistema operacional do controlador, com acesso direto ao *hardware* dos dispositivos de E/S.

Apesar de não apresentar as mesmas funcionalidades e velocidade do sistema de comunicação proprietário da *ABB*, o sistema *IRBCom* demonstrou ser uma alternativa viável, principalmente se considerarmos seu baixo custo de implementação, e interessante para utilização no meio acadêmico por apresentar uma estrutura flexível e com o código fonte aberto. Isto permite uma fácil integração a *softwares* desenvolvidos pelos usuários, ampliando muito a flexibilidade e as possibilidades de aplicação do robô.

6. Conclusões

O sistema apresentou um bom desempenho, sendo sua taxa de transmissão limitada apenas pela velocidade de varredura da Placa E/S. As taxas médias de transmissão atingidas com a placa ABB DSQC 223 foram:

- 12,5 *bytes*/s, equivalente a 112,5 bps em uma interface serial, na transmissão do PC para o robô;

- 6,1 *bytes*/s do robô para o PC, equivalente a 61 bps em uma interface serial.

Estas taxas mostraram-se adequadas para aplicações do tipo manipulação de objetos (*pick-and-place*) com controle em tempo real pelo *PC*. O envio de trajetórias mais refinadas se mostrou possível,

porém com um baixo rendimento, devido a necessidade de interrupção do movimento para a leitura dos dados do próximo comando, impossibilitando assim um movimento suave e com altas velocidades.

Quanto à confiabilidade o sistema se mostrou bastante robusto, nas condições em que foi testado, não apresentando problemas de erros durante a transmissão. Isso devido principalmente a sua simplicidade, com níveis de tensão relativamente elevados e baixa velocidade de transmissão, justificando a não implementação de procedimentos de detecção e correção de erros.

Conclui-se portanto que foi atingido o objetivo proposto de apresentar um sistema alternativo de comunicação entre um robô industrial e um microcomputador PC padrão, mantendo-se um baixo custo de implementação, confiabilidade e flexibilidade de utilização, com velocidade de transmissão que permita o controle em tempo real.

7. Sugestões para continuidade deste trabalho

Visando aprimorar o sistema proposto são apresentadas as seguintes sugestões de continuidade do trabalho:

- Desenvolver rotinas otimizadas para aplicações específicas, tais como manipulação e pintura.

- Implementar um *Módulo IRBCom* microprocessado, permitindo a comunicação através do protocolo serial *RS232* ao invés da porta paralela.

- Implementação de rotinas de verificação da integridade das informações durante a transmissão.

- Desenvolver programas cliente que permitam a comunicação com protocolos usuais em redes de chão-de-fábrica.

- Desenvolver um sistema de programação do robô através de uma interface gráfica simplificada.

8. REFERÊNCIAS BIBLIOGRÁFICAS

ABB Flexible Automation AB, 1996, "**RAPID On-line Reference Manual**", Sweden.

ABB Robotics Products AB, 1993, "**Product Manual: IRB1400 M94A/REV1T**", Sweden.

Boucher, T.O., 1996, "**Computer Automation in Manufacturing**", Chapman & Hall, London.

Buchanan, W. J., Mahalik, N.P., 2003, **"Some Studies on CAN Specification"**, "Fieldbus Technology", cap. 23, Springer, Berlin.

Chantal, P., Mahalik, N.P., 2003, **"Industrial Ethernet Protocol Wars: Fieldbus Revisited"**, "Fieldbus Technology", cap. 2, Springer, Berlin.

Costa, Luis S.S., Caulliraux, Heitor M., 1995, "**Manufatura Integrada por Computador**", Editora Campus, Rio de Janeiro.

Derenzo, Stephen E., 1990, "**Interfacing: A Laboratory Approach Using Microcomputer for Instrumentation, Data Analysis and Control**", Prentice Hall, California.

Groover, M.P., 1986, "**Industrial Robotics. Technology, Programming and Applications**", McGraw-Hill, Singapore.

Groover, M.P., 1987, "**Automation, Production Systems, and Computer Integrated Manufacturing**", Prentice-Hall International, USA.

JIA, S., TAKASE, K., 2002, **"Internet-Based Robotic System Using CORBA as Communication Architecture"**, Journal of Intelligent and Robotic Systems 34: 121–134, Netherlands.

Kirchhoff, U., Furgac, I., 1997, **"Robot System Integration into Computer-Integrated Manufacturing"**, Robotics & Computer-Integrated Manufacturing, vol. 3, Nº1, pp 1-10, Great Britain.

Loffler, M.S., Chitrakaran, V., Dawson, D.N., 2002, **"Design and Implementation of the Robotic Platform"**, Journal of Intelligent and Robotic Systems 34: 121–134, Netherlands.

Lorini, F.J., 2002, "**Integração De Sistemas Robotizados Em Redes De Manufatura**", Anais do II Congresso Nacional de Engenharia Mecânica, João Pessoa, PB, Brasil.

Mahalik, N.P., 2003, "**Fieldbus Technology: Industrial Network Standarts for Real-Time Distributed Control**", Springer, Berlin.

Meynard, J.P., 2000, "**Control of industrial robots through high-level task programming"**, Department of Computer and Information Science, Linköpings Universitet, Sweden

Motorola Inc., 1987, **"Optoeletronics Device Data"**, Motorola Technical Information Center, USA.

Murugeshan, R., Mahalik, N.P., 2003, **"Fieldbus and Contemporary Standarts"**, "Fieldbus Technology", cap. 1, Springer, Berlin.

Raji, R. S., Mahalik, N.P., 2003, **"Control Networks and the Internet"**, "Fieldbus Technology", cap. 7, Springer, Berlin.

Spur, G., Furgat, I., Kirchhoff, U., 1987, "**Robot System Integration into Computer-Integrated Manufacturing"**, Robotics & Computer-Integrated Manufacturing, vol. 3, Nº1, pp 1-10, Great Britain.

Thompson, H. A., 2003, **"Wireless and Internet Communications Technologies for Monitoring and Control"**, Journal of Control Engineering Practice (In Press), UK.

Warp Nine Eng., "**IEEE 1284 Parallel Port Information"** http://www.fapo.com/ieee1284

Yuan, M., Mahalik, N.P., 2003, **"Foundation Fieldbus and its Interoperability"**, "Fieldbus Technology", cap. 2, Springer, Berlin.

APÊNDICE I

1. Características da porta paralela padrão

Quando a *IBM*, em 1981, introduziu o *PC*, a interface paralela foi incluída como alternativa para as lentas portas seriais utilizadas como interface para impressoras, ficando a transferência de dados pelo menos 8 vezes mais rápida que a serial.

Posteriormente a interface paralela passou a ser usada para outros fins além daquele ao qual foi projetado. Por exemplo, para acesso a *drivers* (disco rígido, flexível, removível, *CDROM's*), como interface de rede local e para interface com equipamentos de controle.

A larga utilização se deve ao fato de que, no *PC* padrão, esta é a forma mais simples, que não necessita de recursos adicionais para interconexão e apresenta uma performance adequada.

A interface paralela é composta por um conector com 17 linhas de sinal e 8 linhas de terra.

Os sinais podem ser divididos em 3 grupos:

· Linhas de *Controle* (4 vias)

· Linhas de *Estado* (5 vias)

· Linhas de *Dados* (8 vias)

A Tabela 1 fornece a definição e a descrição de cada um dos sinais no padrão *SPP* (*Standard Parallel Port*).

Tabela 1 – Sinais da porta paralela

Grupo	**Sinal SSP**	**E/S**	**Pino**	**Descrição**
Controle	*Strobe*	S	1	Ativo em baixo. Indica que os dados das vias estão válidos
	AutoFeed	S	14	Ativo em baixo. Instruía impressora para inserir avanço de linha a cada *carriage return*.
	SelectIn	S	17	Ativo em baixo. Usado para indicar seleção da impressora.
	Init	S	16	Ativo em baixo. Usado para inicializar a impressora.
Estado	*Ack*	E	10	Sinal (baixo) indicativo da correta recepção do caractere pela impressora.
	Busy	E	11	Sinal (alto) enviado pela impressora quando esta estiver ocupada.
	PE	E	12	Sem papel.
	Select	E	13	Sinal (alto) indicativo de que a impressora está pronta.
	Error	E	15	Sinal (baixo) indicativo de existência de alguma condição de

				erro.
Dados	*D0 – D7*	S	2-9	Bit 0 do dado
Terra	*GND*	-	18-25	Conectados ao terra.

Na sua concepção original as linhas de *Controle* são utilizadas para o controle da impressora. As linhas de *Estado* permitem que o *PC* possa verificar o estado da interface e também o estado da impressora. Inicialmente, as linhas de *Dados* foram concebidas de forma unidirecional do *PC* para a impressora.

Os sinais são acessados através de bits específicos em registradores da interface paralela. Mapeados em blocos contíguos de três registradores a partir do endereço base da porta paralela. Estas portas são normalmente referenciadas como *LPT* e possuem endereço base de E/S *3BCh*, *378h* e *278h*.

Novos padrões para a porta paralela referenciada no *IEEE 1284* (norma para interfaces paralelas originária do comitê *IEEE 1284* de 1994) fazem uso de 8 a 16 registradores localizados a partir de *378h* ou *278h*, ou ainda podem ser relocáveis nos casos de portas paralelas *Plug and Play*.

A tabela 2 mostra os registradores de acesso aos dados definidos para o padrão *SPP*.

Tabela 2 – Registradores para acesso aos dados da Porta Paralela

Offset	**Nome**	**Escrita / Leitura**	**Descrição**
0	Dados	E/L	Porta para escrita e leitura dos dados.
1	Estado	L	Contém o valor dos bits de estado.
2	Controle	E	Usado para estabelecer os sinais de controle.
3-7	Vários	NC	Usados em diferentes implementações.

Os modos de transferência de dados na porta paralela são definidos pelos modos referenciados no *IEEE 1284* e são limitados pela versão do *hardware* implementado. Interfaces paralelas de versões mais antigas são denominadas *SSP* e permitem modos de transferência do tipo compatibilidade, *byte* e *nibble*. Inicialmente implementadas para serem unidirecionais, estas interfaces podem ser configuradas para efetuar transferências bidirecionais dependendo do protocolo utilizado.

Implementações mais recentes da porta paralela permitem que protocolos originariamente bidirecionais, tais como *EPP*(*Enhanced Parallel Port*) e *ECP*(*Extended Capability Port*) possam ser utilizados.

2. Especificação Elétrica da Interface *IEEE 1284*

A implementação original da porta paralela não define uma especificação elétrica que garanta a compatibilidade entre equipamentos. Dispositivos e periféricos eram produzidos com diferentes configurações e padrões elétricos. Este tipo de variação de desenho tornava impossível criar um novo protocolo para a interface sem definir explicitamente os parâmetros elétricos requeridos para garantir a operação.

O padrão *IEEE 1284* definiu dois níveis de compatibilidade para a interface, *Nível I* e *Nível II*. A interface de *Nível I* é definida para produtos que não deverão operar em modos mais avançados e de alta velocidade. A interface *Nível II* é para dispositivos que operam em modos avançados, com cabos longos, e a altas taxas de transmissão, e é o mais amplamente utilizado. Estas especificações definem os parâmetros elétricos a serem encontrados nas interfaces, sendo medidos no conector.

As principais características definidas para interfaces paralelas de *Nível II* são, para o *driver*:

- A voltagem para o nível alto, em circuito aberto, não deve exceder +5,5V.

- A voltagem para o nível baixo, em circuito aberto, não deve ser menor que -0,5V.

- Em regime constante, a voltagem para o nível alto deve ser no mínimo 2,4V para uma corrente de 14mA.

- Em regime constante, a voltagem para o nível baixo deve ser menor que 0,4V para uma corrente de 14mA.

- A taxa de queda da voltagem deve estar entre 0,05 e 0,40 V/ns

Para o receptor os requisitos são:

- O receptor deve suportar picos de voltagem transientes entre -2,0V e +7,0V sem sofrer danos ou mal funcionamento.

- O valor de referência para o nível alto não deve ser maior do que 2,0V

- O valor de referência para o nível baixo não deve ser no mínimo 0,8V

- Deve haver uma histerese de entrada de no mínimo 0,2V e no máximo 1,2V.

- A capacitância do circuito não deve exceder 50pF.

APÊNDICE II

1. O sistema de computação do controlador *ABB S4*

O sistema de computação controlador *ABB S4* utilizado pelo robô *IRB 1400* é constituído de três computadores, com duas placas de circuito. Os computadores consistem de:

- Placa do computador principal- contém o computador principal do robô e controla do robô por inteiro, utilizando um microprocessador Motorola 68040.

- Placa do computador do robô - contém um computador de E/S que atua como um link entre o computador principal, os equipamentos periféricos e o computador dos eixos o qual regula a velocidade dos eixos do robô.

Os computadores são o centro de processamento de dados do robô. Eles possuem todas as funções necessárias para criar, executar e armazenar um programa do robô. Eles também possuem funções para coordenar e regular os movimentos dos eixos. A figura 1 mostra como o sistema de computação se comunica com outras unidades.

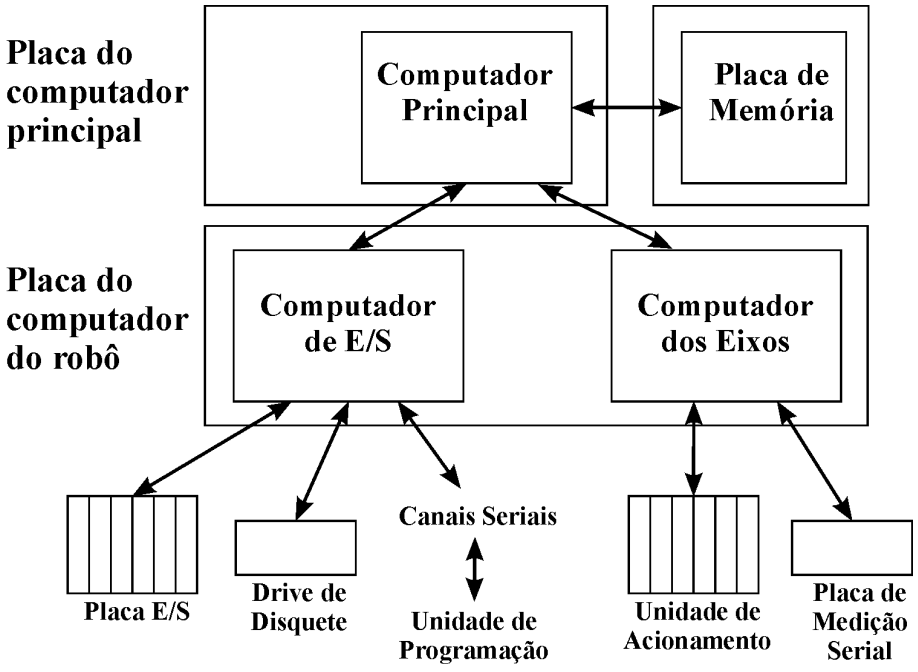


Figura 1 - Diagrama do sistema de computação do controlador ABB S4

2. Características da Placa E/S DSQC 223

As tabelas 1 e 2 apresentam as especificações técnicas das entradas e saídas digitais da Placa E/S *ABB DSQC 223*.

Tabela 1 – Especificação técnica das entradas da Placa E/S Digital *ABB DSQC 223*

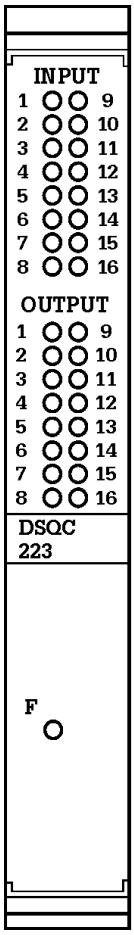
Isolação ótica	Sim
Faixa de alimentação de tensão	19-35V (Nominal 24V)
Nível lógico "1"	15-35V
Nível lógico "0"	0-5V
Consumo de corrente de entrada:	5,5mA
Máxima tensão admitida	500V
Atraso de tempo na Placa E/S:	
Hardware	=8ms
Software entrada simples	1-15ms
Software várias entradas	1-20ms
Atraso de tempo na placa do sistema	
Hardware	=8ms
Software entrada simples	<5ms
Software várias entradas	<15ms

Tabela 2 – Especificação técnica das saídas da Placa E/S Digital ABB DSQC 223

Isolação ótica	Sim
Proteção contra curto-circuito	Sim
Faixa de alimentação de tensão	19-35V (Nominal 24V)
Mínima tensão de saída	2V
Consumo máximo por saída	200mA
Consumo máx. grupo de 8 saídas	1 A
Máxima tensão admitida	500V
Atraso de tempo, caso normal:	
Hardware	=0,15ms

Software entrada simples	<2ms
Software várias entradas	2-10ms

A figura 2 mostra o painel frontal da placa ABB DSQC 223 onde se pode ver os *leds* indicadores de estado da placa, os quais são divididos em três grupos.



Descrição	**Cor**	**Descrição**
INPUT	**Amarelo**	Acende quando um sinal alto vem da entrada Quanto maior a tensão da entrada maior será a intensidade de brilho do LED. Isto quer dizer que se a tensão estiver um pouco abaixo do nível "1" , o LED terá pouco brilho.
OUPUT	**Amarelo**	Acende quando um sinal alto vem da saída. Quanto maior a tensão da saída maior será a intensidade de brilho do LED.
F	**Vermelho**	Apaga quando a placa aprova a inicialização.

Figura 2 – Painel frontal da Placa E/S *ABB DSQC 223*

APÊNDICE III

1. Acopladores óticos

Acopladores óticos são dispositivos que possuem pelo menos um fotoemissor, que é oticamente acoplado a um fotodetector através de um meio isolante, conforme o diagrama de blocos apresentado na figura 1. Esta configuração permite a passagem da informação de um circuito, onde se encontra o emissor, para outro contendo o detector sem nenhuma conexão elétrica entre os mesmos.

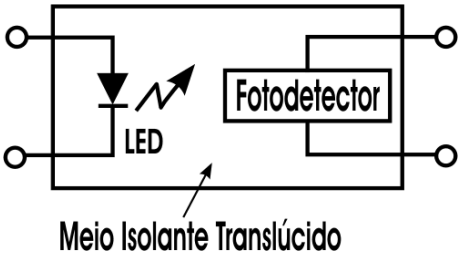


Figura 1 – Diagrama de um acoplador ótico

Devido a informação ser transferida oticamente através de um espaço isolante, esta comunicação é unidirecional, ou seja, o detector não afeta o circuito de entrada. Isso é importante pois o emissor pode ser acionado por sinais de baixa voltagem, como os provenientes de circuitos digitais, enquanto o fotodetector de saída pode acionar diretamente circuitos de alta tensão ou mesmo cargas em corrente alternada. A isolação ótica evita a interação ou mesmo danos causados por problemas nos circuitos de saída.

Estes dispositivos apresentam ainda várias vantagens sobre outros tipos de acopladores: alta velocidade de comutação, nenhuma parte mecânica, baixo consumo e isolamento total.

O encapsulamento mais comum dos acopladores óticos é *DIP* (*Dual Inline Package*) com seis pinos. Nesta configuração geralmente os pinos 1 e 2 são conectados ao emissor, enquanto os pinos 4, 5 e 6 são conectados ao detector.

Várias geometrias para a cavidade interna entre o emissor e o conector são usadas. Estas incluem bases opostas, co-planar, túnel de luz e *sandwich*. As duas primeiras são reconhecidas como os melhores métodos para se obter os altos níveis de isolação exigidos atualmente. Dispositivos em

configuração DIP seis pinos com estas geometrias são disponíveis para tensões de isolação de até 7.500V.

Atualmente dispõe-se de uma grande variedade de acopladores óticos, os quais diferenciam-se principalmente pelo tipo de foto-detector utilizado. Destes os mais comuns utilizam fototransistores, configuração *Darlington* para alta sensibilidade e fotoSCR ou fotoTRIACS para altas potências. A figura 2 apresenta os diagramas destes componentes.

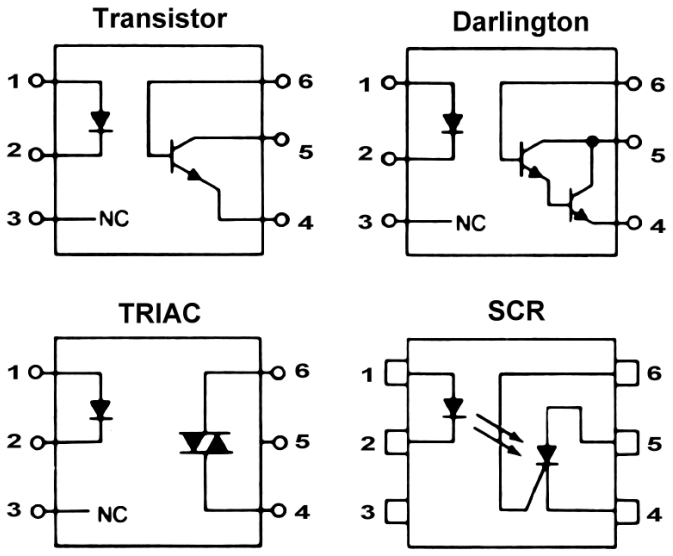


Figura 2 - Configurações comuns dos acopladores óticos

As duas características operacionais mais importantes dos acopladores óticos são tensão de isolação e eficiência. A eficiência define o nível de corrente de entrada necessária para se obter uma nível desejado de corrente na saída. Para fototransistores a eficiência é expressa por um valor denominado Razão de Transferência de Corrente (*CTR*), que é simplesmente a corrente de saída do detector dividida pela corrente de entrada do emissor.

2. Características dos acopladores óticos modelo *4N33*

As principais características operacionais e elétricas dos acopladores óticos modelo *4N33* são apresentadas na tabela 1.

Tabela 1 – Características operacionais do modelo *4N33*

Característica	**Símbolo**	**Valor**	**Unidade**
Tensão de Isolação	V_{ISO}	1.500	V
Razão de Transferência de Corrente	CTR	500	-
Emissor			
Máxima Corrente – Contínua	I_F	60	mA
Máxima Tensão Reversa	V_R	3	V
Dissipação @ T_A = 25º C	P_D	120	mW
Detector			
Máxima Corrente – Contínua	I_C	150	mA
Máxima Tensão Reversa	BV_{ECO}	5	V
Máxima Tensão Direta	BV_{CEO}	30	V
Dissipação @ T_A = 25º C	P_D	150	mW

APÊNDICE IV

Declaração das funções de comunicação da Camada de Aplicação

Todos os arquivos, tanto compilados quanto o código fonte, encontram-se em um CD anexo a este trabalho.

1 - Sintaxe das funções declaradas em *Pascal*:

Todas as funções utilizam o prefixo IRB_ para evitar conflitos no código fonte.

Código fonte das declarações:

function **IRB_ReadCommand** (var CommandIn: byte): integer; export; stdcall;

function **IRB_SendCommand** (CommandOut: byte): integer; export; stdcall;

procedure **IRB_SendInteger** (number: integer); export; stdcall;

procedure **IRB_SendReal** (number: real); export; stdcall;

Procedure **IRB_Move** (posX, posY, posZ: real); export; stdcall;

Procedure **IRB_SetSpeed** (speed: integer);export; stdcall;

Procedure **IRB_DoHome**;export; stdcall;

Procedure **IRB_Check**;export; stdcall;

Procedure **IRB_SetDO** (signal: integer);export; stdcall;

Procedure **IRB_ResetDO** (signal: integer);export; stdcall;

2 – Sintaxe para utilização das funções em programas escritos em *Pascal*:

Para a utilização das funções de comunicação em programas escritos em *Pascal* ou *Delphi*, basta incluir o arquivo *irbcom.dll* na pasta do arquivo executável e acrescentar a declaração das funções como *external*, conforme sintaxe abaixo:

function **IRB_ReadCommand** (var CommandIn: byte): integer; stdcall; external 'irbcom.dll';

function **IRB_SendCommand** (CommandOut: byte): integer; stdcall; external 'irbcom.dll';

procedure **IRB_SendInteger** (number: integer); stdcall; external 'irbcom.dll';

procedure **IRB_SendReal** (number: real); stdcall; external 'irbcom.dll';

Procedure **IRB_Move** (posX, posY, posZ: real); stdcall; external 'irbcom.dll';

Procedure **IRB_SetSpeed** (speed: integer); stdcall; external 'irbcom.dll';

Procedure **IRB_DoHome**; stdcall; external 'irbcom.dll';

Procedure **IRB_Check**; stdcall; external 'irbcom.dll';

Procedure **IRB_SetDO** (signal: integer); stdcall; external 'irbcom.dll';

Procedure **IRB_ResetDO** (signal: integer); stdcall; external 'irbcom.dll';

2 – Sintaxe para utilização das funções em programas escritos em *Visual Basic*:

Para a utilização das funções de comunicação em programas escritos em *Visual Basici*, basta incluir o arquivo *irbcom.dll* na pasta do arquivo executável e acrescentar a declaração das funções conforme sintaxe abaixo:

Private Declare Function **IRB_ReadCommand** Lib "irbcom.dll" (ByVal CommandIn As Byte) As Integer

Private Declare Function **IRB_SendCommand** Lib "irbcom.dll" (ByVal CommandOut As Byte) As Integer

Private Declare Function **IRB_SendInteger** Lib "irbcom.dll" (ByVal number As Integer)

Private Declare Function **IRB_SendReal** Lib "irbcom.dll" (ByVal number As Double)

Private Declare Function **IRB_Move** Lib "irbcom.dll" (ByVal posX As Double, ByVal posY As Double, ByVal posZ As Double)

Private Declare Function **IRB_SetSpeed** Lib "irbcom.dll" (ByVal speed As Integer)

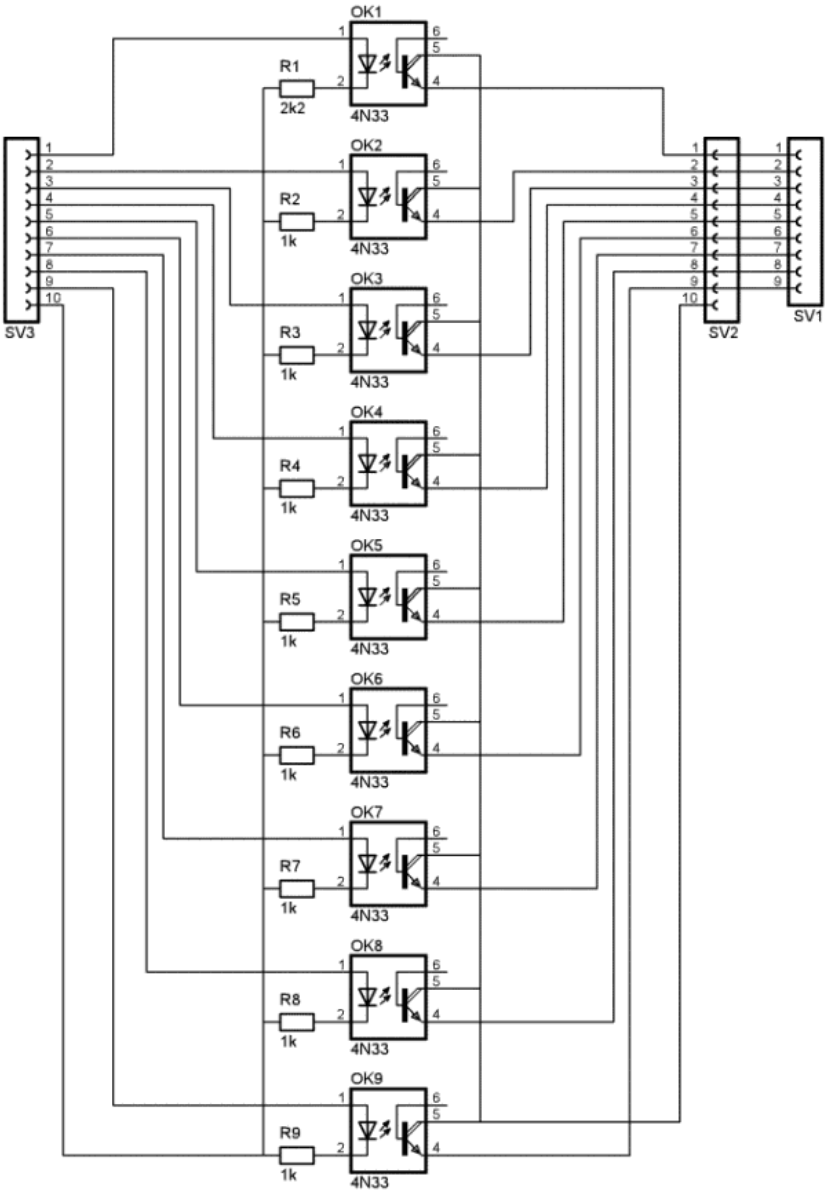
Private Declare Function **IRB_DoHome** Lib "irbcom.dll"

Private Declare Function **IRB_SetDO** Lib "irbcom.dll" (ByVal signal As Integer)

Private Declare Function **IRB_ResetDO** Lib "irbcom.dll" (ByVal signal As Integer)

APÊNDICE V

Esquemas elétricos, leiautes das placas de circuito impresso e de montagem do módulo IRBCom.



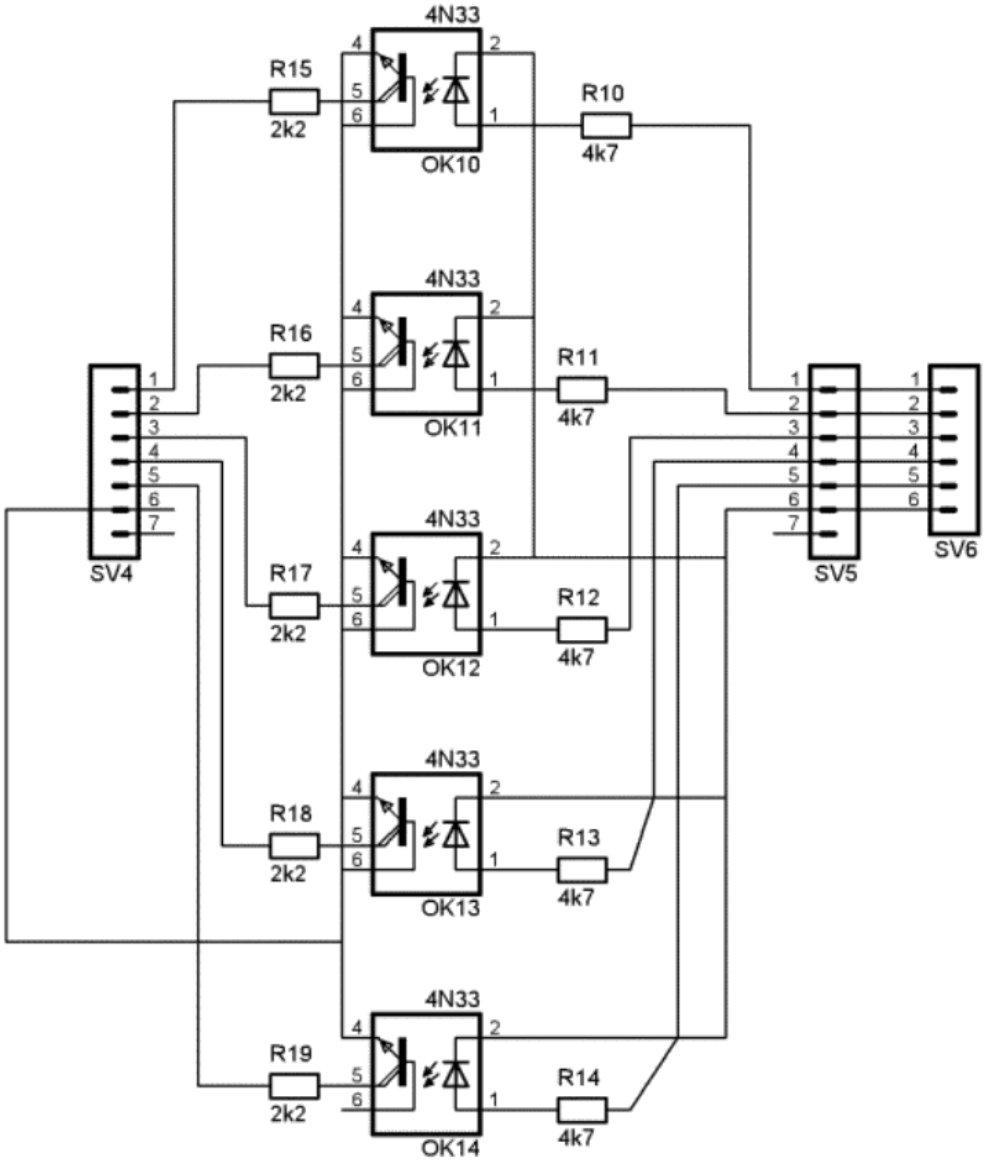
UFRGS - Engenharia Mecânica

Módulo IRBCom - Esquema elétrico da interface PC / Robô

GPFAI - Grupo de Projeto, Fabricação e Automação Industrial

Nome: **Fernando M. Bayer** Data: **Janeiro de 2004** Visto:

1



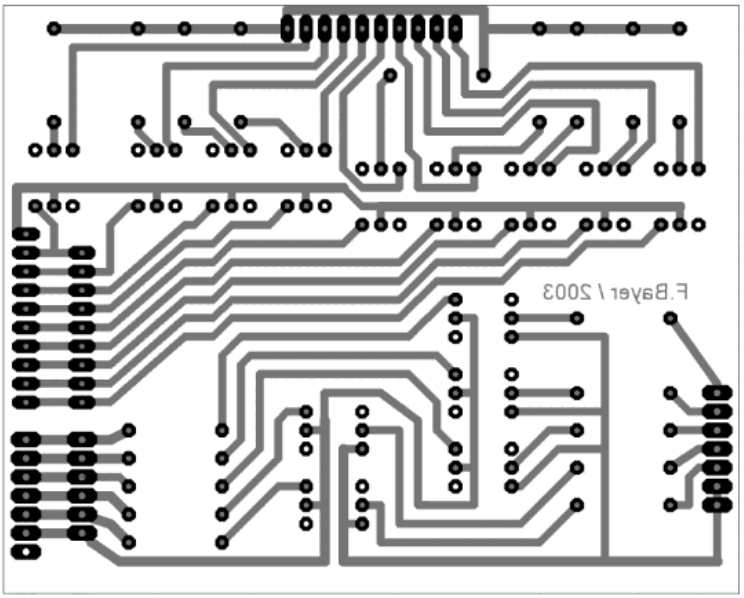
UFRGS - Engenharia Mecânica

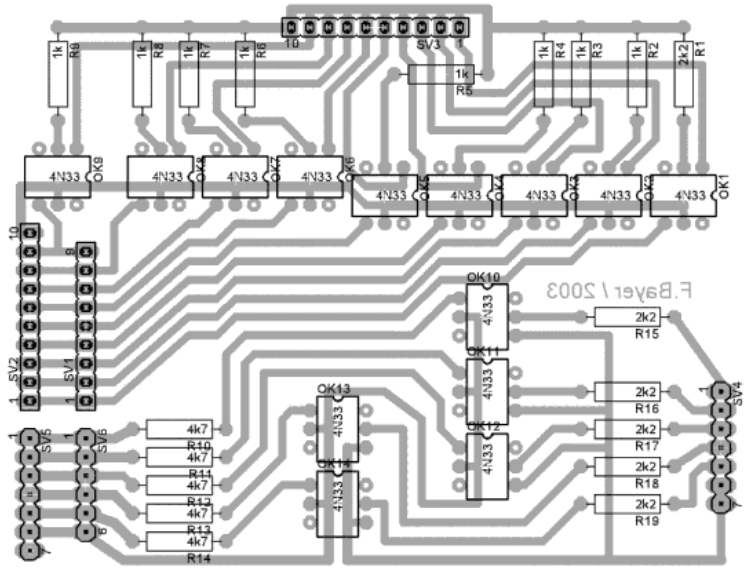
Módulo IRBCom - Esquema elétrico da interface Robô / PC

GPFAI - Grupo de Projeto, Fabricação e Automação Industrial

Nome: **Fernando M. Bayer** Data: **Janeiro de 2004** Visto:

2





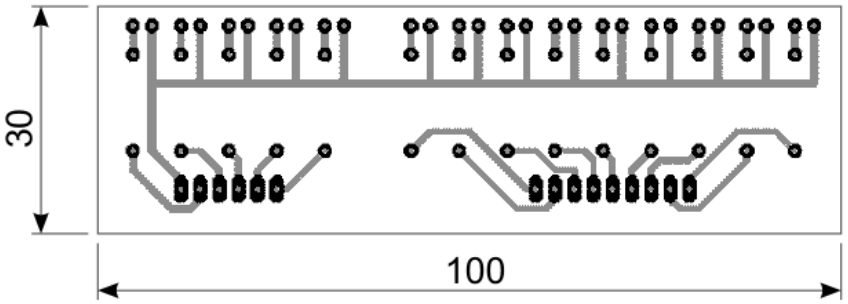
UFRGS - Engenharia Mecânica

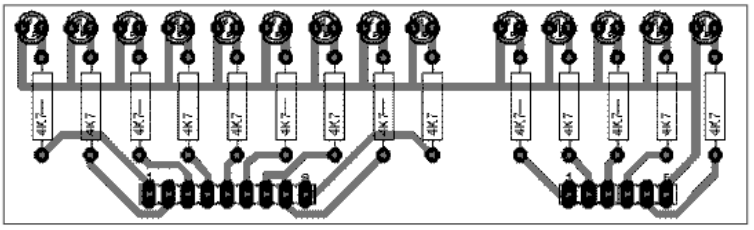
Módulo IRBCom - Placa de Circuito Impresso Principal

GPFAI - Grupo de Projeto, Fabricação e Automação Industrial

Nome: **Fernando M. Bayer** Data: **Janeiro de 2004** Visto:

3



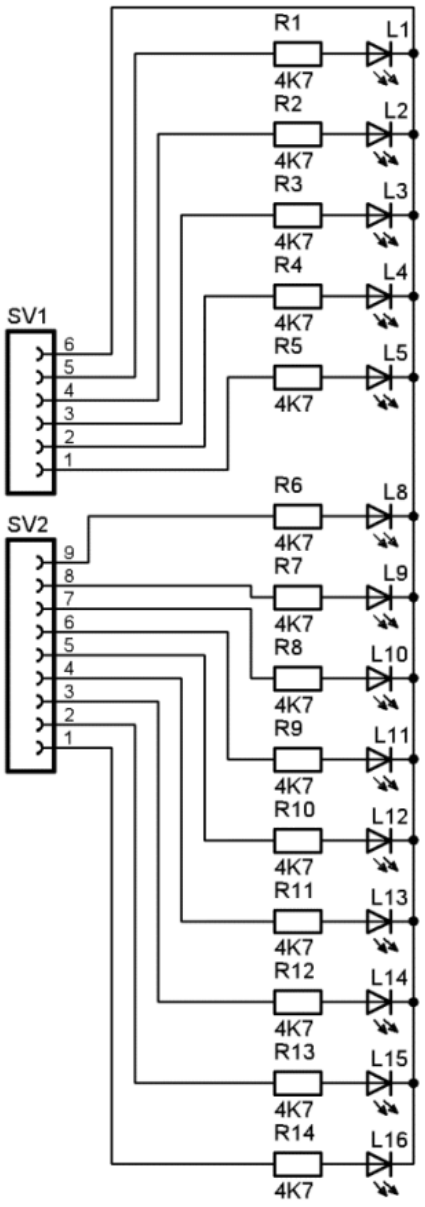


UFRGS - Engenharia Mecânica

Módulo IRBCom - Placa de Circuito Impresso Painel

GPFAI - Grupo de Projeto, Fabricação e Automação Industrial

Nome: **Fernando M. Bayer** Data: **Janeiro de 2004** Visto:	**4**
Escala: 1:1 Unidades: **mm**	



UFRGS - Engenharia Mecânica

Módulo IRBCom - Esquema Elétrico do Painel

GPFAI - Grupo de Projeto, Fabricação e Automação Industrial

Nome: **Fernando M. Bayer** Data: **Janeiro de 2004** Visto:

Escala: NA Unidades: NA

5

APÊNDICE VI

Disco em formato CD-ROM contendo os arquivos fonte dos programas desenvolvidos.

O conteúdo deste disco pode ser obtido no endereço:

http://www.xt600.com.br/irbcom/cd_irbcom.zip