INTERFACE DE COMUNICAÇÃO PARALELA DE BAIXO CUSTO PARA ROBÔ INDUSTRIAL

Fernando M. Bayer
Universidade Federal do Rio Grande do Sul – Departamento de Engenharia Mecânica
E-mail: fernandobayer@yahoo.com.br

Flávio J. Lorini
Universidade Federal do Rio Grande do Sul – Departamento de Engenharia Mecânica
Rua Sarmento Leite 425, 90050-170, Porto Alegre, RS, Brasil.
E-mail: lorini@ufrgs.br

Resumo. *Este trabalho apresenta o desenvolvimento de um sistema para comunicação bidirecional entre um robô industrial, originalmente desprovido de uma interface de comunicação integrada, e um microcomputador PC padrão. Os objetivos propostos são permitir a integração à redes de chão-de-fábrica, manter um baixo custo de implementação, confiabilidade, flexibilidade de utilização e uma velocidade de transmissão que permita o controle em tempo real. A interface desenvolvida utiliza-se da porta paralela do PC e da Placa I/O digital do controlador do robô. O método de sincronismo da transmissão e o protocolo de comunicação foram desenvolvidos de forma a permitir um sistema robusto e simples. Serão apresentadas a descrição do sistema, seus componentes principais, protocolos de comunicação e resultados experimentais.*

Palavras-Chave: *Robótica, Redes de Chão-de-fábrica, Integração de Máquinas.*

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 (Groover, 1987).

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. No cenário atual as linhas de produção tendem a ser altamente automatizadas e integradas em redes de chão-de-fábrica, gerando produtos diversificados e de grande complexidade (Kirchhoff, 1997).

Neste contexto, uma máquina que não permite a comunicação com sistemas automatizados de controle da produção perde grande parte de sua potencialidade 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, tem tornado seus controladores obsoletos em tempos menores devido à evolução do hardware e a 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 para uma mesma instalação.

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

Este trabalho visa oferecer um sistema alternativo de comunicação entre um robô industrial, no caso um *ABB IRB1400* equipado com um controlador *ABB S4*, e um microcomputador *PC* padrão. 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.

2. ANÁLISE E ESPECIFICAÇÃO DO SISTEMA

Analisando-se o problema segundo o modelo de referência *OSI* (Mahalik, 2003), a concepção para o sistema foi estruturada em três camadas, conforme ilustrado na Fig. (1).

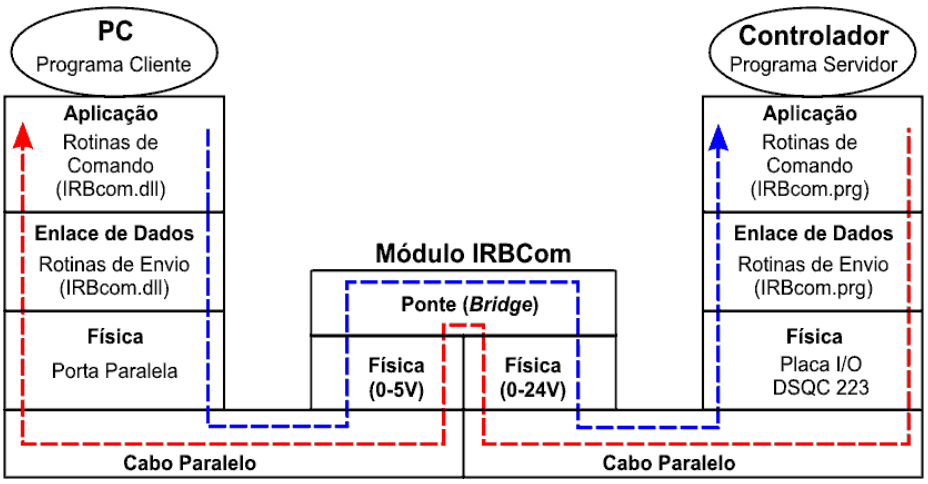

Figura 1. Definição do sistema segundo o modelo de referência *OSI*

- **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.

O funcionamento do sistema dá-se no modo Cliente/Servidor, tendo o controlador do robô como servidor. Nesta configuração o *PC* atua inicialmente como emissor, só recebendo informações quando as solicitadas ao controlador do robô.

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 I/O* do robô, através do *Módulo IRBCom*, o qual faz os ajustes de tensão necessários.

No controlador do robô a *Camada de Enlace de Dados* do programa residente recebe a seqüência de bytes através da *Placa I/O*, 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.

2.1 Camada Física do PC

A interface de comunicação utilizada no *PC* é a porta paralela, 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 que destes foram utilizadas oito linhas para transmissão de dados, duas linhas para os sinais de sincronismo e quatro linhas para o recebimento de dados (Derenzo, 1990).

Os sinais disponíveis nesta interface apresentam níveis compatíveis com *TTL*, ou seja, tensões menores do que 0,8V para o nível baixo e maiores do que 2,8V para o nível alto.

A função de cada sinal da porta paralela, bem como o seu correspondente na *Placa I/O* do controlador do robô, é apresentada na Fig. (2).

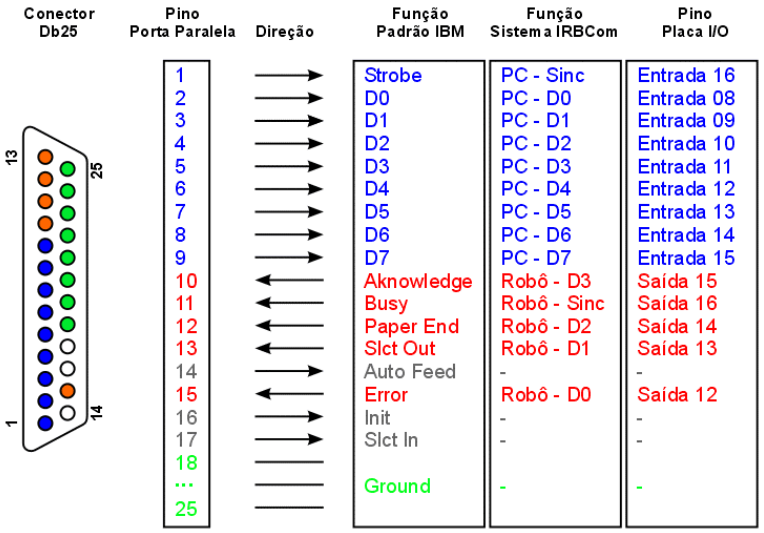

Figura 2. Diagrama de conexões e suas funções.

O sinal de controle *Strobe*, pino 1 do conector, é utilizado como sinal de sincronismo do *PC*, sendo recebido pelo controlador do robô através do sinal de entrada 16 da *Placa I/O*. Os sinais de dados, pinos 2 a 9, são utilizados para o envio dos dados, sendo conectados as entradas 8 a 15 do controlador.

O sinal de sincronismo do robô, enviado através do sinal de saída 16 da *Placa I/O*, 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.

2.2 Camada Física do Controlador do Robô

O controlador *ABB S4* utilizado originalmente não dispõe de uma interface comunicação de dados com *PC*, dispondo apenas uma *Placa I/O* digital de 16 canais de entrada e 16 canais de saída, destes canais são utilizados nove canais de entrada para recepção de informações do PC e cinco de saída para a transmissão (*ABB*, 1993). A função de cada canal no sistema é ilustrada na Fig. (2).

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.

Os níveis de sinais típicos desta interface são tensões inferiores a 5V para o nível baixo e tensões maiores de 15 V, tipicamente 24 V, para o nível alto.

2.3. Módulo IRBCom

Para o sistema desenvolvido, devido à incompatibilidade dos sinais elétricos entre as duas interfaces utilizadas, foi necessário projetar um dispositivo visando ajustar os níveis de tensão e garantir a segurança de funcionamento.

Este dispositivo é formado de uma série de opto-acopladores *Darlington* modelo *4N33* (Motorola, 1987) os quais permitem a transmissão dos dados sem nenhuma conexão elétrica entre a porta paralela do PC e a *Placa I/O* do controlador do robô. Este sistema permite total segurança de operação, garantindo isolamento galvânico entre as duas interfaces para tensões da ordem de até 5kV.

As características da porta paralela possibilitam evitar o uso de uma fonte externa de energia, isto devido aos sinais de saída terem corrente suficiente para acionar diretamente os opto-acopladores, e também ao fato das linhas de entrada possuírem resistores *pull-up*, fazendo com que o controle do nível da entrada possa ser feito pelo simples aterramento do pino correspondente. Na seção do circuito conectado ao controlador do robô a tensão utilizada é de 24V disponível na interface da própria *Placa I/O*.

A Figura (3) apresenta o circuito de acoplamento dos sinais de uma linha de dados da porta paralela para os níveis necessários para o acionamento da linha correspondente na *Placa I/O*, incluindo-se o acionamento do *LED* de controle do painel frontal. Este circuito é replicado nove vezes, sendo um para cada linha de transmissão do *PC*.

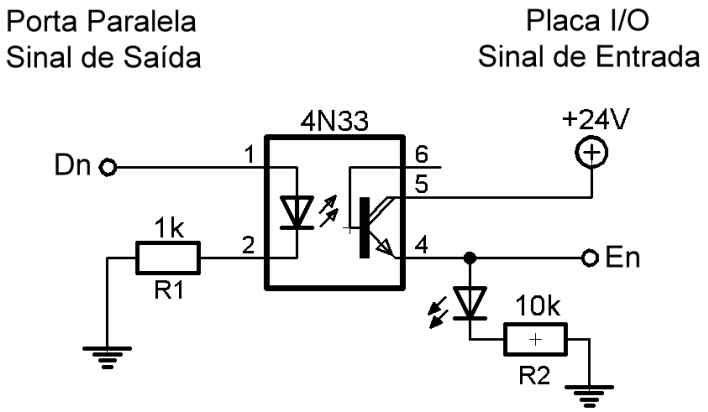

Figura 3. Acoplamento óptico de um sinal entre a porta paralela da *Placa I/O*

O circuito para o ajuste dos níveis de voltagem das linhas de dados enviados pela *Placa I/O* para o *PC* é apresentado na Fig. (4). 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*. O *Módulo IRBCom* possui cinco circuitos iguais a este, um para cada linha de sinal enviado do controlador do robô para o PC.

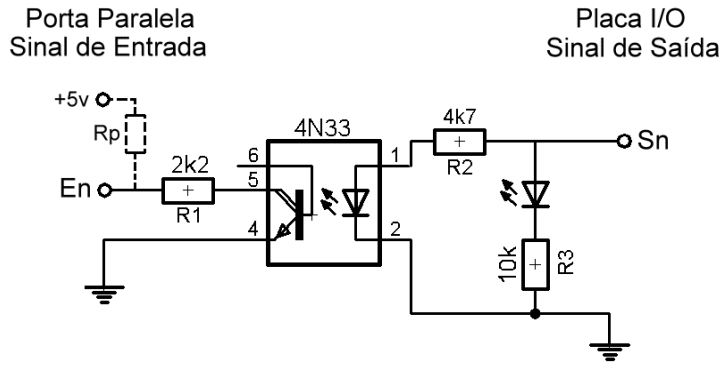

Figura 4. Acoplamento óptico de um sinal entre a *Placa I/O* e a porta paralela

Para a montagem final dos circuitos foram elaboradas placas de circuito impresso, Fig. (5), as quais foram acondicionadas em um gabinete com conectores externos, conforme Fig. (6). No painel frontal existem *LEDs* indicadores do nível de sinal presente em cada uma das linhas de comunicação, permitindo assim monitorar o funcionamento do sistema, Fig (7).

As conexões do sistema são feitas através de cabos paralelos padrão de 25 vias com conectores *DB25*.

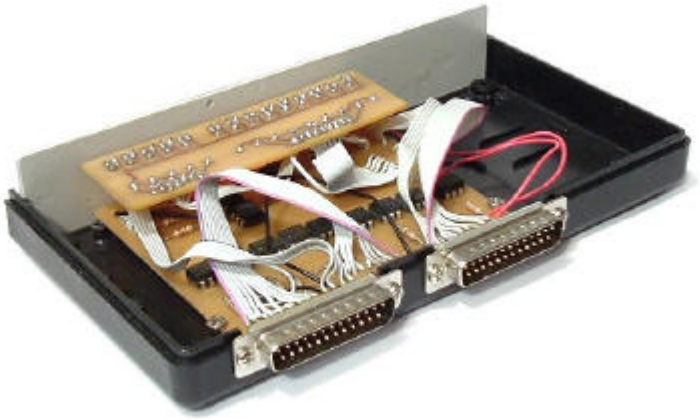

Figura 5. Circuitos montados em placas de circuito impresso.

Figura 6. Gabinete do módulo *IRBCom*, vista posterior

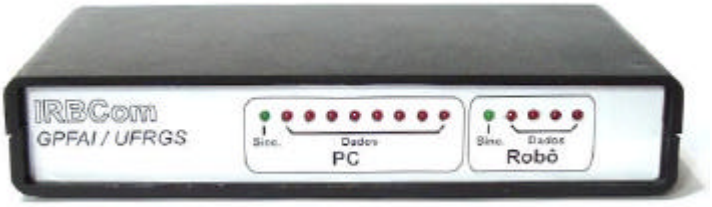

Figura 7. Gabinete do módulo *IRBCom*, vista frontal

2.4. Camada de Enlace de Dados

Esta camada consta de rotinas pré-definidas para a transmissão e recebimento de dados. As rotinas têm 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 o acesso às rotinas por qualquer linguagem de programação desenvolvida 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.

Devido a baixa velocidade de processamento do controlador do robô, que pode bloquear a comunicação durante o processamento do programa, foi necessário implementar sinais de sincronismo para garantir a máxima taxa de transmissão e evitar conflitos. O sincronismo da transmissão, representado na Fig. (8), ocorre na seguinte seqüência:

- receptor ativa seu sinal de sincronismo, indicando estar pronto 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.

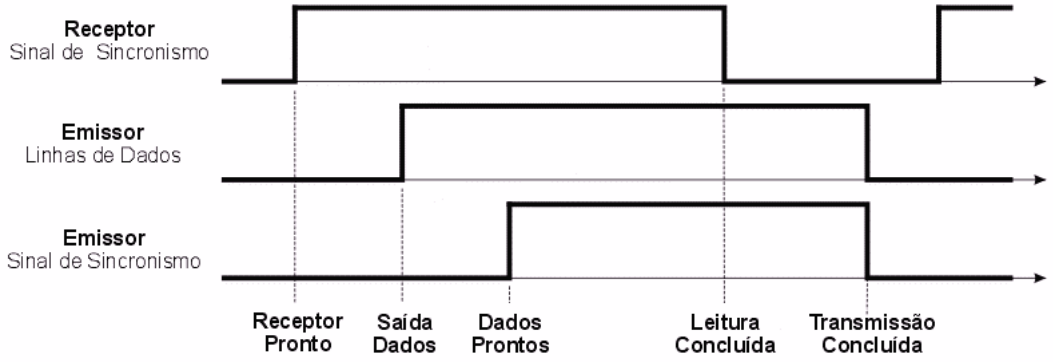

Figura 8. Esquema de sincronismo da transmissão

A implementação de rotinas dedicadas na camada de *Enlace de Dados* viabiliza o protocolo de comunicação para os diversos tipos de informações. Na Tabela (1) são indicadas as características operacionais destas rotinas.

Tabela 1. Rotinas de envio e recebimento de dados na 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.	*ReceiveInteger*
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* do emissor, em conjunto com *ReceiveByte* no receptor, é responsável pelo envio de um byte de dados do emissor para o receptor, implementando todo o procedimento de sincronismo de transmissão. As rotinas *SendInteger* e *SendReal* apenas formatam os dados durante o envio ou recebimento e outros tipos de dados podem ser compostos por rotinas em níveis superiores, utilizando-se destes três tipos padrões.

Por exemplo, para enviar um número inteiro à rotina *SendInteger* recebe o valor no formato 16 *bits* e o desmembra em dois *bytes*, os quais são enviados em seqüência utilizando *SendByte*. No receptor a rotina *ReceiveInteger* irá receber estes dois *bytes* através de *ReceiveByte*, e então irá recuperar o valor do número inteiro. No caso de envio de dados do controlador do robô para o *PC* o procedimento é o mesmo, porém como existem apenas quatro linhas de dados a rotina *SendByte* envia a informação em dois pacotes de quatro *bits*.

2.5. Protocolo de Comunicação e Camada de Aplicação

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. 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.

Como exemplo para o comando *Move*, que move a posição do *TCP* do Robô para o ponto relativo 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, Fig. (9).

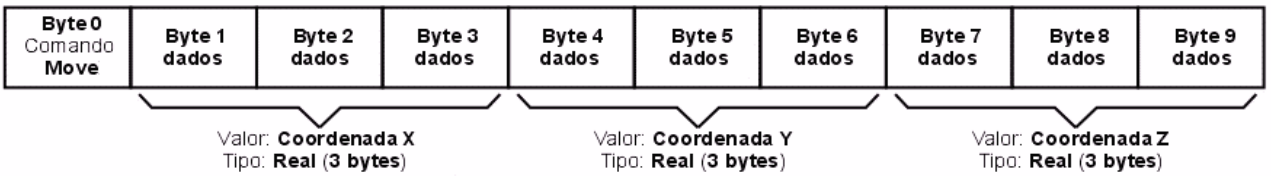

Figura 9. Exemplo de envio do comando *Move*

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.

Trabalhando-se com as rotinas padrão da *Camada de Enlace de Dados* já é possível realizar a comunicação de forma efetiva. Porém, visando facilitar a utilização 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. A Tabela (2) 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 2. Rotinas da camada de Aplicação.

Emissor	Comando	Parâmetros	Descrição
DoHome	24	-	Configura a posição atual do TCP como zero do sistema de coordenadas dos comandos Move enviados pelo PC
SetSpeed	23	1 Inteiro (2 *bytes)*	Configura a velocidade de movimento do TCP para os comandos Move enviados pelo PC.
SetZone	22	1 *byte*	Configura a zona de passagem para movimentos fly-by.
SetDO	20	1 *byte*	Ativa a saída digital da *Placa I/O* do controlador.
ResetDO	21	1 *byte*	Desativa uma saída digital da *Placa I/O* do controlador.
Move	10	3 Reais (9 *bytes*)	Move linearmente o TCP para a posição indicada pelos valores x, y e z enviados pelo PC.

2.6. O Programa Cliente Rodando no PC

O *Programa Cliente* deve ser desenvolvido pelo usuário final, sendo este o responsável pela lógica e funcionamento do sistema.

Este programa pode ser desenvolvido em qualquer linguagem que possua um compilador para a plataforma *Win32* e possa 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.

Este programa deverá ser desenvolvido em conjunto com o *Programa Servidor* para ser executado no controlador do robô, pois estes dois programas deverão 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 Cliente* utiliza 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.

2.7. O Programa Servidor Rodando no Controlador do Robô

O *Programa Servidor* é responsável pelo controle do robô, com base nos dados recebidos do *PC*. Este programa deve ser desenvolvido na linguagem nativa do controlador, no caso *ABB RAPID*. Para permitir a utilização do sistema de comunicação *IRBCom* deve-se acrescentar ao código fonte do programa as rotinas da *Camada de Enlace de Dados* e da *Camada de Aplicação* que forem necessárias.

Em uma aplicação típica, o *Programa Servidor* executado no controlador do robô tem como elemento principal 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 uma tarefa, o programa retorna ao laço principal e chama *ReceiveByte* para receber o próximo comando.

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

3. RESULTADOS

Para a validação do sistema de comunicação foi elaborado um programa de manipulação de objetos, utilizando-se de uma instalação didática presente no Laboratório de Robótica do GPFAI - Grupo de Projeto, Fabricação e Automação Industrial da UFRGS.

Neste sistema o *Programa Cliente* apresenta uma interface gráfica que permite ao usuário, através do *mouse*, selecionar o objeto a ser manipulado, sua trajetória e ponto final de descarga. Com base nesses dados é elaborada uma seqüência de comandos que são enviados ao controlador do robô através do sistema de comunicação *IRBCom*. Durante a recepção dos comandos o *Programa Servidor* realiza a interpretação em tempo real, desenvolvendo as tarefas programadas.

Para verificar a possibilidade de integração do sistema em redes de chão-de-fábrica foi desenvolvido também um módulo de tele-operação, onde um programa sendo executado em um segundo *PC*, conectado a rede interna do laboratório, envia comandos para o *Programa Cliente*. Este programa de controle remoto apresenta uma interface gráfica idêntica ao programa principal e as mesmas funcionalidades.

Durante os testes o sistema de comunicação apresentou um desempenho muito bom, apresentando taxas médias de 12,5 *bytes*/s para a transmissão e 6,1 *bytes*/s para a recepção, a partir do *PC*.

4. CONCLUSÕES

O sistema mostrou-se uma alternativa viável para a implementação da comunicação entre o robô *ABB IRB1400* e um microcomputador *PC* padrão, atingindo os objetivos iniciais de permitir a integração a redes de chão-de-fábrica, manter um baixo custo de implementação, confiabilidade e flexibilidade de utilização.

As taxas de transmissão alcançadas mostraram-se satisfatórias para o controle das tarefas, mesmo em tempo real, mostrando-se inadequadas apenas em aplicações onde se necessite envio intensivo de dados.

5. REFERÊNCIAS

ABB Robotics Products AB, 1993, “Product Manual: IRB1400 M94A/REV1T”, Sweden, 381p.

Boucher, T.O., 1996, “Computer Automation in Manufacturing”, Chapman & Hall, London, 370p.

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

Groover, M.P., 1987, “Automation, Production Systems, and Computer Integrated Manufacturing”, Prentice-Hall International, USA, 808p.

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

Mahalik, N.P., 2003, “Fieldbus Technology: Industrial Network Standarts for Real-Time Distributed Control”, Springer, Berlin, 590p.

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

6. DIREITOS AUTORAIS

LOW COST PARALLEL COMMUNICATION INTERFACE FOR INDUSTRIAL ROBOT

Fernando M. Bayer
Universidade Federal do Rio Grande do Sul – Departamento de Engenharia Mecânica
E-mail: fernandobayer@yahoo.com.br.

Flávio J. Lorini
Universidade Federal do Rio Grande do Sul – Departamento de Engenharia Mecânica
Rua Sarmento Leite 425, 90050-170, Porto Alegre, RS, Brasil.
E-mail: lorini@ufrgs.br

Abstract. *The work presented in this paper describes the development of a system for bidirectional communication between an industrial robot, originally unprovided of an integrated communication interface, and a standard PC. The developed interface uses the PC's parallel port and the robot controler's digital I/O board. The method of the transmission synchronism and the communication protocol had been developed to allow a robust and simple system. The present work discusses the description of the system, its main components, communication protocols and experimental results.*

Keywords. *Robotics, Industrial Networks, Machine Integration.*