UMA PROPOSTA DE ARQUITETURA ABERTA PARA O CONTROLE DE POSIÇÃO E FORÇA EM ROBÔS MANIPULADORES

ANDRE S. OLIVEIRA, RAUL GUENTHER

Laboratório de Robótica, Departamento de Engenharia Mecânica, Universidade Federal de Santa Catarina
Campus Universitário, Trindade, Cep 88040-90, Florianópolis, SC, Brasil.
E-mails:oliveira@emc.ufsc.br, guenther@emc.ufsc.br

Abstract— In this work we present the design and the development of a robotic controller with a totally open architecture built to force and position control. This architecture provides flexibility, the knowledge of all the control structures and gives the opportunity of changes in all the layers of the controller. The used controller conception aims to fulfill the following requirements: high capacity of processing, low cost, connectivity with other systems, availability for the remote access, flexibility in the implementation, integration with a personal computer, programming in high level and easiness of maintenance.

Keywords— Force control, Robotic controller, Open-architecture.

Resumo— Este trabalho apresenta o projeto e o desenvolvimento de um controlador robótico de arquitetura totalmente aberta feito para o controle de força e posição. Esta arquitetura fornece flexibilidade, o conhecimento de todas as estruturas de controle e dá a possibilidade de alterações em todas as camadas do controlador. A concepção de controlador utilizada visa cumprir os seguintes requisitos: alta capacidade de processamento, baixo custo, conectividade com outros sistemas, disponibilidade para acesso remoto, facilidade de manutenção, flexibilidade na implementação dos algoritmos, integração com um computador pessoal e programação em alto nível.

Palavras-chave— Controle de Força, Controlador Robótico, Arquitetura Aberta.

1. Introdução

A inclusão de realimentação de força e visão, a possibilidade de cooperação entre dois ou mais manipuladores, o controle de robôs de topologia irregular estão abrindo um novo ramo de aplicações na robótica industrial. O desenvolvimento de algoritmos de controle para esses fins leva a necessidade de se utilizar controladores de arquitetura aberta.

O "grau de abertura" de um controlador robótico varia de um sistema para outro. Geralmente, o controle de vários componentes do sistema (e.g., a unidade de potência e o controle em baixo nível) são proprietários e não podem ser modificados pelo usuário, os demais, considerados abertos (e.g. a interface de comunicação e o controle em alto nível), permitem um acesso controlado das informações.

Ford (1994) define este "grau de abertura" baseado no conceito de acesso às camadas do controlador. O autor classifica os controladores robóticos em três categorias: proprietário, híbrido e aberto. O proprietário é um sistema fechado e neste sistema é extremamente difícil ou até mesmo impossível de se integrar um *hardware* externo. Em um sistema híbrido, a maioria das camadas de controle é acessível, porém, vários aspectos da lei de controle são fechados. Em uma arquitetura aberta, todos os aspectos do projeto podem ser modificados e é permitida a adição de novos *hardwares*, a modificação da estrutura de *software* dos servos motores, etc. Esses sistemas de controle podem facilmente se adaptar a novas tarefas.

Adicionalmente é ressaltado o desenvolvimento anterior de um controlador de arquitetura aberta para o controle de posição em um espaço de trabalho confinado, para o projeto *Roboturb*, no Laboratório de Robótica (*LAR*) na Universidade Federal de Santa Catarina. O projeto *Roboturb*, em sua nova etapa, requer a adição do controle de força, o que motiva fortemente o trabalho apresentado nesse artigo. Outra motivação é realizar um *retrofit* (i.e., processo de modernização ou atualização tecnológica) em um antigo manipulador industrial, possibilitando a operação com estruturas de controle de posição e força e a implementação de leis avançadas de controle.

A maioria dos controladores robóticos de arquitetura aberta existentes é baseada nos padrões de *hardware* para *PC* e em sistemas operacionais convencionais, porque, placas de *I/O* e de comunicação para robôs têm um custo superior em relação as similares para *PC*. Outra razão é a falta de padronização dos periféricos para robôs, cada fabricante desenvolve seus próprios periféricos e interfaces, forçando os usuários a adquirir todos os componentes de um único fabricante (Lages et al. (2003)). Adicionalmente, notamos que um controlador baseado em *PC* facilita a integração de periféricos adicionais, disponíveis comercialmente como: dispositivos de armazenamento de dados, interfaces *Ethernet* e outros dispositivos de *I/O*. Assim, a facilidade de se integrar novas funcionalidades é uma forte razão para o uso de *hardwares* de *PC* nas arquiteturas abertas de controladores de robôs.

Outra razão relevante são as linguagens de programação, que para robôs são de baixo nível, mais próximas das linguagens *Assembly* do que das modernas linguagens de alto nível. Isto restringe as formas de implementação (Lages et al. (2003)). Em um controlador baseado no padrão de *PC*, as ferramentas convencionais de desenvolvimento de *software*, po-

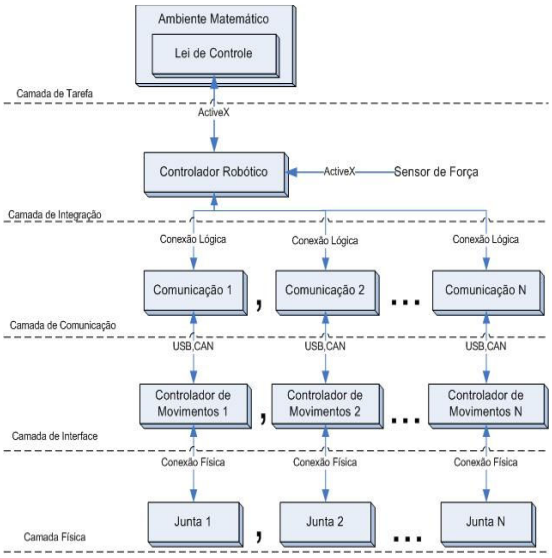

Figura 1. Arquitetura funcional em camadas.

dem ser utilizadas (e.g., *Visual C++*, *Visual Basic* ou *Delphi*).

Neste trabalho é apresentado o projeto e o desenvolvimento de um controlador robótico com uma arquitetura totalmente aberta, feito para o controle de posição e força. Esta arquitetura proporciona o conhecimento de todas as camadas de controle e fornece a possibilidade de realizar alterações em todos os níveis do controlador. A concepção de controlador utilizada visa a cumprir os seguintes requisitos: alta capacidade de processamento, baixo custo, conectividade com outros sistemas, disponibilidade para acesso remoto, facilidade de manutenção, flexibilidade na implementação dos algoritmos, integração com um computador pessoal e programação em alto nível.

Este trabalho está organizado na seguinte forma: na seção 2, é descrita a arquitetura funcional do controlador, na seção 3, é apresentado o desenvolvimento do *software*, na seção 4, é abordado o desenvolvimento do *hardware* e na seção 5 é realizando uma breve apresentação do sistema controlador.

2. Arquitetura funcional do controlador

O controlador de robôs foi desenvolvido com uma estrutura modular e uma arquitetura de comunicação derivada do padrão *IEEE 802.4* (*Token Bus*). A arquitetura funcional foi baseada na *ISO 7498-1* (Interconexão de Sistemas Abertos) e isso resulta em uma arquitetura hierárquica apresentada na Fig.1. Esta inclui as camadas de: tarefa, integração, comunicação, interface e física.

O requisito principal deste projeto é realizar um controlador com uma arquitetura totalmente aberta. Permitindo o acesso e a alteração das informações em todas essas camadas do projeto, incluindo o *software* processado pelo computador e o *firmware* processado pelos controladores de sinais digitais (*DSC*).

Figura 2. Sensor de força e o efetuador final.

Na seqüência são descritas as principais funções de cada camada.

2.1. Camada de tarefa

A camada de tarefa consiste no nível de maior abstração do sistema, onde todas as informações de controle já estão previamente tratadas. Neste ponto é gerada, armazenada e processada a lei de controle do manipulador, as rotinas de geração de trajetória e os processos supervisórios, com o auxílio de um ambiente matemático.

A informação precedente das N juntas está disponível em matrizes $n \times 1$, que correspondem ao vetor de posições e ao vetor de velocidades, onde as linhas representam as juntas robóticas. As grandezas provenientes do sensor de força são armazenadas em uma matriz 6×1 ($h = [f_x\ f_y\ f_z\ \mu_x\ \mu_y\ \mu_z]^T$), que contém as medidas de forças e momentos no efetuador final. As informações destinadas ao manipulador (i.e., perfis de posição, velocidade e aceleração) estão em uma matriz de controle $n \times 3$ denominada de matriz u.

2.2. Camada de integração

A camada de integração é responsável pela adequação (i.e., concatenação e organização) de todas as informações trocadas entre o computador e o manipulador. No caso da inclusão de um novo *hardware* ao sistema, é necessário adicionar a sua estrutura de controle a esta camada. Nesta também é realizado o controle do gabinete de acionamento do manipulador (composto pela unidade de potência e os acionamentos elétricos), habilitando e desabilitando o manipulador, e prevenindo movimentos irregulares e situações de perigo.

Este é o ponto da arquitetura onde o sistema tem seu processamento distribuído e essa camada é responsável por esse controle. Um *software* de controle, denominado de *middleware*, controla as comunicações com os processadores que regem os sistemas embarcados e a intercomunicação de processos para a aquisição de dados do sensor de força. Para a reliamentação de força, foi acoplado ao punho do manipulador robótico, um sensor de força e momentos de seis eixos. Cujo suporta cargas de 200N nos eixos *x* e *y*, e 400N no eixo *z*, visualizado na Fig. 2.

Em sistemas de processamento distribuído, o *middleware* realiza a adequação dos sinais vindos

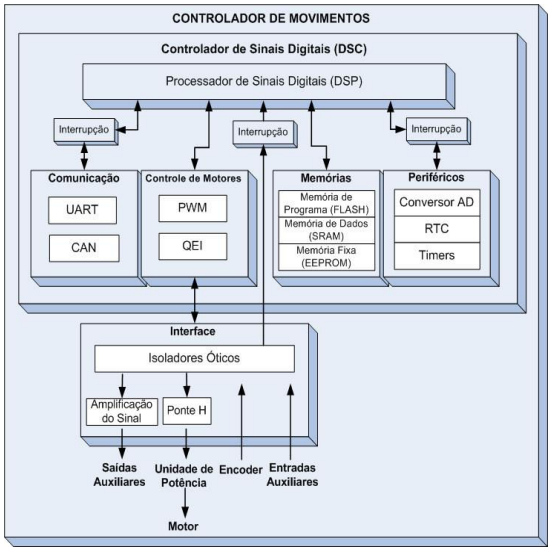

Figura 3. Diagrama de blocos do controlador de movimentos.

dos *N* processadores que compõem o sistema. Sua função primordial é abstrair a camada superior a ação de computação distribuída. Em sistemas distribuídos com ramos diferenciados, o *middleware* oculta diferenças de protocolos de comunicação, plataformas e dependências do sistema operacional. Fornecendo as camadas superiores uma visão de heterogeneidade do sistema e dando suporte para o compartilhamento de recursos. São dotados de interfaces de programação de aplicativos (*APIs*) de alto nível para proporcionar a sua integração com aplicações desenvolvidas em quaisquer linguagens de programação.

2.3. Camada de comunicação

A camada de comunicação controla a transferência de dados pela interface *USB 2.0* (*Universal Serial Bus*) e pelo protocolo industrial *CAN* (*Campus Area Network*), ambos, métodos de comunicação de alto desempenho (i.e., alta velocidade de transferência de informações). A *USB* realiza uma interconexão do sistema através de uma topologia em estrela, tendo um computador como nó central. Cada porta *USB* suporta até 127 dispositivos, desta forma é possível se conectar uma grande quantidade de juntas ao controlador. O protocolo *CAN* forma um barramento entre os nós secundários (controladores de movimentos).

A composição da topologia estrela com o barramento fornece uma rede de processadores, com um perfil derivado do modelo *Token-Bus* definido pela norma *ISO/IEC 8802-4*. O diferencial é que o anel não é lógico, mas sim uma forma de comunicação redundante ao sistema. Um ponto importante deste padrão é a independência sobre a ordem física de conexão, visto que todas as estações recebem todos os quadros (i.e, informações ou dados) transmitidos, descartando os que não forem endereçados a ela. Assim, a inclusão ou exclusão de nós a rede deve ser organizada (Tanenbaum, 1997).

Figura 4. Robô *REIS Rv15*.

A inclusão das formas de comunicação possibilita ao sistema operar com qualquer um dos métodos ou com ambos, ampliando suas funcionalidades. O barramento fornece uma independência sobre a utilização de um nó central especializado, assim é possível montar o sistema sem o *PC*, com um controlador totalmente embarcado. Neste caso, o sistema deverá operar com uma hierarquia de juntas, para a adequação e processamento completo dos dados.

2.4 Camada de interface

A camada de interface é composta pelos sistemas embarcados que realizam o controle das juntas robóticas, denominados de controladores de movimentos. Seu diagrama de blocos internos é apresentado na Fig. 3.

Como pode ser observado na Fig. 3, cada um desses controladores digitais de sinais decodifica o seu correspondente sinal de encoder e gera a modulação por largura de pulso (*PWM*) para o controle do respectivo atuador. Cada um desses sistemas tem uma interface de isolamento ótico para prevenir qualquer retorno inadequado aos motores. Ainda possuem uma grande quantidade de portas de expansão, as quais permitem a conexão de outras ferramentas.

O desenvolvimento do controlador visou uma arquitetura modular, de forma a obter um controle independente das juntas, dividindo a complexidade matemática entre os processadores do sistema. Isto resulta em um processamento distribuído organizado pelo nó central (computador), onde as operações ocorrem em paralelo. Essa metodologia facilita a expansão e a manutenção do sistema.

2.5. Camada física

A camada mais inferior, aqui denominada de camada física, compreende na unidade de potência, integrante do gabinete de acionamento do manipulador, e no robô industrial.

O manipulador industrial utilizado é do modelo *REIS Rv15* (Fig.4) com seis graus de liberdade. Sua arquitetura é comum em aplicações industriais, sendo

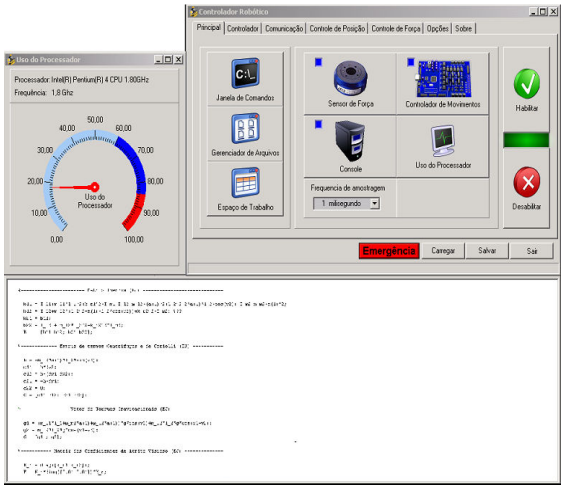

Figura 5. Software controlador robótico.

constituído por um braço antropomórfico com um punho esférico.

Do ponto de vista do controlador, o manipulador consiste em seis juntas rotativas atuadas por motores elétricos de corrente contínua e *encoders* óticos incrementais (com 16 bits de resolução) para as medidas das posições angulares da juntas.

3. Desenvolvimento do *software*

O *software* foi desenvolvido utilizando uma linguagem de alto nível orientada a objetos (*C++*). Em sua página principal oferece opções automatizadas de utilização (i.e., inicializações e configurações) do controlador, como pode ser visto na Fig. 5. Porém, as paginas secundárias disponibilizam configurações avançadas (i.e., configurações individuais e especificas do controlador, da comunicação, do gabinete de acionamento, do controle de posição e do controle de força), como demonstrado na Fig.6.

Ainda existe um monitoramento das atividades do processador do *PC*, pois a maioria dos processos funciona por meio de *threads*. Assim quando o processador chegar a um nível crítico, as prioridades das *threads* são alteradas. Favorecendo as tarefas mais essenciais ao controlador. Essa opção como as demais do sistema, pode ser habilitada ou desabilitada a qualquer instante.

A comunicação com o sensor de força e com o ambiente matemático se dá pelo uso da intercomunicação de aplicações, utilizando a tecnologia do *ActiveX*. Atualmente, a *Microsoft*, vem substituindo gradativamente pela plataforma *.Net Framework*, a qual aprimora seus recursos.

Esse método possibilita que o controle seja feito em qualquer programa que possua a biblioteca para esse tipo de transferência de dados.

O programa foi desenvolvido inicialmente para o sistema operacional *Windows*, com o objetivo de validar o controlador. Porém, uma segunda versão do sistema já esta sendo desenvolvida em *Linux OS*, usando a sua variante para processamento em tempo

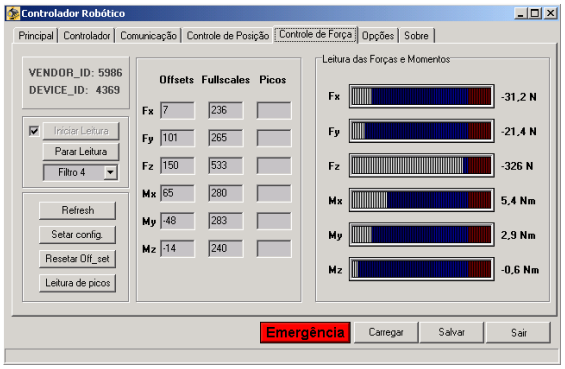

Figura 6. Configurações avançadas do *software* (controle de força).

real (*Real Time Application Interface*), a fim de melhorar a velocidade de processamento, mantendo os mesmos recursos da versão já implementada.

4. Desenvolvimento do *hardware*

A estrutura hierárquica da arquitetura funcional adotada, junto com a articulação entre os diferentes módulos, sugere uma implementação de um *hardware* que explore os recursos da computação distribuída e interconectada, por meio de canais de comunicação apropriados (Oliveira et al. (2006)). Essas características adéquam o *hardware* desenvolvido a operar cumprindo o requisito especial para um controlador robótico operar com controle de força apresentado a seguir.

4.1 Requisito especial para controladores robóticos que incluem o controle de força

Geralmente, os controladores robóticos são desenvolvidos para aplicações que necessitam apenas do controle de posição, ou seja, o efetuador do manipulador não realiza contato algum durante o movimento em seu espaço de trabalho. Em aplicações que necessitam do controle de força, o efetuador tem contato com alguma superfície em seu espaço de trabalho. Esta interação gera forças de contato que devem ser controladas, de maneira a cumprir corretamente a tarefa, sem causar danos a ambos, a ferramenta do robô e ao objeto trabalhado.

As intensidades das forças de contato, originadas pelo movimento da ferramenta comandada pelo controlador robótico, depende de ambos, da rigidez da ferramenta e da rigidez da superfície, que devem ser controladas. Um pequeno movimento da ferramenta pode originar grandes intensidades de força, no caso da superfície da ferramenta e do objeto serem muito rígidas. Pode-se notar que, com a introdução de complacência na ferramenta gera um atraso na aplicação de forças e isto pode ser inaceitável para várias aplicações. Conseqüentemente, o sistema deve ter um pequeno tempo de resposta a estas forças, para prevenir que a ferramenta, o robô ou o objeto se danifiquem.

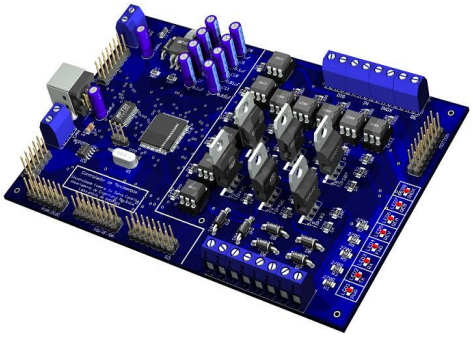

Figura 7. Layout do controlador de movimentos.

O uso de sistemas de alto desempenho que conseqüentemente possibilitam um baixo tempo de resposta, a ação de forças sobre o efetuador do manipulador, é um requisito para controladores que empregam o controle de força.

4.2 Descrição do desenvolvimento

O *hardware* do sistema foi projetado e construído visando o alto desempenho e a confiabilidade, o baixo custo e a utilização de componentes facilmente encontrados no mercado.

O principal componente dos controladores de movimento é um controlador de sinais digitais (*DSC*) produzido pela *Microchip Technology Inc.* denominado de *dsPIC30F6010A*. Ele opera com 16 bits a uma freqüência de 120MHz, tem encapsulamento *TQFP* de 80 pinos e é integrante da família de controle de motores. Ainda possui uma grande variedade de módulos internos bem diferenciados, como: uma ampla memória de programa, com 144KB, e uma memória não-volátil, com 4 KB para o armazenamento de informações. O *DSC* também possui 16 vias de conversão analógico-digital (*A/D*) e os módulos necessários para implementar a topologia de comunicação do sistema.

Para a comunicação através da *USB* nós utilizamos um componente que realiza a conversão do módulo receptor/transmissor universal assíncrono (*UART*) para este barramento. Este componente suporta taxas de transferências de até 3 *Megabaud* e é produzido pela *Future Technology Devices International Ltd.* O componente também possui outras funcionalidades como, por exemplo, a geração de um sinal oscilador digital externo com freqüências variáveis. Além disso, o fabricante disponibiliza *drivers Royalty-Free* do componente para uso em diversos sistemas operacionais.

Para implementar todos os requisitos da camada física definidos pela *ISO-11898*, foi conectado ao protocolo industrial *CAN* um transciever de alta velocidade, construído de acordo com essa norma, que suporta taxas de até 1Mb/s.

A implementação do sistema utiliza a linguagem de programação *C* que é completamente modular e organizado em bibliotecas, para facilitar as modificações. Todos os módulos operam com interrupções do

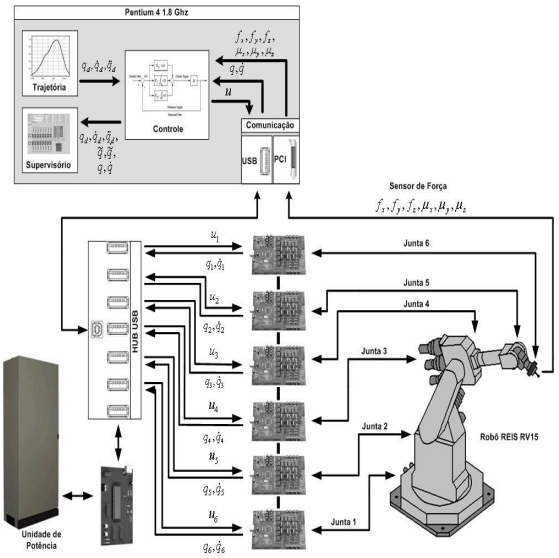

Figura 8. Diagrama completo do controlador robótico de arquitetura aberta.

processador de prioridades distintas, de forma que nenhuma operação de menor prioridade atrase um processo de maior importância.

O módulo de controle de movimentos é composto também por um gerador de *PWM* de 16 bits e um módulo de leitura de encoders de quadratura (*QEI*), que foi ampliado neste projeto para 32 bits. Conectados a esses, existe uma barreira de desacoplamento ótico e uma ponte-H, para o controle da unidade de potência que suporta até 100V e 8A. Onde também estão conectados os canais amplificados de saídas auxiliares, que operam até 100V e 6A. Para proteger o sistema, as entradas do encoder e as entradas auxiliares também estão conectadas a essa barreira. Internamente ainda existe uma grande quantidade de recursos que não foram utilizados e podem ser úteis em futuros *upgrades* do sistema.

O *layout* resultante do controlador de movimento é mostrado na Fig. 7.

5. O Controlador de robôs

A Fig. 8 apresenta o diagrama completo do controlador de robôs com arquitetura aberta. Onde pode ser identificado o alto grau de modularidade, que promove o processamento paralelo e distribuído entre os sistemas embarcados (i.e., nós secundários) que compõem o sistema. Também é demonstrado o fluxo de informações e as interconexões, desde o computador (i.e., o nó central) até o manipulador industrial (i.e., os atuadores, *encoders* e o sensor de força)

6. Conclusão

Neste trabalho é apresentado o projeto e o desenvolvimento completo (i.e., do *software*, *middle-*

ware, *hardware* e *firmware*) de um controlador de robôs com arquitetura aberta, visando o controle de posição e força em manipuladores industriais.

O resultado é um controlador com as seguintes propriedades: alta capacidade de processamento, baixo custo, conectividade com outros sistemas, disponibilidade de acesso remoto, facilidade na manutenção, flexibilidade de implementação, integração com um computador pessoal e programação em alto-nível.

Onde também é realizada a atualização tecnológica (i.e., *retrofitting*) do robô *REIS Rv15* com a utilização do modelo de controlador proposto, e com a adição de um sensor de força ao sistema.

Atualmente, o sistema está passando por sua validação experimental com a implementação de uma estrutura de controle de força indireto, o controle de impedância.

Agradecimento

Este trabalho foi parcialmente apoiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (*CNPQ*).

Referências bibliográficas

Ford, W. (1994). What is an Open Architecture Robot Controller?, *9th IEEE Int. Symp. On Intelligent Control*, pp. 027-032,

IS Association, (1990). IEEE Standard 802.4 : Token Bus.

ISO, (1993). ISO Standard: 11898: Road Vehicles, Interchange of digital information – Controller Area Network (CAN) for high speed communications.

ISO, (1994). ISO/IEC Standard 7498-1: Information Technology – Open Systems Interconnection – Basic reference model.

Lages, W. F., Henriques, R. V. B., e Bracarense, A. Q. (2003). Arquitetura Aberta para Retrofitting de Robôs. *Manet Notes Workshop, Bragança Paulista*, São Paulo.

Oliveira, A.S. e Andrade, F. S. (2006). Sistemas Embarcados : Hardware e Firmware na Prática. *Ed. Érica*, São Paulo.

Pires, J.N., Ramming, J., Rauch, S. e Aráujo, R. (2002). Force/torque sensing applied to industrial robotic deburring, *Sensor Review Journal*, Vol. 22 No.3, pp. 232-241.

Sciavicco, L. e Siciliano, B. (2002). Modeling and control of robot manipulators, *The McGraw-Hill Companies*.

Tanenbaum, A. S. (1997). Redes de Computadores, Rio de Janeiro, *Editora Campus*.