ANDRÉ SCHNEIDER DE OLIVEIRA

RETROFITTING DE ROBÔS MANIPULADORES COM INCORPORAÇÃO DE CONTROLE DE POSIÇÃO E FORÇA: APLICAÇÃO EM UM ROBÔ INDUSTRIAL

Florianópolis – SC
2007

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

RETROFITTING DE ROBÔS MANIPULADORES COM INCORPORAÇÃO DE CONTROLE DE POSIÇÃO E FORÇA: APLICAÇÃO EM UM ROBÔ INDUSTRIAL

Dissertação submetida à

UNIVERSIDADE FEDERAL DE SANTA CATARINA

para a obtenção do grau de

MESTRE EM ENGENHARIA MECÂNICA

por

ANDRÉ SCHNEIDER DE OLIVEIRA

Florianópolis, setembro de 2007.

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA MECÂNICA

RETROFITTING DE ROBÔS MANIPULADORES COM INCORPORAÇÃO DE CONTROLE DE POSIÇÃO E FORÇA: APLICAÇÃO EM UM ROBÔ INDUSTRIAL

ANDRÉ SCHNEIDER DE OLIVEIRA

Esta dissertação foi julgada adequada para a obtenção do título de

MESTRE EM ENGENHARIA

ESPECIALIDADE ENGENHARIA MECÂNICA
sendo aprovada em sua forma final.

_____
Edson Roberto De Pieri, Dr.- Orientador
Raul Guenther, D.Sc. (in memoriam)

_____
Daniel Martins, Dr.Eng. – Co-orientador

_____
Fernando Cabral, Ph.D.
Coordenador do Programa de Pós-Graduação em Engenharia Mecânica

Banca Examinadora

_____
Altamir Dias, D.Sc.

_____
Carlos Alberto Martin, Dr.Ing.

_____
Marcelo Ricardo Stemmer, Dr.Ing.

AGRADECIMENTOS

Um agradecimento especial ao saudoso Professor Raul Guenther pela orientação, pelo conhecimento e principalmente pela amizade.

Ao professor Edson Roberto De Pieri que assumiu este trabalho em caráter especial e ajudou a terminá-lo.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (*CNPQ*) por ter financiado parte deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

SÍMBOLOGIA

B	Matriz de inércia
C	Matriz de forças e torques centrífugos e Coriolis
F_v	Matriz de atrito viscoso
g	Vetor de termos gravitacionais
u	Entrada de controle
h	Forças externas exercidas sobre o manipulador
J	Jacobiano Geométrico
K_P	Ganho proporcional
K_D	Ganho derivativo
q	Posição
$\dot{q}$	Velocidade
$\ddot{q}$	Aceleração
$\ddot{q}_D$	Aceleração desejada
$\dot{q}_D$	Velocidade desejada
q_D	Posição desejada
x	Posição Angular
$\dot{x}$	Velocidade Linear e Angular
$\ddot{x}$	Aceleração Linear e Angular

RESUMO

Várias aplicações atuais da robótica estão limitadas pelo estado da arte dos algoritmos de controle dos robôs manipuladores. A inclusão de realimentação de força e visão, a possibilidade de cooperação entre dois ou mais manipuladores, o controle de robôs de topologia não-convencional estão abrindo um novo ramo de aplicações na robótica industrial. A implementação de algoritmos de controle para esses fins leva a necessidade de se utilizar controladores de arquitetura aberta.

Geralmente, os controladores de robôs são desenvolvidos para o controle de posição, não cumprindo integralmente os requisitos das tarefas onde ocorre a interação com o ambiente, tornando esta uma das principais áreas de pesquisa da robótica atualmente. Para considerar esta interação, o controlador de robôs prioriza o tempo de resposta do controle de força, pois no instante em que o efetuador entra em contato com uma superfície, várias forças atuam sobre o sistema. Dependendo das velocidades e acelerações envolvidas no processo, podem acontecer danos ao sistema. Para prevenir esses efeitos, complacências são comumente inseridas na ferramenta ou na superfície de operação.

Este trabalho apresenta o projeto e o desenvolvimento de um controlador de robôs de arquitetura aberta para o controle de posição e força, utilizando técnicas de processamento paralelo e distribuído, diminuindo a necessidade de complacência no sistema, permitindo um processamento em tempo real de aplicação e o controle total das informações. Esta arquitetura permite flexibilidade, conhecimento de todas as estruturas de controle e possibilidade de alterações em todas as camadas do controlador. A concepção de controlador utilizada visa cumprir ainda 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 de Robôs e Arquitetura Aberta.

ABSTRACT

Many current robotic applications are limited by the industry state of art of the manipulators control algorithms. The inclusion of force and vision feedbacks, the possibility of cooperation between two or more manipulators, the control of robots with irregular topology will certainly enlarge the industrial robotics applications. The development of control algorithms to this end brings the necessity of the use of controllers with open architecture.

Generally the robotic controllers are developed for position control, without accomplishing integrally the requirements of tasks in which interactions with the environment occur. Therefore, this is currently one of the main research areas in robotics. To consider this interaction the robot controller has to give priority to the force control time response, because in the instant of end-effector's contact with the surface, several forces act on the system. Depending on the speeds and the accelerations involved in the process, damages (errors) can occur. To avoid these effects, compliances are inserted in tool or in surface of operation.

This work presents the development of an open-architecture robotic controller for position and force control, which uses parallel and distributed processing techniques, and avoids the necessity of compliance in system, allowing a real-time processing of the application and the total control of information. This architecture provides flexibility, the knowledge of all the control structures and allows the user to modify all the layers of the controller. The used controller conception aims to fulfill with the following requirements: high capacity of processing, low cost, connectivity with other systems, availability for the remote access, easiness of maintenance, flexibility in the implementation, integration with a personal computer and programming in high level.

Keywords: Retrofitting, Force control, Robotic controller and Open-architecture.

Capítulo 1. Introdução

Há algum tempo, as aplicações de robôs industriais foram expandidas, deixando de operar apenas em tarefas de posicionamento do efetuador ou na manipulação de cargas. Atualmente, a maior parte das pesquisas usando robôs inclui tarefas onde o manipulador realiza algum contato ou exerce forças durante a execução de sua trajetória, ou da tarefa.

A ampliação das aplicações para manipuladores industriais deu-se pelo avanço nas pesquisas sobre as estruturas de controle que levaram ao suporte de uma grande diversidade de formas de realimentação de sinal. Entretanto, as pesquisas nesta área desenvolveram-se principalmente devido aos avanços tecnológicos que sofreram os controladores de robôs. Estes cada vez mais incrementam sua capacidade de processamento e assim, possibilitam o emprego de estruturas de controle mais avançadas.

1.1 Contextualização do problema

A inclusão da realimentação de força e visão, a possibilidade de cooperação entre dois ou mais manipuladores, o controle de robôs de topologia não-convencional ampliaram o ramo de aplicações da robótica industrial. O desenvolvimento de algoritmos de controle para esses fins leva à necessidade de se utilizar controladores de arquitetura aberta. A seguir serão apresentadas resumidamente essas áreas de aplicação de robôs industriais, caracterizando os requisitos que exigem a adoção dessa modalidade de controladores.

1.1.1 *Manipuladores de topologia não-convencional*

Diversas topologias de robôs manipuladores são consideradas regulares, pois são muito difundidas e utilizadas, como: o braço antropomórfico com punho esférico e o *SCARA*. Quando se utiliza essas concepções, informações como o efeito das dinâmicas, o espaço de trabalho alcançável, o espaço de trabalho *dexterous* (com destreza) e a formulação matemática para cinemática, são bastante discutidas na literatura, contrariamente ao uso de manipuladores de topologia não-convencional, cuja documentação é escassa e muito pouco difundida. Onde

geralmente, todas essas grandezas devem ser rigorosamente obtidas. Uma das principais dificuldades de se operar com esse tipo de sistema robótico está na influência de parâmetros dinâmicos (do manipulador e do ambiente de trabalho) imprecisamente calculados. A lei de controle deve ser preparada para operar com esse nível de incerteza.

O controle de manipuladores de topologia irregular pode requerer estruturas mais avançadas, que necessitem: uma matemática mais elaborada, exploração da redundância ou a manipulação com destreza[1], operação em ambientes confinados requerendo um controle ativo para evitar colisão, integração sensores especiais para a modelagem do ambiente ou a leitura do estado do manipulador. Um controlador de robôs com arquitetura aberta promove flexibilidade na implementação[2] de estruturas de controle e na integração de dispositivos, proporcionando as necessidades para a operação de manipuladores com topologia irregular.

1.1.2 *Cooperação robótica*

O ramo de aplicações robóticas cooperativas, atualmente, está em evidência, visto que robôs já desempenham suas tarefas eficientemente quando independentes e então poderiam ser alocados para novas funções mais complexas com o uso de apenas um grupo de manipuladores. Além disso, existem tarefas que não conseguem ser cumpridas com o uso de um robô. Tarefas que necessitam de cooperação de dois ou mais robôs incluem a manipulação e transporte de objetos largos (Fig.1) ou longos, barras pesadas, objetos flexíveis (Tzafestas, Prokopiou *et al.*, 1998).

---

[1] *A manipulação com destreza combina a modelagem das características físicas da ferramenta com o ambiente de interação para a geração das trajetórias de cumprimento da tarefa. Esta aptidão condiz com um movimento em relação ao comportamento de velocidade, aceleração, resistência do meio e força. A conceituação define que a tarefa é modelada em relação ao objeto e as forças atuantes sobre ele. Esta habilidade necessita do conhecimento das relações geométricas do sistema de manipulação do objeto com esse rigor, incluindo as localizações de contato, o objeto, as geométricas do efetuador final, elos e juntas, e, a cinemática e dinâmica do conjunto (Okamura, Smaby et al., 2000).*

[2] *Termo comumente utilizado na área da informática, o qual corresponde à elaboração e preparação de um algoritmo.*

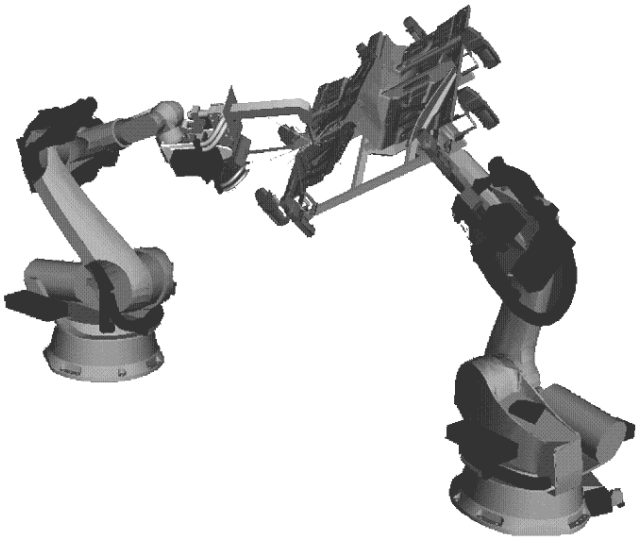

Figura 1 - Cooperação robótica (Weierstrass Institute for Applied Analysis and Stochastics, 2007).

.

A cooperação entre robôs pode ser classificada em duas categorias, segundo Tzafestas, Prokopiou *et al.*, (1998), relativas a seus graus de interação. A primeira enquadra sistemas multi-robôs em que cada um realiza suas tarefas independentemente, mas compartilham seus espaços de trabalho. Na região de intersecção de dois ou mais espaços de trabalho, existe uma forma segura de utilização do meio, evitando colisões. Os movimentos na região de possível colisão devem ser bem controlados e previstos. Artifícios, como atrasos, são inseridos na execução da tarefa, garantindo que apenas um manipulador acesse o meio em um determinado momento, enquanto os demais esperam o espaço estar livre para realizar o movimento. Porém, essa forma de controle de acesso promove uma perda de rendimento, devido à possibilidade de haver um compartilhamento simultâneo do meio, o que requer técnicas de controle mais elaboradas. A classe incorpora leis de controle mais complexas, pois, gerencia situações onde os robôs realizam uma interação direta para o cumprimento da tarefa. Neste grupo os robôs encontram-se fortemente acoplados e o espaço de trabalho de cada manipulador é determinado pela trajetória desejada para o objeto. As tarefas de cooperação podem ser classificadas segundo seu grau de interação (Tzafestas, Prokopiou *et al.*, 1998), em:

- *Tarefas onde não existe movimento relativo* entre o objeto e os efetuadores finais, ambos os corpos estão em contato, formando um único corpo rígido que parte de uma posição e orientação inicial para outra. Nesta categoria são

abordados problemas de controle desacoplado, controle de força, controle híbrido de posição/força e distribuição de força/carga.

- *Tarefas de manipulação de partes móveis*, os manipuladores possuem uma relação bem definida de movimento e/ou força para o cumprimento da tarefa.
- *Tarefas de manipulação de grandes objetos* consistem na manipulação de objetos que não podem ser seguros com apenas um efetuador, devido a suas dimensões. Neste caso, os robôs aplicam forças para empurrar o objeto, assim, o sistema causa um confinamento unilateral e o objeto pode apenas realizar um deslocamento linear ou uma rotação sobre a superfície de contato.

A utilização de controladores de arquitetura aberta em tarefas de cooperação robótica é ressaltada pelo acoplamento dos manipuladores que compõem o sistema. O sistema necessita de uma interconexão ativa entre todos os robôs, formalizando uma rede de comunicação. Esses sistemas podem possuir manipuladores diferentes, com protocolos e interfaces de comunicação distintas, que necessitam ser integradas. A implementação de protocolos de comunicação mais elaborados (ou avançados) pode ser uma necessidade caso seja necessário incrementar a velocidade de transferência de dados. Controladores de robôs de arquitetura aberta permitem a flexibilidade de integração necessária para realizar a interconexão do sistema.

1.1.3 *Realimentação por visão*

Robôs industriais geralmente utilizam unicamente sensores proprioceptivos (i.e., sensores que fazem a leitura do estado interno do manipulador) para posição de junta, os quais realizam a leitura do deslocamento rotativo fornecendo uma grandeza interna do sistema, ou seja, uma componente do movimento absoluto do sistema. A inclusão de sensores exteroceptivos (i.e., sensores que realizam a leitura do estado do manipulador em relação ao ambiente, e.g., sensores de força e visão) fornece a possibilidade de um controle de interações reais entre o sistema e o ambiente.

Força e visão são dois métodos complementares de realimentação, utilizados em controle inteligente (ou avançado) de robôs. Os sensores de força fornecem a informação localizada sobre o ambiente (na região de contato), importante para os movimentos

complacentes dos robôs. Sensores de visão promovem a informação global sobre a posição dos objetos dentro da área de manipulação (Smits, Bruyninckx *et al.*, 2006).

A visão robótica utilizando sensores de força é uma visão de uma dimensão, assim o manipulador pode realizar operações como o seguimento de uma superfície, com uma determinada força aplicada. A realimentação de visão adiciona a capacidade de uma aplicação baseada em duas dimensões, visto a utilização de apenas uma câmera. Ao unir força e visão tem-se um conhecimento em três dimensões do ambiente, pois visualmente se tem uma visão planar sobre o espaço de trabalho da tarefa e o sentido de profundidade é proporcionado pelo transdutor de força (Smits, Bruyninckx *et al.*, 2006).

O acoplamento da realimentação de visão e força, para promover a leitura do posicionamento do efetuador em três dimensões, necessita da integração de novos sensores ao sistema. O controle de posição e força necessita de uma resposta rápida à ação de forças, de forma a prevenir danos ao sistema. O processamento de vídeo, para realizar a realimentação por visão, é uma tarefa que requer um alto processamento em um curto intervalo de tempo (caracterizado como operação em tempo real de aplicação). Ambas as necessidades levam ao emprego de arquiteturas abertas para controladores de robôs.

1.1.4 *Realimentação de força*

Controladores de arquitetura aberta são necessários para estruturas de controle que possuem realimentação de força, de forma a minimizar o tempo de reposta do sistema à aplicação de força, sem perder seu desempenho no processamento de informações.

A realimentação de força vem sendo aplicada no controle de manipuladores, pois, possibilita a execução de tarefas de interação com um objeto ou uma superfície. Porém, a maioria das pesquisas utiliza esse tipo de recursos para tarefas como o polimento, o esmerilhamento, a furação ou o seguimento de superfície. A seguir serão apresentadas sucintamente tarefas diferenciadas e não menos importantes, para os sistemas de controle de posição e força em robôs manipuladores.

1.1.4.1 ***Cirurgia robótica***

Atualmente, existe uma grande tendência na utilização de robôs manipuladores para executar cirurgias devido principalmente à precisão de movimento que é proporcionada pelo uso de um sistema robótico. Nesses, o cirurgião opera um console que fornece uma visão *3D* da área a ser operada e dos instrumentos cirúrgicos. Esses sistemas realizam uma realimentação de força fornecendo a sensibilidade tátil. Dessa forma, a força exercida sobre o efetuador do braço robótico também é exercida sobre o instrumento de manipulação do cirurgião.

Sistemas robóticos para cirurgia são utilizados em diversos ramos da medicina, o mais expressivo é produzido pela *Intuitive Surgical. Inc.*, denominado de Sistema Cirúrgico *da Vinci*.

Em Esposito, Ilbeigi *et al.*, (2005) é apresentada uma nova técnica para o tratamento de câncer de próstata com a utilização de um sistema robótico *da Vinci* com três braços interativos e um braço manipulador da câmera. Estes promovem a capacidade de manipulação de três ferramentas cirúrgicas diferentes em um ambiente tridimensional. A cada braço pode ser acoplada uma ferramenta diferenciada para a laparoscopia[3], que permitem sete graus de liberdade de movimento. Uma questão relevante nesse sistema é a possibilidade de colisão dos braços manipuladores, que deve ser evitada. A Fig.2 mostra o sistema robótico para cirurgias com quatro braços articulados e o console de manipulação. Está técnica utilizando o sistema robótico foi validada em 154 pacientes.

---

[3] *Técnica que consiste na introdução de um micro-dispositivo para a aquisição de imagens internas ao corpo humano, mais precisamente na área do abdômen.*

Figura 2 - Sistema cirúrgico da Vinci de quatro braços (Esposito, Ilbeigi et al., 2005).

A utilização desses sistemas robóticos também é realizada em operações torácicas, conforme descrito em Bodner, Wykypiel *et al.*, (2004), onde é realizada uma endoscopia (procedimento para o diagnostico através de imagens na região torácica, através da introdução de um endoscópio). Nestes experimentos foi utilizado um sistema cirurgico robótico *da Vinci*, com três braços, onde: o central manipula o endoscópio e os laterais movimentam os instrumentos médicos. Os equipamentos cirúrgicos são introduzidos por pequenos orifícios, de 10 mm para a câmera e de 8 mm para os instrumentos. O sistema robótico tem uma redução de movimentos de 3:1, isso determina que a uma manipulação do cirurgião, o braço determinado realizará um movimento três vezes menor, aumentando a precisão dos movimentos.

Figura 3 - Robô cirúrgico da Vinci com três braços articulados em operação (Menzel e D'aluisio, 2001).

A Fig.3 apresenta um robô cirúrgico *da Vinci*, composto por três braços articulados, em uma micro-cirurgia de demonstração realizada sobre um cadáver. Neleum cirurgião cardíaco realiza uma operação através de um orifício de um centímetro de espessura, sensivelmente inferior às técnicas utilizadas sem o equipamento robótico. A diminuição do corte de tecido humano reduz a dor e o tempo de recuperação do paciente após os procedimentos operatórios (Menzel e D'aluisio, 2001).

1.1.4.2 *Veiculo Subaquático*

Algumas pesquisas podem ser encontradas aplicando o uso de braços robóticos acoplados a veículos aquáticos. O diferencial desse controle é a relação direta entre a força exercida pelo braço sobre a superfície com a pressão gerada pelos propulsores do veículo (Antonelli, 2003), (Santos, 2006).

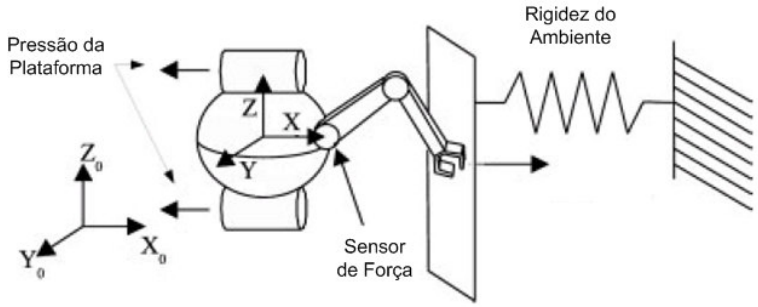

Figura 4 - Veículo subaquático com braço mecânico acoplado numa interação com o ambiente (Lionel Lapierre, 2003).

Em Lionel Lapierre, (2003), é proposto um método de controle de força para estabilizar um veículo subaquático quando o manipulador acoplado a este, opera em ambiente livre ou confinado. O torque produzido pelo braço da plataforma é estimado através da utilização de um transdutor de força, que está entre a base do manipulador e do veículo. O diagrama da aplicação pode ser visto na Fig.4, onde são consideradas as dinâmicas do acoplamento do sistema.

1.1.4.3 ***Mão Robótica***

Outra linha de desenvolvimento para aplicações robóticas com controle de força é fornecer o sentido tátil a uma mão robótica, Fig.6, (DLR, 2007). Um dos projetos mais difundidos na literatura é o do Centro de Pesquisas Aeroespacial Alemão, denominado de *DLR Hand*. O projeto foi inteiramente desenvolvido por eles, incluindo um pequeno transdutor de força de seis eixos para o punho que pode ser visualizado na Fig.5.

Figura 5 - Sensor de força produzido para a mão robótica DLR II (DLR, 2007).

Figura 6 - Mão robótica DLR II (DLR, 2007).

O controle de uma mão robótica é complexo, principalmente, pelo fato que ela consiste em uma forma de cooperação robótica, quando vemos os dedos como diferentes sistemas robóticos. Para que o equipamento possa segurar um objeto, o sensor de tato (i.e., a pele robótica) fornece o sentido de força aos dedos inclusos na operação e esse sinal é corrigido até se obter a quantidade de força desejada (DLR, 2007).

1.2 Objetivos

O presente trabalho aborda uma proposta de modelo de referência para o desenvolvimento de controladores de robôs com arquitetura totalmente aberta. O enfoque principal é o desenvolvimento de um padrão de arquitetura funcional em uma estrutura hierárquica, levando em consideração os requisitos e as tendências atuais para o

desenvolvimento de controladores de robôs de arquitetura aberta. Este trabalho pode ser dividido em três grandes etapas:

- *Planejamento*, estudo dos requisitos, definições e categorias de controladores de arquitetura aberta e a apresentação de uma proposta de modelo de referência para o desenvolvimento de controladores de robôs com arquitetura aberta.
- *Desenvolvimento*, a criação integral de um controlador de robôs com arquitetura totalmente aberta, baseado no modelo de referência proposto, enfatizando tarefas de controle de posição e força.
- *Validação e Experimentação*, aplicar o controlador de robôs desenvolvido no manipulador industrial *REIS Rv15* e adicionar um transdutor de força ao sistema, realizando um *retrofitting*[4], para a implementação de uma estrutura de controle de força indireto.

O objeto de trabalho é o manipulador *REIS Rv15* disponível no Laboratório de Robótica (*LAR*) da Universidade Federal de Santa Catarina (*UFSC*). Um manipulador industrial antigo, que já sofreu modificações para retomar seu funcionamento e agora passará por novas atualizações tecnológicas para operar em tarefas de posicionamento e aplicação de forças. Ao se retirar o sistema de controle proprietário, que contém uma série de informações obscuras devido as suas abstrações, possibilita-se a implementação de estruturas de controle avançadas de robôs. Ainda promovendo ao laboratório, o conhecimento da concepção de uma arquitetura de controlador totalmente aberta, composta por componentes (*hardware* e *software*) de alto desempenho, aumentando potencialmente as possibilidades de estudos na área da robótica, com um custo significativamente inferior às soluções disponíveis no mercado.

A concepção de controlador desenvolvida visa cumprir os seguintes requisitos: alta capacidade de processamento, baixo custo, conectividade com outros sistemas, disponibilidade de acesso remoto, facilidade de manutenção, flexibilidade na implementação dos algoritmos, integração com um computador pessoal e programação em alto nível.

---

[4] *Processo de modernização ou atualização tecnológica de um equipamento, com o intuito de incorporar mudanças ou melhorias (SEMATECH, 2007).*

Considerando esses aspectos, este trabalho vem a expor o fluxo de informações, em todos os níveis, de um controlador de robôs com arquitetura aberta, através de uma proposta de modelo de referência. O modelo de referência vai ser aplicado para o desenvolvimento de um controlador de robôs visando à atualização tecnologia do manipulador industrial *REIS Rv15*, para operações em tarefas de controle de posição e força.

1.3 Organização

Nos capítulos 2 e 3 é realizada a fundamentação teórica do sistema, i.e., a etapa de planejamento. Inicialmente tratando dos conceitos relativos ao controle de robôs manipuladores, até a exposição do controle indireto de força. Na seqüência é realizada uma revisão bibliográfica sobre os controladores de robôs, expondo a tendência atual desse modelo de sistema e apresentado um modelo de referência para controladores de robôs. Após, são abordados os controladores de arquitetura aberta, suas definições, categorias, requisitos e arquiteturas. Finalizando é explorada a norma que define a interconexão de sistemas abertos.

Esses temas são utilizados como base teórica para a proposta de modelo de referência para controladores de arquitetura aberta, que é apresentada no capítulo 4, i.e., etapa de desenvolvimento. Esse modelo de referência é aplicado para operar com o controle de posição e força. Na seqüência, são descritas individualmente as camadas que compõe o mesmo. Detalhando os processos que ocorrem na camada de menor abstração (camada de tarefa), onde é desenvolvida a estrutura de controle que rege o manipulador durante a execução da tarefa. Em seguida, é discutida a camada de integração, onde ocorre uma intercomunicação entre os processos ativos, integrando as diversas fontes de informação a um único ambiente alocado na camada superior a esta. A próxima camada, de comunicação, aborda os métodos de comunicação do sistema, explicando a ligação entre os processadores que o compõe e as possibilidades de utilização. Após, é explicada a camada de interface, que abrange os sistemas embarcados que controlam as juntas robóticas. Com um detalhamento necessário das configurações de execução destes componentes, que visão sempre um alto desempenho. O capítulo final tratada da camada física, ou seja, dos componentes de atuação e medição do manipulador robótico. Levantando as informações necessárias para realizar uma tarefa de controle e também gerando uma documentação sobre esse, que até o momento não existia. Completando assim a descrição de todo o sistema controlador proposto.

O capítulo 5 apresenta a validação e experimentação detalhando os procedimentos utilizados para a implementação da estrutura de controle de impedância. Também são apresentados alguns resultados práticos obtidos em experimentos com o controlador.

Finalizando o trabalho são apresentadas as conclusões e perspectivas para pesquisas futuras.

Capítulo 2. Controle de Posição e Força em Robôs Manipuladores

Este capítulo tem por finalidade a apresentação dos principais conceitos teóricos necessários ao desenvolvimento deste trabalho. Inicia pelo controle de robôs até alcançar as leis clássicas de controle de força, focando nos métodos indiretos. Após, são descritos os controladores de robôs com a teoria relacionada. Na seqüência são abordados os controladores de arquitetura aberta, tratando da definição, das categorias, dos requisitos, apresentando algumas arquiteturas e finalizando com a exposição da metodologia para a interconexão de sistemas abertos.

2.1 Controle de Posição

Um controlador de robôs realiza o controle do manipulador através da geração do sinal de controle (i.e., comando dos atuadores) originado da diferença entre a posição desejada (i.e., o comportamento atual da trajetória) e a posição atual do sistema. Dessa forma, são necessárias leituras do estado do sistema. As medidas fundamentais de um sistema robótico são as posições de juntas. Em sistemas de controle, a correção do sinal de controle devido à aquisição do estado do sistema é denominado de controle realimentado.

Porém, existe uma particularidade no controle de robôs manipuladores, ele pode ser realizado em dois espaços distintos: o espaço das juntas e o espaço operacional. O primeiro define um controle relacionado às juntas robóticas, assim, definindo perfis individuais de posição (q), velocidade ($\dot{q}$) e aceleração ($\ddot{q}$), para cada atuador.

No espaço operacional as ações de controle são realizadas em relação à extremidade do manipulador (i.e., efetuador), dessa forma os controladores corrigem os sinais de posição e orientação (i.e., posição angular) (x), velocidade linear e angular ($\dot{x}$), e aceleração linear e angular ($\ddot{x}$). Porém, o acionamento do manipulador sempre acontece no espaço de juntas, pois, os atuadores encontram-se neste, tornando necessária uma transformação que relaciona as variáveis do espaço operacional ($x,\dot{x},\ddot{x}$) às variáveis no espaço das juntas ($q,\dot{q},\ddot{q}$), onde ocorre a entrada de controle.

A transformação entre os espaços operacional e de juntas é denominada de cinemática de robôs. A cinemática direta converte o espaço de juntas em espaço operacional, em outras

palavras, converte ângulos de junta em posição e orientação do efetuador final. A conversão oposta é chamda de cinemática inversa.

A tarefa inicial de um sistema de controle de robôs, em ambos os espaços, é a geração da trajetória das juntas (i.e., o perfil de posições e orientações ao longo do tempo), em seguida, o sinal é repassado ao controlador que aciona os atuadores ou a unidade de potência, que movimentam as transmissões e geram o movimento. Os transdutores do manipulador fornecem as informações sobre o estado do sistema, realimentando o controlador para a correção do sinal de controle, de forma a atingir a posição e orientação desejada para o efetuador final. O diagrama de blocos de um sistema de controle de robôs para ambos os espaços pode ser visualizado na Fig.7.

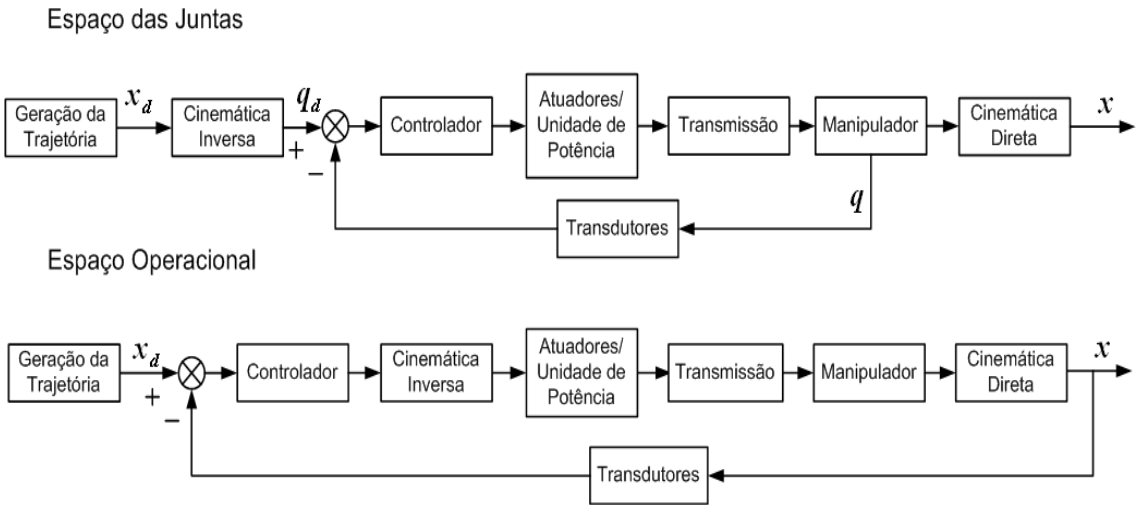

Figura 7 - Controle de robôs.

Em Eppinger, Seering *et al.*, (1992), já se expunha que, a maior parte dos robôs industriais era utilizada unicamente em aplicações de controle de posição onde se empregavam apenas as leituras de posição de junta (ou atuador) como realimentação do sistema. Isso, em geral, limitava as aplicações nas operações onde o manipulador possui pouca ou nenhuma interação com o ambiente. Essas situações não sofreram evolução significativa nos últimos anos conforme pode ser constatado nas referências bibliográficas atuais (Murrugarra, Grieco *et al.*, 2006, Zhu, Zhu *et al.*, 2005, Fan, Peng *et al.*, 2004, etc.) onde a maioria dos problemas tratados concentra-se no controle de posição de manipuladores.

Tarefas de controle de robôs manipuladores para o posicionamento do efetuador, sem contato com o ambiente, são modeladas conforme apresentado por Sciavicco e Siciliano, (2004) e apresentado na equação (2.1):

$$B(q)\ddot{q} + C\left(q,\dot{q}\right)\dot{q} + F_V\dot{q} + g(q) = u \tag{2.1}$$

Onde,

B - Matriz de inércia

C - Matriz de forças e torques centrífugos e Coriolis

F_v - Matriz de atrito viscoso

g - Vetor de termos gravitacionais

u - Entrada de controle

A implementação de estruturas de controle para robôs manipuladores consiste na forma de determinação de u. Existem diversas abordagens sobre o assunto, duas formas clássicas de controle serão apresentadas a seguir para exemplificar como é realizado o controle de posição de robôs, salientando que ambas são detalhadas mais aprofundadamente em Sciavicco e Siciliano, (2004).

Uma das modalidades mais simples de controle de robôs, e também uma forma clássica na teoria de controle linear, é o controle *PD* (i.e., proporcional e derivativo), aqui apresentado com uma compensação perfeita da gravidade, cuja equação é apresentada a seguir, e o diagrama de blocos desta modalidade de controle é apresentado na Fig.8.

$$u = g(q) + K_p\left(q_d - q\right) - K_d\dot{q} \tag{2.2}$$

Onde,

K_p - Ganho proporcional

K_d - Ganho derivativo

q_d - Posição desejada

A equação (2.2) representa o modelo de um robô em malha fechada e as matrizes K_p e K_d devem ser escolhidas para garantir que o sistema seja estável, assegurando que o erro de seguimento do perfil de posição desejada $\tilde{q} = (q_d - q)$ tenda a zero.

Substituindo a lei de controle apresentada na equação (2.2) na equação (2.1), temos a modelagem de um sistema robótico sob a atuação de um controle *PD* com compensação perfeita da gravidade dada por:

$$B(q)\ddot{q} + C\left(q,\dot{q}\right)\dot{q} + F\dot{q} + g(q) = g(q) + K_p\left(\tilde{q}\right) - K_d\dot{q} \tag{2.3}$$

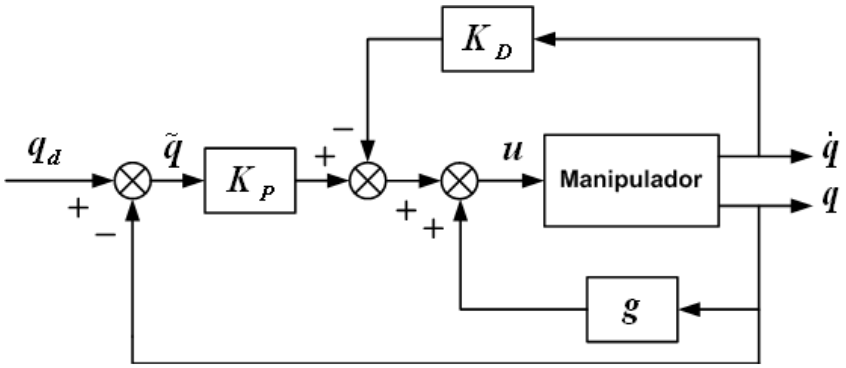

Figura 8 - Diagrama de blocos do controle PD com compensação de gravidade (Sciavicco e Siciliano, 2004).

Outra abordagem clássica para o controle de posição é baseada na definição de uma lei de controle que compense a dinâmica do manipulador resultando em um sistema em malha fechada escrito sob a forma de uma equação diferencial linear em termos dos erros de seguimentos de trajetória. A equação representativa do diagrama de blocos da Fig.9, é dada por:

$$u = B(q)\,y + C\left(q,\dot{q}\right)\dot{q} + F\dot{q} + g(q) \tag{2.4}$$

Ao substituir a lei de controle do sistema apresentado na equação (2.4) na equação (2.1), obtém-se um modelo de controle para um manipulador robótico sob a ação do controle por dinâmica inversa, com a compensação integral da dinâmica, dado por:

$$B(q)\ddot{q} + C\left(q,\dot{q}\right)\dot{q} + F\dot{q} + g(q) = B(q)\,y + C\left(q,\dot{q}\right)\dot{q} + F\dot{q} + g(q) \tag{2.5}$$

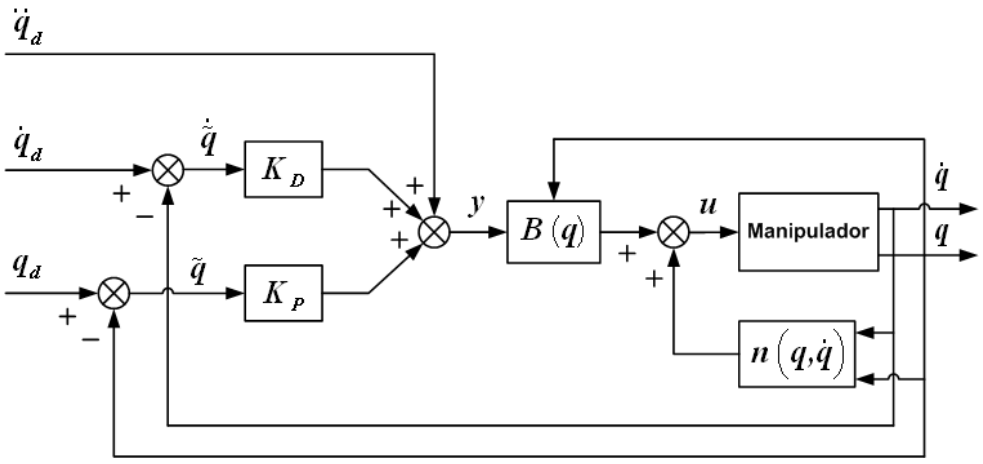

Figura 9 - Diagrama de blocos do controle por dinâmica inversa (Sciavicco e Siciliano, 2004).

Ou equivalentemente,

$$\ddot{q} = y \tag{2.6}$$

Onde, y é uma nova lei de controle necessária para impor o desempenho desejado para o sistema. Uma possibilidade de escolha para y é definir uma estratégia de controle do tipo *PD* (Proporcional Derivativa) dada por:

$$y = \ddot{q}_d + K_d\left(\dot{q}_d - \dot{q}\right) + K_p\left(q_d - q\right) = \ddot{q} + K_d\dot{\tilde{q}} + K_p\tilde{q} \tag{2.7}$$

Onde,

$\ddot{q}_d$ - Aceleração desejada

$\dot{q}_d$ - Velocidade desejada

q_d - Posição desejada

$\tilde{q}$, $\dot{\tilde{q}}$, $\ddot{\tilde{q}}$ - Definem os erros de posição, velocidade e acelereação respectivamente.

A aplicação desta lei de controle na equação (2.6) fornece:

$$\ddot{q} - q_d + K_d\left(\dot{q} - \dot{q}_d\right) - K_p\left(q - q_d\right) = 0 \tag{2.8}$$

Portante é possível encontrar os ganhos K_p e K_d tal que a equação (2.8) seja assintóticamente estável assegurando que $q \to q_d$ quando $t \to \infty$.

A abordagem de tarefas de controle de robôs, onde existe interação do manipulador com o ambiente, é uma das linhas de pesquisa deste trabalho, por isso, algumas das estratégias de controle de força clássicas serão detalhadas mais profundamente a seguir.

2.2 Controle de Força

Eppinger, Seering *et al.*, (1992), descrevem que em vários processos automáticos de manufatura é necessário que o robô realize uma interação com um ambiente desconhecido de maneira controlada. Essas operações requerem a adição de novos sistemas de sensoriamento em robôs industriais, como transdutores de força e momentos, que permitem o emprego de estruturas de controle de posição e força. O termo "controle de força", na robótica industrial, descreve estratégias de controle onde as medidas das forças de interação são utilizadas como realimentação ao sistema de controle, influenciando diretamente na atuação das juntas robóticas.

Tarefas que contenham interação do manipulador com o ambiente, necessitam da adição de uma realimentação de força no sistema. Assim, modificando a equação (2.1) obtém-se o modelo de um sistema robótico para tarefas de interação com ambiente, dada por:

$$B(q)\ddot{q} + C\left(q,\dot{q}\right)\dot{q} + F\dot{q} + g(q) = u - J^{T}(q)\,h \tag{2.9}$$

Onde,

h - Forças externas exercidas sobre o manipulador

J - Jacobiano Geométrico[5]

Independentemente da tarefa realizada, o problema de controle de robôs envolve sempre o cálculo dos sinais para os atuadores seguirem a trajetória desejada (i.e. pode ser composta por perfis de: posição, velocidade, aceleração, força, etc.). A precisão da geração destes sinais está relacionada com a modelagem do sistema. Em Miljanovic e Croft, (1999), é salientado que, o desenvolvimento de um sistema de controle de posição e força necessita da

[5] *Descreve o relacionamento entre as velocidades das juntas e as velocidades lineares e angulares do efetuador. Uma descrição mais aprofundada pode ser vista em Sciavicco e Siciliano, (2004). Em Leite, (2003), é apresentada a aplicação deste para o manipulador REIS Rv15.*

modelagem matemática do robô e do ambiente, implícita ou explicitamente. As características e incertezas associadas aos modelos definem a seleção da arquitetura e do algoritmo de controle. A análise e projeto de leis de controle para robôs necessitam de um desenvolvimento eficiente da equação dinâmica do sistema. Assim, uma análise das características da estrutura mecânica, sensores e atuadores, contribuem para um comportamento adequado do efetuador sobre os perfis de trajetória desejada.

Em tarefas onde há contato, a descrição das características do ambiente é essencial para um controle adequado. O ambiente pode ser classificado como: inercial (empurrar), resistivo (deslizar, polir e furar), complacente (efeito mola, elástico) ou rígido. Cada categoria tem uma arquitetura de controle que melhor se enquadra. O maior problema no controle de posição e força em tarefas de interação de robôs com um ambiente dinâmico é manter a estabilidade sobre o movimento desejado e as forças de interação (Vukobratovic, 1997).

Incertezas associadas ao modelo dinâmico do robô ou do ambiente podem degradar o desempenho da tarefa (Vukobratovic, 1997). O principal problema da síntese da estratégia de controle em tarefas de contato está na representação das incertezas no modelo dinâmico do ambiente. Essas incertezas vêm geralmente, da dificuldade na identificação e previsão dos parâmetros dos modelos e dos comportamentos do ambiente. É importante que a estratégia de controle seja robusta à ação das incertezas, para que estas não comprometam o cumprimento da tarefa (Miljanovic e Croft, 1999).

O controlador necessita de um sinal de referência que seja fornecido em intervalos de tempo regulares e deve receber todas as demais informações necessárias dentro de um intervalo de tempo limitado, para o seu correto funcionamento. A existência de eventos imprevisíveis ou incertezas pode resultar no não recebimento das informações no intervalo de tempo apropriado. Cada evento imprevisível pode incluir uma perda ou atraso no sinal de referência, no estado da informação, no sinal de perturbação ou na informação proveniente do sensor. A disponibilidade e a validade das informações provenientes dos sensores são fatores decisivos na escolha da lei de controle (Miljanovic e Croft, 1999). Em tarefas em que o manipulador possui interação com o ambiente, a confiabilidade dessas informações é um requisito para o correto funcionamento do sistema. Essa característica será aprofundada a seguir.

2.2.1 *Requisito especial para controladores robóticos que incluem o controle de força*

A maioria dos manipuladores robóticos é desenvolvida para realizar tarefas de posicionamento, não prevendo o cumprimento dos requisitos de tarefas onde é necessário o controle de força. Em aplicações que necessitam do controle de força, em algum momento da trajetória, o efetuador realizará o contato com uma superfície em seu espaço de trabalho. Esta interação gerará forças de contato, que devem ser controladas de maneira a cumprir corretamente a tarefa, sem causar danos à ferramenta, ao manipulador e ao objeto trabalhado. As intensidades das forças de contato, originadas pelos movimentos da ferramenta, comandados pelo controlador robótico, depende da rigidez da ferramenta e da rigidez da superfície, e estas, também 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 rígidos. A utilização de manipuladores extremamente rígidos, o caso dos manipuladores para tarefas de posicionamento, incrementa a intensidade dessas forças. Isso requer um sistema com um pequeno tempo de resposta a estas forças, para prevenir danos.

Uma alternativa a esse efeito é iniciar a tarefa após o contato do manipulador com a superfície, que geralmente é realizado manualmente com a menor velocidade possível. Porém algumas tarefas não suportam essa postura, neste caso alguma forma de flexibilidade (no caso de manipuladores é denominada de complacência) deve ser inserida ao sistema. A *PushCorp Inc.* desenvolve algumas ferramentas (Fig.10) para processos que necessitam de uma complacência com precisão e uma força aplicada que seja continuamente ajustável. Assim, a inclusão da ferramenta, adiciona uma complacência ao sistema, diminuindo as intensidades das forças no momento de contato e fornecendo um intervalo de tempo maior para a resposta a estas forças.

A seleção da taxa de amostragem (i.e., o intervalo de aquisição) das informações para o sistema controlador é um comprometimento de vários fatores. Em tarefas de contato é complicado se determinar a taxa de amostragem apropriada devido ao seu acoplamento com o ambiente, perturbações externas, etc. Uma alta taxa de amostragem é importante para um ambiente altamente rígido devido ao efeito causado pelas perturbações externas ao sistema (Miljanovic e Croft, 1999).

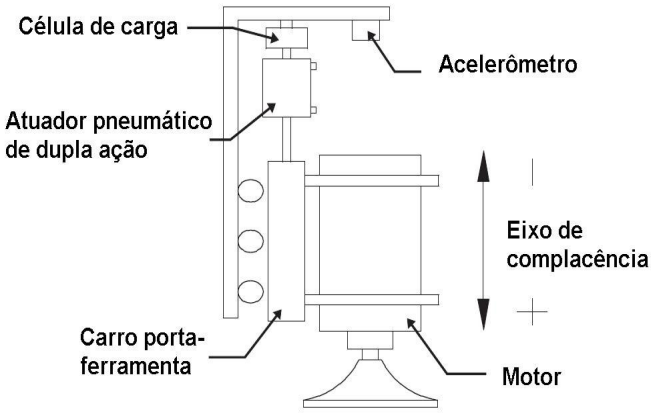

Figura 10 - Ferramenta para o controle ativo de força (Erlbacher, 2000).

O uso de sistemas de alto desempenho é um requisito para controladores que aplicam o controle de força devido, principalmente, à quantidade de processamento matemático que deve ser realizado em um curto espaço de tempo. Algumas partes necessitam de prioridades especiais de processamento. Esses requisitos reforçam a utilização de controladores de arquitetura aberta, onde é possível a utilização de sistemas operacionais adaptados a cumprir tarefas em tempo real[6] de aplicação e possibilitando o gerenciamento aprofundado dos processos que compõem o sistema de controle.

Recentemente, foi divulgado o projeto *ADVOCUT*, que consiste em uma parceria de diversas empresas especializadas em áreas distintas, porém que em conjunto operam nos principais ramos de desenvolvimento robótico, para a adaptação completa de um robô industrial para operações de fresamento. Traduzindo o nome do projeto temos: máquina-ferramenta adaptativa com módulo de fresamento mecatrônico altamente integrado para usinagem em alta velocidade (Abele, Weigold *et al.*, 2007).

Nesse projeto, foram realizadas modificações sobre um manipulador robótico produzido pela *REIS Robotics* modelo *RV130*. Primeiramente, para aumentar a precisão absoluta de deslocamento, são acoplados sensores de deslocamento diretamente ao lado do atuador. Visto que, robôs convencionais realizam leituras indiretas dos ângulos de junta, pois, possuem seus transdutores angulares do lado oposto ao atuador, utilizando a transmissão

---

[6] *Refere-se à interação simultânea das informações, o processamento não pode exceder o tempo máximo (i.e., o intervalo entre a chegada de duas amostras), o não cumprimento implicará em conseqüências graves, como a perda de informações, que pode acarretar em danos ao sistema.*

como um fator de multiplicação dos pulsos de medição e conseqüentemente de precisão. Assim, necessitando de sensores menos precisos e de menor custo. Essa medida, em tarefas de usinagem, causa um erro devido à elasticidade da transmissão. Com leituras diretas e indiretas obtêm-se o fator de erro devido a esses aspectos, para uma posterior correção da posição absoluta do manipulador (Abele, Weigold *et al.*, 2007). A Figura 11 apresenta o projeto, com o manipulador em funcionamento.

A transformação mais expressiva encontra-se na extremidade do manipulador. Seu punho foi modificado para poder manipular ferramentas longas, necessárias para cumprir os requisitos das tarefas. Esta possui em uma de suas extremidades seu atuador, que em um punho normal limitaria muito a amplitude de movimentos do efetuador final (Abele, Weigold *et al.*, 2007).

O projeto foi realizado visando tarefas de limpeza e rebarbação de materiais fundidos, operações que necessitam de baixas forças para sua atuação. As flexibilidades do manipulador em tarefas de furação e serramento levam a erros considerados intoleráveis pelas aplicações. Esses são minimizados com a utilização de apoios para as ferramentas. Porém ainda em Abele, Weigold *et al.*, (2007), é dito que esses fatores podem ser otimizados com melhorias de *software* e *hardware*. Ressaltam também, que ao serrar geralmente ocorre a utilização de serras de disco de grandes dimensões, que geram grandes forças centrífugas que precisam ser consideradas nas operações de controle.

Figura 11 - Projeto ADVOCUT (ADVOCUT Project. , 2007).

As estratégias de controle de robôs em tarefas de interação são usualmente agrupadas em duas categorias: controle de força indireto e controle de força direto. A primeira forma de controle trata o movimento com uma realimentação implícita de força, baseando-se no controle do movimento, a outra fornece a possibilidade do controle de força para um valor desejado, através de uma realimentação explícita de força (Sciavicco e Siciliano, 2004). Abordaremos a primeira categoria de controle de força, tratando a seguir o controle de impedância e o controle de rigidez.

2.2.2 *Controle de Impedância*

O controle por impedância trata indiretamente a força de contato, modelando a interação como um conjunto massa-mola-amortecedor. O relacionamento indireto ocorre pela forma de controle do sistema sobre o sinal de força, cujo objetivo é adequar o comportamento dinâmico do manipulador em contato com o ambiente e não manter uma trajetória de posição e/ou força. Desta forma não existe uma malha explícita de controle de força no sistema, pois a realimentação de força fornece apenas a impedância do sistema em contato com uma superfície (Fig.12). Assim não existe especificação de força de aplicação desejada, mas sim, um perfil de dinâmica desejada de interação entre o manipulador e a superfície (Zeng e Hemami, 1997).

A filosofia fundamental do controle de impedância, de acordo com Hogan, (1985), é que o sistema de controle do manipulador não é projetado unicamente para seguir uma trajetória, mas para regular a impedância do manipulador. Define-se como impedância mecânica a relação entre a velocidade e a força aplicada (Zeng e Hemami, 1997).

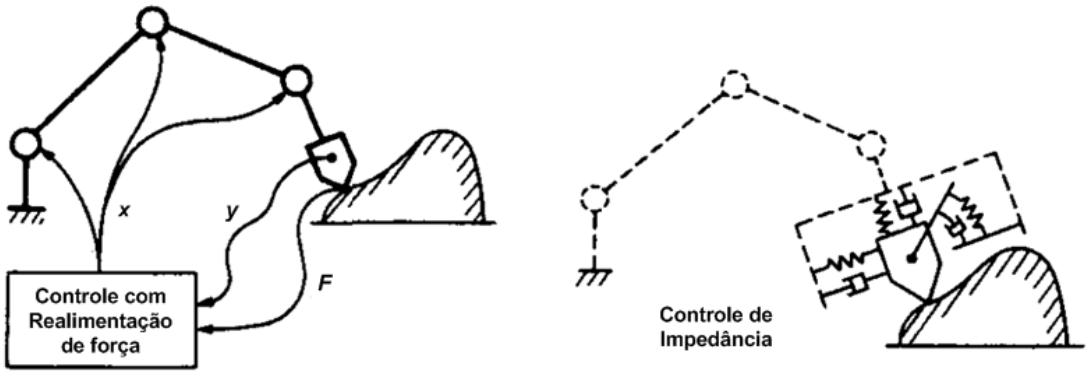

Figura 12 - Sistemas de controle de força (Yoshikawa, 2000).

O controle por impedância pode ser implementado de diversas maneiras, de acordo com a forma em que os sinais (i.e., posição, velocidade e força) são utilizados. E dependendo também da origem dos sinais, se a aquisição dessas grandezas é realizada por meio de medição ou estimativa dos parâmetros (Golin, 2002). A seguir, é apresentada a dedução matemática da lei de controle de impedância baseada em Sciavicco e Siciliano, (2004).

O modelo dinâmico de um manipulador robótico para tarefas de interação com o ambiente é definido na equação (2.9), admitindo que $n\left(q,\dot{q}\right)=C\left(q,\dot{q}\right)\dot{q}+F\dot{q}+g(q)$, a equação pode ser reescrita como:

$$B(q)\ddot{q}+n\left(q,\dot{q}\right)=u-J^{T}(q)\,h \tag{2.10}$$

Seguindo o mesmo critério, a lei de controle da dinâmica inversa (Fig.9) é dada por:

$$u=B(q)\,y+n\left(q,\dot{q}\right) \tag{2.11}$$

Onde, y - Vetor de entradas, i.e., a lei de controle

Substituindo a equação (2.11) na (2.10), o resultado é dado pela equação (2.12), onde é descrito um manipulador controlado sobre o efeito de forças de contato. Nesta equação pode ser visualizado um acoplamento não-linear devido às forças de contato.

$$\ddot{q}=y-B^{-1}(q)\,J^{T}(q)\,h \tag{2.12}$$

Sugerindo, para um manipulador não redundante, a lei de controle para o espaço operacional demonstrada na equação (2.13). Onde M_d, K_d e K_p são matrizes (diagonais) definidas positivas.

$$y=J_A^{-1}(q)\,M_d^{-1}\left(M_d\ddot{x}_d+K_d\dot{\tilde{x}}+K_P\tilde{x}-M_d\dot{J}_A\left(q,\dot{q}\right)\dot{q}\right) \tag{2.13}$$

Onde, J_A é o Jacobiano Analítico[7].

Substituindo a equação (2.13) na (2.12), obtém-se:

$$M_d \ddot{\tilde{x}} + K_d \dot{\tilde{x}} + K_P \tilde{x} = M_d J_A(q) B^{-1}(q) J_A^T h_A \tag{2.14}$$

Onde, h_A é o vetor de forças equivalentes generalizadas

Na equação (2.14) é possível visualizar a matriz de inércia no espaço operacional:

$$B_A^{-1}(q) = J_A(q) B^{-1}(q) J_A^T \quad \text{ou} \quad B_A(q) = J_A^{-T} B(q) J_A^{-1}(q) \tag{2.15}$$

Assim, a equação (2.14) pode ser reescrita como:

$$M_d \ddot{\tilde{x}} + K_d \dot{\tilde{x}} + K_P \tilde{x} = M_d B_A^{-1}(q) h_A \tag{2.16}$$

Na equação (2.16) é possível visualizar a relação entre o vetor de forças resultantes aplicadas ao efetuador do manipulador $M_d B_A^{-1}(q) h_A$ e o vetor de deslocamentos do manipulador no espaço operacional $\tilde{x}$, i.e., a impedância mecânica do manipulador. Caracterizada pela matriz de massa M_d, pela matriz de amortecimento K_d e pela massa de rigidez K_p. A presença de $B_a^{-1}(q)$ torna o sistema acoplado. Para tornar o sistema desacoplado e linear durante a interação com o ambiente torna-se necessária a leitura das forças de contato h, o que nos permite empregar a seguinte lei de controle:

$$u = B(q)\, y + n\left(q, \dot{q}\right) + J^T(q)\, h \tag{2.17}$$

Substituindo a equação (2.17) na (2.10) e empregando a equação (2.13) temos:

$$M_d \ddot{\tilde{x}} + K_d \dot{\tilde{x}} + K_P \tilde{x} = 0 \tag{2.18}$$

A equação (2.18) é resultado da compensação perfeita da força, que torna o sistema em malha fechada insensível à força aplicada, i.e., o robô torna-se totalmente rígido em relação à ação de forças. Para permitir um comportamento complacente ao robô é introduzido o termo $K_h = -J_A^{-1}(q) M_d^{-1} h_A$ na equação (2.13), o que resulta em:

---

[7] *Descreve a localização do efetuador final através da representação mínima do espaço operacional e é calculado pela diferenciação da cinemática direta em relação às variáveis de junta. Uma descrição mais aprofundada pode ser vista em Sciavicco e Siciliano, (2004).*

$$y = J_A^{-1}(q) M_d^{-1}\left(M_d \ddot{x}_d + K_d \dot{\tilde{x}} + K_P \tilde{x} - M_d \dot{J}_A\left(q,\dot{q}\right)\dot{q} - h_A \right) \tag{2.19}$$

Assim, obtém-se a equação do controle de impedância:

$$M_d \ddot{\tilde{x}} + K_d \dot{\tilde{x}} + K_P \tilde{x} = h_A \tag{2.20}$$

O digrama de blocos do sistema de controle de impedância, equação (2.20), é apresentado na Fig.13.

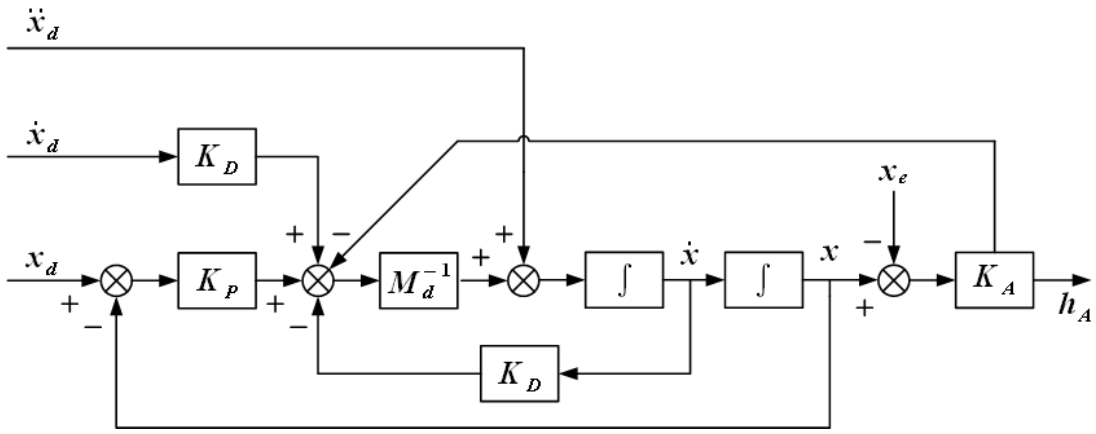

Figura 13 - Diagrama de blocos de um manipulador em contato com um ambiente elástico sobre o efeito do controle de impedância (Sciavicco e Siciliano, 2004).

Da mesma forma que o controle de posição por dinâmica inversa, no qual o controle por impedância se baseia, é admitido o conhecimento integral da dinâmica do manipulador o que anula essa grandeza, devido a sua ocorrência na dinâmica do manipulador e na lei de controle, conforme pode ser visto na equação (2.20). Yoshikawa, (2000) ressalta que o conhecimento preciso das características de elasticidade do objeto ou do ambiente de contato não são necessárias nesta modalidade de controle.

2.2.3 *Controle de Rigidez*

O controle de rigidez realiza um controle indireto da força, modelando a interação do manipulador sobre a superfície similarmente ao efeito de uma mola, que exerce uma força sobre o ambiente. A rigidez da mola determina o comportamento dinâmico do contato entre o efetuador final e o meio. Caso haja medição da força de contato, a rigidez pode ser ajustada com uma realimentação de força para se adequar a tarefas em diferentes ambientes. A rigidez do manipulador é dada por seus ganhos, os quais podem ser designados independentemente

para cada direção, determinando diferentes relações entre a rigidez do meio e a do manipulador (Sciavicco e Siciliano, 2004).

Golin, (2002), ressalta que o controle de rigidez enquadra-se como um caso particular do controle de impedância, onde o sistema massa-mola idealizado é considerado como uma impedância simplificada. Dessa forma, desprezando a realimentação de aceleração e velocidade da equação (2.20), ou seja, simplificando a impedância, tem-se:

$$K_P \tilde{x} = h_A \tag{2.21}$$

O vetor de forças equivalentes generalizadas h_A pode ser reescrito em função do vetor de forças de contato exercidas pelo efetuador final sobre o ambiente h pela relação:

$$h_A = T_A^T(x)\, h \tag{2.22}$$

Onde,

T_A - Matriz de transformação

O modelo de forças no espaço operacional é dado por:

$$h = KT_A(x)\, dx \tag{2.23}$$

Onde,

dx - Deslocamento do meio

O deslocamento do meio (dx) ocorre quando o efetuador final está em contato com um ambiente flexível e é caracterizado pela diferença entre a deformação do meio que se iguala a posição do efetuador final (x) em relação à posição do meio sem deformação (x_e), i.e., sem o contato do efetuador, as quais podem ser vistas na Fig. 14. Ou seja:

$$dx = x - x_e \tag{2.24}$$

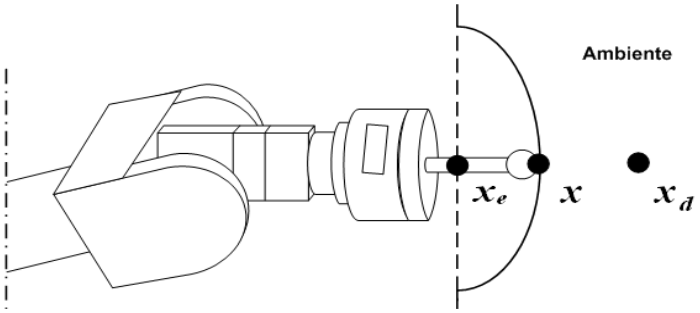

Figura 14 - Efetuador final em contato com um ambiente flexível.

Substituindo a equação (2.24) na (2.23) e o resultado na equação (2.22), obtém-se:

$$h_a = T_A^T(x)\, K T_A(x)\, dx = K_A(x)(x - x_e) \tag{2.25}$$

Substituindo a equação (2.25) na (2.21) e rearranjando, é obtido o controle de rigidez, dado por:

$$\tilde{x} = K_p^{-1} K_A(x)(x - x_e) \tag{2.26}$$

Após abordar as formas de controle indireto de força serão explorados os conceitos teóricos inerentes aos controladores de robôs, a parte do sistema de controle que armazenará e processará a estratégia de controle.

Capítulo 3. Controladores de Robôs Manipuladores

Os controladores de robôs consistem em toda estrutura de *software* e *hardware* responsável pelo gerenciamento do sistema robótico. Sua função é processar a estratégia de controle e reger corretamente o manipulador. Sciavicco e Siciliano, (2004), definem que um sistema de controle supervisiona as atividades do sistema robótico e fornece as seguintes funcionalidades:

- *Movimentação*: capacidade de movimentar fisicamente objetos no ambiente de trabalho.
- *Sensoriamento*: capacidade de obter informações do estado do sistema e do ambiente de trabalho.
- *Comportamento inteligente*: capacidade de explorar as informações do sistema, para modificar o comportamento de uma maneira pré-determinada.
- *Processamento de dados*: elaborar, fornecer e armazenar os dados sobre as atividades do sistema.

Os projetos de sistemas controladores de robôs, comumente vêm seguindo a tendência de basear-se em computadores pessoais. Essa questão será explorada a seguir.

3.1.1 *Tendência*

Antigamente, os sistemas de controle eram heterogêneos (i.e., eram compostos por: equipamentos, fabricantes e formas de interconexão diferenciadas) e orientados a *hardwares* proprietários. A implementação de estratégias de controle era realizada via *hardware*, os parâmetros (i.e., ganhos) eram modificados através de dispositivos mecânicos e analógicos, e as alterações na lei de controle acarretavam na substituição do *hardware*.

Atualmente, a maioria dos controladores robóticos possui sua arquitetura baseada no padrão de *software* e *hardware* para computadores pessoais (*PC*), fornecendo homogeneidade ao sistema e a possibilidade de operação com sistemas operacionais convencionais. Esta postura incrementa potencialmente a flexibilidade do sistema facilitando a interconexão com

periféricos e fornecendo uma estrutura de controle totalmente orientada a *software*, simplificando as alterações.

Os periféricos (i.e, dispositivos auxiliares) para robôs têm um custo superior em relação às similares para *PC*. A falta de padronização dos periféricos específicos para robôs faz com que cada fabricante desenvolva seus próprios padrões e protocolos, "forçando" seus usuários a adquirir componentes de um único fabricante (Lages, Henriques *et al.*, 2003). Em um controlador baseado em *PC* pode ser mais fácil integrar periféricos adicionais. A facilidade de adicionar novas funcionalidades é uma forte razão para usar *hardwares* de *PC* nas arquiteturas abertas de controladores robóticos. Prischow, (2001), ilustra a utilização de sistemas de controle baseados em *PC,* confrontando o passado e o presente/futuro, conforme pode ser visualizado na Fig.15, uma situação que condiz totalmente com a tendência atual da utilização de *PC* em sistemas de controle para robôs.

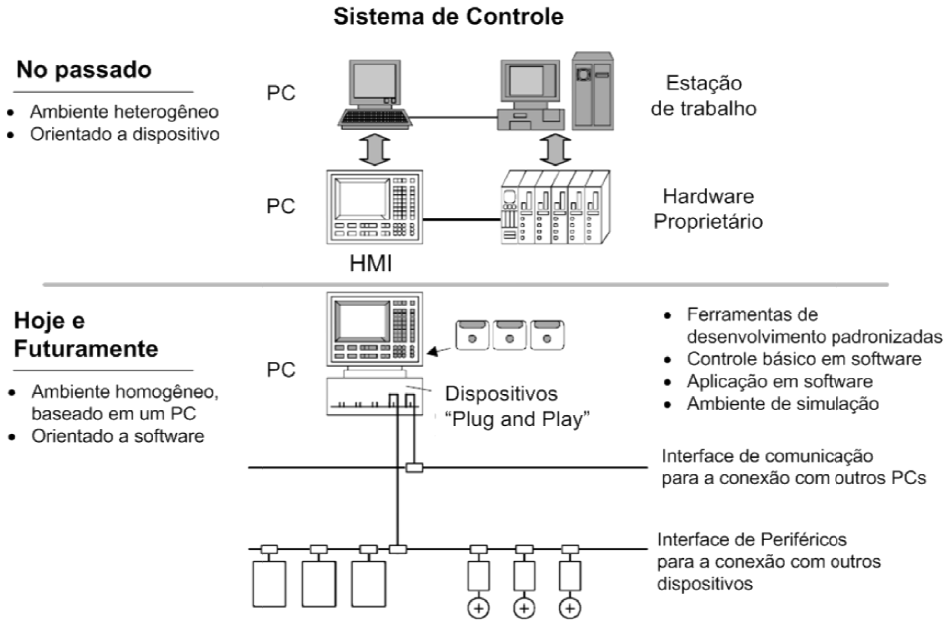

Figura 15 - Sistemas de Controle (Prischow, 2001).

Outra razão para a utilização deste tipo de controlador é o fato das linguagens de programação de robôs serem de baixo nível, mais similares a linguagens de máquina *Assembly* do que as modernas linguagens de alto nível (i.e, alta abstração e reutilização), o que introduz dificuldades nas implementações das soluções e aumenta o tempo de desenvolvimento do *software* (Lages, Henriques *et al.*, 2003). Em controladores baseados no

padrão de *PC*, ferramentas de desenvolvimento de *software* convencionais podem ser utilizadas (e.g., *Visual C++*, *Visual Basic* ou *Delphi*).

3.1.2 *Modelo de Referência para Controladores de Robôs*

Sciavicco e Siciliano, (2004), definem um modelo de referência para a arquitetura funcional de um controlador, com quatro níveis hierárquicos potencialmente relevantes para os sistemas de controle de robôs em aplicações industriais, representado na Fig.16.

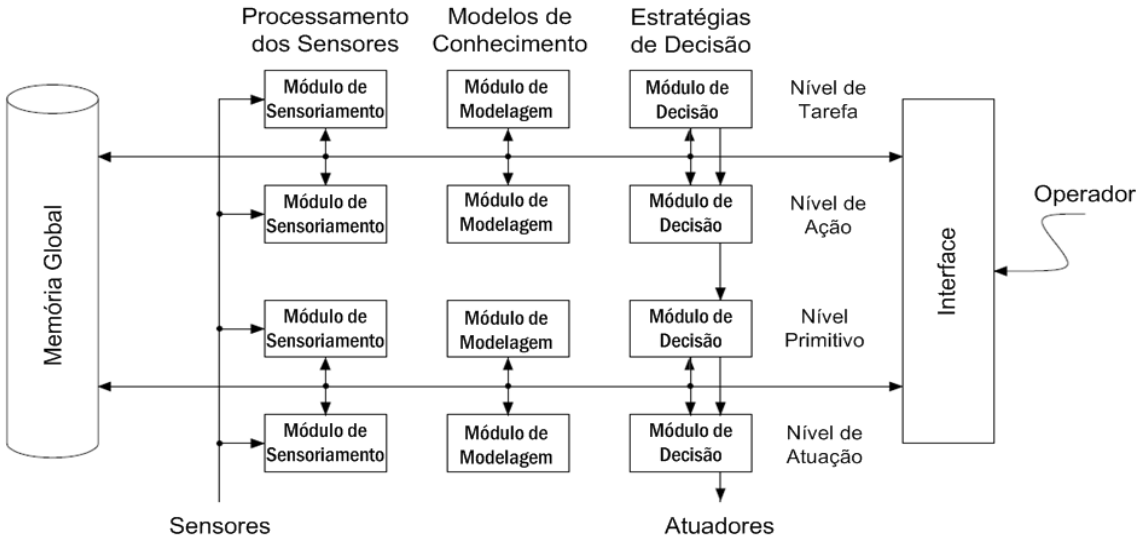

Figura 16 - Modelo de referência para uma arquitetura funcional do sistema de controle (Sciavicco e Siciliano, 2004).

Os *níveis hierárquicos* definem a tarefa, a decompõem em tarefas elementares, atribuem as primitivas das ações e implementam as ações de controle dos atuadores (Sciavicco e Siciliano, 2004). Os níveis serão descritos a seguir.

No *nível de tarefa*, o usuário especifica a tarefa que o robô executará, em um alto nível de abstração. A tarefa é analisada e decomposta em uma seqüência de ações coordenadas no espaço e no tempo.

No *nível de ação*, os comandos simbólicos provenientes da camada superior são convertidos em uma seqüência intermediária de configurações, que caracterizam o movimento para cada ação elementar. A escolha destas seqüências é realizada com base nos modelos do manipulador e do ambiente onde a ação será realizada. A viabilidade da ação é verificada em termos de colisão com obstáculos, movimento próximo a região de configurações singulares,

os limites mecânicos das juntas e eventualmente da utilização dos graus de mobilidade redundantes disponíveis.

No *nível primitivo*, a seqüência recebida pela camada superior é utilizada para calcular a trajetória de movimento e a estratégia de controle é decidida. A trajetória é interpolada para gerar as referências para o nível de atuação. A escolha do movimento e das primitivas de controle está condicionada às características mecânicas e aos graus de interação com o ambiente.

No *nível de atuação*, os algoritmos de controle são implementados para fornecer os sinais que regem os atuadores das juntas. O algoritmo de controle opera sobre o erro da referência sobre o valor medido. Uma microinterpolação é realizada na trajetória de referência de forma a explorar as características dinâmicas dos componentes de atuação (i.e., unidades de potência e servomotores), então é calculada a lei de controle para gerar os sinais de controle (i.e., tensão e corrente elétrica) para o sistema de atuação. É realizada a leitura dos sensores proprioceptivos e estas grandezas são utilizadas para a realimentação e cálculo dos erros do movimento.

Os níveis possuem diferentes taxas de tempo relativas à suas complexidades e requisitos. Funções associadas aos níveis mais altos não demandam um processamento em tempo real, pois estão relacionados com a etapa de planejamento. Nos níveis mais baixos é um requisito as operações ocorrerem em tempo real de aplicação, visando obter um alto desempenho dinâmico da estrutura mecânica. Sciavicco e Siciliano, (2004), ainda ressaltam que no nível de atuação, onde são gerados os sinais para os atuadores e a leitura dos sensores proprioceptivos, as operações possuem taxas de amostragem na escala de milisegundos.

As camadas de níveis mais altos são orientadas à lógica do planejamento das ações enquanto as mais baixas são orientadas para a execução do movimento físico. O tráfego da informação das camadas mais abstratas para as mais especializadas se dá pela decomposição dupla da tarefa em ações de baixo nível. Ocorre a decomposição no tempo em ações seqüenciais e no espaço em ações concorrentes (Sciavicco e Siciliano, 2004). Cada nível ainda é composto por três módulos, com funções distintas, descritos a seguir:

Os *módulos de sensoriamento* adquirem, elaboram, correlacionam e integram os dados provenientes dos sensores em relação a tempo e espaço, medindo o estado do sistema e as características do ambiente.

Os *módulos de modelagem* contêm os modelos de conhecimento do sistema e do ambiente, os quais são atualizados pelas informações provenientes dos módulos de sensoriamento.

Os *módulos de decisão* realizam a decomposição de tarefas de alto nível em tarefas de menor nível de abstração. A decomposição é realizada em relação ao tempo, em ações seqüenciais e em relação ao espaço em ações concorrentes.

Atualmente, os sistemas de controle de robôs não são dotados de todas as funções apresentadas no modelo de referencia da arquitetura funcional, por limitações tecnológicas e custos. Nesses, o nível de tarefa não é utilizado, pois não existem pacotes de aplicações confiáveis e eficazes que dêem suporte as funções complexas requeridas neste nível. Os níveis funcionais do modelo de referência são tipicamente utilizados em sistemas de controle avançados para robôs industriais (Sciavicco e Siciliano, 2004), conforme pode ser ilustrado na Fig.17.

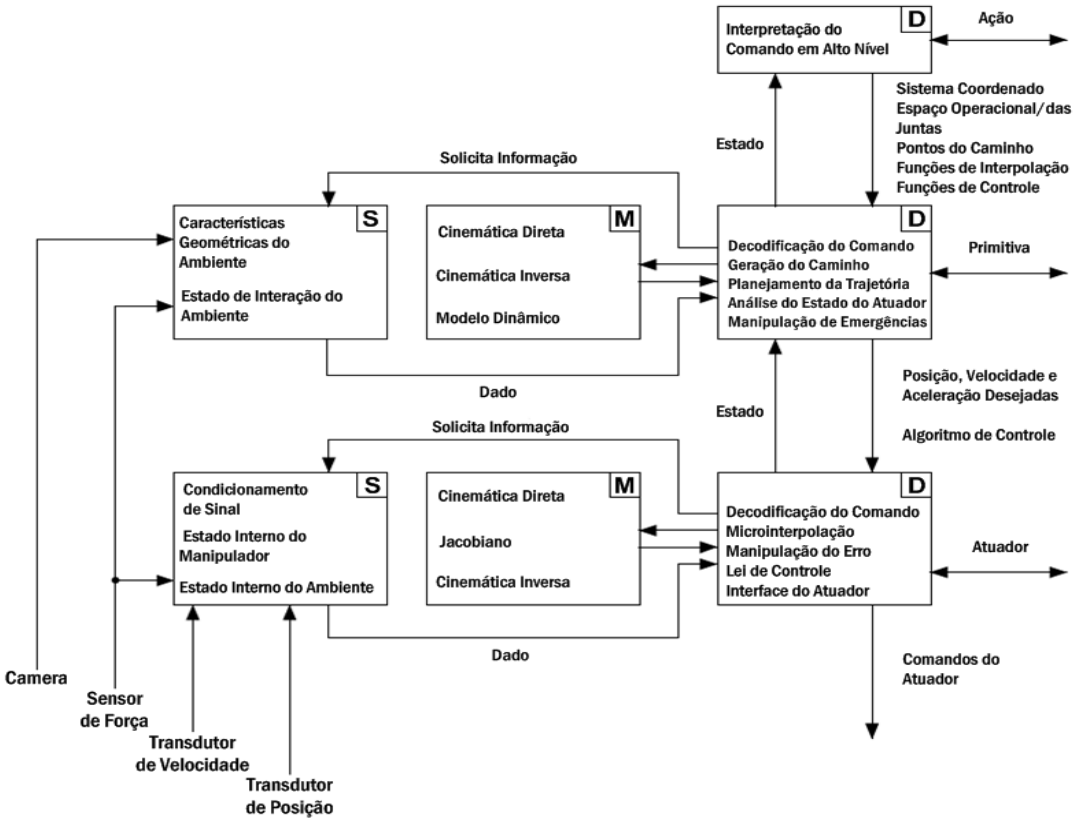

Figura 17 - Arquitetura funcional em níveis hierárquicos para robôs industriais (Sciavicco e Siciliano, 2004).

Os tópicos seguintes abordarão os fundamentos teóricos referentes aos controladores de arquitetura aberta. Serão apresentadas as definições, as categorias, os requisitos, algumas

arquiteturas desenvolvidas atualmente e a apresentação da que define a interconexão entre sistemas abertos.

3.2 Controladores de Arquitetura Aberta

A tendência de utilização de sistemas de controles com arquitetura aberta é tão atual que empresas que fabricantes de robôs manipuladores estão divulgando informações, que anteriormente eram sigilosas, para aumentar o acesso a informações do controlador. Mustapic, Andersson *et al.*, (2003) apresentam a experiência de "abertura" dos controladores da *ABB Robotics*, justificando que essa ação ocorreu no intuito de que parceiros venham a criar funcionalidades que a empresa não tenha priorizado desenvolver. Em plataformas fechadas, a organização desenvolvedora é responsável por toda produção, o que é um fator limitante. Os autores ressaltam que no futuro, a velocidade de desenvolvimento será incrementada pelo grau de "abertura" de acesso do usuário ao sistema pela organização desenvolvedora. O número de novas aplicações para robôs aumentará quando as empresas adaptarem os manipuladores para tipos específicos de aplicações e de clientes.

O sistema de controle de robôs *ABB Robot Controller* consiste em três computadores altamente interconectados: o computador principal, que gera a trajetória, o computador eixo, responsável pelo controle das juntas robóticas, e o computador *I/O* (i.e., entradas e saídas), que interage com sensores e atuadores externos. Apenas o computador principal é aberto para o acesso pelo usuário final e sua programação é realizada por uma linguagem própria, denominada de *RAPID*. A programação pode ser realizada *off-line* (i.e., fora de operação) pelo computador pessoal ou *on-line* (i.e., em operação) pelo *Teach Pendant* (Mustapic, Andersson *et al.*, 2003). Ainda é realizada uma comparação dos métodos de integração de aplicações *.Net Framework* com o *Program Server Subsystem* da *ABB Robotics*. Ilustrando a "pseudo-abertura" lógica existente nos controladores *ABB*, que está relacionada apenas com a possibilidade de comunicação entre aplicativos, porém é um diferencial significativo em relação ao acesso de informações realizado anteriormente a utilização desses métodos.

Na seqüência será descrita a definição dos controladores de arquitetura aberta, suas categorias, seus requisitos, algumas arquiteturas e será apresentada a norma que define a interconexão de sistemas abertos.

3.2.1 *Definição*

O Comitê Técnico de Sistemas Abertos do Instituto de Engenheiros Eletricistas e Eletrônicos (*IEEE*) estabeleceu que: "Um sistema aberto promove capacidades que possibilitam a implementação de aplicações que sejam executadas em plataformas de diferentes fabricantes, interoperam com outras aplicações do sistema e apresentam um estilo consistente de interação com o usuário" (Prischow, 2001). Um sistema com arquitetura de controle aberta tem a capacidade de operar conjuntamente com os melhores componentes de diferentes fabricantes. Uma característica que acarreta uma fácil integração de novas funcionalidades ao sistema.

Do ponto de vista do usuário a "abertura" de um sistema condiz com capacidades de integrar, estender e reutilizar os módulos que compõem o *software* de controle (Sperling e Lutz, 1997). Em Prischow, (2001), o "grau de abertura" de um sistema é definido por alguns critérios, como:

- *Portabilidade:* Os módulos podem ser executados em diferentes plataformas sem modificações, mantendo suas funcionalidades, i.e., devem seguir padrões de *software* e *hardware* de forma a manter sua compatibilidade a outras plataformas.
- *Extensibilidade*: Um número variável de módulos pode ser executado simultaneamente em uma mesma plataforma, sem causar conflitos, i.e., essa característica depende principalmente do sistema operacional, o qual deve possuir um processamento multitarefa, e também do nível de acoplamento dos módulos, que deve permitir essas operações.
- *Interoperabilidade*: Os módulos trabalham em conjunto consistentemente e podem trocar informações de uma maneira definida. Essa característica é enfatizada no modelo de referência apresentado anteriormente (Fig.17), onde está demonstrada a interconexão entre os módulos.
- *Escalabilidade:* Dependendo dos requisitos do usuário, as funcionalidades dos módulos, o desempenho e o tamanho do *hardware* podem se adaptar.

Essas características definem o "grau de abertura" de um sistema, quanto mais estendidas e aprimoradas, maior será o nível da abertura do sistema. Ainda é possível se

adicionar, mais uma característica que vem sendo constantemente utilizada em sistemas abertos.

- *Modularidade:* O sistema é dividido em subsistemas, denominados de módulos, que podem ser substituídos. O modelo de referência (Fig.17), apresentado anteriormente, é baseado em uma arquitetura modularizada em níveis hierárquicos (i.e., camadas).

Para um sistema possuir os requisitos da definição da *IEEE* e conseqüentemente alcançar um "grau de abertura" aceitável, o sistema de controle deve ser (Prischow, 2001):

- *Vendor neutral*: é uma forma de certificação que garante independência de propriedade sobre o sistema, assim não ocorre a existência de nenhum vinculo de produto ou fabricante.
- *Consensus-driven*: determina que seja controlado por um grupo de usuários ou fabricantes.
- *Standards-based*: assegura que a distribuição ocorra por padrões nacionais e/ou internacionais.
- *Freely available*: é de acesso livre para qualquer parte interessada.

3.2.2 *Categorias*

Os controladores são caracterizados por sua liberdade de acesso a informações, ou simplesmente por seu "grau de abertura". Geralmente o controle de diversos componentes do sistema (e.g., a unidade de potência e o controle em baixo nível) são proprietários e não podem ser modificados pelo nível de usuário, outros, são considerados abertos (e.g., a interface de comunicação e o controle em alto nível), i.e., são baseados em padrões de *hardware* e *software* com especificações de interface aberta.

Em Ford, (1994), e Prischow, (2001), o "grau de abertura" de um sistema possui sua definição baseada no conceito de acesso às camadas do controlador, vistas no modelo de referência (Fig.17). Os autores classificam os controladores robóticos em três categorias (Fig.18):

- *Proprietários*: A abertura dessa modalidade de sistema está concentrada apenas no acesso à camada de aplicação, sendo portanto, um sistema fechado. Nesses sistemas é extremamente difícil ou até impossível a integração de módulos externos.
- *Híbrido ou Restrito*: Essa categoria disponibiliza o acesso à camada de aplicação e um acesso controlado à parte referente ao sistema operacional. O sistema operacional tem uma topologia fixa, porém permite pequenas alterações nos módulos de controle do sistema (e.g., ganhos e parâmetros).
- *Aberto*: Sistemas com arquitetura aberta promovem o acesso integral das camadas de aplicação e de sistema operacional, fornecendo uma visão única do sistema. Permitindo a manipulação e alteração de todos os módulos que compõem o sistema. Assim oferecendo permutabilidade[8], escalabilidade, portabilidade e interoperabilidade.

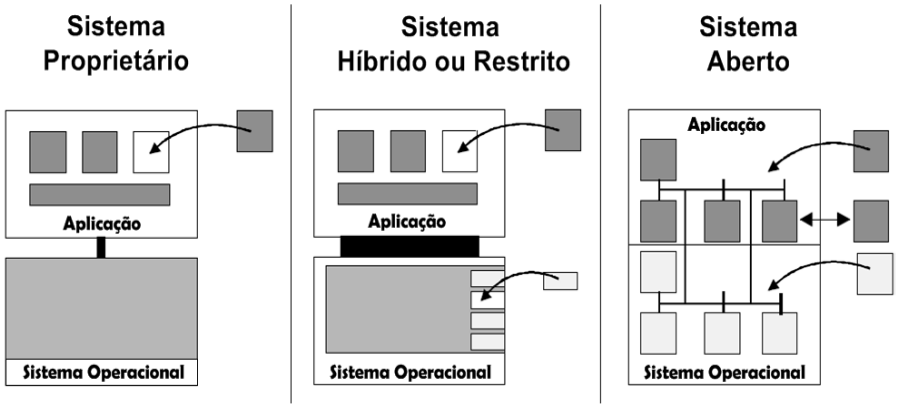

Figura 18 - Categorização de sistemas por seu "grau de abertura" (Sperling e Lutz, 1997).

3.2.3 *Requisitos*

Um sistema de controle da modalidade *vendor neutral* pode ser realizado unicamente se as funcionalidades do controle forem subdivididas em unidades funcionais (i.e., modularizado) e se houver um relacionamento bem definido entre esses subsistemas (Fig.19).

[8] *Habilidade de um objeto ser substituído por outro sem alterações significativas.*

Exatamente a fundamentação do modelo de referência apresentando anteriormente, onde na Fig.17 é demonstrada a modularização e a definição de seus relacionamentos. Conseqüentemente a modularidade torna-se fundamental para um sistema de controle ter uma arquitetura aberta (Prischow, 2001).

A determinação da complexidade do módulo deve considerar fatores como o "grau de abertura" desejado e o custo de integração. Módulos pequenos fornecem uma alta "abertura", mas incrementam a complexidade e os custos de integração. Uma baixa modularização pode conduzir a alta demanda de recursos e deteriorar o desempenho do sistema, comprometendo até que a articulação dos dados seja em tempo real (Prischow, 2001).

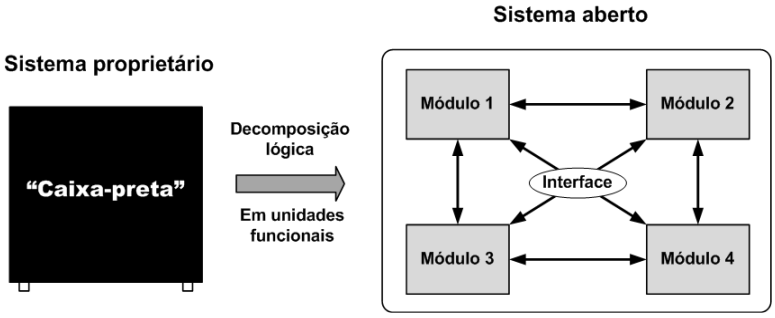

Figura 19 - Decomposição em funcionalidades do sistema de controle (Prischow, 2001).

A estruturação do sistema através de uma interação modular requer um conjunto detalhado de métodos de relacionamento, composto por Interfaces para Programação de Aplicação (*APIs*). Estas são um conjunto de rotinas e padrões de um *software* para a utilização externa de suas funcionalidades por programas aplicativos. Em sistemas de controle aberto as interfaces necessitam ser padronizadas (Prischow, 2001).

Nacsa, (2001) realiza uma comparação aprofundada entre os três modelos de referência para arquitetura aberta de controladores mais expressivos, em três categorias distintas relativos à: propriedades de intercomunicação dos aplicativos (*API*), características arquitetura de referência e infra-estrutura (i.e., características dos *hardwares* e dos ambientes de *software* respectivamente). Os modelos de referência abordados são o *OSACA* da Europa, explicado detalhadamente em Sperling e Lutz, (1997), o americano *OMAC* e o japonês *OSEC,* também abordados por Prischow, (2001). O modelo proposto se baseou em algumas características utilizadas nestes modelos de referência, principalmente no método de comunicação de aplicativos do modelo *OSACA*, demonstrado na Fig.20.

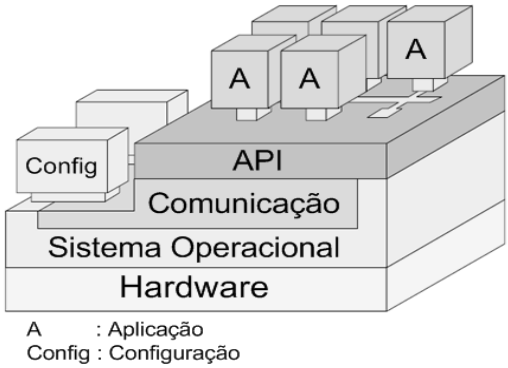

Figura 20 - Plataforma para um sistema de controle aberto do modelo de referência OSACA (Sperling e Lutz, 1997).

As plataformas modulares encapsulam os métodos específicos dos sistemas computacionais, absorvendo as características de *hardware*, sistema operacional e comunicação. Promovendo um intercâmbio de dados em alto nível, tal forma de abstração requer um módulo de mediação de informações entre *softwares*, denominado de *middleware*. Estes pontos de concatenação de dados incrementam a portabilidade e a interoperabilidade das aplicações distribuídas em ambientes heterogêneos (Prischow, 2001).

3.2.4 *Arquiteturas*

Recentemente Lippiello, Villani *et al.*, (2007) resumem que varias arquiteturas abertas de controle para robôs industriais vem sendo desenvolvidas pelos fabricantes de robôs e controladores e pelos laboratórios de pesquisa. Assim, os projetos mais recentes, diferenciados e expressivos são resumidos na seqüência, sobre os quais é apresentada uma breve visão crítica, ressaltando os pontos positivos e negativos.

Hong, Kim *et al.*, (2001) relata o desenvolvimento de um sistema de controle aberto para robôs baseado em uma computador pessoal, denominado de *PC-ORC* (*PC-based Open Robot Control*), fundamentado no modelo de referencia *OSACA* (*Open System Architecture for Controls within Automation Systems*). O sistema preza pela modularidade e reusabilidade. Assim utilizou a orientação a objetos como ferramenta para obtenção destes requisitos lógicos. O foco de desenvolvimento foi um manipulador de topologia clássica, um *SCARA*. A concepção do sistema utilizou o sistema operacional *Windows NT*, com a justificativa de explorar sua estabilidade em computadores pessoais. Porém devido a não ser um sistema com operações em tempo real, torna-se difícil garantir o processamento de um ciclo de tarefas em

pequenas frações de tempo (e.g., milisegundos). Desta forma, fora utilizado um *hardware* dedicado para este processamento e o *PC* opera nas tarefas que não possuem esse requisito, i.e., tarefas de planejamento e geração do movimento. Não são disponibilizadas muitas informações sobre o *hardware*, denominado de *PMAC*, apenas, que é composto por memórias do tipo *RAM*, para armazenar até 2000 pontos de trajetória recebidos pelo *PC* e sua capacidade pode ser expandida.

A modularidade e reusabilidade adotadas para o *software* são características fundamentais para uma rápida prototipagem de sistemas de controle. Porém a falta de informações do *hardware* levanta algumas duvidas sobre o projeto, não permitindo muitas conclusões. O ponto mais importante é o desenvolvimento de um conjunto de instruções aberto para robôs, denominado de *ORIS* (*Open Robot Instruction Set*), composto por quatro bibliotecas de controle: de movimento, de configuração, de programação e para a expansão do sistema.

Bona, Indri *et al.*, (2001) propõem uma arquitetura em tempo real para o desenvolvimento de sistemas de controle robótico. O sistema foi criado para a utilização com um manipulador planar de 2 elos de juntas rotacionais e foi desenvolvido com o intuito de reduzir e simplificar o ciclo de projeto. Assim, baseou sua arquitetura em um computador pessoal, que através de uma comunicação via interface paralela se conecta ao *hardware* de gerenciamento, denominado de *OpenDSP*. Este é composto com um processador de sinais digitais (*DSP*), que rege o sistema, seus sinais de entrada e saída passam por dois dispositivos de lógica programável (*PLDs*) e opera com um sistema operacional em tempo real para sistemas embarcados, o *RTOS* (*Real Time Operating Systems*).

A simplicidade do sistema deu-se pela baixa quantidade de juntas. Para um sistema mais complexo, grandes alterações tornam-se necessárias. A interface de comunicação utilizada é extremamente lenta e obsoleta, e extinta em computadores modernos. Porém o controle é centralizado em um dispositivo de alto processamento, um *DSP*, que é a tendência para sistemas embarcados que necessitam de alto desempenho. Outro beneficio de sistema, é a integração com um ambiente matemático (*MATLAB*) para simplificar o desenvolvimento das leis de controle.

Donald e Dunlop, (2001), apresentam um *retrofitting* de um sistema de controle de trajetória para um robô hidráulico de 6 eixos, o *Unimate 2000B*, o intuito dessa operação era promover um controlador para um controle de trajeto, diferentemente do original que realizava um controle ponto a ponto. O controlador foi realizado com a utilização de uma

placa de desenvolvimento da *Xilinx In*., composta por um *FPGA* (*Field Programmable Gate Array*) [9] e conectada a um computador pessoal pelo barramento *ISA* (*Industry Standard Architecture*) de 16 bits. O *hardware* possui um sistema operacional embarcado o *RTSS* (*Real-Time Sub-System*). O computador é regido pelo sistema operacional *Windows NT* operando com a Extensão para Tempo Real (*RTX*) da *VenturCom.* O sistema de controle foi implementado utilizando uma *IDE* de alto nível o *Microsoft Visual C++*.

A solução adotada diminui o "grau de abertura" do sistema, pois o *hardware* é proprietário. Porém, promove uma alta flexibilidade de implementações pela reconfigurabilidade do *FPGA*, mas requer um alto nível de especialização para essa tarefa.

Macchelli e Melchiorri, (2002), apresentam um sistema de controle em tempo real baseado no *RTAI-Linux* (*RealTime Application Interface for Linux*) para um robô industrial de seis graus de liberdade, o *Comau SMART 3-S*. O controlador de robôs C3G-9000 é "aberto" em algumas camadas, para a inclusão de um efetuador final avançado (i.e., três graus de liberdade e realimentação por visão). Em Lippiello, Villani *et al.*, 2007 e Caccavale, Lippiello *et al.*, 2005 são apresentadas a expansão desse projeto, para um sistema de controle completo para dois manipuladores, possibilitando a cooperação robótica.

O *retrofitting* é realizado apenas nas camadas mais superiores, visto que a maior parte do controle encontra-se no *PC*. Dessa forma, o "grau de abertura" do sistema é muito limitado, sendo que existe apenas uma abertura lógica das informações. As interfaces utilizadas nesta parte aberta do controlador são muito limitadas e obsoletas (e.g., interface paralela e serial e barramentos *ISA* e *VME*).

Regenstein e Dillmann, (2003), apresentam o projeto de uma arquitetura de *hardware* para um robô humanóide *ARMAR*. Através de uma ótima documentação do projeto, é vista uma arquitetura baseada em *PC*, porém uma modalidade geralmente utilizada para aplicações móveis o *PC/104*, operando com a plataforma *RT-Linux* (*Real Time Linux*). Assim, um *PC* ou um PC/104 se conectam através de um barramento de comunicação (e.g., *Firewire*), com uma camada intermediária (i.e., *middleware*), com a função de multiplexar a informação proveniente dos hardwares de controle, cuja troca de informações utiliza o barramento

---

[9] *Denominado de Arranjo de Portas Programável em Campo, um dispositivo lógico programável que permite que seu hardware seja configurável. Sua mutação interna é realizada com linguagens de programação Verilog e VHDL (VHSIC Hardware Description Language).*

industrial *CAN* (*Controller Area Network*). O *hardware* é composto por um processador de sinais digitais (*DSP*) conectado com um dispositivo de lógica programável (*PLD*) para incrementar a flexibilidade do sistema. Cada *hardware* é responsável pelo controle de duas juntas, i.e., uma arquitetura modular.

A utilização de interfaces de comunicação de alto desempenho (e.g., *CAN* e *Firewire*) demonstra a preocupação com a velocidade de transferência de dados, não permitindo que a comunicação torne-se o "gargalo" limitante do sistema. Novamente, são utilizados artifícios para fornecer uma flexibilidade de implementação de *hardware*. A modularidade provê ao sistema facilidade de manutenção e expansão.

Lages, Henriques *et al.*, (2003) relatam a experiência de um *retrofitting* de um robô industrial *ASEA IRB6* ocorrido no Brasil. A parte do processo referente ao controlador, composta por etapas de *software* e *hardware*, apresenta o desenvolvimento de um sistema de controle com um enfoque de distribuição do processamento. Uma arquitetura de controle aberto e modular e descentralizado, onde cada junta possui seu processamento e todas estão ligadas a um computador pessoal, uma rede com topologia de estrela. O computador opera com o sistema operacional Linux e sua variante para o processamento em tempo real *RTAI*. A comunicação dos módulos é realizada pelos barramentos *CAN* e *Ethernet*, com funções de transferências dos dados de controle e supervisão, respectivamente.

A arquitetura selecionada nesse projeto é a tendência para o desenvolvimento de um sistema de controle de alto desempenho. Porém, o *hardware* não fora totalmente desenvolvido pela universidade foi utilizado um módulo de processamento *TINI* da *Dallas Semiconductor Corporation*, o que acarreta uma dependência com o fabricante, aumento no custo do projeto e restringe a "abertura" do sistema. Impossibilitando pequenas alterações nesta unidade de processamento, caso haja necessidade, está deverá ser substituída.

Sumarizando as características das arquiteturas apresentadas, que são igualmente vistas em outros projetos, nota-se que o desenvolvimento de controladores robóticos de arquitetura aberta segue alguns padrões como: sistemas baseados em computadores pessoais (para facilitar a expansão, manutenção e manuseio), utilização de sistemas operacionais para operações em tempo real, alto grau de modularidade, reaproveitamento de código, *hardwares* e comunicações de alto desempenho e o desenvolvimento integral do sistema (para promover uma abertura total do sistema de controle, minimizar os custos e possuir uma solução personalizada que cumpra todos os requisitos do projeto).

Capítulo 4. Proposta de Arquitetura Aberta para Controladores de Robôs Manipuladores

O modelo de referência para sistemas de controle de robôs em aplicações industriais proposto por Sciavicco e Siciliano, (2004), apresentado na Fig.16, possui um enfoque prioritário na estrutura de controle, pouco explorando os demais níveis que compõem um controlador de robôs. A proposta de modelo de referência para controladores de robôs com arquitetura aberta, apresentado na Fig.21, baseia-se neste modelo, porém adéqua suas camadas de acordo com a norma *ISO 7498-1*, apresentada no Apêndice A, e ainda leva em consideração a definição, as categorias e os requisitos, apresentados no capítulo 2, para controladores de arquitetura aberta. Formalizando um modelo estruturado hierarquicamente em cinco camadas, as quais serão individualmente descritas a seguir.

No modelo de referência proposto, a camada de tarefa compreende as tarefas do controle de robôs industriais, as quais podem ser vistas na Fig.22, que podem ser agrupadas em três categorias: geração da trajetória, sistema supervisório e a estrutura de controle. O processamento dessas operações ocorrerá no equipamento central do sistema, que em controladores de robôs de arquitetura aberta segue a tendência apresentada e justificada anteriormente na seção 3.1.1. Ou seja, é utilizado um computador pessoal (PC) convencional para essa operação. Em operações de controle do manipulador à distância, essas operações podem ser dividias em dois softwares, com relação cliente-servidor. A geração da trajetória e o sistema supervisório serão processados com menores requisitos de tempo no cliente, enquanto a estrutura de controle será processada em tempo real de aplicação no servidor.

A estrutura hierárquica da arquitetura funcional adotada, junto com a articulação entre diferentes módulos, sugere uma implementação do sistema que explore os recursos da computação distribuída e interconectada por meio de canais de comunicação apropriados. A *camada de integração* do modelo de referência proposto realiza a adequação (i.e., concatenação e organização) das informações provenientes dos diversos processadores que compoem o sistema distribuído. Fornecendo à camada superior uma visão de heterogeneidade do sistema ao compartilhamento de recursos. Periféricos com alto nível de abstração (e.g. sensores exteroceptivos) também são adequados neste nível.

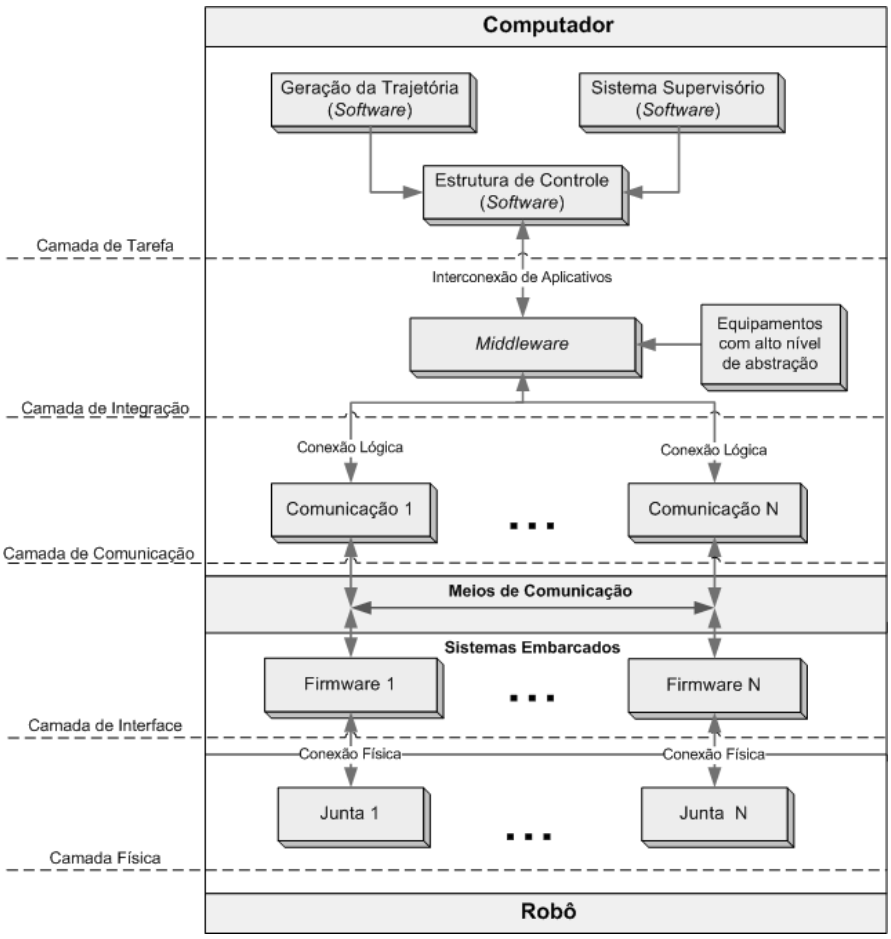

Figura 21 - Modelo de referência para controladores de robôs com arquitetura aberta.

.

A *camada de comunicação* realiza a interconexão das informações entre os processadores do sistema, geralmente utilizando barramentos com alta velocidade de transmissão de dados. O modelo de topologia da rede utilizado é indiferente, porém é ressaltada a importância de caminhos redundantes, que formalizem uma interconexão entre os sistemas embarcados, e por um meio de comunicação alternativo, interligue esses ao nó central da rede. As interconexões realizadas por esta camada seguem a norma *ISO 7498-1* abordada anteriormente na seção 3.2.5.

A *camada de interface* comporta os sistemas embarcados, i.e., *hardware* que processa unicamente um *software* específico à tarefa (*firmware*), que fica encapsulado em seu interior. Tendência abordada na breve visão crítica, realizada sobre alguns projetos recentes de controladores de robôs de arquitetura aberta, apresentada na seção 3.2.4, onde é ressaltada a utilização de sistemas embarcados, que realizam a modularização do *hardware* e distribuem o processamento do sistema. Essa tendência adiciona, ao sistema, processadores dedicados à

tarefa, que garantem um tempo de resposta fixo e mínimo. O grau de distribuição do processamento total do sistema é proporcional ao nível de utilização dos processadores dedicados do sistema.

O acesso físico ao manipulador industrial, mais precisamente aos atuadores e aos sensores proprioceptivos, ocorre na *camada física*, onde estão alocados unicamente os canais de entrada e saída de informações do robô. Geralmente, o acionamento dos atuadores é realizado indiretamente, pois, os sinais de atuação regem a unidade de potência e estas, adéquam os sinais aos atuadores.

Ao aplicar o modelo de referência proposto, para alcançar uma concepção de controlador de robô com arquitetura totalmente aberta (i.e., totalmente acessível e modificável), deveriam ainda ser considerados os requisitos para o desenvolvimento do controlador levantados durante a fase de planejamento. São esses:

- 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.

A arquitetura funcional, fundamentada em uma estrutura hierárquica, da proposta de modelo de referência para controladores de arquitetura aberta (Fig.21) fornece, ao sistema controlador desenvolvido (Fig.22), um alto grau de modularização, proporcionando uma visão de rede de processadores ao sistema. Assim, puderam ser utilizadas topologias convencionais de rede de computadores. A interconexão do sistema utilizou uma derivação das topologias barramento e estrela.

O sistema baseou-se na utilização de um computador pessoal como nó central, adicionando diversas funcionalidades, previamente descritas, ao sistema. O processamento foi distribuído aos oito processadores (incluindo o *PC*) que compõem o sistema.

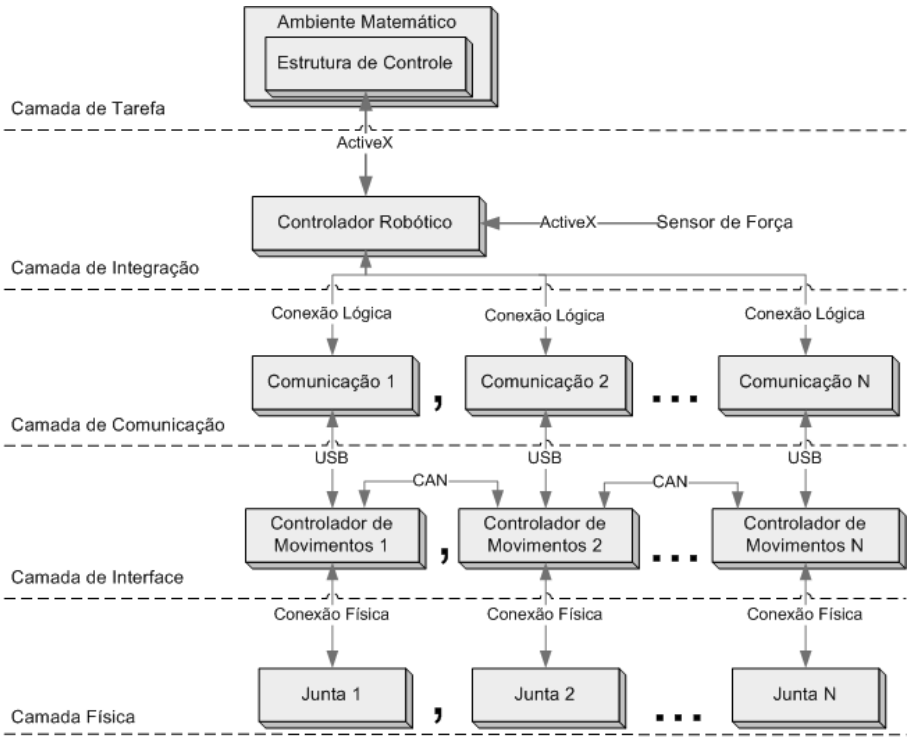

Figura 22 - Arquitetura funcional do controlador.

Na seqüência serão detalhadas individualmente as camadas de compõem a arquitetura funcional do sistema controlador de robôs desenvolvido: a camada de tarefa, a camada de integração, a camada de comunicação, a camada de interface e a camada física.

4.1 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 estrutura de controle do manipulador, com o auxílio de um ambiente matemático.

4.1.1 *Tarefa*

O sistema foi integrado a um ambiente de operações matemáticas tradicional, de forma a facilitar a implementação das estruturas de controle, que podem possuir alta complexidade matemática. Fundamentalmente, a matemática utilizada no controle de robôs manipuladores é matricial, igualmente ao ambiente matemático, fornecendo uma grande diversidade de funções básicas a estas operações.

O uso de ambientes matemáticos para o controle de robôs manipuladores é comum, desta maneira, estruturas de controle implementadas para simulações, podem facilmente ser migradas para operações em ambientes reais.

Tarefas de planejamento (e.g., geração da trajetória) têm uma grande flexibilidade no tempo de processamento, diferentemente das tarefas de controle (e.g., estrutura de controle) que necessitam sem processadas em intervalos de tempo fixo, i.e., em tempo real de aplicação.

4.1.2 *Ambiente*

Os dados vindos do manipulador e provenientes a ele são armazenados em variáveis do ambiente matemático, possuindo um acesso extremamente simplificado.

A informação precedente das N juntas está disponível em matrizes $nx1$, que correspondem ao vetor de posições q e ao vetor de velocidades $\dot{q}$, onde as linhas representam as juntas robóticas. As grandezas provenientes do sensor de força são armazenadas em uma matriz $6x1$, denominada de $h=\left[f_x\ f_y\ f_z\ \mu_x\ \mu_y\ \mu_z\right]^T$, que contém as medidas de forças e momentos no efetuador final. As informações destinadas ao manipulador devem estar em uma matriz de controle $nx3$ denominada de matriz u.

4.2 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. 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 a comunicação entre os processadores e a intercomunicação entre os processos para a aquisição de dados do sensor de força.

4.2.1 *Middleware*

Em sistemas de processamento distribuído, o *middleware* realiza a adequação dos sinais vindos 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 *APIs* de alto nível para proporcionar a sua integração com aplicações desenvolvidas em quaisquer linguagens de programação.

No sistema de controle de robôs desenvolvido, o *middleware* integra os seis sistemas embarcados que regem as juntas robóticas, o processador que controla o gabinete de acionamento do manipulador e a interface que adquire o sinais provenientes do sensor de força. Neste são disponibilizadas todas as configurações e inicializações necessárias para a utilização do manipulador robótico. A comunicação com a camada superior é realizada por um método de interconexão de aplicativos, discutido a seguir.

4.2.2 *Comunicação entre Aplicativos*

O *ActiveX*, também conhecido como controle *OLE* (*Object Linking and Embedding*), é uma ferramenta que gerencia o uso de softwares reutilizáveis desenvolvida pela *Microsoft Corporation*. O uso desta tecnologia permite a uma aplicação o acesso, a criação e a manipulação de objetos em outras aplicações, independentemente de diferenças entre elas. O controle dos recursos *ActiveX* são realizados através do encapsulamento em containeres (i.e.,

um objeto composto por um ou mais objetos). Este é uma representação visual dos recursos, que também possibilita a troca das informações e comandos com outras aplicações.

O controle *ActiveX* opera como servidor de processo e pode ser utilizado em qualquer container de controle (Fig.23). A funcionalidade completa de um controle *ActiveX* estará disponibilizada apenas no uso de um container preparado.

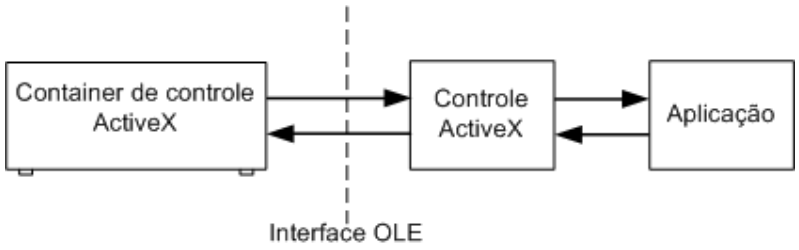

Figura 23 - Comunicação básica por controle ActiveX (Microsoft Corporation, 2007).

O conjunto de funções (i.e., métodos) e atributos (i.e., propriedades) permitidos pelo controle *ActiveX* são disponibilizados em um arquivo, denominado de mapa de distribuição. Ainda existe a possibilidade de operar com eventos entre aplicativos, por todas essas funcionalidades esse método de comunicação tornou-se bastante popular. Atualmente, a *Microsoft*, vem substituindo gradativamente pela plataforma *.Net Framework*, a qual aprimora seus recursos.

4.2.3 *Sensor de Força*

A adição da realimentação de força (Fig.24) é realizada com o acoplamento de um sensor de força ao punho do manipulador. Apesar de ser um item físico, este é visualizado na camada de integração, pois é um dispositivo proprietário, com alto nível de abstração, desenvolvido pela *JR3 Inc.*. Tendo suas camadas inferiores a esta, implementadas e fechadas, permitindo poucos acessos controlados.

Figura 24 - Sensor de força com o efetuador final.

Os sensores de força operam de duas formas distintas, qualitativamente ou quantitativamente. O primeiro tipo opera como um filtro sobre os níveis de força, fornecendo um sinal assim que a aplicação atingir um coeficiente de força pré-determinado. O outro modelo opera com a informação sobre a quantidade de força, fornecendo dados gradativos da força aplicada sobre sua freqüência de amostragem (Pires, 2000). O autor ainda define quatro maneiras de se detectar uma força (ou força por unidade de área = pressão):

- Balanceando a força medida com a força gravitacional sobre uma massa conhecida.
- Medindo a aceleração provocada pela força medida sobre uma massa conhecida.
- Balanceando a força medida uma força eletromagnética gerada.
- Medindo a deformação causada pela força a ser medida, sobre um material elástico.

A maioria dos sensores de força e momento empregados na robótica industrial realiza a medida da deformação sobre um componente elástico, geralmente de alumínio ou aço inoxidável, e fornecem uma medida gradativa de força em seis eixos (três de força e três de momento). Para realizar a medida desta grandeza são empregados principalmente dois métodos, por extensômetros ou sensores piezelétricos.

Os sensores piezelétricos baseiam-se na propriedade de alguns cristais alterarem sua freqüência de ressonância natural quando se encontram sobre a atuação de uma força. Já os

sensores que operam com extensômetros, manipulam o efeito piezoresistivo, apresentados em alguns materiais quando sujeitos a uma força (Pires, 2000).

Na robótica industrial é muito comum o uso de sensores de força e momento que utilizam extensômetros. Internamente ainda podem ser compostos por circuitos eletrônicos, baseados em processadores de sinais digitais (*DSP*), para realizar o devido tratamento do sinal. Essa forma de sensoriamento provê um sinal analógico, necessitando de uma digitalização e a transmissão do sinal até a base de processamento onde será calculada a estrutura de controle do manipulador.

Tabela 1 - Especificações do sensor de força e momento.

	Capacidade de Carga (definida eletronicamente)	Capacidade de Carga (Sensor)	Resolução
FX	200 N	220 N	$6.10\ 10^{-3}$
FY	200 N	220 N	$6.10\ 10^{-3}$
FZ	400 N	440 N	$12.20\ 10^{-3}$
MX	20 Nm	22 Nm	$6.10\ 10^{-4}$
MY	20 Nm	22 Nm	$6.10\ 10^{-4}$
MZ	20 Nm	22 Nm	$6.10\ 10^{-4}$

O sensor adquirido para este trabalho é do modelo *100M40A-I63 200N20*, com um diâmetro de nominal de 100 mm, altura nominal de 40 mm e peso nominal de 570 g. É um dispositivo feito sobre encomenda, personalizado para a aplicação, desta forma, a furação do sensor é compatível com os orifícios de acoplamento do punho do manipulador utilizado. O qual contém uma eletrônica interna que realiza o condicionamento adequado do sinal. Sua interface com um computador pessoal foi realizada com a aquisição de uma placa de integração, do mesmo fabricante, através da interface *PCI*. As especificações de limites de carga deste sensor são apresentadas na Tabela 1.

4.3 Camada de Comunicação

Em Tanenbaum, (1997), uma rede de computadores é definida como: “um conjunto de computadores autônomos e interconectados”. A articulação das informações para um sistema com computação distribuída, nos fornece uma visão do mesmo como uma rede de

computadores. Desta forma, a interconexão do sistema pode seguir normas e barramentos padrões em computadores.

A composição da topologia estrela (Fig.25) com o barramento[10] (Fig.25) gera um perfil redundante para a rede de computadores, pois o anel não é lógico como utilizado em alguns padrões tradicionais, 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).

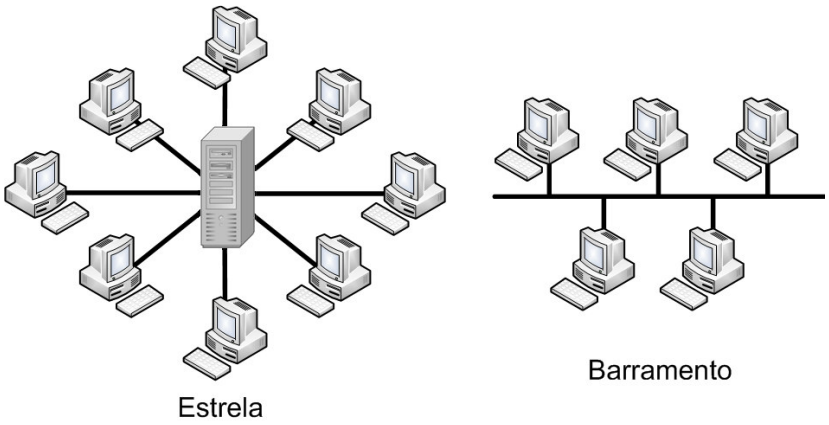

Figura 25 - Topologias de rede utilizadas.

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 hierárquica de juntas, para a adequação e o processamento completo dos dados.

A utilização conjunta dos meios de comunicação permite uma maior independência do processamento do nó central, sendo que os nós secundários são de alta capacidade de processamento e suas funcionalidades ainda não foram totalmente exploradas. Possibilitando

---

[10] *Meio físico onde ocorre o tráfego estruturado entre os periféricos.*

a utilização do PC, apenas como uma *IHM* (Interface Homem-Máquina) e um sistema supervisório.

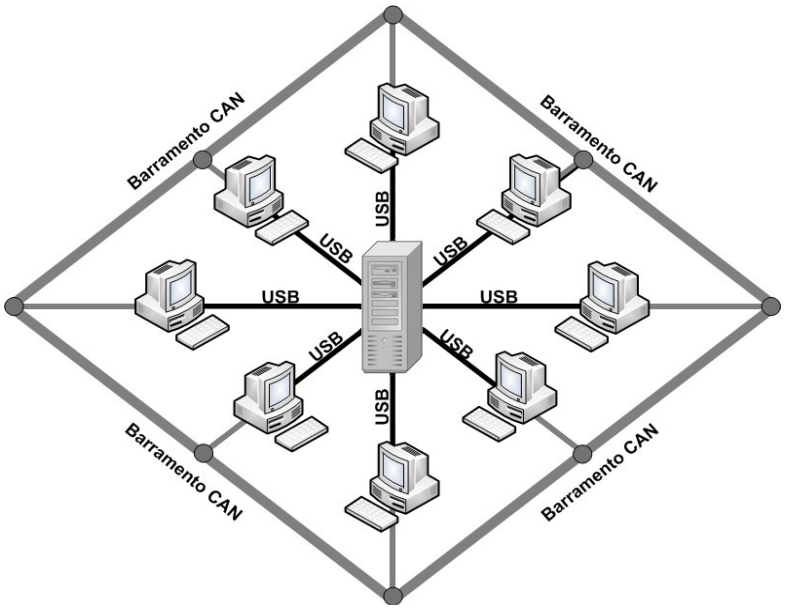

Figura 26 - Interconexão do sistema proposto.

A velocidade de comunicação (ou largura de banda) é um fator importante em sistema de controle robótico de arquitetura aberta, principalmente em sistemas com alto grau de modularidade. A arquitetura proposta utiliza processadores individuais para cada junta. Assim, são *n* vias de comunicação dos processadores periféricos com o nó central. Visto esses requisitos, a velocidade de transferências das informações com o nó central pode ser o "gargalo" do sistema. Para isso, foram pesquisados os principais métodos de comunicação de alta velocidade. A Fig.27 mostra um comparativo de algumas formas de comunicação em relação a velocidade de transmissão.

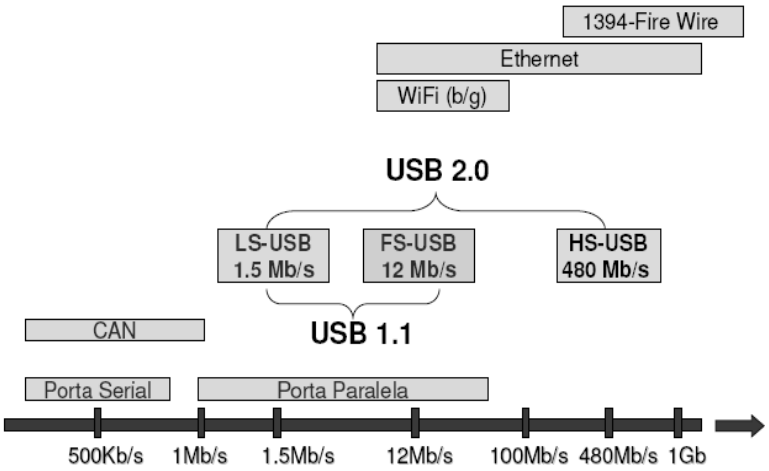

Figura 27 – Comparação entre os métodos de comunicação mais populares (Microchip Technology Inc., 2004).

A seguir serão apresentados os métodos de comunicação utilizados, o *USB* (*Universal Serial Bus*) e o barramento *CAN* (*Controller Area Network*), justificando suas escolhas.

4.3.1 *Barramento USB*

O protocolo *USB* (*Universal Serial Bus*) foi desenvolvido em conjunto por um grupo de empresas líderes no mercado de comunicação e de *PCs*, incluindo: *Compaq*, *Hewlett-Packard*, *Intel*, *Lucent*, *Microsoft*, *NEC* e *Philips.* A motivação para essa ação conjunta veio da necessidade de criação de um protocolo de comunicação com alta velocidade, de fácil utilização e baixo custo. Ainda, deveria se adequar a uma ampla variedade de aplicações e permitir o funcionamento simultâneo de uma grande quantidade de equipamentos. Todas essas características levaram a utilização deste barramento na arquitetura de controlador proposta. A *USB* possui uma especificação bastante rigorosa que pode ser vista em Compaq, Hewlett-Packard *et al.*, (2000), na qual foram baseadas as informações apresentadas na seqüência.

Atualmente a interfaces USB possui três versões, a primeira, denominada de *LS-USB 1.0* (*Low Speed*) atinge velocidades de transmissão de até 1.5 Mbps. Na seqüência, veio a *FS-USB 1.1* (*Full Speed*) que suporta velocidade de até 10 Mbps. A versão atual, chamada de *HS-USB 2.0* (*High Speed*), alcança taxas de transferência de até 480 Mbps. A compatibilidade com as versões anteriores é mantida nas versões mais atuais. Essas taxas de transferências são em relação a todo o barramento do servidor, i.e, são divididas por todos os periféricos conectados ao sistema.

Os dispositivos podem ser conectados em topologias de rede em série (ou cadeia), árvore ou estrela. Sendo que sempre o nó central (ou inicial) será um servidor (ou *Host*), que controlará o tráfego do barramento. Cada servidor suporta até 127 dispositivos operando simultaneamente e sem conflitos.

Os dispositivos com comunicação *USB* têm seus conectores padronizados (Fig.28). Atualmente existem dois conectores distintos segundo sua especificação: o tipo *A* e o tipo *B*. O tipo *A* (fêmea) é o servidor e o tipo *B* (fêmea) é o dispositivo. Os conectores mini *B*, são uma derivação do tipo *B*, para dispositivos menores. Os cabos são compostos por quatro fios, dois para alimentação e dois de comunicação, cujos conectores também são padronizados e podem ser visualizados na (Fig.28), os machos. Cada cabo pode ter o comprimento máximo de cinco metros, caso seja necessária uma distância superior, deve ser utilizados *hubs*[11] com alimentação para a interligação dos cabos. Suportando no máximo cinco níveis de *hubs*, que totalizam em 30 metros de extensão, do servidor a maior extremidade. Um ponto fundamental que fez com que esta tecnologia de comunicação se tornasse tão popular, é a disponibilidade para a alimentação do equipamento diretamente pelo *USB*. O barramento pode fornecer até 500 mA, a uma tensão de 5 Volts.

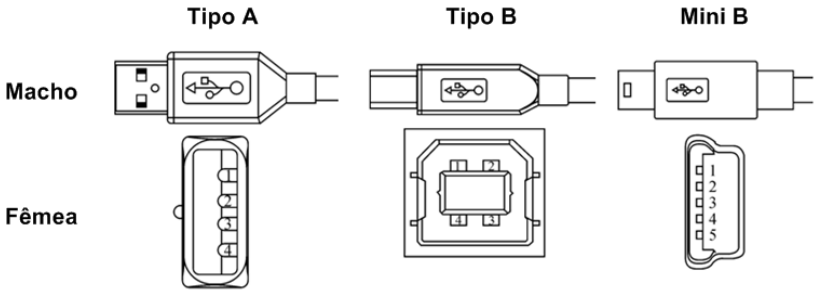

Figura 28 - Conectores USB (Compaq, Hewlett-Packard et al., 2000).

A tecnologia *USB* não é de domínio público, dessa forma cada produto necessita possuir seus códigos de identificação, *Product ID* (i.e., identificador de produto), e cada desenvolvedor necessita de sua identificação, o *Vendor ID* (i.e., identificador de fabricante).

---

[11] *Dispositivos que permitem compartilhar o meio de transferência, e conseqüentemente a largura de banda, entre diversos equipamentos.*

Ambos os códigos são utilizados para fornecer ao sistema a característica de *Plug and Play* (*PnP*), que significa ligar e usar.

Ao conectar um equipamento em um servidor *US*B (e.g. um computador), os identificadores são transmitidos ao servidor, cujo sistema operacional reconhece o dispositivo e que tenta localizar o *driver*[12] correspondente. Caso não haja, uma solicitação de localização do mesmo será enviada ao usuário, que deve indicar o local de armazenamento do *driver*. Após, o *software* de controle é instalado e as regras de comunicação definidas, o sistema operacional armazena essas configurações, para que não seja necessário repeti-las nas próximas utilizações do dispositivo. Assim, o equipamento torna-se *PnP*, pois na próxima conexão com o servidor *USB*, o sistema operacional rapidamente instalará e configurará, tornando-o disponível para o uso automaticamente.

Uma alternativa de utilização dessa forma de comunicação sem a aquisição dos direitos de utilização é realizar o desenvolvimento com a utilização de componentes, cujos identificadores já estão previamente adquiridos pelo fabricante. Os quais possuem *drivers* de utilização de modalidade de *Royalty-Free*, i.e., permitem a utilização sem o pagamento pro direitos de uso.

A crescente disponibilidade desta interface em computadores pessoais, a ampla quantidade de periféricos encontrados no mercado, a simplificada integração dos equipamentos, a possibilidade de conexão com uma grande quantidade equipamentos simultaneamente e a alta velocidade de transmissão, levaram a utilização desta na arquitetura proposta.

4.3.2 *Barramento CAN*

A escolha do barramento *CAN* (*Controller Area Network*), veio de sua característica de operação sem um servidor fixo e a sua fácil adequação a ambientes industriais. Atualmente, é muito utilizado na indústria, em veículos automotivos, navios e tratores.

---

[12] *Software que realiza uma abstração dos procedimentos de comunicação do sistema operacional com algum dispositivo, i.e., hardware.*

O protocolo *CAN* é padronizado pela norma *ISO 11898* e estruturado de acordo com o modelo *OSI* (i.e., norma ISO/IEC 7498-1, 1994), conforme visto na Fig.29.

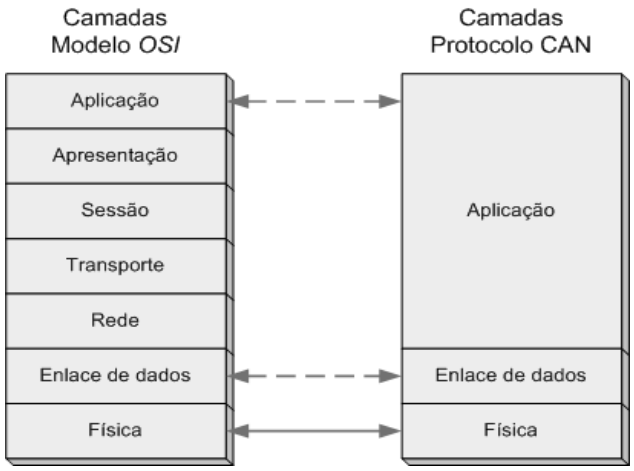

Figura 29 - Comparação protocolo CAN com o modelo OSI.

O protocolo *CAN* realiza uma comunicação através da transferência serial de dados com sincronismo entre os módulos. Baseia-se no conceito de multimestre, i.e., o nível de mestre é articulado entre os nós que compõem a rede em intervalos de tempo pré-determinados e regulares (Farsi, Ratcliff *et al.*, 1999).

Em um protocolo com topologia de barramento, o acesso ao meio é rigorosamente controlado, procedimento que ocorre na camada de enlace segundo o modelo *OSI*, de forma a evitar colisão (i.e., acesso e envio de informações de mais de um nó simultaneamente) no meio de transmissão. A organização do compartilhamento do canal de comunicação previne a ocorrência de colisão no barramento. Obrigando o nó transmissor de "ouvir" o meio, verificando se a mensagem enviada é semelhante à recebida, caso não seja, houve a ocorrência de uma colisão, todos os nós transmissores encerram suas transmissões e aguardam um intervalo de tempo aleatória para a retransmissão (Tanenbaum, 1997).

Assim, apenas um nó pode enviar uma mensagem caso o meio esteja livre, i.e., caso nenhuma mensagem de maior prioridade esteja trafegando no barramento. Se isso ocorrer, o nó que estive enviando a mensagem de menor prioridade encerrará sua transmissão e o de maior prioridade continuará sua transmissão. Essa topologia ainda possui como característica o envio das mensagens ser realizado por um regime de *broadcasting*, i.e., todas as mensagens são enviadas a todos os nós da rede, que devem verificar se são o destinatário, caso não, devem descartar a mensagem (Tanenbaum, 1997).

Tabela 2 - Situações típicas de transmissão do protocolo CAN (Farsi, Ratcliff et al., 1999).

Taxa de transferência	1 Mbps	50 kbps
Comprimento do barramento	40 metros	1000 metros

A velocidade de transmissão de dados pelo protocolo *CAN* é 1 Mbps para 40 metros, porém devido aos atrasos de propagação do sinal em um barramento, a taxa de transferência de dados tem relação direta com o comprimento do barramento, o protocolo possui duas situações típicas apresentadas na Tabela 2.

No sistema proposto, as interfaces *CAN* ficam próximas determinando um barramento curto, assim, possibilitando a utilização da taxa de transferência máxima. Aliando essa característica com a dinâmica de controle dos nós e com a sua comum utilização em ambientes industriais, levaram a escolha deste protocolo de comunicação.

4.4 Camada de Interface

A *camada de interface* compreende os sistemas embarcados, que realizam a modularização do *hardware* e distribuem o processamento do sistema, tendo como funcionalidade principal o processamento e a geração dos sinais que regem o robô manipulador. São *hardwares* que processam unicamente um *software* específico (*firmware*) para o cumprimento da tarefa, o qual fica alocado em sua memória interna. Desta forma, são processadores dedicados a tarefa, os quais, garantem um tempo de resposta fixo e mínimo. O diagrama de blocos internos dos sistemas embarcados desenvolvidos pode ser visualizado na Fig.30.

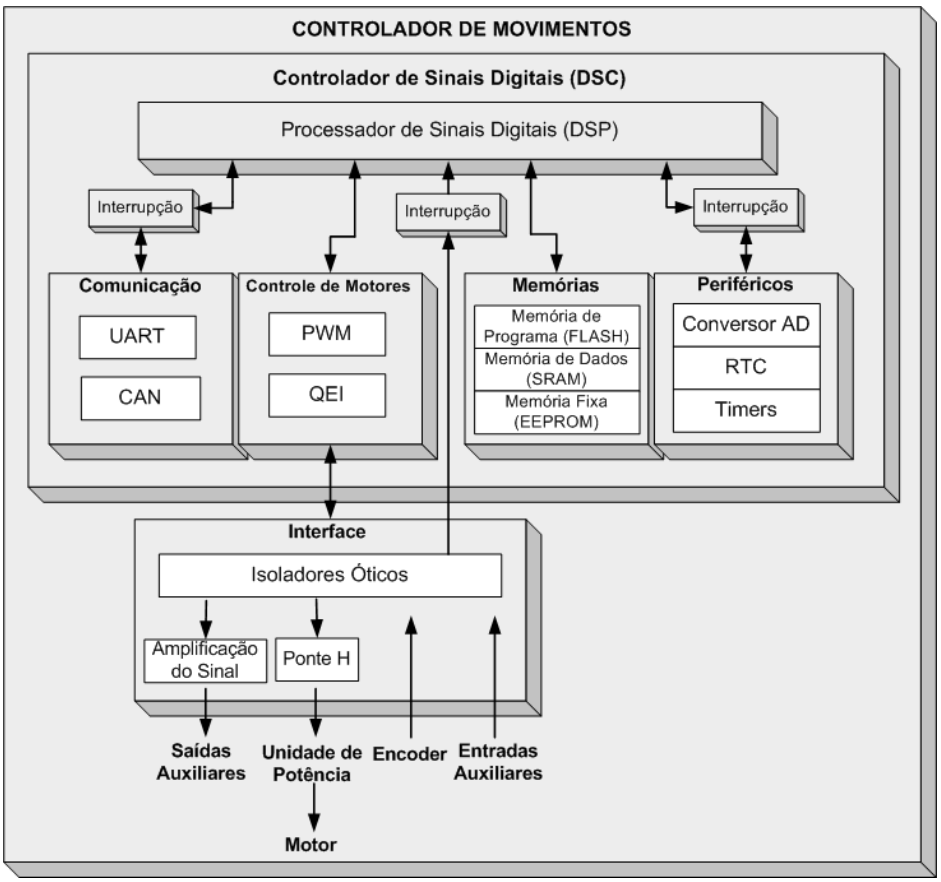

Figura 30 - Diagrama de blocos dos sistemas embarcados desenvolvidos.

Os sistemas embarcados desenvolvidos utilizam controladores digitais de sinais (*DSC*) como a unidade de processamento central do sistema. Tendo como função fundamental a decodificação dos sinais provenientes dos *encoders* incrementais e geração do sinal com modulação por largura de pulso (*PWM*) para o controle dos atuadores. O acoplamento entre o *hardware* e o manipulador é protegido por uma interface de isolamento ótico, com o intuito de prevenir danos.

O detalhamento dessa seção se dará inicialmente pela discussão da forma de armazenamento do *firmware*, para a apresentação da técnica utilizada, que promove uma maior flexibilidade ao sistema. Em seguida, virá a abordagem do algoritmo base desse *firmware*, detalhando as configurações, interrupções e a execução. Após, é tratado do *hardware* deste sistema embarcado, explorando individualmente os principais blocos que o compõem.

4.4.1 *Firmware*

Os sistemas embarcados são *hardwares* que processam unicamente um *software* específico à tarefa, denominado de *firmware*, o qual fica armazenado no interior de sua memória. A alocação do *firmware* nesta memória segue algumas etapas e requisitos, os quais serão abordados a seguir.

4.4.1.1 ***Conversão em Linguagem de Máquina***

Ao se realizar a implantação de um algoritmo para um sistema embarcado é necessário realizar a conversão do código, de maneira que ele se torne compreensível ao processador. Os ambientes e desenvolvimento integrado (*IDE)* realizam todas essas tarefas automaticamente, chamando, geralmente um programa específico para executar cada etapa desta tradução. O processo de conversão é realizado segundo as etapas descritas na Fig.31, que serão detalhadas a seguir.

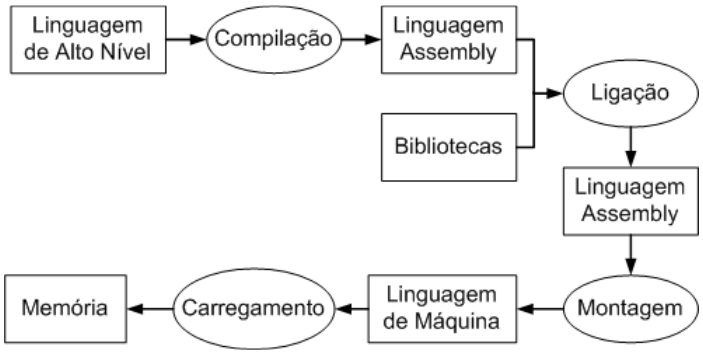

Figura 31 – Etapas da tradução de um software em linguagem de máquina.

A primeira etapa da conversão é a *compilação*, onde ocorre a tradução do *software* de alto nível para a linguagem *Assembly*. O processo de compilação realiza uma análise sobre o código em relação a erros e caso não haja nenhum, gera o código de máquina. Alguns compiladores realizam três etapas em seu processo de compilação. Primeiro geram um código de máquina intermediário, após, este passa por um processo de otimização, então é gerado o código final.

O código *Assembly* gerado pelo compilador geralmente não é o definitivo, pois, pode haver o acoplamento do algoritmo com bibliotecas externas, que ainda não foram

incorporadas ao programa. O compilador inclui ao código, a cada chamada a uma biblioteca externa, uma chamada a rotina correspondente e o endereço dos dados que devem ser passados para a rotina. Na *ligação* ocorre a procura, no código *Assembly,* das chamadas da biblioteca e substituição pelo devido código. Fornecendo como saída um código *Assembly* completo.

A *montagem* é a etapa que realiza a tradução efetiva do programa para linguagem de máquina. Onde ocorre uma análise de todas as instruções, verificando a existência de códigos inválidos, caso exista, este é substituído pelos códigos numéricos equivalentes. Caso tenha algum erro no código o processo é interrompido e uma mensagem de erro é emitida. Em seguida os nomes simbólicos de referência a memória utilizados pelo programador são convertidos em endereços reais de memória. Após isso, os valores constantes são convertidos em binário. Resultando na conversão completa do código para linguagem de máquina.

A *depuração* é uma ferramenta utilizada para encontrar erros em um algoritmo. É um componente que faz parte dos ambientes de desenvolvimento integrados (*IDE*). Os erros podem ser agrupados em três categorias:

- *Erro de sintaxe*: ocorrem quando o programa não está escrito na estrutura da linguagem de programação utilizada.
- *Erros de execução*: são erros que só aparecem quando o programa é executado, indicando a ocorrência de algo não previsto.
- *Erros de semântica*: são erros de lógica, ou seja, o programa está sem erros mais não executa a ação que deveria.

O compilador utilizado para o desenvolvimento dos sistemas embarcados que compõem o controlador de robôs é baseado no padrão *ANSI C*, tendo todas as funções básicas desta regulamentação implantadas, e.g., funções de: manipulação de string, alocação de memória dinâmica, conversão de dados e funções matemáticas (trigonométricas, exponenciais e hiperbólicas). Para o processamento de sinais digitais, contém bibliotecas para manipulação de vetores e matrizes, transformadas, filtros e para controle de sistemas. Ainda estão inclusas bibliotecas para a manipulação de todos os periféricos contidos nos processadores de sinais digitais (*DSC*) deste fabricante. A Fig.32 mostra o diagrama funcional da tradução do algoritmo em linguagem de máquina pelo compilador *C* utilizado.

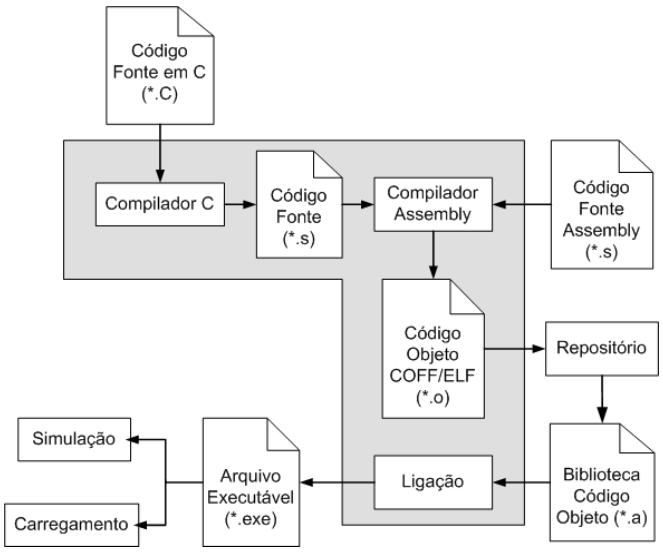

Figura 32 - Diagrama do fluxo de dados da tradução do algoritmo em linguagem de máquina pelo compilador utilizado (Microchip Technology Inc., 2005b).

Após a tradução do *software* para linguagem de máquina, o próximo passo é gravá-lo na memória de programa do sistema embarcado, o *carregamento*. Um sistema embarcado é caracterizado por um processador dedicado, porque processa apenas um algoritmo, que está alocado internamente em uma memória. A seguir, será abordada sucintamente a estrutura de memórias internas de um sistema embarcado.

4.4.1.2 *Estrutura de Memórias de um Sistema Embarcado*

Internamente a um sistema embarcado, o componente mais evidente é a memória, pois, é necessário haver um local para armazenamento do *software* embarcado (*firmware*) e para o armazenamento de dados temporários e fixos. Também é necessária uma memória para alocar as instruções básicas do processador. As memórias eletrônicas têm como características:

- *Tempo de acesso*: intervalo necessário para o acesso a memória e a realização de operação de leitura ou gravação.
- *Capacidade*: quantidade efetiva de dados que podem ser armazenados no interior da memória.

- *Não-Volatilidade*: capacidade de a memória manter seus dados quando não houver energia elétrica.
- *Tempo de Latência*: intervalo mínimo entre cada operação de leitura ou escrita na memória. Devendo ser essencialmente respeitado, pois, pode acarretar na utilização de espaços indevidos da memória, podendo causar a perda de dados ou manipulação incorreta de dados.

Um sistema embarcado é composto principalmente por três tipos de memórias: de dados, de armazenamento e de programa, as quais serão descritas a seguir.

A *memória de dados* compreende na parte onde serão alocados os registros, as variáveis e todos os espaços reservados para o processamento temporário. É uma memória do tipo *RAM* (*Random Access Memory*), volátil (ou seja, não mantém seus dados sem energia elétrica). Sempre que alimentada, inicia com dados inconsistentes. A aleatoriedade consiste no fato de que, para acessar um determinado endereço, não é necessário percorrer todos os endereços anteriores desde o início, mas realizar um acesso direto ao endereço desejado.

A *memória de armazenamento* é utilizada para guardar dados importantes ou alguns parâmetros de processamento. Tem como característica principal a não-volatilidade e permite operações de gravação e leitura. É uma memória do tipo *EEPROM* (*Electrically Erasable Programmable Read-Only Memory*) e pode ser programada e apagada várias vezes por processos elétricos.

Na *memória de programa* fica armazenado o *firmware* do sistema embarcado, comumente uma memória do tipo *FLASH*, ou seja, é programável e apagável eletricamente. Esses processos são realizados rapidamente quando comparados aos outros tipos de memórias. Em casos especiais, o espaço livre dessa memória pode ser reutilizado como extensão memória de dados, devido a uma característica especial desse tipo de *hardware*. É uma memória não-volátil, devido a essa característica, um sistema embarcado pode manter sempre seu *firmware* internamente, sem a necessidade de uma gravação a cada execução.

Então, o algoritmo convertido em linguagem de máquina deve ser armazenado no interior da memória de programa. Geralmente, esta operação necessita da utilização de um *hardware* específico. Porém, existe a possibilidade da implantação de uma solução mais sofisticada que não tenha esse requisito, a qual será apresentada a seguir.

4.4.1.3 *Armazenamento do Firmware*

O armazenamento do *firmware* nos sistemas embarcados desenvolvidos para o controlador de robôs utiliza a técnica denominada de *bootloader*, de forma a promover flexibilidade e independência do *hardware* de carregamento.

O *bootloader* é um pequeno *software* cuja única função é carregar outro *software* para inicializar as operações do processador. Em um sistema embarcado, o processador é dedicado a uma tarefa específica. O *bootloader* é geralmente armazenado no início da memória de programa (Fig.33), sendo a primeira tarefa a ser executada. Ocupando uma pequena parte da memória de programa. Ao ser executado, fornece duas funcionalidades, permitir a atualização do *firmware* através de algum método de comunicação e executá-lo.

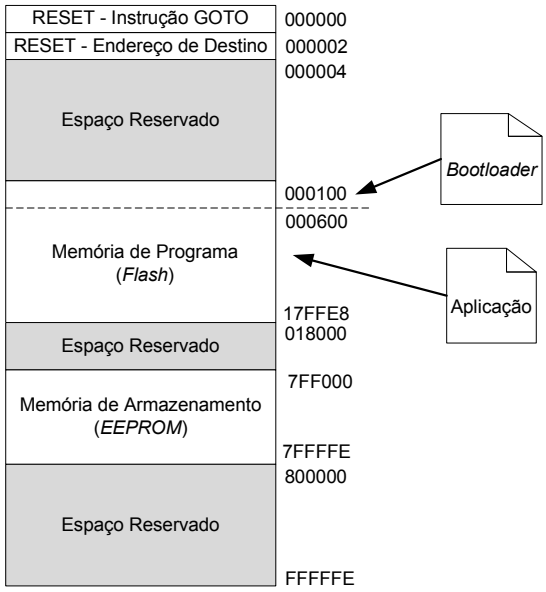

Figura 33 – Estrutura de memória do controlador de sinais digitais.

O funcionamento do *bootloader* acontece por quatro estados distintos. Na inicialização o sistema encontra-se no estado de espera, onde fica no aguardo da atualização do *firmware*, através da comunicação via interface *USB*. Caso não receba informação dentro de um intervalo de tempo pré-determinado, o sistema muda para o estado de execução e processa o *firmware* armazenado na memória. Se dentro desse intervalo o sistema receber informações, sua postura é alterada para o estado de identificação, onde ocorre a verificação do modelo de controlador de sinais digitais, caso este esteja correto o sistema altera-se para o estado de

programação, caso contrário, vai para o estado de execução. No estado de programação, o sistema realiza uma comunicação via interface *USB* para a atualização do *firmware.* Ao finalizar esta tarefa, executa o novo *firmware* no estado de execução. O diagrama de estados de execução do *bootloader* é mostrado na Fig.34.

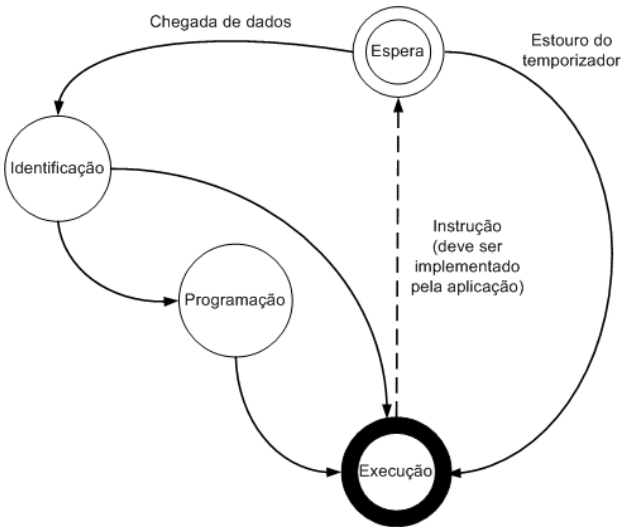

Figura 34 – Diagrama de estados do bootloader.

4.4.2 *Algoritmo*

Como já foi exposto anteriormente, um código desenvolvido em linguagem de alto nível passa por uma série de conversões até poder ser carregado no processador. As quais, geralmente, não são tão otimizadas quanto uma implementação direta em linguagem de máquina. Mesmo com esse fator, as mais de mil linhas de código, implantadas nos sistemas embarcados, só ocupam 13.5% da memória de programa e 1.15% da memória de dados.

O algoritmo foi desenvolvido considerando alguns fatores, como a otimização, a modularização e a documentação de código, de forma a fornecer flexibilidade a alterações e facilidade de entendimento. Assim, o código foi dividido em três categorias: principal, inicialização e interrupções. A seguir, será apresentado individualmente cada um desses blocos, salientando que todas as informações técnicas, apresentadas nessa seção, sobre os controladores de sinais digitais foram baseadas em *Microchip Technology Inc.,* (*2006*).

4.4.2.1 ***Principal***

A rotina principal consiste em um laço de repetição infinito, que realiza o controle proporcional, integral e derivativo (*PID*) sobre a velocidade do atuador, levando em consideração: a referência, a leitura de velocidade e os ganhos. O fluxograma desta rotina é apresentado na Fig.35.

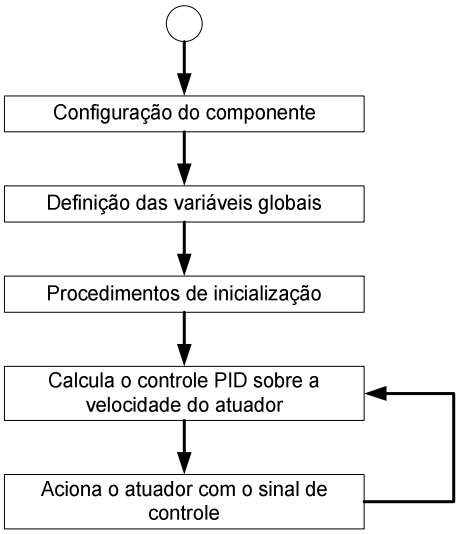

Figura 35 - Fluxograma da rotina principal do firmware.

Dentre os blocos apresentados no fluxograma, é necessário abordar mais detalhadamente a configuração do componente, pois, esta parte é de fundamental importância para o correto funcionamento do controlador digital de sinais (*DSC*). O Apêndice B discute alguns aspectos de funcionamento, que apesar de muito intrínsecos, são extremamente necessários para o entendimento.

4.4.2.2 ***Inicialização***

A rotina de inicialização dos módulos periféricos internos do controlador de sinais digitais é executada no início da rotina principal do *firmware*, onde todos esses módulos tem suas características de funcionamento definidas, e após, são inicializados. O fluxograma de operação desta rotina é apresentado na Fig.36.

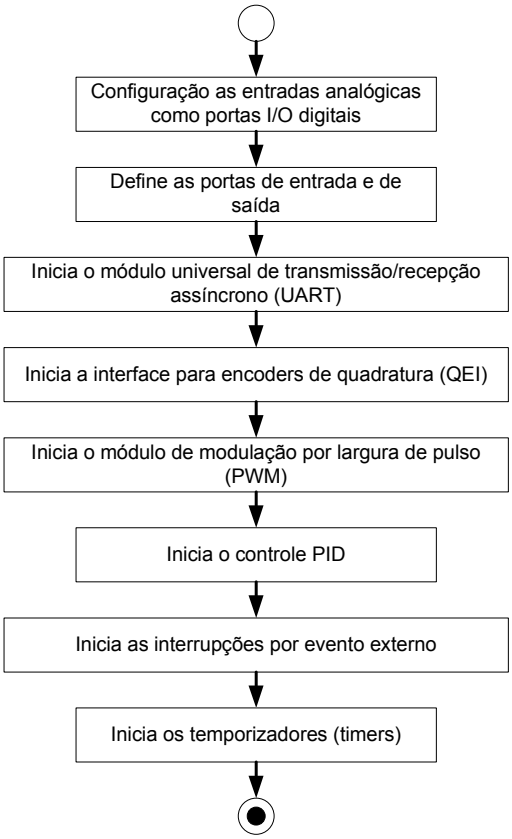

Figura 36 - Fluxograma da rotina de inicialização.

Assim, tornando-se um procedimento de fundamental importância para o correto funcionamento do sistema. Por isso, o Apêndice C realiza um detalhamento dos módulos de maior importância, como: o módulo universal de transmissão/recepção assíncrono (*UART*), a modulação por largura de pulso (*PWM*), a interface para *encoders* de quadratura (*QEI*) e o controle proporcional, integral e derivativo (*PID*).

4.4.2.3 ***Interrupções***

As interrupções são as formas de atualização das informações do sistema em cada interação. Pois, a rotina principal apenas trata do controle *PID* do atuador, levando em consideração os dados contidos nas suas variáveis de referência. As interrupções são responsáveis por armazenar nestas variáveis os dados referentes à manipulação do motor, também responsáveis pelas rotinas de leitura do *encoders* e pelo gerenciamento da troca de informações nos meios de comunicação.

Chegada de dados pelo módulo *UART*

O módulo *UART* é responsável pela comunicação através da interface *USB*, desta forma, sua interrupção é gerada a cada nova chegada de dados por esse meio. Sua função é receber, converter e armazenar corretamente os dados recebidos nas variáveis adequadas. Podem ser recebidos dois tipos de informações, a de atualização dos ganhos do controle *PID* e o pacote de controle *U* (i.e., perfis de posição, velocidade e aceleração). A Fig.37 demonstra o fluxograma de funcionamento do algoritmo para tratar essa interrupção.

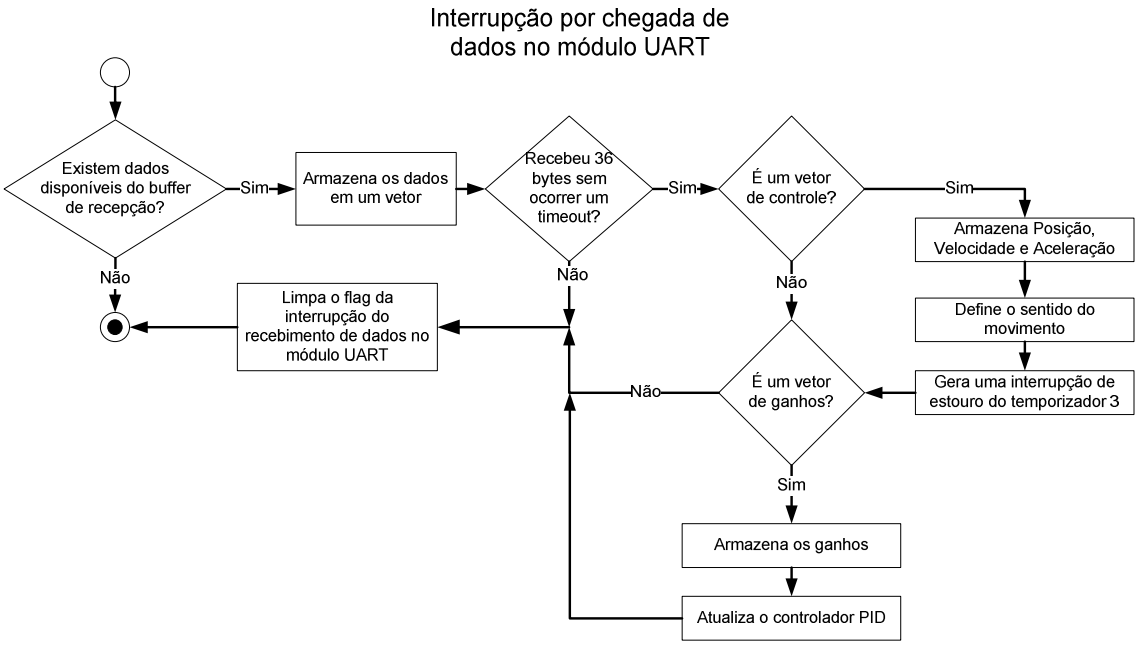

Figura 37 - Interrupção por chegada de dados no módulo UART.

Estouro do temporizador 1

Esta interrupção está relacionada com a atualização do registro temporário de posição e o cálculo da velocidade. A cada evento desses, o contador de posição é armazenado temporariamente em uma variável, onde é realizado um acoplamento dessas informações com o acumulador de posição, e ocorre a conversão em radianos. Após, a grandeza é armazenada

em uma posição do vetor de armazenamento temporário. Ao mesmo tempo, a posição anterior é armazenada em outra posição do mesmo vetor para o cálculo da velocidade, que é em radianos/segundos. O fluxograma geral de tratamento dessa interrupção é apresentado na Fig.38.

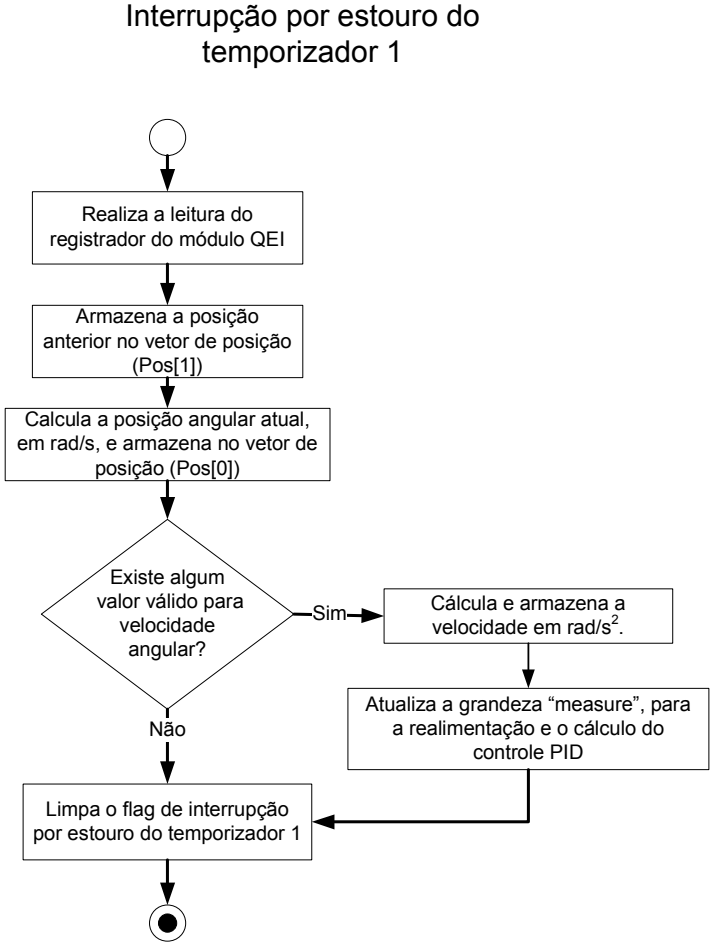

Figura 38 - Interrupção por estouro do temporizador 1.

Estouro do temporizador 2

Este evento é responsável pelo envio de informações para o computador através do módulo *UART*, o limite do temporizador determina a freqüência de envio e, conseqüentemente, de realimentação da lei de controle. Porém, esses dados ainda passam por um filtro de freqüência nas camadas superiores. A cada ocorrência desta interrupção, os dados

são desconcatenados e armazenados na variável de envio, para após serem transferidos. O fluxograma de atuação desse evento é demonstrado na Fig.39.

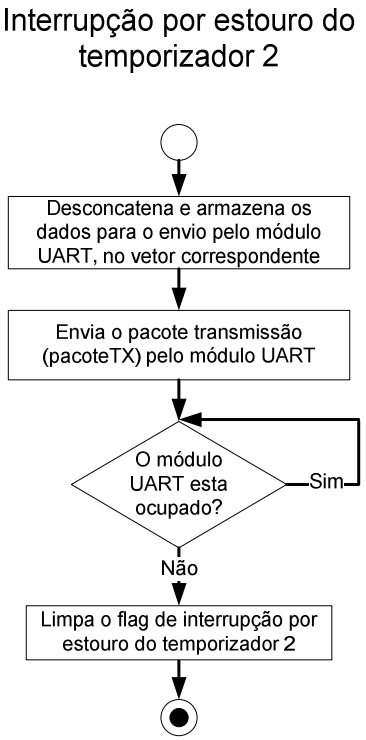

Figura 39 - Interrupção por estouro do temporizador 2.

Estouro do temporizador 3

O estouro do temporizador 3 realiza a atualização dos dados de controle do atuador, levando em consideração informações sobre a velocidade e a aceleração provenientes do vetor de controle *U*, recebido pelo módulo *UART*. Esse acionamento leva em consideração o tempo de estouro de temporizador, para gerar correntemente os pontos de aceleração, pois, temos uma aceleração discreta, ou ponto a ponto. O fluxograma completo de tratamento deste evento pode ser visualizado na Fig.40.

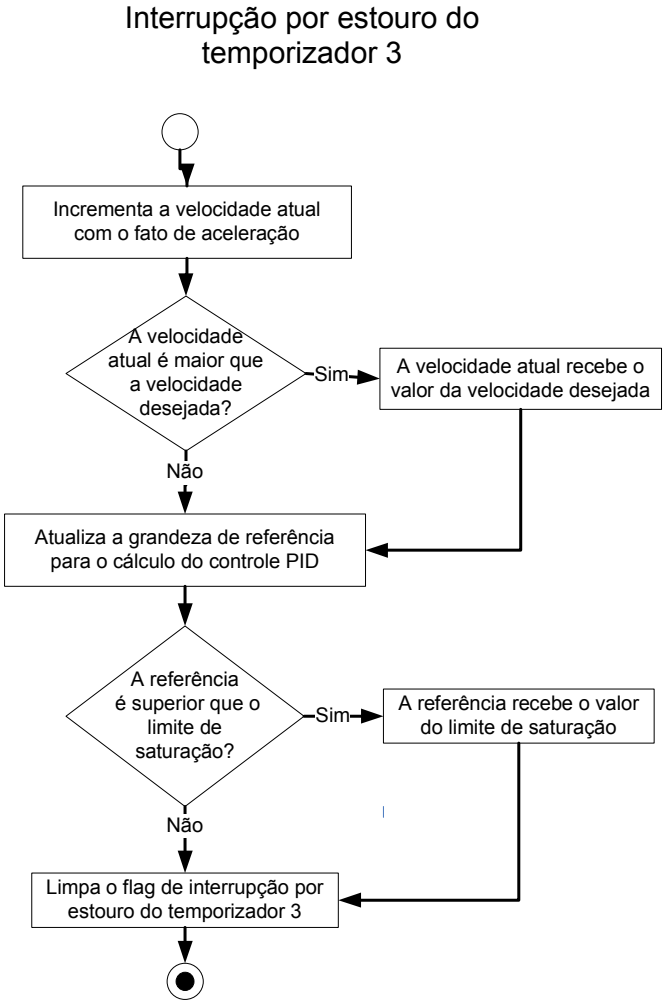

Figura 40 - Interrupção por estouro do temporizador 3.

Estouro do contador do módulo *QEI*

O módulo *QEI* é responsável pela decodificação dos sinais provenientes do *encoder* incremental de quadratura, sua contagem é armazenada em um contador que ao estourar causa este evento de interrupção. Na seção 4.4.2.3, foi explicado esse evento mais detalhadamente, justificando sua importância, pois, está associado com a ampliação da resolução de leitura do *encoder* para 32 bits, utilizando a acumulação em um contador de maior capacidade.

Neste procedimento todos os referentes contadores e armazenamentos temporários são reiniciados ou atualizados, conforme demonstrado no fluxograma de tratamento desse evento, Fig.41.

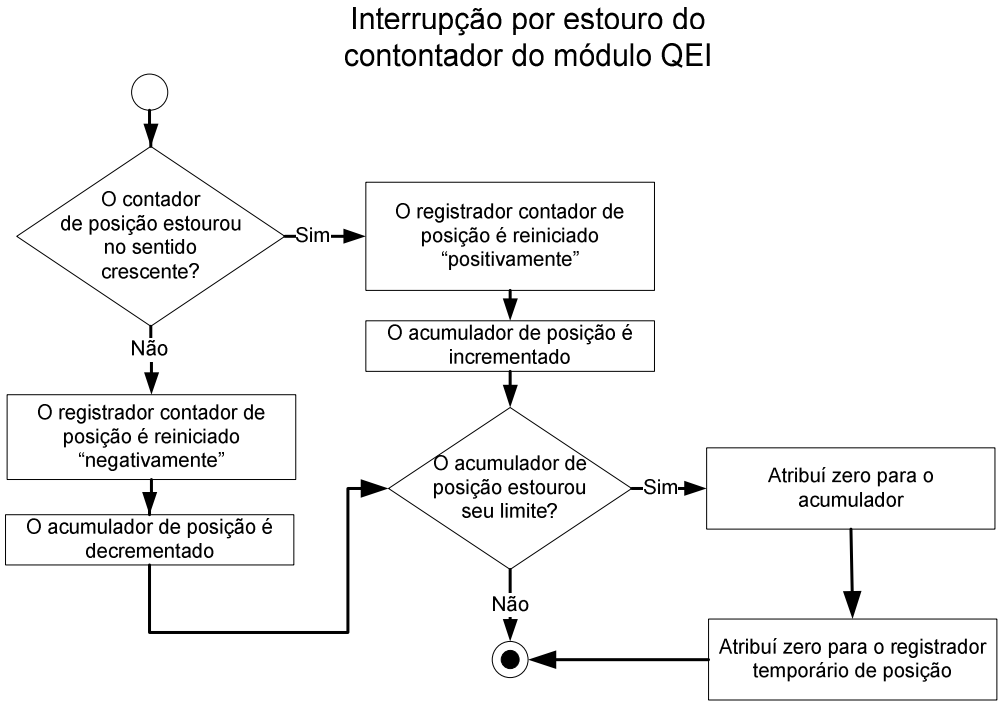

Figura 41 - Fluxograma da interrupção por estouro do contador do módulo QEI.

Evento externo 1

Devido à utilização do modo especial de interrupção para o módulo de controle de *encoders* (*QEI*), ficou impossibilitado o uso do sinal de *index* do mesmo módulo. Assim, o evento externo 1 é referente a ocorrência de um sinal desse. O tratamento desse evento é demonstrado na Fig.42.

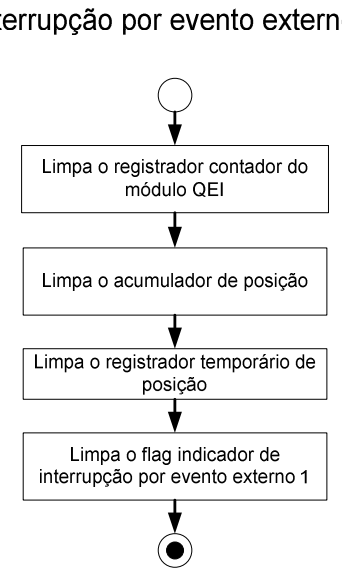

Figura 42 - Interrupção por ocorrência de evento externo 1.

Evento externo 2

Devido à grande periculosidade do sistema, é extremamente necessário um rápido tratamento no caso de emergências, esse evento externo é referente ao pressionamento do botão de emergência, suas ações estão descritas em seu fluxograma apresentado na Fig.43.

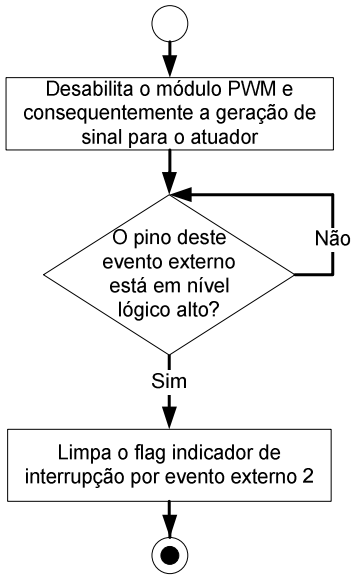

Figura 43 - Interrupção por ocorrência de evento externo 2.

Após ter sido detalhado o *firmware* do sistema, será apresentada a parte física, i.e., o *hardware*, que possibilita o funcionamento do algoritmo apresentado anteriormente.

4.4.3 *Hardware*

Para a integração da camada de comunicação com a camada física, foram desenvolvidos *hardwares*, que dentre outras qualidades processam *softwares* embarcados (*firmware*), não apenas, realizando uma interface dos sinais, mas sim, um processamento dos mesmos, decodificando as informações provenientes das camadas superiores, realizando os devidos cálculos e manipulações de dados, e controlando os dispositivos da camada inferior. Desenvolveu-se uma arquitetura modular, com um processador dedicado para cada junta do manipulador visando uma facilidade de manutenção e expansão, com um alto desempenho. O *layout* do sistema embarcado denominado de controlador de movimentos pode ser visualizado na Fig.44.

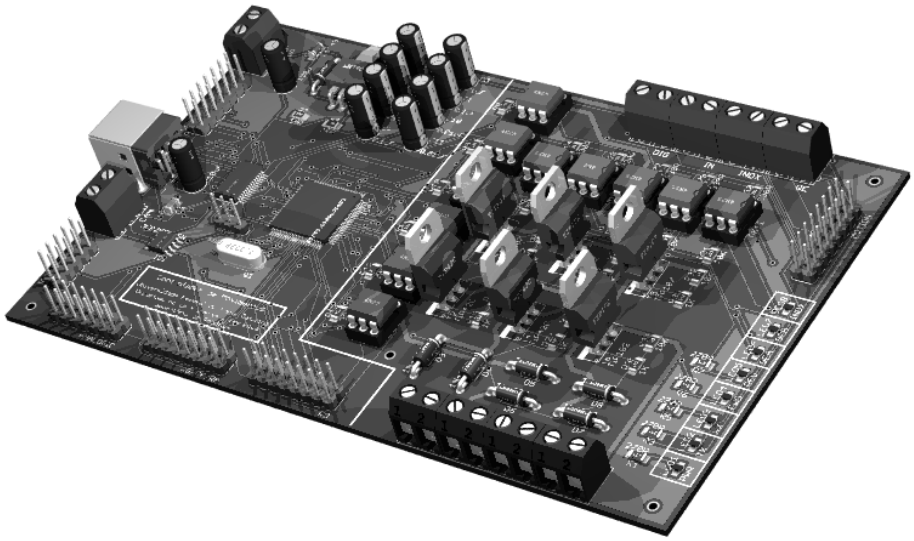

Figura 44 - Controlador de movimentos.

Resumindo, é um sistema embarcado regido por um controlador de sinais digitais (*DSC*), com uma alta capacidade de processamento, pois é preparado para o processamento de sinais digitais. Integrante de uma família de componentes modelados para o controle de

motores, contendo em seu interior, componentes preparados para essa tarefa. Ainda tendo amplas memórias de programa e dados, promovendo a capacidade de armazenar um *firmware* extenso e complexo. Aliado a esta característica, contém bibliotecas avançadas que elevam a abstração no desenvolvimento do *software* e manipulam componentes importantes no desenvolvimento do controle robótico, e.g., matrizes de dados. A seguir, serão explorados individualmente os blocos de componentes do controlador de movimentos.

4.4.3.1 ***Alimentação***

Sendo um dispositivo eletrônico, seu funcionamento esta associado a passagem de corrente elétrica controlada. Desta forma, iniciamos a descrição por sua unidade de alimentação (Fig.45).

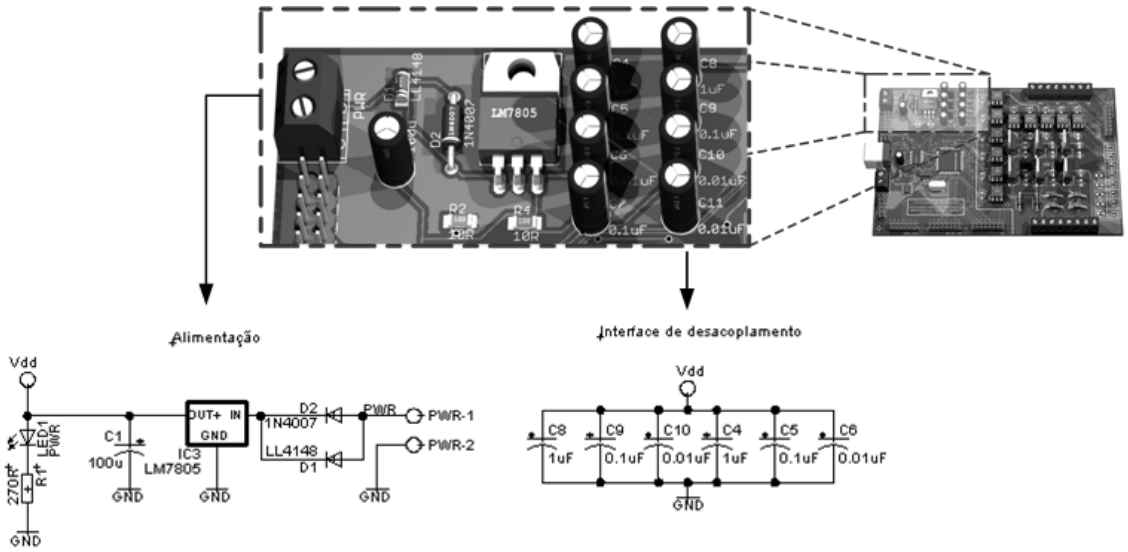

Figura 45 - Alimentação.

Inicialmente, após o conector, temos um diodo de proteção, para garantir o sentido da corrente, no caso de haver algum fluxo indevido de energia. Existem dois diodos para esta função que podem ser escolhidos, levando em consideração seu encapsulamento e sua capacidade de permitir a passagem de corrente elétrica. O controle tensão do sistema se dá, pela inclusão de um regulador de tensão, que possibilita que a alimentação do sistema seja realizada com até 35V. Porém a sua saída para o restante do circuito é saturada em 5V. Esta

parte do circuito, ainda contém um diodo luminoso (*LED*) para a sinalização de alimentação no sistema. Ainda está incluso um capacitor para a filtragem inicial das inconformidades da corrente elétrica do sistema.

4.4.3.2 ***Comunicação***

A próxima parte a ser abordada é o bloco de comunicação do sistema. Cujo funcionamento é de primordial importância, pois este realiza a troca de informações com os demais processadores do sistema. Têm-se duas formas de comunicação empregadas, através do barramento *CAN* e da interface *USB*. Ambas, tiveram seu funcionamento previamente discutido nesta obra. O diagrama da fatia de comunicação dos controladores de movimento é apresentada na Fig.46, juntamente com o seu diagrama elétrico (esquemático).

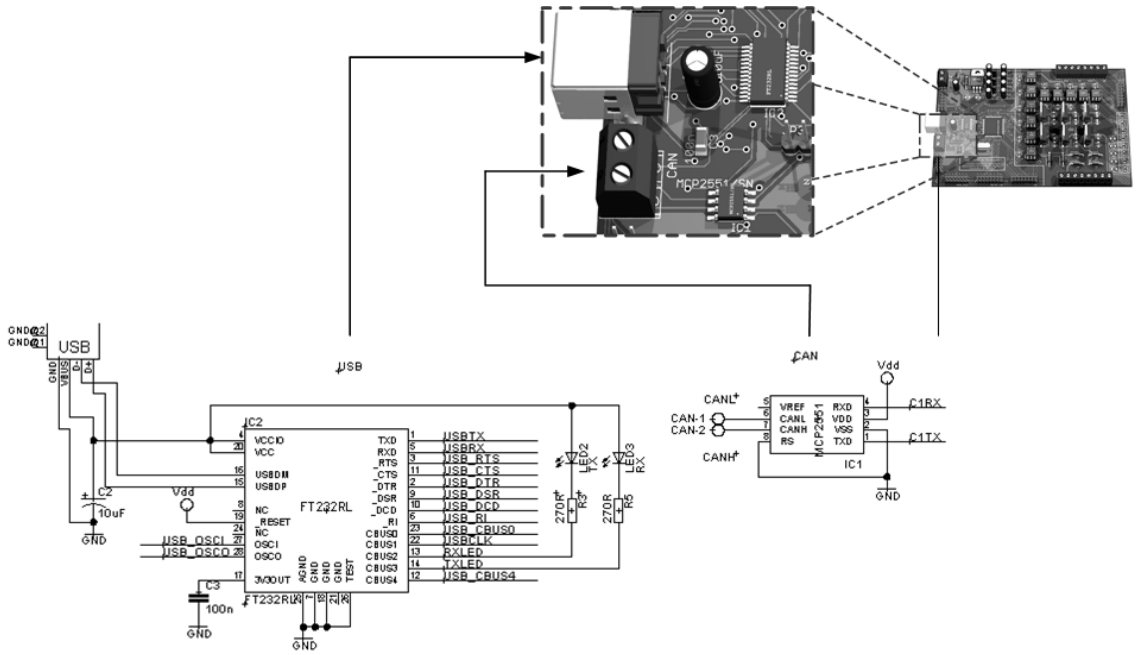

Figura 46 - Comunicação.

A interface *USB* tem como funcionalidade a mudança de sua visualização lógica regulada pelo seu *driver*[13]. Assim, ela pode assumir diferentes funções e aspectos. No caso dos controladores de movimentos, sua "mutação" é dada para uma porta de comunicação serial padrão, muito comum em computadores pessoais (denominadas de *COMx*). Ao se conectar esta interface em um computador pessoal é observada, apenas, a inclusão de um novo dispositivo, que após os procedimentos de instalação assume simplesmente a forma de uma porta de comunicação. Sua implementação é dada pela utilização do conversor *USB/UART* denominado de *FT232R* da *Future Technology Devices International Ltd.* Esse dispositivo é uma evolução de outros componentes bastante difundidos comercialmente, que tem como vantagem ter em seu interior, a maioria dos componentes necessários para o seu funcionamento. Além de realizar esta tradução das informações, ele ainda dispõe de uma memória de armazenamento e vias de entrada e saída que podem ser configuradas para atuações bem diferenciadas. Uma dessas portas foi utilizada para a geração de um sinal digital oscilador externo com freqüência regulável, que pode gerar a referência de tempo para o controlador de sinais digitais.

As antigas portas de comunicação serial perderam espaço principalmente por sua limitação e velocidade, que era dada por seus componentes *drivers/receptor*[14]. Ao retirá-los, o que acontece na implementação com a interface *USB*, pode-se atingir velocidades muito superiores. No caso deste componente, a comunicação pode alcançar uma taxa de transferência de até 3 Mbps. O fabricante disponibiliza *drivers* "*Royalty-Free*" para o acesso ao dispositivo por diversos sistemas operacionais.

O controlador de sinais digitais utilizado tem em seu interior dois módulos específicos para a manipulação do barramento *CAN*, necessitando apenas do acréscimo de um *transceiver*[15] de alta velocidade efetivar a comunicação. Suportando taxas de transferência de 1 Mbps. A adição deste, garante ao barramento os requisitos definidos pela *ISO-11898*, *Road vehicles: Controller Area Network* (*CAN*), para a camada física.

---

[13] *É um software, geralmente desenvolvido pelo fabricante do dispositivo, que possibilita a comunicação do sistema operacional com o periférico.*

[14] *São componente ques traduzem os sinais recebidos para os níveis de tensão correspondentes.*

[15] *Consiste em uma associação de transmissor e receptor, com um driver conversão de níveis de tensão.*

4.4.3.3 ***Controlador de sinais digitais***

O componente principal deste sistema é um controlador de sinais digitais (*DSC*), modelo *dsPIC30F6010A*, produzido pela *Microchip Technology Inc*. Um componente com um conjunto de instruções de 16 *bits*, atingindo uma freqüência de operação máxima de 120Mhz e 30*Mips*[16]. Sendo encapsulado em uma pastilha no formato *TQFP* de 80 pinos. É integrante da família de controle de motores, desta forma, possui em seu interior um conjunto de módulos periféricos altamente especializados para esta finalidade. Ainda, possui uma grande quantidade de portas de *I/O* e amplas memórias de programa e dados, tornando-se um dos componentes mais completos de sua categoria.

Como todo sistema embarcado, este necessita de um sinal oscilatório que o forneça a noção de tempo. Para isso foi incluso ao sistema, algumas possibilidades para a geração dessa onda quadrada. A seleção é realizada por meio de um *jumper*[17], de dois estágios. Possibilitando a seleção de um oscilador externo proveniente de um cristal ou do componente *FT232R*, que gera um sinal oscilador digital com freqüência configurável. Internamente, este ainda contém um cristal de 7.37MHz. Ambas as formas de utilização podem utilizar ainda o multiplicador (*PLL*) e o divisor de *clock*, porém devem respeitar os limites do componente detalhados anteriormente. O controlador de sinais digitais e os modos de oscilador podem ser visualizados na Fig.47.

---

[16] Grandeza que define a velocidade de processamento em milhões de instruções por segundo.
[17] É uma ligação "removível" entre um circuito eletrônico.

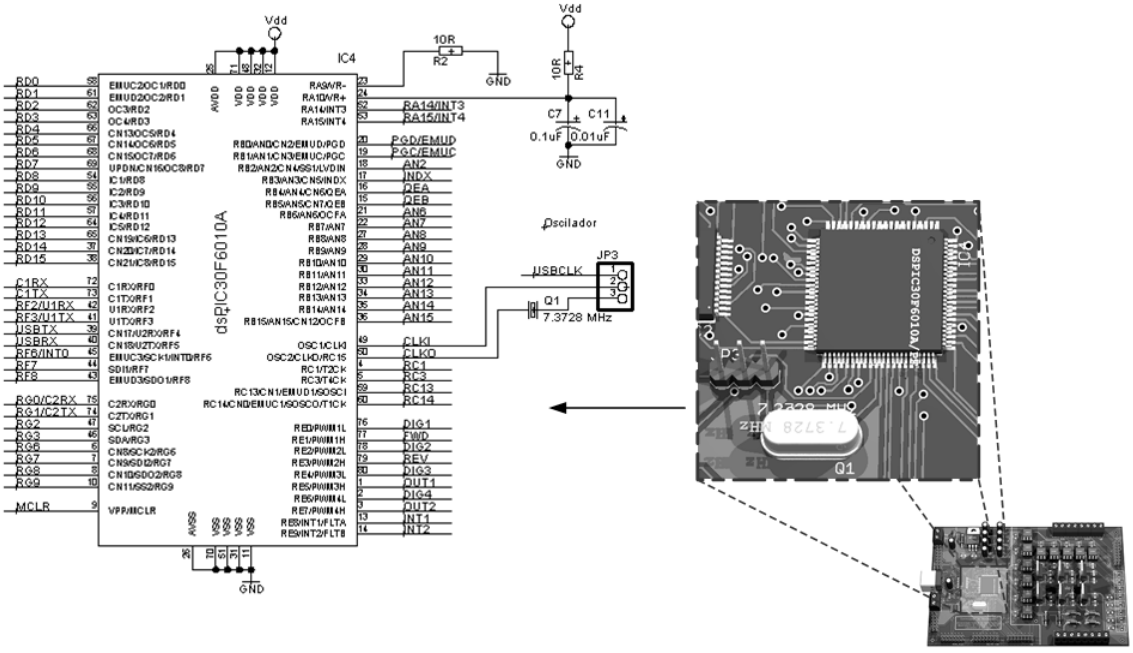

Figura 47 - Controlador de Sinais Digitais.

4.4.3.4 *Entradas*

Os controles de movimentos são compostos por um conjunto de entradas protegidas, para o acoplamento dos sinais do *encoder* de quadratura e de demais sensores que compõem uma junta robótica, e.g., sinal de *home*[18]. Para que nenhum sinal inadequado alcance o resto do sistema, principalmente o controlador de sinais digitais, as entradas passam por uma barreira de desacoplamento ótico, composta por optoacopladores.

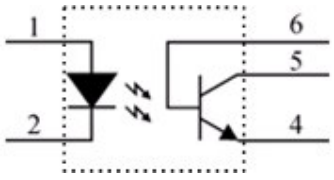

Figura 48 - Arquitetura interna de um optoacoplador.

O optoacoplador é um conjunto composto por: um diodo emissor de luminosidade (*LED*) e um optotransistor, apresentado na Fig.48. Quando o *LED* recebe a corrente

[18] *Sinal que identifica a posição inicial do manipulador, geralmente onde este está em um equilíbrio de forças.*

necessária para a geração de sua luminosidade, o optotransistor permite a passagem de corrente na outra extremidade. O conjunto é geralmente encontrado encapsulado em uma pastilha de silício, similarmente a outros circuitos integrados. No caso de qualquer falha, estes componentes serão os primeiros a ser danificados. Por isso, foi selecionado um encapsulamento muito comum em qualquer eletrônica, montado sobre um *socket*[19] de encaixe fácil. Fornecendo um desacoplamento físico ao sistema, pois o conjunto não possui ligação física.

As entradas cujo sinal é proveniente do *encoder* de quadratura após passar pela barreira de desacoplamento ótico são ligadas diretamente ao módulo de interface de *encoders* (*QEI*). Os demais sinais são conectados á módulos geradores de interrupção por evento externo. Ainda, existem algumas entradas sem o desacoplamento ligadas diretamente ao *DSC*, que também podem operar como saídas. O diagrama das entradas está apresentado na Fig.49.

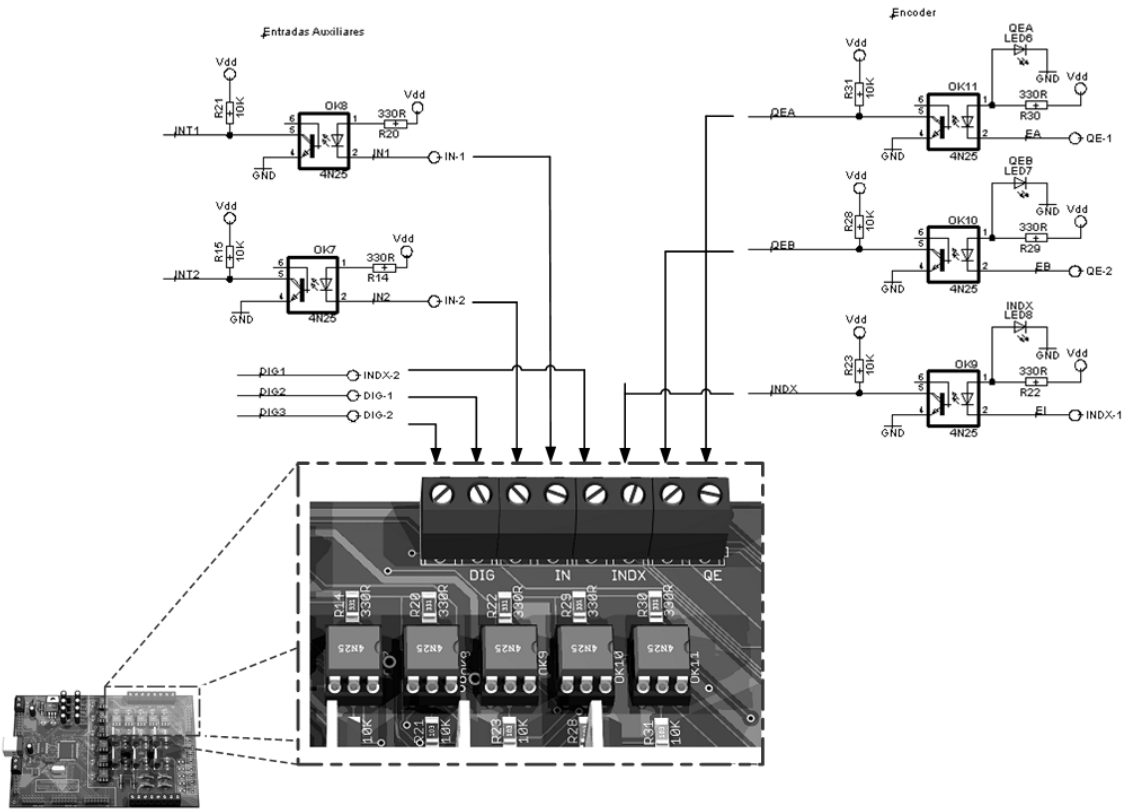

Figura 49 - Entradas.

[19] *Designação utilizada para um conector, que realiza uma conexão física, permitindo a passagem de corrente elétrica.*

4.4.3.5 ***Saídas***

O principal sinal de saída do sistema é o que rege o atuador das juntas do manipulador. Isso é realizado através do módulo *PWM*, cujo funcionamento foi previamente explicado. Novamente para prevenir qualquer retorno inadequado a parte de controle do sistema é utilizada uma barreira de desacoplamento ótica, composta por optoacopladores. O sinal modulado aciona um conjunto de transistores *darlington*[20] de uso industrial (operando no modo de chave) ligados em uma estrutura de ponte-H. A arquitetura de transistores formando uma ponte-H (Fig.50) fornece ao sistema a possibilidade do controle do sentido da corrente e conseqüentemente do atuador. Os transistores ainda possibilitam o uso da técnica de modulação por largura de pulso, devido principalmente ao seu tempo de reposta, e podem operar com grandes taxas de tensão e corrente elétrica, no caso dos utilizados, até 100V e 8A. Se utilizada com bastante cuidado, pode-se ainda ter algumas funções avançadas, como freios.

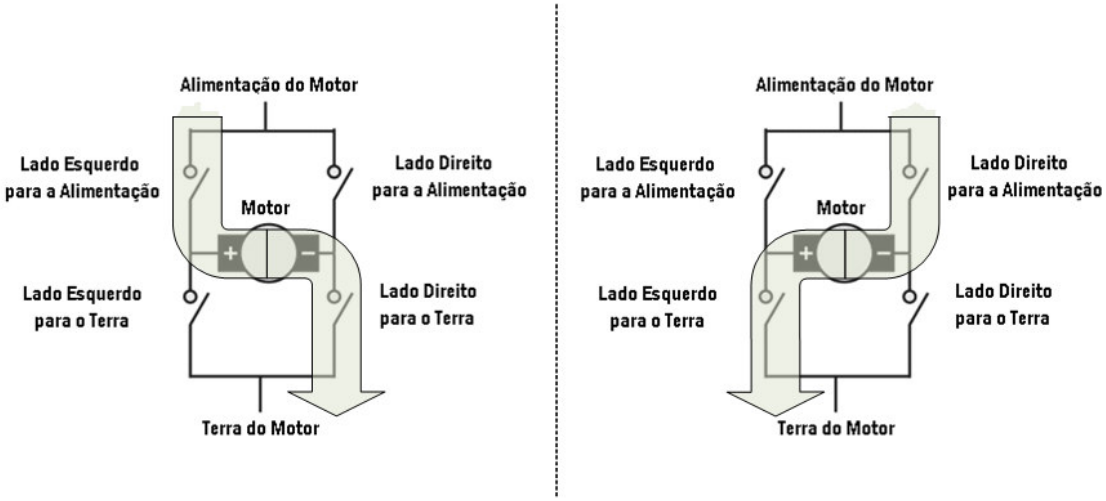

Figura 50 - Ponte-H.

Além deste conjunto de saídas amplificadas, ainda existem duas saídas auxiliares. Estas, igualmente isolada oticamente, são acionadas por transistores puros (sem nenhuma forma de acoplamento) e protegidos por diodos externos. Suportam cargas de até 100V e 6A e

[20] *Acoplamento de vários transistores em uma mesma pastilha, o que proporciona um aumento de corrente pela arquitetura e ocupa um menor espaço. Alguns ainda contêm em seu interior diodos de proteção.*

também está ligados a saídas *PWM*, cujo sinal pode ser modulado ou não. A alimentação de todas essas saídas se dá externamente, desta forma pode-se alimentar com qualquer tensão dentro da faixa de operação dos transistores. O diagrama elétrico do conjunto de saídas amplificadas é exposto na Fig.51.

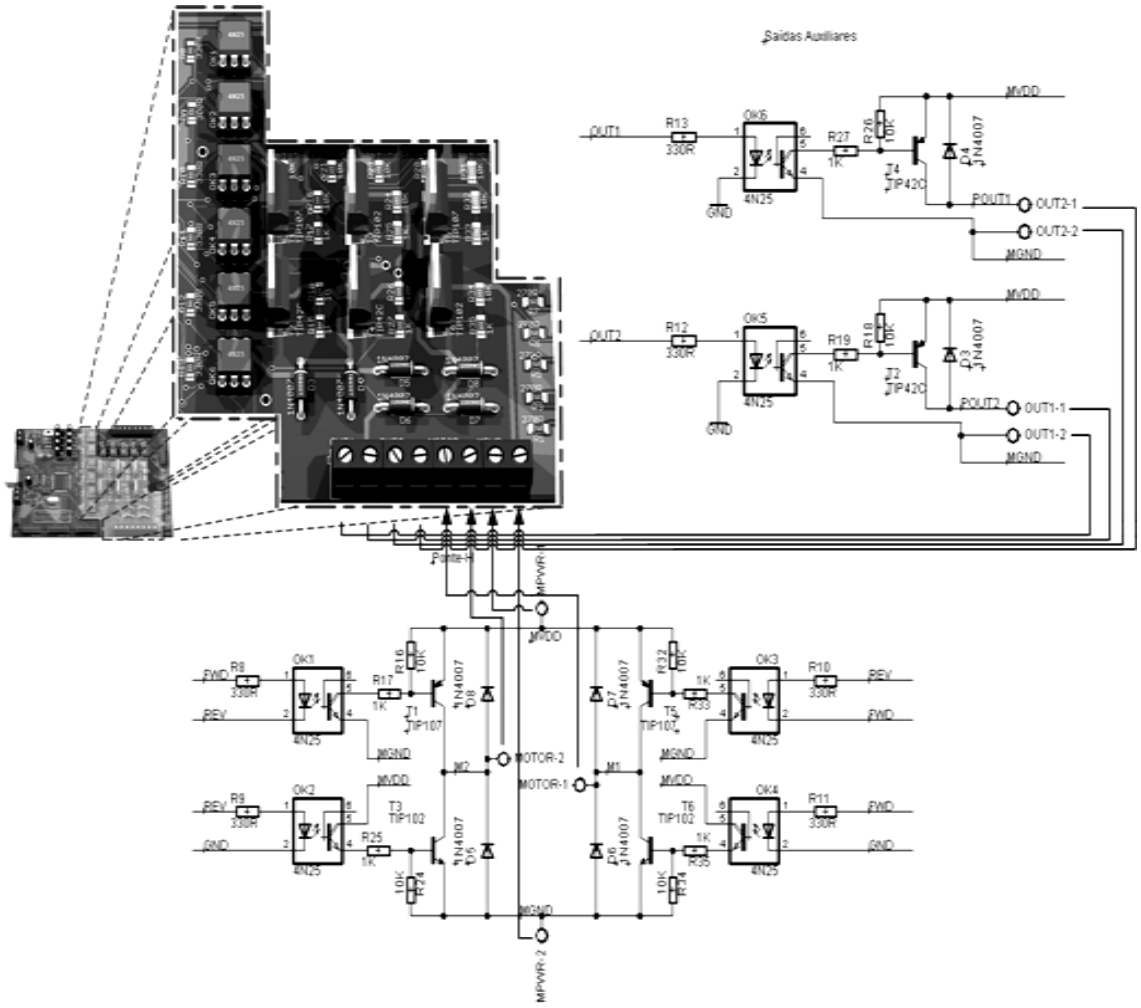

Figura 51 - Saídas.

As saídas e as entradas amplificadas que passam pela barreira ótica, tem suas extremidades compostas por conectores de engate rápido a cabos. Os quais necessitam ser parafusados individualmente, para o caso de serem ligados diretamente aos dispositivos lidos e acionados. Caso, o controlador de movimentos fique em algum tipo de gabinete (como na implementação realizada) e passe por fios e cabos, podem-se utilizar o conector "*motion*", que é um espelhamento dos sinais desses conectores para um cabo *flat* de dezesseis vias.

Ressaltando que nesse tipo de implementação os limites de carga do fio devem ser levados em consideração. Esse espelhamento em um conector de cabo flat pode ser visualizado na Fig.52.

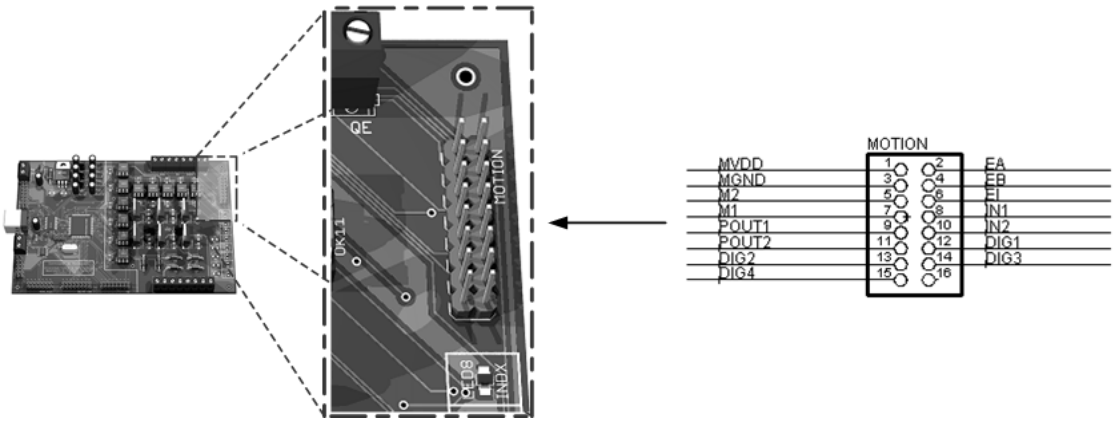

Figura 52 - Conector para cabo flat.

4.4.3.6 ***Expansão***

O controlador de sinais digitais (*DSC*) e o conversor *USB/UART* utilizados contêm uma série de funcionalidades e periféricos que não foram utilizados inicialmente neste projeto e também não foram descartados, dessa forma, foram todos disponibilizados em conectores *flat* (Fig.53).

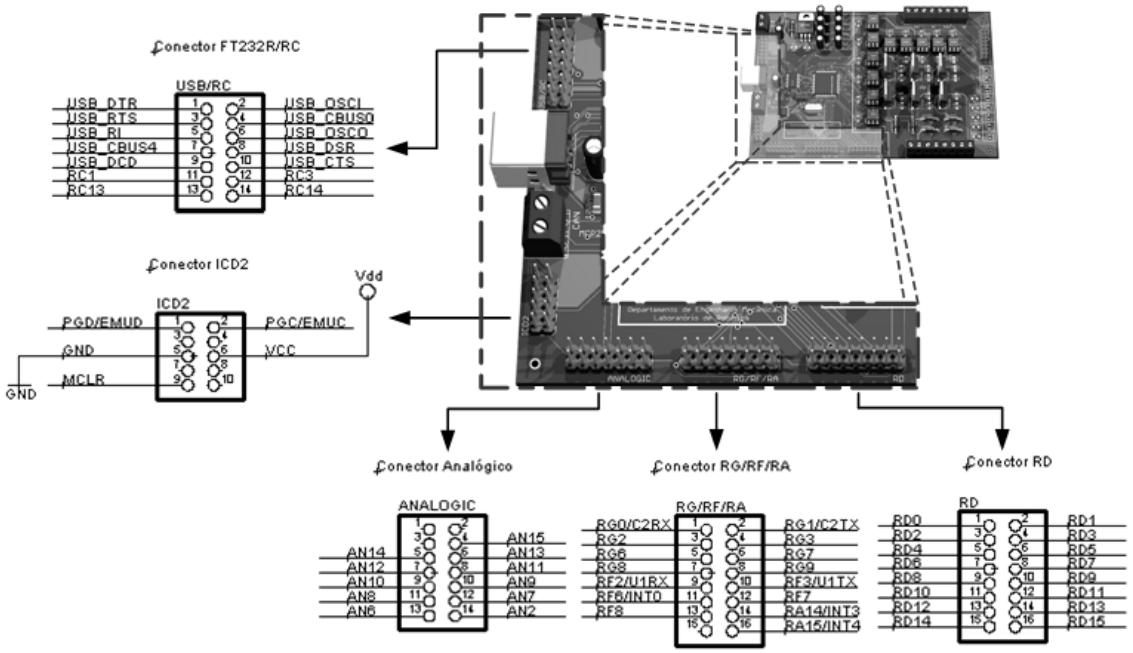

Figura 53 - Expansão.

No caso do *FT232R*, ainda são oferecidos pinos com funcionalidade configurável e os demais sinais para uma comunicação mais aprimorada. Para o controlador de sinais digitais, existe o conector de programação do componente, caso de deseje utilizar algum programador específico. Ainda existe uma série de portas de entrada e saída digital, portas de entrada analógica. Dentre essas ainda estão diversos módulos de comunicação e interrupção.

4.5 Camada Física

A camada mais inferior, aqui denominada de camada física, compreende na unidade de potência, integrante do gabinete de controle do manipulador e no robô industrial. O qual será detalhado a seguir.

4.5.1 *Robôs REIS Rv15*

O manipulador industrial utilizado é do modelo *Rv15* produzido pela *REIS Robotics*, com seis graus de liberdade, demonstrado na Fig.54. Foi acoplado à sua extremidade um

sensor de força produzido pela *JR3 Inc*. para realizar a realimentação de força no sistema. Ao mesmo tempo, o robô está sendo atualizado tecnologicamente com a substituição do antigo controlador proprietário pelo controlador de arquitetura aberta desenvolvido.

O robô *REIS* tem seis juntas rotativas atuadas por motores elétricos de corrente contínua e para a medida das posições angulares são utilizados *encoders* óticos incrementais. Este manipulador possui uma topologia muito comum nas aplicações industriais, sendo constituído por um braço antropomórfico (juntas 1,2 e 3) com um punho esférico (juntas 4,5 e 6).

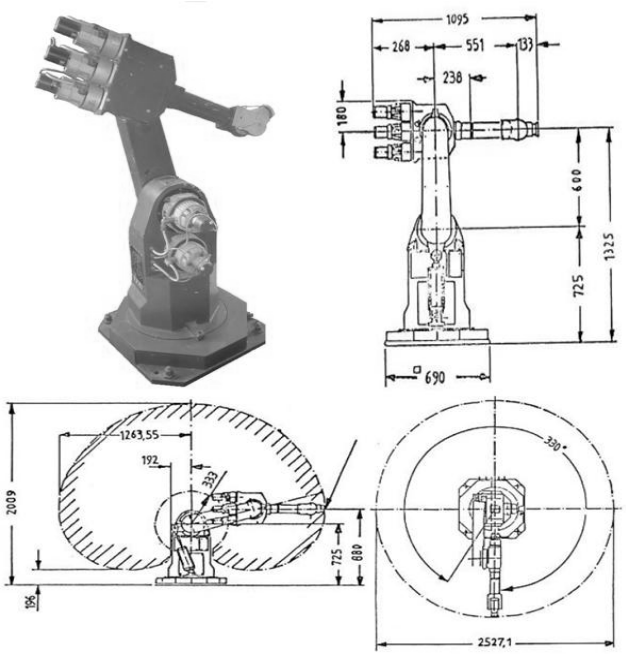

Figura 54 - Robô REIS Rv15 (dimensões em milímetros).

4.5.2 *Motores*

O manipulador é composto por seis motores (Fig.55) de dois tipos distintos. As juntas iniciais ou inferiores, que realizam o posicionamento, movimentam uma carga mais elevada, dessa forma, são motores com uma potência mais elevada. As juntas superiores, que realizam a orientação do efetuador, são compostas por motores de dimensões e potências inferiores. Ambos os atuadores possuíam uma documentação escassa, porém algumas características puderam ser levantadas e são apresentadas na Tabela 4. As juntas 2 e 3 são no sentido gravitacional e possuem um cilindro pneumático para a compensação dessas forças, que deve

estar com uma pressão de 5 Bar para a operação. O *retrofitting* realizado neste manipulador incluiu o desenvolvimento de uma linha pneumática fixa para a alimentação desse cilindro.

Tabela 3 - Características dos motores elétricos do robô REIS Rv15.

Fabricante	Mavilor Motors S.A.	
Tipo	Motor *DC*	
Juntas	1,2 e 3	4,5 e 6
Modelo	800	200
Tensão	106 Volts	39 Volts
Velocidade	3.000 RPM	3.000 RPM
Potência	826 Watts	200 Watts
Corrente	9.2 A	6.85 A
Velocidade Máxima	6.000 RPM	8.000 RPM

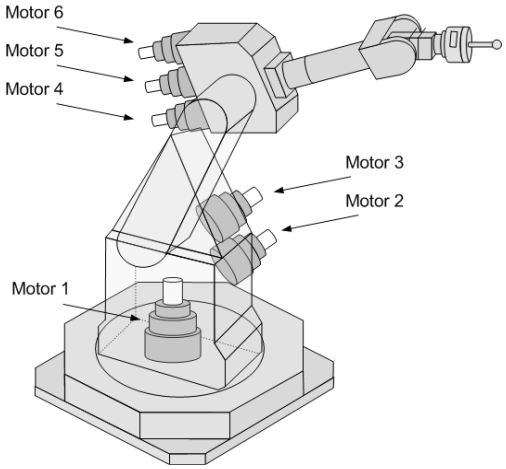

Figura 55 - Identificação dos atuadores no robô REIS Rv15.

O acesso aos atuadores não é realizado diretamente, pois os sinais de atuação regem a unidade de potência e estas, adéquam os sinais aos atuadores. A unidade de potência do manipulador *REIS*, é antiga e não foi encontrada documentação sobre ela. Em Taouil, (2004), é dito que a unidade de potência opera com sinais de -5V á +5V, porém na prática, a faixa de operação é de -1.7V à +1.7V. Acima desses limites, a unidade de potência desliga-se, como uma medida de segurança. Essa faixa promove velocidades satisfatórias dentro de um limite de segurança.

A unidade de potência opera com um controlador para manter o sinal de saída proporcional a entrada. Porém, seus ganhos estão desregulados e aparentemente as chaves de

ajuste manual dos ganhos não estão realizando nenhuma alteração na estrutura de controle. Isso acarreta em flutuações da saída em movimentos longos com a mesma junta principalmente a altas velocidades.

4.5.3 *Encoders*

Acoplados anteriormente aos motores do manipulador encontram-se *encoders* óticos incrementais com resolução de 16 *bits*, contendo 1000 furos por revolução. Assim, utilizando o método de leitura quadruplicada de pulsos, tem-se 4000 pulsos por giro do eixo. Porém ainda existe a redução conectada posteriormente ao motor que tem uma taxa de 100:1, dessa forma, a resolução do *encoder* é amplificada, tendo 400.000 pulsos por revolução, conformo pode ser visto na Fig.56.

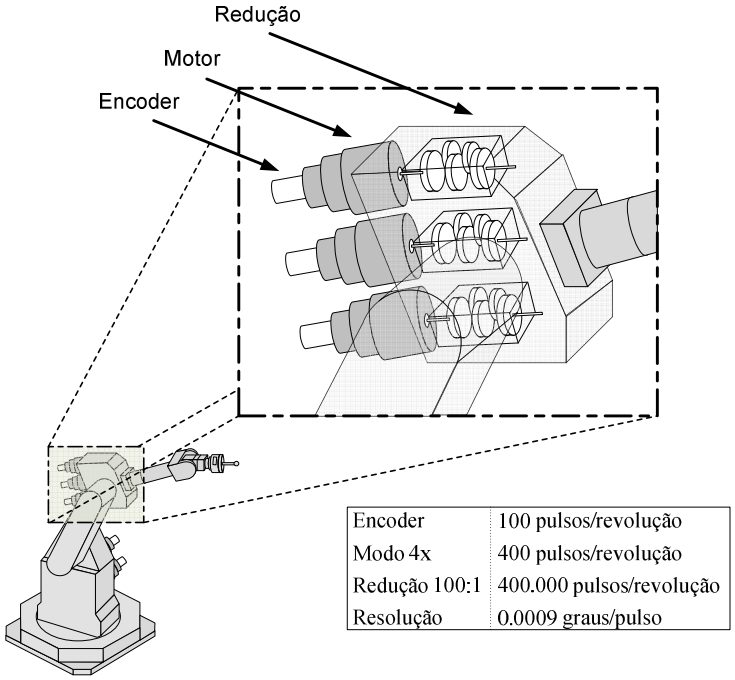

Figura 56 - Sistema de atuação do manipulador REIS Rv15.

Cada junta do manipulador possui seus limites de movimento, que definem o número máximo de pulsos, apresentados na Tabela 4.

Tabela 4 - Limites de junta do manipulador REIS Rv15.

	Ângulo		Pulsos
Junta 1	319º	5.567 rad	354.000
Junta 2	135º	2.356 rad	150.000
Junta 3	270º	4.712 rad	300.000
Junta 4	360º	6.282 rad	400.000
Junta 5	229º	3.996 rad	254.000
Junta 6	360º	6.282 rad	400.000

4.5.4 *Sinal de configuração inicial*

A configuração inicial do robô é um conjunto de posições de junta, que determinam o posicionamento do manipulador para um estado de equilíbrio de forças. No caso do robô *REIS*, existe um sinal digital que determina essa posição. Porém, é necessário realizar um procedimento para a determinação desta configuração, que é descrito na Fig.57.

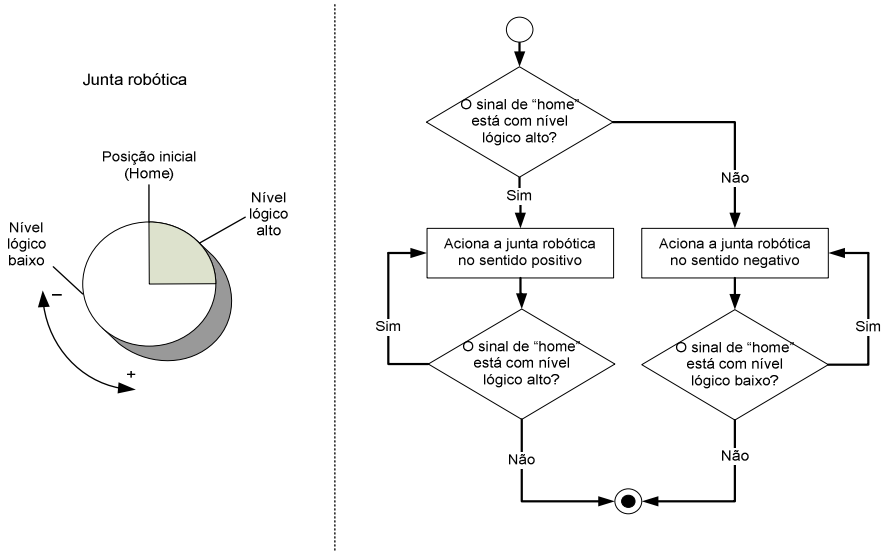

Figura 57 - Procedimentos para a detecção da configuração inicial.

O manipulador tem dificuldades para a obtenção de sua configuração de origem, devido às juntas dois e três não terem um sinal de *home* adequado e principalmente a limitação lógica que teve de ser adotada a algumas juntas. As juntas no sentido gravitacional possuem instabilidade no seu movimento perto dos limites, o que ocasiona um desligamento automático pelo sistema de proteção da unidade de potência. Devido principalmente, a compensação inadequada da gravidade pelo cilindro pneumático.

Capítulo 5. Implantação no Robô Reis Rv15

Baseado na proposta de modelo de referência para controladores de robôs de arquitetura aberta foi desenvolvido um sistema controlador para o manipulador *REIS Rv15.* Retirando o sistema de controle proprietário, que contém uma série de informações obscuras devido as suas abstrações que em estruturas de controle avançadas de robôs podem ser necessárias, e incorporando um sensor de força ao sistema, promoveu-se uma modernização (ou o *retrofitting*) do manipulador, tornando o sistema apto a operações de controle de posição e força.

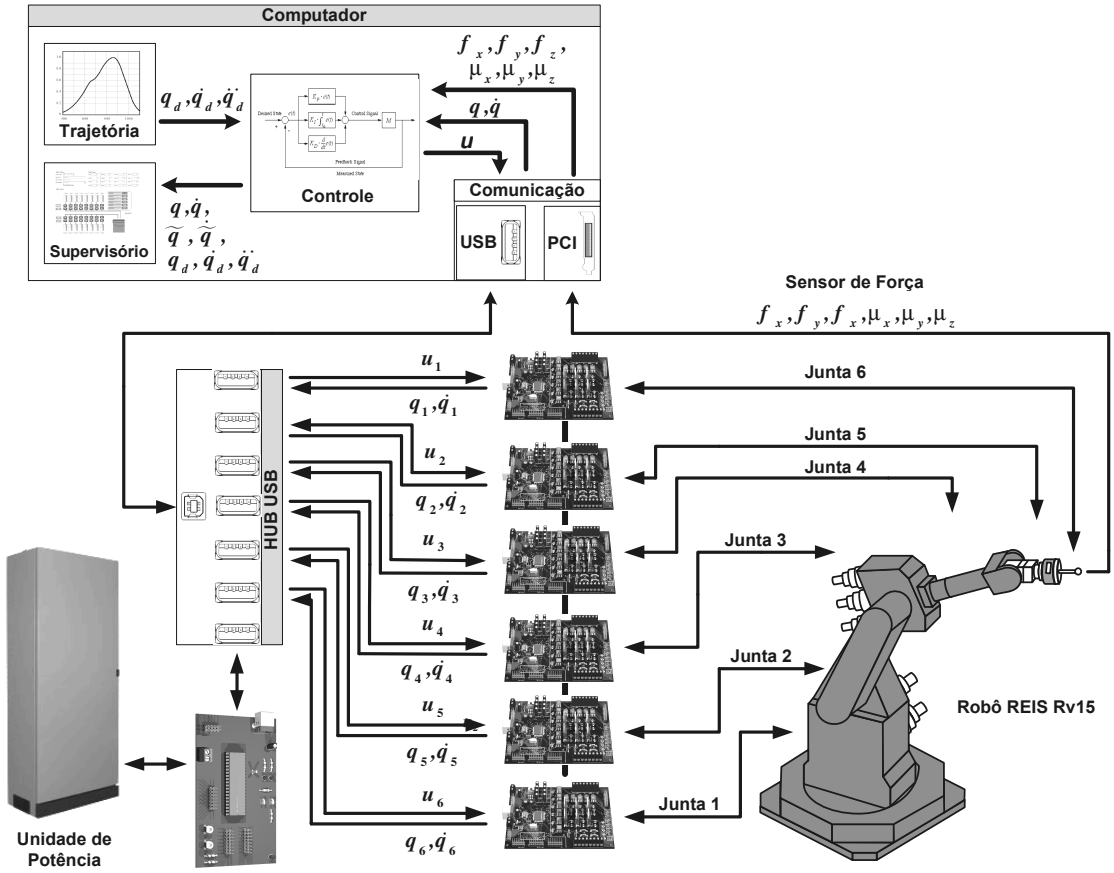

Figura 58 - Diagrama completo do sistema controlador desenvolvido.

A implantação do sistema de controle utilizou oito processadores, o que promove um alto grau de modularidade, entre eles: seis processadores dedicados, i.e., os controladores de movimentos, um computador e um processador que gerencia a unidade de potência e os

mecanismos de proteção do sistema. A Fig.58 apresenta o diagrama completo do sistema controlador aberto desenvolvido, ilustrando o fluxo de informações e as interconexões desde o computador (i.e., o nó central) até o manipulador industrial (i.e., os atuadores, *encoder* e o sensor de força).

Inicialmente foram realizados testes individuais com os atuadores e os sensores de deslocamento angular, os quais serão apresentados a seguir.

5.1 Experimentos com os Atuadores

Os testes iniciais do sistema foram realizados individualmente sobre os motores e *encoders*, de forma a garantir a confiabilidade e a corretude das informações em freqüências e posições distintas. Assim, os atuadores foram acionados de diversas formas e situações e os sinais dos sensores foram adquiridos com o auxílio de um osciloscópio, esses ensaios são apresentados na Fig.59 e na Fig.***60***, onde é vista a saída do *encoder* (neste caso, da junta 6) à uma atuação do motor, com velocidades de 4.79 *rad/s* e 0.8 *rad/s* respectivamente. Nota-se que a saída possui uma interferência regular, cujas causas não foram totalmente identificadas. Porém uma parte desta perturbação vem de uma barra de contatos antiga, interna ao manipulador, que deixa os contatos livres e próximos aos atuadores, sendo assim sensível a interferência eletromagnética.

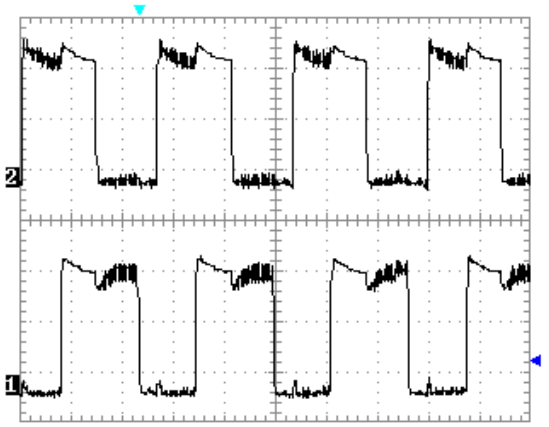

Figura 59 – Ensaio com o encoder, velocidade 4.79 rad/s.

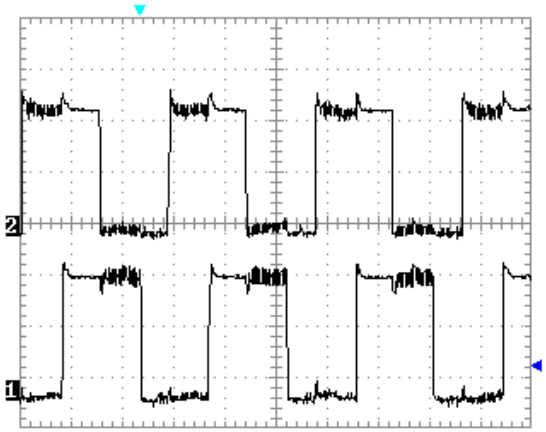

Figura 60 - Ensaio com o encoder, velocidade 0.80 rad/s.

Por ser um manipulador antigo, existe uma escassez de informações sobre o mesmo. Os dados levantados sobre os atuadores, mostrados na seção 4.5.2, não foram o suficiente

para a geração de um modelo matemático preciso. Assim, a dinâmica do atuador foi modelada experimentalmente. Foram realizados diversos acionamentos do atuador, ou seja, foram aplicadas diferentes freqüências de modulação por largura de pulso (*PWM*) e foi lida a velocidade gerada em *rad/s*. Os ensaios referentes aos dois modelos de atuadores são apresentados nas Fig.61 e na Fig.62.

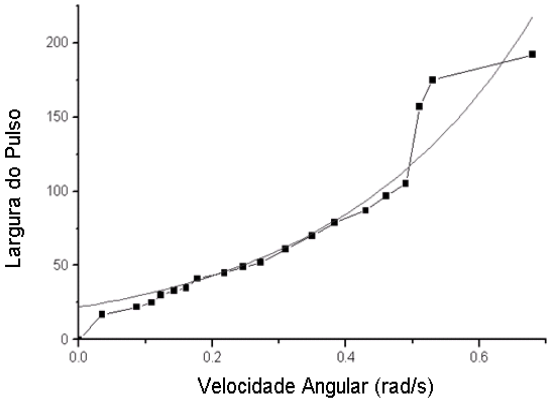

Figura 61 – Ensaio do motor 200.

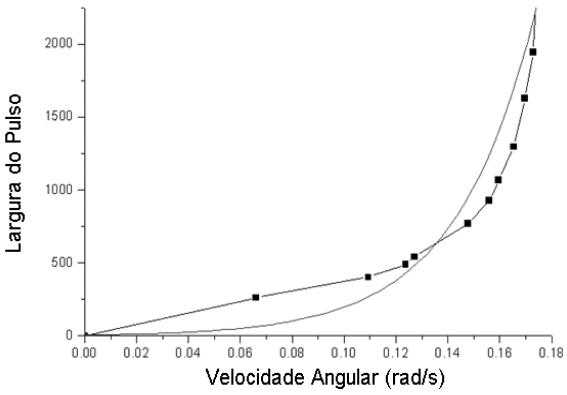

Figura 62 –Ensaio do motor 80.

O comportamento dos motores assemelhou-se a uma curva exponencial crescente que pode ser modelada conforme a equação (5.1).

$$y(t) = y_0 + A e^{\frac{x}{t}} \tag{5.1}$$

Onde,

x - Velocidade angular (*rad/s*)

y_0 - Largura do Pulso

Com a ajuda de um *software* de análise matemática, foi encontrada a melhor curva exponencial crescente que se adéqüe aos pontos obtidos experimentalmente, também apresentadas nas Fig.61 e na Fig.62. Dessa forma, foram obtidos os coeficientes que definem os modelos dos atuadores, apresentados na Tabela 5.

Tabela 5 - Coeficientes do modelo dos atuadores.

	y_0	A	t
Motor 200	0	80.29966	2.17527
Motor 800	0	21.96836	0.29649

Com a inclusão do modelo dinâmico dos atuadores, a saída aproxima-se do valor desejado, dessa forma, o controle *PID* não precisa realizar correções de grande amplitude. Porém os modelos foram realizados em relação aos tipos de atuadores e não em relação a dinâmica das juntas. Principalmente pelo fato de a maioria das juntas terem um ângulo de atuação muito limitado, dificultando o ensaio a velocidades elevadas. A Fig.63 apresenta os ensaios dos atuadores, com a utilização do modelo dinâmico e do controle *PID*. Salientando que os ganhos foram obtidos empiricamente para o sistema, ou seja, um mesmo fator de ganhos é aplicado a todos os motores. Nesta é possível visualizar que a junta 2 possui um perfil de saída mais distante do desejado, devido principalmente a cilindro pneumático que realiza a compensação gravitacional. Sua dinâmica não foi levada em consideração, o que acarreta atraso da resposta. O perfil oscilatório obtido na saída da junta 3 é devido a força gravitacional, visto que seu movimento é exatamente no mesmo sentido. Igualmente ao que ocorre na junta 5, pois nas condições do ensaios, está junta também estava alinhada com a força gravitacional. As demais juntas, seguiram satisfatoriamente o perfil de trajetória desejada.

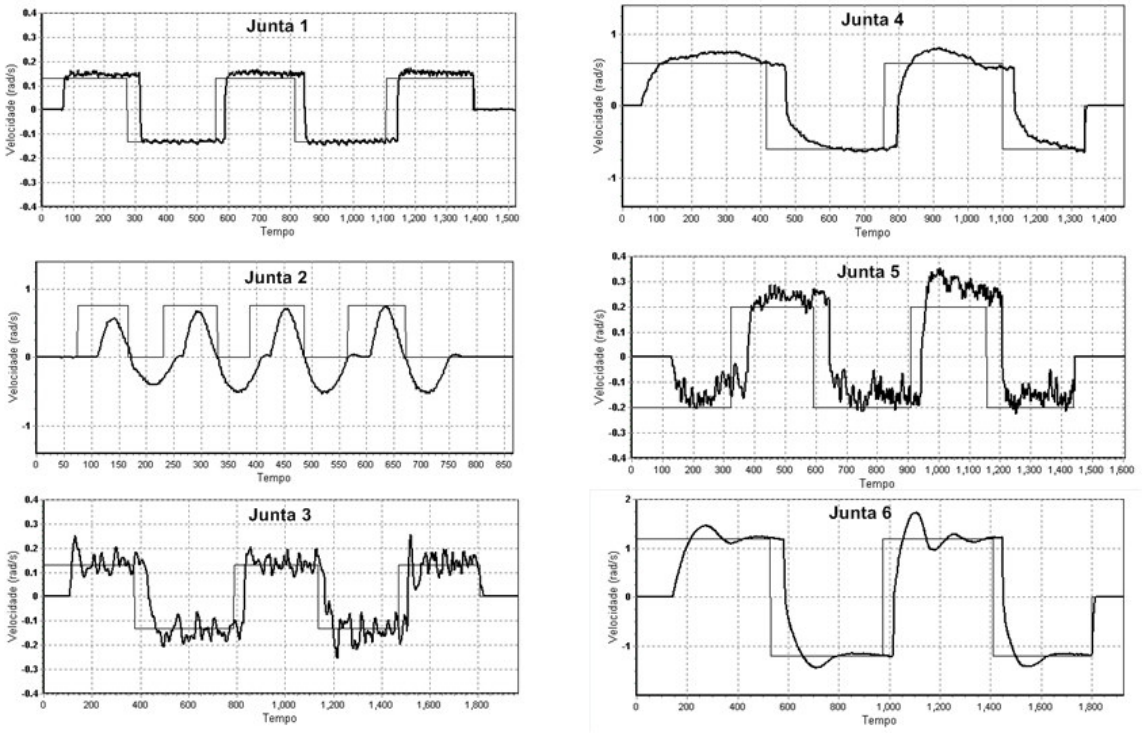

Figura 63 - Ensaios dos atuadores com controle PID.

Após apresentados os ensaios individuais das juntas, será mostrado o procedimento de validação do controlador.

5.2 Validação do Controlador

A validação do sistema foi realizada com a implementação de uma modalidade de controle indireto de força, o controle de impedância, apresentado na seção 2.2.2 . O controle por impedância trata indiretamente a força de contato, modelando a interação como um conjunto massa-mola-amortecedor. O relacionamento indireto ocorre pela forma de controle do sistema sobre o sinal de força, cujo objetivo é adequar o comportamento dinâmico do manipulador em contato com o ambiente. A realimentação de força fornece apenas a impedância do sistema em contato com uma superfície. Assim, a filosofia fundamental do controle de impedância, de acordo com Hogan, (1985), é que o sistema de controle regula a impedância do manipulador, que é definida pela relação entre a velocidade e a força aplicada (Zeng e Hemami, 1997).

Esta modalidade de controle utiliza a realimentação de força para regular a impedância do manipulador, admitindo que o mesmo já se encontre em contato com o ambiente, i.e., com alguma superfície. Dessa forma, sobre a ação de forças o mesmo regulará a impedância para atingir o perfil desejado.

A aplicação utilizada para validar o controlador para tarefas de controle de posição e força utiliza essa característica do controle de impedância, mas não coloca o efetuador em contato com uma superfície. Assim, o manipulador encontra-se imóvel, admitindo estar sobre seu perfil de impedância desejado, e quando sobre a ação de forças, que neste caso será a manipulação do transdutor de força com a mão, o manipulador adéqua sua impedância, movimentando-se apenas quando é exercida alguma força manual sobre o efetuador (Fig.64), e conseqüentemente sobre o sensor de força. Quando alguma força é exercida sobre outra parte do manipulador o mesmo permanece imóvel (Fig.65).

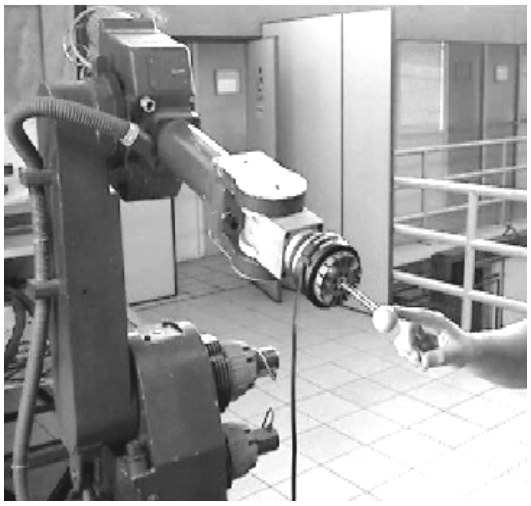

Figura 64 - Manipulador se adequando a impedância.

Figura 65 - Manipulador imóvel.

Os ensaios do controle de impedância utilizaram apenas três juntas (Juntas 1,3 e 5) do manipulador. As juntas 4 e 6 não foram utilizadas pois mudam a referência do sensor de força e assim complicam a aplicação. A junta 2 foi excluída devido a seu perfil “agressivo”, pela não modelagem da dinâmica do cilindro pneumático. A identificação das juntas utilizadas no experimento de validação é apresentada na Fig.66.

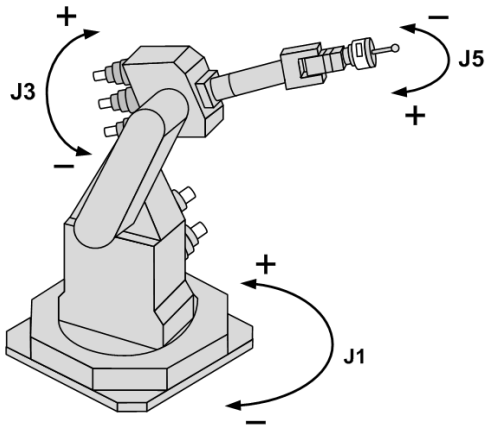

Figura 66 - Juntas utilizadas no controle de impedância.

O primeiro ensaio realizado utiliza as juntas 1 e 5, ou seja, utiliza apenas a referência da força no eixo *Y* para movimentação do manipulador, pois a junta 5 está alinhada sobre esse eixo. O perfil da força e das velocidades de junta para esse experimento é apresentado na Fig.67.

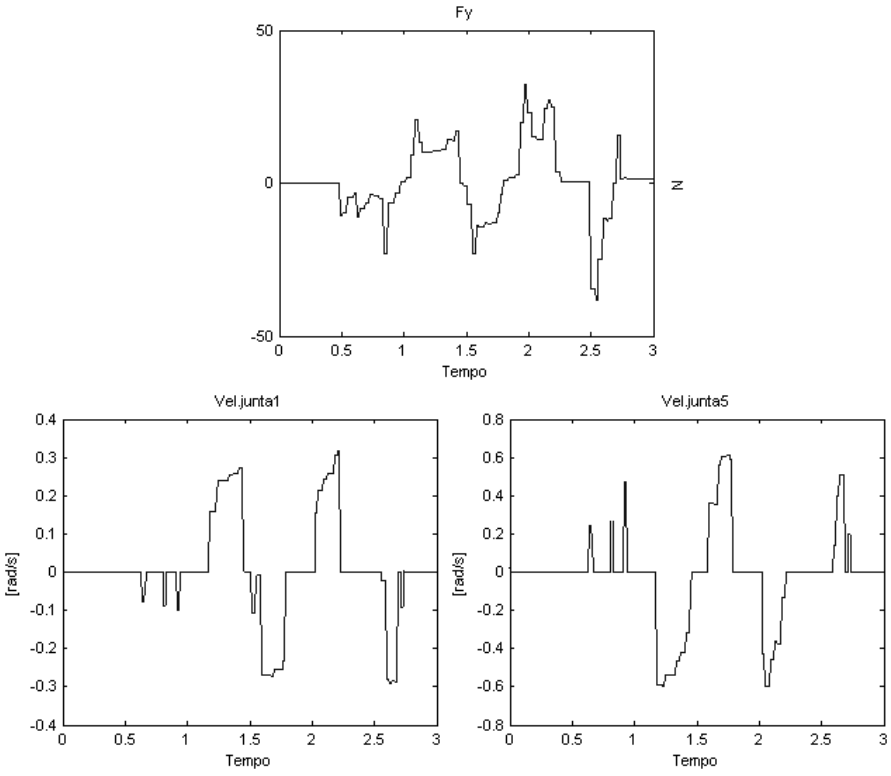

Figura 67 - Primeiro ensaio do controle de impedância.

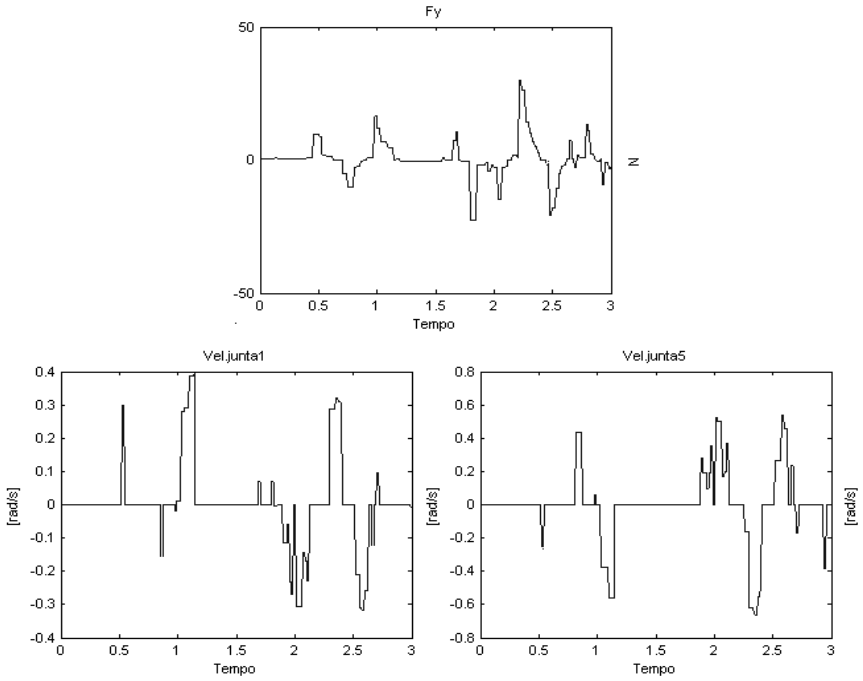

Figura 68 - Segundo ensaio do controle de impedância.

O segundo ensaio é semelhante ao primeiro, porém, a sensibilidade à ação de forças é aumentada. Isso é realizado tornando o manipulador menos rígido, ou seja, mais complacente. Dessa forma, a uma menor intensidade de forças, maiores velocidade de juntas são aplicadas. O perfil da força e das velocidades de junta para esse experimento é apresentado na Fig.68.

O terceiro ensaio inclui a junta 5 para uma manipulação paralela sobre o limite do espaço de trabalho do manipulador, ou seja, sobre os eixos *X* e *Y*. O perfil da força e das velocidades de junta para esse experimento é apresentado na Fig.69.

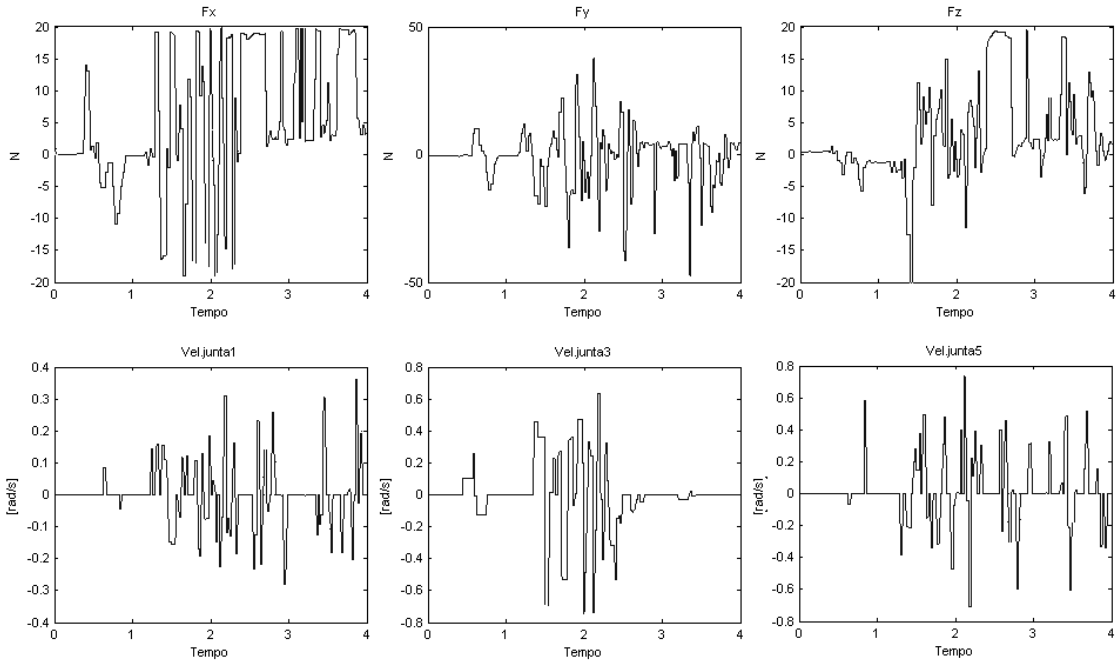

Figura 69 - Terceiro ensaio do controle de impedância.

Os experimentos realizados comprovam a eficiência do sistema desenvolvido, o qual modernizou o sistema controlador do manipulador *REIS Rv15* e assim, possibilitando o seu emprego em tarefas que necessitam do controle de posição e força.

Capítulo 6. Conclusões, Dificuldades e Perspectivas

6.1 Conclusões

O presente trabalho abordou o *retrofitting* de manipuladores, utilizando a proposta de modelo de referência para o desenvolvimento de controladores de robôs com arquitetura totalmente aberta. O enfoque principal foi o desenvolvimento de um padrão de arquitetura funcional organizada em uma estrutura hierárquica, levando em consideração os requisitos e as tendências atuais para o desenvolvimento dessa modalidade de sistema.

Esse modelo de referência foi aplicado para o desenvolvimento integral (i.e., do *software*, *middleware*, *hardware* e *firmware*) de um controlador de robôs com arquitetura aberta, utilizado no processo de *retrofitting* de um manipulador *REIS Rv15*. A concepção de controlador desenvolvida visou cumprir os seguintes requisitos: alta capacidade de processamento, baixo custo, conectividade com outros sistemas, disponibilidade de acesso remoto, facilidade de manutenção, flexibilidade no desenvolvimento de algoritmos, integração com um computador pessoal e programação em alto nível. Foi também incorporado ao sistema um transdutor de força da *JR3 Inc*., fornecendo aptidão ao sistema para operar em tarefas de controle de posição e força. A validação do sistema é realizada com a implantação de um método de controle indireto de força, o controle de impedância.

O estudo realizado neste trabalho permitiu ao Laboratório de Robótica (*LAR*) o conhecimento integral da estrutura interna e do desenvolvimento de um controlador de robôs com arquitetura aberta, aumentando potencialmente as possibilidades de estudos na área da robótica, com maior flexibilidade e um custo significativamente inferior as soluções disponíveis no mercado.

6.2 Dificuldades

O desenvolvimento do sistema controlador não enfatizou a implantação de mecanismos de segurança para uma utilização do manipulador. Futuros trabalhos devem levar em conta esse fator, baseando-se na norma *ANSI/RIA R15.06-1999: Industrial Robots and Robot Systems – Safety Requirements.* Um fator primordial para utilização do sistema é a alimentação do cilindro pneumático. Se o robô for habilitado sem essa compensação gravitacional, as juntas nesse sentido realizam movimentos bruscos, devido a força gravitacional. O que pode acarretar em danos ao manipulador e em situações de risco. O sistema possui um sensor de pressão, porém não se encontra em funcionamento e necessariamente deve ser trocado. Para que seja introduzido um mecanismo de segurança que não possibilite a alimentação do manipulador sem que o cilindro esteja com a pressão de operação.

Existe ruído em alguns sinais, como nos *encoders*, devido à exposição dos contatos pelo antigo circuito eletrônico que utiliza uma barra de contato próxima aos atuadores e sujeitas e interferências eletromagnéticas. As ligações deveriam ser feitas diretamente entre os cabos e ser devidamente isoladas, pois em altas freqüências essas perturbações podem confundir a leitura do sinal.

Outra dificuldade mecânica encontrada no manipulador é a falta do sinal de *home* em algumas juntas que impossibilitou a implantação de uma rotina de inicialização do manipulador para determinação da posição de *home*.

Uma grande dificuldade de implantação e experimentação do sistema de controle no manipulador *REIS Rv15* está na montagem e desmontagem contínua dos equipamentos, pois o local é aberto e de livre circulação. Estes procedimentos acabam acarretando em danos nos equipamentos. É imprescindível que o manipulador esteja em uma sala fechada para que o sistema de controle seja montado definitivamente e também possibilitando a implantação de medidas de segurança previstas na norma anteriormente mencionada.

6.3 Perspecitvas

O sistema necessita passar por mais testes para consolidar sua robustes e afinar o ajuste dos ganhos. Em virtude das dificuldades apresentadas anteriomente, os ensaios do sistema ocorreram em uma quantidade inferior ao esperado, dessa forma, as configurações ainda não estão completamente validadas para um funcionamento genérico.

A nova versão do sistema de controle deve ser baseada no sistema operacional *Linux* com sua variante para tempo real *RTAI-Linux* (*RealTime Application Interface for Linux*), de forma a fornecer uma maior robustez e otimizar o tempo de resposta do sistema. A utilização do sistema *Orocos* (*Open Robot Control Software*) vem sendo uma tendência adotada em vários projetos e sua viabilidade deve ser aprofundadamente estudada. Ele consiste em um pacote de programas que oferecem suporte para a as aplicações de controle de robôs. Incluindo um módulo para a programação em tempo real, bibliotecas para controle de robôs, com a geração de movimentos, cinemática e dinâmica, além de uma ferramenta para inferência de redes bayesianas.

Rerências Bibliográficas

Abele, E., Weigold, M.*, et al.* Robôs industriais na usinagem: sonho ou maturidade do mercado? Revista Máquinas e Metais, v.494, p.34-49. 2007.

ADVOCUT Project. . Disponível em: <http://www.advocut.de/>. Acesso em: 12 de junho de 2007.

Antonelli, G. Underwater Robots: Motion and Force Control of Vehicle-Manipulator Systems: Springer-Verlag. 2003

Bodner, J., Wykypiel, H.*, et al.* First experiences with the da VinciTM operating robot in thoracic surgery. European Journal of Cardio-Thoracic Surgery, v.25, n.5, p.844. 2004.

Bona, B., Indri, M.*, et al.* Open system real time architecture and software design for robot control. Advanced Intelligent Mechatronics, 2001. Proceedings. 2001 IEEE/ASME International Conference on, v.1. 2001.

Caccavale, F., Lippiello, V.*, et al.* RePLiCS: an environment for open real-time control of a dual-arm industrial robotic cell based on RTAI-Linux. Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on, p.2493-2498. 2005.

Compaq, Hewlett-Packard*, et al.* Universal Serial Bus Specification Revision 2.0 2000.

DLR. Dextrous Robot Hands, Disponível em: <http://www.dlr.de/rm-neu/en/desktopdefault.aspx/tabid-3802>. Acesso em: 06 de julho de 2007.

Donald, S. D. e Dunlop, G. R. Retrofitting Path Control to a Unimate 2000B Robot. Proc. 2001 Australian Conference on Robotics and Automation, v.14, p.15. 2001.

Eppinger, S., Seering, W.*, et al.* Three dynamic problems in robot force control. Robotics and Automation, IEEE Transactions on, v.8, n.6, p.751-758. 1992.

Erlbacher, E. A. Force control basics. Industrial Robot: An International Journal, v.27, n.1, p.2-2. 2000.

Esposito, M. P., Ilbeigi, P.*, et al.* Use of fourth arm in da Vinci robot-assisted extraperitoneal laparoscopic prostatectomy: novel technique. Urology, v.66, n.3, p.649-52. 2005.

Fan, W., Peng, G.-Z.*, et al.* A Study on the Position Control of a Spherical Parallel Robot Actuated by Pneumatic Muscle Actuators. Beijing Ligong Daxue Xuebao(Transactions of Beijing Institute of Technology)(China), v.24, n.6, p.516-519. 2004.

Farsi, M., Ratcliff, K.*, et al.* An overview of controller area network. Computing & Control Engineering Journal, v.10, n.3, p.113-120. 1999.

Ford, W. E. What is an open architecture robot controller? IEEE International Symposium on Intelligent Control, 1994. 27-32 p.

Franklin, G. F. Feedback Control of Dynamic Systems: Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA. 1993

Golin, J. F. Implementação de Controle de Força e Compensação de Atrito em Robô Industrial. Dissertação de Mestrado, Engenharia Mecânica, UFSC. 2002.

Hogan, N. Impedance control-An approach to manipulation. I-Theory. II-Implementation. III-Applications. ASME, Transactions, Journal of Dynamic Systems, Measurement, and Control (ISSN 0022-0434), v.107, p.1-24. 1985.

Hong, K.-S., Kim, J.-G.*, et al.* A PC-based open robot control system: PC-ORC. Robotics and Computer Integrated Manufacturing, v.17, n.4, p.355-365. 2001.

ISO/IEC 7498-1. Information technology - Open Systems Interconnection - Basic reference model: The basic model. 1994.

Lages, W. F., Henriques, R. V. B.*, et al.* Arquitetura Aberta para Retrofitting de Robôs. Manet Notes Workhsop. 2003.

Leite, S. H. A. R. Comparação de métodos utilizados na cinemática diferencial inversa de robôs seriais. Relatório de Estagio, Engenharia de Controle e Automação Industrial, UFSC. 2003.

Lionel Lapierre, P. F. P. D. Position/Force Control of an Underwater Mobile Manipulator. Journal of Robotic Systems, v.20, n.12, p.707-722. 2003.

Lippiello, V., Villani, L.*, et al.* An open architecture for sensory feedback control of a dual-arm industrial robotic cell Industrial Robot: An International Journal v.34, n.1, p.46-53. 2007.

Macchelli, A. e Melchiorri, C. Real-time control system for industrial robots and control applications based on real-time Linux. 15th IFAC World Congress, Barcelona, Spain, July, p.21-26. 2002.

Menzel, P. e D'aluisio, F. Robo Sapiens: Evolution of a New Species: MIT Press. 2001

Microchip Technology Inc. Advanced Solutions : USB, Seminário Técnico Avançado - Masters Brasil. 2004.

Microchip Technology Inc. 16-Bit Language Tools Libraries. 2005a.

Microchip Technology Inc. MPLAB C30 : C Compiler User's Guider. 2005b.

Microchip Technology Inc. dsPIC30F Family Reference Manual : High-Performance Digital Signal Controllers. 2006.

Microsoft Corporation. COM: Component Object Model Technologies, Disponível em: <http://www.microsoft.com/com/default.mspx>. Acesso em: 15 de junho de 2007.

Miljanovic, D. M. e Croft, E. A. A taxonomy for robot control. Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, v.1. 1999.

Murrugarra, C., Grieco, J.*, et al.* Design of a PD Position Control based on the Lyapunov Theory for a Robot Manipulator Flexible-Link. Robotics and Biomimetics, 2006. ROBIO'06. IEEE International Conference on, p.890-895. 2006.

Mustapic, G., Andersson, J.*, et al.* A Dependable Real-Time Platform for Industrial Robotics. 2003.

Nacsa, J. Comparison of three different open architecture controllers. Proceedings of IFAC MIM, Prague, p.2–4. 2001.

Okamura, A. M., Smaby, N.*, et al.* An overview of dexterous manipulation. Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, v.1. 2000.

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

Pires, J. N. Sensores de Força/Momento. Revista Robótica. 40/41**:** 116 p. 2000.

Prischow, G. Open Controller Architecture-Past, Present and Future. Annals of CIRP, v.50, n.2, p.126-135. 2001.

Regenstein, K. e Dillmann, R. Design of an open hardware architecture for the humanoid robot ARMAR. Proceedings of the 2003 International Conference on Humanoid Robots, Karlsruhe und München (CD-ROM), p.3. 2003.

Santos, C. H. F. D. Movimento Coordenado de Sistemas Veículo-Manipulador Submarino utilizando Técnicas de Inteligência Artificial e Sistemas Híbridos. Tese, Engenharia Elétrica, UFSC. 2006.

Sciavicco, L. e Siciliano, B. Modelling and control of robot manipulators: Springer. 2004

SEMATECH. Dictionary of Semiconductor Terms, Disponível em: <http://www.sematech.org/publications/dictionary>. Acesso em: 21 de agosto de 2007.

Smits, R., Bruyninckx, H.*, et al.* Model Based Position-Force-Vision Sensor Fusion for Robot Compliant Motion Control. Multisensor Fusion and Integration for Intelligent Systems, 2006 IEEE International Conference on, p.501-506. 2006.

Sperling, W. e Lutz, P. OSACA–the vendor neutral control architecture. Proceedings of the European Conference on Integration in Manufacturing IiM, v.97, p.12-22. 1997.

Tanenbaum, A. S. Redes de Computadores. Rio de Janeiro: Editora Campus. 1997

Taouil, R. M. Ensaios, Desenvolvimento e Implementação de Software para Soldagem Robotizada. Relatório de estágio em Controle e Automação Industrial, Engenharia Mecânica, UFSC. 2004.

Tzafestas, C. S., Prokopiou, P. A., *et al.* Path Planning and Control of a Cooperative Three-Robot System Manipulation Large Objects. Journal of Intelligent and Robotic Systems, v.22, p.99-116. 1998.

Vukobratovic, M. How to Control Robots Interacting with Dynamic Environment. Journal of Intelligent and Robotic Systems, v.19, n.2, p.119-152. 1997.

Weierstrass Institute for Applied Analysis and Stochastics. Realtime visualization of production processes involving industrial robots, Disponível em: <http://www.wias-berlin.de/research-groups/optim/robots_eng.html>. Acesso em: 15 de junho de 2007.

Yoshikawa, T. Force control of robot manipulators. Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, v.1. 2000.

Zeng, G. e Hemami, A. An overview of robot force control. Robotica, v.15, n.05, p.473-482. 1997.

Zhu, D.-S., Zhu, Q.-D., *et al.* Underactuated manipulator position control by using neural network. Journal of Harbin Engineering University, v.26, n.3, p.307-310. 2005.

Apêndice A. Norma 7498-1

O Modelo de Referência *OSI* (Interconexão de Sistemas Abertos), definido na norma ISO/IEC 7498-1, trata-se de uma padronização realizada pela Organização Internacional de Padrões (*ISO*), para os protocolos de interconexão que compõem as diversas camadas dos sistemas abertos, i.e., neste caso, sistemas que estão em comunicação com outros sistemas (Tanenbaum, 1997). Este consiste em um modelo estruturado em sete camadas, apresentado na Fig.70, onde cada uma é responsável por uma função específica. Não são especificados os serviços e protocolos utilizados em cada camada, isso é realizado por padronizações individuais, são determinadas apenas as abstrações entre elas.

Para a construção da arquitetura em camadas é necessário saber apenas quais os serviços oferecidos e não a composição desses. A modularização permite a alteração de uma camada sem modificar as demais, fornecendo a possibilidade do acréscimo de novas funções (Tanenbaum, 1997). Essas características suprem alguns dos requisitos de sistemas abertos como: modularidade e permutabilidade. A estrutura hierárquica apresentada no modelo de referência desta norma define que a camada é sempre usuária dos serviços oferecidos pela camada inferior a ela (ISO/IEC 7498-1, 1994). Dessa forma, esta norma é composta por um conjunto de regras que possibilitam o intercâmbio de informações entre máquinas com características distintas, ou seja, garantindo também a interoperabilidade.

Em Tanenbaum, (1997), são apresentados os princípios aplicados para se alcançar as setes camadas que compõem o modelo de referência *OSI*, são estes:

- Uma camada deve ser criada onde houver a necessidade de uma abstração diferente.
- As camadas devem desempenhar funções pré-definidas, sem ambigüidade.
- A função de cada camada deve ser definida baseada nas definições de padrões internacionais.
- As fronteiras entre as camadas devem ser delimitadas, minimizando o fluxo de informações entre as interfaces.
- O numero de camadas deve ser definido de forma que funções distintas não pertençam a mesma camada e que a arquitetura torne-se de fácil manuseio.

A seguir serão descritas sucintamente as camadas pertencentes ao modelo *OSI*, levando em consideração as características que são pertinentes ao trabalho, as informação foram baseadas em Tanenbaum, (1997), e na norma ISO/IEC 7498-1, (1994).

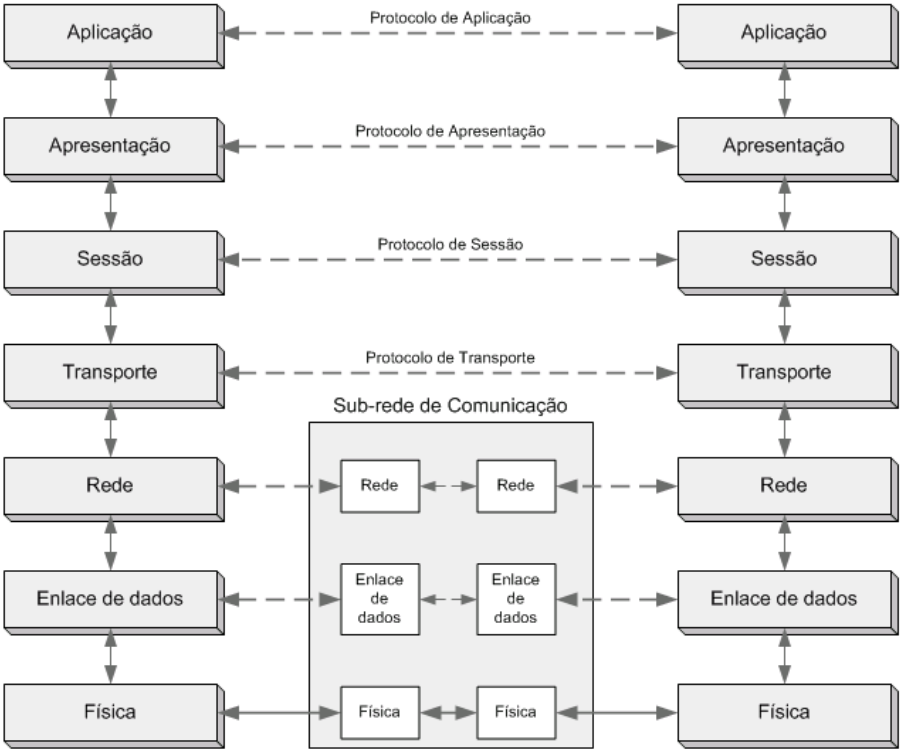

Figura 70 - Modelo OSI (Tanenbaum, 1997).

A *camada de aplicação* realiza uma comunicação "indireta" com outros aplicativos, sendo a categoria com maior nível de abstração e conseqüentemente mais próxima ao usuário. Fornecendo o acesso a rede a diversos serviços, porém convertendo os métodos de transmissão e recepção de dados em um padrão comum de comunicação, permitindo que os dados sejam reconhecidos e interpretados pelos outros usuários (i.e., nós[21]) da rede.

A *camada de apresentação*, diferentemente das camadas inferiores a ela, não possui preocupação com a confiabilidade da operação de transferência de dados, mas, com a representação desses. A função desta camada é assegurar que as informações transmitidas sejam corretamente interpretadas pelo receptor. Dessa forma, esta tem permissão para

[21] *Denominação utilizada para caracterizar um dispositivo conectado a uma rede, que possua endereço próprio.*

modificar a sintaxe[22] dos dados sem alterar a semântica[23]. Funcionalidades como criptografia e compressão de dados também são tarefas desta camada.

A *camada de sessão* é responsável por estabelecer sessões entre os usuários, i.e., uma forma de transporte de dados que possibilita a inserção de serviços especializados (e.g., um sistema de autenticação para conexão ou uma transferência de arquivo), de grande utilidade. Outra funcionalidade desta camada é realizar o controle de tráfego de dados, maximizando a utilização do meio de transmissão evitando a colisão de dados[24]. Nesta camada também ocorre o gerenciamento do *token*, nos protocolos que utilizam este conceito. O *token* é uma espécie de habilitação, ou seja, apenas o detentor do mesmo pode realizar a transmissão no meio de comunicação. Assim, ele fica flutuando na rede e quando algum nó deseja enviar algum dado, ele segura o *token* (caso ele esteja disponível) e após a transferência o coloca novamente no canal de dados. Outra funcionalidade inerente a esta camada é a sincronização. Esta característica é importante, por exemplo, quando ocorre uma transferência de arquivos. Em intervalos regulares de tempo são inclusos pontos de sincronização no fluxo dos dados, assim, se a operação for cancelada, ao invés de recomeçar todo o processo é possível continuar do ultimo ponto de sincronização.

A *camada de transporte* tem como responsabilidade garantir a corretude[25] e a ordenação dos dados em uma transferência eficiente, confiável e econômica, independentemente da tecnologia e da topologia da rede, ou das subredes[26], existentes entre a origem e o destino. Gerando uma abstração sobre o tipo de rede física às camadas superiores a esta. Tem como característica ser verdadeiramente "fim a fim", i.e., nesse nível um processo troca informações diretamente com o seu processo de destino, ignorando as máquinas intermediarias na rede.

A *camada de rede* tem como funcionalidade a transferência de pacotes (i.e., conjunto de dados) da origem até o destino, considerando os caminhos de tráfego de dados. Esta característica é denominada de roteamento, i.e., a determinação das rotas apropriadas para a

---

[22] *São as estruturas ou padrões formais de como o dado é expresso.*
[23] *Refere-se ao significado do termo em todos os sentidos.*
[24] *Acontece quando um dado está trafegando no meio de transmissão e outro dado é enviado no mesmo meio sem que o anterior tenha sido recebido e retirado. De forma, que ambos podem se misturar. Isso acontece em topologias de redes que não suportam o envio simultâneo em ambas as direções.*
[25] *É um termo comumente utilizado na informática, que se refere ao ato de estar correto.*
[26] *Consiste em uma pequena rede de computadores, que está inclusa em uma rede maior.*

transmissão dos dados entre a origem e o destino. Diferentes pacotes de dados (nesta camada, denominados de datagramas), podem possuir rotas distintas. Possibilitando que um pacote chegue antes que o seu pacote antecessor, assim, esta camada deve garantir o seqüenciamento, i.e., a seqüencia dos pacotes. Esta camada possui também a responsabilidade de resolver problemas decorrentes da incompatibilidade de endereços, protocolos e tamanhos de pacotes.

A principal tarefa da *camada de enlace* é abstrair do canal de comunicação a possibilidade de erros. Nesta ocorre a divisão da seqüência de *bits*, provenientes da camada física, em quadros de dados (i.e., blocos de dados). Pois, a camada física, apenas transmite e recebe os *bits* não levando em consideração a estrutura e o significado. Nesta operação são adicionados *bits* especiais, delimitadores de início e final de quadro de dado. Esta camada possui também a tarefa de resolver problemas de quatros repetidos, perdidos e danificados. Outra funcionalidade executada neste nível é o controle de fluxo de dados, garantindo que o *buffer*[27] de transmissão e recepção de dados não excedam suas capacidades, situação que pode ocorrer quando, por exemplo, a transmissão se realizar entre um transmissor rápido e um receptor lento. A adição de um *buffer* a um sistema de comunicação aumenta a garantia do recebimento correto dos dados. Em uma transmissão de dados, o *buffer* de recepção armazenará o dado recebido, enquanto o receptor processa o dado recebido anteriormente, porém, pode acontecer o estouro de sua capacidade e conseqüentemente a perda de dados. A camada de enlace ainda realiza o controle de acesso ao meio de comunicação, principalmente quando o meio é compartilhado, garantindo que apenas um nó transmissor acesse o meio a cada instante, ou quando suporta comunicação em ambos os lados, diferenciando os dados que trafegam em ambas as direções.

A *camada física* é a única que possui acesso físico ao meio de transmissão, devendo preocupar-se com as especificações elétricas, mecânicas e procedurais da interface física, entre o equipamento e o meio de transmissão. Controlando o fluxo de *bits* no canal de comunicação, dessa forma, tendo que garantir que quando um *bit* for enviado, o mesmo, chegará à outra extremidade do canal. As questões fundamentais neste caso são os níveis de tensão e o tempo de duração desses sinais, que determinarão os níveis baixos e altos dos *bits*. Esta camada define o método de comunicação, que pode realizar-se em apenas uma direção

---

[27] *Designação para alguma forma de memória temporária para a alocação de dados.*

(i.e., *simplex*), em duas direções, porém não simultaneamente (i.e., *half-duplex*) ou em ambas as direções simultaneamente (i.e., *full-duplex*). Outro fator especificado nesta camada são os procedimentos de inicialização e encerramento da conexão entre o emissor e o receptor de dados.

A Fig.71 demonstra a transmissão de dados através da estrutura hierárquica de comunicação do modelo *OSI*. Onde as conexões entre as camadas são virtuais (representadas pelas linhas pontilhadas), exceto na camada física, onde existe uma ligação real (ou física). Durante os procedimentos de transmissão de dados, são inseridos cabeçalhos de identificação e controle, que são utilizados nos processos de recepção de dados para determinar a corretude das informações recebidas.

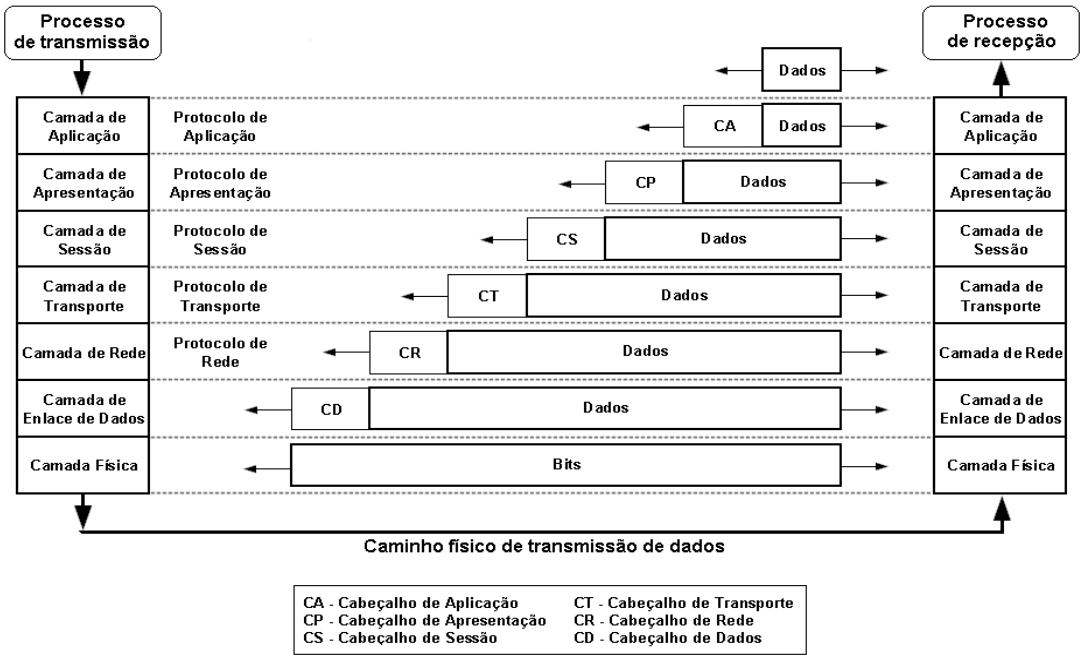

Figura 71 - Transmissão através do modelo OSI (Tanenbaum, 1997).

O presente trabalho aborda uma proposta de modelo de referência para controladores de robôs de arquitetura aberta, baseando-se no modelo de referência para sistemas de controle de robôs em aplicações industriais proposto por Sciavicco e Siciliano, (2004), porém expandindo suas funcionalidades com a sua mesclassem com o modelo de referência para a interconexão de sistemas abertos definido pela norma *ISO 7498-1*. São levados em consideração também os aspectos relevantes para os controladores de arquitetura aberta definidos anteriormente, como os requisitos e as tendências.

Apêndice B. Configurações do Processador

Osciladores

O módulo de oscilador é responsável por definir a velocidade de processamento do sistema. Um sinal oscilatório (*clock*) lento define uma baixa taxa de processamento das informações, a recíproca também é verdadeira. Todos os módulos periféricos internos utilizam esse sinal como referência de seus processamentos.

Vista sua importância, no caso dos controladores de movimentos, foram previstas três formas de geração do sinal oscilatório, através de: um cristal externo, um sinal oscilatório externo e os métodos de oscilação interna.

A primeira forma corresponde ao uso de um cristal externo, a freqüência do cristal utilizado determinará o modo de funcionamento do oscilador. Cristais na faixa de 4MHz á 10MHz são denominados de cristal (*XT*), enquanto cristais de 10MHz á 25MHz são chamados de modo de alta velocidade (*HS*). A outra forma de operação consiste na utilização de um oscilador externo, que no caso dos controladores de movimento, é um sinal proveniente do circuito integrado utilizado para a comunicação através da interface *USB*. Um sinal oscilador digital, ou seja, com uma possibilidade de imprecisão bem menor que os analógicos e que pode ter sua freqüência configurável digitalmente, esse é o modo de *clock* externo (*EC*). A última forma é a utilização da geração do sinal oscilatório internamente, a qual pode ser realizar por duas maneiras: pelo oscilador de baixo consumo (*LPRC*) e pelo oscilador rápido (*FRC*).

O módulo de oscilação ainda disponibiliza um multiplicador de *clock* denominado de *PLL* (*Phase Locked Loop)* e um divisor de *clock*. Ambos geralmente estão associados aos métodos com maiores freqüência de oscilação. O uso dessas funcionalidades deve sempre respeitar os limites do componente para cada modo e, principalmente, o limite da freqüência máxima, que é de 120MHz. Por exemplo, um sistema com um cristal de 24MHz (modo *HS*), pode optar pelo funcionamento com o multiplicador em 8x e o divisor em 2x, cuja representação seria *HS2_PLL8*, dessa maneira:

$$Frequência = \left(\frac{24Mhz \times PLLx8}{dividido\ por\ 2} \right) = 96MHz \tag{3.1}$$

Cão de guarda

O cão de guarda do sistema, i.e, o *Watchdog Timer* (*WDT*), é um supervisor do funcionamento do *firmware*. Possuindo a função de realizar uma reinicialização do sistema caso aja algum erro de execução, travamento ou o processamento indevido de um laço de repetição infinito. Consistem em um *timer*, que deve ter seu contador constantemente zerado para que o sistema não seja reiniciado.

Atraso na inicialização do sistema

O atraso na inicialização do sistema, denominado de evento *Power-on Reset* (*POR*), consiste em um atraso de tempo pré-determinado na inicialização do sistema. Permitindo que o processador inicie suas operações após um determinado tempo, quando a fonte de alimentação já se encontra estabilizada, evitando picos de tensão. O seu funcionamento pode ser visualizado na Fig.72.

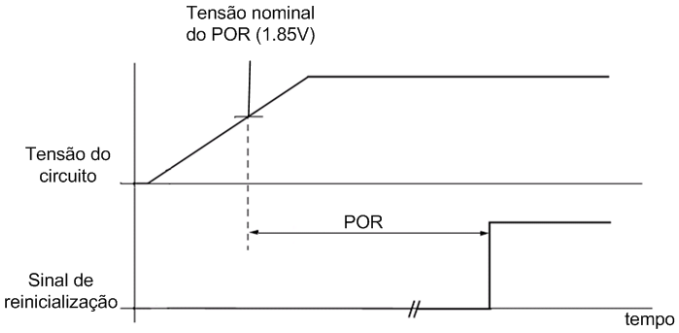

Figura 72 - Evento de atraso na inicialização do sistema (Microchip Technology Inc., 2006).

Reinicialização por queda de tensão

A reinicialização por queda de tensão (Fig.73), i.e., *Brown-out Reset* (*BOR*), é um mecanismo de proteção do componente, que tem com funcionalidade reinicializar o sistema caso ocorra uma queda de tensão. O módulo *BOR* tem seu funcionamento baseado no circuito interno de tensão de referência, ou seja, opera sobre a comparação da tensão do circuito com a tensão de referência.

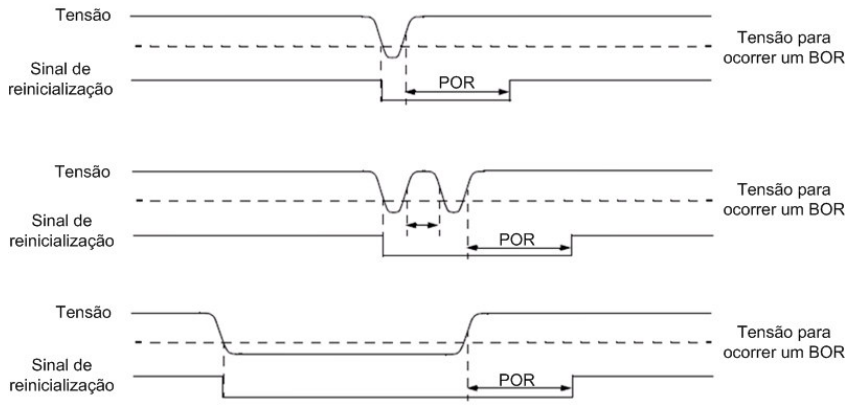

Figura 73 - Evento de reinicialização por queda de tensão (Microchip Technology Inc., 2006).

Outro ponto que necessita ser detalhadamente apresentado é a inicialização do sistema, onde os módulos periféricos internos são configurados.

Apêndice C. Rotinas de Inicialização

Módulo universal de transmissão/recepção assíncrono (*UART*)

A comunicação através da interface *USB* é realizada com a troca de informações entre o conversor do barramento com o módulo universal de transmissão/recepção assíncrono (*UART*) do controlador de sinais digitais (*DSC*). Desta forma, promovendo uma forma de comunicação *full-duplex*, i.e., dois canais de comunicação independentes com possibilidade de operação simultânea. Fornecendo também uma transferência de dados assíncrona, i.e., sem a necessidade do sincronismo entre o emissor e receptor. Porém ambas as vias de comunicação operam com a mesma taxa de transferência e tamanho do pacote de dados. Em uma transferência de dados, cada dado a ser enviado, composto por seus caracteres de informação e controle, é denominada de quadro de dados, que tem tamanho fixo e pré-definido pelos parâmetros da comunicação. Onde, cada conjunto de quadros é denominado de pacote de dados.

A principal característica desta forma de comunicação é o envio serial dos dados. A transferência é realizada seqüencialmente, como uma fila. A dificuldade dessa modalidade de comunicação encontra-se em delimitação de início e final de quadro. De forma a resolver esta e outras questões, são inseridos caracteres especiais. Vale ressaltar, que nesta camada, tratamos de *bits*, exemplificando, uma letra consiste em um conjunto de oito *bits*. A Fig.74 demonstra a transferência de um pacote de dados, com alguns dados com seus delimitadores de inicio e fim de quadro.

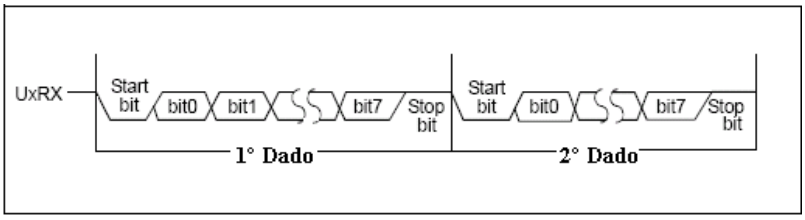

Figura 74 - Sinal de uma comunicação serial (Oliveira e Andrade, 2006).

A configuração deste módulo de comunicação serial utiliza o formato padrão *8N1*, o que significa, um dado de 8 *bits*, sem paridade e com *bit* de parada igual a 1. O principal fator a ser determinado para uma correta utilização deste periférico é a velocidade de transferência

de dados. Esta grandeza é denominada de gerador de taxa de transmissão, i.e., *UART Baud Rate Generator* (*BRG*), este é um valor que necessita ser calculado da seguinte maneira:

$$UxBRG = \left(\frac{Frequência\ do\ oscilador\ (Hz)}{16 \times Taxa\ de\ transferência\ (bps)} - 1 \right) \tag{3.2}$$

O registrador que armazena esta configuração é do tipo inteiro, dessa forma, talvez seja necessário realizar um arredondamento, causando um erro no cálculo. Exemplificando, seja um sistema que opere com um cristal com freqüência de 4MHz e almejamos uma taxa de transferência de 9600bps, assim:

$$UxBRG = \left(\frac{4 \times 10^6}{16 \times 9600} - 1 \right) = 25.042 \cong 25$$

É possível se determinar a taxa de transmissão em relação a um valor do registrador *UxBRG* da seguinte maneira:

$$Taxa\ de\ transmissão = \left(\frac{Frequência\ do\ oscilador\ (Hz)}{16 \times (UxBRG + 1)} \right) bps \tag{3.3}$$

Aplicando a equação (3.3) no exemplo anterior obtém-se que:

$$Taxa\ de\ transmissão = \frac{\left(4 \times 10^6\right)}{16 \times (25 + 1)}) \cong 9615 bps$$

Com esses dados é possível se dimensionar o erro de cálculo, o qual é dado por:

$$Erro = \left(\frac{Taxa\ de\ transferência\ calculada - Taxa\ de\ transferência\ desejada}{Taxa\ de\ tranferência\ desejada} \right) \tag{3.4}$$

Aplicando a equação (3.4) no exemplo anterior obtém-se:

$$Erro = \left(\frac{9616 - 9600}{9600} \right) = 0.16\%$$

Nos sistemas embarcados do controlador de robôs desenvolvido, o módulo de comunicação pela *UART*, opera a uma taxa de transferência de 1.5Mbps, com o sistema oscilador configurado conforme a equação (3.1), o que resulta no valor zero para o registrador *UxBRG*.

Interface para *encoders* de quadratura (*QEI*)

Os *encoders* de quadratura ótico incrementais são utilizados para a medição da posição e o cálculo da velocidade em movimentos de atuadores rotacionais. Um motor elétrico não é regido por grandezas como posição e velocidade, o único sinal de entrada suportado é a tensão elétrica. Dessa forma, o controle sobre o atuador necessita de uma malha fechada, onde um *encoder* terá a função de obter as grandezas para a realimentação do sistema e possibilitar as correções necessárias ao sistema.

Internamente, o funcionamento de um *encoder* se dá pela emissão luminosa através de um disco perfurado até um receptor ótico. A posição do disco determina o nível lógico do sinal de saída, ou seja, quando um orifício estiver alinhado sobre o emissor e o receptor, o sinal terá nível lógico alto (um ou 5V), quando não estiver, o nível lógico do sinal será baixo (zero ou 0V).

Existem dois tipos essenciais de *encoders*: o absoluto e o incremental. O *encoder* absoluto (Fig. 75) possui a codificação direta em seu disco, dessa forma, o sinal de saída já é a posição angular. O sentido do movimento é dado comparando a posição atual com a anterior.

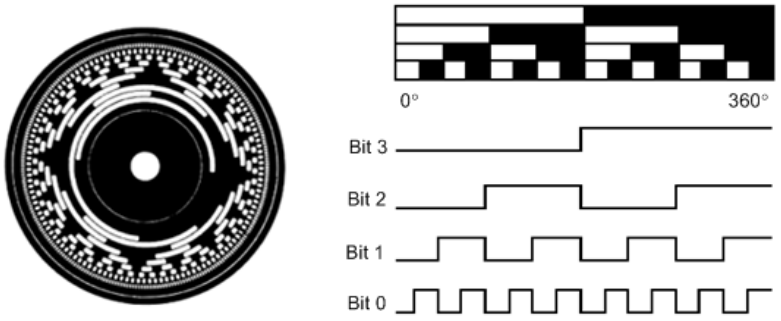

Figura 75 - Encoder absoluto (Oliveira e Andrade, 2006).

O *encoder* incremental possui orifícios igualmente espaçados em seu disco, assim fornecendo apenas o sinal de incremento ou decremento. Este tipo de sensor, geralmente possui três sinais distintos: a fase *A* (*QEA*), a fase *B* (*QEB*) e o índice (*INDX*), os quais necessitam ser decodificados para informar corretamente as características do movimento.

O relacionamento entre os sinais de fase *A* e de fase *B* define a direção do movimento. Quando a fase *B* encontra-se defasada em relação a fase *A*, o movimento é denominado positivo, na situação oposta se obtém o movimento negativo. Estes sinais de quadratura

produzidos pelo *encoder* possuem quatro estados, denominados por suas combinações, estes indicam um ciclo de contagem. Quando o sentido do movimento é oposto, a ordem desses estados é alterada. O terceiro sinal, o índice, ocorre uma vez por revolução, é utilizado como referência para o cálculo da posição. A Fig.76 demonstra o relacionamento entre esses sinais.

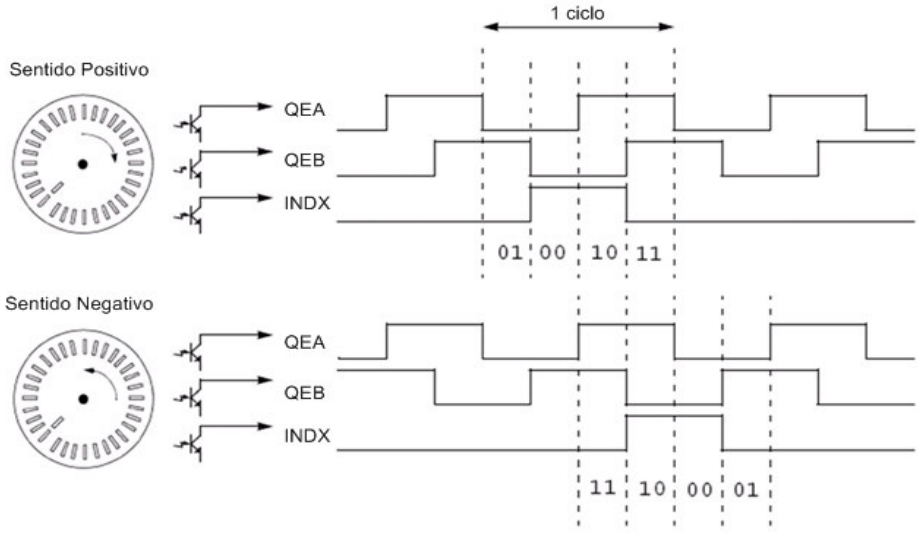

Figura 76 - Sinais do módulo QEI (Oliveira e Andrade, 2006).

O módulo de interface para *encoders* de quadratura (*QEI*), opera sobre os *encoders* incrementais, pois, a lógica de leitura de um *encoder* absoluto é apenas uma conversão do sistema de numeração binário para o sistema desejado. Como componente principal desse módulo, tem-se o decodificador de quadratura, responsável por capturar sinais provenientes do *encoder* incremental e convertê-los em um contador numérico de posição. O diagrama de blocos deste módulo é apresentado na Fig.77.

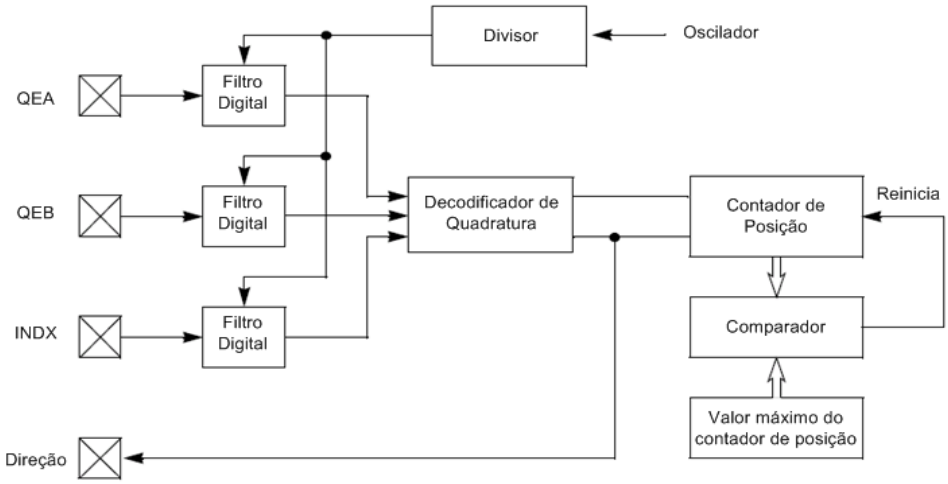

Figura 77 - Diagrama de blocos do módulo QEI (Microchip Technology Inc., 2006).

Os filtros realizam uma pequena amostragem do sinal, permitindo apenas a passagem de um nível lógico, após a ocorrência consecutiva de três sinais iguais. Tecnicamente, o filtro só permite a aquisição de um sinal, após ter um nível lógico estabilizado em três *ciclos de clock*[28] consecutivos, esta característica é demonstrada na Fig.78.

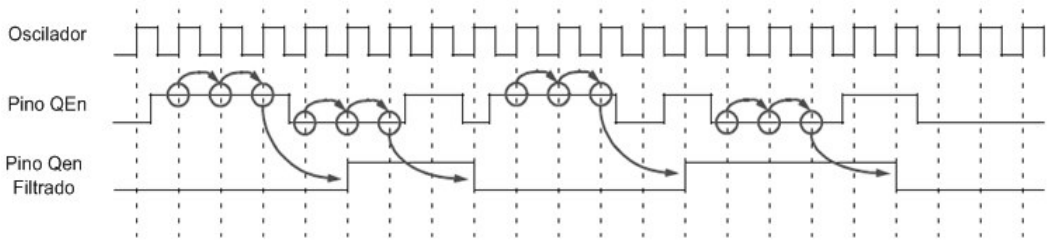

Figura 78 - Amostragem realizada pelo filtro digital do módulo QEI (Microchip Technology Inc., 2006).

O fator de oscilação desses filtros determina a banda baixa passante, pois, tem-se um filtro passa-baixa. Um *clock* lento para o filtro determina que a banda passante seja de menores freqüências que um filtro com o *clock* mais rápido. O *clock* do filtro é o *clock* do oscilador dividido pelo divisor programável.

Os filtros digitais também estão relacionados às entradas, que assumem os níveis lógicos seguindo o padrão *Schmitt Trigger*, o qual auxilia na filtragem de ruídos, conforme apresentado na Fig.79. Estas entradas possuem uma maior faixa de incerteza entre os níveis lógicos, assim pequenas alterações na tensão não alteram os níveis lógicos, filtrando ruídos.

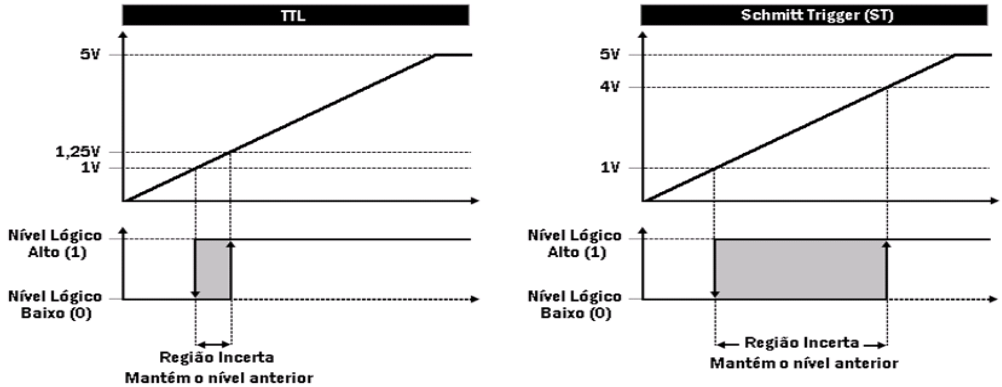

Figura 79 - Comparação dos níveis TTL e Schmitt Trigger (Oliveira e Andrade, 2006).

[28] *Um pulso do sinal gerado pelo sistema oscilatório.*

O decodificador de quadratura opera com dois métodos de decodificação dos sinais de fase. O primeiro método, denominado de modo *2x*, está relacionado apenas com sinal da fase *A*. Assim, a cada transição de estado (subia e descida) é realizado um incremento ou decremento do contador, de acordo com o sentido. O segundo modo, o *4x*, está relacionado com ambas as fases, aumentando a resolução do leitor de posição angular. A cada transição desses sinais, uma correspondente alteração do contador de posição é realizada. As formas de operação descritas acima estão ilustradas na Fig.80.

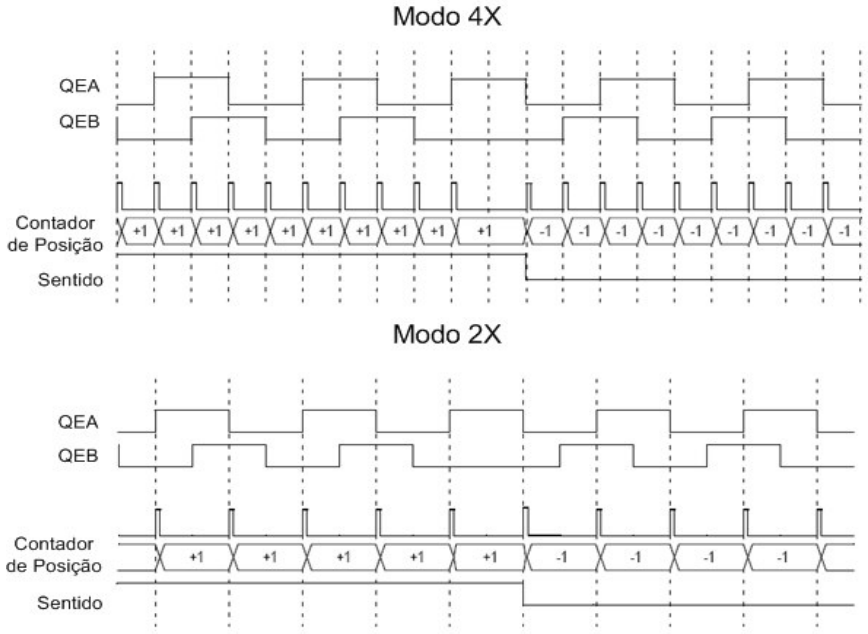

Figura 80 - Formas de operação do decodificador de quadratura (Oliveira e Andrade, 2006).

A velocidade de atuação do motor determinará as freqüências dos sinais de fase do encoder. Os sinais de quadratura podem ser decodificados desde que, um sinal de *clock* do contador seja gerado a cada borda (subida ou descida) do sinal de quadratura. Isto garante uma medida da posição angular com uma resolução de até quatro vezes a contagem do *encoder*. Exemplificando, um motor que opera a *10.000 RPM* com uma resolução de $8192\ ^{passos}\!/_{revolução}$ fornece uma freqüência de quadratura de:

$$Frequência = \left(\frac{10000}{60}\right) \times 8192 \times 4\ ) = 5.46Mhz$$

O módulo de decodificação de encoders de quadratura opera com uma freqüência de $Frequência\ do\ oscilador\,(Hz)/3$. Um sistema trabalhando com um oscilador de freqüência de 30MHz, os sinais de fase *A* e *B* poderão operar em uma freqüência de até 10MHz.

O controlador de sinais digitais utilizado possui um processamento em 16 *bits*, dessa forma, o módulo *QEI* também opera com essa resolução, fornecendo uma contagem de até $65.535\ passos/revolução$, o que é insuficiente. Esse módulo foi ampliado para 32 *bits*, utilizando a acumulação do contador antigo que ocorre no evento de interrupção (Fig.81) por estouro do contador ou quando o contador possui o valor zero.

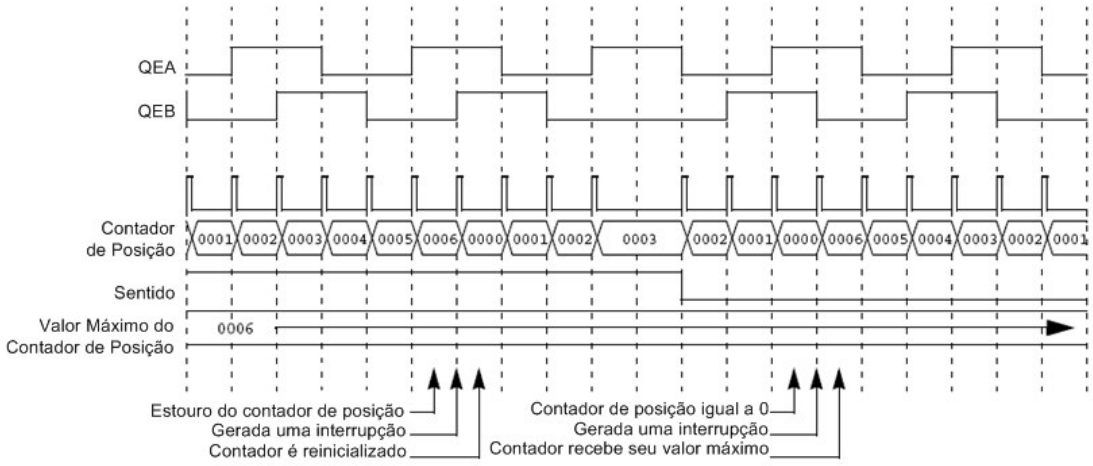

Figura 81 - Interrupção pelo registrador MAXCNT (Oliveira e Andrade, 2006).

Módulo de modulação por largura de pulso (*PWM*)

O modulo de geração dos sinais com modulação por largura de pulso, i.e., *Pulse Width Modulation* (*PWM*), é um poderoso recurso utilizado para o controle de motores. A partir deste, é possível gerar um sinal analógico (i.e., sinal com tensão variável), através de um sinal digital que assume apenas os níveis lógicos: alto e baixo. O sinal de saída é uma onda quadrada, com a freqüência constante (i.e., período fixo) e largura de pulso (denominado de *Duty Cicle*) variável. A freqüência de uma onda pode ser definida como a quantidade de vezes em que ela se repete no tempo, e o período como as frações da onda que irão se repetir no tempo.

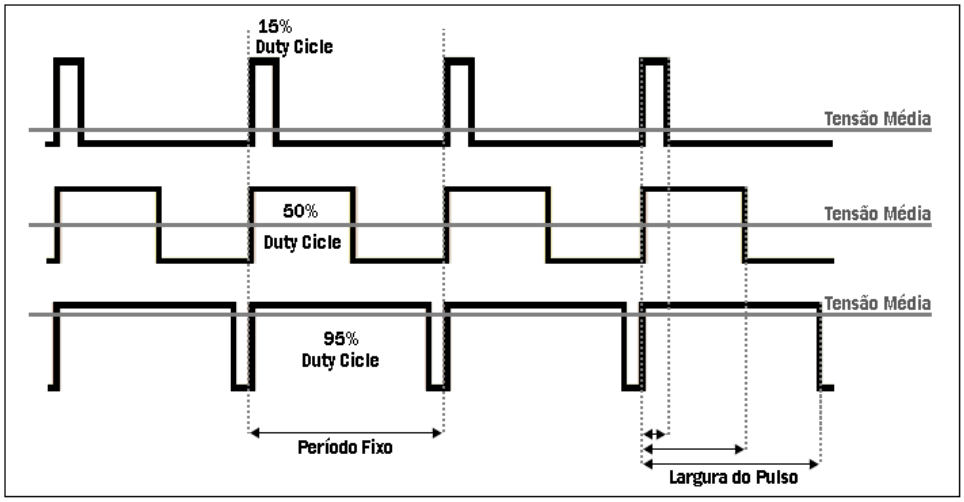

Figura 82 - Sinais de Modulação por Largura de Pulso (Oliveira e Andrade, 2006).

O ciclo ativo, i.e., *Duty Cicle*, define o tempo de sinal ativo (nível lógico alto) em um período fixo. Assim quando temos um *Duty Cicle* de 100%, temos nível lógico alto por todo o período, e quando temos 0%, temos nível lógico baixo por todo o período. Analogamente, um *Duty Cicle* de 50%, define a metade do período em nível lógico alto e a outra metade em nível lógico baixo, fornecendo uma tensão de saída média. Se estivermos operando em níveis *TTL* (Fig.79) com a tensão em nível alto de 5V, esse ciclo ativo fornecerá uma tensão média de 2,5V. Estes conceitos são demonstrados na Fig.82.

A freqüência de geração do sinal de modulação por largura de pulso utilizada nos sistemas embarcados que compõem o controlador de robôs é de 18.51kHz. A qual foi levantada empiricamente de forma a realizar o melhor controle sobre os atuadores do manipulador, essa questão será mais profundamente abordada no capítulo 4.

Controle *PID*

O controlador proporcional, integral e derivativo (*PID*) é uma associação clássica de três controladores, combinando as vantagens desses. A ação integral está diretamente ligada à precisão do sistema, sendo responsável pelo erro nulo em regime permanente. O efeito desestabilizador do termo integrativo é contrabalançado pela ação derivativa que tende a aumentar a estabilidade relativa do sistema, ao mesmo tempo que torna a resposta do sistema mais rápida, devido ao seu efeito antecipatório (Franklin, 1993). O diagrama de blocos desta modalidade de controle é apresentado na Fig.83.

O controle *PID* digital é executado em intervalos periódicos de amostragem, porém deve possuir uma freqüência adequada ao sistema, para que o mesmo possa ser corretamente controlado. As parcelas que compõem essa modalidade de controlador serão individualmente explicadas a seguir, baseadas em *Franklin,* (1993) e em Microchip Technology Inc., (2005b).

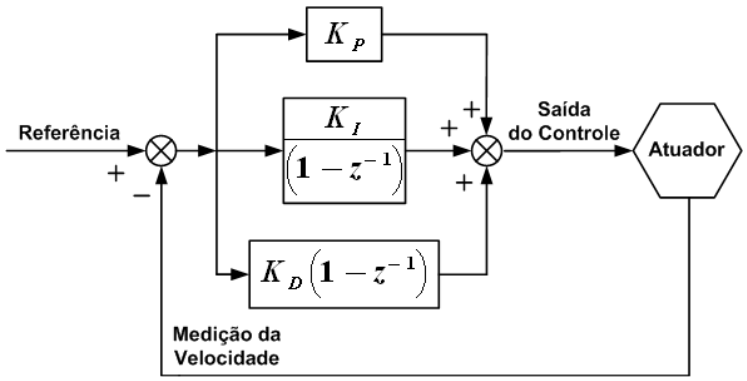

Figura 83 - Controle PID (Microchip Technology Inc., 2005a).

A parcela proporcional (*P*) do controle realiza uma multiplicação do sinal de correção (i.e., realimentação) por um ganho *P*. Isso influência o controlador a gerar um sinal de resposta em função da magnitude do erro. Dessa forma, se o sinal de erro for amplo, o termo proporcional causará uma correção de mesma dimensão.

O efeito da parcela proporcional reduz o erro à aproximadamente zero. Em vários sistemas, o erro aproxima-se de zero, mas não converge para esse valor, o que resulta no erro em regime permanente (i.e., após a estabilização do sinal). O termo integral (*I*) é introduzido para corrigir pequenos erros em regime permanente. Um pequeno erro nessas condições pode acarretar em um grande erro acumulado no tempo. A saída do controlador é gerada pela multiplicação desse erro acumulado pelo fator de ganho *I*.

O termo diferencial (*D*) aumenta a velocidade de resposta do controlador e opera sobre a variação do sinal de erro, i.e., a diferença entre o valor atual do erro e o valor anterior. A multiplicação dessa variação pelo fator de ganho *D* gera a saída do controlador.

Nem todos os controladores possuem o termo *D*, é menos comum ainda a utilização da parcela *I*. Isso ocorre porque certos efeitos tornam-se desprezíveis devido ao comportamento do sistema. Por exemplo, no controle velocidade de um motor, comumente, não é encontrada a parcela *D*, devido às características físicas do motor, que possui uma resposta relativamente lenta.