ESTE EXEMPLAR CORRESPONDE A REDAÇÃO FINAL DA TESE DEFENDIDA POR EDGARD DE OLIVEIRA E APROVADA PELA COMISSÃO JULGADORA EM 04 / 07 / 2008

ORIENTADOR

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Prototipagem Rápida de Sistemas Mecatrônicos Baseada em Instrumentação Virtual

Autor: Edgard de Oliveira
Orientador: Prof. Dr. João Mauricio Rosário

88 / 2008

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

Prototipagem Rápida de Sistemas Mecatrônicos Baseada em Instrumentação Virtual

Autor: Edgard de Oliveira
Orientador: Prof. Dr. João Maurício Rosário

Curso: Engenharia Mecânica
Área de Concentração: Mecânica dos Sólidos e Projeto Mecânico

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

Campinas, 2008.
SP – Brasil

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

OL41p

Oliveira, Edgard de
Prototipagem rápida de sistemas mecatrônicos baseada em instrumentação virtual / Edgard de Oliveira. – Campinas, SP: [s.n.], 2008.

Orientador: Prof. Dr. João Maurício Rosário
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Automação industrial 2. Prototipagem rápida 3. Controle 4. Mecatrônica I. Rosário, João Maurício. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

Titulo em Inglês: Rapid prototyping of mechatronics systems based in virtual instrumentation
Palavras-chave em Inglês: Industrial automation, Virtual Instrumentation, Rapid Prototyping, Control, Mechatronics
Área de concentração: Mecânica dos Sólidos e Projeto Mecânico
Titulação: Mestre em Engenharia Mecânica
Banca examinadora: João Maurício Rosário, Humberto Ferasoli Filho, Antônio Batocchio
Data da defesa: 04/07/2008
Programa de Pós Graduação : Engenharia Mecânica

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

DISSERTAÇÃO DE MESTRADO

Prototipagem Rápida de Sistemas Mecatrônicos Baseada em Instrumentação Virtual

Autor: Edgard de Oliveira
Orientador: Prof. Dr. João Maurício Rosário

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:

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

Prof. Dr. Humberto Ferasoli Filho
Universidade Estadual Paulista - UNESP.

Prof. Dr. Antônio Batocchio
Universidade Estadual de Campinas - UNICAMP.

Campinas, 04 de julho de 2008

Dedicatória:

Dedico este trabalho à meu pai, minha mãe, minha esposa e filho.

Agradecimentos

Este trabalho não poderia ser terminado sem a ajuda de diversas pessoas às quais presto minha homenagem:

A meus pais, minha esposa e filho, pelo incentivo em todos os momentos da minha vida.

Ao meu orientador, prof. Dr. João Maurício Rosário, pela forma como conduziu este trabalho e principalmente pela confiança que depositou em mim.

Aos amigos Almiro, Maurício, Gastão e a todos os demais que ajudaram de forma direta e indireta na conclusão deste trabalho.

A Qualisys Engenharia e National Instruments pelo indispensável apoio no desenvolvimento deste trabalho de pesquisa.

Resumo

Oliveira, Edgard, *Prototipagem Rápida de Sistemas Mecatrônicos Baseada em Instrumentação Virtual,* Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2008. 254 p. Dissertação (Mestrado)

Através da integração de conhecimentos nas diferentes áreas de engenharia, física, matemática e computação, surgiu uma ciência multidisciplinar denominada mecatrônica. Com o rápido avanço das tecnologias e a crescente demanda por soluções que as acompanhem, criou-se uma necessidade no mercado tecnológico de reduzir o tempo de implementação de um projeto através da utilização de ferramentas de prototipagem rápida que permitissem a integração de conhecimentos e implementação do ciclo em V de desenvolvimento de um produto em ambiente virtual. O objetivo do presente trabalho é a apresentação de metodologias de Prototipagem Rápida de Dispositivos Mecatrônicos com ênfase em Instrumentação Virtual, através de ambiente voltado à capacitação e desenvolvimento de projetos na área de automação industrial, onde os principais conceitos possam ser verificados e posteriormente implementados e validados através de problemas industriais, fornecendo subsídios para a análise e estratégias para concepção destas aplicações.

Palavras Chaves

Automação industrial, Instrumentação Virtual, Prototipagem Rápida, Controle, Mecatrônica

Abstract

Oliveira Edgard: *Rapid Prototyping of Mechatronics Systems based in Virtual Instrumentation*, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2008. 254 p. Dissertação (Mestrado)

Through the integration of knowledge in the different areas of engineering, physics, mathematics and computation had appeared a science to multidiscipline called mecatronic. With the fast advance of the technologies and the increasing demand for solutions follow that them, created a necessity in the technological market to reduce the time of implementation of a project being originated tools of fast prototyping that they allowed to the integration of knowledge and implementation of the cycle in V of development of a product in virtual environment. The objetive of this work is the presentation of methodologies Rapid Prototyping of Mechatronics Systems based in Virtual Instrumentation, through environment facing the training and development of projects in the area of industrial Automation, can be checked and subsequently implemented and validated in practice, providing subsidies for the analysis and strategies for design of these applications.

Keywords

Industrial automation, Virtual Instrumentation, Rapid Prototyping, Control, Mechatronics

Sumário

Lista de Figuras

Lista de Tabelas

Nomenclatura

Abreviações

CCD - Charge-Coupled Device

CLP - Controladores Lógicos Programáveis

DSP – Digital Signal Processor

FDB - Function Block Diagram

FPGA - Field Programmable Gate Array

GL - Graus de Liberdade

GRAFCET - Grafo de Comando Etapa e Transição

HIL - Hardware In-the-Loop

HTML - HyperText Markup Language

IEC - International Electrotechnical Commission

IL - Instruction List

IP - Internet Protocol

LabVIEW - Laboratory Virtual Instrument Engineering Workbench

LAN - Local Area Network

LD - Ladder Diagram

NI – National Instruments

PAC - Programmable Automation Controller

PC - Personal Computer

PCI - Peripheral Component Interconnect

PXI - PCI eXtensions for Instrumentation

RdP – Redes de Petri

SA - Sistema Automatizado

SCADA - Supervisory Control And Data Acquisition

SED - Sistemas a Evento Discreto

SFC - Sequential Function Chart

USB - Universal Serial Bus

WWW - World Wide Web

Capítulo 1

Introdução

Com rápido avanço das tecnologias e a crescente demanda por soluções que as acompanhem, criou-se uma necessidade no mercado tecnológico de reduzir o tempo de implementação de um projeto, seja ele voltado a uma melhoria em algum processo de produção ou acadêmico.

Este trabalho tem como objetivo a utilização de ferramentas para implementação de novas arquiteturas de controle para sistemas mecatrônicos a fim de torná-los mais precisos e eficientes na execução de tarefas, aliando conceitos de prototipagem rápida para a implementação de controladores. As novas tecnologias envolvendo a concepção de microprocessadores, interfaces de potência e comunicação, aliados ao conceito de prototipagem rápida possibilitam pesquisas científicas, onde a implementação de controladores requer menores custos.

Este capítulo apresenta um panorama geral do trabalho desenvolvido, sendo apresentado um estado da arte do conceito Instrumentação Virtual, com ênfase na área de Mecatrônica, justificando o desenvolvimento desse projeto de pesquisa, seus objetivos gerais e específicos, a necessidade de capacitação nesta importante área e a forma como foi delimitada esta questão, além da estrutura geral dos capítulos da tese.

1.1 **Apresentação do Problema de Pesquisa**

O desenvolvimento tecnológico permitiu uma melhoria nos processos de produção, através do aumento da quantidade produzida, como também, na qualidade do produto. A partir daí surgiu uma importante área multidisciplinar onde a integração de varias ciências foi introduzida, a qual denominou-se Mecatrônica.

A Mecatrônica integra conhecimentos nas diferentes áreas de engenharia, matemática, física e computação, permitindo o desenvolvimento com maior rapidez de sistemas compostos de uma estrutura mecânica complexa, geralmente composta por atuadores, sensores de posicionamento e um sistema de acionamento e controle, inserindo-se dentro desse contexto a Robótica.

Robôs são encontrados nos diversos meios, desenvolvendo as mais diversas funções, em atividades insalubres, educacional, ou simplesmente em trabalhos que requerem força e agilidade. Um robô industrial (manipulador) é composto de uma estrutura mecânica complexa, com acoplamento dinâmico entre as articulações. Cada grau de liberdade é geralmente composto por atuadores, sensores de posicionamento e pelo sistema de acionamento e controle.

Os elementos básicos do manipulador como estrutura, eixos, mancais, transmissão, devem ser dimensionados do ponto de vista de resistência e rigidez, com o objetivo de atingir um volume de trabalho e precisão mecânica exigidos numa operação automatizada.

A robótica industrial possibilita que processos sejam realizados de maneira mais rápida e eficiente do que no passado. Para isso, tanto o desenvolvimento da mecânica de precisão quanto da eletrônica foram indispensáveis. Mas isso pode ser melhorado, principalmente no que se refere ao tempo de execução de diferentes tarefas, pela interação do controle de robô com as grandes possibilidades que um ambiente automatizado pode oferecer.

O estudo de novas arquiteturas de controle torna-se muito importante para que haja uma melhoraria no desempenho de robôs industriais face às mudanças bruscas de parâmetros associados à posição, velocidade e aceleração, durante a realização de uma determinada trajetória. A Automação Flexível possibilita que robôs sejam cada vez mais rápidos podendo interferir em um grande volume de trabalho, dessa forma, novas técnicas de controle são estudadas com a finalidade de torná-los mais eficientes.

Mais especificamente na área de controle, se faz necessária uma ferramenta que possibilite a simulação e a rápida implementação da metodologia aplicada, validando assim o projeto na prática. Esse tipo de ferramenta pode ser extremamente útil na área de desenvolvimento industrial para a validação de estudos e comprovação de sua eficiência.

As validações dos conceitos e ferramentas de prototipagem rápida desenvolvidos nos capítulos desta tese de mestrado foram realizadas através da implementação no Laboratório de Automação Integrada e Robótica da UNICAMP de diferentes projetos direcionados ao Ensino, Formação e Pesquisa em Instrumentação Industrial e Controle.

1.2 **Motivação**

Considerando-se o atual cenário tecnológico e o alto desenvolvimento de soluções cada vez mais integradas para concepção e validação de dispositivos mecatrônicos, a utilização de ferramentas de prototipagem rápida para Instrumentação e Controle Industrial torna-se uma indispensável opção para agilizar e facilitar a implementação de diferentes arquiteturas de controle, possibilitando assim a redução do tempo de finalização de um projeto e ou estudo cientifico, atendendo uma eficiência desejada.

Diante do apresentado, a motivação encontrada para o desenvolvimento deste trabalho de pesquisa deve-se ao fato da necessidade de uma metodologia para a concepção e implementação de projetos na área de mecatrônica, visando permitir de forma mais rápida e com significativa redução de custos, a utilização mais adequada das tecnologias que compõem a área.

1.3 **Objetivos do Trabalho**

O objetivo do presente trabalho é a apresentação de metodologias de Prototipagem Rápida de Dispositivos Mecatrônicos com ênfase em Instrumentação Virtual, através de ambiente voltado à capacitação e desenvolvimento de projetos na área de automação industrial, onde os principais conceitos possam ser verificados e posteriormente implementados e validados na prática, fornecendo subsídios para a análise e estratégias para concepção destas aplicações.

A implementação de sistemas de controle, envolvendo o gerenciamento de entradas e saídas do sistema, pode às vezes ser demorada e ter um alto custo. Neste trabalho de pesquisa são apresentadas ferramentas para prototipagem rápida de sistemas mecatrônicos utilizando-se o conceito de instrumentação virtual. Um dos objetivos desta dissertação é disponibilizar uma ferramenta de fácil acesso e que faça com que o projetista ou estudante tenha seu foco no assunto desenvolvido e não em como implementar sua solução.

Com a utilização desta ferramenta, diferentes técnicas de controle poderão ser facilmente implementadas, testadas e validadas, permitindo verificar os seus desempenhos com diferentes variações de parâmetros de controle. Esses procedimentos de prototipagem rápida para a concepção de dispositivos mecatrônicos baseados em instrumentação virtual permitem a realização de grande parte do projeto dentro de um ambiente de instrumentação virtual, diminuindo tempo de projeto e custos envolvidos durante a sua fase de concepção.

1.4 **Delineamento do trabalho**

O desenvolvimento deste trabalho de pesquisa envolve o Laboratório de Automação Integrada e Robótica (LAIR) do DPM/FEM/UNICAMP (Brasil) e a National Instruments^{TM}, onde foram realizados ao longo dos últimos anos inúmeros projetos de parceria e pesquisa em conjunto, com ênfase na área de implementação de dispositivos mecatrônicos utilizando conceitos de instrumentação virtual baseada em LABVIEW^{TM} associado a uma metodologia para a organização de projetos que pode ser resumida em três etapas de desenvolvimento:

1) Elaboração e desenvolvimento da solução.
2) Simulação e Prototipagem.
3) Implementação e validação.

A partir da realização deste projeto de Pesquisa torna-se possível o desenvolvimento de projetos na área de Mecatrônica, utilizando conceitos de Prototipagem Rápida, permitindo assim a simulação de implementação de plantas industriais mecatrônicas, próximas a realidade do mercado. A figura 1.1 mostra uma visão sistêmica geral dada à área de Concepção de Sistemas Mecatrônicos, ênfase principal deste trabalho de pesquisa.

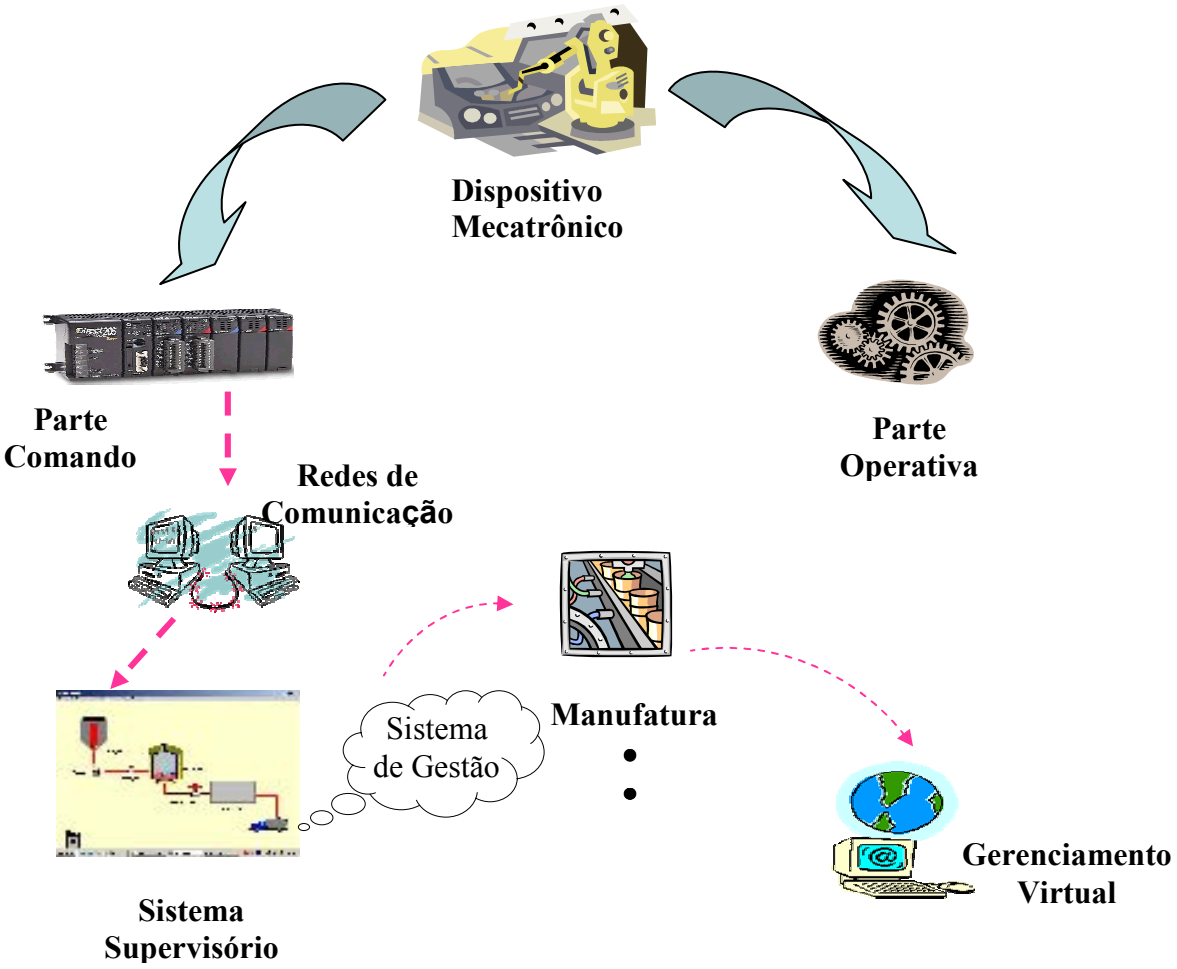


Figura 1.1 – Controle e Supervisão de um Dispositivo Mecatrônico.

1.5 **Estrutura do trabalho**

Esta dissertação de mestrado tem como proposta validar uma ferramenta que permita a rápida implementação de dispositivos de controle de sistemas mecatrônicos e que permita uma grande flexibilidade no uso dos diversos tipos de controle disponíveis. O trabalho foi dividido nas seguintes etapas:

Neste Capítulo 1 foram introduzidos os objetivos e apresentada a estrutura geral deste trabalho de pesquisa.

No Capítulo 2 são apresentados conceitos de prototipagem rápida nas áreas de engenharia mecânica, eletrônica, células flexíveis de manufatura e instrumentação virtual assim como também as principais etapas que envolvem este importante conceito para o desenvolvimento de arquiteturas de controle mais eficientes, apresentando custo e tempo de implementação reduzida.

No Capítulo 3 é feita uma Descrição de Elementos de um Sistema Automatizado e são apresentadas algumas Ferramentas de Modelagem.

O Capítulo 4 descreve o ambiente LabVIEW^{TM} utilizado para instrumentação virtual e principalmente como ferramenta para implementação de controle de dispositivo mecatrônicos. São descritas suas ferramentas para controle de processos e outros tipos de interação possíveis com o sistema em estudo. Neste capitulo é apresentada também uma experiência envolvendo uma simulação de controle de dispositivos por diferentes metodologias, capacitando a ferramenta também na área de simulação virtual.

A seguir, o capitulo 5 concerne a validação do trabalho, sendo apresentados alguns exemplos práticos implementados no Laboratório de Automação Integrada e Robótica da UNICAMP utilizando-se a ferramenta LabVIEW^{TM} e as etapas dos processos, demonstrando pelos resultados obtidos que o ambiente escolhido é eficaz e eficiente para a solução proposta.

Finalizando, no último capítulo deste trabalho, são apresentadas as conclusões e perspectivas pertinentes ao desenvolvimento deste trabalho, bem como o uso de novas ferramentas do ambiente que justificam ainda mais a escolha do mesmo para soluções com prototipagem rápida.

Nos anexos desse trabalho são apresentados conceitos básicos envolvendo a utilização de Visão Industrial, e outros resultados relativos aos exemplos práticos implementados no capitulo 5 desta dissertação de mestrado, que não se encontram no corpo da tese, mas que permitem uma melhor compreensão por parte do leitor.

Capítulo 2

Conceito de Prototipagem Rápida para Concepção de Sistemas Mecatrônicos

2.1 Introdução

Este capítulo apresenta uma revisão de conceitos de prototipagem rápida para concepção de sistemas mecatrônicos, referente às áreas mecânica, eletrônica e células de manufatura, como também a introdução deste conceito aplicado a instrumentação virtual. Também serão apresentadas as etapas de desenvolvimento de um produto e as vantagens da utilização de prototipagem rápida neste processo.

2.2 Conceitos Gerais de Prototipagem Rápida

Nos dias atuais, a evolução tecnológica está direcionando uma grande gama de produtos na área de engenharia, acarretando em alterações drásticas na concepção de um projeto de um sistema mecatrônico no que diz respeito às evoluções do projeto e concepção mecânica, sistema de acionamento e de controle. Como exemplos, podemos citar a intensidade crescente de componentes eletromecânicos inteligentes, de máquinas automatizadas, de veículos com grande número de sensores e eletrônica embarcada e de dispositivos mecânicos de precisão.

Estes novos produtos mecatrônicos emergem da combinação apropriada dos sistemas mecânicos, da eletrônica e do processamento em tempo real dos sistemas de controle e tratamento de informações, através da implementação de diversificadas funções de controle embarcadas e integradas ao sistema.

Um fator de grande importância industrial consiste na redução de tempo de desenvolvimento de um produto com a diminuição do número de etapas, gerando a produção de novos e diferenciados produtos. Isso é indispensável no mundo globalizado e particularmente necessário nas indústrias de alta tecnologia, como por exemplo, no que diz respeito à implementação de controladores cada vez mais rápidos e eficientes.

Muitos processos e produtos técnicos nas diferentes áreas da engenharia, principalmente na engenharia mecânica e elétrica, apresentam uma integração crescente dos sistemas mecânicos com processamento da eletrônica digital e de informação. O seu desenvolvimento envolve a busca de uma solução otimizada entre a estrutura mecânica básica, o sistema de sensoriamento e o elemento de atuação e controle, através do processamento automatizado de informações e controle global do sistema.

Tudo isso acarretará no desenvolvimento e utilização de ferramentas para o projeto simultâneo dos sistemas mecânicos e hardware de acionamento e controle, através da implementação do software e funções de controle embarcadas dentro de um ambiente de simulação, tendo como resultado e principal objetivo a eliminação da totalidade ou parte de protótipos intermediários e a geração de um componente ou um sistema integrado de controle. Para tal propósito descrito existe um conceito atual denominado prototipagem rápida de sistemas mecatrônicos.

A prototipagem rápida é uma ferramenta que permite a construção de protótipos de uma maneira econômica e segura. Um conceito de prototipagem rápida utilizado anteriormente referia-se a construção de protótipos de peças mecânicas a partir de um projeto desenvolvido em CAD (Computer Aided Design) ou da implementação de componentes eletrônicos em FPGA's a partir de um CAD de eletrônica. Mais recentemente, esse conceito é utilizado de forma mais ampla, envolvendo a concepção de todo um projeto de um sistema mecatrônico desde as fases de modelagem, simulação e arquitetura de controlador até a sua implementação final em hardware dedicado.

Assim sendo, a prototipagem rápida tem como objetivo a geração automática do código equivalente ao controlador para testar os sistemas reais, resultando em diminuição nos custos de implementação de um controlador, principalmente se esse for desenvolvido para um projeto específico, ou seja, a prototipagem rápida é uma ferramenta que possibilita a construção de protótipos de uma maneira econômica e segura, onde hardware pode ser implementado num sistema embarcado (embedded system) a partir de componentes virtuais.

2.3 **Prototipagem Rápida em Mecânica**

O conceito de prototipagem rápida possui um vínculo muito forte relacionado à área mecânica devido a construção de peças mecânicas a partir de um projeto desenvolvido em CAD (Computer Aided Design). Na figura 2.1 temos um exemplo de aplicação de técnicas de prototipagem rápida na elaboração de projetos de dispositivos mecânicos.

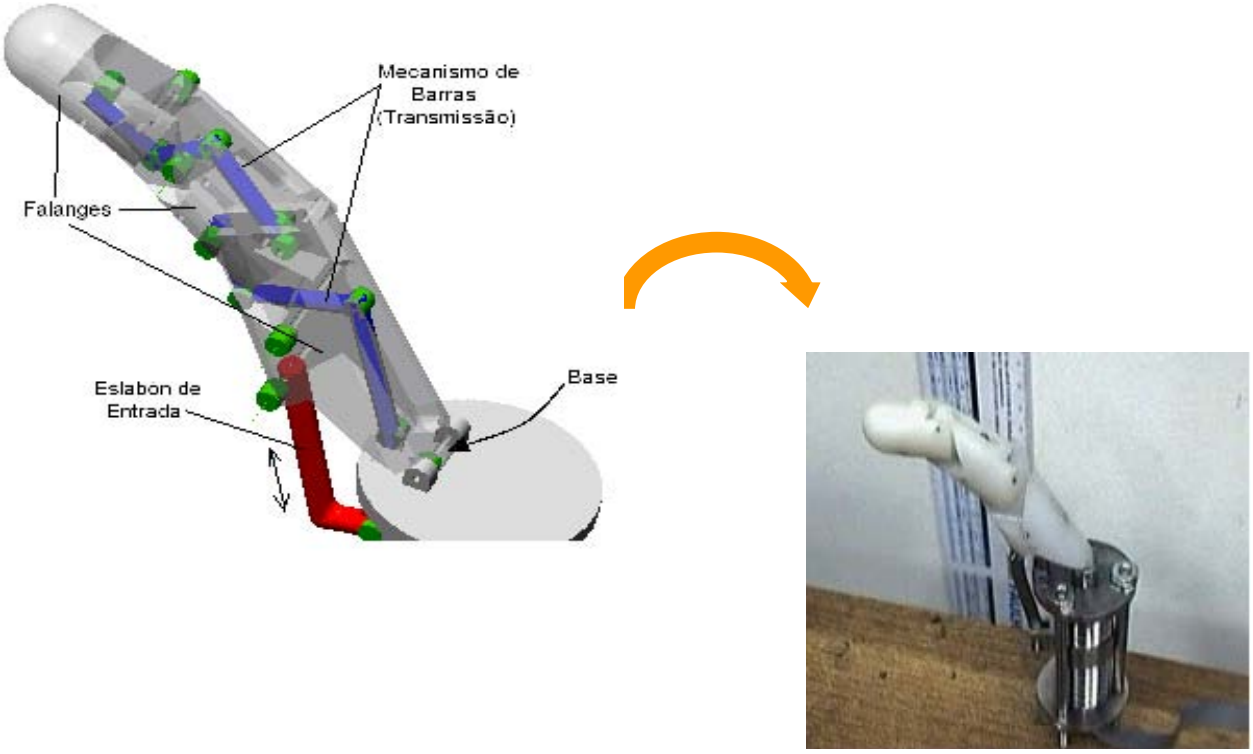


Figura 2.1 –Concepção de um produto desenvolvido em CAD.

2.4 **Prototipagem Rápida em Eletrônica**

O conceito de prototipagem rápida em eletrônica refere-se inicialmente a implementação de componentes eletrônicos em FPGA´s a partir de um CAD de eletrônica. Um exemplo de uso deste conceito é a aplicação de técnicas de Hardware In the Loop (HIL).

Uma das principais vantagens dos sistemas de prototipagem rápida é que após a simulação do modelo virtual do sistema, o sistema total ou partes desse modelo poderão ser facilmente trocados pelo hardware de controle real para validação e testes do modelo, de modo que o protótipo virtual se transformará num produto muito próximo do real, simplificando etapas de concepção, validação e testes. Conseqüentemente, após a fase de simulação do sistema, que deverá incluir o modelo virtual do hardware, esta parte do modelo poderá ser trocada pelo hardware real, similar ao modelo real, fazendo que o hardware passa ser integrado no modelo de simulação virtual e possibilitando o estudo do comportamento do novo componente com o resto do sistema, em particular o estudo do comportamento global do sistema.

Essa fase de testes é chamada de Hardware In the Loop (HIL), que deverá permitir a integração automática do hardware físico dentro do ambiente de simulação. As principais ferramentas necessárias para realização desse procedimento são as seguintes:

- Ferramentas de programação e hardware aberto para implementação do protótipo;
- Unidade de cálculo que permita assegurar a comunicação com o hardware e a área de trabalho, que deverá comportar: portas de entradas e saídas e conversores analógico/digital e digital/analógico.
- Implementação de programa que permita a importação de modelos representando o sistema e que seja capaz de gerar as entradas e programável por blocos.

As aplicações do HIL são utilizadas para avaliar e validar os elementos desenvolvidos por um novo sistema. Elas consistem em testar esses elementos, antes de concretizar o sistema real, a partir unicamente da simulação do resto do sistema. Os componentes do hardware testados respondem aos sinais enviados pelo computador e simulam o resto do sistema, como se encontrassem dentro de um sistema real.

Atualmente existem várias empresas que oferecem soluções que realizam o hardware in loop (HIL), como por exemplo: dSPACE®, National Instruments® (NI), Altera® e Opal-RT® que apresentam produtos dedicados a prototipagem rápida.

A figura 2.2 apresenta esquematicamente um exemplo de implementação através do ambiente MathWorks® utilizando HIL. Neste exemplo observa-se que a partir do ambiente de simulação utilizado para validação e testes do modelo físico implementado, o hardware de controle poderá ser automaticamente gerado e incorporado dentro do simulador para validação e testes dentro desse mesmo ambiente de prototipagem. As principais etapas a serem implementadas no HIL são as seguintes:

- Desenvolvimento de um programa de simulação usando ambiente tal como o Matlab/Simulink®;

- Configuração das entradas e saídas necessárias para o funcionamento do sistema simulado com o elemento hardware;

- Geração automática do código (com possibilidade de troca de parâmetros em tempo real e iniciação imediata da simulação em HIL);

- Visualização dos resultados em tempo real.

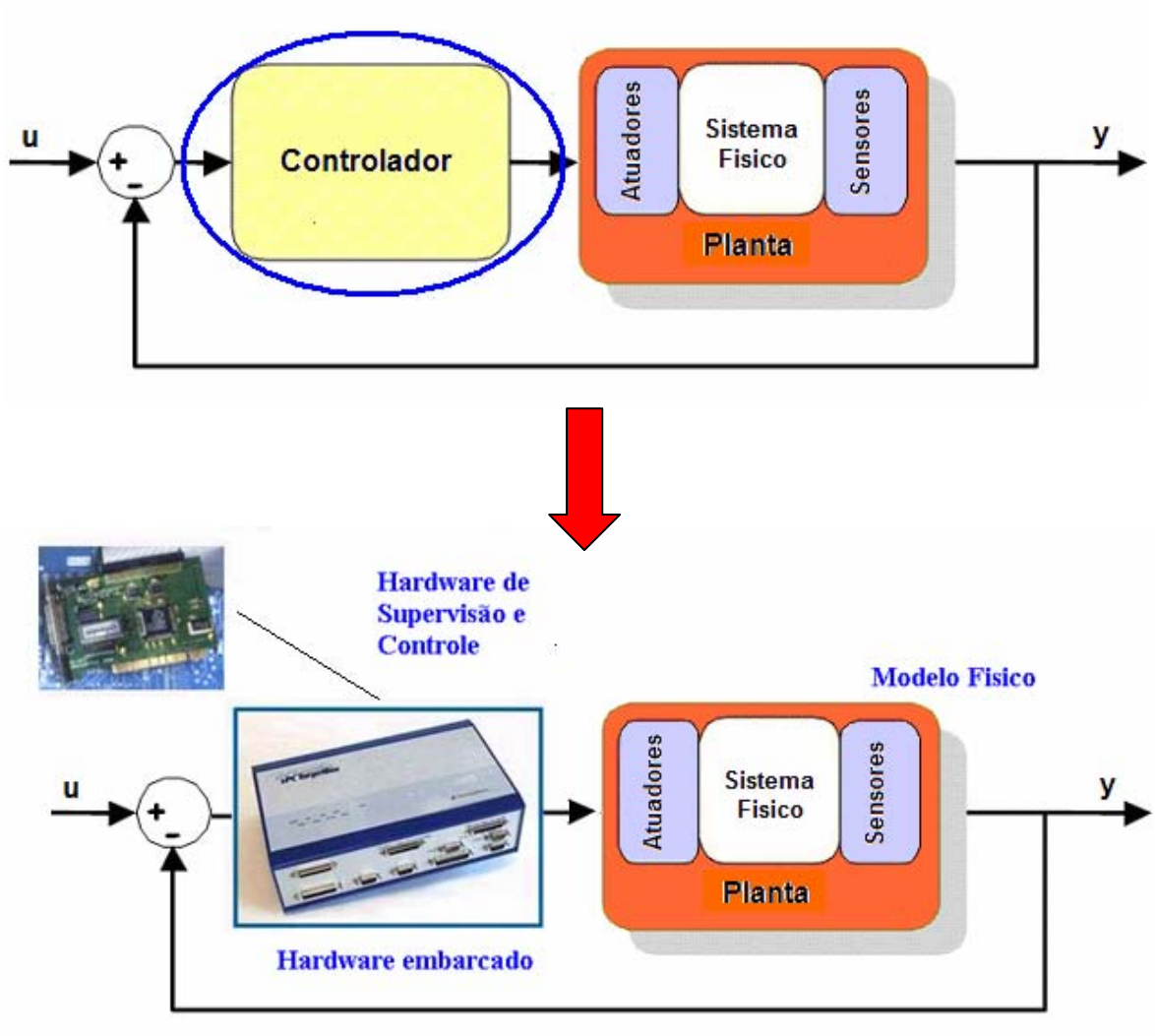


Figura 2.2 – Hardware In the Loop através do Mathworks®.

2.5 **Prototipagem Rápida em Células flexíveis de Manufatura**

O conceito de prototipagem rápida em células flexíveis de manufatura refere-se à integração de diferentes dispositivos em células automatizadas de manufatura (Computer Integrated Manufacturing). A figura 2.3 ilustra uma aplicação de prototipagem rápida em células flexíveis de manufatura.

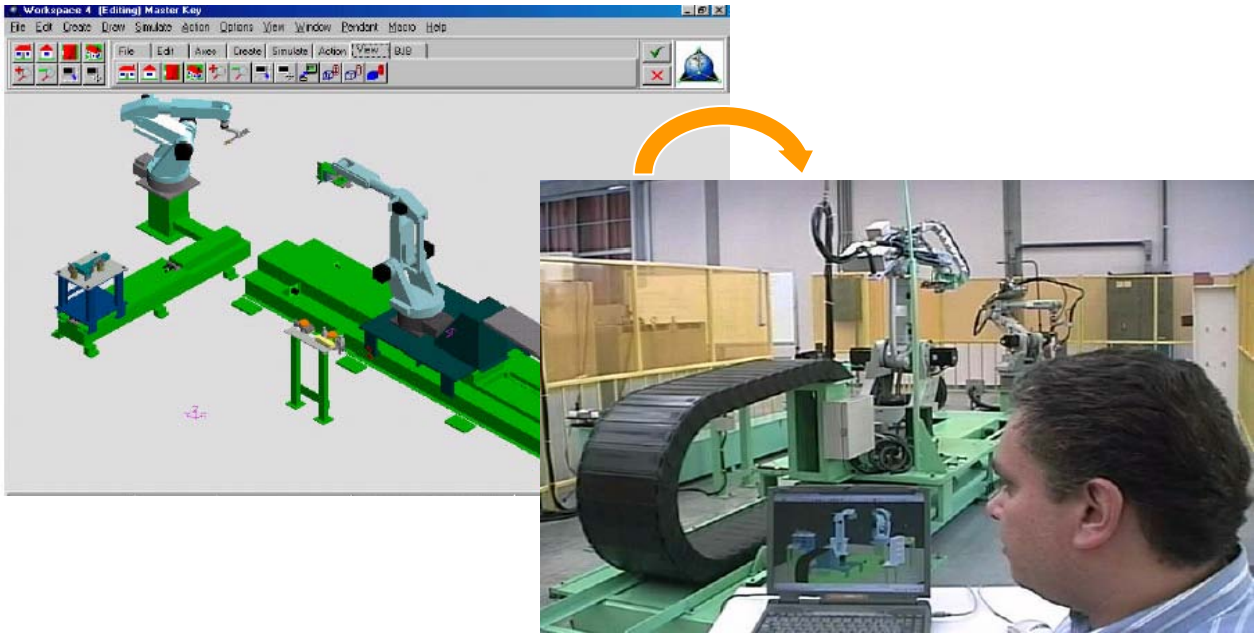


Figura 2.3 – Prototipagem de uma Célula Flexível de Manufatura

2.6 **Prototipagem Rápida em Instrumentação Virtual**

Podemos definir prototipagem rápida em instrumentação virtual como a implementação de controle de sistemas mecatrônicos através de software, hardware e Controladores Programáveis para Automação (PAC).

Atualmente, conceito de Prototipagem Rápida envolve a concepção de todo o projeto de um sistema mecatrônico através de ferramentas colaborativas, desde as etapas de modelagem, simulação e arquitetura de controlador, até a sua implementação final em hardware dedicado.

Ampliando esse conceito podemos incluir a implementação em ambiente virtual do modelo do sistema (modelagem cinemática e dinâmica), simulação e hardware de supervisão e controle.

2.7 **Descrição de Etapas para Prototipagem Rápida**

O fluxograma apresentado na figura 2.4 mostra das diferentes fases de implementação de um sistema robótico para a obtenção de um protótipo, que pode ser generalizada para qualquer modelo de sistema mecatrônico, ou qualquer mecanismo controlável que se deseja construir.

Para efeito de concepção de um sistema mecatrônico, essa metodologia deverá ser aplicada utilizando a fase de ***Projeto***, conforme mostra o fluxograma a partir do modelo de um sistema físico.

Como mostrado na figura 2.4, existem várias etapas para a concepção de um sistema de controle utilizando a técnica de prototipagem rápida. Para a diminuição de custos, a maior parte da concepção resulta na parte de simulações antes da implementação real. Deste modo, o desenvolvimento de um ambiente virtual que permita analisar o melhor desempenho da técnica de controle a ser implementada se faz necessário. Nessa figura são mostradas as etapas necessárias que foram implementadas no simulador virtual desenvolvido.

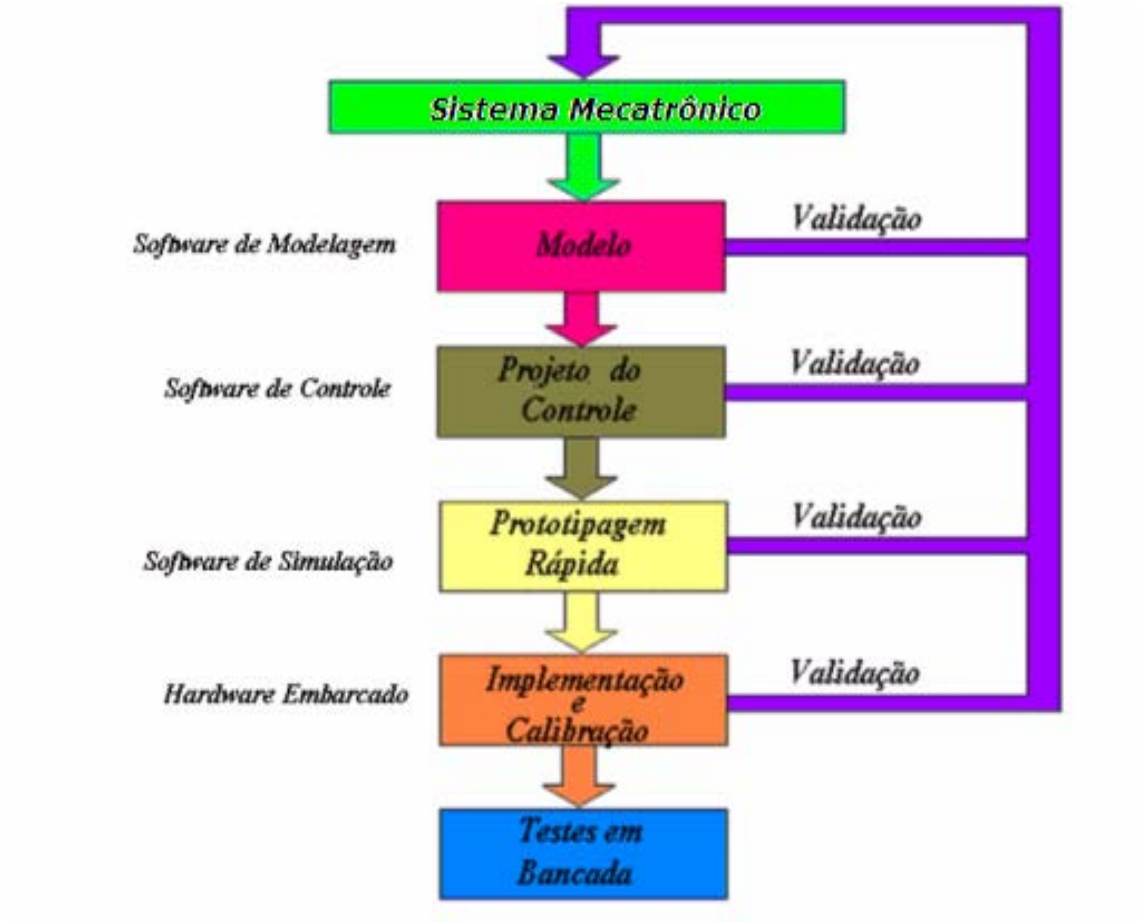


Figura 2.4 – Fluxograma das diferentes fases de implementação de um sistema mecatrônico a partir do modelo de um sistema físico.

A implementação das diferentes etapas relacionadas à prototipagem rápida de um sistema mecatrônico deverá seguir a orientação conforme mostra o modelo ciclo em V, geralmente utilizado para projeto e concepção de sistemas automatizados (fig. 2.5).

O ciclo em V, apresentado na figura 2.5 (Isermann, 2005), sintetiza as diferentes etapas associadas à concepção de um Produto Inteligente, onde pode ser visto uma fase inicial de ***Projeto e Concepção*** (abrangendo fase de requisitos, especificações, modelo, componentes e prototipagem) e uma segunda fase de implementação final associada à ***Integração de Sistemas*** (abrangendo a integração final a nível hardware e software, testes e certificações, produção e supervisão).

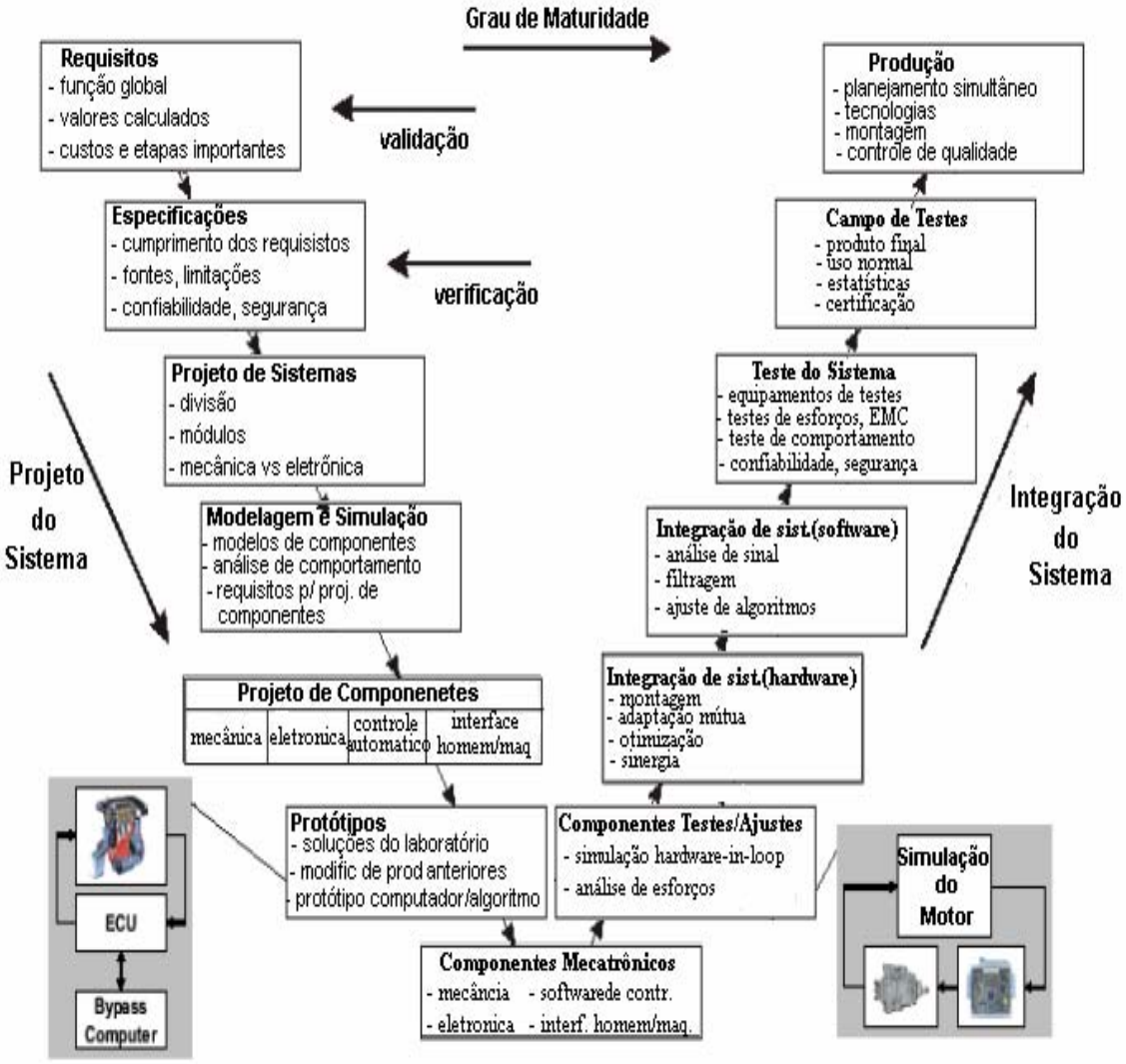


Figura 2.5– Ciclo em V - Metodologia para Concepção e Desenvolvimento para sistemas Mecatrônicos (Isermann, 2005).

A figura 2.6 mostra uma outra forma gráfica de representação das etapas de desenvolvimento de um produto proposta pela National isntruments.

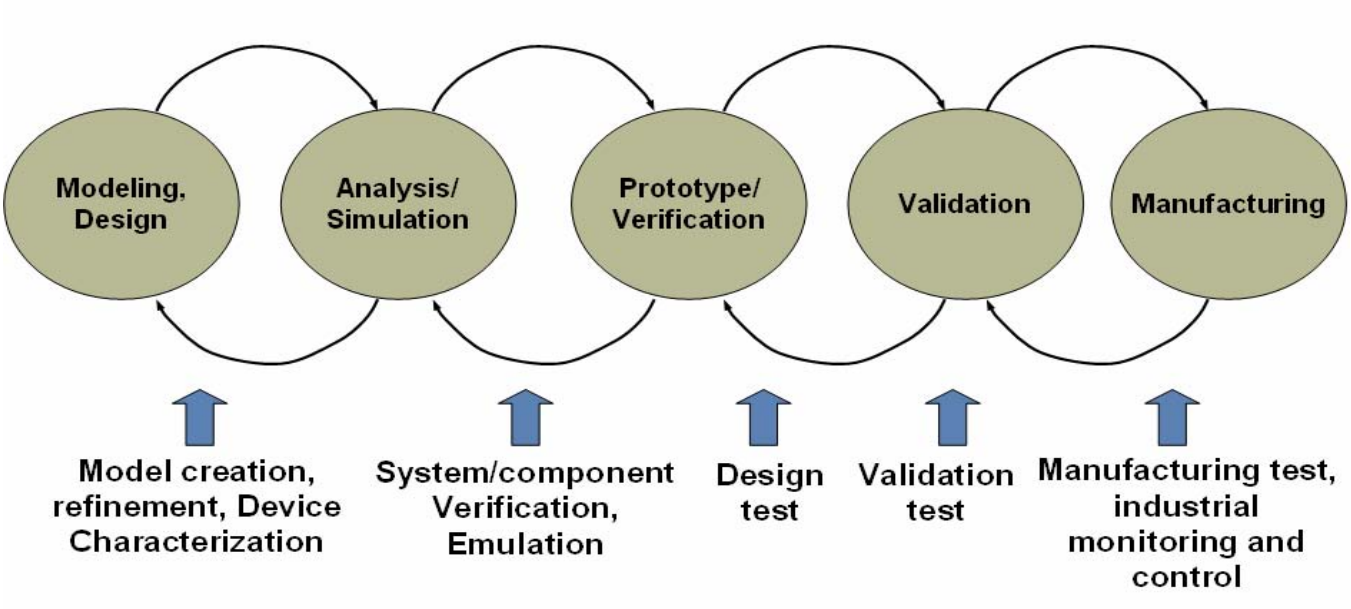


Figura 2.6 – Representação das diferentes fases de implementação de um sistema mecatrônico.

Dentre as principais vantagens da utilização de Sistemas de Prototipagem Rápida em Mecatrônica podemos citar as seguintes:

- Detecção mais rápida de possíveis erros decorrentes da fase de implementação de um projeto, acarretando um menor custo de correção e/ou modificação do projeto,
- Concepção dentro de um ambiente de simulação e prototipagem comum, acarretando de tal modo na economia no desenvolvimento do projeto atual e de futuros projetos,
- Ambiente apropriado para Engenharia Colaborativa, com forte integração e conceito de equipe de trabalho.

Como principais resultados da utilização de Sistemas de Prototipagem Rápida podem ser destacados:

- Melhor produtividade, acarretando menores atrasos, custos de desenvolvimento reduzidos e uma melhor qualidade.
- Busca de ambiente de integração único, possibilitando o cálculo cientifico, modelagem de sistemas e simulação, análise de dados e visualização, e implementação de software embarcado em tempo real.

Atualmente, os principais softwares tradicionais de modelagem e simulação de sistemas mecânicos e/ou eletrônicos direcionaram á área de prototipagem rápida em mecatrônica, com módulos e bibliotecas direcionadas a aplicações no campo automobilístico, indústria aeronáutica e aeroespacial, etc. Esses ambientes de prototipagem rápida permitem a simulação, implementação e testes num ambiente cooperativo integrado. Dentre os principais aplicativos utilizados na indústria, podemos destacar dentre os mais utilizados os seguintes: LabVIEW da NI - National Instruments, DSpace, AMESim, Mathworks (Matlab-Simulink), Altera.

2.8 **Considerações finais**

Este capítulo apresentou o conceito de prototipagem rápida para concepção de sistemas mecatrônicos, apresentando ferramentas disponíveis no mercado, onde a ênfase neste trabalho de pesquisa foi a utilização de ferramentas de instrumentação virtual para a concepção de arquiteturas de controle para sistemas mecatrônicos utilizando técnicas de prototipagem rápida.

Dentre as principais vantagens da utilização de prototipagem rápida em mecatrônica, pode-se destacar a possibilidade de programação através de blocos estruturados, com uma modelagem hierarquizada; a utilização de softwares que permitem a modelagem de componentes mecânicos a partir da simulação do movimento; a análise direta dos movimentos sem necessidade de derivar as equações; a diminuição do ciclo de concepção e interfaces convenientes com o usuário, e a possibilidade de implementação de algoritmos de controle adequados ao problema em estudo.

Dentre os principais inconvenientes, podemos citar que a maior parte destas ferramentas são dedicadas e desenvolvidas em função das necessidades pela própria indústria, e apresentam alto custo, e necessidade constante de serem atualizadas.

No próximo capítulo serão apresentadas algumas ferramentas de modelagem e feita uma descrição de elementos de um sistema automatizado.

Capítulo 3

Descrição de Elementos de um Sistema Automatizado e Ferramentas de Modelagem

3.1 **Introdução**

Segundo Rosário (2004), desde os anos 80, as estruturas das plantas fabris vêm se modificando rapidamente, em busca de melhor produtividade e racionalização dos recursos investidos a fim de atender as exigências do mercado e competição entre os fornecedores e à exigência dos clientes. Para tanto, a implementação de novos métodos de produção como as células flexíveis de manufatura e de linhas de produção automatizadas tornam-se necessárias para a obtenção destas melhorias.

Neste capítulo são apresentados conceitos teóricos e definições básicas referentes à Sistema Automatizados, desde ferramentas para Modelagem até a Integração de Sistemas Automatizados, permitindo assim ao leitor uma melhor familiaridade com os conceitos de Instrumentação Virtual e Sistemas Embarcados apresentados nos próximos capítulos desse trabalho.

3.2 **Sistemas Automatizados – Conceitos e Definições**

3.2.1 **Conceitos Básicos**

Para compreender um sistema automatizado é necessário entender inicialmente o conceito de sistema. Um sistema é qualquer interação de elementos cujo funcionamento visa alcançar um objetivo comum e que evoluiu com o tempo.

Portanto, segundo a definição citada, aquilo que pode ser definido como sistema num contexto pode ser apenas um componente de outro sistema, originando o conceito de subsistema.

Pode-se definir ainda o sistema como um conjunto complexo de coisas diversas que relacionadas entre si, contribuem para determinado objetivo ou propósito. Os Sistemas Automatizados podem ser classificados como:

- Automatismos Combinatórios: O estado das saídas depende do estado das entradas, ou seja, as saídas são determinadas unicamente em função do estado corrente das entradas, conseqüentemente o funcionamento do sistema não depende do tempo, conforme mostra a figura 3.1a.
- Automatismos Seqüenciais: O estado das saídas depende do estado atual das entradas do sistema. O funcionamento depende do seu passado. Conseqüentemente, o estado das saídas no instante t é função do estado das entradas neste tempo t e dos estados das saídas no tempo (t-1), conforme mostra a figura 3.1b.

a) Automatismos Combinatórios.

b) Automatismos Seqüenciais.

Figura 3.1 Classificação de Sistemas Automatizados.

Segundo Saramago (2002), sob o ponto de vista prático, define-se um sistema como um conjunto de elementos dinamicamente relacionados entre si, formando uma atividade para atingir um objetivo, operando sobre entradas (informação, energia ou matéria) e fornecendo saídas (informação, energia ou matéria) processadas, tendo como principais componentes:

- Fronteiras: limites do sistema, que podem ter existência física ou apenas uma delimitação imaginária para efeito de estudo.
- Subsistemas: elementos que compõem o sistema.
- Entradas: representam os insumos ou variáveis independentes do sistema.
- Saídas: representam os produtos ou variáveis dependentes do sistema.
- Processamento: engloba as atividades desenvolvidas pelos subsistemas que interagem entre si para converter as entradas e saídas.
- Retroação (feedback): é a influência que as saídas do sistema exercem sobre as suas entradas no sentido de ajustá-las ou regulá-las ao funcionamento do sistema.

3.2.2 **Modelagem**

Pode-se definir a Modelagem de um Sistema como uma representação de um objeto, sistema ou idéia em uma forma diferente ao elemento propriamente dito. Desta forma, o Modelo de um Sistema é um conjunto de informações sobre um determinado sistema com o propósito de entender o mesmo, ou seja, um modelo é uma descrição do sistema real.

Literalmente pode-se dizer que o modelo é a representação de alguma coisa, podendo ser definido também como a representação simplificada de um sistema com o propósito de estudar o mesmo.

Desta forma um modelo passa a ser uma réplica ou uma abstração com a característica essencial de um sistema ou processo. Através destes problemas que desobedecem a soluções diretas por causa do tamanho, complexidade ou estrutura, são freqüentemente avaliados através de modelos de simulação, ou seja, o Modelo passa a ser uma representação simplificada de alguma parte da realidade de sistemas, podendo ser estes sistemas de diferentes tipos.

Segundo Soares (1992), existe a concepção forte de que os modelos devam ser construídos para resolver problemas específicos. Desta forma, podemos ter vários modelos de um sistema, cada um mais adequado à resolução de um problema particular do que o outro. Apesar de serem construídos dependentes do problema a resolver, os modelos requerem uma estrutura organizada. Uma linguagem para simulação pode fornece uma destas estruturas, e sua compilação é o que vai traduzir a descrição do sistema em uma forma aceitável por um sistema de computação.

Segundo Morcelli (2004), ao desenvolver um modelo de simulação, o modelador precisa selecionar a estrutura conceitual na qual o modelo vai se apoiar para a descrição do sistema. A estrutura conterá o "enfoque" (ou visão), dentro do qual as relações funcionais entre os elementos do sistema são percebidas e descritas.

Estes modelos podem ser classificados como: físico (escala natural e reduzida) e matemático (numérico/algorítmico), sendo que as principais etapas necessárias para a obtenção dos mesmos são a análise do sistema, através da identificação de entidades, atributos, etc, e a simplificação, através da desconsideração das entidades e atributos considerados irrelevantes.

3.2.3 **Simulação**

A simulação de um sistema pode ser definida como a capacidade de conduzir experimentos utilizando um modelo de sistema real de forma a compreender o comportamento deste e possibilitar a avaliação de estratégias para a operação do mesmo, ou seja, é a técnica de resolver problemas seguindo as variações ocorridas ao longo do tempo.

Segundo Morcelli (2004), um importante fator para a larga utilização da simulação é a sua flexibilidade. A maioria das outras técnicas, tal como modelagem analítica, requer que os sistemas reais sejam transformados em um modelo idealizado em uma estrutura bastante específica. Quando esta idealização é possível sem um significativo compromisso da natureza do sistema considerado, a técnica analítica pode ser usada para obter uma solução.

Segundo Adam (1979), os objetivos da simulação podem ser classificados em quatro categorias:

- Comparação de estratégias com respeito a um problema individual no mundo real: A maioria das aplicações de simulação inclui-se nesta categoria. Estas aplicações iniciam com alguma análise estatística em relação a um problema específico e através da simulação procura-se encontrar uma solução para um problema.

- Desenvolvimento de afinidade funcional. Nesta situação, a utilização da simulação tem por objetivo obter critérios de relação entre variáveis que, posteriormente, com uma análise matemática poderá possivelmente chegar a uma solução ótima.

- Validação e avaliação de métodos analíticos recentemente desenvolvidos: É a utilização da simulação como ferramenta ou fonte de idéias para novas técnicas analíticas, como a aplicação para validação de métodos matemáticos ou novas teorias.

- Propósitos educacionais ou de treinamento: A simulação permite colocar uma pessoa frente a situações difíceis e analisar seu comportamento e suas decisões sem que isso possa colocar em risco o andamento normal do sistema. Desta forma, a Simulação é um importante dispositivo de aprendizado.

Porém, a utilização da simulação possui limitações, dentre as quais podemos destacar a dependência entre resultados e estímulos, o desenvolvimento de bons modelos podendo ser onerosos e a falta de precisão e/ou qualidade da modelagem no fornecimento de valores das variáveis em todos os instantes de tempo.

3.2.4 **Elementos de Modelagem de um Sistema Automatizado**

A evolução tecnológica está levando à crescente complexidade dos sistemas automatizados, implicando numa grande dificuldade por parte do usuário, na definição de uma maneira clara, concisa e não ambígua das especificações funcionais associadas a esses sistemas. Esta complexidade tende aumentar ainda mais, com a utilização de um número elevado de troca de informações entre os elementos constituintes deste sistema. Desta forma é necessário descrever o sistema através de uma linguagem de descrição adequada. Esta linguagem precisa ser do ponto de vista do homem, uma forma que expresse de modo natural a especificação do sistema e do ponto de vista do sistema, uma descrição simples que seja facilmente interpretada e executada (Aihara, 2000).

As linguagens verbais e textuais não são as mais indicadas para a modelagem dos SA, pois pode levar a mais de uma interpretação, e até mesmo a informações ambíguas. Portanto, em sistemas complexos, onde haja ações simultâneas e decisões com múltiplas possibilidades, deve-se evitar a utilização da linguagem verbal e textual. Sempre que possível e necessário, as descrições de sistemas automatizados devem ser representadas na forma gráfica, pois possuem uma facilidade maior em serem interpretadas e executadas, porém, existe a dificuldade de se encontrar uma forma que seja aceita e entendida por todos.

Visando a padronização de uma linguagem de descrição para os sistemas automatizados, o "International Electrotechnical Committe" estabelece uma nomenclatura internacional para sistemas automáticos, a norma internacional [IEC 61131-3]. Esta norma divide o Sistema Automatizado (SA) em duas partes distintas, como ilustrado na figura 3.2:

- **Parte Operativa (PO)** – corresponde ao processo físico a automatizar, que opera sobre a matéria prima e o produto. É constituída pelos atuadores que realizam as operações, agindo sobre componentes e dispositivos de automação, tais como válvulas, atuadores, motores, lâmpadas, etc;
- **Parte Comando (PC)** – caracterizado por receber as informações vindas do operador e/ou do processo a ser controlado e emitir informações ao sistema controlado, coordenando assim, as ações da Parte Operativa (PO).

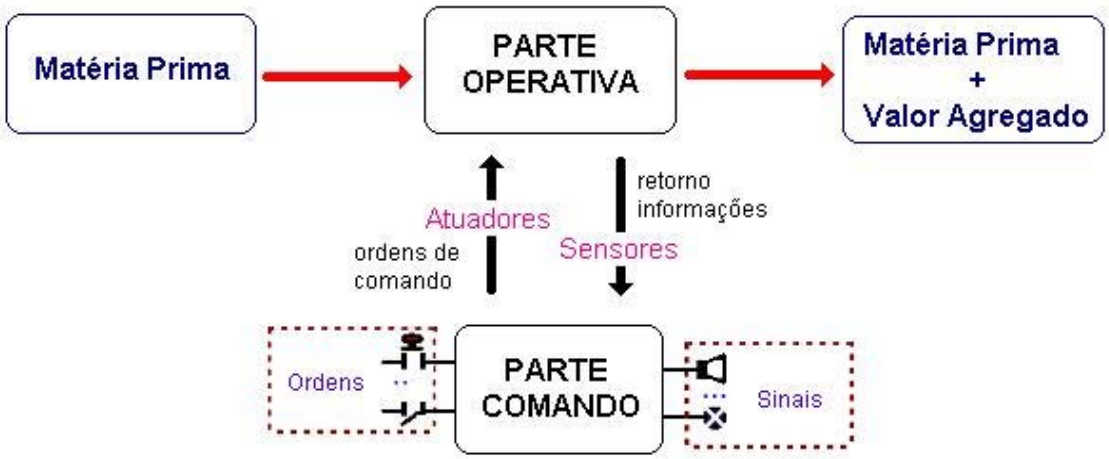


Figura 3.2 Sistema Automatizado (SA) – Parte Operativa e Parte Comando.

A Parte Comando de cada processo necessita, dentre as diferentes tecnologias de comando disponíveis, a mais adequada e a de melhor adaptação. Dentre as diferentes tecnologias existentes, podem-se citar os comandos pneumáticos, hidráulicos, eletro-mecânicos e os Controladores Lógicos Programáveis (CLP).

Desta forma, a Parte Comando é mantida informada sobre o estado dos subsistemas, através das informações fornecidas pela Parte Operativa. Esta pode ainda trocar informações com o exterior do sistema, de onde pode receber indicações, ordens (botões de comando, chaves, sensores, etc.) e fornecer sinalizações sonoras e/ou luminosas (sinalizadores, buzinas, lâmpadas, etc.).

Portanto, para o desenvolvimento de um SA deve-se inicialmente descrevê-lo de modo a não ficar nenhuma dúvida sobre os objetivos a serem atingidos no projeto proposto, sem a preocupação com detalhes tecnológicos, quando então se devem descrever os elementos específicos do sistema automatizado.

Segundo Araújo (1997), para se tentar obter a melhor descrição dos Sistemas Automatizados deve-se desenvolver seu Caderno de Tarefas, cujo objetivo é a descrição de seu comportamento em função da evolução das etapas que descrevem o processo a ser automatizado.

Um Caderno de Tarefas deve apresentar uma descrição clara, precisa e sem ambigüidades, nem omissões, do papel das etapas constituintes do processo a ser automatizado. Com isto, a descrição do sistema deve ser dividida em dois níveis sucessivos e complementares: nível 1 e nível 2.

O **nível 1 – Especificações Funcionais** - deve descrever o comportamento da parte comando PC em relação a parte operativa PO. As especificações funcionais permitem a compreensão das funções que o automatismo deve realizar, face às diferentes situações que podem surgir. Neste nível, pouco importa qual a forma que realizará um determinado movimento, mas é importante conhecer em que circunstância o deslocamento deve ser realizado. É neste nível que aspectos de segurança previstos para o funcionamento devem ser incorporados nas especificações funcionais, na proporção em que eles não dependam diretamente da tecnologia empregada.

O **nível 2 – Especificações Tecnológicas e Operacionais** - deve acrescentar às exigências funcionais um detalhamento das condições de funcionamento dos constituintes do processo a ser automatizado, através de especificações tecnológicas e operacionais. Neste nível, deve haver indicações sobre a exata natureza dos transdutores e atuadores, do modo como estes elementos deverão ser inseridos fisicamente no processo que compõe o sistema automatizado e informações acerca do seu meio ambiente de atuação.

Estas considerações são muito importantes para a exploração do processo a automatizar, considerando-se suas repercussões sobre o aspecto econômico, que são freqüentemente esquecidas durante a elaboração do caderno de tarefas, pois são difíceis de serem expressas de maneira quantitativa.

3.2.5 **Sistemas a Evento Discreto (SED)**

Segundo Cardoso e Valette (1997), sistema discreto é um sistema no qual as mudanças de estado ocorrem em instantes precisos. Porém esta classificação depende do ponto de vista em que se coloca o observador e depende do grau de abstração desejado. Este fato pode ser observado em um processo de fresagem, em um sistema de manufatura.

Do ponto de vista da operação de fresagem o sistema deve ser modelado por um modelo contínuo. Do ponto de vista da coordenação do sistema de manufatura, considerando os eventos relacionados ao inicio e fim de fresagem, o sistema deve ser modelado por um modelo a eventos discretos.

Desta forma, os Sistemas a Eventos Discretos (SED) são sistemas cujas variáveis de estado mudam somente num conjunto discreto de pontos no tempo, ou seja, os valores das variáveis nos estados seguintes podem ser calculados diretamente a partir dos valores precedentes e sem ter que considerar o tempo entre estes dois instantes. Como exemplo pode verificar um banco, cuja variável de estado, é o número de clientes no banco, sendo que este muda somente quando um cliente chega ou quando o serviço prestado a um cliente é completado. Portanto, os Sistemas a Eventos Discretos são sistemas cujos sinais:

- Podem assumir valores num conjunto discreto, tais como on/off, início, fim, verde, amarelo, vermelho, etc;

- Podem ter seus valores alterados rapidamente, de modo a permitir ser modelado como instantâneas, em qualquer instante t;

- Podem ser alterados por duas possíveis razões: ocorrência de eventos instantâneos externos, isolados e independentes; ocorrência de eventos internos, definidos por rigorosas cadeias lógicas.

Segundo Cassandras (1993), atualmente são inúmeros os sistemas a eventos discretos, sendo de fundamental importância na ordenação da vida civilizada contemporânea; ocorrendo em todas as indústrias, nos serviços prestados ao público, nos processos burocráticos, nos softwares de tempo real e dos bancos de dados e nas manufaturas. Nestes sistemas em geral intervêm eventos externos importantes, enquanto internamente existe uma lógica rigorosa de causas e efeitos.

Desta forma segundo Ramadge e Wonham (1989), a crescente complexidade dos sistemas automatizados tende a aumentar a utilização de um número elevado de informações de entradas e saídas, implicando na grande dificuldade por parte do usuário na definição das especificações funcionais associadas a esses sistemas.

3.2.6 **Sistemas Robóticos**

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

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

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

A robótica hoje, continua sendo utilizada para o desenvolvimento, estrutural e funcional de máquinas como: (Ferreira, 1991)

- Robôs manipuladores, com estrutura antropomórfica ou não, capazes de pegar objetos e deslocá-los, ou de atuar sobre objetos com ferramentas específicas;
- Robôs móveis, que se deslocam sobre rodas, patas ou lagartas;
- Robôs de supervisão, que verificam e selecionam objetos.

A robótica é considerada hoje a mola mestra de uma nova mutação dos meios de produção, isto devido a sua versatilidade, em oposição à automação fixa. Os robôs, graças ao seu sistema lógico ou informático, podem ser reprogramados e utilizados em uma grande variedade de tarefas, sendo que a reprogramação não é o fator mais importante na versatilidade desejada e sim a adaptação às variações no seu ambiente de trabalho, mediante um sistema adequado de percepção e tratamento de informação.

3.2.7 **Sistema de Supervisão e Controle em Automação Industrial**

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

Segundo Cosso (2002), atualmente pode-se definir um sistema supervisório como uma interface homem máquina (IHM) amigável, com recursos tecnológicos capazes de controlar e/ou supervisionar um sistema automatizado.

Portanto, um Sistema Supervisório é também a rigor, um sistema de comunicação no sentido mais amplo da palavra, pois engloba a visualização de todo o processo, que aliado a um sistema de informação tem a finalidade de manter um banco de dados atualizado, fornecendo em tempo real o posicionamento do sistema e, acoplado à uma interface com o usuário, pode interagir numa intervenção e/ou controle, e ainda conectar a parte operacional dos processos com os sistemas mais altos em hierarquia de planejamento. A figura 3.3 mostra uma representação de um sistema supervisório, segundo esta visão.

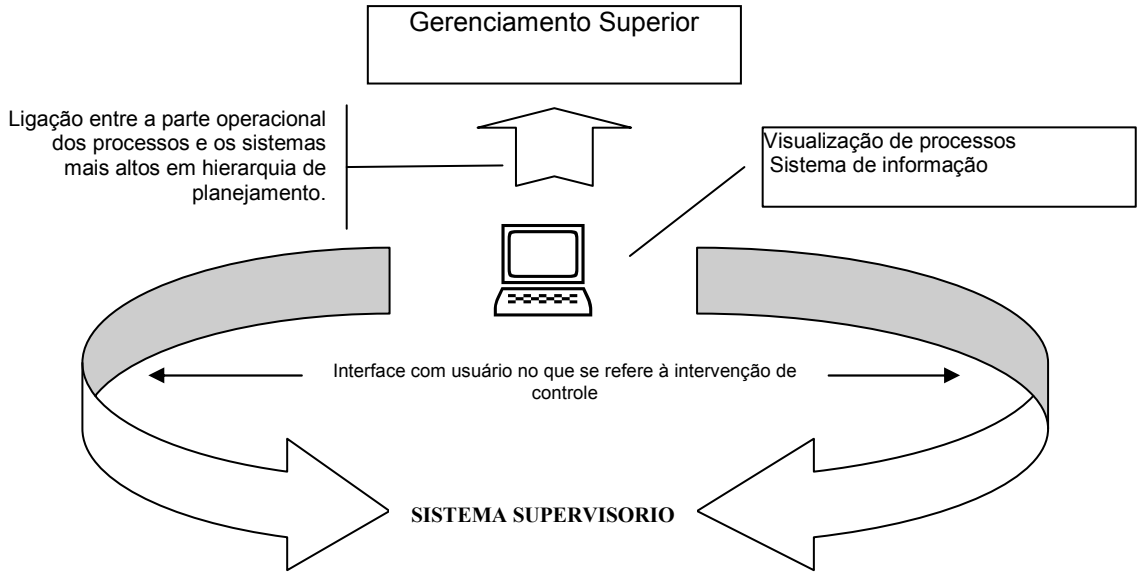


Figura 3.3 Representação de um sistema supervisório

Segundo Silva et al. (1998), nos sistemas automatizados o controle pode ser organizado nos seguintes níveis: Planta, Controle Local, Supervisão, Ordenação e Planejamento. A figura 3.4 também ilustra estes níveis, cuja forma piramidal referencia tanto a quantidade de recursos humanos envolvidos em cada nível, como também o número de controladores utilizados, ou seja um controlador do nível supervisão controla diversos controladores do nível Controle Local.

Nos Sistemas Automatizados a integração entre os postos de trabalho e a gestão de produção realizada a partir do Sistema de Supervisão tem como uma das tarefas receber os dados do "chão de fábrica" e colocá-los a disposição dos níveis superiores de gerenciamento. Esta integração possibilita o acompanhamento em tempo real de variáveis e estados representativos das operações em curso no chão da fábrica (postos de trabalho), com a finalidade de tomada de decisões operacionais, otimização dos processos e criação de históricos. As aquisições dos dados a serem repassados ao Sistema Supervisório provenientes dos postos de trabalho, são realizadas por CLP's e dispositivos de controle com interfaces de aquisição de informações, ou seja, o Controle Local interage com os dispositivos físicos da Planta.

Figura 3.4 Sistema de Controle - Níveis

O nível de Supervisão possui como características principais agir sobre o sistema de controle local de modo a executar as tarefas determinadas pelo nível de controle superior e supervisionar a evolução do processo na planta.

Desta forma o sistema de controle no nível supervisão é conhecido como sistema supervisório. Estes sistemas revelam-se de crucial importância na estrutura de gestão de Sistemas, e, por isso, deixaram de ser vistos como mera ferramenta operacional ou de engenharia, e passaram a ser visto como uma importante fonte de informação e controle.

O Sistema Supervisório recebe também orientações do Sistema de Gestão da Produção para determinar as operações de produção, conseqüentemente deve dialogar com os sistemas localizados hierarquicamente acima e abaixo dele. A figura 3.5 apresenta a arquitetura típica de um sistema supervisório do tipo SCADA (*Supervisory Control and Data Acquisition*).

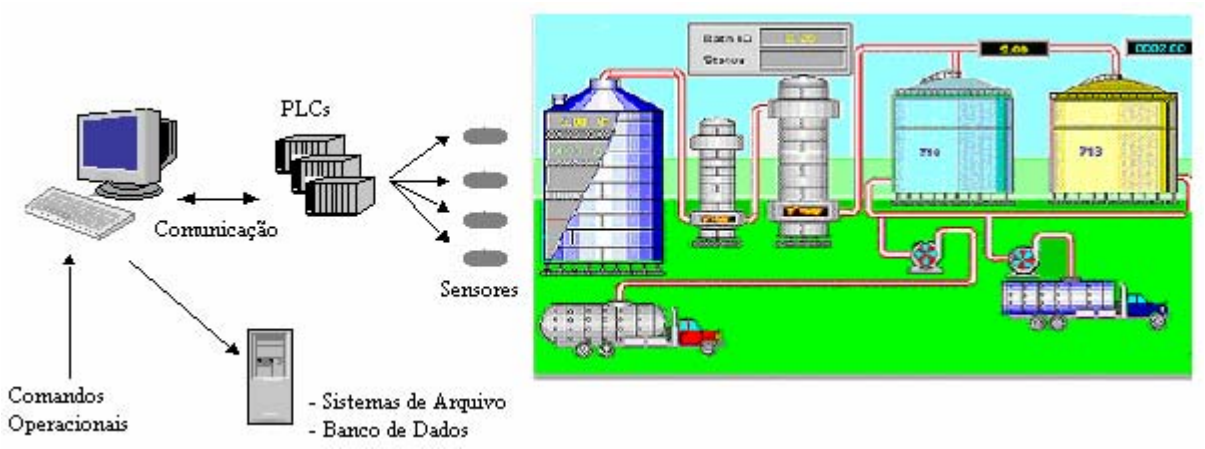


Figura 3.5 Arquitetura Sistema SCADA

Os primeiros sistemas SCADA eram basicamente telemétricos, permitiam informar periodicamente o estado corrente do processo industrial, através da monitoração de sinais representativos de medidas e estados de dispositivos, utilizando-se de painéis de lâmpadas e indicadores sem que houvesse qualquer interface de aplicação com o operador.

Através da evolução tecnológica, os computadores passaram a assumir um papel de gestão na aquisição e tratamento de dados, permitindo a sua visualização em períodos curtos de tempo e ainda permitindo a geração de funções de controle complexas.

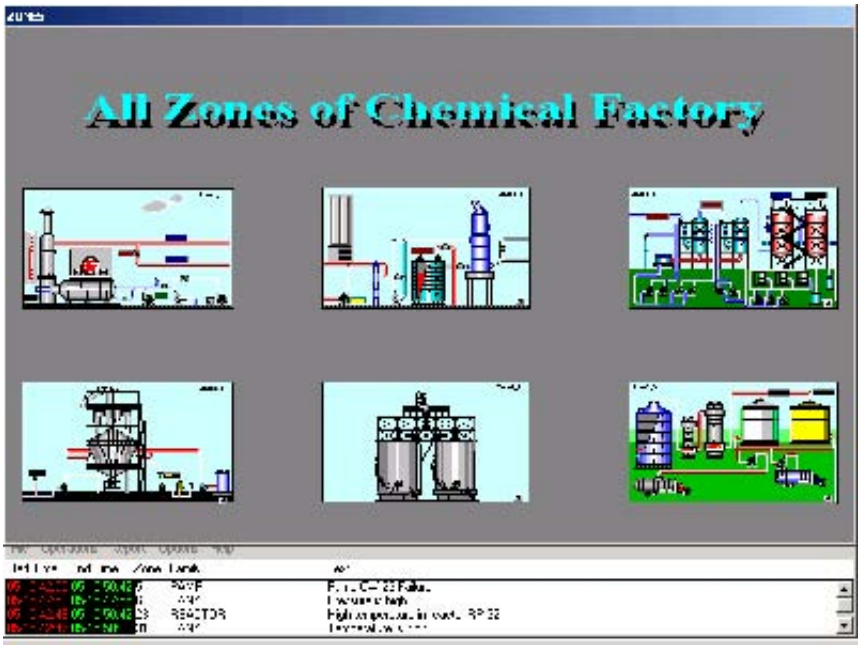
Dentro desta evolução tecnológica, a evolução dos sistemas de comunicação através das redes de comunicação, permite tal controle, uma vez que a mesma é utilizada como plataforma pelos sistemas supervisórios para a transferência de informações.

Atualmente os sistemas supervisórios estão sendo utilizados para automatizar a monitoração e o controle de Sistemas Automatizados, através do recolhimento de dados em ambientes complexos, podendo estes ambientes se encontrarem eventualmente dispersos geograficamente, além de apresentar uma visualização de modo amigável para o operador, através de Interface Homem-Máquina (IHM) altamente sofisticada.

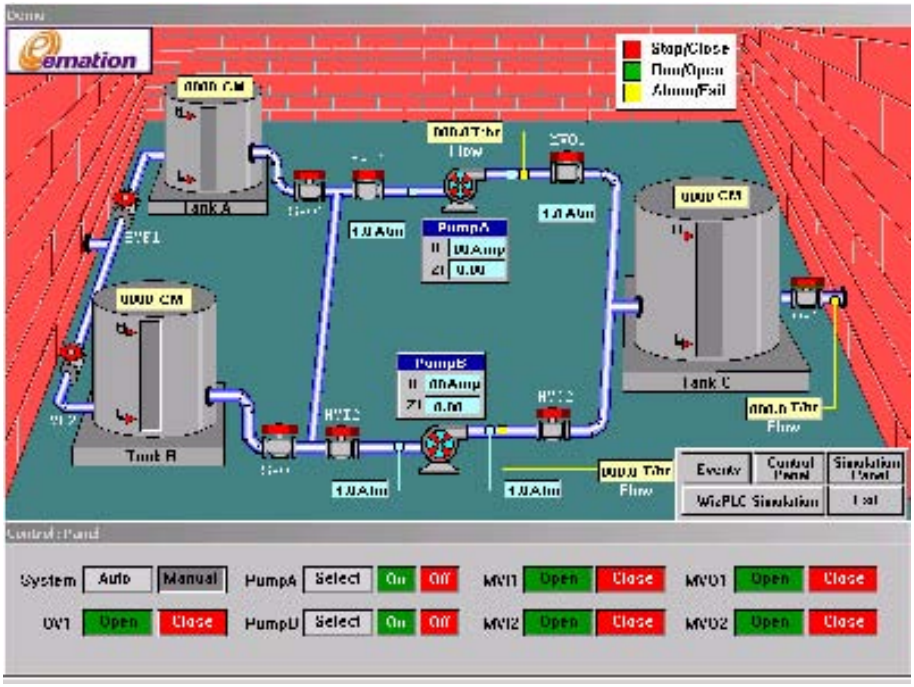
Num ambiente industrial cada vez mais complexo e competitivo, os fatores relacionados com a disponibilidade e segurança da informação assumem elevada relevância tornando-se necessária a garantia de que a informação estará disponível e segura, quando necessária, independentemente da localização geográfica. Portanto, tornam-se necessárias implementações de mecanismos de acessibilidade, mecanismos de segurança e mecanismos de tolerância à falhas.

Com isto, os sistemas SCADA melhoram a eficiência do processo de monitoração e controle, disponibilizando em tempo útil o estado atual do sistema através de um conjunto de previsões gráficas e relatórios, de modo a permitir a tomada de decisões operacionais apropriadas, quer automaticamente, quer por iniciativa do próprio operador, ou seja, a supervisão atua de maneira automática e normalmente conta com o auxílio de um operador que poderá interferir no sistema por intermédio de uma interface. Esta se constitui num elemento de fundamental importância no Sistema de Supervisão, devendo permitir a monitoração dos processos de modo hierárquico. A figura 3.6a e 3.6b apresenta algumas telas gráficas de monitoração e controle de variáveis em processos industriais.

Desta forma, um sistema supervisório de controle e aquisição de dados – SCADA passa a ser uma ferramenta largamente utilizada na indústria, para o desenvolvimento de aplicações que permitam aos integradores de sistemas gerarem sofisticadas aplicações industrias para uma variedade de indústrias.



a. Tela Gráfica Típica



b. Tela Gráfica de monitoramento de Variáveis

Figura 3.6 Telas Gráficas

3.3 **Ferramentas de Modelagem**

3.3.1 **Grafo de Comando Etapa e Transição - GRAFCET**

Devido à variedade de CLP's existentes no mercado, o desenvolvimento de programas e ferramentas de descrição devem ser adequadas permitindo o desenvolvimento de tarefas, independente da PC utilizada e que adicionalmente ofereçam a opção de se programar em uma linguagem mais natural para os vários níveis de usuários. Desta forma pode-se dividir o problema em várias etapas, tornando-o mais simples e facilitando a visualização das seqüências de operações, alteração de especificação e detecção de falhas conceituais no programa.

O SFC - "Sequential Function Chart" ou Diagrama Funcional Seqüencial, também conhecida pelo nome de GRAFCET, oferece vantagens aos usuários e programadores, principalmente na modelagem de problemas complexos de automação, pois se pode dividir o problema em várias partes, tornando mais simples a programação, facilitando a visualização das seqüências de operações, alteração de especificação e a detecção de falhas conceituais no programa.

O GRAFCET (Grafo de Comando Etapa - Transição) foi criado na França em 1975, através de um grupo de pesquisadores e gerentes industriais, envolvidos com sistemas discretos de grande complexidade, sendo coordenados pela AFCET – "Association Française pour la Cybernètique, Economique, Technique" e posteriormente padronizado pela ADEPA – "Agencie Nationale pour la Developpement de la Prodution Automatisée", sendo considerado uma particularização das Redes de Petri, pois as redes possuem uma possibilidade de aplicações bem superior às que estão restritos os comportamentos cíclicos das máquinas e sistemas automáticos.

Com a combinação de elementos temos a representação estática do Sistema Automatizado. Aplicando-se as Regras de Evolução, obtemos a visão dinâmica do mesmo. Isto pode ser verificado, no exemplo da figura 3.7, em que um motor inicia desligado. Quando acionado um botão o motor liga e somente após a desativação da botoeira o motor é desligado e assim sucessivamente.

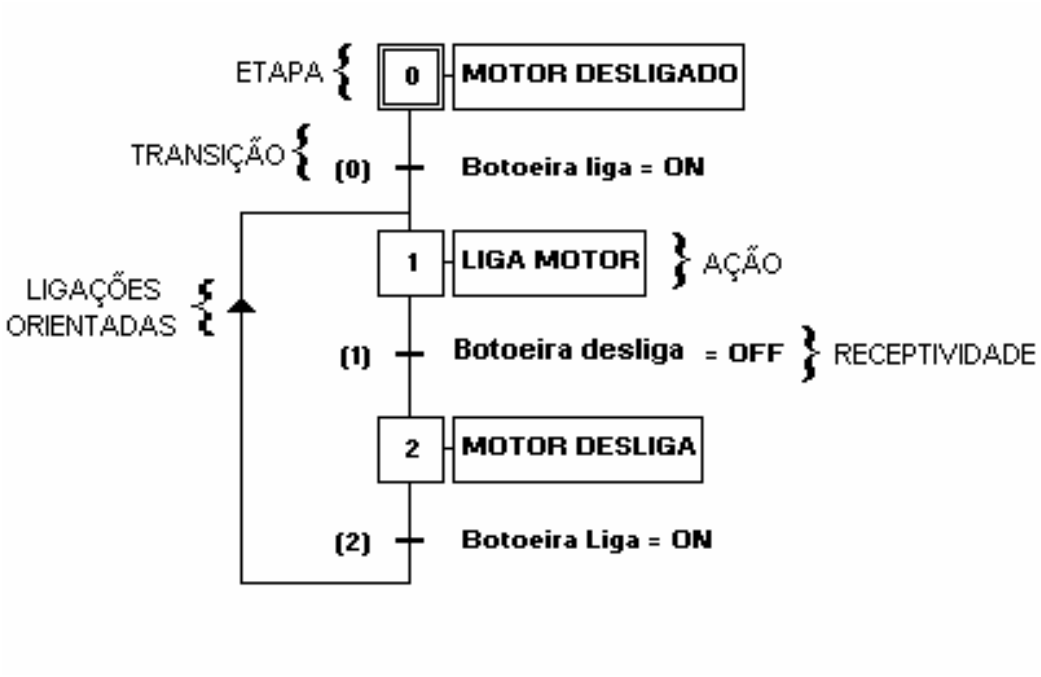


Figura 3.7 Exemplo de GRAFCET

Algumas técnicas utilizadas atualmente para descrever comportamento seqüencial em sistemas automatizados incluem fluxogramas, diagramas de variáveis de estado, Rede de Petri, diagrama trajeto-passo e o GRAFCET.

3.3.2 **Redes de Petri - Conceitos Básicos e Definições**

A metodologia de modelagem de Sistemas a Eventos Discretos (SED) utilizando Redes de Petri, (RdP) foi proposta em 1962, por Carl Petri, matemático alemão, que através de uma tese de doutoramento criou esse método de estudo para sistemas dinâmicos a evento discreto, direcionado às Comunicações com Autômatos, originando posteriormente, duas grandes linhas de desenvolvimento nas áreas de Ciências da Computação e em Engenharia de Sistemas padronizando as Redes de Petri (Huber, Jensen e Shapiro, 1990).

Portanto as Redes de Petri são ferramentas gráficas e matemáticas de modelagem para descrição e/ou especificação que podem ser aplicadas a diversos tipos de sistemas apresentando um bom nível de abstração em comparação com outros modelos gráficos.

Além disso, as Redes de Petri possibilitam a verificação do sistema especificado. Usando-se RdP, pode-se modelar sistemas paralelos, concorrentes, assíncronos e não-determinísticos (Melo e Sobreira, 2003).

Desta forma a RdP é uma linguagem formal que permite a modelagem de sistemas dinâmicos discretos com grande poder de expressividade, permitindo representar com facilidade todas as relações de causalidade entre processos em situações de: seqüencialidade, conflito, concorrência e sincronização (Matos e Santos, 2004). Sendo que a sua aplicabilidade em diversas áreas transformou-o em tema alvo de investigação básica e aplicada, e sua utilização na Modelagem de Sistemas Automatizados apresenta algumas vantagens como:

- captura das relações de precedência e os vínculos estruturais dos sistemas reais;
- graficamente expressivas, permitindo a modelagem de conflitos e filas;
- possui fundamento matemático e prático;
- admite várias especializações (RP's temporizadas, coloridas, estocásticas, de confiabilidade etc.).

Portanto, podem-se definir as Redes de Petri (RdP's) por meio de conjuntos, funções e também por grafos, de maneira que suas propriedades possam ser obtidas pela teoria dos conjuntos e/ou pela teoria dos grafos.

As redes de Petri permitem modelar sistemas constituídos por componentes que apresentem características de funcionamento concorrente e interatuantes. Segundo Peterson (1981) a sua utilização poderá ser realizada de modos diversos, dependendo do objetivo em vista.

Uma abordagem possível de utilização das RdP considera-as como uma ferramenta auxiliar de análise, sendo que nesta abordagem, outras técnicas e formalismos são utilizados para especificar o sistema. Com base nessa especificação, o sistema é então modelado através de uma RdP que será posteriormente analisada. Se forem detectados problemas, procedem-se as alterações na especificação e o ciclo será repetido até que mais nenhum problema seja detectado. A figura 3.8 demonstra este processo.

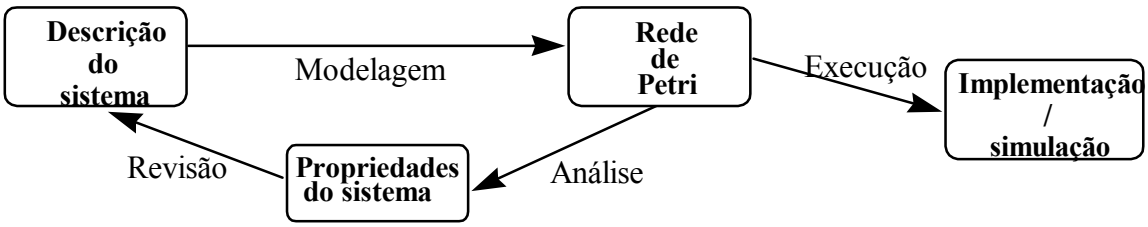


Figura 3.8 Utilização de redes de Petri no modelamento e análise de sistemas

3.4 **Conclusão**

Este capítulo apresenta a multidisciplinaridade que envolve a área de automação industrial através da apresentação dos diversos assuntos que se encontram inseridos no conhecimento de sistemas automatizados de produção. Desta forma, ao permitir o conhecimento dos assuntos tratados neste capítulo, permite-se também uma visão da decomposição do Sistema Automatizado em PO e PC, através dos elementos de cada parte e seu sistema de controle, de maneira a abordar através de uma visão geral do sistema para os subsistemas que o compõe este todo.

Capítulo 4

Instrumentação Virtual e Prototipagem Rápida

4.1 **Introdução**

Durante os últimos anos, engenheiros e pesquisadores vêm utilizando a instrumentação virtual agregando a potencialidade e a flexibilidade de software e a tecnologia de computadores pessoais em aplicações de teste, controle e projetos, assim realizando medições analógicas e digitais com grande exatidão em altas faixas de freqüência (até 2.7 GHz).

Atualmente, a instrumentação virtual se tornou necessária, pois fornecem a rapidez de adaptação necessária para as atuais necessidades de processos de projetos, desenvolvimentos, testes e entrega de produtos, já que, com a instrumentação virtual, é possível implementar novas funcionalidades no dispositivo sem que o hardware seja alterado, o que aumenta o tempo de vida útil do produto, reduz custos e possibilita o lançamento mais rápido de um determinado produto. Isto é possível porque a instrumentação virtual combina as principais tecnologias comerciais, como um PC, com software flexível e uma grande variedade de hardware para controles e medições.

A Instrumentação Virtual trouxe muitas contribuições a importantes áreas da Mecatrônica, dentre elas podemos destacar:

a) **Controladores Programáveis para Automação (PACs):** que combinam a funcionalidade de um PC e confiabilidade de um CLP, e hoje estão cada vez mais sendo incorporados em sistemas de controle.
b) **Tecnologia de Hardware reconfigurável (FPGA):** utilizado para Prototipagem Rápida de Controle e Simulação por Hardware In the Loop (HIL) que estão sendo utilizados com grande freqüência no Campo da Mecatrônica, evitando assim a construção de muitos protótipos durante as etapas de desenvolvimento de um produto, implicando em significativa redução de custo e tempo de desenvolvimento de um produto.

c) **Sistema de Visão Robótica:** utilizado cada vez mais freqüentemente na área de Mecatrônica, aonde muitos fabricantes vêm desenvolvendo sistemas dedicados para visão de máquina e processamento de imagens.

Neste capitulo relataremos conceitos básicos de Instrumentação Virtual, tendo como principal referência a plataforma LABVIEW™ da National Instruments, e suas diversas áreas de aplicação, apresentando também os principais benefícios proporcionados por esta tecnologia.

4.2 **Instrumentação Virtual**

O conceito de Instrumentação Virtual surgiu a alguns anos, mudando a forma como engenheiros e pesquisadores abordem problemas na área de Automação e Controle Industrial. Nos dias atuais, este conceito tem atingido níveis de aceitação no mercado bastante elevados, passando a ser usada em milhares de aplicações ao redor do mundo, como em indústrias automotivas, de bens de consumo e de óleo e gás.

Instrumentos Virtuais são instrumentos definidos pelo usuário por meio do desenvolvimento de uma aplicação. Isso significa desenvolver um software baseado nos requerimentos do usuário e definir um hardware de propósito geral para medição e controle.

A Instrumentação Virtual combina as tecnologias comerciais mais recentes com software e uma vasta variedade de hardware para medição e controle. Assim, engenheiros e cientistas podem criar sistemas que atendam exatamente as necessidades do usuário. Além disso, conseguem reduzir tempo de desenvolvimento, projetar produtos de maior qualidade e baixar os custos de seus projetos. Isso pode ser notado, por exemplo, na indústria de telefonia móvel. Há pouco tempo, os aparelhos celulares possuíam áudio, agenda e enviavam mensagens. Já os aparelhos de última geração, além dessas características, também possuem câmera, MP3 player, Bluetooth e conexão à Internet. Isso só é possível porque os aparelhos eletrônicos têm se tornado cada vez mais centrado em software.

Assim, engenheiros e pesquisadores devem adicionar novas funções ao aparelho sem substituir o hardware, resultando em melhores produtos, sem os custos de re-desenvolvimento do hardware, podendo acarretar uma interação com o usuário não previsto, sujeita a presença de erros. Nesse sentido, também se torna necessária a presença de ferramentas para a verificação constante das alterações feitas na aplicação. A única forma de atender a esses requisitos é pelo uso de arquiteturas de teste e controle também centradas em software.

A Instrumentação Virtual está muito associada ao software, tornando-se assim uma arquitetura capaz de atender características, adaptar-se a novas funcionalidades de um novo cenário de desenvolvimento de produtos, tornando mais rápido a desenvolvimento de produtos na medida em que provê um ambiente gráfico de programação.

Nesse ambiente, funções são colocadas dentro de blocos gráficos interligados, tornando o trabalho do programador muito mais rápido e fácil, possuindo também as seguintes características:.

a) **E/S modulares:** projetadas para rapidamente atender a qualquer tipo ou quantidade de sinais, tornando possível a monitoração e controle de processos presentes nas fases de desenvolvimento de um projeto. Assim, drivers de instrumento bem projetados possibilitam que engenheiros e pesquisadores façam uso de modo rápido e eficiente dos módulos de E/S da aplicação.

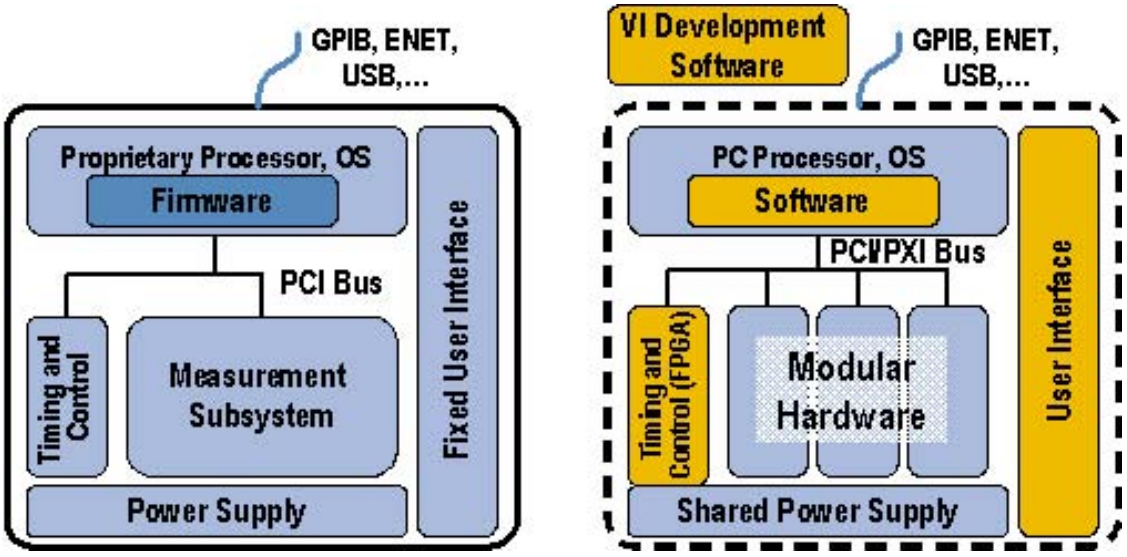
b) **Uso de plataformas comerciais:** freqüentemente incrementadas com sincronização precisa de sinais. Isso assegura o aproveitamento das capacidades tecnológicas mais recentes, bem como de taxas de transferência de dados mais rápidas. Esse aspecto faz da Instrumentação Virtual uma tecnologia duradoura que dispensa altos investimentos feitos em processadores, barramentos e outros.

A Instrumentação Virtual faz uso de software, E/S modulares e plataformas comerciais, o que a torna unicamente qualificada para manter-se adaptada ao atual conceito de desenvolvimento de produtos.

4.2.1 **Instrumentação Virtual x Tradicional**

Instrumentos Virtuais são definidos pelo usuário, enquanto que instrumentos tradicionais possuem funcionalidades fixas e definidas pelo fabricante. Um instrumento virtual é constituído de duas partes: software e hardware. Ao adotar hardware e software não definidos pelo fabricante, engenheiros e cientistas contam com a flexibilidade que a definição pelo usuário oferece.

Um instrumento virtual é tipicamente mais caro inicialmente se comparado ao tradicional quando ambos realizam tarefas de medição similares. Contudo, esse custo é dissolvido ao longo do tempo, pois Instrumentos Virtuais são muito mais flexíveis quanto a mudanças nas tarefas de medição. A figura 4.1 apresenta propostas de arquitetura tradicional e virtual.



a) Tradicional **b) Virtual**

Figura 4.1: Arquitetura de Instrumentação baseadas em software (NI Instruments).

Um Instrumento Tradicional é definido pelo fabricante, que fornece o software e hardware de medição combinados em um produto, que possui uma lista fixa e finita de funcionalidades. Ele é definido pelo usuário, que fornece o software e o hardware necessários para atender uma tarefa de medição ou controle.

Através de um Instrumento Virtual, o usuário pode customizar a aquisição, a análise, a armazenagem de dados, o compartilhamento de informações e a forma como elas são apresentadas por meio de um software poderoso e produtivo.

A figura 4.2 apresenta uma comparação entre as capacidades de hardware de medição entre Instrumentação Virtual e Tradicional. O objetivo da Instrumentação Virtual é deslocar a curva no sentido de crescimento da freqüência e da resolução e promover inovações no interior da curva.

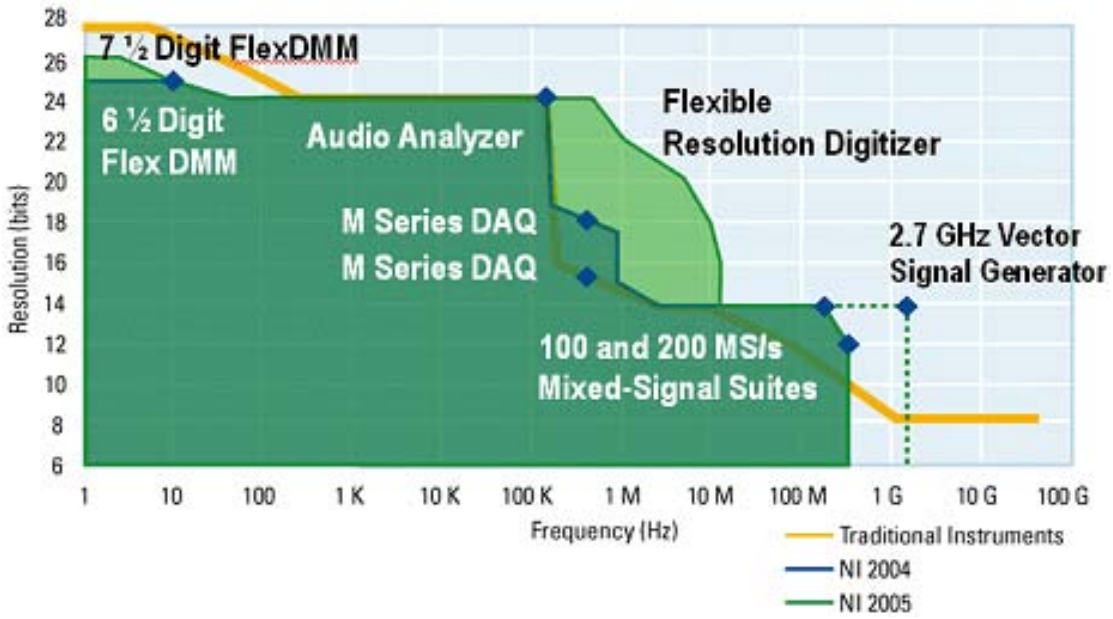


Figura 4.2: Comparação entre hardware para Instrumentação Virtual e Tradicional (National Instruments).

Devido à Instrumentação Virtual ser baseada em software, se algo pode ser digitalizado, ele pode ser medido. Conseqüentemente, o hardware de medição pode ser resumido em duas características: resolução (bits) e freqüência de amostragem. Na figura 4.4 podemos observar como as potencialidades de hardware para instrumentação virtual são comparadas a instrumentos tradicionais.

Muitos engenheiros e pesquisadores têm combinado Instrumentação Virtual e Tradicional em seus laboratórios, uma vez que alguns instrumentos tradicionais oferecem medição especializada, que no caso o maior interesse seria um instrumento definido por fabricante do que definido por ele mesmo.

É importante destacar que Instrumentos Virtuais são compatíveis quase sem restrições aos Tradicionais. Isso é possível, pois software para Instrumentação Virtual fornece bibliotecas para interfaceamento com barramentos comuns tais como GPIB, serial, USB ou Ethernet.

Além disso, existem mais de 200 fabricantes que fornecem com mais de 4000 drivers de instrumento para Instrumentação Virtual. Drivers de instrumento fornecem uma série de funções de alto nível para interfaceamento com instrumentos. Cada driver de instrumento é especificamente elaborado para um dado modelo de instrumento e provê uma interface para suas características únicas.

Devido a esses fatores, uma forte tendência da indústria de testes automatizados é a adoção de sistemas de teste baseados em software, como é o caso do Departamento de Defesa dos Estados Unidos (DoD), um dos maiores consumidores de equipamentos de teste automatizados (ETA) do mundo, que teve como principal objetivo a redução do custo da aquisição sistemas de teste e aumentar o reaproveitamento desses sistemas, estabelecendo uma arquitetura baseada em hardware modular e software reconfigurável, denominada instrumentação sintética.

Instrumento Sintético é definido como “um sistema reconfigurável que conecta hardware e software com interfaces padrões para gerar sinais ou fazer medições usando técnicas de processamento numérico”. Essas propriedades são muito semelhantes às da Instrumentação Virtual, que é “um sistema definido por software, em que o software se baseia nos requerimentos do usuário e define as funcionalidades de um hardware genérico para medição”. Ambas as definições utilizam de instrumentação definida por software e executada em hardware comercial. No tocante a hardware reconfigurável e acessível ao usuário, os dois tipos de instrumentação adotam os benefícios de tal arquitetura alcançando maior flexibilidade e reconfigurabilidade dos sistemas, o que aumenta o desempenho de suas capacidades e reduzem custo.

É importante destacar, assim, que para se obter êxito em implementações de sistemas de teste baseados em software, tais como a instrumentação sintética, é necessário ter conhecimento sobre as plataformas de hardware e as ferramentas de software presentes no mercado.

4.2.2 **Concepção de Instrumentos Virtuais**

Na concepção de Instrumentos Virtuais em Automação podemos utilizar diferentes modos de integração, função de fatores como flexibilidade, custo, rapidez e arquitetura aberta em nível de software ou hardware. Dois procedimentos descritos a seguir:

a) **Integração de diferentes tipos de dispositivos numa mesma aplicação**: onde não importando a escolha, um único programa poderá ser usado para qualquer das aplicações implementadas, sem a necessidade de se fazer qualquer mudança em seu código (figura 4.2).

b) **Integração de diferentes aplicações num mesmo dispositivo:** onde diferentes projetos poderiam ser combinados em uma única aplicação (programa), utilizando um único dispositivo de controle e aquisição de dados (figura 4.3).

A figura 4.3 exemplifica a integração de três diferentes dispositivos numa mesma aplicação (programa) usando Instrumentação Virtual através de um desktop e barramento PCI. Essa aplicação consiste de dispositivo para medir tensão DC e temperatura, e sistema para distribuir a aplicação para um sistema PXI, que trabalhará em chão de fábrica. O padrão USB para as tarefas poderá ser utilizado no caso que a aplicação exija portabilidade.

A figura 4.4 mostra a integração de diferentes aplicações num mesmo dispositivo, através de um projeto em que um dispositivo de aquisição de dados e encoders de quadratura são utilizados para medir a posição de um motor e outro dispositivo para monitorar sua potência. Neste exemplo, embora ambos os projetos pudessem ser combinados numa única aplicação utilizando um único dispositivo de aquisição de dados, são utilizados sistemas diferentes dentro de um mesmo projeto (programa).

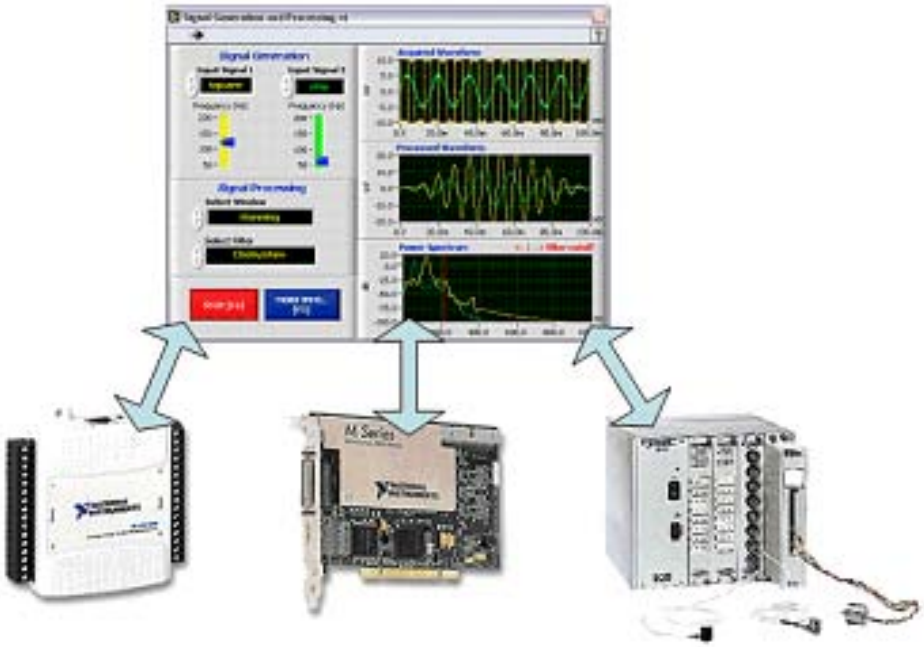


Figura 4.3: Integração de diferentes tipos de hardware numa mesma aplicação (National Instruments).

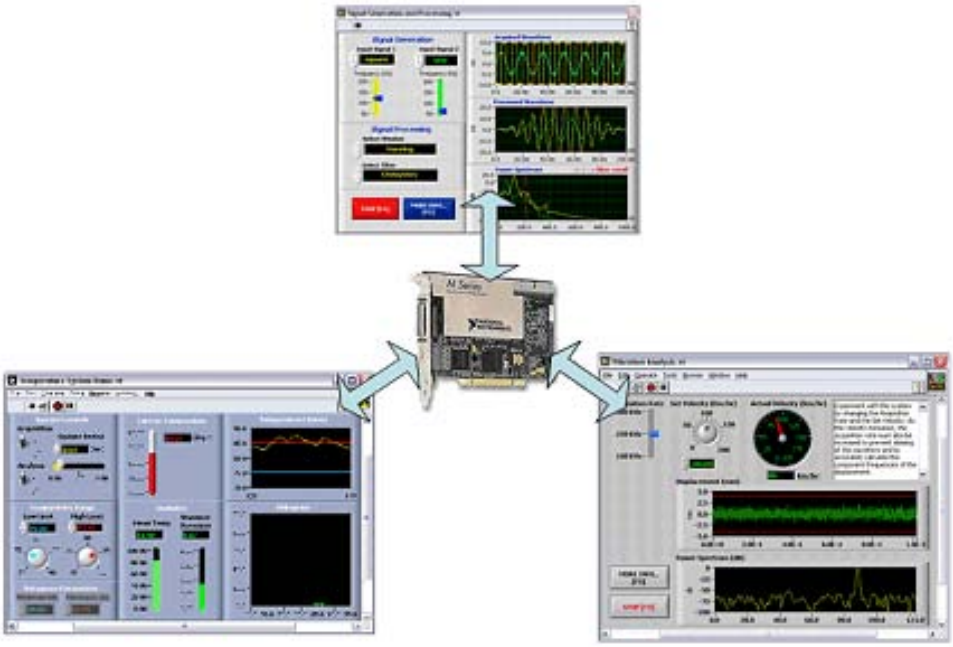


Figura 4.4: Integração num mesmo hardware de diferentes tipos de aplicações (National Instruments).

Outro importante conceito da Instrumentação Virtual é a estratégia adotada para se adaptar à evolução de hardware e software. Essa estratégia é focada no uso dos altos investimentos de grandes companhias de tecnologia e visa oferecer hardware e software compatíveis com o mercado. Como a Instrumentação Virtual é baseada em software, uma tarefa de medição só é possível devido à digitalização. Assim, hardware de medição pode ser analisado sob dois aspectos: resolução (bits do conversos AD) e freqüência (taxa de amostragem).

4.3 **A função do Software na Instrumentação Virtual**

Todo Instrumento Virtual é construído com um software poderoso e flexível por um especialista que aplica seu conhecimento para personalizar uma aplicação de controle e medição. O resultado é um instrumento específico definido pelo usuário de acordo com as necessidades da aplicação. Ele é constituído de três camadas (figura 4.5):

a) **Software de Aplicação:** Esta camada é a mais comum, constituindo um ambiente preliminar de desenvolvimento para construção da aplicação. Como exemplos de software para aplicação podemos citar LabVIEW, LabWindows/CVI (ANSI C), Visual .NET, etc.

b) **Software para testes e gerenciamento de dados:** Acima da camada de software para aplicação encontramos a camada de software para execução de testes e gerenciamento de dados. Esta camada de software incorpora todas as funcionalidades desenvolvidas pela camada de aplicação e provê um gerenciamento completo dos dados do sistema.

c) **Software para Serviços de Controle e Medição:** A última camada é freqüentemente negligenciada, ainda que crítica para manutenção na produtividade em desenvolvimento de software. A camada de serviços para controle e medição inclui drivers, que realizam a comunicação com o hardware, devendo acessar e preservar as funções e o desempenho do hardware além de ser inter-operável, ou seja, deve trabalhar com todos os outros drivers e com diversos tipos de E/S modulares que podem fazer parte da solução desenvolvida.

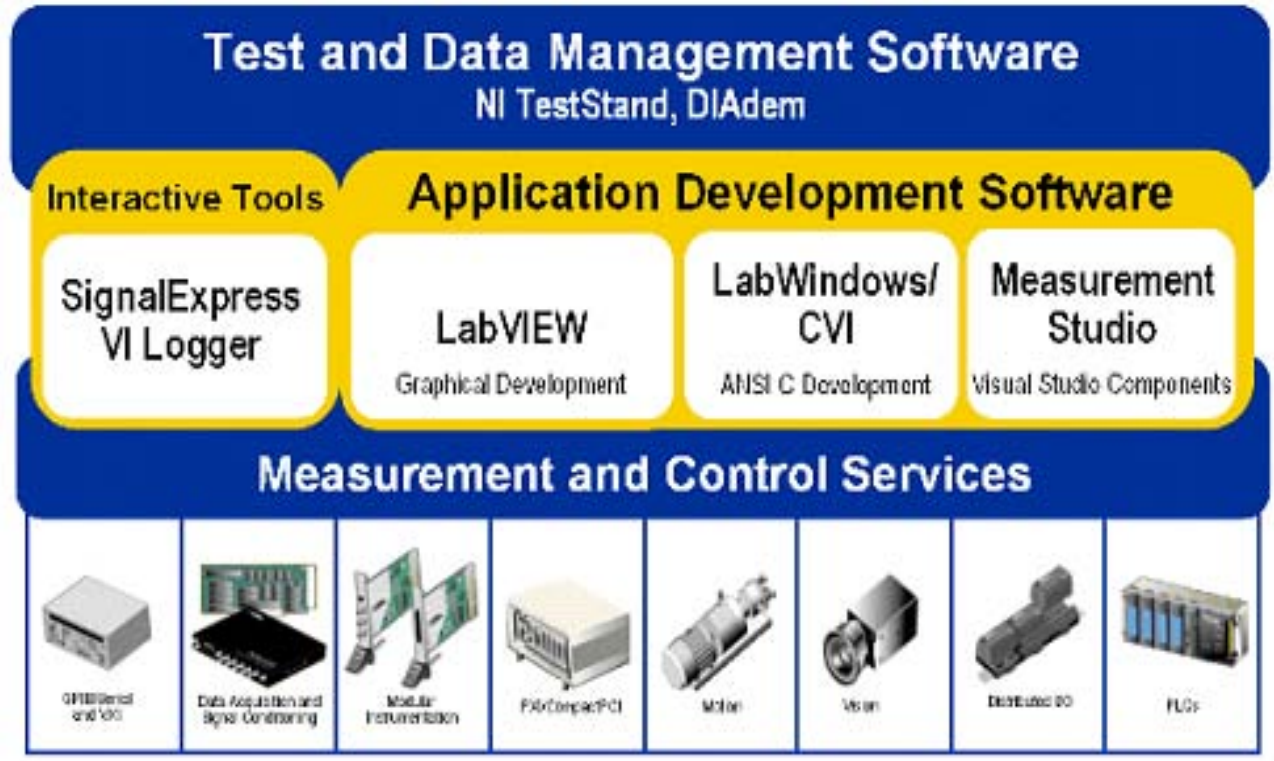


Figura 4.5: Camadas de Software para Instrumentação Virtual (NI Instruments).

Um software de serviços para controle e medição é equivalente a camada do software de driver de E/S, entretanto, este é mais complexo do que somente drivers. Embora freqüentemente negligenciado, este é um dos elementos mais cruciais no rápido desenvolvimento de uma aplicação, pois o mesmo conecta o software de instrumentação virtual com o hardware para medição e controle. Isto inclui a implementação de interfaces para programação das aplicações, drivers para comunicação com instrumentos, ferramentas de simples configuração, assistentes para configuração de E/S e outros tipos de funcionalidades.

Soluções utilizando instrumentação tradicional, de acordo com a natureza de suas funcionalidades fixas e definidas pelos fabricantes, não podem rapidamente se adaptar a novas tecnologias de software. Devido a esta necessidade de flexibilidade, a instrumentação virtual é mais bem adaptada para incorporar novas ferramentas e tecnologias, pois os usuários podem simplesmente atualizar a versão do software ao invés de adquirir um novo sistema.

4.4 **A função do Hardware na Instrumentação Virtual**

Um importante conceito na Instrumentação Virtual é a estratégia que potencializa evolução do dispositivo de hardware e software utilizado atualmente. As E/S possuem um importante papel na instrumentação virtual. Elas permitem acelerar aplicações de teste, controle e projeto, as E/S de hardware, e conseqüentemente precisam ser rapidamente adaptáveis a novos conceitos e produtos.

A Instrumentação Virtual proporciona esta capacidade na forma de modularidade a partir de plataformas de hardware escaláveis. Existe uma diversidade e hardwares de instrumentação virtual que abrangem diversos tipos de E/S que podem ser utilizadas em diversas aplicações como entradas e saídas analógicas e digitais, contadores e temporizadores, aquisição de imagens e controle de movimento.

As E/S modulares também incluem instrumentos modulares como osciloscópios, medidores LCR, geradores de função, dentre outros. Desta maneira há a possibilidade de escolher de maneira arbitrária os tipos de E/S necessários para uma determinada aplicação e pode-se assegurar que estes diversos tipos de E/S trabalhem com uma forte integração, já que é possível a utilização dos mesmos recursos, como por exemplo, temporização ou sincronismo.

Plataformas de hardware padrão que possuem E/S são importantes para modularidade. Computadores desktop e laptop proporcionam uma excelente plataforma onde a instrumentação virtual pode ser aplicada utilizando o máximo dos padrões de barramentos existentes como USB, PCI, Ethernet, PCMCIA, Express Card e PCI Express.Escolher a plataforma apropriada para criar uma aplicação de instrumentação virtual depende dos requisitos específicos da aplicação. Por exemplo, portabilidade, sincronização e taxas de aquisição.

4.4.1 **Os barramentos USB e PCI Express da NI para Instrumentação Virtual**

A Instrumentação Virtual utiliza os avanços das tecnologias de computadores disponíveis no mercado para tornar as medições mais rápidas e com melhor desempenho, mas com um custo menor se comparado aos instrumentos tradicionais. Um exemplo disto são os barramentos do PC, enquanto as interfaces de comunicação com instrumentos como serial e GPIB não tem sido modificadas por décadas, os novos barramentos dos PCs proporcionam drásticas melhorias em relação à largura de banda e facilidade de uso.

Atualmente, os barramentos de dados como o PCI Express e o USB 2.0 estão evoluindo de maneira semelhante em relação à velocidade, e o desenvolvimento de Bons programas de instrumentação virtual utilizam a evolução destas novas tecnologias enquanto minimizam o impacto na aplicação.

A largura de banda de 132 MB/s proporcionada pelo barramento PCI 33MHz de 32-bits ainda presente na maioria dos PCs Desktop foi fortemente adequado para periféricos plug-in nos últimos 10 anos, porém atualmente pode ser monopolizado por um único dispositivo, como um drive Serial-ATA. Os cartões LAN Gigabit – a 1000 Mb/s – utilizam aproximadamente 95 % da banda disponível no barramento PCI. A arquitetura do barramento PCI necessita que os 132 MB/s sejam compartilhados entre todos os dispositivos no barramento, desta maneira, dispositivos de grande largura de banda como drives Serial ATA e placas LAN Gigabit interferem no desempenho dos outros dispositivos do barramento PCI. Devido a estas limitações, um novo barramento chamado PCI Express começou a aparecer recentemente nos novos PCs.

O PCI Express da National Instruments mantém a compatibilidade de software com o barramento PCI, mas substitui o barramento físico por um barramento serial de alta velocidade (2.5 Gb/s) Os dados são enviados em pacotes através de sinais de envio e recebimento chamados lanes com cerca de 200 MB/s de largura de banda por direção, por lane. Múltiplas lanes podem ser agrupadas em x1 (“by-one”), x2, x4 e x8. Diferente do PCI, que compartilha a largura de banda com todos os dispositivos conectados no barramento, esta banda é proporcionada para cada dispositivo no sistema.

Uma das principais vantagens do PCI Express para instrumentação virtual é os dispositivos plug-in como placas de aquisição de dados ou imagem que podem utilizar o aumento de banda para aquisições e transferências mais rápidas e múltiplas dispositivos no sistema se beneficiam da largura de banda garantida (figura 4.6).

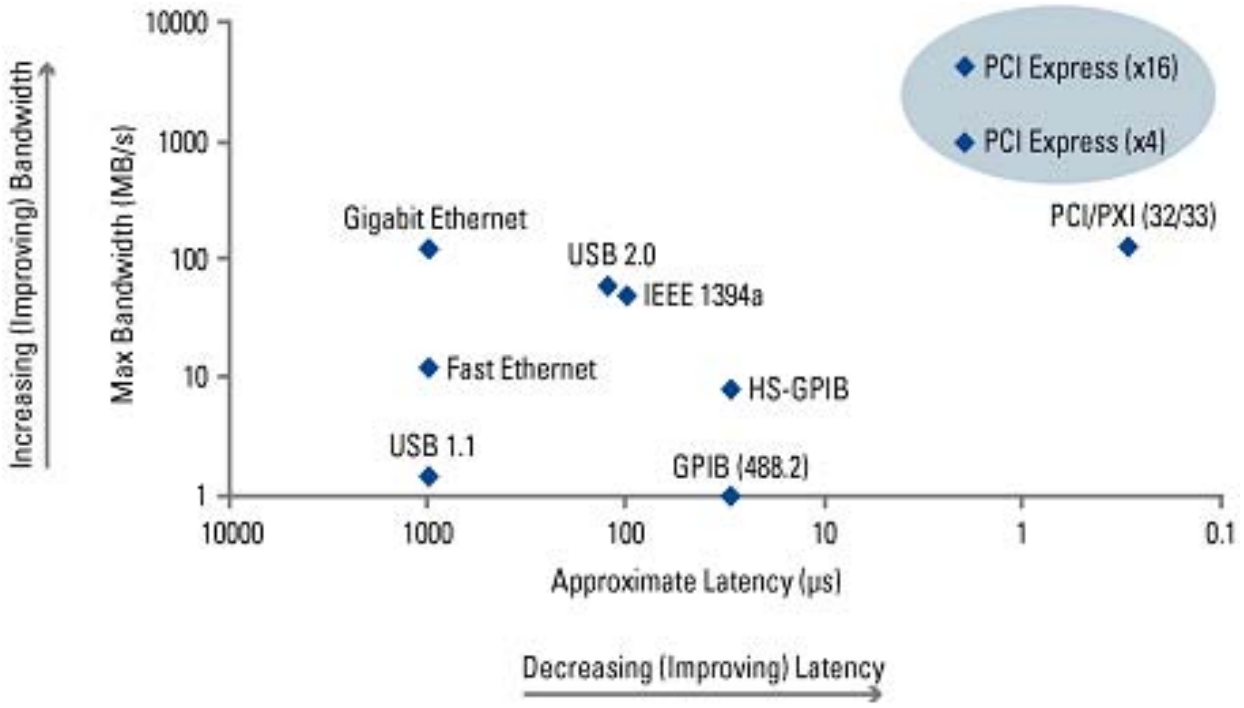


Figura 4.6: A evolução dos barramentos de PCs (National Instruments).

A interface USB 2.0, padronizada em todos os novos PCs oferece significantes benefícios à instrumentação virtual. Inicialmente criado para conectar periféricos como teclado e mouse do PC, o USB se tornou rapidamente o padrão para transmitir dados entre o PC e dispositivos eletrônicos, incluindo câmeras digitais, MP3 players e outros dispositivos de aquisição de dados.

A natureza plug-and-play do USB torna a usabilidade e a portabilidade do dispositivo extremamente simples. O PC automaticamente detecta quando um novo dispositivo foi conectado, requisita informações para identificação e configura apropriadamente os drivers requisitados.

Além disso, o USB possui “conexão a quente”, assim, independente de outros barramentos, não há necessidade de desligar o PC para conectar ou remover um dispositivo. A alta velocidade do USB 2.0 melhora a transferência de dados em 40X se comparado ao USB 1.1, aumentando a largura de banda para 480 Mb/s.

4.4.2 **Benefícios do barramento Ethernet para instrumentação Virtual**

Sistemas de instrumentação virtual freqüentemente utilizam Ethernet para controle e teste remoto do sistema, E/S distribuídas e compartilhamento de dados na rede empresarial. O benefício primário do barramento Ethernet é o baixo custo, em quase todos os casos, a rede Ethernet é precedida de um sistema de medição, este fato geralmente adiciona um pequeno custo ao sistema de medição. A Ethernet provê um baixo custo e um método moderado para transferência de dados e comandos de controle em longas distâncias. Entretanto, devido a sua arquitetura baseada em pacotes, a Ethernet não é determinística e possui uma latência relativamente grande. Para algumas aplicações, como sistemas de instrumentação, a falta de determinismo e a grande latência tornam a Ethernet uma má escolha para integração de módulos de E/S adjacentes. Estas situações são melhores resolvidas com um barramento dedicado como PXI, VXI ou GPIB.

Freqüentemente, um sistema de instrumentação virtual utiliza outros barramentos em conjunto com a Ethernet. Tipicamente, um nó na rede consiste em conjuntos de E/S modulares, cada conjunto utiliza um barramento de alta velocidade e baixa latência para trocar dados entre os diferentes módulos de E/S. Para comunicar com os outros nós, transferir dados para uma localização remota ou aceitar comandos de uma localização remota, os nós da rede utilizam uma rede Ethernet.

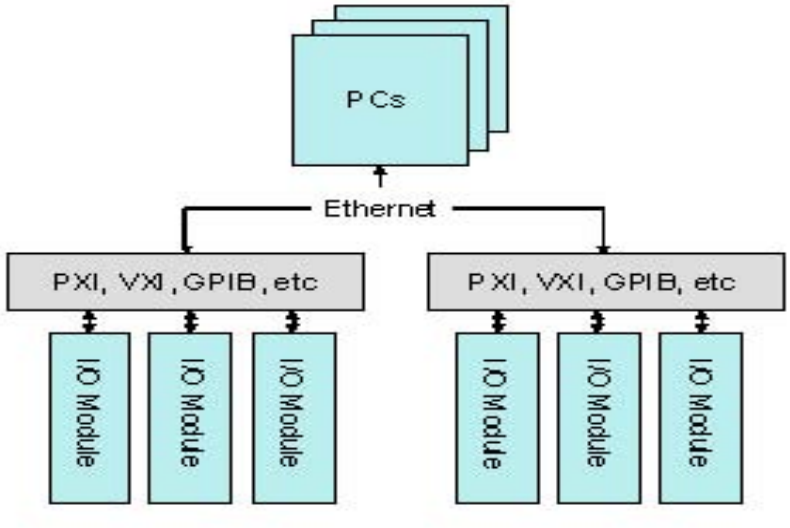


Figura 4.7: Exemplo de sistema de instrumentação virtual baseado em LAN/Ethernet

4.4.3 **Instrumentação Virtual para testes, controle e projeto**

4.4.3.1 **Testes**

À medida que as inovações tecnológicas aumentam juntamente com a pressão por produtos novos, diferenciados e lançados no mercado rapidamente, as expectativas do consumidor continuam aumentando; na indústria de eletrônicos, por exemplo, a aumento da integração de funções é cada vez mais necessário em um menor espaço com um custo também menor.

O campo de testes tem sido amplamente consolidado pela instrumentação virtual, onde se estima que mais de 30000 empresas (a maior parte sendo empresas de testes e medições) utilizam o conceito de instrumentação virtual.

Uma desaceleração da economia e restrições de recursos não representa um fator de limitação da necessidade de inovação, onde a relação de necessidades de ume empresa, de modo rápido, consistente e com maior confiabilidade é um fator para o sucesso de um negócio, representando uma maior vantagem em relação à competitividade.

Todas estas condições conduzem a novas necessidades de validação, verificação e testes de fabricação. Uma plataforma de testes que pode acompanhar estas inovações não é mais opcional, mas essencial. A plataforma deve incluir ferramentas para rápido desenvolvimento de sistemas de teste adaptáveis para serem utilizadas durante o fluxo de desenvolvimento de um produto. A necessidade de disponibilizar estes produtos no mercado rapidamente e produzi-los de maneira eficiente requer um sistema de teste de alta-velocidade. Para testar os complexos produtos multifuncionais que os consumidores demandam são necessárias potencialidades de medições precisas e sincronizadas e ao passo que as empresas incorporam estas inovações para diferenciar seus produtos, os sistemas de teste devem ser rapidamente adaptados a estas novas características.

4.4.3.2 **Controle e E/S industriais**

Tanto os PCs quanto os CLPs possuem um papel importante em aplicações de controle industrial. Os PCs trazem uma grande flexibilidade e potencialidade de software enquanto os CLPs proporcionam uma grande robustez e confiabilidade. Ao passo que as aplicações de controle se tornam mais complexas, é de comum acordo a necessidade de acelerar as potencialidades enquanto a robustez e a confiabilidade são mantidas.

Especialistas de diversos segmentos da indústria reconhecem a necessidade de ferramentas que podem atender a crescente necessidade por controles mais complexos, dinâmicos, adaptativos e baseados em algoritmo. Atendendo esses requisitos por parte da instrumentação virtual surgiram os PACs (Controladores Programáveis para Automação), que detalharemos a seguir.

4.4.3.3 **Projetos**

Uma das principais dificuldades para o desenvolvimento de um projeto é a necessidade de utilizarmos uma grande variedade de ferramentas de software.

Considerando a não existência de uma boa interface entre as fases de projeto e de validação e testes, muitas vezes a fase de projeto deverá ser finalizada para iniciar a fase de teste/validação, porém em função dos problemas observados na fase de teste pode-se necessitar de uma reiteração na fase de projeto (figura 4.8).

Figura 4.8: Testes e validação para avaliação do desempenho nas diferentes fases do projeto e na fabricação dos dispositivos eletrônicos atuais (NI instruments).

Sistemas com propriedades de integração intrínsecas são facilmente estendidos e adaptáveis para o aumento das funcionalidades do produto. Quando novos testes são requisitados, engenheiros simplesmente adicionam novos módulos à plataforma para realizar medições. A flexibilidade de software e a modularidade de hardware da instrumentação virtual a torna uma necessidade para acelerar o ciclo de desenvolvimento.

4.5 **Controladores Programáveis para Automação (PAC)**

PAC, uma sigla criada pela Automation Research Corporation (ARC), que significa Controlador Programável para Automação (do inglês Programmable Automation Controller) é utilizada para descrever uma nova geração de controladores industriais que combina a funcionalidade de um CLP (controladores lógicos programáveis) e de um PC.

Os PACs são dispositivos que possuem toda a flexibilidade e potencialidade dos PCs mantendo a robustez e a confiabilidade dos PLCs através de uma arquitetura voltada ao chão de fábrica gerenciada por um sistema operacional em tempo real. Esses equipamentos, combinam a funcionalidade de um PC e confiabilidade de um CLP, e estão cada vez mais sendo incorporados em sistemas de controle.

4.5.1 **Histórico**

Na última década, existia um forte debate sobre as vantagens e desvantagens dos CLPs comparados ao controle baseado em PC. À medida que as diferenças entre PC e CLP se estreitam, com os CLPs utilizando tecnologias padrão de mercado e sistemas baseados em PC incorporando sistemas operacionais de tempo real.

O PAC é uma nova classe de controladores, utilizado tanto por fornecedores tradicionais de CLPs para descrever seus sistemas mais completos como por empresas fornecedoras de sistemas de controle baseado em PC para descrever suas plataformas para controle industrial.

Durante as três décadas que seguiram sua introdução no mercado, os CLPs evoluíram para incorporar E/S analógicas, comunicação através de redes e novos padrões de programação tais como IEC 61131-3, mesmo com 80% das aplicações industriais utilizam E/S digitais, poucos pontos de E/S analógicas e técnicas simples de programação. Especialistas da ARC: Venture Development Corporation (VDC) e o centro online de treinamento em CLPs: PLC.net estimam que:

- 77% dos CLPs são utilizados em pequenas aplicações (menos de 128 E/S) · 72% das E/S de CLPs são digitais
- 80% dos desafios em aplicações com CLPs são solucionados com um conjunto de 20 instruções ladder.

Assim, como 80% das aplicações industriais são solucionadas com ferramentas tradicionais, há uma grande demanda por CLPs simples e de baixo custo, o que têm acelerado o surgimento de micro CLPs de baixo custo com E/S digitais que utilizam lógica ladder. A mesma descontinuidade se sucedeu na tecnologia de plataformas de controle, onde 80% das aplicações requerem sistemas simples de baixo custo e apenas 20% necessitam de sistemas tradicionais de controle, utilizados em aplicações que necessitam de taxas de repetição mais elevadas, algoritmos de controle avançados, melhores características analógicas e melhor integração com a rede corporativa.

Nos anos 80 e 90 os PCs para controle industrial disponibilizavam as características de software para desempenhar tarefas avançadas, ofereciam uma programação rica e ambiente para usuário e utilizavam componentes padrões de mercado possibilitando a engenheiros de controle tirar vantagem de tecnologias desenvolvidas para outras aplicações. Estas tecnologias incluem processadores de ponto flutuante, barramentos de E/S de alta velocidade tais como PCI e Ethernet, armazenamento de dados não volátil e ferramentas gráficas para desenvolvimento de software. O PC também disponibilizava flexibilidade sem igual, software altamente produtivo e hardware avançado e com baixo custo.

Embora muitos engenheiros utilizem computadores industriais especiais com hardware robusto e sistemas operacionais especiais, a maioria deles evitava os PCs para controle por causa de problemas com confiabilidade. Além disso, os dispositivos utilizados dentro de um PC para diferentes tarefas de automação tais como E/S, comunicações ou movimento podem ter diferentes ambientes de desenvolvimento.

Conseqüentemente, os PCs dificilmente alcançavam a funcionalidade de um CLP ou criavam um sistema que incluía um CLP para a porção do controle do código e um PC para as funcionalidades mais avançadas. Esta é a razão para que hoje muitas fábricas tenham em suas plantas, CLPs utilizados em conjunto com PCs para data logging, conexão com leitores de código de barras, inserção de informação em bases de dados e publicação na web. O grande problema com este tipo de configuração se dá ao fato de que estes são freqüentemente difíceis de construir, depurar e manter. O engenheiro de sistemas freqüentemente é deixado com a inviável tarefa de incorporar hardware e software de múltiplos fornecedores, o que impõe um desafio já que os equipamentos não são projetados para trabalhar em conjunto.

Entretanto, os PCs ainda não eram ideais para aplicações de controle. Embora muitos engenheiros usassem os mesmos ao incorporar funcionalidades avançadas tais como controle e simulação analógica, conectividade com base de dados, funcionalidade baseada em web e comunicação com dispositivos de terceiros, o CLP ainda cumpria o papel na área de controle. O maior problema com controle baseado em computador era o fato de que PCs padrão não eram projetados para ambientes robustos, exigindo três desafios principais:

1. **Estabilidade:** Geralmente, os sistemas operacionais de uso geral dos PCs não eram estáveis o suficiente para controle. Instalações controladas por PCs foram forçadas a lidar com paradas de sistema e re-inicialização não programada.
2. **Confiabilidade:** Com discos rígidos magnéticos e rotativos e componentes não projetados para o ambiente industrial tais como as fontes faziam os PCs mais suscetíveis a falhas.
3. **Ambiente de programação não familiar:** Operadores de planta necessitam ter a habilidade de modificar um sistema para manutenção ou corrigir problemas. Utilizando lógica ladder, eles podiam manualmente forçar um cilindro para um estado desejado e rapidamente corrigir o código afetado para então restabelecer o sistema. Entretanto, sistemas baseados em PC requerem que os operadores aprendam novos e mais avançadas ferramentas em automação (redes de petri e GRAFCET, por exemplo).

4.5.2 Características necessárias para um controlador melhor

Sem uma solução clara entre PC e CLP, os engenheiros com aplicações complexas trabalharam juntamente com fornecedores da área de controle para desenvolver novos produtos, que exigiam capacidades avançadas de software do PC com a confiabilidade de um CLP. Estes usuários ajudaram através de sua experiência o desenvolvimento de produtos para empresas de controle baseado em CLP e PC.

As funcionalidades de software requeriam não só uma programação avançada, mas também um aumento na capacidade de hardware das controladoras. Com a diminuição na demanda mundial de componentes para PC, muitos fornecedores de semicondutores começaram a reprojetar seus produtos para aplicações industriais. Fornecedores da área de controle hoje estão incorporando versões industriais de processadores em ponto flutuantes, DRAM, dispositivos de armazenamento de dados em estado sólido tais como CompactFlash e chipsets Ethernet rápidos em produtos para controle industrial. Isto possibilitou aos fornecedores desenvolver software melhor com a flexibilidade e com a facilidade de uso de sistemas de controle baseados em PC que podem rodar em sistemas operacionais de tempo real para confiabilidade.

As novas controladoras resultantes, projetadas para atender os “20%” das aplicações, combinam as melhores características do CLP com as melhores características do PC. Analistas da indústria da ARC chamaram estes dispositivos de controladores programáveis para automação (do inglês programmable automation controllers), ou PACs. Em seu estudo mundial chamado de “Programmable Logic Controllers Worldwide Outlook”, a ARC identificou cinco características principais dos PACs. Este critério caracteriza a funcionalidade do controlador definindo as funcionalidades do software:

1. **Funcionalidade múlti-domínio:** Com funções lógicas, controle de movimento, controle PID, drives e processos em uma única plataforma, e algumas funções dedicadas de software para atender a protocolos específicos como lógica, movimento, processo e PID. De forma geral, controle de movimento se trata de uma malha de controle em software que lê entradas digitais de um encoder de quadratura, executa loops de controle analógicos e gera um sinal analógico de saída para controlar um driver.

2. **Plataforma de desenvolvimento única e multi-disciplinar:** incorporando configuração simples de tags e uma única base de dados para acesso a todos os parâmetros e funções. Os PACs são projetados para aplicações mais avançadas tais como projetos multi-domínio, exigindo um software mais avançado, de modo que para tornar o projeto do sistema mais eficiente, o mesmo precisa ser um pacote único e integrado ao invés de ferramentas isoladas que não foram projetadas para se integrarem e trabalharem juntas.
3. **Ferramentas de software que possibilitam o projeto processando seu fluxo através de várias máquinas ou unidades de processo:** utilizando o padrão IEC61131-3 que trata dos padrões para o usuário e gerenciamento de dados, além de ferramentas de desenvolvimento gráficas de alto nível que tornam fácil traduzir um conceito do engenheiro sobre o processo em um código que controla a máquina.
4. **Arquiteturas abertas e modulares**: que espelham as aplicações industriais de layouts de máquinas em fábricas a unidades de operação em plantas de processos. Já que todas as aplicações industriais requerem razoável nível de personalização, o hardware precisa oferecer modularidade, conseqüentemente, o engenheiro pode escolher os componentes mais apropriados, assim, o software precisa possibilitar ao desenvolvedor a capacidade de adicionar e remover módulos para projetar o sistema necessário.

5. **Utilização de padrões para interfaces de rede, linguagens, etc., tais como TCPIP, OPC & XML e SQL**: Comunicação com redes corporativas é crítico para sistemas de controle modernos. Embora os PACs incluam uma porta de rede, o software para comunicação é a chave para integração sem problemas com o restante da planta.

4.6 **Abordagem Software**

O software é a diferença chave entre PACs e CLPs, e os fornecedores desses equipamentos têm diferenças ao abordar o tema de fornecer software avançado. Normalmente, são adicionados nos softwares de controle já existentes, aspectos referentes à funcionalidade, confiabilidade e facilidade de uso requerido para programar os PACs. Geralmente, isto cria dois campos de fornecedores de software para PAC: com base em controle em CLP e com base em controle através de PC.

4.6.1 **Softwares Baseado na filosofia do CLP**

Fornecedores tradicionais de software para CLP apresentam uma arquitetura de scan confiável e fácil de usar, adicionando novas funcionalidades à mesma. Conseqüentemente, o software para CLP segue um modelo geral de varredura de entradas, execução do código de controle, atualização das saídas e desempenho de tarefas de manutenção.

Por outro lado, um engenheiro de controle está preocupado apenas com o projeto do código de controle por causa dos ciclos de entrada, de saída e de manutenção que estão todos ocultos. Com muito do trabalho sendo realizada pelo fornecedor, esta arquitetura de controle restrita torna mais fácil e rápida a criação de sistemas de controle. A rigidez destes sistemas, um dos pontos fortes do CLP, elimina a necessidade do engenheiro de controle de entender completamente a operação em baixo nível do CLP para criar programas confiáveis, entretanto um grande inconveniente de sistema muito rígido é que pode torná-lo inflexível.

Fornecedores de CLP criam software para o PAC adicionando novas funcionalidades à arquitetura de varredura existente tais como comunicação Ethernet, controle de movimento e algoritmos avançados, mantendo ainda uma aparência familiar da programação do CLP pontos fortes de lógica e controle. O resultado é um software para o PAC geralmente projetado para suprir tipos específicos de aplicação tais como lógica, movimento e PID, mas sendo menos flexível para aplicações personalizáveis como comunicação, data logging ou algoritmos de controle personalizáveis.

4.6.2 **Softwares Baseado na filosofia do PC**

Fornecedores de software para PC iniciam com uma linguagem de programação de uso geral bastante flexível, que possibilita acesso em baixo nível ao hardware. Este software também incorpora confiabilidade, determinismo e arquiteturas de controle padrão. Embora engenheiros possam criar a estrutura de varredura normalmente disponibilizada ao programador de CLP, esta não é própria do software de controle baseado em PC. Isto torna o software do PC extremamente flexível e bem apropriado para aplicações complexas que requerem estruturas avançadas, técnicas de programação ou controle em baixo nível, mas ao mesmo tempo mais difíceis para aplicações simples.

O primeiro passo para estes fornecedores é disponibilizar confiabilidade e determinismo, que normalmente não estão disponíveis em um sistema operacional de uso geral tal como o Windows. Isto é alcançado através de sistemas operacionais de tempo real (RTOS) tal como o Phar Lap da Ardence (antiga Venturcom) ou o VxWorks da Wind River. Estes sistemas operacionais possibilitam o controle de todos os aspectos do sistema de controle, desde as taxas de leitura e escrita até a prioridade de tarefas individuais rodando na controladora.

Estes fornecedores então adicionam estruturas de leitura/escrita de E/S para tornar mais simples para os engenheiros construir aplicações de controle confiáveis. O resultado é um software flexível apropriado para controle personalizado, *data logging* e comunicação, mas esquecendo as arquiteturas de programação familiares aos CLPs, tornando o desenvolvimento da aplicação mais exigente.

4.6.3 **Alguns Softwares da National Instruments**

A National Instruments está estendendo a funcionalidade do PAC, para serem utilizados tanto para simples E/S, como também, incorporando aplicações industriais direcionadas a medições a taxas de aquisição mais elevadas, funcionalidades de visão de máquina, sistemas de medições em alta velocidade para aplicações de vibração e qualidade de energia. Os dados coletados são utilizados para monitorar a condição de máquinas rotativas, determinarem agendas de manutenção, identificar condição de motores e ajustar algoritmos de controle.

Esses dados são normalmente coletados utilizando sistemas de aquisição de dados especializados ou instrumentação independente e então incorporados em sistemas de controle utilizando barramentos de comunicação. Os PACs da National Instruments podem realizar medições com alta exatidão a taxas de milhões de amostras por segundo, que então são passadas diretamente os seus sistemas de controle para processamento imediato.

A National Instruments fabrica uma família de plataformas PAC que rodam no software LabVIEW, considerado padrão para teste e medição. Trata-se de um estilo de programação gráfico e intuitivo, similar aos fluxogramas, disponibilizando a funcionalidade de uma linguagem de programação completa com uma interface de fácil utilização.

Através dos Módulos Real-Time e FPGA, podemos combinar o LabVIEW com um sistema operacional de tempo real e a habilidade de acessar FPGAs (Field Programmable Gate Arrays) diretamente para obter confiabilidade e determinismo.

4.6.3.1 **Plataforma PAC**

Os PACs representem o que há de mais atual em controladores programáveis, o futuro para os PACs reside na incorporação de tecnologia embarcada. Um exemplo é a habilidade de utilizar software para definir hardware. Field Programmable Gate Arrays (FPGAs) são componentes eletrônicos utilizados de forma comum por fabricantes de eletrônicos para criar chips personalizados, tornando possível inserir inteligência em novos dispositivos.

Estes dispositivos consistem de blocos lógicos configuráveis que podem desempenhar uma série de funções, interconexões programáveis que agem como chaves para conectar os blocos de funções e blocos de E/S que passam dados para dentro e fora do componente. Definindo a funcionalidade dos blocos lógicos configuráveis e o modo como são conectados entre eles e as E/S, projetistas eletrônicos podem criar componentes personalizados sem o custo de produzir um componente eletrônico personalizado.Os FPGAs são comparáveis a ter um computador que literalmente reconecta seus circuitos internos para rodar sua aplicação específica.

Há algum tempo, a tecnologia FPGA eram disponíveis apenas aos projetistas de hardware que eram experientes em linguagens de programação de baixo nível como VHDL. Atualmente, essa ferramenta se tornou imprescindível aos engenheiros de controle que podem utilizar o LabVIEW FPGA para criar algoritmos de controle personalizados que são descarregados em componentes FPGA.

Esta capacidade torna os engenheiros capazes de incorporar funções extremamente críticas no tempo em hardware tais como detecção de sensores de limite e proximidade e monitoramento da saúde do sensor. Já que o código de controle roda diretamente no silício, é possível para engenheiros criar aplicações que incorporam protocolos de comunicação personalizados ou loops de controle de alta velocidade: loops de controle digital com até 1 MHz e loops de controle analógicos com até 200 KHz. Esse tópico será abordado detalhadamente posteriormente neste capítulo.

4.6.3.2 **Visão de Máquinas e Sistema de Medição em PACs**

A utilização de um sistema de Visão Computacional esta cada vez mais freqüênte na área de Mecatrônica, e muitos fabricantes vem desenvolvendo sistemas dedicados para visão de máquina e processamento de imagens para as mais diversificadas aplicações na área de engenharia e com ênfase especial em sistemas de controle. Em um ambiente de produção, existem muitos gargalos ou erros que podem ser identificados através de inspeção visual e que são difíceis de detectar utilizando técnicas tradicionais de medição.

Aplicações comuns incluem inspeção de partes para verificação na fabricação ou montagem tais como verificação da correta posição de componentes em placas de circuito impresso, reconhecimento de caracteres (OCR) para exame de códigos ou para selecionar produtos e medições óticas para encontrar falhas em produtos para seleção baseada em critérios de qualidade. Muitas plantas utilizam atualmente câmeras inteligentes que precisam se comunicar com o controle de processo da produção.

Nos dias atuais softwares industriais contêm um grande numero funções de visão, que poderão ser implementadas em outros softwares de Instrumentação Industrial, tais como o LabVIEW da National Instruments, que já possuem funções dedicadas para diferentes aplicações industriais tais como sistemas de inspeção, alinhamento, identificação e medição visual. Estes ambientes possuem interatividade para configurar, testar e distribuir aplicações de visão sem muita programação, trabalhando diretamente com muitas interfaces de aquisição de imagem disponíveis no mercado.

Considerando a rapidez e eficiência na sua manipulação e implementação, alem da inúmera quantidade de recursos disponíveis nestes sistemas, que cobrem grande maioria das aplicações na área de Mecatrônica estes sistemas de Visão Automatizada são cada vez mais utilizados nesta área. A escolha de um software de visão adequado para uma dada aplicação devera seguir as recomendações seguintes:

a) Escolha da Câmera,
b) Fator de Escala do Hardware,
c) Software de simples utilização,
d) Abrangência e exatidão do algoritmo implementado,
e) Desempenho do algoritmo implementado,
f) Facilidade de integração com outros dispositivos,
g) Custo-Beneficio envolvido,
h) Parceiros e Integradores,
i) Suporte Técnico,
j) Crescimento e Estabilidade da Empresa

Os PACs da National Instruments incorporam visão ou medições em alta velocidade com lógica e controle de movimento eliminando a necessidade de engenheiros de integrar plataformas de hardware e software são semelhantes. No anexo I desta dissertação são apresentados mais detalhes as principais características do software de visão da National Instruments.

4.6.4 **O Labview utilizado para controle**

Pelas potencialidades do LabVIEW e a facilidade de uso da linguagem de programação gráfica, os PACs baseados em LabVIEW são bastante apropriados para aplicações que requerem:

- **Gráficos:** O programador constrói automaticamente a interface de usuário, podendo ser incorporado facilmente em sistemas de controle: gráficos e Interfaces Homem-Máquina - IHM.
- **Medições:** (aquisição de dados em alta velocidade, visão e movimento). Implementação de E/S de alta velocidade, incluindo aquisição de imagem. Assim, pode-se incorporar sistemas de medições tais como vibração e visão de máquina.
- **Características de processamento:** Possibilidade de implementarmos algoritmos de controle especializados, processamento de sinal avançado ou data logging, podendo incorporar código de controle personalizado ou ferramentas desenvolvidas por terceiros, e implementar processamento de sinal como JTFA ou gravar dados de forma local ou remota.
- **Plataformas:** Criação de código que roda em uma variedade de plataformas incluindo um PC, controlador embarcado, chip FPGA ou dispositivo PDA.
- **Comunicação:** Possibilidade de transmitir dados à rede corporativa com ferramentas como conectividade a base de dados, OPC e interfaces de operador via navegador web.

4.6.5 **PACs desenvolvidos pela National Instruments**

A National Instruments oferece cinco plataformas PAC baseadas em LabVIEW: PXI, CompactPCI, CompactFieldPoint, CompactVisionSistem, CompactRIO.

4.6.5.1 **PXI e CompactPCI**

O PXI é um PAC padrão da indústria baseado na arquitetura CompactPCI que oferece um sistema industrial modular, compacto e robusto. Um sistema PXI possui um controlador embarcado com um processador com alto desempenho com clock na faixa GHz.

Ele pode funcionar com módulos da National Instruments ou de outros fornecedores de produtos PXI ou CompactPCI. O PXI oferece a mais ampla gama de E/S incluindo entrada analógica isolada para a faixa de 1000 V, E/S digital de alta densidade, placas para captura de imagens analógicas e digitais para visão de máquina e controle de movimento para múltiplos eixos. A plataforma PXI oferece uma grande quantidade de módulos de medição e conectividade a dispositivos de campo utilizando CAN, DeviceNET, RS-232, RS-485, Modbus e Foundation Fieldbus.

4.6.5.2 **Compact FieldPoint**

O Compact FieldPoint consiste de módulos de E/S analógicas e digitais que são *hot-swappable* e controladoras com interfaces Ethernet e serial. Os módulos de E/S oferecem conectividade direta com termopares, RTDs, *strain gauges*, sensores de 4-20mA e sinais de 5-30 VDC e 0-250 VAC.

As interfaces de comunicação de rede do Compact FieldPoint publicam automaticamente medições através da rede Ethernet. Assim, podemos acessar pontos de E/S próximos ou a quilômetros de distância da rede utilizando a mesma ferramenta simples de software para leitura/escrita.

Através de simples interface de software, este sistema é simples de configurar e programar, oferecendo potencial suficiente para desempenhar controle complexo, data logging e comunicação.

4.6.5.3 **Compact Vision System**

O Compact Vision System combina um processador Intel de alto desempenho com um FPGA, E/S digitais e três portas 1394. Este PAC é projetado para incorporar visão em aplicações de controle através do uso da tecnologia FireWire (IEEE 1394), compatível com mais de 80 câmeras industriais.

Através de um FPGA reconfigurável e linhas digitais de E/S no CVS, podemos ter uma plataforma com alguns sinais digitais ou até mesmo um controlador para motor de passo. Quando programado com LabVIEW, o sistema pode ser configurado tanto para visão de máquina com alto desempenho e controle digital de alta velocidade como para controle de motor de passo.

4.6.5.4 **CompactRIO**

O CompactRIO é um sistema de controle e aquisição de dados baseado em FPGA reconfigurável para aplicações que requerem um alto nível de personalização e controle de alta velocidade. Sua arquitetura combina um processador de tempo real embarcado para algoritmos complexos e cálculos personalizados com uma plataforma FPGA com E/S reconfiguráveis (RIO – reconfigurable I/O). Esta plataforma acomoda até oito módulos de E/S analógicas ou digitais fabricados pela National Instruments ou outras empresas.

Esta plataforma é ideal para aplicações complexas e de alta velocidade tais como controle de máquinas e tendo um FPGA, sendo uma boa opção para aplicações que normalmente requerem desenvolvimento de hardware personalizado.

PCs industriais padrão podem também ser utilizados com uma grande variedade de módulos PCI produzidos pela National Instruments. Estas interfaces incluem hardware projetado para E/S analógicas e digitais, controle de movimento e visão de máquina. Para obter desempenho determinístico e de tempo real, basta combinar hardware PCI com o LabVIEW Real-Time sendo executado em um sistema operacional de tempo real num PC, que pode ser carregado em grande parte dos PCs industriais padrão para oferecer uma plataforma para medição e controle industriais com baixo custo.

4.7 **Controle e Simulação baseada Hardware Reconfigurável**

A utilização da tecnologia de hardware reconfigurável (FPGAs) para Prototipagem Rápida de Controle e Simulação Hardware in the Loop, esta cada vez mais freqüente no campo da Mecatrônica, evitando assim, a construção de muitos protótipos durante as etapas de desenvolvimento de um produto, implicando em significativa redução de custo e tempo de desenvolvimento de um produto.

Neste tópico apresentaremos etapas para o desenvolvimento de sistemas de controle baseados em modelo podem ser acelerado utilizando técnicas de prototipagem rápida de controle e testes hardware in the loop (HIL) ferramentas de hardware reconfigurável em conjunto com uma linguagem de programação aberta.

4.7.1 **Ciclo de Desenvolvimento de um Produto**

O "Diagrama V" é freqüentemente utilizado para descrever o ciclo de desenvolvimento de aplicações de controle embarcado, comuns nas indústrias automotivas, aeroespaciais e de defesa. Originalmente desenvolvidas para encapsular o processo de projeto de diferentes aplicações de software, diversas versões deste diagrama podem ser encontradas para descrever uma variedade de ciclos de projetos de produtos.

O objetivo de um desenvolvimento rápido é tornar o ciclo o mais eficiente possível minimizando as iterações necessárias na fase de projeto. Se o eixo X do diagrama é representado pelo tempo, o objetivo é tornar o "V" o menor possível, mantendo as duas pontas do diagrama o mais próximo possível, assim reduzindo o tempo de desenvolvimento.

Através da Prototipagem Rápida de Controle (RCP) e simulações Hardware in the Loop (HIL) é possível otimizar o ciclo de desenvolvimento, conforme esta representada na fig. 4.9, através de uma visão geral dos estágios de modelagem e projeto do diagrama em V.

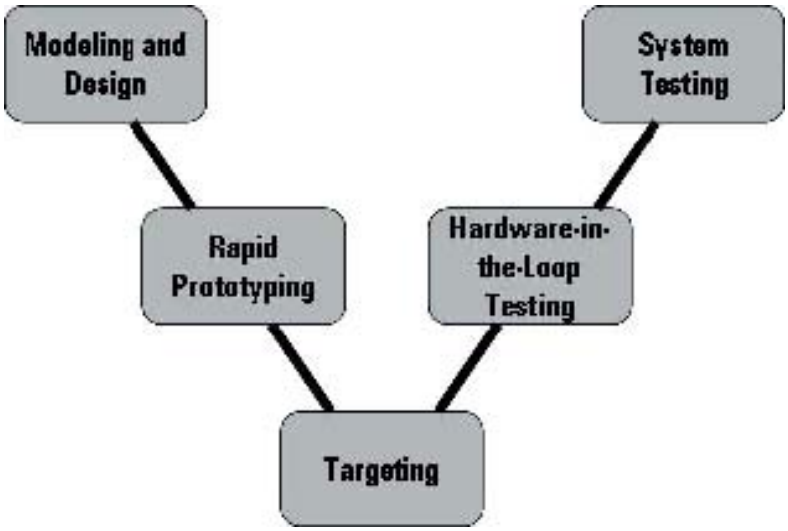


Figura 4.9: Ciclo de Desenvolvimento de um Produto Embarcado (National Instruments).

4.7.2 **Projeto e Modelagem**

Um fator imprescindível na implementação um ciclo "V' através de um projeto baseado no modelo é o desenvolvimento do controle embarcado o mais cedo possível no ciclo de projeto. A modelagem proporciona a habilidade de iniciar a simulação do comportamento do controle enquanto os protótipos de hardware ainda estão indisponíveis. Além disso, modelos de projetos anteriores podem ser reutilizados para reduzir ainda mais o esforço necessário na criação de um modelo.

Caso o protótipo ou hardware existente está disponível, a modelagem pode ser complementada utilizando dados reais de entrada/saída para produzir modelos com técnicas e ferramentas para identificação e modelagem de sistemas.

Diversos softwares disponíveis no mercado, tais como Matlab-Simulink, MATRIXx e outros, oferecem uma variedade de ferramentas para projeto de controles avançados em um ambiente interativo que auxilia o projetista a rapidamente avalia um sistema de controle sem a necessidade de prototipar um hardware, proporcionando assim, ao engenheiro de projeto a habilidade de determinar especificações, requisitos e modelar erros imediatamente ao invés de aguardar que testes sejam feitos futuramente no fluxo de projeto.

O projeto e modelagem de sistemas de controle incluem ferramentas gráficas para projetar e analisar estes sistemas. Através das mesmas é possível integrar simulações dinâmicas, e assim modelar plantas ou sistemas de controle linear, não-linear, discreto e contínuo a partir da simulação, depois executar a simulação em um sistema operacional convencional como o Windows ou em um hardware executando um sistema operacional de tempo real.

Plataformas abertas como o LabVIEW da National Instruments podem integrar modelos e códigos que foram criados com outras ferramentas de software, incluindo modelos desenvolvidos no Simulink da The MathWorks, Inc., MSC CarSim assim como aplicativos nas linguagens C, Fortran, etc. Devido à arquitetura aberta existentes em algumas ferramentas de software e hardware, é possível manipular modelos a partir de praticamente qualquer ferramenta de software.

4.7.3 **Software de tempo real**

Após realizar a simulação via software, é possível partir para uma implementação em tempo real do modelo. Dentre as diversas ferramentas software e hardware de tempo real considerada ideal para execução determinística de um modelo e realização de interface com uma grande quantidade de E/S, podemos destacar o módulo LabVIEW RealTime que expande o LabVIEW para plataformas de hardware de tempo real disponíveis no mercado, podendo também ser executado com o desempenho de um hardware de tempo real com ciclos em centenas de microssegundos e um baixo jitter.

Outra ferramenta para execução de aplicações em tempo real é o módulo LabVIEW FPGA, você pode criar dispositivos HIL personalizados que geram rapidamente alterações de sinais como sinais dos eixos, comando de válvulas e transdutores de deslocamentos como LVDTs. Utilizando FPGAs, você também pode adquirir sinais e desempenhar processamento de sinais on-board para modular sinais PWM, decodificar protocolos digitais personalizados e analisar sinais analógicos.

4.7.4 **Hardware em tempo real**

Podem-se utilizar diversas plataformas para aplicações HIL e RCP dependendo dos requisitos e E/S e processamento da aplicação. A plataforma PXI (pxi.org), um sistema de alto desempenho projetado para testes e medições, é ideal para aplicações HIL. Um PC desktop padrão também pode ser utilizado como uma plataforma de tempo real. Esta solução pode ser utilizada em laboratórios de pesquisa para controladores que utilizam um baixo número de canais de E/S. Uma terceira opção é o CompactRIO é uma plataforma versátil baseada em FPGA que é ideal para sistemas de prototipagem de controles.

A figura 4.10 mostra alguns tipos de diferentes plataformas capazes de executar tarefas em tempo real.

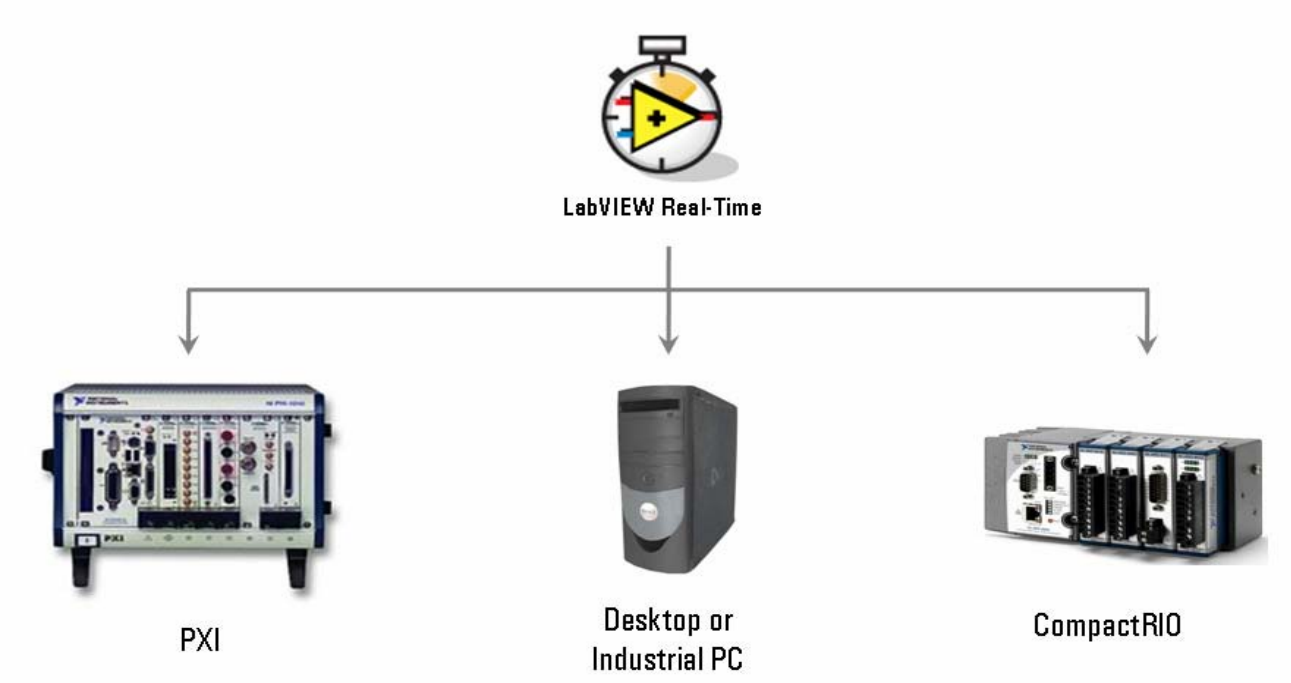


Figura 4.10: Hardware em Tempo real (National Instruments).

4.7.5 **Prototipagem Rápida de um Sistema de Controle utilizando FPGA**

Devido a simulações de software não poderem contar com todos os comportamentos peculiares de um ambiente dinâmico, um hardware de protótipo é desenvolvido para auxiliar no teste do algoritmo de controle. Esta prototipagem rápida do sistema de controle é o segundo estágio do projeto de um controlador de acordo com o diagrama V.

Diversas plataformas de tempo real poderão ser utilizadas na implementação de um protótipo, entretanto, um sistema baseado em FPGA como o CompactRIO é ideal devido a sua blindagem, robustez e flexibilidade. Este sistema embarcado baseado em FPGA inclui um processador de tempo real que pode executar algoritmos de controle de maneira determinística, desempenhar armazenamento de dados, atuar como servidor de páginas Web, etc.

A figura 4.11 apresenta um exemplo de uso de uma plataforma de controle em prototipagem rápida.

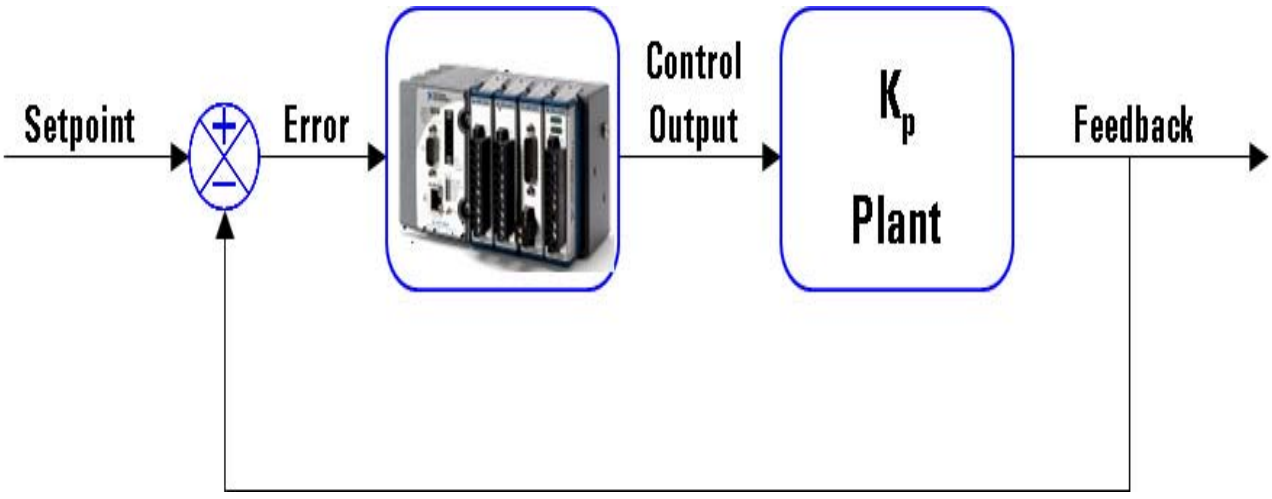


Figura 4.11: Prototipagem Rápida do Sistema de Controle baseada em FPGA.

Uma plataforma baseada em FPGA também proporciona a flexibilidade e desempenho para aquisição e geração de sinais em alta velocidade. Com esta plataforma é possível interagir sinais reais como analógicos, digitais, CAN, PWM e outros, utilizando módulos disponíveis no mercado.

4.7.6 **Implementação em Tempo Real de Simuladores desenvolvidos em Matlab-Simulink**

Utilizando o RealTime Workshop e o compilador Microsoft Visual Studio, um modelo desenvolvido no ambiente Simulink pode ser utilizado para gerar automaticamente uma DLL que pode ser chamada em outra plataforma de software como o LabVIEW, permitindo a integração destas plataformas, e configuração das E/S reais que serão as entradas e saídas do modelo de simulação. Pode-se também configurar os dispositivos de aquisição de dados, dispositivos FPGA e dispositivos CAN através desta interface, e a seguir, o LabVIEW deverá gerar o código que manipula o modelo com o código de aquisição e geração de E/S que interage com a planta.

Quando executado o programa em LabVIEW, o código automaticamente faz o download para o hardware de tempo real para uma execução determinística do modelo. A interface do operador é atualizada através de uma conexão Ethernet entre o a plataforma de tempo real e o host, conforme mostra a figura 4.12.

4.7.7 **Plataforma de Hardware**

O controlador é implementado na plataforma de hardware que pode ser uma unidade de controle eletrônico (ECU), um controlador e chassis disponível no mercado como o PXI e o CompactRIO, ou um hardware personalizado.

Pode-se manipular o código na plataforma baseada em software e no modelo do controlador, utilizar ferramentas de geração automática de código como o LabVIEW Embedded ou utilizar uma combinação de ambos para chegar à implementação do hardware. A figura 4.12 apresenta um exemplo de implementação de um controlador.

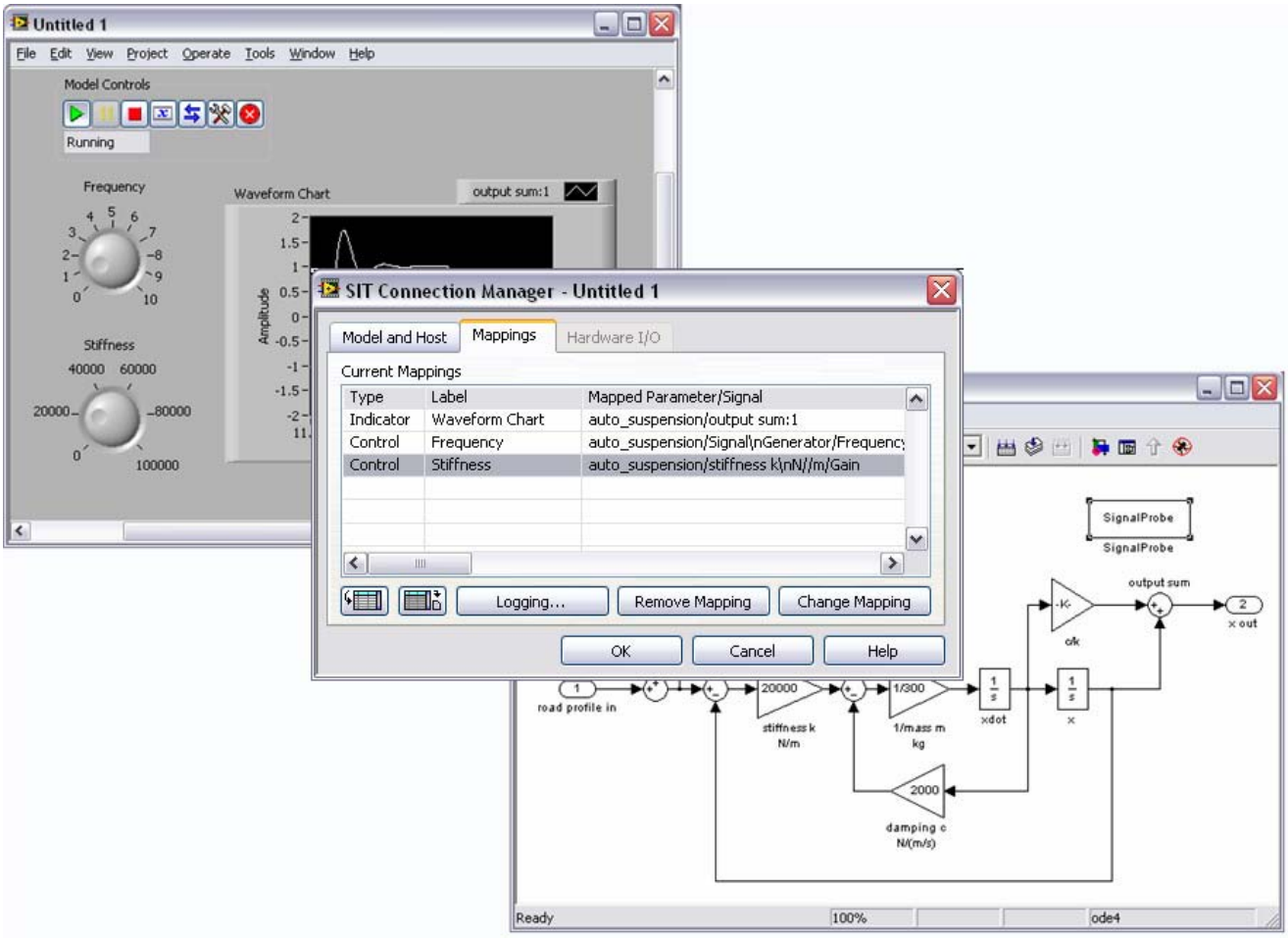


Figura 4.12: Implementação utilizando Plataforma Matlab-Simulink.

4.7.8 **Teste HIL com PXI**

Durante a fase HIL, o projetista pode simular o comportamento em tempo real e características do sistema final para verificar o sistema de produção do controlador sem a necessidade de possuir o hardware fisicamente.

A figura 4.13 mostra a implementação de modelos de controle em uma plataforma de hardware. A figura 4.14 mostra o algoritmo de controle sendo executado na plataforma de hardware do controlador enquanto a planta é simulada em tempo real no computador de teste.

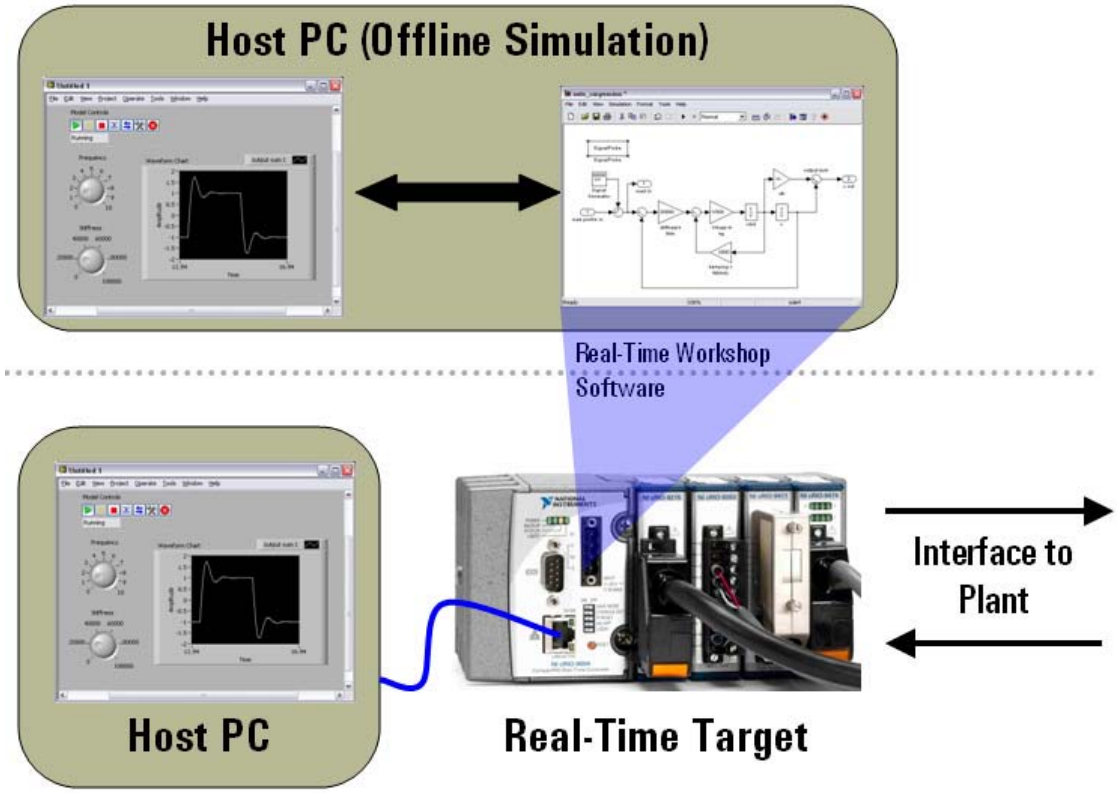


Figura 4.13: Implementação dos Modelos de Controle implementados em ambiente Simulink.

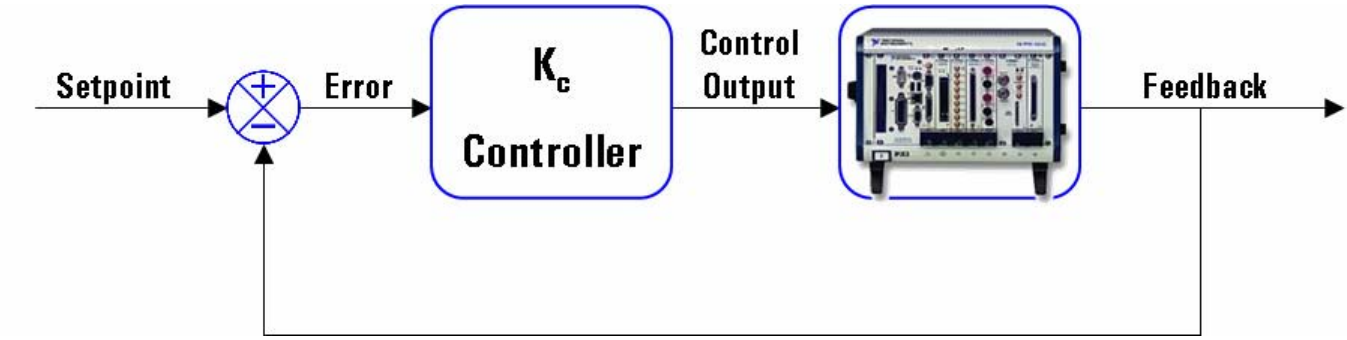


Figura 4.14: Teste HIL com PXI.

Durante esta fase, é importante testar a funcionalidade completa do controlador, que oferece diversas vantagens, tais como a conexão do hardware com a planta real e realização de testes e obtenção de resultados a partir da planta simulada. A fase HIL possui um custo/benefício maior, não necessitando da reprodução do modelo físico da planta. Nesta fase, uma variedade de condições de operação da planta poderá ser simulada, até mesmo condições de falha, como travamento de um determinado acionador, que seria difícil, de alto custo e perigoso reproduzir com uma planta real.

O PXI da National Instruments, baseado padrão CompactPCI é uma plataforma ideal para sistemas HIL, e consiste de um rack com um controlador, e módulos de E/S escolhidos de acordo com os requisitos do sistema. O controlador inclui um processador, disco para armazenamento de dados, memória, etc. Quando utilizado como um sistema de tempo real, a aplicação em LabVIEW é enviada para o processador embarcado no controlador. Quando a aplicação é executada, esta acessa dados dos módulos de E/S no sistema. O PXI oferece alto desempenho e a habilidade de ser escalado para centenas ou até milhares de canais. A figura 4.15 mostra a arquitetura de uma simulação HIL.

Existe uma grande quantidade de E/S disponíveis nos sistemas PXI incluindo analógicas e digitais, CAN, PWM, sinais dinâmicos, controle de movimento, aquisição de imagem e módulos de terceiros. Assim como na plataforma CompactRIO, você pode utilizar módulos FPGA e programá-los com LabVIEW para personalização do hardware de controle. Os sistemas PXI também são bem integrados com barramentos padrões necessários para sistemas HIL como CAN, Ethernet, MILSTD 1553 e ARINC 429. Devido o PXI ser uma arquitetura aberta, módulos PXI ou cPCI personalizados ou de terceiros podem ser facilmente integrados em um sistema PXI.

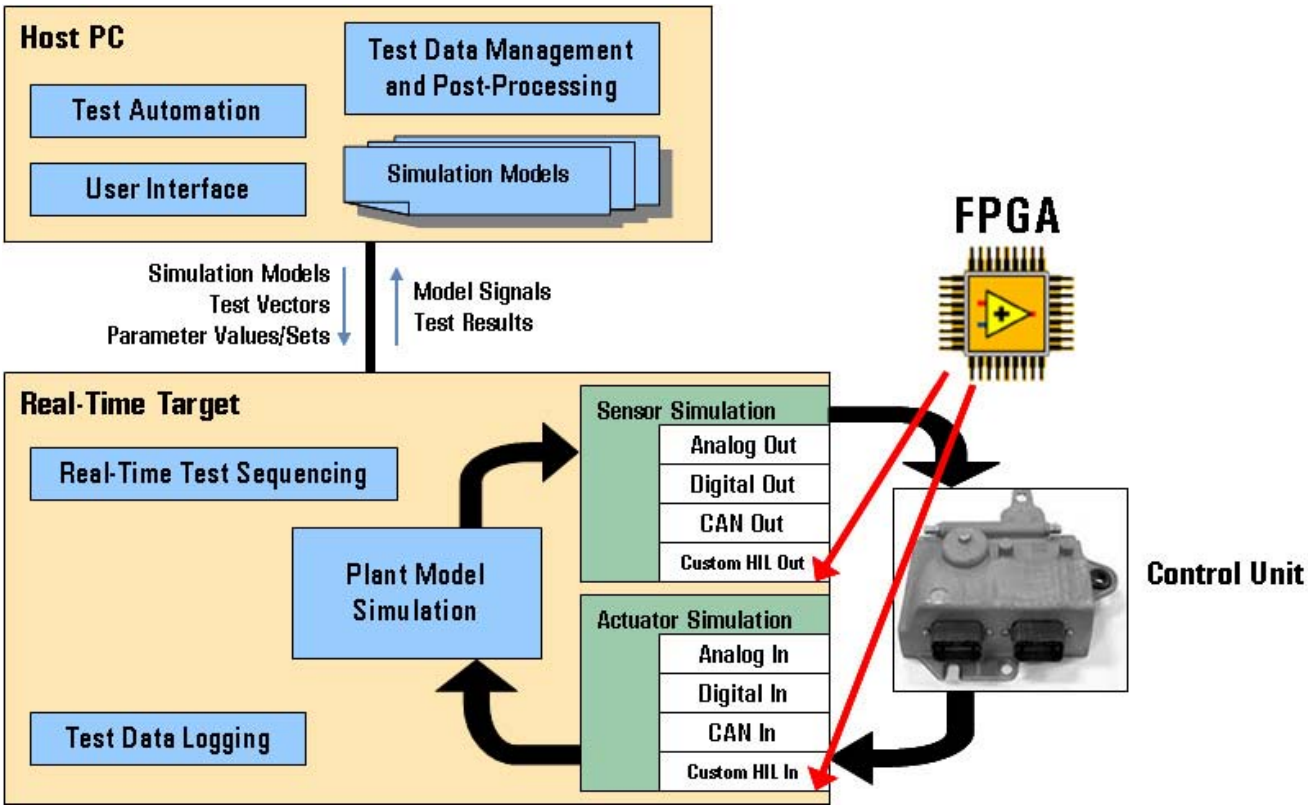


Figura 4.15: Arquitetura de Simulação HIL.

4.7.9 **Arquitetura da Simulação HIL**

Um sistema HIL executa em tempo real o modelo da planta, e pode conter diversos componentes, sendo a parte principal a simulação do modelo da planta, incluindo suas características dinâmicas. Os módulos de E/S são usados para receber as saídas do controlador e responder com sinais simulados a partir da planta.

Com uma plataforma baseada em FPGA você é possível criar E/S personalizadas para atender as necessidades particulares da simulação. Outros componentes de um sistema HIL podem incluir testes de gravação de dados assim como execução de testes em seqüências pré-definidas. Para completar o sistema, é utilizado um PC host (fig. 4.16) com uma interface de operador, uma ferramenta para gerenciamento completo do sistema de teste e ferramentas para análises posteriores disponíveis no mercado. Esta plataforma é aberta para implementação de arquiteturas de controle aberta, com hardware modular.

Figura 4.16: Exemplo de PC host.

4.8 **Conclusão**

Neste capítulo, foram apresentados conceitos básicos de Instrumentação Virtual, tendo como principal referência a plataforma LABVIEW^{TM} da National Instruments, e suas diversas áreas de aplicação, apresentando também os principais benefícios proporcionados por esta tecnologia. No próximo capitulo serão apresentados exemplos práticos utilizados para validar os principais conceitos apresentados neste capitulo, permitindo ao leitor uma avaliação bem completa dessas ferramentas direcionadas a prototipagem rápida de Sistemas Mecatrônicos.

Capítulo 5

Validação de resultados a partir da implementação de exemplos industriais

5.1 **Introdução**

Este capítulo apresenta alguns exemplos práticos implementados no Laboratório de Automação Integrada e Robótica da UNICAMP utilizando-se a ferramenta LabVIEW^{TM} . Estes exemplos utilizam conceitos apresentados nos capítulos anteriores dessa dissertação, sendo adicionados blocos de hardware e software dentro do conceito do sistema aberto proposto neste trabalho.

A escolha desses sistemas práticos para validação desses conceitos pode ser justificada por fatores como facilidade de implementação, capacidade de agregar sensores e atuadores em diferentes aplicações, sendo apresentadas detalhadamente as diferentes etapas de implementação de seis diferentes plataformas baseadas em aplicações industriais, demonstrando pelos resultados obtidos que as ferramentas de prototipagem rápida utilizada se mostram eficientes e de fácil implementação. Assim, estes conceitos podem ser sintetizados sob a forma de cinco aplicações experimentais apresentadas a seguir:

a) Controle de uma Plataforma Robótica Paralela (Stewart-Gough) através do LABVIEW com Vários Graus de Liberdade capaz de Reproduzir o Espectro de Movimentos do Mar;
b) Sistema Automatizado para Posicionamento de Robôs em Células de Soldagem de Carrocerias de Automóveis baseado em Plataforma de Visão da National Instruments
c) Robô Cartesiano de Posicionamento baseado na solução Control Motion da National Instruments
d) Integração de robôs industriais para operações cooperativas baseada no conceito de Prototipagem Rápida e integração baseada em Labview
e) Plataforma Robótica Web baseada em integração CLP – sistema supervisório baseado em Labview

5.2 **Plataforma Robótica Paralela controlada através de LABVIEW**

Neste projeto foi implementado no LAR-UNICAMP, uma plataforma robótica seis graus de liberdade para simulação dos movimentos da superfície do mar controlada através do software LABVIEW, a partir de arquivo de dados que descreve o comportamento de formas das ondulações marítimas expressos em séries temporais transformadas em parâmetros de posição e orientação.

A estrutura física da plataforma é conhecida e amplamente aplicada em uma grande variedade de problemas industriais. Trata-se da plataforma de Stewart-Gough, uma estrutura de manipulador paralelo, inicialmente desenvolvida para simuladores de vôo e que é encontrada em muitas aplicações atuais como teste de veículos terrestres, simulação de vôo em naves espaciais, exposições e mecanismos de entretenimentos.

5.2.1 **Introdução**

A plataforma de Stewart-Gough projetada e construída pode atingir um elevado padrão de precisão de posicionamento e grande capacidade de carga, uma característica dos manipuladores paralelos, como discutiremos a seguir. A estrutura de controle implementada dará ênfase ao controle de posição dos atuadores, não sendo abordada a análise dinâmica da plataforma como o realizado em outros desenvolvimentos (Luc Baron and Jorge Angeles -2000 e Cong-Xin Li Min-Jie Liu and Chong-Ni Li – 1992).

Durante o desenvolvimento dessa aplicação foram implementados procedimentos geométricos para o tratamento posicionamento da plataforma, designada de base uma superfície plana hipotética ou real presa à peça superior, a partir dos dados fornecidos sobre a movimentação do mar, i.e., o problema cinemático inverso. Incluiremos também deduções dos procedimentos para o problema cinemático direto, ou seja, dada certa postura, quais são seus parâmetros de rotação e translação.

Após o desenvolvimento analítico do problema em estudo, a validação e testes destes algoritmos foram realizados a partir de simulações em ambiente MATLAB^{TM}, convertendo

nossos algoritmos posteriormente para o ambiente de programação visual LabVIEW^{TM} e, finalmente, para implementação do Hardware de supervisão e controle do protótipo implementado.

A partir da montagem de um protótipo que simule a movimentação da superfície do mar, o modelo da plataforma escolhido permite que este protótipo de manipulador paralelo possa ser utilizado para simulação de qualquer condição de movimentação, devido à sua liberdade e às características de precisão no posicionamento viabilizadas através da construção mecânica.

5.2.2 **Descrição da Plataforma de Posicionamento Implementada**

Esta plataforma baseada num protótipo comercial possui uma base superior a uma base inferior ligadas por seis cilindros que habilitam os graus de liberdade, e a mesma deverá ser capaz testar e validar em computador o modelo teórico desenvolvido e algoritmos de controle de posição, conforme mostra a figura 5.1. Ela é constituída de uma chapa de metal hexagonal sustentada por seis atuadores constituídos de cilindros hidráulicos fixos a uma base também hexagonal.

a) Plataforma Comercial b) Plataforma Implementada

Figura 5.1 Plataforma de Posicionamento**.**

A geometria escolhida da plataforma distribui os esforços nos cilindros, além de possuir as outras características dos manipuladores paralelos deste tipo como a de ter seis graus de liberdade, associados à rotação e translação, necessários para a representação completa do movimento que queremos simular.

A construção mecânica da plataforma é relativamente simples, duas bases de aço de formato hexagonal são ligadas uma a outra por seis atuadores lineares, sendo a base superior de menor tamanho. Os atuadores são fixados as bases através de mancais e juntas rotuladas. Essa geometria permite seis graus de liberdade para o sistema. Na figura 5.2 são apresentadas algumas fotos dos elementos antes da montagem, e na figura 5.3 é apresentado um detalhe desses elementos na montagem final da plataforma implementada.

a) Base Superior

b) Base Inferior

c) Conjunto Mancal – rótula

d) Conjunto Montado

Figura 5.2 Elementos Mecânicos da Plataforma de Posicionamento.

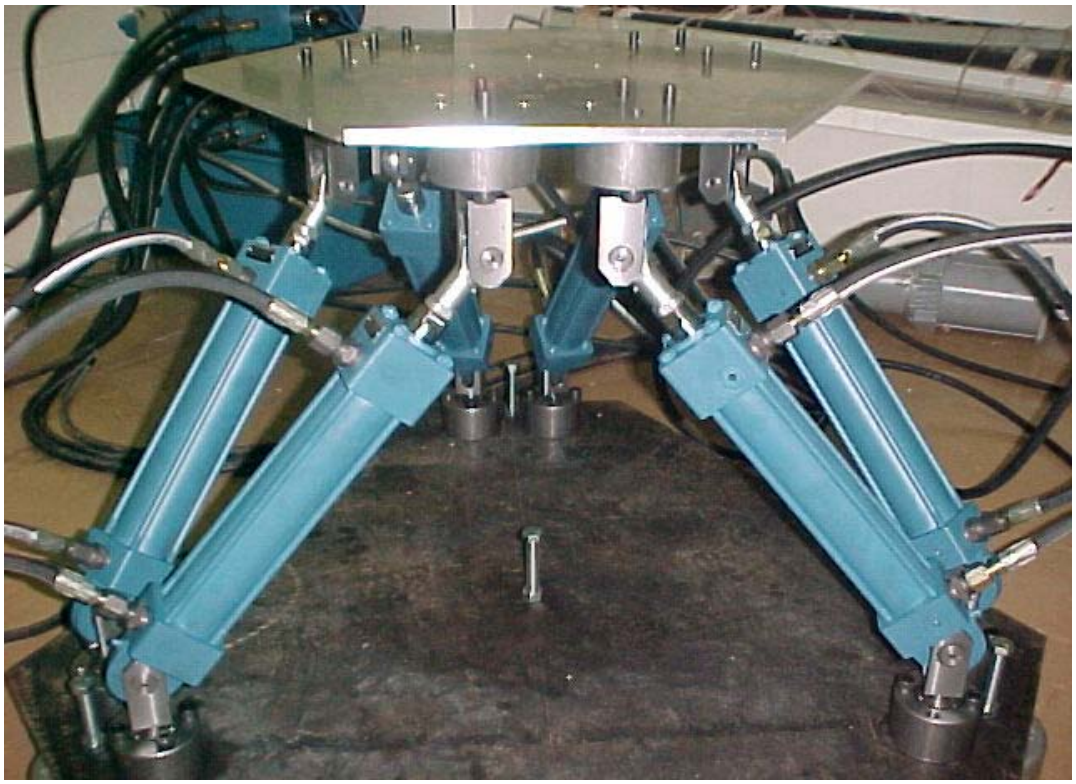


Figura 5.3 Plataforma de Posicionamento implementada no LAR-UNICAMP.

5.2.3 **Implementação do Controlador de Posição – Nível Junta**

5.2.3.1 **Controle de Movimentos da Plataforma**

O controle dos movimentos de cada atuador da Plataforma Robótica constitui um problema complexo, pois, o movimento da estrutura mecânica se realiza através da composição de movimentos de cada juntas linear que devem ser controladas simultaneamente, com possível acoplamento dinâmico.

Por outro lado, o comportamento da estrutura articulada é fortemente não linear e dependente das condições operativas. Estas condições devem ser levadas em conta na estratégia de controle escolhida. O posicionamento desejado é definido pela posição, velocidade, aceleração e orientação de qualquer ponto da plataforma.

5.2.3.2 **Modelo de uma Junta Robótica**

Nesta aplicação é suficiente que o controle da Plataforma de Posicionamento considere apenas o modelo cinemático, considerando o aspecto das juntas não serem acopladas, e a malha de controle de cada junta ser independente, não sendo considerada a influência da dinâmica dos outros graus de liberdade sobre uma junta específica (figura 5.4).

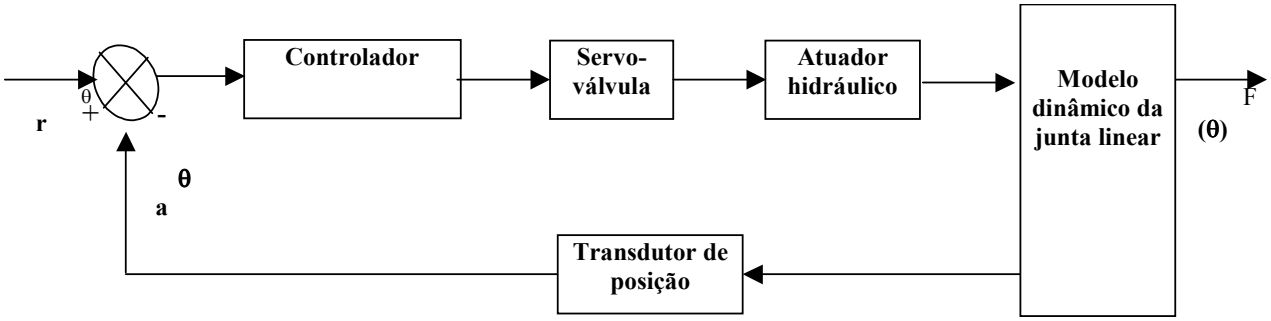


Figura 5.4: Diagrama de blocos de uma junta linear.

Os sinais de referência θ_{ri} podem ser gerados através de um interpolador de trajetórias gerado em LABVIEW^{TM} (Coutinho, L, A F-1993). A partir da comparação destes sinais de referência com as posições angulares provenientes dos transdutores de posição lineares (cilindros hidráulicos) de cada junta. Estes sensores serão constituídos de um dispositivo com potenciômetro de precisão acoplado a sistema cabo-polia, conforme mostra a figura 5.5.

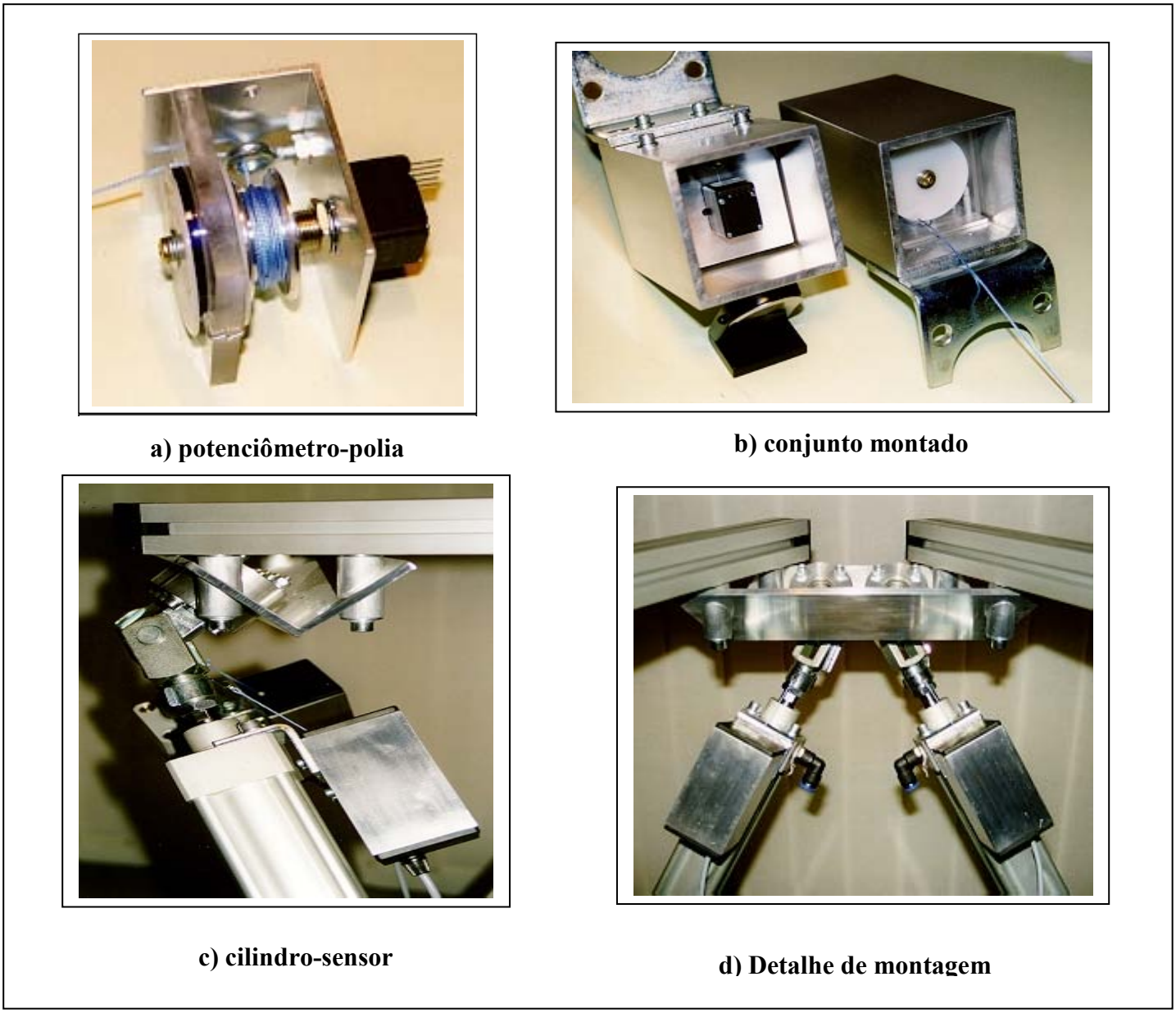
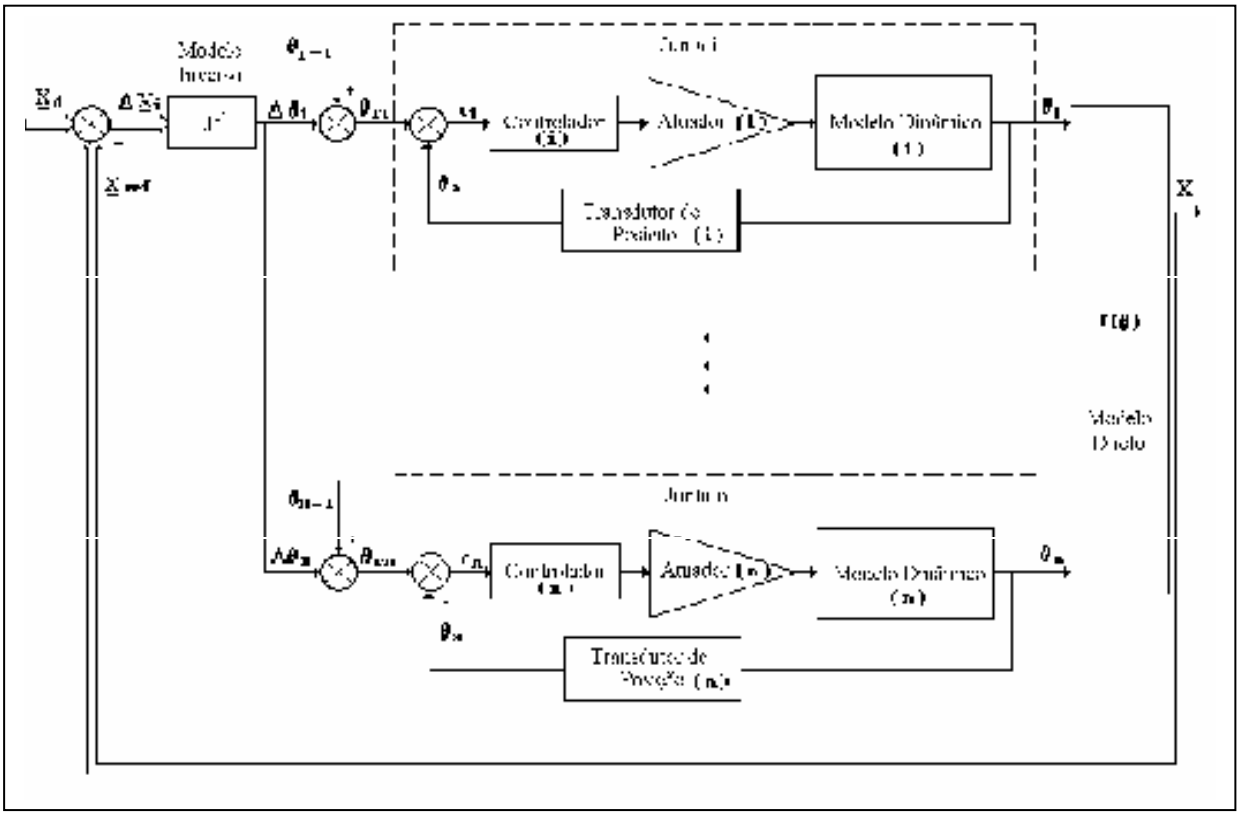


Figura 5.5 Sensor de posicionamento de junta linear.

Um controlador PID de junta fará as devidas correções levando-se em consideração os parâmetros dinâmicos da plataforma em estudo. Inicialmente este controlador será implementado diretamente em LABVIEW^{TM}, e em trabalho futuro deverá ser implementado encoders incrementais, e interface dedicada para controle de cada junta utilizando lógica reconfigurável.

O controle de posição da Plataforma pode ser executado através do controle de cada junta isolada (Nogueira, R. G-1995). O modelo de cada junta está assim definido por questões de facilitar os cálculos, mas toda junta deve estar coordenada como mostra na figura 5.6, de modo que o modelo dinâmico da estrutura poderá também estar definido.



J(θ) : Matriz Jacobiana,

F(θ): Modelo cinemático direto.

Figura 5.6 Estrutura de Controle da Plataforma de Posicionamento (6 juntas).

5.2.4 **Implementação Experimental**

5.2.4.1 **Simulação em MATLAB™**

A partir das equações obtidas no modelo apresentado anteriormente, podemos implementar algoritmos para simulação do problema, o primeiro ambiente foi o MATLAB™, de autoria da MathWorks Inc., escolhido devido a sua simplicidade e rapidez de uso e permitir a geração de gráficos tridimensionais imediatamente a partir dos resultados dos cálculos, e conseqüente validação de resultados.

Os gráficos foram traçados simplesmente determinando as coordenadas dos vértices das figuras e parametrizando retas entre eles. A função responsável pelo desenho de gráficos tridimensionais encarrega-se de corrigir o desenho com a perspectiva adequada e a atualizar essas correções caso seja requisitada uma nova posição para o ponto de vista.

Para simular o algoritmo desenvolvido utilizamos os recursos de animação do MATLAB™, a partir de um conjunto fictício de dados geramos seqüencialmente gráficos do movimento da mesa para cada conjunto de valores. Os gráficos foram armazenados e obtivemos uma animação da movimentação da plataforma.

Tanto os algoritmos para os cálculos escritos na forma de programas em MATLAB™ quanto o programa escrito para gerar a animação foram convertidos para aplicações dedicadas e independentes em C^{++} utilizando ferramentas de conversão disponíveis no próprio MATLAB™, que disponibiliza versões em C^{++} de suas funções e implementações orientadas a objetos de estruturas de dados que manipulam matrizes e vetores e definem as operações matemáticas sobre essas estruturas por meio de classes.

A funcionalidade dessas aplicações, tanto da simulação em MATLAB™ quanto do programa independente é apenas ilustrativa, apenas a aplicação dedicada poderia ser estendida para incluir a comunicação com o hardware do protótipo.

5.2.4.2 **Implementação em LABVIEW™**

Para supervisão e controle dos movimentos da plataforma, foi implementado um programa problema em linguagem visual utilizando o software LabVIEW™, desenvolvido pela National Instruments.

Um programa funcional é desenvolvido nesse estágio, uma vez que no momento da montagem do protótipo a comunicação com as interfaces estará ambientada nesse software. A linguagem visual desse ambiente é direcionada a criação de interfaces homem-máquina para sistemas de controle; o resultado de seus projetos é geralmente um aplicativo dependente com interface gráfica amigável, a partir da programação realizada sob a forma de diagramas de blocos, mostrados em anexo, as saídas e entradas dos blocos são retornos, e os parâmetros das funções via de regra, polimórficas, de onde se observa que a abordagem de programação do ambiente é a orientação a objetos.

Esse programa computacional se comunica com a interface (hardware), a qual controla diretamente sobre cada válvula dos atuadores hidráulicos. O programa consiste de menus que definem dois modos de funcionamento do protótipo: a movimentação automática ou manual. A figura 5.7 apresenta um exemplo de arquivo de entrada do programa que gera arquivos de distensões e de receitas e as suas saídas.

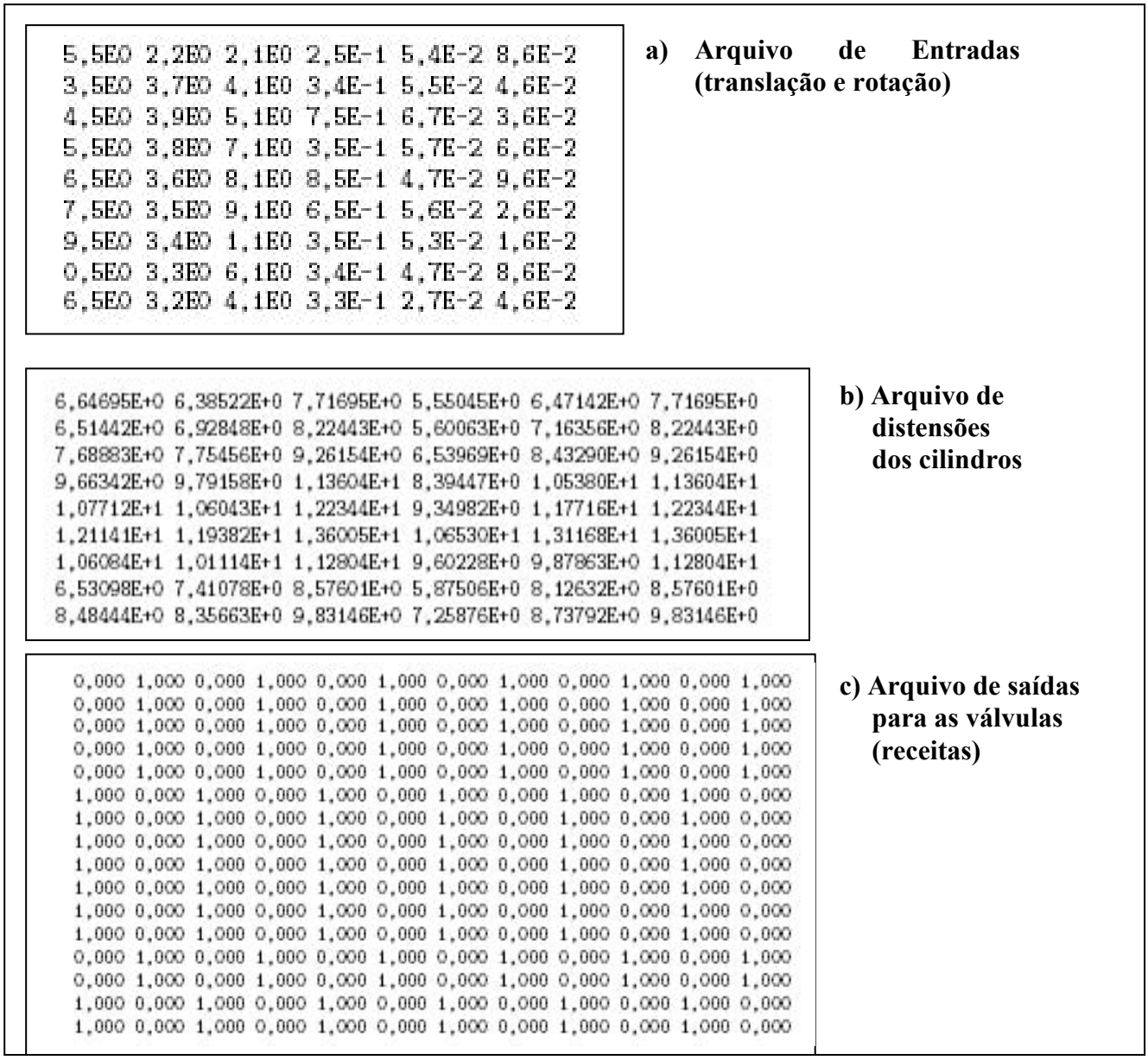


Figura 5.7: Estrutura de Controle da Plataforma de Posicionamento (seis juntas).

A movimentação manual é determinada apenas indicando quais cilindros hidráulicos devem avançar ou recuar. Os cilindros que não tem ação informada pelo programa são forçados a assumir uma movimentação definida pelos cilindros que se movem determinada e pela inércia do

sistema. A movimentação automática é controlada através de um arquivo de entrada, indicando quais cilindros devem atuar, e em que sentido; sendo este chamado arquivo de receitas, de formato (*.rct).

O programa de interface é independente do programa de modelagem implementado, designado programa de posicionamento, necessário para indicar uma posição exata ou um caminho de movimentação para a plataforma.

Conforme descrevemos no capítulo anterior, referente ao protótipo, as válvulas utilizadas são do tipo *on-off*, conseqüentemente o controle das mesmas é somente no sentido de movimentação e tempo de parada.

A utilização de válvulas de atuação proporcional, as quais funcionam impondo um sinal de tensão de entrada proporcional ao deslocamento desejado aos cilindros que é uma função linear. Assim, os valores numéricos obtidos no programa de posicionamento inicialmente elaborado, comprimentos teóricos para os cilindros, que correspondem em cada instante, às relações de proporcionalidade entre as tensões e as distensões dos cilindros.

Conseqüentemente, o programa de posicionamento foi alterado para que este se torne um gerador de arquivos de receita, ou seja, a partir do arquivo de descreve a movimentação, gera-se outro arquivo com as seqüências de movimentação dos cilindros (em função do tempo de atuação) permitindo assim, que a plataforma passe, ao longo das mudanças de direção de atuação, pelas configurações desejadas.

O controle do sentido de acionamento é realizado a partir de codificação apresentada no arquivo de saída de dados. A tabela 5.1 apresenta o principio de implementação: cada cilindro terá associado um par de bits indicando sentido e condição de acionamento.

Código		Atuação
1	0	Avanço
0	1	Recuo
0	0	Parado

Tabela 5.1: Código que determina condição e direção de acionamento dos cilindros.

A abordagem descrita limita fortemente a precisão do posicionamento da plataforma, uma vez que as posições não podem ser determinadas diretamente, apenas o conjunto de decisões de acionamento pertinente para chegar até elas. Assim, para obtermos uma posição aproximada em relação à requerida é preciso supor uma velocidade de avanço e recuo aproximadamente constante para todos os cilindros. A partir dessa velocidade e sabendo a diferença entre uma configuração e corrente seguinte estima-se o tempo em que certo conjunto de decisões de acionamento para os cilindros deve permanecer inalterado.

Os parâmetros e os retornos dos blocos dos diagramas em LabVIEW™ podem ter comunicação ao com o que se denomina nesse ambiente de "painel". O painel é a interface com o usuário, enquanto que o diagrama representa o código fonte, numa analogia com as linguagens de representação textual. A linguagem é puramente interpretada, ou seja, não há compilação, o programa executa interpretando diretamente o diagrama de blocos que descreve seu funcionamento, assim como as funções em MATLAB™, que são interpretadas de sua linguagem textual. As figuras 5.8, 5.9 e 5.10 apresentam alguns dos diagramas de blocos criados em linguagem LabVIEW™.

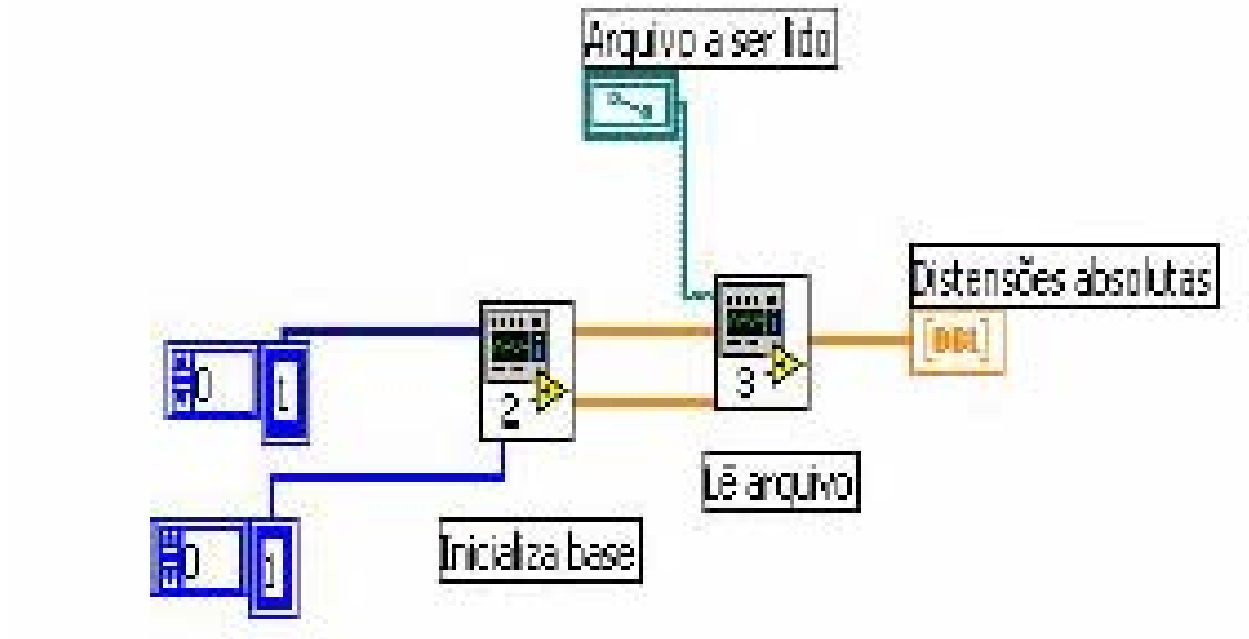


Figura 5.8 Diagrama Principal: Para a leitura de dados e o cálculo das distensões dos cilindros:

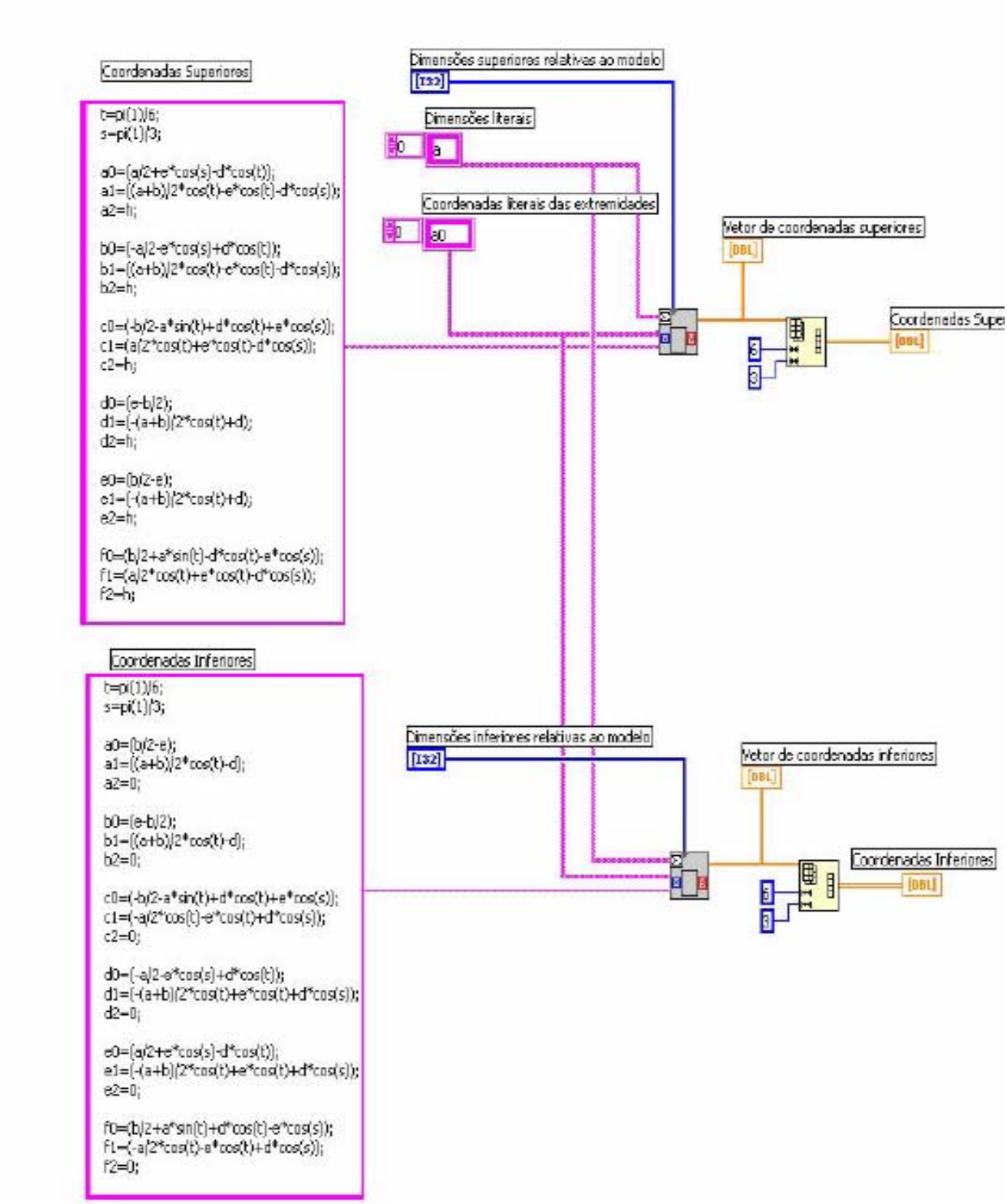


Figura 5.9 Programa de Inicialização dos Atuadores Hidráulicos da Plataforma.

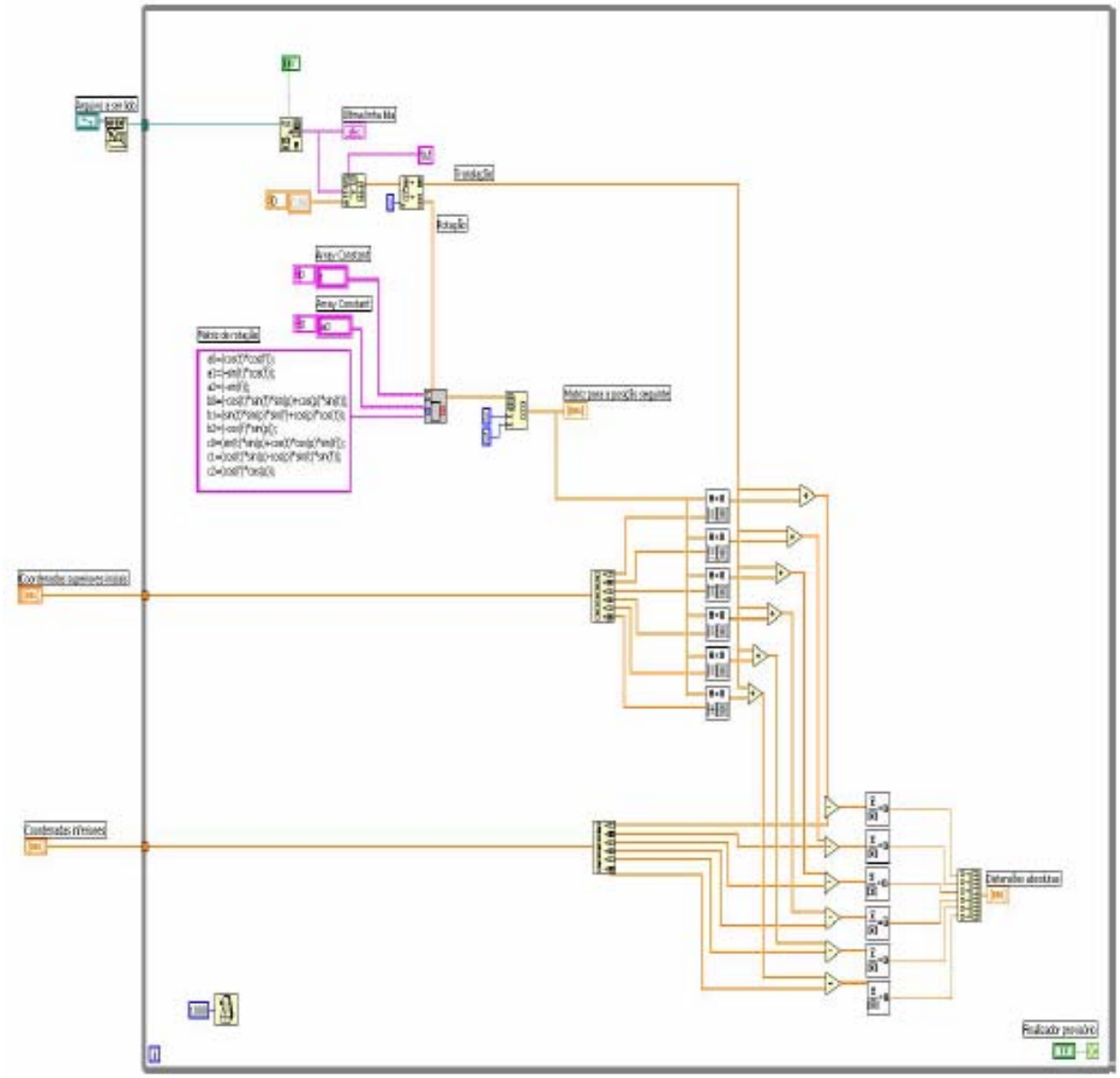


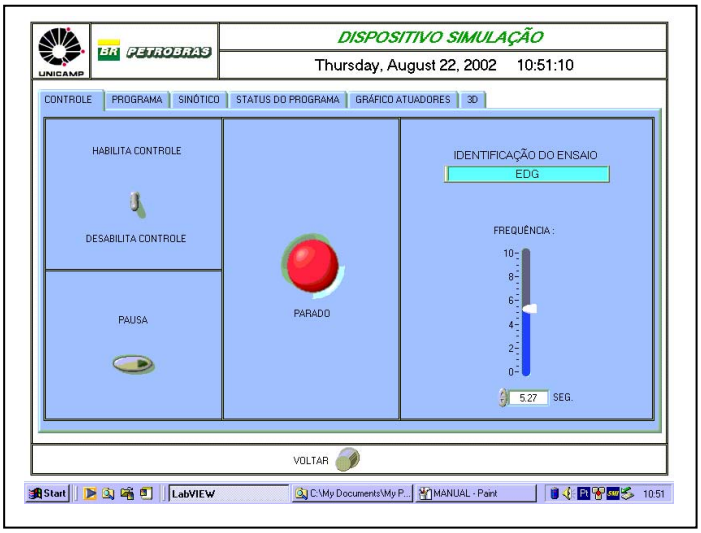
Figura 5.10: Leitura do arquivo de dados e cálculo das distensões de cada cilindro.

5.2.4.3 **Telas Gráficas Implementadas e Calibração**

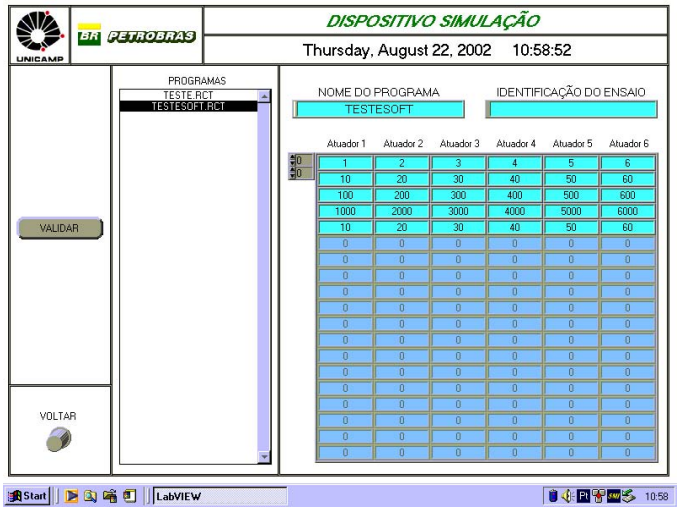
Conforme foi apresentado anteriormente, foram implementadas várias telas visuais utilizando o software LabVIEW^{TM} (figura 5.11). Entretanto o procedimento utilizado para descrição do movimento da mesa, gerando o arquivo de receitas é bastante impreciso, tornando imprescindível à implementação de um procedimento de calibração automática, sendo necessária

a implementação de um algoritmo para calibração das medidas de posição utilizando um robô industrial como ferramenta de calibração.

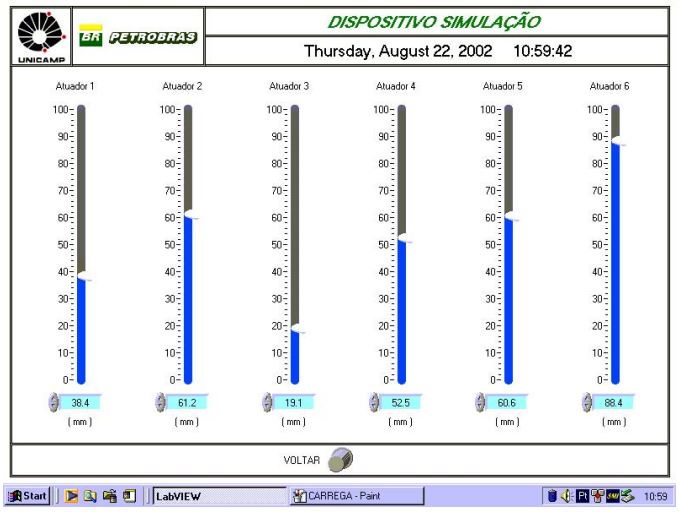
a) Tela principal do programa



b) Tela de controle (modo automático)



c) Tela carregamento do arquivo



d) Tela de controle manual

Figura 5.11: Exemplos de telas gráficas implementas em LabVIEW™

Ao mesmo tempo, foi implementado em cada um dos cilindros, um sensor externo para medida de posição, tais como régua potenciométrica para medida direta de posição ou um sistema potenciômetro de precisão, cabo e polia para medida indireta. Esse sistema fornecerá um sinal de tensão elétrico associado ao seu comprimento de distensão. Dessa forma, temos uma informação direta dos comprimentos dos cilindros que será utilizada na malha de controle de cada grau de liberdade da mesa a partir das entradas de referência de posicionamento provenientes do programa de supervisão e controle implementado em LABVIEW^{TM}, permitindo assim a correção em tempo real, através de um controlador PID implementado em cada um dos graus de liberdade.

Inicialmente, a implementação do controlador será feita diretamente em LABVIEW^{TM}, dispensando assim a criação de um arquivo de dados para posterior execução pela unidade hidráulica, sendo alterada ligeiramente, a solução proposta anteriormente, no programa visual já mostrado para geração do arquivo de receitas. Esta modificação consiste basicamente, de ao invés de gravar as linhas de código de acordo com o comprimento de atuação a ser imposto, o programa estará todo o tempo realizando sensoriamento do comprimento de cada cilindro e comparando com o valor requerido, ordenando avanço ou recuo conforme a diferença entre comprimento real e exigido seja positiva ou negativa.

Com o controle obtido da forma descrita, temos apenas de especificar um tempo de tolerância para a plataforma atinja a configuração requerida, pois mesmo que o tempo seja maior do que o necessário, as ordens de avanço e recuo oscilarão em torno da posição desejada até estabilização.

5.2.4.4 **Plataforma de Controle**

O hardware utilizado foi o Field Point da National Instruments, sendo constituído de uma controladora e módulos de entradas e saídas.

A figura 5.12 mostra a implementação deste dispositivo de controle no laboratório.

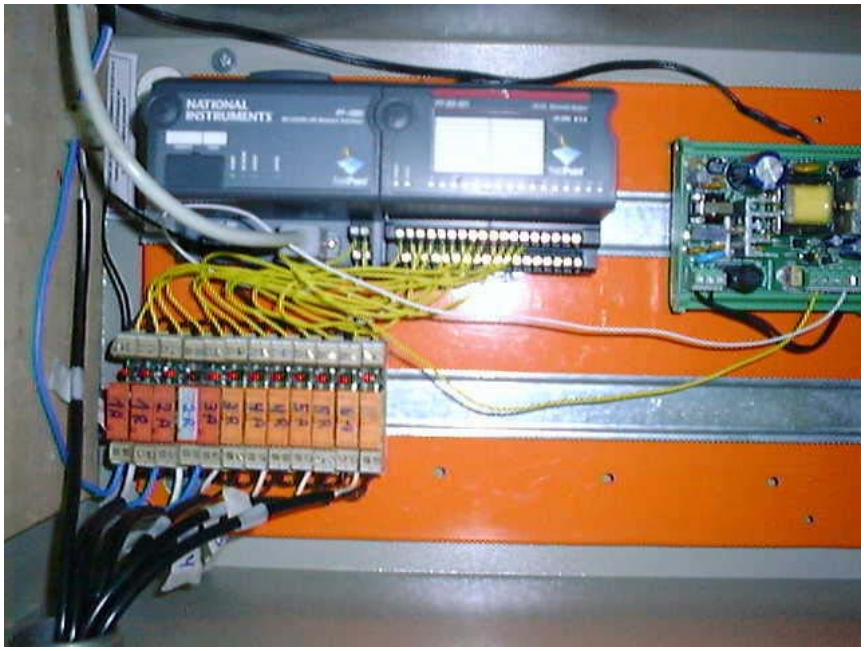


Figura 5.12 - Controlador Field Point National Instruments

5.2.5 **Conclusões da Implementação Proposta**

Do ponto de vista matemático, podemos observar que a simplicidade na implementação de um modelo cinemático para manipuladores paralelos, que facilita a utilização de hardware reconfiguravel.

Considerando a utilização de sensores de posicionamento, baseados em potenciômetros de precisão, verificamos que o programa de cálculo de posicionamento da plataforma de Stewart-Gough implementado é fortemente dependente das condições de calibração e inicialiação,

embora a plataforma implementada apresentou a capacidade de movimentação requerida pelo projeto (precisão e capacidade de carga).

Melhorias poderiam ser introduzidas no procedimento de inicialização, a partir da utilização de réguas potenciométricas, que permitiriam a obtenção direta dos parâmetros de movimentação a partir do conhecimento dos comprimentos dos cilindros. Entretanto, isto acarretaria num custo significativo de implementação.

Na fase de implementação, conseguimos uma noção exata de todas as nuances da modelagem geométrica com a necessidade de realizá-la em dois ambientes diferentes. Foram observadas as diferenças e semelhanças entre linguagens visuais e textuais na forma como se apresentam a um uso em engenharia.

Como possível etapa futura neste projeto de pesquisa pode-se contemplar a implementação de um controlador local para cada junta baseado em arquitetura reprogramável FPGA e de um sistema de comunicação entre software e hardware.

5.3 Sistema Automatizado para Posicionamento de Robôs em Célula de Soldagem de Veículos através de Visão Robótica

5.3.1 Introdução

Uma célula robotizada de soldagem de veículos (fig.5.13) exige o desenvolvimento de um conjunto de procedimentos para calibração do sistema de referência do Zero Veículo, que consiste do cálculo do posicionamento real de um determinado robô constituinte da célula em relação a um determinado ponto de referência (Zero Veículo) para re-posicionamento do referencial base do robô, permitindo assim a utilização de arquivos de tarefas gerados através de um software de programação off-line.

Figura 5.13: Célula Automatizada.

Nesta aplicação, serão apresentados programas computacionais implementados em ambiente Windows a partir do Sistema de Visão da National Instruments. Este pacote de programas foi designado Programa Gestor (tela principal apresentada na figura 5.14), sendo constituído de quatro módulos básicos descritos a seguir:

a) Calibração da Ferramenta terminal do robô;
b) Identificação do Zero Veículo através de pontos perfeitamente conhecidos no palete de medição através do cálculo da nova posição e orientação do robô utilizado;
c) Programa de Conversão de Ângulos de Orientação da ferramenta;
d) Entrada automática de dados do palete.

Figura 5.14: Programa Gestor.

Estes procedimentos de calibração e identificação foram realizados de duas maneiras:

a) **Contato Mecânico:** Método tradicional de identificação que utiliza uma ferramenta de calibração para ser anexada à pinça do robô de soldagem, utilizando um procedimento manual através de toques em dispositivo fixado no palete;

b) **Visão Robótica:** Através do desenvolvimento de software dedicado em linguagem LABVIEW para calibração automática e identificação de pontos do palete através de sistema de Visão Robótica, sem a necessidade que o operador realize contatos manuais com a peça.

5.3.2 **Metodologia Proposta**

A figura 5.15 apresenta a metodologia implementada para correção do erro de posicionamento do robô (cálculo do posicionamento real). Esta metodologia consiste de um novo cálculo do zero de posicionamento (posição e orientação da base do robô em relação ao palete do veículo), com correção a ser realizada posteriormente no robô.

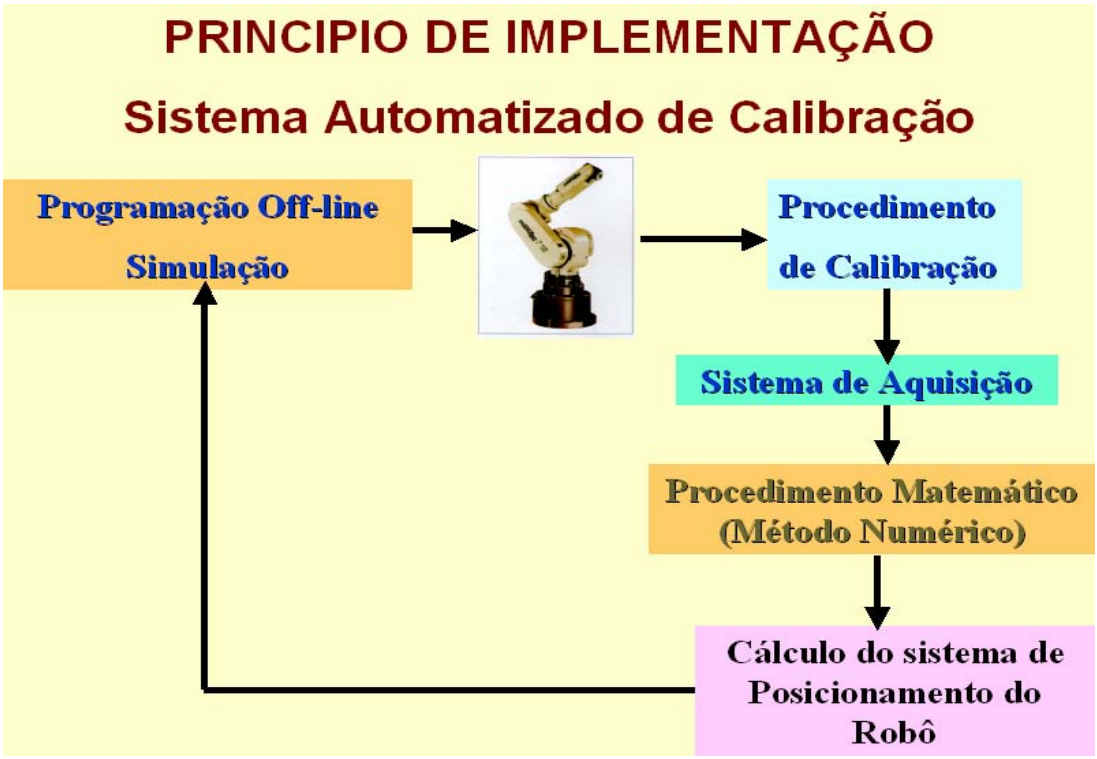


Figura 5.15: Sistemática utilizada para cálculo do posicionamento.

Durante o desenvolvimento desse projeto de pesquisa as seguintes etapas foram realizadas:

a) Implementação de programa off-line do robô utilizando dimensões reais do processo;

b) Calibração da Ferramenta de Calibração desenvolvida através do toque em diferentes orientações da haste (procedimento manual) ou no caso do sistema de visão robótica desenvolvido, através da visualização do diâmetro constante de circulo de esfera colocada num furo de precisão do palete (procedimento Automatizado). Os pontos adquiridos deverão ser feitos diretamente do robô, e juntamente com as dimensões aproximadas da ferramenta servirão de informação de entrada do programa de calibração desenvolvido;

c) Processamento destas informações através do módulo computacional CALIBRAÇÃO DA FERRAMENTA, permitindo assim a obtenção das dimensões precisas da ferramenta, que deverão ser utilizadas no programa de calculo do zero do robô;

d) Entrada de dados do palete, através de módulo computacional específico desenvolvido (PALETE). Neste módulo deverá ser introduzida a seqüência de pontos de calibração que deverão ser utilizadas no programa CALIBRAÇÂO;

e) Implementação de programa off-line do robô, contendo as dimensões da FERRAMENTA de calibração, e pontos de precisão do palete e seqüência de posicionamento, contemplando etapas de aproximação da ferramenta de calibração (haste ou visão robótica), utilizando as dimensões reais do processo. Com o objetivo de minimizarmos possíveis erros, a orientação da ferramenta deverá ser mantida aproximadamente constante;

f) Carregamento no robô do programa desenvolvido na etapa anterior, e posicionamento automático da ferramenta em relação ao palete (utilizando a haste de contato no pino de precisão introduzido nos furos do palete, ou através de sistema de visão robótica, que deverá visualizar circulo de diâmetro constante na esfera introduzida no furo do palete). Os pontos adquiridos deverão ser feitos diretamente sobre o robô devendo ser obtido um arquivo com extensão (*.lsv).

g) Processamento das informações obtidas deste arquivo (*.lsv), que juntamente as entradas automáticas das dimensões do palete e seqüência de pontos, zero do robô servirão de informação de entrada do módulo computacional PALETE, que permitirá assim a obtenção do novo zero do robô em relação ao palete,

5.3.3 **Requisitos necessários e configuração mínima para a aplicação**

Para utilização do procedimento descrito anteriormente, torna-se necessário a utilização da seguinte infra-estrutura (dispositivos mecânicos, hardware e software) relacionada a seguir:

5.3.3.1 **Dispositivos Mecânicos**

Como principal dispositivo deverá ser utilizada uma ferramenta universal de calibração contendo dispositivo de fixação da haste de calibração ou da câmera com possibilidade de rotação em diferentes orientações, permitindo que a mesma se adapte facilmente ao ambiente e espaço disponível de calibração. Assim os acessórios necessários serão:

a) **Dispositivo de Calibração por toque (procedimento tradicional):** constituído de pinos de calibração e uma ponta de precisão a serem introduzidos nos furos de precisão do palete, para posicionamento da haste de calibração;

b) Dispositivo de Calibração através de Visão Robótica (procedimento automático): constituída de esfera de calibração a ser disposta no furo de precisão do palete, para aproximação da câmera CCD, fixada na ferramenta de calibração.

5.3.3.2 **Hardware e Software**

Além do software industrial de programação do Robô e programação off-line (ROBCAD) disponibilizados para a aplicação, foi desenvolvido um programa Gestor para cálculo do Zero do Robô e Calibração da Ferramenta, e um programa de ajuda à aprendizagem de pontos do palete e calibração da ferramenta utilizando um sistema de Visão Industrial da National Instruments.

Os dois softwares desenvolvidos neste trabalho trabalham em ambiente Windows. A seguir são descritos a configuração básica (mínima) dos equipamentos necessários para esta aplicação:

a) Dispositivo de Calibração por toque (procedimento tradicional)

Interface de Visão National IMAQ PCI/PXI – 1407.

Câmera Digital ·Fonte de Alimentação (12V) – GANZ – Modelo YCH-02 – Color-High-Solution – 24 VAC, 12 VDC - 3,5-10,5 mm 1:1.0 IR 1/3 pol.

Dispositivo para acoplamento a Ferramenta de Calibração.

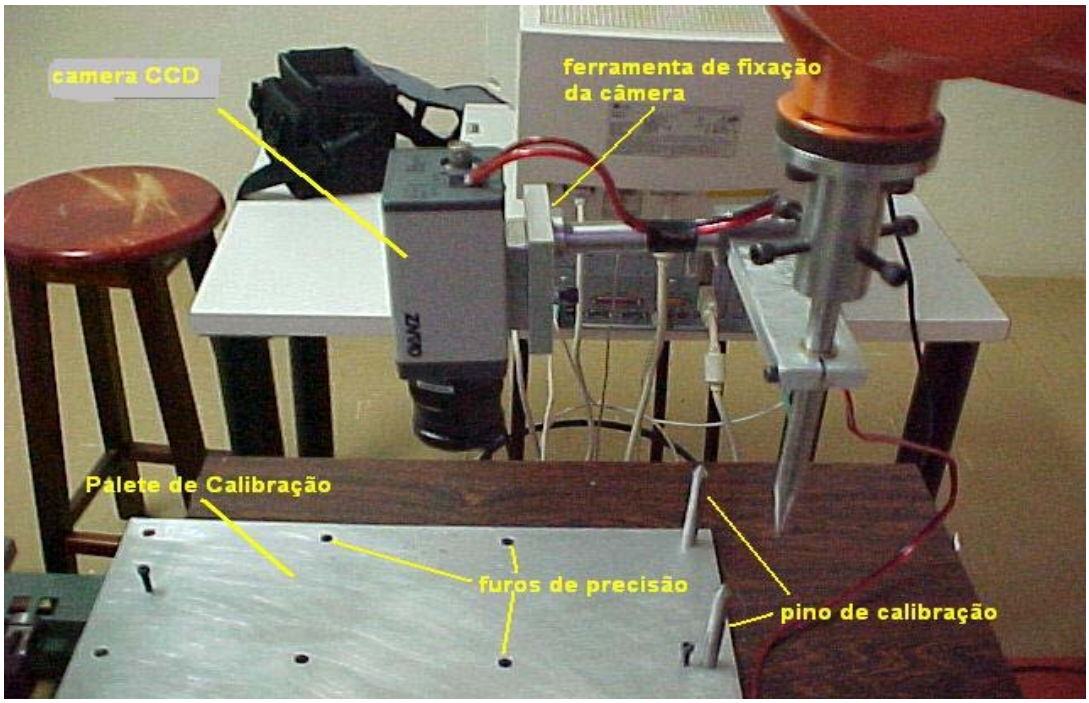
b) Computador

micro-computador PENTIUM III com slot PCI disponível, com memória RAM 128 MB e Monitor de Vídeo SVGA (800 x 600) (configuração mínima requerida)

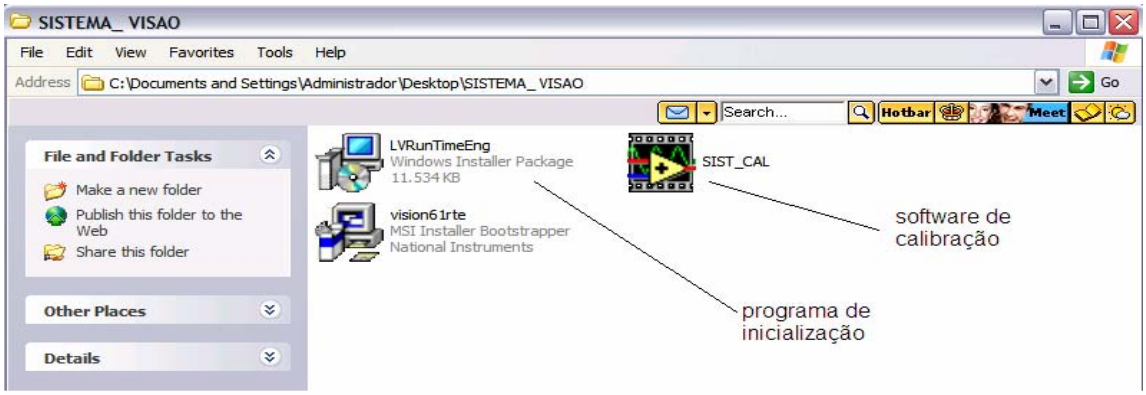
5.3.4 Descrição do software de Calibração da Ferramenta e Identificação de Pontos no Palete utilizando Visão Robótica

Para automação do procedimento de identificação dos furos de precisão do palete foi confeccionada uma ferramenta de calibração utilizando uma câmera CCD para Visão Robótica anexada à pinça de soldagem, conforme é mostrado na figura 5.16.

Para esta aplicação foi desenvolvido um software dedicado em linguagem LABVIEW para calibração da ferramenta para identificação de pontos do palete, a partir da utilização de uma câmera CCD, sem a necessidade de contato do operador com a peça. Esses programas computacionais permitiram a obtenção dos dados necessários para utilização do conjunto de programas computacionais descritos anteriormente para cálculo das dimensões da ferramenta e para identificação do Zero Veículo através da medição dos pontos de precisão do palete pela movimentação do robô.



a) Sistema de Visão Industrial.



b) Aplicativo de Visão Robótica desenvolvido em LABVIEW.

Figura 5.16: Sistema de Visão Robótica implementado (FEM-UNICAMP).

5.3.5 **Especificação dos softwares e drives desenvolvidos**

Conforme especificação anterior, foi implementado em linguagem LABVIEW, um programa computacional de Visão Robótica para a calibração da ferramenta e palete de precisão. Através deste aplicativo tornou-se possível:

a) **O ajuste focal e obtenção do diâmetro do circulo de referencia a partir de uma esfera de calibração:** permitindo-se assim a configuração dos parâmetros do dispositivo de visão robótica, que foram utilizados posteriormente para calibração da ferramenta e identificação do zero do robô a partir do palete de precisão.

b) **A calibração da Ferramenta utilizando Visão Robótica:** permitindo-se assim a obtenção de pontos necessários para calibração da ferramenta de calibração adaptada a pinça de soldagem do Robô (leitura de três pontos, através do mesmo posicionamento, mas em diferentes orientações da ferramenta do Robô).

c) **A obtenção dos Pontos de Precisão do palete pelo robô:** permitindo assim, a obtenção de pontos de precisão do palete através da movimentação do robô, conservando a mesma orientação da ferramenta.

5.3.6 **Descrição detalhada dos aplicativos desenvolvidos**

5.3.6.1 **Sistema de Calibração (SIST_CAL)**

Ao executarmos esse programa é apresentada uma tela inicial do sistema de calibração (figura 5.17), onde podemos observar duas possibilidades:

a) Configuração de parâmetros: permite configurar todos os parâmetros necessários para calibração da esfera (busca do diâmetro), ajuste focal, tolerância máxima, etc. Esses parâmetros são salvados automaticamente, permanecendo para serem utilizados por *default*, até alteração ou nova calibração pelo usuário;

b) Efetuar a calibração: neste modo, é realizada a calibração tanto da ferramenta quanto do palete, utilizando os valores de referência obtidos anteriormente e armazenados em variável de memória.

Figura 5.17: Tela inicial do Sistema de Calibração.

5.3.6.2 **Ajuste de parâmetros de calibração**

Este módulo permite o ajuste de parâmetros de calibração do sistema de Visão Robótica (foco e zoom). Ele contém ferramenta de suporte que permite a obtenção do raio do circulo de projeção da câmera sobre a esfera que deverá ser utilizado como referência no procedimento de calibração da ferramenta e de obtenção do novo zero do robô (figura 5.18). Durante o ajuste focal, será permitida a visualização de tolerância de medição, que poderá ser modificada para mais ou menos em função da precisão desejada durante o processo.

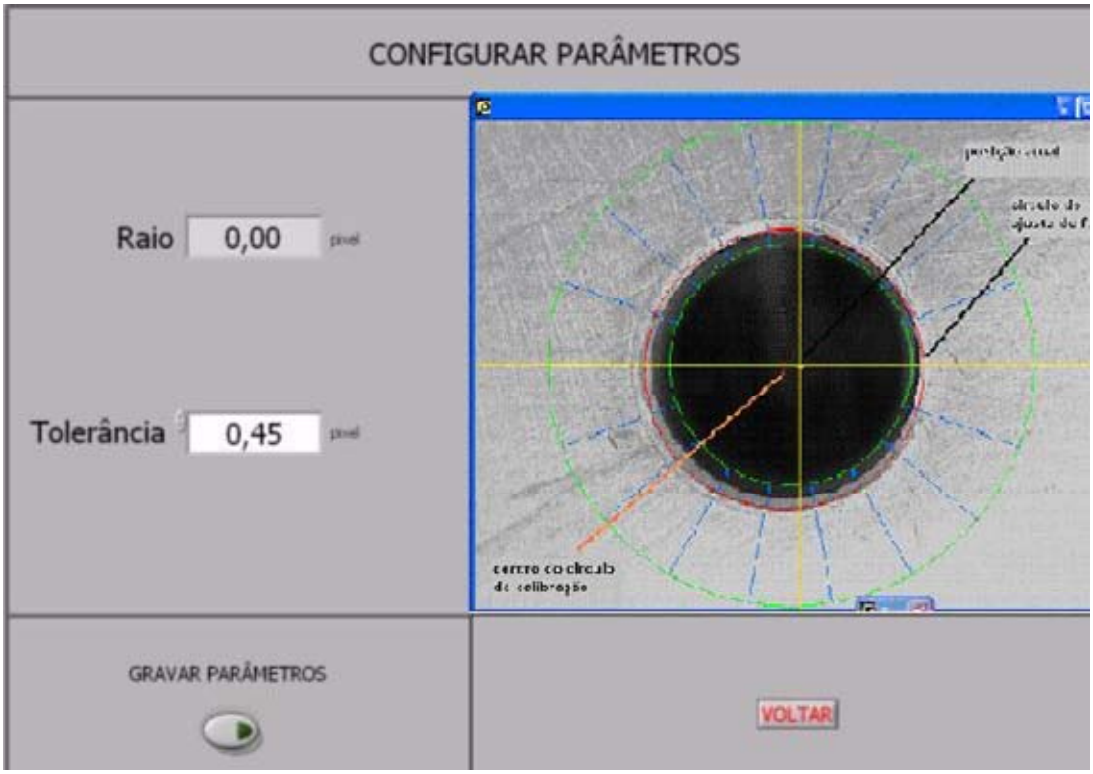


Figura 5.18: Tela de Calibração de Parâmetros.

5.3.6.3 **Procedimento de Ajuste Focal**

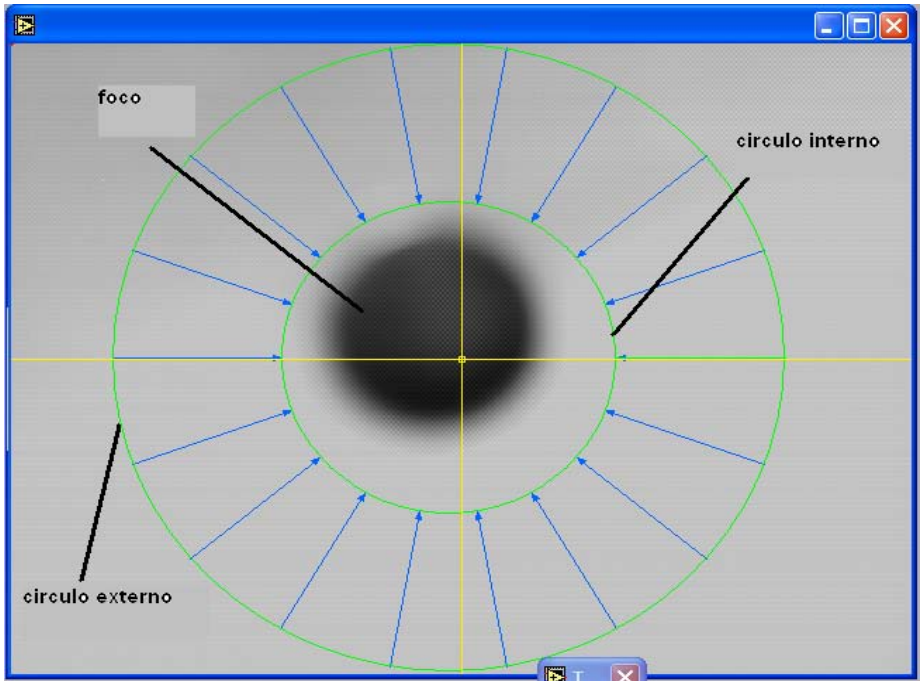
Este procedimento consistirá basicamente do ajuste do foco e zoom da câmera robótica em torno do diâmetro de uma esfera de calibração, que deverá ser mantido constante até o processo final de calibração da ferramenta e do palete (obtenção do novo zero do robô). Após esses parâmetros serem armazenados, os mesmos são utilizados como default até o final do processo de calibração (o foco e zoom da câmera não poderão ser alterados, pois isto implicaria numa nova etapa de ajuste de parâmetros).

Os seguintes passos deverão ser utilizados pelo usuário durante o processo de obtenção do raio de referência para calibração:

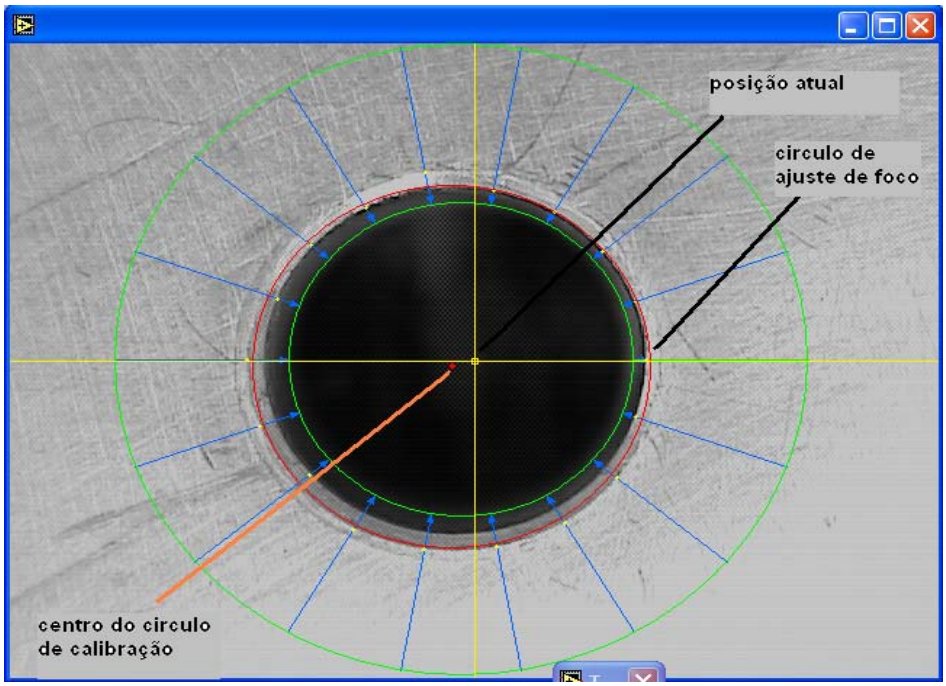
i) **Passo 1:** regulagem do foco e zoom da câmera robótica (figura 5.19).
ii) **Passo 2:** ajuste da câmera em relação a esfera de calibração de modo identificarmos um circulo de raio constante, que será utilizado a partir desse momento, como sistema de referencia da câmera. Em outras palavras em qualquer orientação do robô, será visualizado pela câmera um circulo de mesmo raio.
iii) **Passo 3:** No momento em que todos esses parâmetros estiverem bem ajustados pelo usuário, estes valores deverão ser salvos, e permanecerão como *default* até realização de novo ajuste e salvaguarda utilizando a mesma tela de ajustes.

A interface de visualização apresentada na figura 5.20 mostra as facilidades de operação para o usuário. Nesta tela é mostrado o raio do circulo escolhido para referencia de calibração. Esse raio aparecerá no momento que o ajuste focal estiver dentro dos limites mínimos e máximos do software desenvolvido.

Em função da precisão do processo de medida, o usuário poderá ajustar através de botão especifico a tolerância estipulada por *default*.



a) Ajuste de foco – tela típica.



b) Ajuste de parâmetros de foco.

Figura 5.19: Telas típicas obtidas durante o Ajuste Focal.

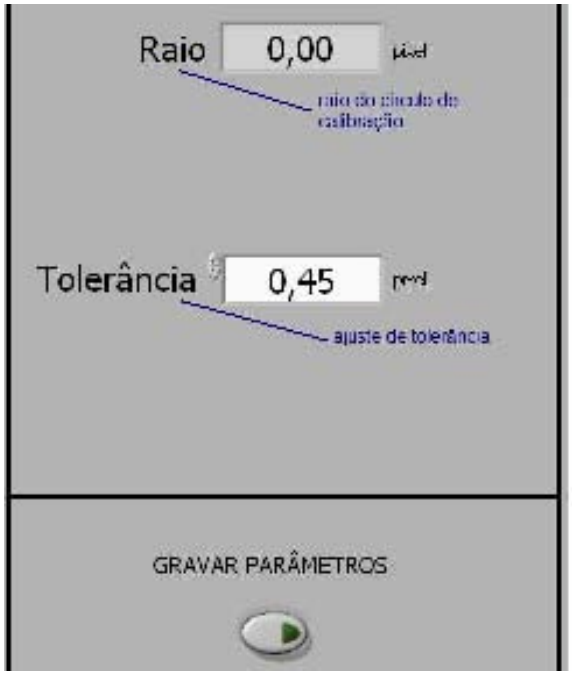


Figura 5.20: Tela de Ajuste do raio de referência da esfera de calibração e tolerância.

5.3.6.4 **Operação de calibração**

Este módulo é utilizado para a calibração da ferramenta do robô e do palete (obtenção do novo zero do robô). Quando utilizado para calibração da ferramenta: os três pontos são obtidos a partir do mesmo posicionamento e alteração da orientação da ferramenta terminal e no caso da calibração do palete: a orientação da ferramenta é mantida constante alterando somente o posicionamento da mesma em relação ao palete. A figura 5.21 mostra a tela típica desse módulo que permite:

a) Visualização do círculo de projeção da esfera de calibração, permitindo ao usuário o re-posicionamento do robô através da aproximação do ponto em vermelho (centro desejado) ao ponto amarelo (centro de referência).

b) Contém ferramentas gráficas que permitem uma indicação visual do raio do circulo de projeção da câmera sobre a esfera e posicionamento do robô nas direções XY e também para afastamento ou aproximação da esfera utilizada para calibração, com visualização do erro médio quadrático de distância do ponto (vermelho) ao centro (amarelo) (figura 5.22).

c) Informações para ajuda ao usuário relativo ao valor do raio e desvio quadráticos atuais e raio e tolerância de referencia.

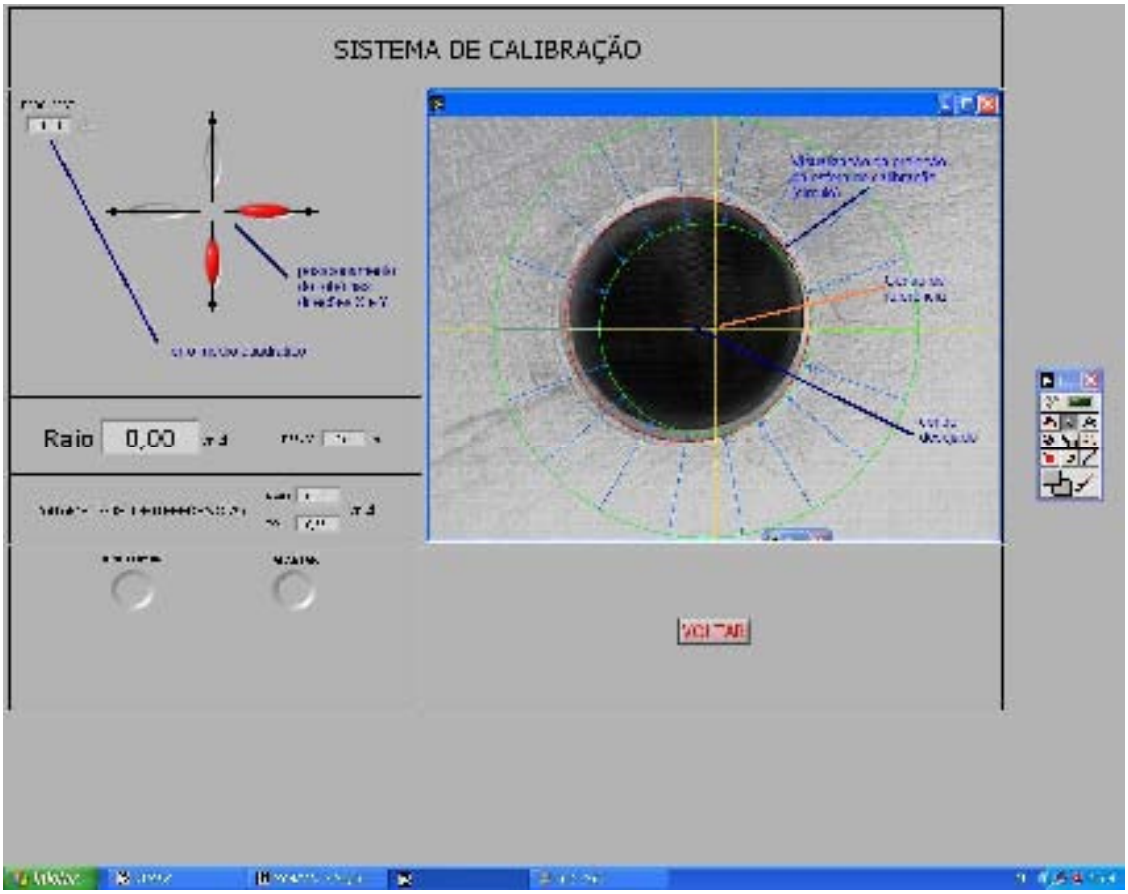


Figura 5.21: Tela típica do programa de calibração desenvolvido em LABVIEW.

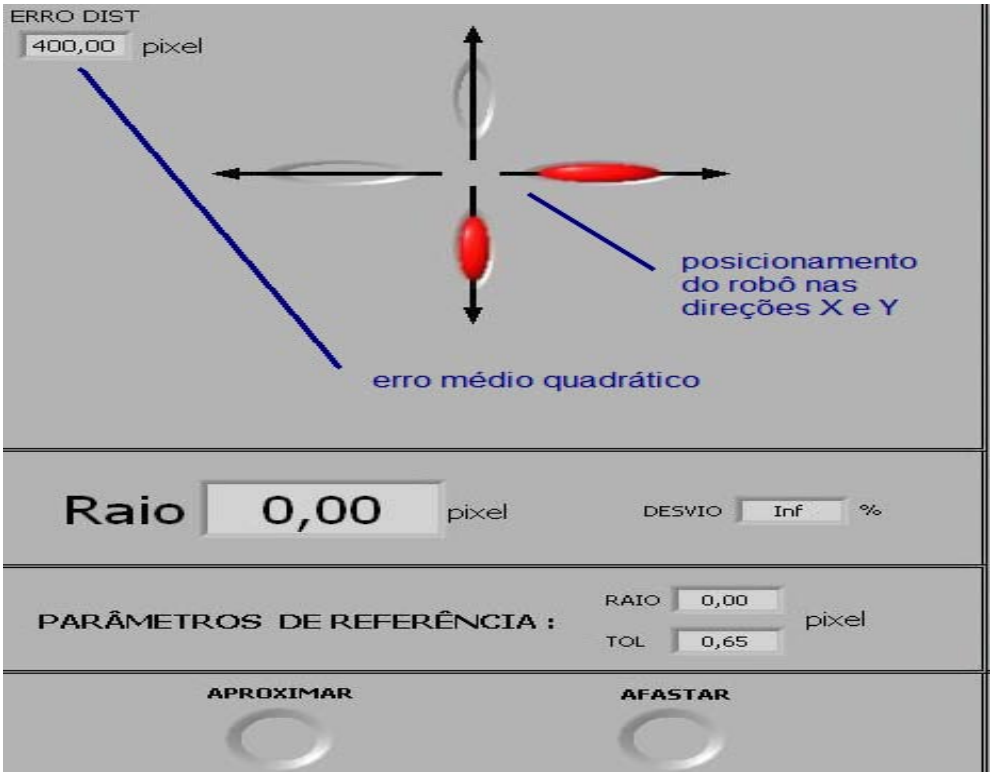


Figura 5.22: Tela de Calibração e parâmetros.

5.3.7 **Calibração da ferramenta utilizando Visão Robótica**

O aplicativo apresentado anteriormente é um facilitador ao operador do robô para obtenção de três ou quatro pontos necessários para calibração da ferramenta terminal do robô, a partir da orientação da ferramenta terminal sobre uma esfera de calibração colocada como referencia em posição fixa no espaço de trabalho do robô, permitindo obter esses pontos através da mudança de orientação da ferramenta, mantendo sempre o mesmo diâmetro do circulo de projeção da esfera de calibração utilizada.

Para ajuste do posição da posição atual em relação ao centro do circulo, uma tela de ajuda a movimentação do usuário nas direções X, Y do robô, permitirá a movimentação manual do robô pelo operador nestas direções..

Os botões para afastamento ou aproximação da peça serão sinalizados (alteração da cor do botão) no momento que o raio da câmera saia da tolerância arbitrada, indicando ao usuário informações para que este, aproxime ou afaste a câmera da esfera na direção do eixo Z dentro da tolerância utilizada. O ponto de calibração será considerado preciso no momento que as setas assinaladas em vermelho desaparecem, aparecendo uma mensagem informando que o ponto esta calibrado.

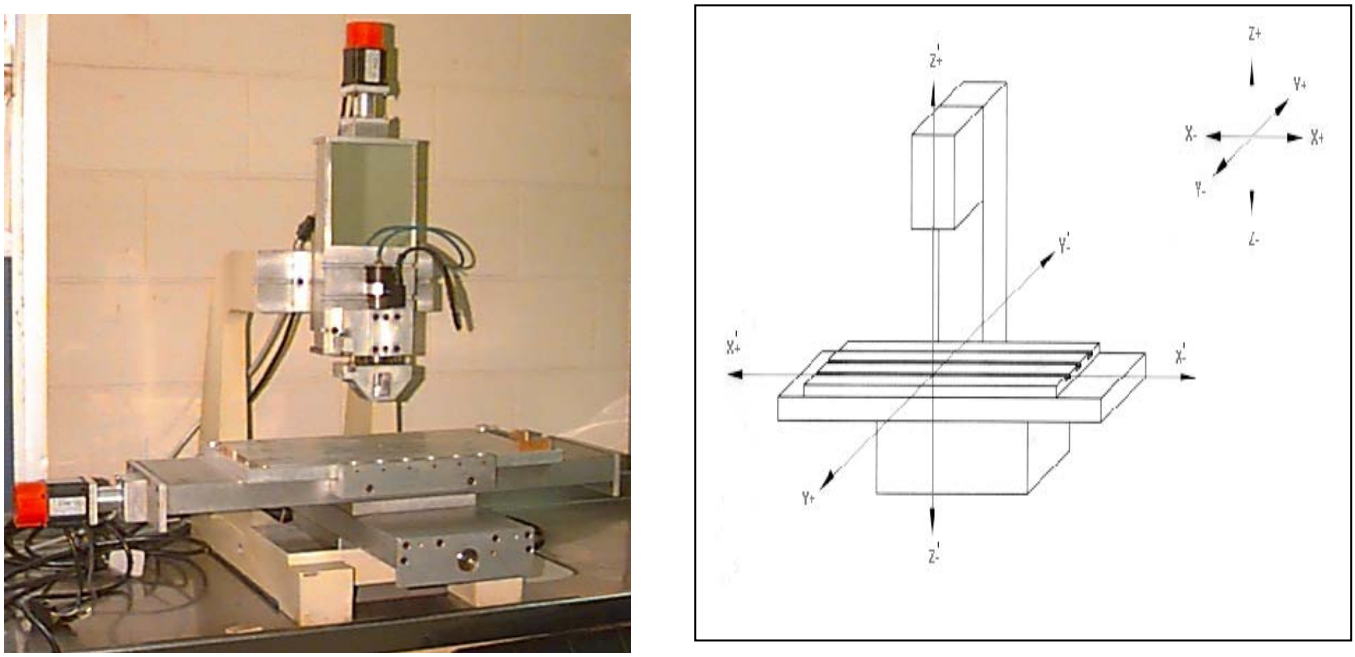
5.3.8 **Aplicativo para calibração do palete**

Este aplicativo computacional tem como principal objetivo facilitar o operador a obter os pontos necessários para obtenção do posicionamento da ferramenta terminal do robô em relação ao palete de calibração. O procedimento apresentado anteriormente para a calibração da ferramenta é basicamente o mesmo utilizado para calibração do palete, utilizando-se as mesmas informações obtidas durante a fase de calibração da ferramenta (diâmetro do circulo de referência da esfera) para obtenção de 3 ou 4 pontos do palete de precisão, sendo aconselhável que seja mantida a mesma orientação da ferramenta e altura da mesma em relação ao palete, garantida através do mesmo diâmetro do circulo de projeção da esfera de calibração utilizada.

5.4 **Robô Cartesiano de Posicionamento com controle baseado Control Motion da NI**

5.4.1 **Introdução**

Um dispositivo robótico cartesiano (fresadora) (fig. 5.23) é muito empregado industrialmente como uma maquina-ferramenta de controle numérico (CNC) para prototipagem de placas de circuito impresso – *Printed Circuit Board* (PCB), a partir da remoção mecânica de uma fina camada de cobre ao longo do contorno de cada trilha individual a fim de isolá-la eletricamente do restante do cobre da placa. Trata-se de um sistema mecânico de grande precisão, constituída de guias de rolamentos de fuso esférico e acionamento através de motores elétricos, (Souza, 1998).



a) Dispositivo Robótico Cartesiano b) Descrição dos movimentos do dispositivo

Figura 5.23: Dispositivo Robótico Cartesiano.

As máquinas-ferramenta devem executar movimentos automáticos, precisos e consistentes, de acordo com as instruções contidas no programa de usinagem previamente elaborado, sem necessidade de intervenções por parte do operador. Entretanto, estes dispositivos possuem uma arquitetura de controle pouco flexível e aberta, e com o passar dos anos podem se tornar obsoletas, necessitando de um *retrofitting* do seu sistema de acionamento e controle.

Um dos objetivos dessa aplicação é utilizarmos técnicas de prototipagem rápida descritas nos capítulos anteriores para implementação de eletrônica embarcada aberta, nesse dispositivo mecânico de grande precisão. Neste caso, utilizaremos a solução da National Instruments – o Control Motion, que funciona na plataforma para o ambiente de programação visual LabVIEW^{TM}, tornando um elemento facilitador, para que essa plataforma industrial opere a partir de imagens geradas em arquivos,

O principal objetivo desta aplicação é a geração automática das instruções de usinagem da máquina-ferramenta a partir dos dados gráficos fornecidos por um programa de CAD ou sistema de Visão Industrial, a partir de instruções geradas automaticamente pelo pós-processador implementado para geração de comandos utilizando o código G, linguagem padronizada de operação de máquinas CNC (Souza, 1998). A figura 5.24, apresenta um fluxograma de funcionamento do sistema.

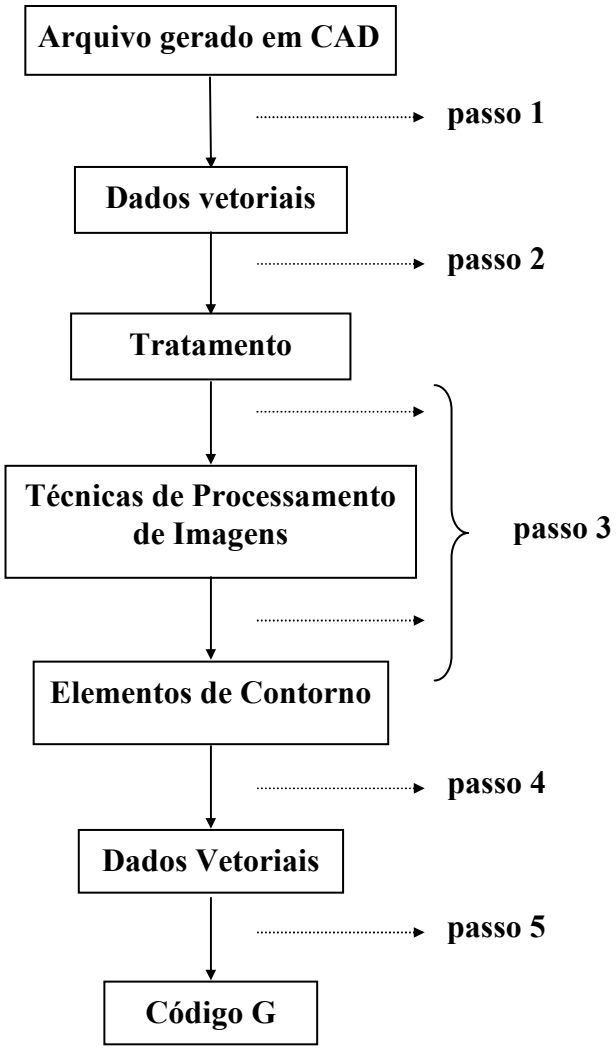


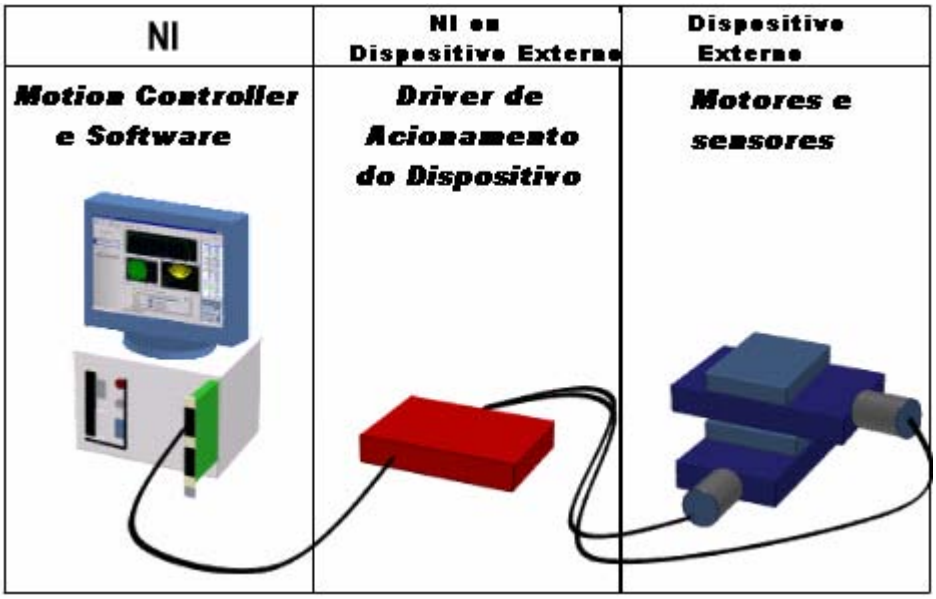
Figura 5.24: Fluxograma do software de funcionamento

5.4.2 **Plataforma NI Motion**

O NI motion da National permite a integração de diferentes dispositivos constituintes do sistema de controle de uma aplicação, constituindo-se um elemento facilitador de integração (figura 5.25).

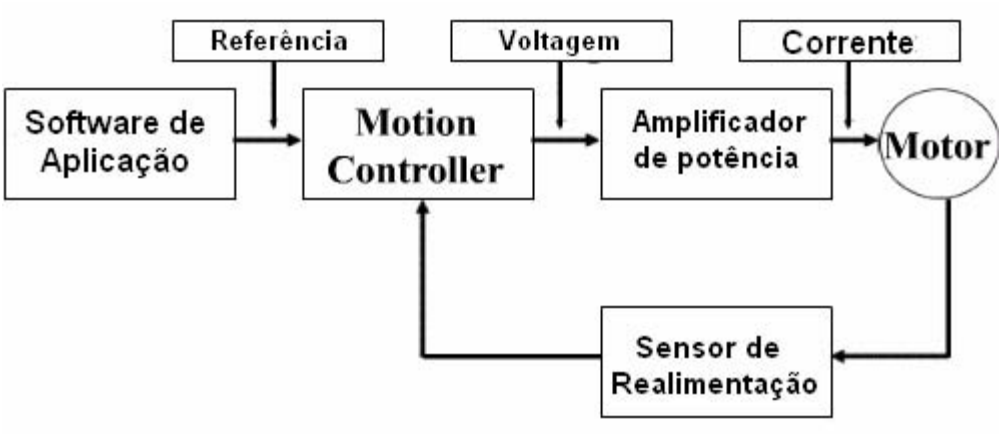
Esta plataforma é baseada nos principais elementos constituintes do sistema de controle, sendo integrados através de etapas de configuração, protótipo e ambiente de aplicação e desenvolvimento (figura 5.26).

a) Tela tipica do Aplicativo.

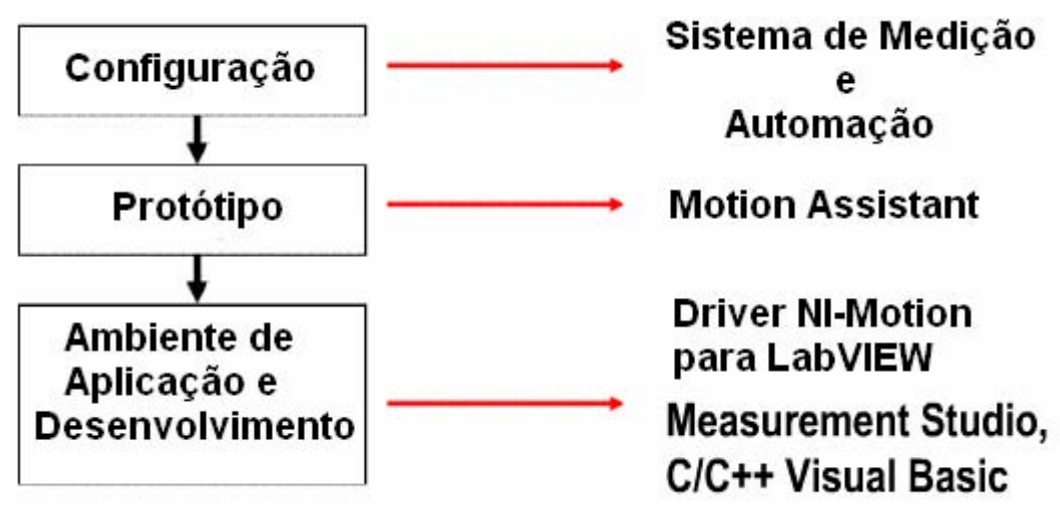


b) Conceito de utilização

Figura 5.25: Control Motion da NI



a) Elementos Básicos de um Sistema de Controle



b) Etapas de Prototipagem a partir do Software Control Motion.

Figura 5.26: Sistema de Controle e etapas de implementação do Control Motion da NI

5.4.2.1 **Principais características do software NI Motion driver**

Esta plataforma pode funcionar nos sistema operacional Windows ou Linux (figura 5.27), apresentando as seguintes características:

a) Sistema Operacional Windows e LabVIEW Real-Time Systems

- VIs e funções para LabVIEW, Visual Basic e C
- Measurement and Automation Explores
    - Configuração do Motion e outros componentes
    - Ajuste de parâmetros de controle de motores

b) Outros Sistemas Operacionais

- Hardware de controle Motion DDK
- Linux e drivers VxWorks para sistemas (www.sensingsystems.com)

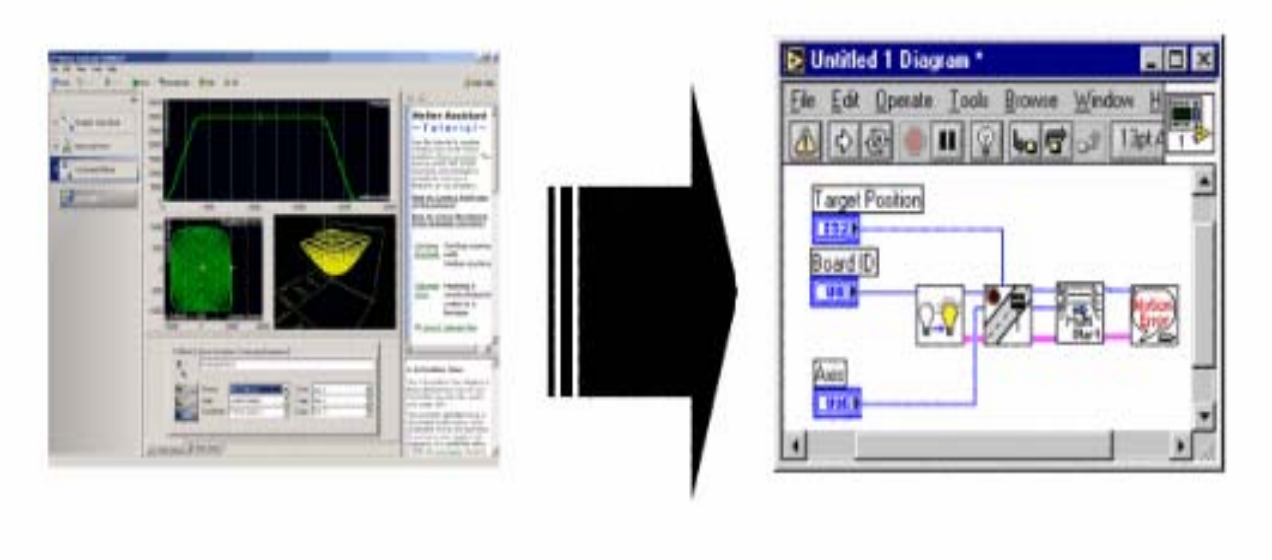


Figura 5.27: Control Motion da NI – assistente do software de prototipagem.

As principais características do software Motion Control estão baseadas na interatividade com os diferentes elementos constituintes de uma malha de controle (fig. 5.28), quer serão descritas nos itens a seguir.

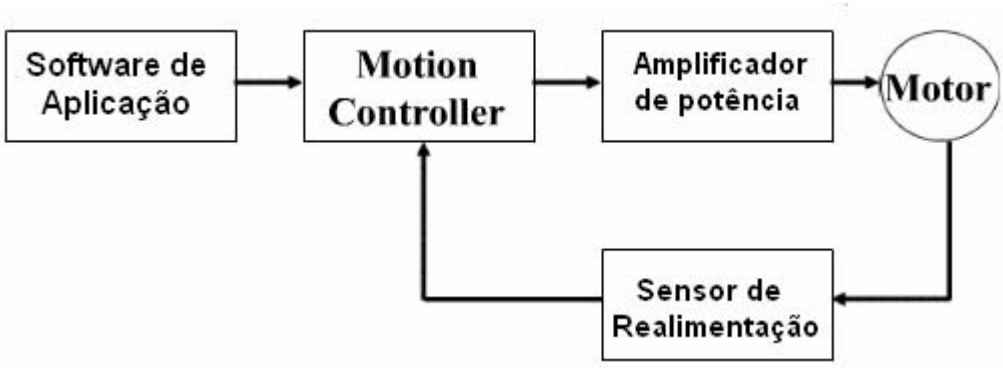


Figura 5.28: Diferentes elementos de uma malha de controle.

5.4.2.2 **Software de Aplicação**

O NI motion possui um assistente de software para prototipagem, que apresenta as seguintes características:

a) **Configurabilidade:** fácil utilização, e interatividade com o usuário, não necessitando de nenhuma programação;
b) **Programação:** alta flexibilidade e facilidade de integrarmos com outros componentes.

5.4.2.3 **Motion Controller**

A figura 5.29 apresenta um exemplo de perfil de velocidade, disponível no Motion Controller para controle de trajetória para uma junta simples. As principais funcionalidades NI Motion Controller são apresentadas a seguir:

a) Calculo das trajetórias de referência para cada movimento comandado interagindo facilmente com o usuário;
b) Calculo do torque necessário para acionamento para o drive ou amplificador do motor;
c) Sistema de Supervisão permite o monitoramento completo do sistema e paradas de emergência;
d) Controlador PID em malha fechada.

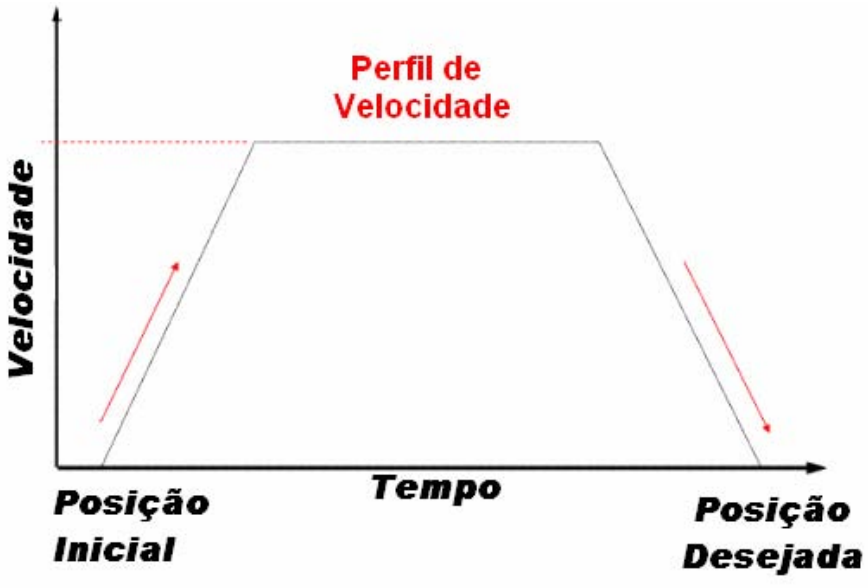


Figura 5.29: Gerador de movimentos para uma trajetória simples.

5.4.2.4 **Requisitos do Amplificador de Potência (Driver de acionamento)**

Os principais requisitos para especificação do sistema de acionamento (motores) e drive de potência são os seguintes:

a) O drive do motor deverá ser especificado em função de sua tecnologia: Por exemplo motores de passo com 2 fases, requerem um drive de potência para motores de passo com 2 fases, etc.

b) O drive de acionamento do motor deverá ser compatível com as suas características tais como: corrente de pico, máxima corrente continua, tensão de alimentação, etc.

5.4.2.5 **Seleção de Motores e Hardware Mecânico**

A figura 5.30 sintetiza dos principais critérios que deverão ser utilizados no processo de seleção de Motores para acionamento e controle de dispositivos mecatrônicos. Os principais requisitos para especificação do sistema de acionamento (motores) e drive de potência são os seguintes:

a) Análise do sistema mecânico;

b) Seleção de Motores e sistema mecânico de transmissão de potência (figura 5.31);

c) Seleção do drive de potência compatível com o motor e escolha do controlador.

Acionamento elètrico	Características Positivas	Características Negativas	Aplicações
Motor de Passo (Stepper Motors)	▪ **Baixo Custo** ▪ **Não necessita de realimentação de posição** ▪ Bom torque até o final do curso a baixa velocidade ▪ Bom para posicionamento básico	▪ Apresenta ruído e ressonância; ▪ Apresenta torque ruim a altas velocidades; ▪ Não é bom para cargas variáveis.	▪ Posicionamento ▪ Micromovimento
Brushed Motors (Motor com Escovas)	▪Custo Relativamente Baixo ▪Velocidade Moderada ▪Usa drives simples	▪ Necessita manutenção; ▪ Não é bom para ambientes limpos; ▪ Apresenta faíscas elétricas; ▪ Requer realimentação .	▪ Controle de Velocidade ▪ Controle de alto desempenho
Brushless Motors (Motor sem Escovas)	▪Sem manutenção ▪Longo tempo de vida ▪Alta velocidade ▪Bom para ambientes limpos	▪Alto custo; ▪Usa drives mais complicados; ▪Requer realimentação.	▪ Robótica ▪ Pick and place ▪ Torque muito alto ou aplicações rápidas

Figura 5.30: Critérios utilizados para Seleção de Motores (NI instruments).

.

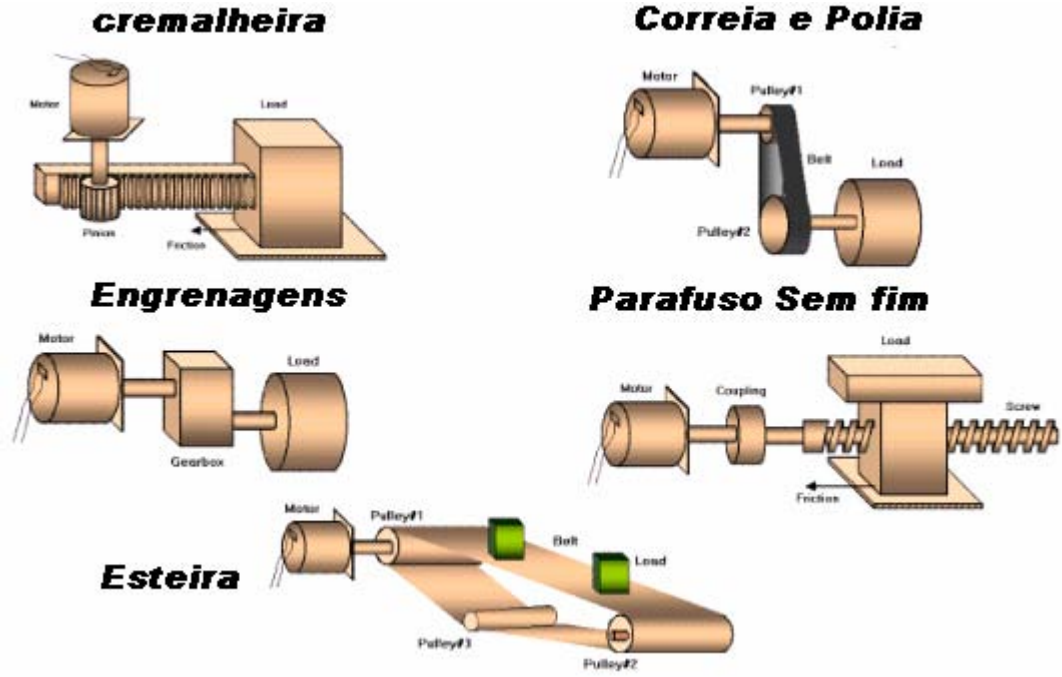


Figura 5.31: Tipos de Sistemas de Transmissão Mecânica.

Para correta escolha do sistema de transmissão mecânica necessitamos das seguintes especificações funcionais e tecnológicas:

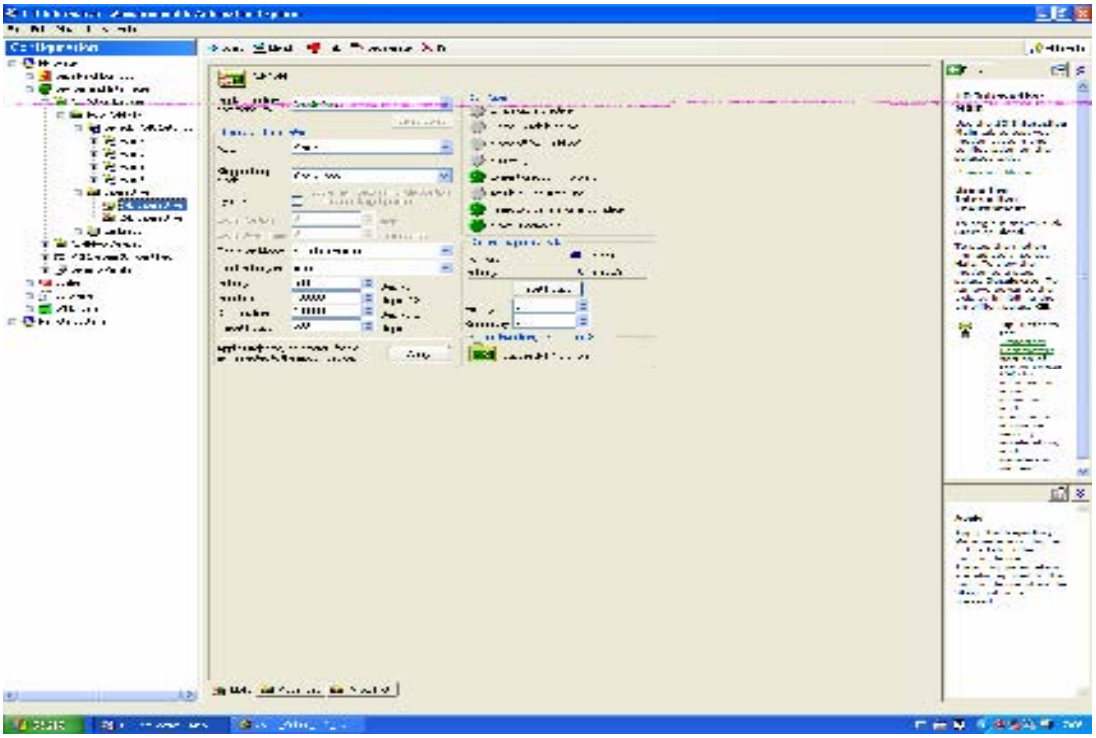
a) Resolução desejada;
b) Dimensões requeridas para operação;
c) Repetibilidade;
d) Capacidade de Carga;
e) Tipo do motor.

5.4.3 **Proposta de Aplicação**

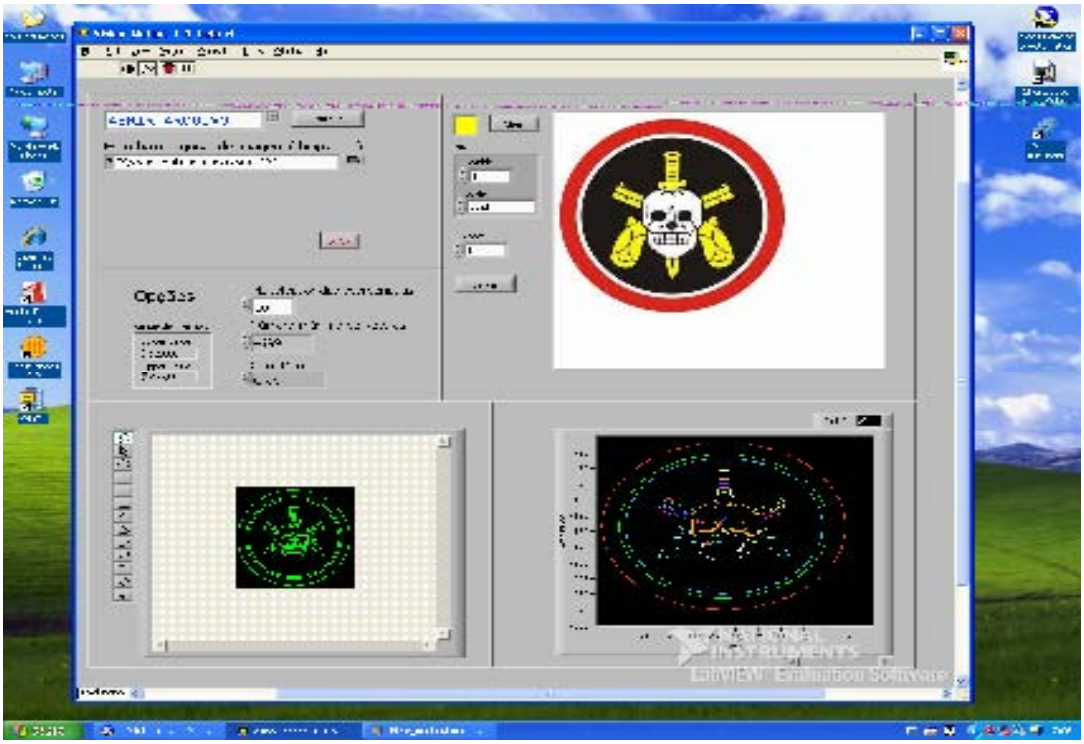
Como proposta de aplicação deste dispositivo foi implementada uma célula para projeto de circuitos impressos, onde nossa intenção não é a de substituir os métodos convencionais de fabricação, como os métodos químicos abrasivos e os de eletrodeposição seletiva de cobre, mas apresentar uma técnica complementar para a prototipagem rápida e econômica de placas de circuito impresso, a partir da fabricação da placa definitiva pelos métodos convencionais (figura 5.32).

Figura 5.32: Sistema Implementado.

Os resultados obtidos na etapa de usinagem são apresentados a seguir e comprovam a eficiência dessa aplicação em termos de precisão de posicionamento, tempo e custo de implementação dessa nova arquitetura.. A figura 5.33 apresenta algumas telas típicas do software implementado no NI Control Motion.



a) Tela de configuração de parâmetros do sistema de acionamento.

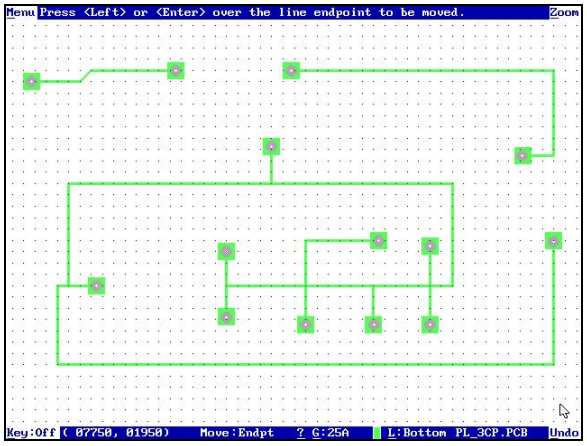


b) Tela de aquisição e tratamento de imagens.

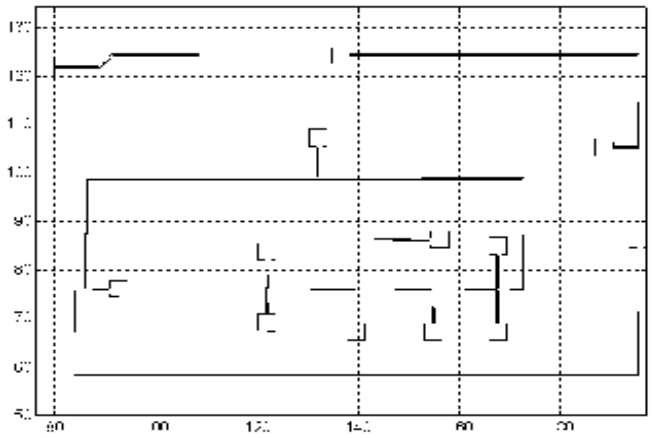
Figura 5.33: Telas típicas do Control Motion da NI Instruments.

5.4.4 **Resultados**

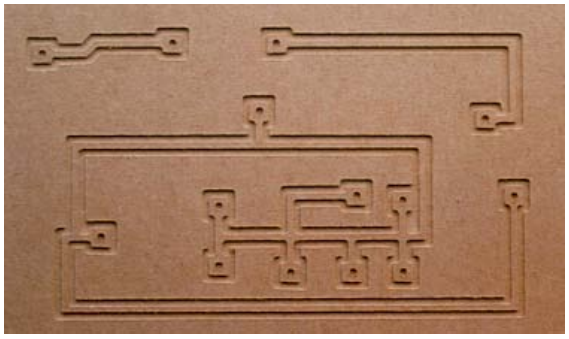
A figura 5.34 mostra as diferentes fases da implementação proposta para esta aplicação, desde o layout do PCB gerado através de CAD (a), os resultados da digitalização realizada com extração de contornos (b), chegando na confecção final da placa de circuito impresso (c).



a) Layout PCB gerado em CAD.



b) Digitalização com extração de contornos.



c) Confecção Final da Placa de Circuito Impresso.

Figura 5.34: Diferentes fases da implementação proposta.

5.5 **Integração de Robôs Industriais para operações cooperativas**

Esta implementação é baseada na coordenação e integração de dois robôs industriais (IRB 140 e IRB 1400 da ABB^{TM}) e uma mesa de posicionamento com três graus de liberdade (robô PRR), desenvolvida na UNICAMP para auxiliar o trabalho cooperativo de robôs manipuladores convencionais em trabalhos de usinagem e soldagem de dispositivos mecânicos complexos que necessitam de mais graus de liberdade para realizarem trajetórias complexas (figura 5.35).

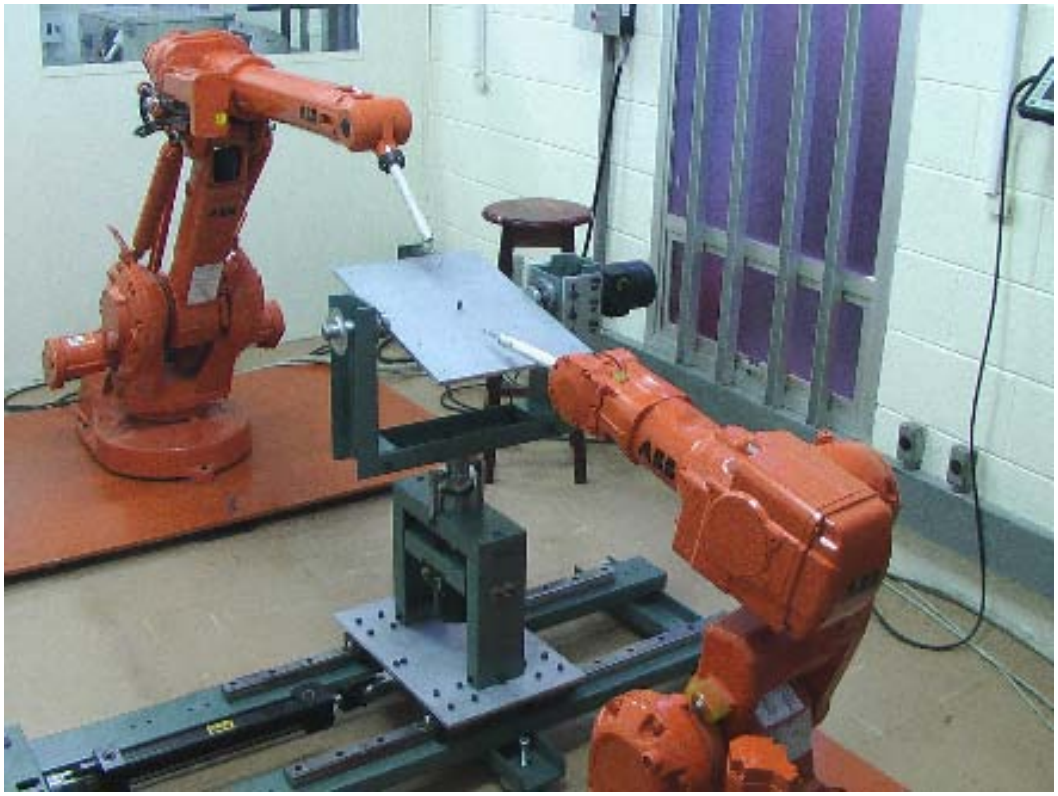


Figura 5.35: Célula integrada de manufatura.

Esta plataforma é constituída de atuadores hidráulicos: onde o primeiro grau de liberdade possuiu um cilindro de posicionamento para movimentação da base numa determinada direção (movimento translacional), e os dois últimos graus de liberdade responsáveis pela orientação espacial da mesma, possuí dois motores (movimento rotacional). Os sensores de posicionamento são encoders incrementais (juntas rotativas) e um potenciômetro linear (junta linear). A figura 5.36 apresenta um detalhamento da mesa de posicionamento implementada.

Figura 5.36: Plataforma de posicionamento.

O sistema de supervisão e controle foi implementado em linguagem LabVIEW™, muito utilizado industrialmente, em projetos de dispositivos relacionados as áreas de medição e controle podem utilizar LabVIEW™ para criar aplicações personalizadas que rodam em plataformas NI (National Instruments) com E/S reconfiguráveis baseadas em FPGAs. Juntos, o LabVIEW™ FPGA e o hardware NI para E/S reconfiguráveis permitem a criação de uma plataforma flexível para o desenvolvimento de sistemas sofisticados que antes somente eram possíveis com hardware projetado de forma dedicada.

O software LabVIEW™ pode ser usado para controle e supervisão. O controle das juntas robóticas é realizado através de uma interface A/D que comanda os motores das juntas. Um programa de supervisão e controle, residente num computador é responsável pelo gerenciamento e controle das informações provenientes de sensores e atuadores do sistema.

Uma interface de visualização implementada em ambiente Windows, foi desenvolvida telas de supervisão que coletam as informações dos sensores, dos programas de tratamento matemático (modelagem cinemática da mesa), as informações sobre o sistema (velocidade, parâmetros do regulador, números de pontos da trajetória, etc), da inicialização e da calibração automatizada.

A interface de aquisição de dados utilizada neste trabalho foi a PCI 9112 da ADLINK Technology. Esta interface recebe as informações provenientes dos encoders e do potenciômetro para determinação da posição atual das juntas constituintes da mesa. Estas informações são comparadas com os valores de referência, logo após esta interface calcula o algoritmo de controle envia para a saída do controlador enviando estas informações (posição, velocidade) aos acionadores de cada grau de liberdade da mesa (fig. 5.37).

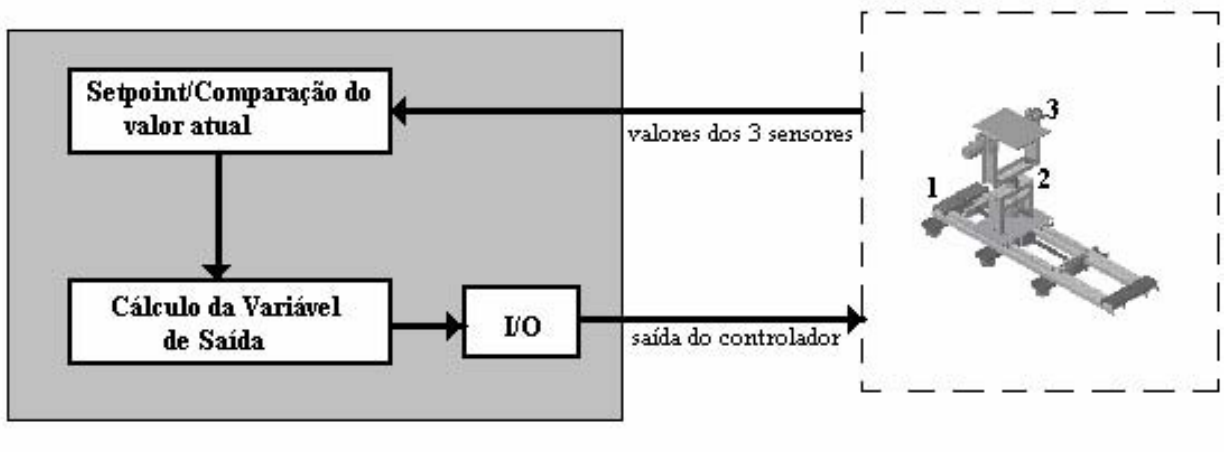


Figura 5.37: Esquema do sistema controlado.

Todos os valores de referência são calculados em tempo real, mesmo a cinemática inversa com as altas não linearidades. Outros tipos de sensores podem ser incluídos na bancada ou simplesmente através de uma VI (Virtual Interface) do software LabVIEW™. Normalmente, o projeto de uma VI, deverá conter os seguintes elementos de base:

a) Painel Frontal: É uma interface interativa entre o usuário e o programa. É onde o usuário entra com os dados usando o mouse ou o teclado, e então vê os resultados na tela do computador. Isto significa que o Painel Frontal é uma janela de execução, onde são encontradas as entradas de informação (control) e as saídas de informação (indicator)

b) Diagrama de blocos: É a representação de um programa ou algoritmo. É onde o programador cria o seu programa. O programa consiste de menus que definem a movimentação automática ou manual da mesa. A movimentação manual é determinada apenas indicando quais motores devem ser acionados. A movimentação automática é controlada pelo programa desenvolvido para tal fim (figura 5.38).

c) Tela principal: A ser utilizada para o controle direto da plataforma, foi elaborado um programa em linguagem visual. Esse programa de interface se comunica com a placa de interface com o hardware, a qual controla diretamente os motores (figura 5.39).

d) Tela contendo os diagramas de conexão de cada bloco: Os diagramas são as conexões de cada bloco, e apresentam características específicas ao projeto implementado, de acordo com os elementos que serão implementados na VI, as conexões são realizadas neste painel .

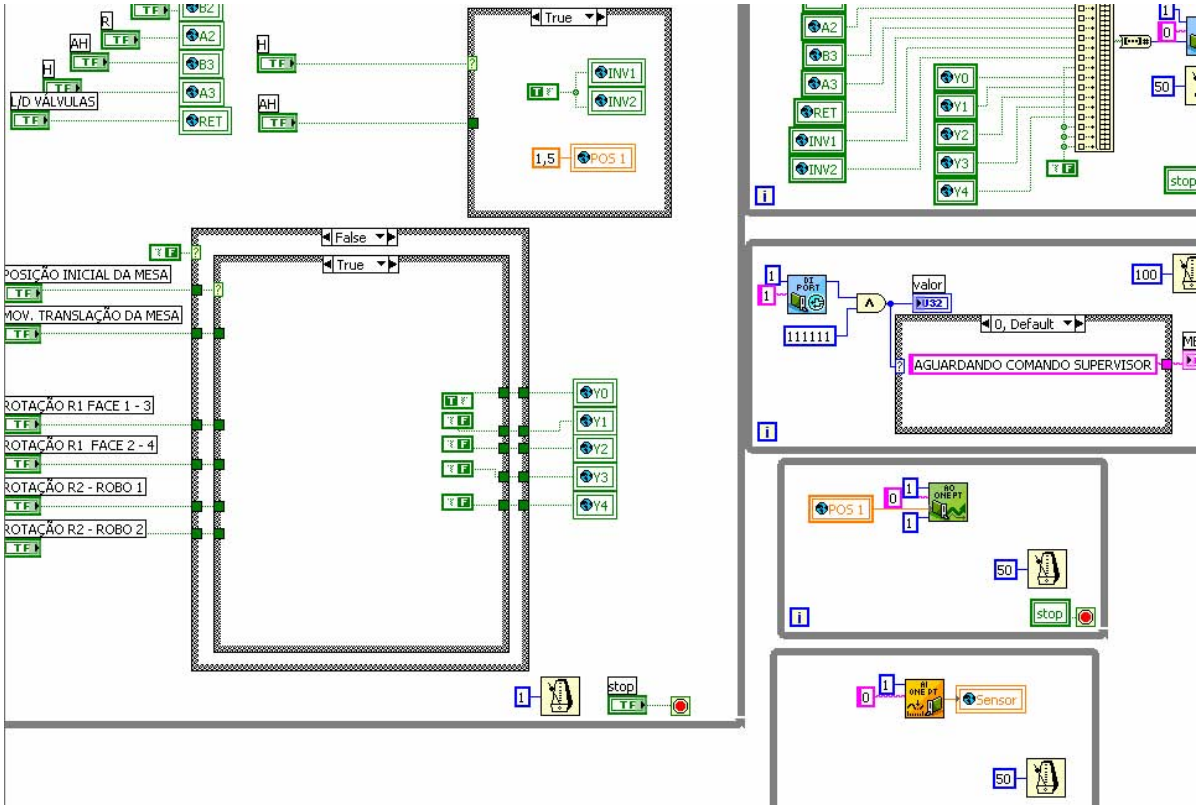
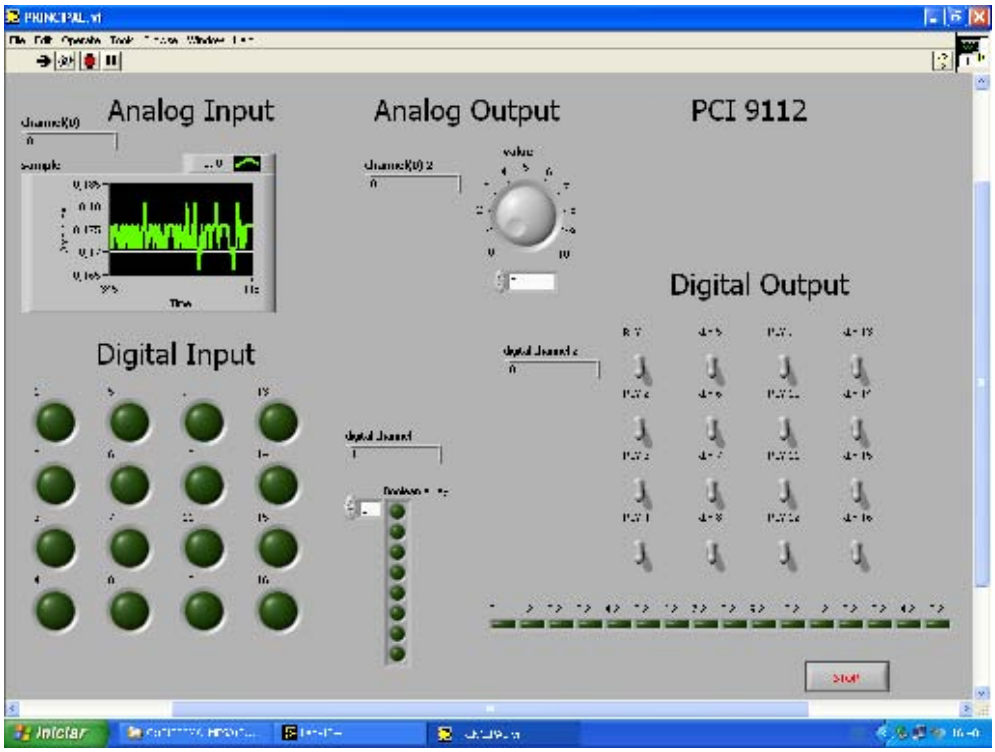
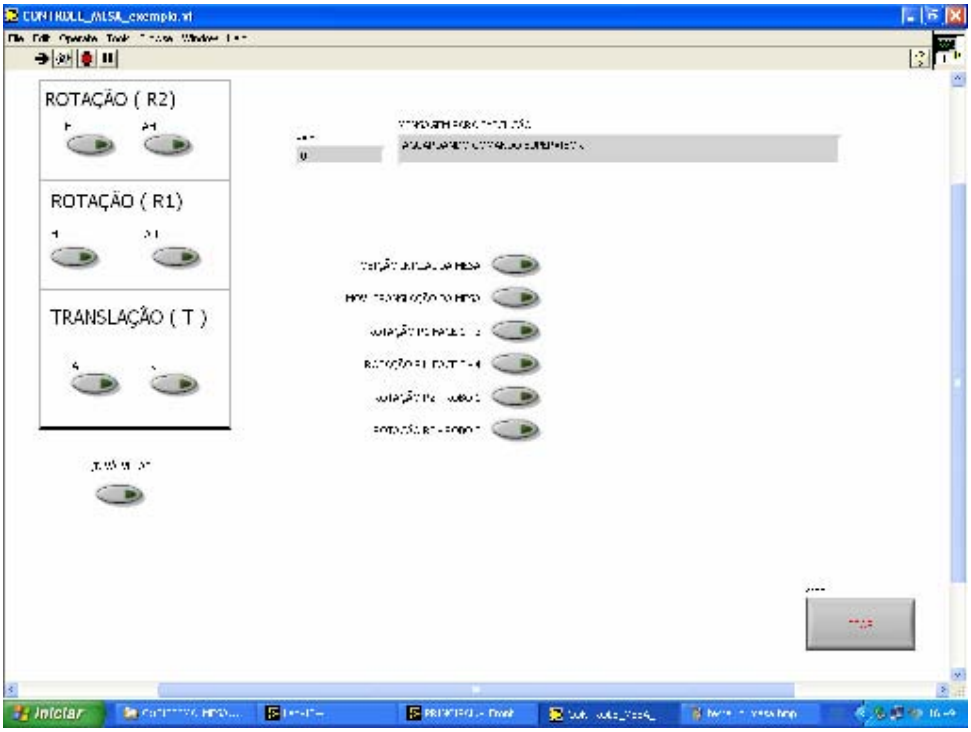


Figura 5.38: Tela diagrama de blocos implementados em LabVIEW.



a) Tela de testes I/O



b) Tela de Supervisão e Controle

Figura 5.39: Telas implementadas para o sistema de Supervisão e Controle.

5.6 **Plataforma Robótica WEB**

Através da evolução das Tecnologias da Informação e das Telecomunicações, a Internet passa a revolucionar a Ciência, a Economia e a Sociedade. Desta forma, a utilização de ambientes educacionais com recursos computacionais permite um maior acesso a novos conhecimentos de maneira mais rápida, acarretando impactos sociais e novas áreas de pesquisa e desenvolvimento.

A FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo, através do programa TIDIA – Programa Tecnologia de Informação no Desenvolvimento da Internet Avançada viabilizou a formação de parcerias entre áreas acadêmicas, governamentais e empresariais do Estado na área de Tecnologia da informação voltada para especificação, projeto e implantação de ferramentas aplicáveis ao processo de ensino e aprendizagem, incentivando o desenvolvimento de novas tecnologias, tanto nas áreas de hardware como de softwares e redes para o desenvolvimento científico e tecnológico.

5.6.1 **Ambientes Colaborativos para Ensino e Pesquisa baseados em WEB**

Segundo Traylot et al. (2003), a utilização de ambientes ou plataformas de ensino deve conter informações que possibilitem que o aprendiz evolua conforme seu ritmo e flexibilidade. Para tanto, estes ambientes devem promover a integração de conhecimentos, inovação e experiências para a resolução de pequenos problemas, além de motivar e melhorar a visualização da continuidade do aprendizado. Visando tais aspectos a utilização de laboratórios virtuais e a Internet passam a aliados no processo de aprendizagem (a figura 5.40 ilustra as relações apresentadas).

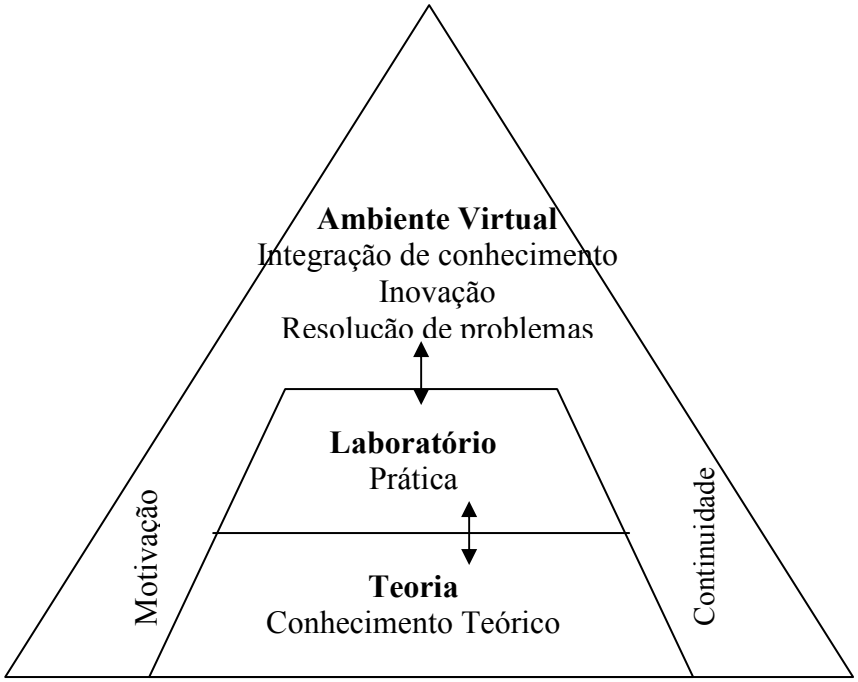


Figura 5.40 - Aspectos de um Ambiente de Ensino e Pesquisa Virtual

5.6.2 **Projeto Kyatera – Laboratório Virtual em Automação**

O Projeto Kyatera está inserido no programa TIDIA, que tem como principal objetivo estabelecer uma rede de fibras ópticas interligando laboratórios de pesquisa, desenvolvimento e demonstração de tecnologias da Internet. Dentre os objetivos deste projeto concernem a implantação de laboratórios controlados através da Internet para o desenvolvimento de experimentos remotos na área de automação da manufatura e robótica a o WebLab.

Os laboratórios virtuais mostram-se como uma solução para a parte prática do processo de aprendizagem, pois oferecem a possibilidade de interagir, criar e/ou modificar novas informações e conteúdos em ensino e pesquisa de Engenharia na área de Automação Industrial. D'Abreu (2002) afirma que laboratórios de simulação de uma linha de produção propiciam uma vivência de ambiente de fábrica dentro de um laboratório da universidade.

Queiroz (1998) classifica estes laboratórios remotos segundo o nível de interação entre usuário e o ambiente de acordo com três tipos:

- **Nível hipermídia:** laboratórios que apresentam nível de interação entre o usuário e o ambiente remoto baixo, permitindo somente à captura de informações que se encontram distantes do usuário (textos, imagens ou vídeo sobre o conteúdo didático a ser ensinado).

- **Nível simulação:** laboratórios que apresentam um maior nível de interação com o usuário, fornecendo a capacidade de fazer uma simulação das experiências que seriam realizadas no laboratório real, onde o usuário envia dados para o experimento previamente implementado e recebe imagens, gráficos e/ou áudio, de acordo com o processo. Na maioria das vezes o laboratório virtual de simulação está diretamente relacionado ao de hipermídia, pois simula a parte prática de um estudo teórico. A parte teórica é apresentada no formato de hipermídia e a simulação é considerada um complemento ao conteúdo didático.

- **Nível tele-presença real:** são laboratórios capazes de interagir com o ambiente remoto e realizar experimentos reais sobre o assunto a ser estudado, havendo a existência de um laboratório físico real que constitui o laboratório virtual por meio de uma camada de abstração de entrada e saído (I/O) e o meio de telecomunicação.

O Laboratório de Automação Integrada e Robótica da Faculdade de Engenharia Mecânica da UNICAMP, é um dos participantes deste projeto e têm como principal objetivo a implementação de um laboratório virtual em Automação baseado na Plataforma LABVIEW, permitindo ao usuário a utilização colaborativa multi-usuários de plataformas didáticas disponibilizadas em rede cooperativa de alta velocidade nas instituições EESC-USP, UNESP e UNICAMP, permitindo assim a capacitação na área através de experimentos remotos e/ou virtuais (figura 5.41).

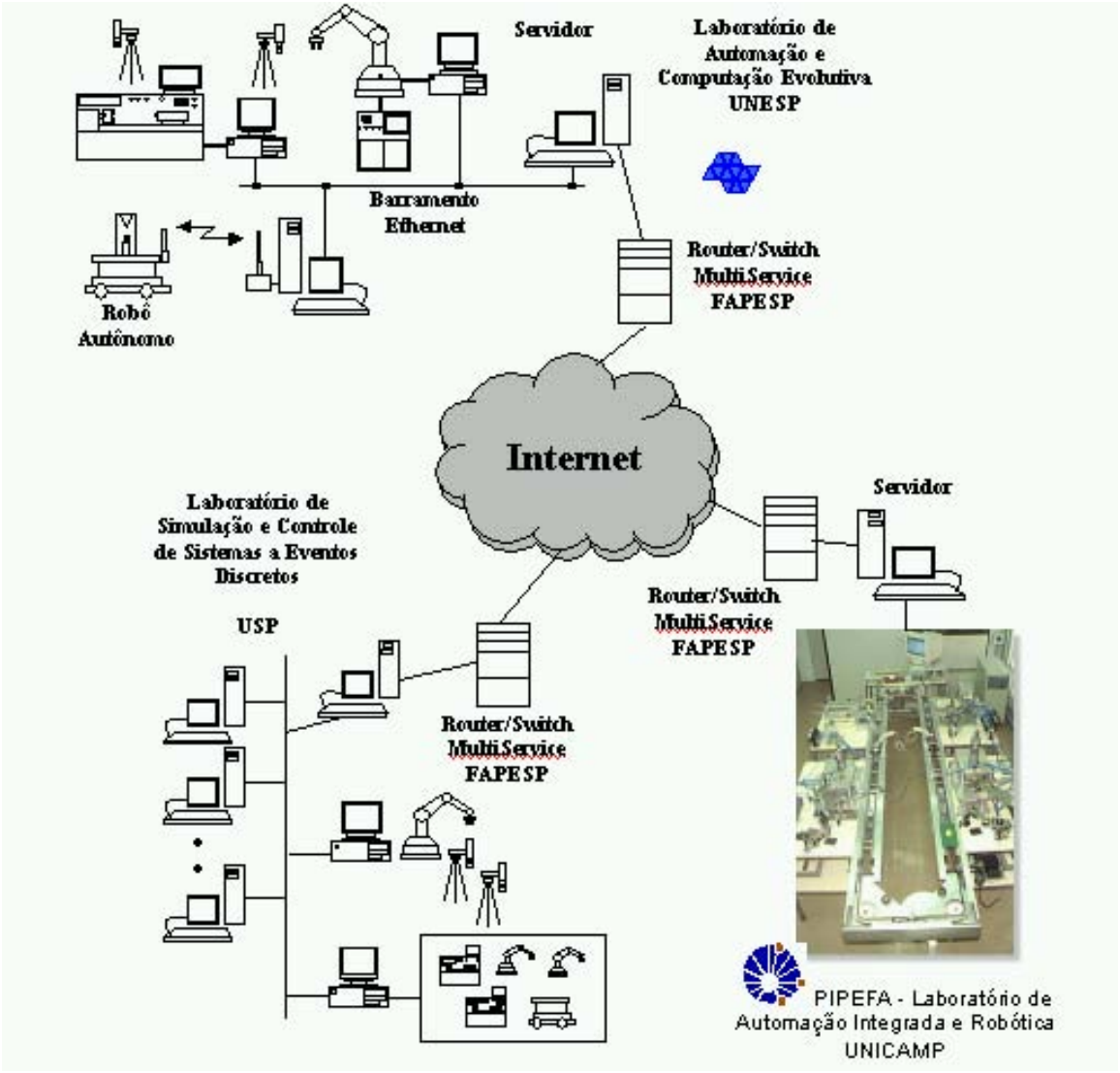


Figura 5.41: Arquitetura proposta para funcionamento do WebLab.

5.6.3 **Arquitetura WEB proposta para o projeto Kyatera**

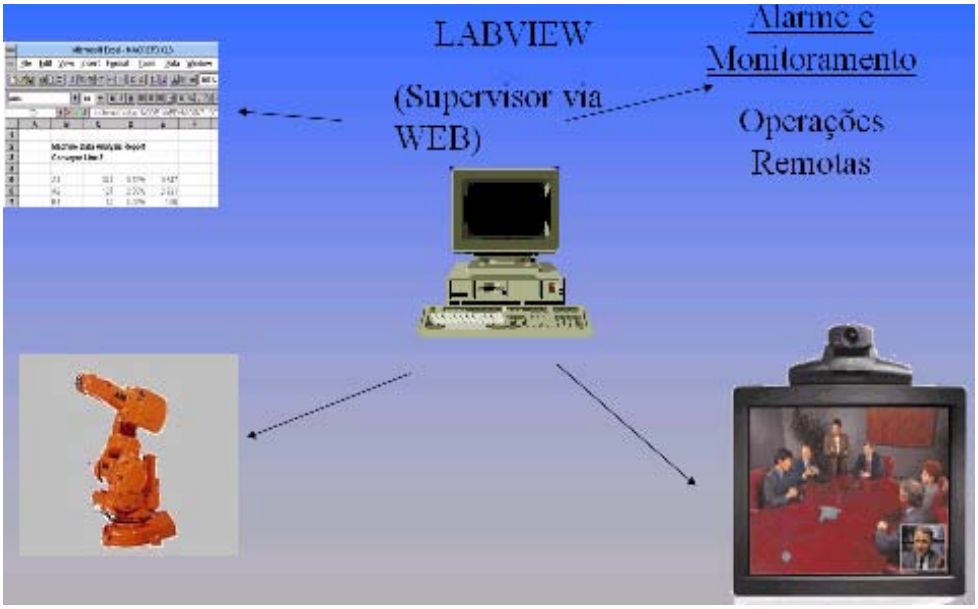
A figura 5.42a apresenta a rede cooperativa implementada a partir da utilização de um robô industrial através de sistema de supervisão remota controlada via WEB. Esta célula é composta de um robô industrial (ABB IRB 1400^{TM}) e um CLP industrial (KOYO^{TM}) que se comunica com o software LABVIEW^{TM} através de configuração OPC Server. Ao mesmo tempo

esta sendo utilizada uma WEBCAM com IP fixo, permitindo assim sua fácil visualização via WEB.

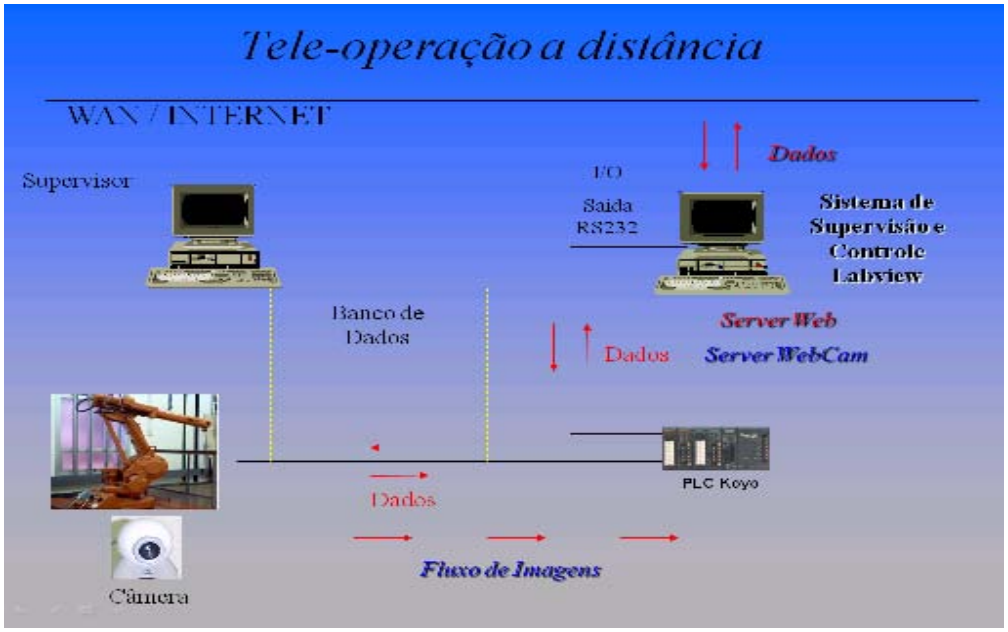
Uma das etapas principais desse trabalho de pesquisa foi a realização de uma série de testes envolvendo a integração do software LabVIEW com dispositivos externos (figura 5.42b), dentre eles:

a) Visualização e controle da WEBCAM com o LabVIEW,
b) Comunicação de variáveis (VI) da CLP Koyo com o software de supervisão e controle implementado em LabView, realizados através do aplicativo OPC Server,
c) Publicação de VI do LabVIEW na rede do Laboratório e externa.

A seguir são descritos detalhes da implementação a partir das figuras 5.43, 5.44, 5.45 e das especificações funcionais e variáveis utilizadas no supervisor implementado. Os programas implementados na CLP Koyo são apresentados em anexo D.



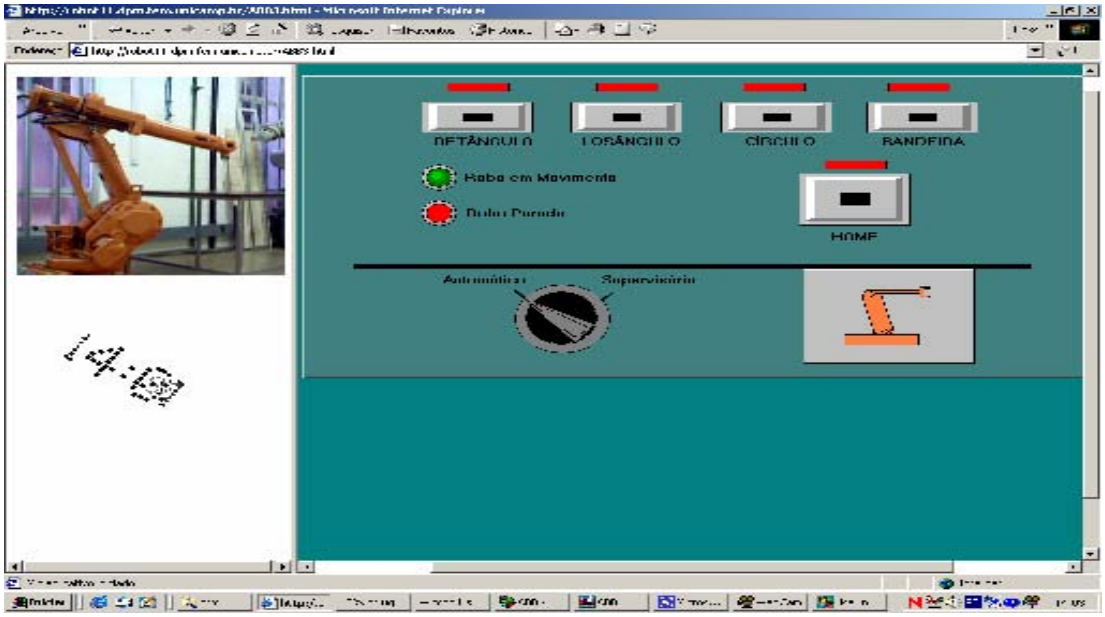
a) Proposta de Aplicação WEB.



b) Arquitetura de Controle e Supervisão Remota

Figura 5.42 – Proposta de Implementação – Projeto Kyatera.

a)



b)

Figura 5.43 – Proposta de pagina HTML - Painel de Controle Implementado em LabView.

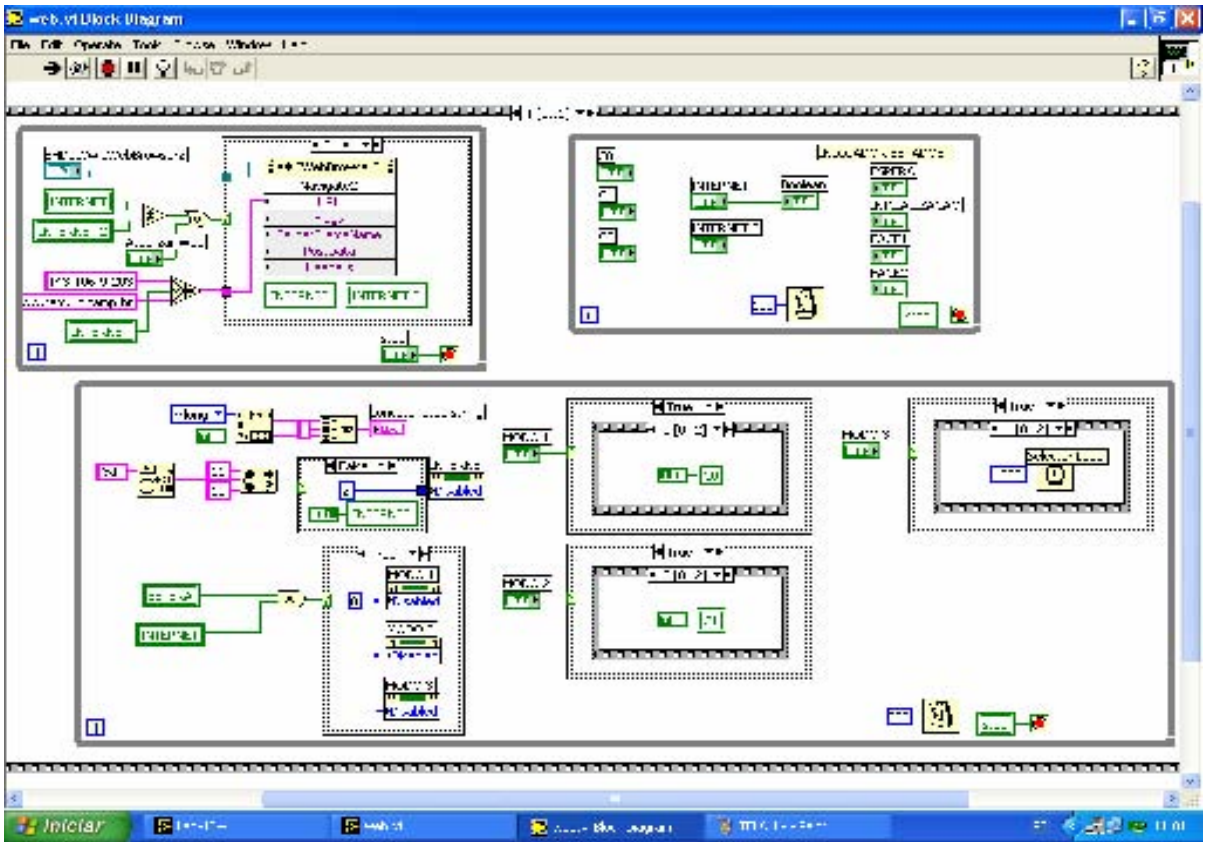


Figura 5.44 – Tela típica de Implementação em LabVIEW.

Figura 5.45 – Tela de Visualização de Câmera CCD WEB CAM com IP fixo.

5.7 **Conclusão**

Este capítulo descreveu cinco exemplos industriais implementados no Laboratório de Automação Integrada e Robótica da UNICAMP utilizando-se a ferramenta LabVIEW^{TM}, que permitiram a validação experimental dos conceitos apresentados nessa dissertação. Através dos conceitos de prototipagem rápida em Mecatrônica, podemos observar que em aplicações diferentes, a metodologia apresentada tem muitas semelhanças, facilitando a sua implementação rápida e com custo final reduzido. Contudo, cada sistema tem suas particularidades no que se refere ao local de operação, sistema de supervisão e controle e modelo de interface com o usuário. Estas particularidades inerentes a cada sistema são uma constante na utilização e implementação de técnicas de prototipagem rápida em mecatrônica, onde a elevada adaptabilidade do sistema proposto esta condicionada aos requisitos de cada projeto.

Capítulo 6

Conclusões e Perspectivas Futuras

6.1 **Conclusões Gerais**

Através do rápido avanço das tecnologias e o crescente conceito de integração de sistemas, criou-se uma necessidade no mercado tecnológico de reduzir o tempo de implementação de um projeto, seja ele voltado a uma melhoria em algum processo de produção ou a concepção de um produto acadêmico. O conceito estendido de prototipagem rápida elimina etapas, reduzindo custos e tempo de implementação de um projeto.

Na área de controle e instrumentação industrial, torna-se imprescindível a implementação de ferramentas que possibilitem a simulação e a rápida implementação da metodologia aplicada, validando assim o projeto na prática, que pode ser extremamente útil na área de desenvolvimento industrial, para a validação de estudos e comprovação de sua eficiência.

Assim, considerando-se o atual cenário tecnológico e o alto desenvolvimento de soluções cada vez mais integradas para concepção e validação de dispositivos mecatrônicos, a utilização de ferramentas de prototipagem rápida para Instrumentação e Controle Industrial torna-se uma indispensável opção para agilizar e facilitar a implementação de diferentes arquiteturas de controle, possibilitando assim a redução do tempo de finalização de um projeto e ou estudo cientifico, atendendo uma eficiência desejada.

Este trabalho teve como principal objetivo a utilização de ferramentas para implementação de novas arquiteturas de controle para sistemas mecatrônicos a fim de torná-los mais precisos e eficientes na execução de tarefas, aliando conceitos de prototipagem rápida para a implementação de controladores. As novas tecnologias envolvendo a concepção de microprocessadores, interfaces de potência e comunicação, aliados ao conceito de prototipagem rápida possibilitam pesquisas científicas, onde a implementação de controladores requer menores custos.

Diante do apresentado, a motivação encontrada no desenvolvimento deste trabalho de pesquisa deve-se ao fato da necessidade de uma metodologia para a concepção e implementação de projetos na área de mecatrônica, visando permitir de forma mais rápida e com significativa redução de custos, a utilização mais adequada das tecnologias que compõem a área.

O principal objetivo do presente trabalho foi a apresentação de metodologias de Prototipagem Rápida de Dispositivos Mecatrônicos com ênfase em Instrumentação Virtual, através de ambiente voltado à capacitação e desenvolvimento de projetos na área de automação industrial, onde os principais conceitos possam ser verificados e posteriormente implementados e validados na prática, fornecendo subsídios para a análise e estratégias para concepção destas aplicações. Esses procedimentos de prototipagem rápida para a concepção de dispositivos mecatrônicos baseados em instrumentação virtual permitem a realização de grande parte do projeto dentro de um ambiente de instrumentação virtual, diminuindo tempo de projeto e custos envolvidos durante a sua fase de concepção. Assim, este trabalho foi desenvolvido a partir das seguintes etapas:

No Capítulo 1 desta dissertação de mestrado apresentou um panorama geral do trabalho desenvolvido, através de um estado da arte do conceito Instrumentação Virtual, com ênfase na área de Mecatrônica, justificando o desenvolvimento desse projeto de pesquisa, seus objetivos gerais e específicos, a necessidade de capacitação nesta importante área.

No Capítulo 2 foram apresentados conceitos de prototipagem rápida nas áreas de engenharia mecânica, eletrônica, células flexíveis de manufatura e instrumentação virtual assim como também as principais etapas que envolvem este importante conceito para o desenvolvimento de arquiteturas de controle mais eficientes, apresentando custo e tempo de implementação reduzida

No Capítulo 3 foi feita uma Descrição de Elementos de um Sistema Automatizado e apresentadas algumas Ferramentas de Modelagem.

No Capítulo 4 descreveu o ambiente LabVIEW^{TM} utilizado para instrumentação virtual e principalmente como ferramenta para implementação de controle de dispositivo mecatrônicos, sendo descritas suas principais ferramentas para controle de processos e outros tipos de interação possíveis com o sistema em estudo.

Finalmente, no Capitulo 5 dedicado a validação do trabalho, foram apresentados alguns exemplos industriais implementados no Laboratório de Automação Integrada e Robótica da UNICAMP utilizando-se a ferramenta LabVIEW^{TM} , demonstrando pelos resultados obtidos que o ambiente escolhido foi eficaz e eficiente para a solução proposta entendida a ambientes colaborativos que poderão ser utilizados em aplicações educacionais (laboratórios virtuais ou weblabs).

Finalizando, no último capítulo deste trabalho, foram apresentadas as conclusões e perspectivas pertinentes ao desenvolvimento deste trabalho, bem como o uso de novas ferramentas do ambiente que justificam ainda mais a escolha do mesmo para soluções com prototipagem rápida.

6.2 Próximas etapas e Perspectivas Futuras

Todos os recursos utilizados para o desenvolvimento desta dissertação de mestrado, por estarem envolvidos com a área de controle e instrumentação industrial, que por sua vez está em constante evolução, requerem o conhecimento multidisciplinar reunindo informações e conhecimento de diferentes áreas. A tendência destes recursos é que haja a inclusão de novos conhecimentos pois uma das características destes elementos é a capacidade de incorporação de técnicas e tecnologias.

Os resultados obtidos nesse trabalho permitem concluir que a utilização de prototipagem rápida embora necessite de uma equipe de trabalho com muita experiência em diversas áreas de atuação dentro da Mecatrônica, se mostra muito eficiente, com redução significativa de tempo de projeto e custos inerentes a confecção de protótipos.

Assim, principal contribuição deste trabalho de pesquisa foi apresentação de ambientes para auxílio no desenvolvimento de protótipos de sistemas embarcados, e um estudo completo de concepção, desenvolvimento, e validação experimental de controladores de juntas robóticas, visando à redução significativa de custos e tempo de implementação, através da utilização mais adequada das tecnologias que compõem a área.

São aspectos promissores do ambiente proposto que sugerem uma possível continuidade desse trabalho de pesquisa enfatizando aspectos relacionados a:

- Flexibilidade: estudo de outras possíveis configurações para implementação de soluções mais aberta para diversos problemas associados com desenvolvimento de protótipos de sistemas móveis embarcados.
- Adaptação a tarefas: apenas os recursos necessários a execução de uma tarefa são utilizados. Sistemas mais simples são tratados com versões mais simples do ambiente proposto, evitando-se perda de recursos.
- Implementação das plataformas existentes em redes de comunicação de alta velocidade;
- Ambiente aberto: permitir que seus blocos componentes sejam totalmente acessados e eventualmente modificados para agregar novas soluções.
- Facilidade de expansão: novos recursos podem ser facilmente adicionados a um projeto, permitindo a adição de novas funcionalidades.
- Desenvolver interfaces para operação de sistemas embarcados cooperativos, como o caso de vários robôs móveis operando em conjunto para um determinado objetivo.

Referências Bibliográficas

Acar, M. Mechatronics Engineering Education in the UK. In: Joint Hungarian British International Mechatronics Conference, 1994, Budapeste. *Proceedings*... Budapest: Computational Mechatronics Publ., 1994, p 763- 769.

Adam, Nabil R., Dogramaci, Ali., Current issues in computer simulation. New York : Academic, 1979. 292 p.

Aihara, C. K., *Projeto e Implantação de Plataforma Didática aplicada ao Ensino e Pesquisa em Automação*, Campinas. Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2000, 104p. Tese (Mestrado).

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

Aihara, C. K., Rosário, J.M., Desenvolvimento de Metodologias Aplicadas ao Ensino e Pesquisa em Automação, In: *World Congress on Engineering and Technology Education, WCETE2004,* Santos, São Paulo, Março 2004.

Aihara, C. K., Rosário, J. M., A Methodology Proposal For Teaching And Research In Automation Using Computer Mediatededucation Tool, In: *11th IFAC Symposium on Information Control Problems in Manufacturing INCOM2004*, Salvador, Brazil, Abril 2004.

Araujo, Emerson dos Santos, Modelagem e Descrição da Parte Comando de um Sistema Automatizado de Produção utilizando o GRAFCET - Aplicado à uma Plataforma Industrial em Automação. Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1997, 91 p. Tese (Mestrado).

Battesini, M., Sistemas Produtivos, Disponível em: www.producao.ufrgs.br/webgrad/ENG09014/SP_II_C1.pdf. Acessado em novembro de 2003.

Bosnardo, R. C., Um sistema de videoconferência par educação à distância baseado em padrões abertos. Dissertação de mestrado. UNICAMP. Faculdade de Engenharia Elétrica, 2001.

Brams G. W. Réseaux de Petri: "Théorie et pratique, tomos 1 et 2, Masson Editions, 1983.

Cardoso, Janete, Valette, Robert, Redes de Petri, Florianópolis: Editora da UFSC, 1997, 212 p.

Cassandras, Christos G., Discrete event systems: modeling and performance analysis. Burr Ridge : Irwin, 1993, 790p.

Chang de Zhang and Shin-Min Song. Forward kinematics of a class of parallel (Stewart) platforms with closed-form solutions. Journal of Robotic Systems, 9(1), February 1992.

Cong-Xin Li Min-Jie Liu and Chong-Ni Li. Dynamics analysis of the Gough-Stewart platform manipulator. Journal of Robotic Systems, 9(1), February 1992.

Coriat, B., Automação Microeletrônica e Competitividade: Tendências no Cenário Internacional. In, *Automação, Competitividade e Trabalho: A Experiência Internacional.* Schmitz, H. & Carvalho, R. Q., Humanismo, Ciência e Tecnologia "Hucitec", São Paulo, 1988.

Cosso, S. G., Integração de Ferramentas de Automação direcionadas à Aplicações de Telerobótica - Implementação de um Sistema de Supervisão e Controle num Sistema Teleoperado, Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2002. 134p. Tese (Mestrado).

Coutinho, L, A F.: "Um Ambiente Integrado de Desenvolvimento de Software a Robótica", mestrado, UNICAMP, Setembro de 1993.

Cruz, J. M.: “Projeto e Desenvolvimento de um Sistema de Geração Automática de Trajetória para Manipuladores”, UNICAMP, mestrado, Outubro de 1993.

D’Abreu, J. V. V., Integração de Dispositivos Mecatrônicos para Ensino-Aprendizagem de Conceitos na Área de Automação, Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2002. Tese (Doutorado).

D’Abreu, João Vilhete Viegas, Construção de um Traçador Gráfico para fins Educacionais, Campinas, Faculdade de Engenharia Elétrica e Computação , Universidade Estadual de Campinas, 1994. Tese (Mestrado).

Damásio, D. Tecnologia Educacional. Disponível em: http://www.superobra.com.br/admin/news.asp?ID_New=1180&Pag=all_news.asp&ID_Sessao_New=1&ID_ANew=11. Acessado em: Outubro de 2004.

Daumas, M., Las grandes Etapas Del Progresso Técnico. Fondo de Cultura Econômica, p. 63-123, México, 1983.

Demongodin, I., Koussoulas, N. T., Differential Petri Nets: Representing Continuous Systems in a Discret-Event World. IEEE Transactions on Automatic Control, vol. 43, n. 4, pp 573-579., 1998

Dias, C. H.: “Implementação de um Supervisor de Controle para Robôs Industriais”, mestrado , UNICAMP, Junho de 1993.

Eby, F. História da Educação moderna. 2. ed, Porto Alegre: Globo, 1976.

Ferreira, Edson P., *Robótica Básica Modelagem de Robôs*, R. Vieira Gráfica e Editora Ltda. Versão Preliminar Publicada para a V Escola Brasileiro-Argentina de Informática, Rio de Janeiro, 1991.

Frigotto, G., As mudanças tecnológicas e Educação da Classe Trabalhadora: Politécnia, Polivalência ou Qualificação Profissional?, In: Trabalho e Educação, p. 45-52, Papirus, Campinas, SP, 1992.

Fayan, B. L.: "Estudo e Especificação de um Supervisor de Controle para um Robô Industrial", mestrado, UNICAMP, Dezembro de 1992.

Groover, M., et al. Robótica: Tecnologia e Programação. McGraw-Hill, São Paulo, 1988.

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

Guimarães, E. et al. Real: A Virtual Laboratory for Mobile Robot Experiments, IEEE Transactions on Education, vol. 46 n.1 February, 2003.

Hartley, S. et al. Enhancing teaching using the Internet: report of the working group on the World Wide Web as an interactive teaching resource. In: Conference on Integrating Technology into computer Science Education, Barcelona, 1996. Disponível em: http://doi.acm.org/10.1145/237466.237649. Acessado em agosto de 2002.

Heer, D., Traylor, R.L., et al. Enhancing the Freshman and Sophomore ECE Student Experience Using a Platform for Learning™. In: IEEE Transactions on Education, vol 46, n. 4, p. 434-446, novembro 2003.

Hervella, C.: "Projeto e Desenvolvimento de um Sistema de um Controlador Programável Flexível para Manipuladores e Robôs Industriais", mestrado, UNICAMP, maio de 1995.

Huber P., Jensen K., Shapiro R. M.; "Hierarchies in Coloured Petri Nets"; In: G. Rozenberg (ed.); Advances in Petri Nets 1990; Lecture Notes in Computer Science, vol. 483, Springer, 1990, pp. 313-341 e em "High-level Petri Nets - Theory and Application"; K. Jensen, G. Rozenberg (Eds.); Springer-Verlag; 1991; pp. 215-243

Jensen, K. An Introduction to the Practical Use of Coloured Petri Nets. In: Lecture Notes on Computer Science n. 1492, p. 237-292, 1998. Disponível em: http://www.daimi.au.dk/~kjensen/papers_books/rec_papers_books.html Acessado em fevereiro de 2003.

Jocarly, P. S.: "Procedimento Automático para Aquisição e Tratamento do Movimento de um Robô, mestrado, UNICAMP, Dezembro de 1992.

LDB, Decreto no.2494 de 10 de fevereiro de 1998. Regulamenta o Art. 80 da LDB (Lei no. 9394/96). Disponível em: http://www.mec.gov.br/sesu/ftp/dec_2494.doc. Acessado em 2000.

Lepkison, H.A., Qualidade da Manufatura Assegurada por Computador, In: Manufatura Integrada por Computador: Contexto, Tendências Técnicas, org. Marilia Markus & Pyrano P. Costa Junior, Fundação CEFETMINAS, Belo Horizonte, 1995.

Loyolla, W., Prates, M., Educação a Distância Mediada por Computador (EDMC) – Uma Proposta Pedagógica, Revista Brasileira de Educação a Distância, v.5, n.29, p. 3-18, 1998.
Machado, S.R.L. Mudanças tecnológicas da classe trabalhadora, in Trabalho e Educação, Campinas, p. 9-25. Papirus, 1992.

Luc Baron and Jorge Angeles. The direct kinematics of parallel manipulators under joint-sensor redundancy. IEEE Transactions on Robotics and Automation, 16(1), February 2000.

Luc Baron and Jorge Angeles. The kinematic decoupling of parallel manipulators using joint-sensor data. IEEE Transactions on Robotics and Automation, 16(6), December 2000

Manacorda, M. A.,História da Educação, 3 ed. São Paulo, Cortez, 1992.

MathWorks Inc. MatLab - User's Guide.

Matos filho, M. V., Santos, W. R., Redes de Petri. Disponível em: http://www.unigran.br/biblioteca/producaointelectual/rede_de_petri.pdf . Acesso em 2004.

Melo, J.J.L, Sobreira P.L., Petri Net. Disponível em: http://www.jameson.hpg.ig.com.br/introd.html, Acessado em agosto de 2003.

Meneghel, L., Desenvolvimento de Laboratórios Virtuais para o ensino fundamental e o ensino superior., Campinas, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, 2003. Tese (Mestrado).

Meriam, Kraige: Dinâmica, LTC, 1996.

Miranda, M. F.: "Controle de um Servomecanismo por um microcomputador Dedicado: Uma contribuição ao Estudo de Controladores para Robôs Industriais", mestrado, UNICAMP, Agosto de 1992.

Moore, M.G., Kearsley, G., Distance Education: a system view. Belmont: Wadsworth Publishing Company, 1996.

Moore, M. G. Theory of transactional distance. In: KEEGAN, D. (ed) Theorical principles of distance education. London and New York: Routledge, 1998.

Morcelli, João Carlos de Moraes, Simulador Seqüencial de sistemas de Filas. Disponível em: http://www.inf.pucpcaldas.br/~morselli/ , Acessado em Agosto de 2004.

Monroe, P., História da Educação, 9 ed. São Paulo, Companhia Editora Nacional, 1970.

M. Shahinpoor. Kinematics of a parallel-serial (hybrid) manipulator. Journal of Robotic Systems, 9(1), February 1992.[8] Irene Hyna Tobias Oetiker, Hubert Partl and Elisabeth Schlegl.

Murata, T. Petri Nets: Properties, Analysis ans Applications; Proceedings of the IEEE, vol. 77, nº 4, Abril 1989, pp. 541-580.

Nascimento, R. B., Trompieri Filho, N., Correio Eletrônico como recurso didático no ensino superior – o caso da Universidade Federal do Ceará, In: Ccia da Informação v.31, n.2, p.86-97, maio/agosto de 2002.

National Instruments. LabVIEW - User Manual, July 2000.

National Instruments, *Ni Developer Zone*: Virtual Instrumentation for Test, Control, and Design. Disponível em: <http://zone.ni.com/devzone/conceptd.nsf/webmain/E55C5FC8AC3111DE86256FEE004065AD?OpenDocument&node=201805_US>. Acesso em 25 jan. 2006.

National Instruments, *Ni Developer Zone*: Software's Role in Virtual Instrumentation. Disponível em: <http://zone.ni.com/devzone/conceptd.nsf/webmain/E7F7F3EA19D776F686256FEE0040C455>. Acesso em 15 mai. 2006.

National Instruments, *Ni Developer Zone*: Hardware's Role in Virtual Instrumentation. Disponível em: <http://zone.ni.com/devzone/conceptd.nsf/webmain/AB848C738E0DD42086256FEE0040E85F>. Acesso em 10 jun. 2006.

National Instruments, *Ni Developer Zone*: Virtual Instrumentation and Traditional Instruments. Disponível em: <http://zone.ni.com/devzone/conceptd.nsf/webmain/446E90868BEE72DD86256FEE003FF6F4>. Acesso em 10 jun. 2006.

Nélis, J., Laloup, J. Homens e máquinas, Herder, São Paulo, 1965.

Nogueira, R. G.: “Controle de Posição e Orientação de um Manipulador através de um Mouse Espacial”, mestrado, UNICAMP, março de 1995.

Nunes, I.B., Noções de educação a distancia. Disponível em: http://www.intelecto.net/ead/ivoniol.html . Acesso em 2002.

O'Brien, As Máquinas, Livraria José Olympio Editora, Rio de Janeiro, 1964.

Otsuka, J.L., Fatores determinantes na efetividade de ferramentas de comunicação mediada por computador no ensino à distância. 1996. Trabalho de conclusão de curso. INSTITUTO DE Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre. Disponível em: http://penta2.ufrgs.br/pesquisa/joice/joice.ti.html. Acessado em agosto de 1998.

Peters, O., Didática do Ensino à Distância, São Leopoldo, editora Unisinos, 2001.

Peterson, J. L. Petri Net Theory and the Modeling of Systems; Prentice-Hall, Inc.; 1981

Queiroz, L. R. Um laboratório virtual de robótica e visão computacional. Campinas, Instituto de Computação, Universidade Estadual de Campinas, 1998. Tese (Mestrado).

Quartiero, E.M., As tecnologias da informação e comunicação e a educação. Revista Brasileira de Informática na Educação, n.4, junho de 2000. Disponível em: http://www.inf.ufsc.br/sbc_ie/revista/nr4/

Queiroz, L.R., Um Laboratório Virtual de robótica e visão computacional, Dissertação de Mestrado, UNICAMP, Instituto de Computação, Campinas, 1998.

Ramadge, P. J. Wonham, W.M. "The Control of Discrete Event Systems. Proceedings of the IEEE, v. 77, n. 1, 1989

Rezende, F., Tecnologia e Educação, Curso de Pós-Graduação em Docência do Ensino Superior, UFRJ, Rio de Janeiro, 2000.

Rosário, J. M. Princípios de Mecatrônica, São Paulo: Pratice Hall, 2004

Rosário, J.M.; Messina, L.P.C.; Aust, E.: "Development of Supervisory Control of Advanced Robots for Underwater Tasks", Mechatronics'96, Portugal, 18-20/08/96.

Rossi, W. G. Capitalismo e educação. 2 ed. São Paulo, Moraes, 1980.

Sá, C. A: "Implementação de Algoritmos Numéricos para a Resolução do Problema Cinemático Inverso de Robôs", mestrado, UNICAMP, agosto de 1996.

Saramago, M. P. A., Integração de dispositivos inteligentes utilizando conceitos de domotica direcionados a automação hospitalar, Campinas. Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2002, 224p. Tese (Doutorado).

Saramago, M. A P.: "Projeto e Desenvolvimento de um Sistema de Calibração e Medida de Precisão para Robôs Industriais", UNICAMP, junho de 1993.

Segnini, L. R. P., Controle e Resistência nas Formas de uso das Forças de Trabalho em Diferentes Bases Técnicas e sua relação com a Educação, In: Trabalho e Educação, p.59-68. Campinas, Papirus, 1992

Silva Junior, E. N. et al., Novas tecnologias para educação no Amazonas. In: Congresso da Rede Iberoamericana de informática educativa Laboratório de Tecnologias Cognitivas, 1998, Brasília. Disponível em: http://www.c5.cl/ieinvestiga/actas/ribie98/205.html. Acesso em 2000

Silveira, P. R., Santos, W. E., Automação e Controle Discreto, São Paulo, Érica, 2ª. edição, 1998, 229p.

Soares, Luiz Fernando Gomes, Modelagem e Simulação Discreta. São Paulo, IME_USP, 1992, 254p.

Sousa, Flavia M. G. *Controle de fresadora para a prototipagem de circuitos impressos*. M.Sc. thesis, State University of Campinas, 1998.

Timm, A., Pequenã historia de la tecnologia. Madrid, Ediciones Guadarrama, 1971.

Traylot, R. L., Heer, D., Fiez, T. S., Using an Integrated Platform for Learning™ to Reinvent Engineering Education, In: IEEE Transactions on Education, vol 46, n. 4, p. 409-419, novembro 2003.

Vargas, M., Educação Tecnológica: Desafios e Perspectivas, org. Mirian P.S.Z. Grinspun, p. 3-14, Editora Cortez, São Paulo, 1999

Vygotsky, L. S., Pensamento e Linguagem, Martins Fontes, São Paulo, 1991.

White, M. A., Ensino à Distância. Disponível em: www.penta2.ufrgs.br/edu/telelab/pavani/fundamen.htm. Acessado em agosto de 2002.

Zhang Lanfang Wu Jiangning and Li Shilun. Posture measurement and structural parameters calibration on parallel six DOF platform. (Reference Internet) .

ANEXO I

Conceitos Básicos de Sistema de Visão Automatizado

A.1 Critérios para escolha de um Software de Visão Industrial

A utilização de um sistema de Visão Automatizada esta cada vez mais freqüência na área de Mecatrônica, e muitos fabricantes vem desenvolvendo sistemas dedicados para visão de máquina e processamento de imagens. Estes pacotes contêm um grande numero funções de visão, que poderão ser implementadas em outros softwares de Instrumentação Industrial, tais como o LabVIEW da National Instruments, que já possuem funções dedicadas para diferentes aplicações industriais tais como sistemas de inspeção, alinhamento, identificação e medição visual. Estes ambientes possuem interatividade para configurar, testar e distribuir aplicações de visão sem muita programação, trabalhando diretamente com muitas interfaces de aquisição de imagem disponíveis no mercado.

Considerando a rapidez e eficiência na sua manipulação e implementação, alem da inúmera quantidade de recursos disponíveis nestes sistemas, que cobrem grande maioria das aplicações na área de Mecatrônica estes sistemas de Visão Automatizada são cada vez mais utilizados nesta área. A escolha de um software de visão adequado para uma dada aplicação devera seguir as recomendações seguintes:

a) Escolha da Câmera,
b) Fator de Escala do Hardware,
c) Software de simples utilização,
d) Abrangência e exatidão do algoritmo implementado,
e) Desempenho do algoritmo implementado,
f) Facilidade de integração com outros dispositivos,
g) Custo-Beneficio envolvido,
h) Parceiros e Integradores,

i) Suporte Técnico,
j) Crescimento e Estabilidade da Empresa

A.2 **Sistema de Visão da National Instruments**

A National Instruments pode ser considerada um dos líderes do mercado em visão de máquina e processamento de imagens. O software NI Vision está disponível em dois pacotes:

a) NI Vision Development Module: contém centenas de funções de visão para serem utilizadas com o LabVIEW da National Instruments, NI LabWindows/CVI, C/C++ ou Visual Basic para programar poderosas aplicações de inspeção, alinhamento, identificação e medição visual.
b) NI Vision Builder for Automated Inspection (AI): é um ambiente interativo para configurar, testar e distribuir aplicações de visão sem programação. Ambos os pacotes de software trabalham com todas as placas de aquisição de imagem da National Instruments e com o NI Compact Vision System.

A.3 **Características do Sistema de Visão e o software da NI**

A.3.1 **Escolha da Câmera**

A primeira consideração ao selecionar um software de visão é determinar se este trabalha com a câmera que melhor atende aos requisitos de sua aplicação. É fácil encontrar câmeras analógicas de baixo custo, mas, frequentemente uma aplicação necessita de uma resolução maior que a VGA, taxas de aquisição maiores que 30 frames/s e maior qualidade de imagem do que as câmeras analógicas podem proporcionar. As ferramentas de hardware e software da National Instruments são compatíveis com milhares de câmeras, desde padrões analógicos de baixo custo a linhas de alta-velocidade. Dentro desse software, existe um modulo Assistente para Câmeras Industriais que permite a seleção da melhor câmera para sua aplicação e auxiliar na escolha correta do hardware de aquisição.

Figura A1: Câmeras disponíveis no mercado.

A.3.2 **Fator de escala do Hardware utilizado**

Ao mesmo tempo em que a escolha de uma câmera é um passo crucial numa dada aplicação, o fator de escala da câmera é outra importante aspecto a ser considerado. Devido à tecnologia de câmeras evoluírem rapidamente, torna-se indispensável considerarmos a atualização de câmeras para melhorar a qualidade de imagem ou realizar outros tipos de medição.

No caso do driver NI-IMAQ da National Instruments suporta todas as interfaces para Aquisição de Imagem da National Instruments e realiza comunicação com milhares de câmeras através de uma interface de fácil utilização. Seu programa não sofre alteração se você substituir uma câmera analógica por uma câmera Camera Link. O mesmo acontece com o software NI-IMAQ for IEEE 1394 Cameras, que comunica e adquire imagens de mais de 100 diferentes câmeras IEEE1394 (FireWire) sem a necessidade de uma placa de aquisição de imagens.

Além de o driver da National Instruments suportar milhares de câmeras, este também é compatível com todas as plataformas de hardware da NI, desde PCs e CompactPCI/PXI a sistemas Compact Vision System. Assim, você pode prototipar sua aplicação em um laboratório com um PC e uma câmera IEEE 1394 (FireWire) de baixo custo e depois distribuí-la na linha de produção em um robusto Compact Vision System sem alterar o código de processamento ou aquisição de imagem.

Figura A2: Hardware utilizado.

A.3.3 **Software de simples utilização**

Após realizar a aquisição da imagem, o próximo passo é o seu processamento. A utilização de algoritmos atuais, localizarem a ferramenta correta através de "tentativa e erro" em uma linguagem de programação pode ser tedioso e ineficaz.

Considerando isto, são necessárias ferramentas de software de visão para auxiliar no máximo o aproveitamento dos algoritmos. Em muitas aplicações, você não precisa de uma linguagem de programação para construir um sistema completo de visão de máquina. Embora menos flexível do que uma programação em C, Visual Basic ou LabVIEW, um software configurável como o NI Vision Builder AI proporciona um ambiente interativo de fácil navegação para configurar, testar e distribuir aplicações de visão de máquina.

O Vision Builder AI inclui quase 50 ferramentas comuns para visão de máquina como pattern matching (reconhecimento de padrões), OCR (reconhecimento de caracteres), leitores de código de barras tipo Matriz (DataMatrix), color matching (reconhecimento de cores), e muitos outros. Ele permite também a aquisição de imagens de qualquer câmera suportada pela NI e comunicar os resultados da inspeção com outros dispositivos utilizando protocolos industriais através de rede Ethernet, Serial ou E/S digitais.

Como a programação de uma aplicação de visão muitas vezes é mais complexa do que configurar o Vision Builder AI, a National Instruments torna fácil o desenvolvimento de aplicações em LabVIEW, C, Visual Basic e potencializa-o com o NI Vision Assistant. Incluso

com o NI Vision Develoment Module, o Vision Assistant é um ambiente de prototipagem que você pode interativamente realizar experimentos com diversas funções de visão para testar como estas funcionarão em sua aplicação e quanto tempo cada função leva para executar.

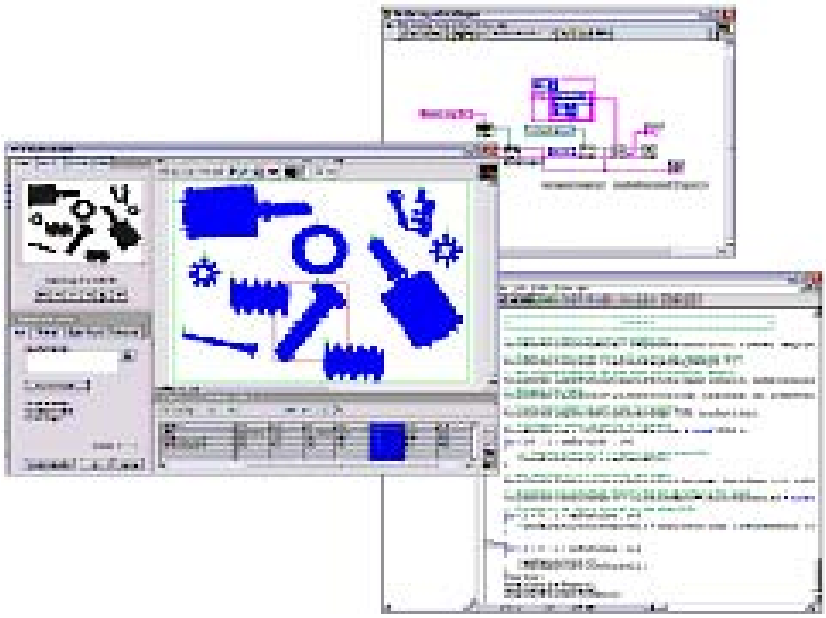


Figura A3: Software Vision Builder para processamento de imagens.

Após ter determinado qual a melhor maneira de solucionar um desafio em sua aplicação, simplesmente clique em um botão e o Vision Assistant gera um código pronto para ser executado em LabVIEW, LabWindows/CVI, C/C++, ou Visual Basic. Você finaliza a maior parte da sua aplicação de visão antes de criar uma linha de código. Você pode executar somente o código gerado pelo Vision Assistant ou adicioná-lo a um grande sistema de controle industrial, aquisição de dados ou controle de movimento.

A.3.4 **Abrangência e Exatidão do Algoritmo**

Existem muitas características a serem consideradas na escolha de um software de visão, entretanto uma das mais importantes é saber se a ferramenta de software pode medir de maneira correta e com exatidão as características do objeto em subpixels. Se o software não é confiável ou

exato, não importa o quão rápido seu computador é ou quantos pixels sua câmera possui. Devemos saber que é muito mais fácil tornar rápido um código preciso do que tornar mais preciso um código que é rápido.

O Vision Development Module e o Vision Builder AI incluem centenas de funções de visão que são precisas e confiáveis. Todas essas funções possuem a exatidão em subpixels para interpolar localizações, distâncias e medições em até um décimo de pixel e um décimo de grau. As cinco áreas mais comuns para aplicações de visão de máquina juntamente com a descrição de algoritmos mais comuns (fig. A3), estão listadas a seguir:

a) **Melhoria de uma imagem:** Utilização de ferramentas de filtro para delimitar bordas, remover ruídos ou extrair informação de freqüência com exatidão. A utilização de ferramentas de calibração de imagem permitira a remoção de erros de não-linearidade e perspectiva causados por distorções da lente ou posicionamento da câmera. As ferramentas para calibração de imagem também poderão ser utilizadas para aplicar unidades de engenharia nas medições, ou seja, transformar os valores em pixels em microns, milímetros ou milhas.

b) **Verificação de presença:** Este é o tipo mais simples de inspeção visual, onde para verificarmos a presença de uma característica, pode-se utilizar qualquer ferramenta de cor, reconhecimento de padrão ou histograma. Uma verificação de presença sempre resulta em um valor sim/não ou passa/falha.

c) **Localização de padrões:** É importante no alinhamento de objetos ou na determinação do deslocamento dos objetos, servindo de padrão para todas as inspeções subseqüentes. Para localizar padrões, estas ferramentas deverão permitir a detecção de: borda, escala de cinza, forma, geometria e padrões de cor. Estas ferramentas retornam a posição do objeto (X, Y) e o ângulo de rotação em até um décimo de pixel. A detecção de geometria é imune à sobreposição de objetos ou objetos que são alterados em escala.

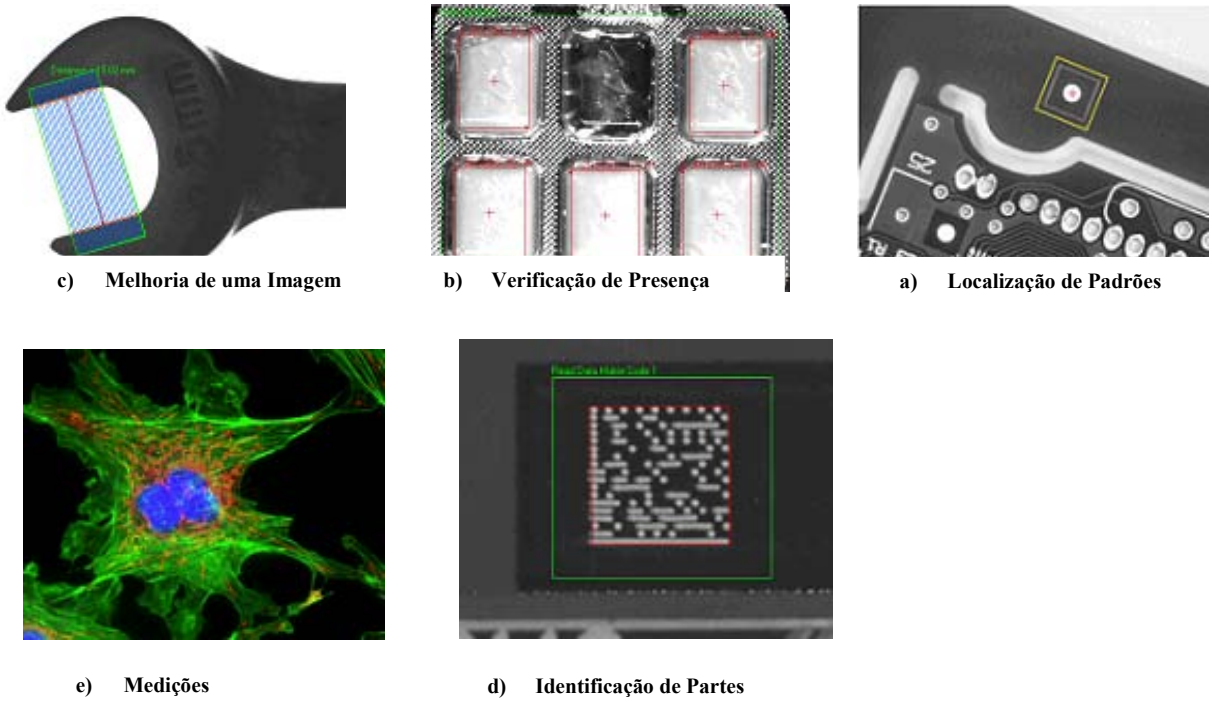
d) **Medição**s: É a razão mais comum na utilização de um sistema de visão, onde normalmente utilizam-se ferramentas com função para detecção de borda, análises de partícula ou geometria para medir distância, diâmetro, quantidade, ângulos e área. Se estivermos calculando o total de

número de células em um microscópio ou o ângulo entre duas bordas de um sistema de freio, estas ferramentas sempre retornam um número ao invés da localização ou um valor passa/falha.

e) **Identificação de partes:** É importante para inspeção de conformidade, rastreabilidade e verificação. Métodos diretos de identificação incluem leitura de código de barras ou dados como DataMatrix e PDF 417. Novos métodos utilizam OCR "treinável" ou classificação de objetos. A Identificação de partes freqüentemente resulta em texto ao invés de uma medição ou uma determinação passa/falha.

A.3.5 **Análise de Desempenho do Algoritmo**

Enquanto a exatidão e a facilidade de uso frequentemente são os dois fatores mais importantes na escolha de um sistema de visão, a velocidade de execução é a terceira consideração. Não importa quantas centenas de algoritmos necessitamos para escolha ou quão rápido você pode construir uma aplicação com eles, se as ferramentas de inspeção são ineficientes e levam muito tempo para executar, muito do seu trabalho será perdido.



c) Melhoria de uma Imagem **b) Verificação de Presença** **a) Localização de Padrões**

e) Medições **d) Identificação de Partes**

Figura A4: Principais aplicações em Visão de Máquinas em Automação.

Características	NI	Líder do Mercado	No de Vezes Aumento Comparativo
Histograma	0.91	2.03	2.2
Transformadas Geométricas	3.1	10.3	3.3
Morfologia	1.8	5.9	3.3
OCR	3.3	5.9	1.8
Detecção de Geometrias	93.0	149.8	1.6
Classificação de Objetos	7.5	–	–

Tabela A1: Comparativo de Velocidade do Software de Visão NI em relação ao Líder do Mercado.

O software de visão da NI é altamente otimizado para ter o melhor desempenho em qualquer possibilidade, resultando em um dos mais rápidos softwares do mercado, conforme mostra tabela comparativa de características de tempo de processamento deste software em relação ao Líder do Mercado (tabela A1).

A.3.6 **Integração com Outros Dispositivos**

Em automação industrial, uma aplicação de visão aplicação de visão esta inserida como uma parte de um grande sistema de controle, podendo necessitar de atuadores para selecionar produtos; comunicar os resultados da inspeção para o controlador de um robô, CLP, ou Controlador Programável para Automação; salvar imagens e dados em servidores de rede; ou comunicar os parâmetros e os resultados de inspeção para uma interface de usuário local ou remota. Freqüentemente, para aplicações em imagens científicas, você deve integrar visão com estágios de movimento, sistemas de aquisição de dados, microscópios, óticos especializados e trigger avançado.

No caso da National Instruments, que pode ser considerada como um dos lideres do mercado em automação, controle industrial, aquisição de dados e controle de movimento, os produtos de visão da NI são projetados para trabalhar com estes e outros componentes comuns. Assim, no caso de uma aplicação que necessitemos comunicar com um CLP através de uma rede DeviceNet ou com um microscópio via barramento Serial, isso será facilmente implementado a partir de um Sistema de Visão da NI.

A.3.7 **Custo-Benefício**

Pacotes de software de visão possuem muitas variações. Muitos fornecem para clientes OEM (Original Equipment Manufacturer) separando suas bibliotecas e vendendo algoritmos individuais. Enquanto cada conjunto destes algoritmos individuais aparenta ter um custo reduzido, o pacote total de um sistema de visão possui um alto custo. Adicionado ao custo de uma licença para cada componente, a distribuição da aplicação se torna complicada e com um custo alto.

No caso do NI Vision Development Module, este produto possui todos os algoritmos que possamos necessitar para solucionar os desafios de visão mais complexos, assim você evita tempo de pesquisa, compra e manutenção de diversos pacotes de software. Assim, a distribuição das aplicações se torna de baixo custo, e através de uma única licença, poderemos distribuir uma versão executável que utiliza qualquer número de algoritmos.

A.3.8 **Parceiros e Integradores**

A National Instruments fornece software e hardware para processamento de imagem e visão de máquina. Pelo fato da NI não fornecer iluminação, câmeras e equipamentos óticos, a empresa trabalha em conjunto com outros especialistas que o fazem.

Através da última década, os produtos de visão da National Instruments têm auxiliado na solução de milhares de desafios em diversas aplicações de visão, desde inspeção em componentes

automotivos a sistemas de pesquisa sobre câncer. Enquanto as ferramentas de visão da NI são projetadas para usuários finais, grandes aplicações podem necessitar da ajuda de um especialista em visão. Para auxiliar no desenvolvimento de sua aplicação, a NI trabalha com muitos parceiros que podem ajudar na seleção dos componentes corretos ou construir uma solução completa.

A.3.9 **Suporte Técnico**

Enquanto o software de visão da NI é projetado para ser de fácil utilização, é importante obter ajuda quando necessário. A NI fornece o software de visão diretamente para seus clientes, oferecendo também um suporte técnico diretos aos seus produtos.

A.3.10 **Crescimento e Estabilidade da Empresa**

Um investimento num software de visão de máquina, deve-se levar em consideração utilizações futuras e necessidade de sempre mantê-lo funcionando. Existem diversas empresas pequenas e especializadas em visão de máquina onde, enquanto suas ferramentas podem funcionar para uma aplicação atual, quando você precisar atualizar uma estação de inspeção em cinco anos, você precisa saber que a empresa e o software ainda existirão e estarão evoluindo. No caso da NI, esta empresa tem um pesado investimento em Pesquisa e Desenvolvimento, e aos poucos tem se estabelecido como uma pioneira em software e hardware para visão, e com certeza que, nos próximos anos, esta empresa continuará a expandir e melhorar seus compromissos com visão de máquina e processamento de imagens.

ANEXO II

Aspectos Experimentais da Implementação da Plataforma de Stewart-Gough

B.1 **Modelagem Matemática**

B.1.1 **Manipuladores Paralelos – Revisão de Conceitos**

Os manipuladores robóticos podem ser classificados em dois tipos: seriais e paralelos. Os manipuladores seriais caracterizam-se por barras unidas por juntas seqüencialmente, isto é, uma barra pode possuir até duas juntas e se ligar diretamente a no máximo duas outras barras, formando ou não um ciclo fechado. Os atuadores podem ser as barras, na forma de cilindros; ou as juntas, na forma de atuadores angulares.

Os manipuladores paralelos têm barras que se ligam diretamente a mais de uma barra, geralmente através de uma estrutura rígida. Pela sua natureza, os manipuladores seriais costumam ter a cinemática direta simples, porém são de baixa precisão, não suportando cargas elevadas. Os manipuladores paralelos possuem características duais às dos seriais.

Ao mesmo tempo podem ser encontradas também algumas estruturas robóticas que procuram aproveitar um pouco das vantagens de cada geometria de construção: os manipuladores híbridos, um estudo interessante pode ser encontrado em M. Shahinpoor (1992).

Existem várias configurações geométricas para manipuladores paralelos, e mesmo para plataformas de simulação e posicionamento. Através da imposição de simplificações geométricas, podem-se obter soluções analíticas para o problema da cinemática direta, como feito em Chang de Zhang and Shin-Min Song (1992), entretanto, suas condições são fortemente restritivas. A seguir serão apresentadas equações matemáticas obtidas a partir da geometria da plataforma, que foram utilizadas para controle de movimentos e calibração.

B.1.2 **Sistema de Posicionamento – Cálculo da distensão dos atuadores**

Com o objetivo de modelarmos geometricamente a Plataforma implementada e determinamos os procedimentos numéricos para o tratamento do problema cinemático e do problema cinemático inverso, para tanto, identificamos no projeto mecânico da base móvel existente os pontos de importância para a determinação completa dos elementos móveis, no caso, cilindros e base.

Notamos que os pontos que determinam a movimentação são as extremidades dos seis cilindros e a solução do problema resume-se a posicionar os cilindros de tal forma que se atinja uma posição fornecida como coordenadas do centro de massa da base superior e ângulos de rotação *roll*, *pitch* e *yaw*, em torno dos eixos coordenados. Sabemos que as posições das extremidades superiores dos cilindros são fixas em relação ao centro de massa da base superior, se arbitramos nesse centro um sistema de coordenadas, determinamos tais posições.

A partir dos parâmetros de movimentação e inclinação, temos novas posições para as extremidades superiores dos cilindros, as quais podem ser obtidas por um procedimento analítico. Os parâmetros de entrada são exatamente os valores que determinam numericamente a matriz de transformação de coordenadas que permite obter as novas posições procuradas.

Com estes dados, denotamos cada cilindro por um segmento orientado de origem na extremidade inferior, que é fixa à base inferior; e esta, no solo, e interpretamos a norma euclidiana deste segmento como o tamanho, mínimo acrescido da distensão atual do cilindro para a posição calculada. É quase desnecessário dizer que a as coordenadas das extremidades inferiores dos cilindros são calculadas pela geometria da base inferior.

Considerando $\mathbf{X_i}$ é o vetor de coordenadas de um ponto, uma extremidade superior de um dos cilindros, e **T** é a matriz de transformação de coordenadas, denotamos por $\mathbf{X_i'}$ o novo vetor de coordenadas da posição almejada (equação B.1). E Assim, conhecendo $\mathbf{X_i}$, poderemos determinar $\mathbf{X_i'}$ (equação B.2).

$$X_i = T \cdot X_i' \qquad \text{B.1}$$

$$X_i' = T^{-1} \cdot X_i \qquad \text{B.2}$$

Logo, se possuímos uma conformação da base superior almejada, cujas coordenadas das extremidades superiores dos cilindros são determinadas pelo procedimento descrito, temos um novo segmento orientado e outra norma representada pelo tamanho almejado de cada cilindro.

A diferença entre o tamanho atual e almejado é a distensão que deve se imposta a cada cilindro para alcançar a nova posição. Deve-se notar que existe um **X** para cada cilindro. Sendo **O,** o vetor de coordenadas da extremidade inferior de cada cilindro, a distensão que se deve solicitar ao cilindro **i**:

$$D = |X_i' - O| - |X_i - O| \qquad \text{B.3}$$

B.1.3 **Modelo de deslocamento de um atuador linear**

O controle de movimentos da plataforma pode ser realizado pela composição de movimentos individuais de cada um dos atuadores. Para isso deduziremos a seguir um algoritmo que posiciona a plataforma de acordo com a movimentação desejada de cada cilindro hidráulico.

Supomos que seja relevante em um caso como este testar apenas se houve a distensão requerida do cilindro em particular; assim a movimentação se dará mantendo a inclinação e atualizando o comprimento. Desta forma, o programa terá como parâmetros os dados sobre a configuração da plataforma, o cilindro a ser controlado e seu novo comprimento; e terá como retorno os dados sobre a nova configuração da mesa.

Seja o cilindro k o que se deseja controlar para um novo comprimento L, P_i a matriz 3×6 com as coordenadas inferiores, em que P_{ji}, sua j-ésima coluna contém as coordenadas inferiores do cilindro j, P_s a matriz com as coordenadas superiores, de constituição análoga. Renomearemos as coordenadas superiores e inferiores do cilindro escolhido como, respectivamente, X_s e X_i. Inicialmente encontramos o centro C da parte superior:

$$C = \frac{1}{6}\sum_{j=1}^{6} P_s^j \qquad \text{B.4}$$

Para obtermos o movimento, será realizado um movimento de rotação da base da plataforma em torno de um ponto O que é a reflexão da extremidade superior do cilindro em questão X_s em torno de C. Se quisermos que o cilindro passe a ter comprimento L, a nova posição da extremidade superior X_{sn}:

$$X_{sn} = X_i + L\frac{(X_s - X_i)}{|(X_s - X_i)|} \qquad \text{B.5}$$

Em seguida, atribuiremos um sistema de coordenadas centradas em O com inclinação conveniente para representar a citada rotação com apenas um ângulo. A base desse sistema representada no sistema original:

$$\vec{n_x} = \frac{(X_s - O) \times (X_{sn} - O)}{|(X_s - O) \times (X_{sn} - O)|}$$
$$\vec{n_y} = \frac{(X_s - O)}{|X_s - O|} \qquad \text{B.6}$$
$$\vec{n_z} = \vec{n_x} \times \vec{n_y}$$

Fazemos M a matriz que tem n_x, n_y, n_z como linhas, α é o ângulo entre X_s - O e X_{sn} - O.

$$\alpha = \arccos\left(\frac{(X_s - O) \cdot (X_{sn} - O)}{|X_s - O||X_{sn} - O|}\right) \qquad 5.7$$

Como n_x foi escolhido perpendicular a X_s - O e X_{sn} - O, a movimentação é uma rotação pura em torno do eixo direcionado por n_x .

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \qquad \text{B.8}$$

R é a matriz de rotação, no sistema representado por M, que transforma X_s em X_{sn} e transformará as outras extremidades superiores, ajustando a posição da mesa. Basta converter as coordenadas subtraindo a nova origem O e multiplicando as posições por M, rotacionar multiplicando por R e reconverter multiplicando por M^T e somando O, para todos os cilindros.

O passo seguinte consiste em fazer a base móvel reproduzir um padrão de movimento, no nosso caso, as ondulações marítimas. Um procedimento a ser implementado na linguagem MATLAB é o que gera os pontos que fornecem a trajetória a partir de uma função expressa por uma composição de senos. Os coeficientes desta função são calculados por um procedimento – já programado – que, a partir de um arquivo de dados de ondulação do mar em um período conhecido encontra a composição de senóides que melhor representa o padrão estatístico do arquivo de dados. Para isso, utilizamos um método de otimização aplicado ao ajuste de curvas chamado método de Nelder-Mead.

Esse método otimiza uma função por um algoritmo que não utiliza cálculo diferencial, mas por um método iterativo que avalia a função nos vértices de um simplex. Obviamente, se o arquivo de dados for extenso o suficiente e não contiver grandes singularidades, podemos repeti-lo iterativamente para representar o movimento apenas com os dados de posição pelo tempo obtidos dele.

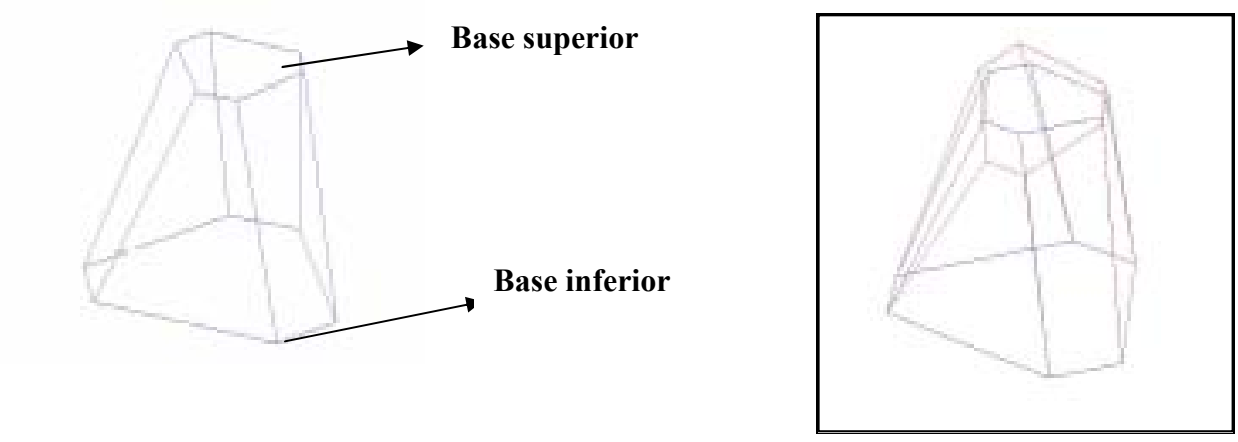
B.1.4 **Modelagem Geométrica**

B.1.4.1 **Modelagem Geométrica Inversa**

A modelagem geométrica pode ser tratada sem qualquer imposição restritiva e sem perda de generalidade para qualquer tipo de plataforma no formato de manipulador paralelo.

Nosso caso de estudo é composto de duas peças rígidas hexagonais, nas quais os ângulos são regulares e os lados são iguais três a três. A peça suposta para ser a parte superior da plataforma móvel é menor do que aquela fixa. As duas peças são unidas por seis cilindros de

atuação que determinarão, com suas distensões, a postura da parte superior. As juntas que unem os cilindros às peças são esféricas, permitem rotação qualquer, mas não translação.



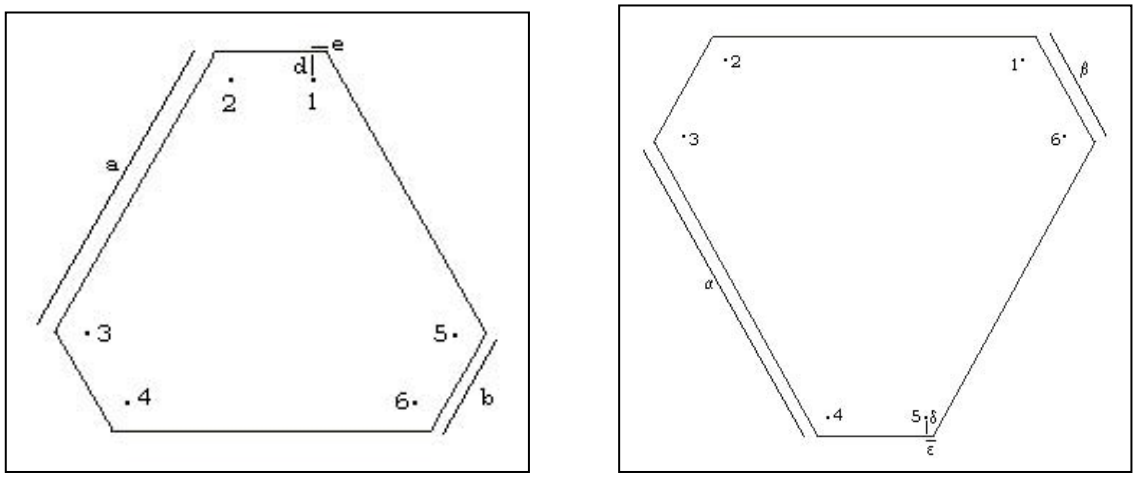
a) Modelo Geométrico

b) Movimentação

Figura B.9 Modelo Geométrico da Plataforma.

Inicialmente, devemos determinar matematicamente a geometria da plataforma; para tanto, atribuímos pontos numerados às extremidades dos cilindros e, em relação a um sistema de coordenadas absolutas e fixo no centróide da peça fixa e inferior, determinamos, por inspeção geométrica, a localização desses pontos, obtendo as coordenadas desejadas após uma movimentação, conforme mostra ilustrativamente a configuração da plataforma antes e depois do movimento na figura B.9.

A base e a placa de metal superior são interpretadas como dois hexágonos irregulares com os pontos de fixação das juntas com os cilindros. A numeração dos pontos para as partes inferiores e superiores são ilustradas na figura B.10.



a) Base Superior b) Base Inferior

Figura B.10 Determinação das coordenados dos pontos nas bases.

Sendo a base fixa, para obter a configuração da parte superior após translação do seu centro geométrico e rotação em torno deste, basta multiplicar cada vetor posição da extremidade superior de cada cilindro pela matriz de rotação que é função dos ângulos de Euler que descrevem a movimentação angular absoluta - em relação a um sistema de referência inicial, e somá-lo ao vetor de translação desejada do centro geométrico. O módulo do vetor obtido é o comprimento necessário do cilindro para a posição requerida da parte superior da plataforma. As coordenadas dos pontos correspondentes a base superior Ps e inferior Pi são apresentadas na equação B.11.

$$P_i = \begin{bmatrix} 1/2\,\alpha - \epsilon & 1/2\,(\alpha+\beta)\,c(t) - \delta & 0 \\ -1/2\,\alpha + \epsilon & 1/2\,(\alpha+\beta)\,c(t) - \delta & 0 \\ -1/2\,\alpha + \epsilon + (2\,\epsilon - \beta)\,c(s) & 1/2\,(\alpha+\beta)\,c(t) - \delta - (\beta - 2\,\epsilon)\,c(t) & 0 \\ -1/2\,\beta + \epsilon & -1/2\,(\alpha+\beta)\,c(t) + \delta & 0 \\ 1/2\,\beta - \epsilon & -1/2\,(\alpha+\beta)\,c(t) + \delta & 0 \\ 1/2\,\alpha - \epsilon - (2\,\epsilon - \beta)\,c(s) & 1/2\,(\alpha+\beta)\,c(t) - \delta - (\beta - 2\,\epsilon)\,c(t) & 0 \end{bmatrix}$$

B.11

$$P_s = \begin{bmatrix} 1/2\,b - e & 1/2\,(a+b)\,c(t) - d & h \\ -1/2\,b + e & 1/2\,(a+b)\,c(t) - d & h \\ -1/2\,a + e + (2\,e - b)\,c(s) & -1/2\,(a+b)\,c(t) + d + (b - 2\,e)\,c(t) & h \\ -1/2\,a + e & -1/2\,(a+b)\,c(t) + d & h \\ 1/2\,a - e & -1/2\,(a+b)\,c(t) + d & h \\ 1/2\,a - e - (2\,e - b)\,c(s) & -1/2\,(a+b)\,c(t) + d + (b - 2\,e)\,c(t) & h \end{bmatrix}$$

onde h é a altura da mesa na configuração inicial e a i-ésima linha de cada matriz representa as coordenadas da extremidade superior - Ps - ou inferior - Pi - do cilindro i. Sejam t = π/6 e s = π/3.

B.1.4.2 **Obtenção das matrizes de transformação de coordenadas**

Para a obtenção das matrizes de transformação de coordenadas de um ponto em relação ao sistema de coordenadas centrado na parte superior, podemos utilizar a expressão a seguir:

$$T(\theta,\rho,\psi)=\begin{bmatrix} c\rho\,c\theta & -s\psi\,s\rho\,c\theta+c\psi\,s\theta & c\psi\,s\rho\,c\theta+s\psi\,s\theta \\ -c\rho\,s\theta & s\psi\,s\rho\,s\theta+c\psi\,c\theta & -c\psi\,s\rho\,s\theta+s\psi\,c\theta \\ -s\rho & -s\psi\,c\rho & c\psi\,c\rho \end{bmatrix} \quad \text{B.12}$$

Esta matriz será designada pela letra R. A matriz de transformação inversa, que neste caso, trata-se de rotações elementares, pode ser obtida a partir da transposta da matriz de transformação de coordenadas.

$$T^{-1}(\theta,\rho,\psi)=\begin{bmatrix} c\rho\,c\theta & -s\theta\,c\rho & -s\rho \\ -c\theta\,s\rho\,s\psi+c\psi\,s\theta & s\theta\,s\rho\,s\psi+c\psi\,c\theta & -c\rho\,s\psi \\ s\theta\,s\psi+c\theta\,c\psi\,s\rho & c\theta\,s\psi-c\psi\,s\theta\,s\rho & c\rho\,c\psi \end{bmatrix} \quad \text{B.13}$$

Essa matriz pode ser interpretada como aquela que transforma o vetor associado a cada cilindro para uma nova configuração, a partir da adição de um termo correspondente ao movimento de translação.

O problema então se resume a determinar as novas coordenadas das extremidades superiores dos cilindros após uma movimentação arbitrária, tendo em vista as extremidades inferiores estão fixas em juntas esféricas.

Imaginamos então uma movimentação, respeitando a rigidez da peça superior, dos vetores posição das extremidades superiores. Isso significa aplicar a esses vetores uma transformação. Para incluir nessa transformação a condição de rigidez da peça superior expressamos como uma

translação do centróide e uma rotação da mesa. Esses parâmetros coincidem convenientemente com os parâmetros pelos quais o movimento a ser simulado é conhecido.

Para deduzirmos matematicamente esta transformação, como observamos, basta multiplicar os vetores posição das extremidades superiores por uma matriz associada ao efeito de rotação e somar um vetor de translação.

A matriz de rotação é função dos ângulos de Euler, ou seja, rotação em torno dos eixos na ordem X, Y e Z. Para obtermos a rotação do vetor posição dos pontos e não do sistema, precisamos inverter o produto das matrizes de rotação em torno de X, Y e Z, respectivamente. Como o sistema é ortogonal, inversa desta matriz produto é a sua transposta.

O comprimento solicitado ao cilindro, tamanho mais curso, é a norma do segmento da extremidade inferior do cilindro até a extremidade superior, esta determinada pela transformação de coordenadas. Assim, o comprimento de cada cilindro quando na configuração inicial e após uma movimentação serão dados, através das equações B.14:

$$L = \sqrt{\sum_{j=1}^{3} \left(\bar{P}_s^{kj} - \bar{P}_i^{kj} \right)^2} \qquad \text{B.14}$$

$$L + \Delta L = \sqrt{\sum_{j=1}^{3} \left[\mathbf{R}_j(\theta, \rho, \psi) \bar{P}_s^k - \bar{P}_i^{kj} + \bar{T} \right]^2}$$

onde R_j é a j-ésima linha da matriz de rotação P^{kj} é a j-ésima coordenada do ponto T^k é o vetor de translação. É necessário lembrar que a configuração inicial da mesa não deve ser aquela com os cilindros totalmente recuados, a menos que a inclinação fornecida seja compensada pela translação, senão teríamos valores negativos de distensão a partir de um valor nulo, o que seria impossível.

B.1.4.3 **Modelagem Geométrica Direta**

Sabemos que, devido aos ruídos nos sinais, imprecisões na distensão dos cilindros e o fato de tratarmos o posicionamento da mesa como um problema puramente geométrico, sem considerações mecânicas, há imprecisões na postura requerida.

Para corrigir erros de maior ordem, definiremos um procedimento para calibrar a plataforma. Qualquer algoritmo para calibração deve, a partir de dados que revelem direta ou indiretamente a postura da mesa, determinar os parâmetros de rotação e translação que, na modelagem inversa, usamos para posicioná-la, a diferença entre esses valores é o erro, que deve ser minimizado.

A literatura, em textos como Zhang Lanfang Wu Jiangning and Li Shilun e Luc Baron and Jorge Angeles (2000), procura desenvolver métodos de calibração automática, em tempo real de movimentação. A abordagem mais eficiente das duas é a exposta nesta última, que utiliza uma equação de desacoplamento entre rotação e translação.

A abordagem usada em Luc Baron and Jorge Angeles (2000) é, partindo da pressuposição de sensores que determinam completa ou incompletamente a posição das juntas superiores, encontrarem a translação inicialmente como a variação ao vetorial do vetor posição do centróide da mesa.

A orientação é encontrada mais facilmente uma vez que a translação é conhecida e o procedimento para tal tem custo computacional de ordem constante, pois não exige iterações. Utiliza-se um método matricial de minimização para encontra a matriz de rotação. O esforço de cálculo desse método é dado pela minimização e pela inversão de uma matriz de projeção de medida, definida pelos autores como aquela que projeta os vetores do espaço real para o sub-espaço de medida dentro do qual os sensores têm todas as informações sobre o posicionamento das juntas.

A calibração que proporemos, sendo ou não utilizada, não será automática, tampouco utilizando sensores. Faremos uso de um robô que determinará as coordenadas de pontos quaisquer sobre e a mesa.

Assim, são definidos três pontos sobre a mesa na forma de um triângulo eqüilátero de maneira a conhecermos suas coordenadas em relação ao sistema absoluto citado na configuração

inicial. Após um movimento arbitrário, fazemos o robô tocar os mesmos três pontos, determinando suas coordenadas. Os pontos são numerados como na figura B.15, apresentada anteriormente.

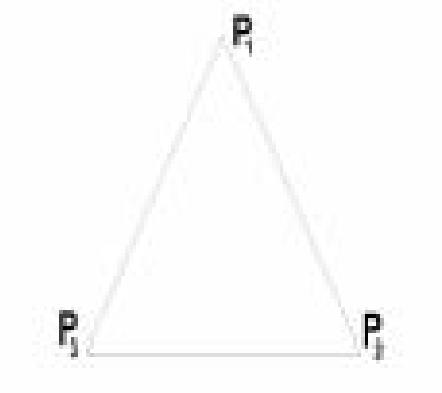


Figura B.15 Triângulo marcado para determinação da cinemática direta.

Como se pode notar, o segmento orientado $\overline{P3P2}$ é paralelo ao canônico i; e o vetor $\overline{P1P3} \times \overline{P1P2}$ é paralelo a k. Normalizando os segmentos orientados, determinaremos a angulação pelas componentes dos vetores normalizados citados. Sabemos que:

$$\vec{n_i} = \mathrm{R}\vec{i} = \begin{bmatrix} c(\theta)c(\rho) \\ -s(\theta)c(\rho) \\ -s(\rho) \end{bmatrix}$$

$$\vec{n_k} = \mathrm{R}\vec{k} = \begin{bmatrix} s(\theta)s(\psi) + c(\theta)s(\rho)c(\psi) \\ c(\theta)s(\psi) - s(\theta)s(\rho)c(\psi) \\ c(\rho)c(\psi) \end{bmatrix} \qquad \text{B.16}$$

Então, obtemos as seguintes equações vetoriais:

$$\mathrm{R}\vec{i} = \frac{\overrightarrow{P_3P_2}}{|\overrightarrow{P_3P_2}|}$$
$$\mathrm{R}\vec{k} = \frac{\overrightarrow{P_1P_3} \times \overrightarrow{P_1P_2}}{|\overrightarrow{P_1P_3} \times \overrightarrow{P_1P_2}|} \qquad \text{B.17}$$

Obtendo-se a partir da resolução os ângulos:

$$\rho = -\arcsin(n_{iz})$$
$$\theta = -\arcsin\left(\frac{n_{iy}}{c(\rho)}\right)$$
$$\psi = \arccos\left(\frac{n_{kz}}{c(\rho)}\right) \qquad \text{B.18}$$

Com os ângulos conhecidos, determinamos a matriz de rotação e, em seguida, a translação do centro de massa da plataforma. Pela forma como marcamos o triângulo, seu centro geométrico coincide com o centro da mesa, assim, o centróide da plataforma.

$$\vec{C} = \frac{1}{n}\sum \vec{P_i} \qquad \text{B.19}$$

A posição do centróide da plataforma após a movimentação é:

$$\vec{C'} = \mathrm{R}\vec{C} + \vec{T} \qquad \text{B.20}$$

sendo T o vetor translação.

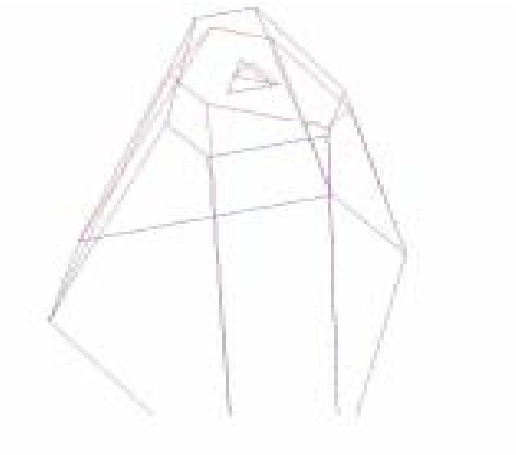


Figura B.21 Movimentação da plataforma e do triângulo marcado.

Assim,

$$\vec{T} = \frac{1}{n}\sum \vec{P}_i - \mathrm{R}\vec{C} \qquad \text{B.22}$$

A partir do conhecimento dos parâmetros de posição, o posicionamento da plataforma poderá ser corrigido baseado no erro encontrado.

B.2 **Implementação do sistema de supervisão e controle**

B.2.1 **Sistema Operativo**

Em um estudo inicial detalhado, tendo como principal objetivo a especificação funcional do Sistema de Supervisão e Controle da Plataforma Hidráulica onde seus diversos graus de liberdade que permitirá a Simulação do Espectro de Movimentos da Superfície do Mar, foi especificado uma Interface de rede Fieldpoint da National Instrumentation™, com interface RS232/RS-485, com módulos com oito saídas analógica (FP AO 200 output), oito saídas a relês, conforme mostra a figura B.23.

a) Sistema Fieldpoint e módulos.

b) Módulos de E/S analógicos.

Figura B.23- Interface Fieldpoint da NI Instrumentation.

A interface com o usuário será desenvolvida utilizando o software LABVIEW™, compatível com esta interface, possibilitando o monitoramento e controle de informações, modos de funcionamento, leitura do arquivo de dados, armazenamento de informações e acionamento dos elementos hidráulicos através da interface D/A.

B.3 **Aplicativo Computacional**

A partir do detalhamento dos espectros de movimentos da superfície do mar a serem simuladas nas maquetes (arquivo básico contendo amplitudes, freqüências correspondentes a cada um dos graus de liberdade), será implementado um aplicativo computacional estruturado sob a forma de bibliotecas modulares de modo a permitir o desenvolvimento de uma estrutura aberta para ser utilizada em futuras aplicações. Este aplicativo deverá conter o seguinte módulo:

Implementação de Software de Supervisão e Controle e Hardware de Acionamento e Controle em ambiente WINDOWS que permitirá a geração automática de sinais correspondentes ao espectro de movimentos (em termos de freqüência e amplitude) da superfície do mar das operações de intervenção na maquete construída, com carga e sem carga.

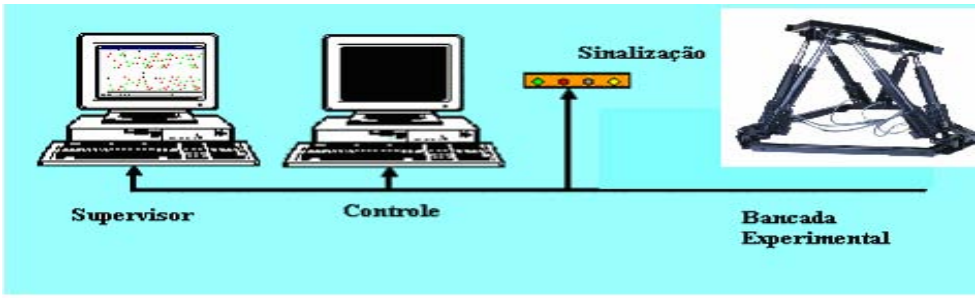


Figura B.24 - Sistema de Supervisão e Controle

A figura B.24 apresenta uma representação esquemática do sistema completo de aquisição e monitoramento de informações da maquete a ser implementada. Ele é constituído de um módulo fieldbus da National Instruments, constituído de interfaces de entrada-saída (I/O), responsáveis pela aquisição das informações I/O provenientes dos sensores externos (por exemplo, chaves de início e final de operação) e digital-analógica (D/A) que deverá permitir a geração de sinais de referência (ou trajetórias) para os cilindros hidráulicos de posicionamento.

Todo o gerenciamento dessas informações deverá ser realizado através de um programa computacional de supervisão e controle residente num computador PC, onde será implementada uma interface de visualização, a partir do desenvolvimento de telas de monitoramento das informações dos sensores (final de curso, segurança, chaves lógicas, etc.), e programas de

tratamento matemático (posição, velocidade e aceleração), e eventualmente informações sobre o status do sistema (velocidade, aceleração, número de pontos da trajetória, etc.). Na parte final deste anexo são apresentadas algumas telas do software de supervisão e controle de movimentos, a ser implementado em LABVIEW^{TM}.

Figura B.25 - Tela principal do programa

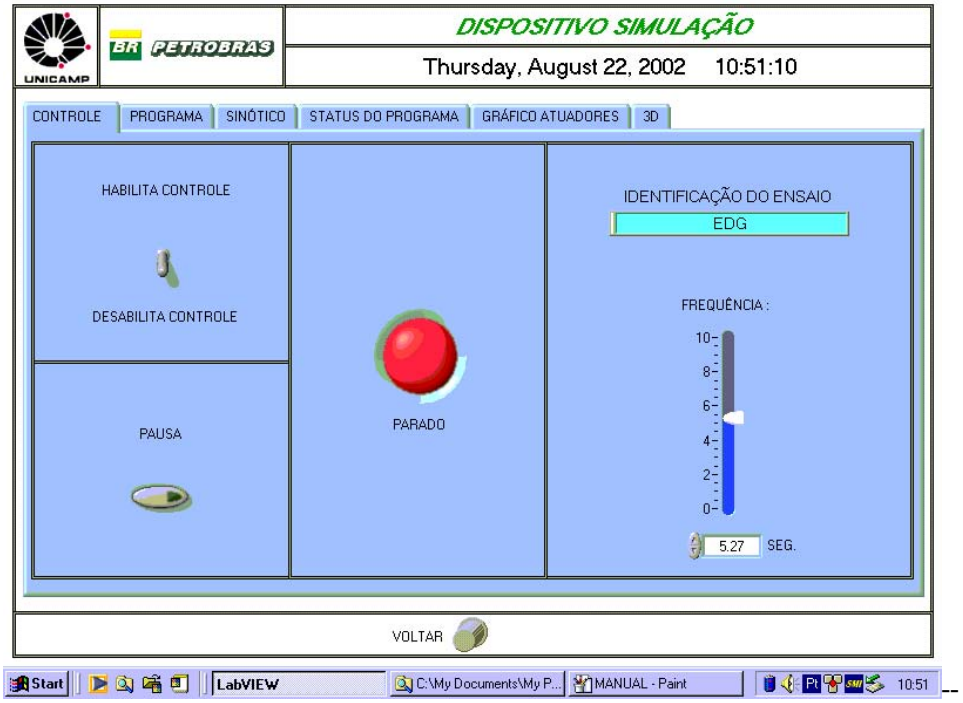


Figura B.26 - Tela de controle (modo automático)

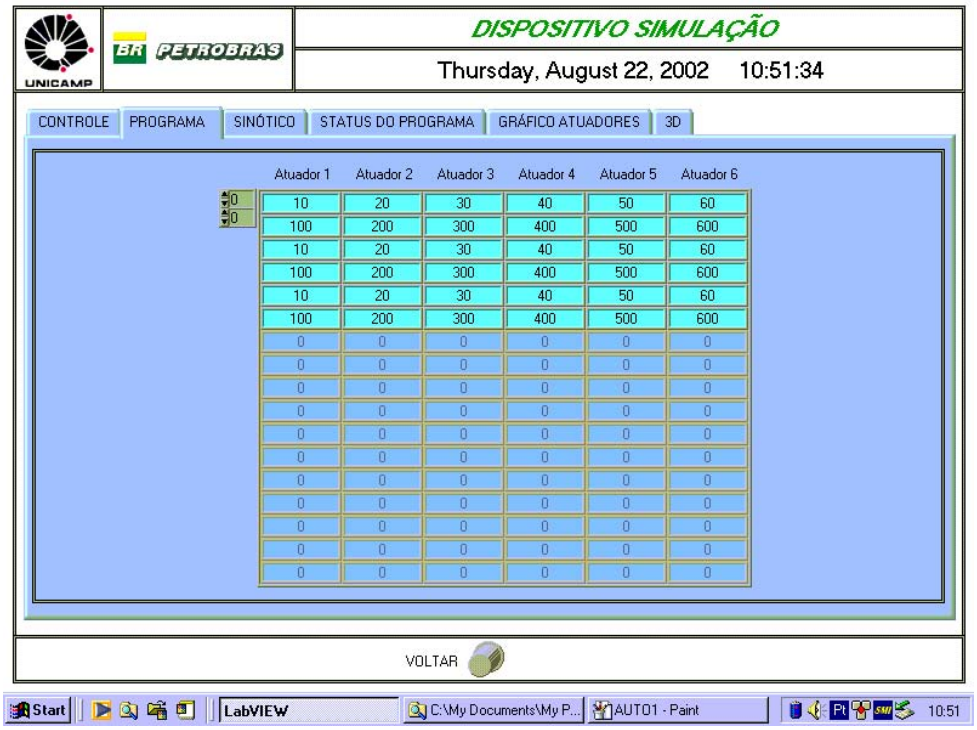


Figura B.27 - Tela de edição de pontos de movimentação

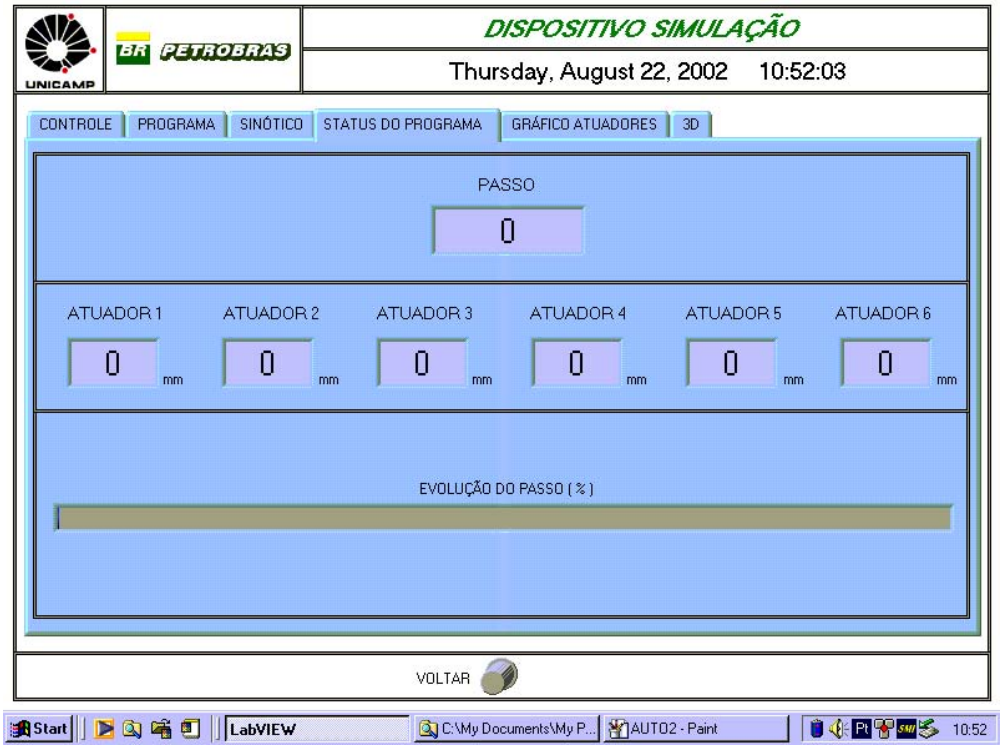


Figura B.28 - Tela de envio de pontos de movimentação

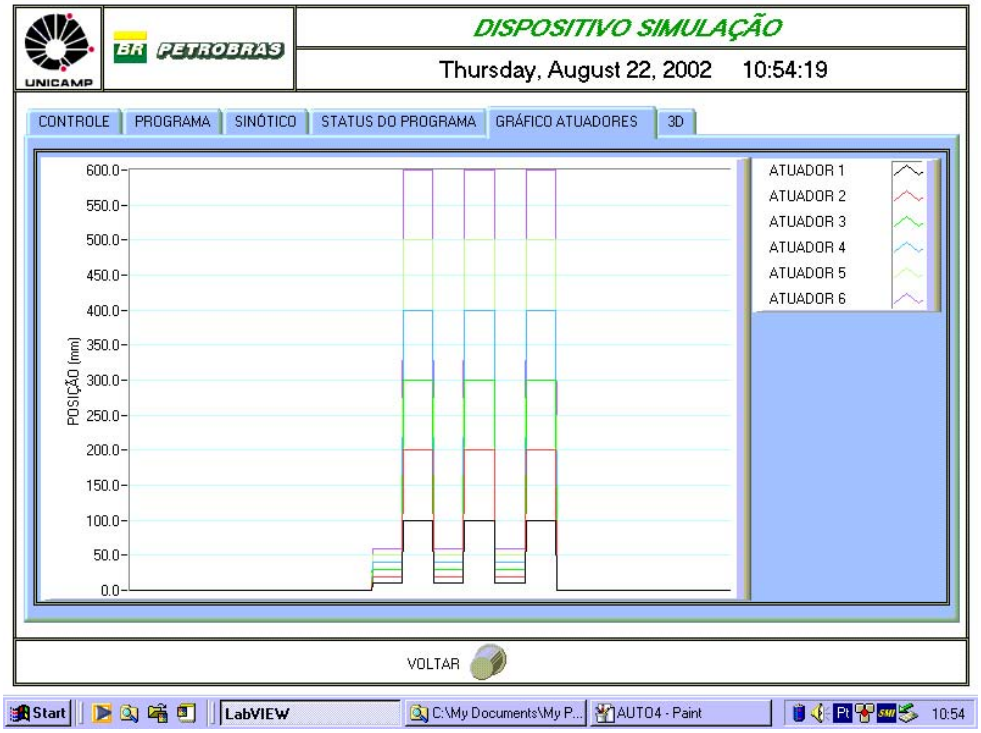


Figura B.29 - Tela gráfica de pontos de movimentação

Figura B.30 - Banco de Dados (arquivos de movimentação)

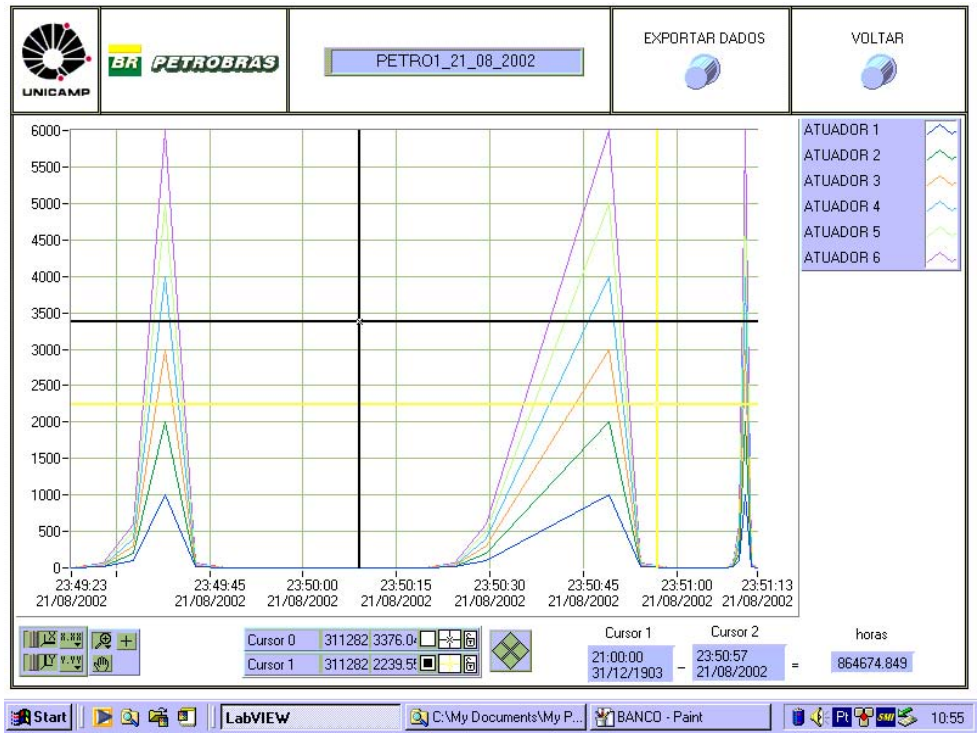


Figura B.31 - Tela de Monitoramento de movimentação da Base

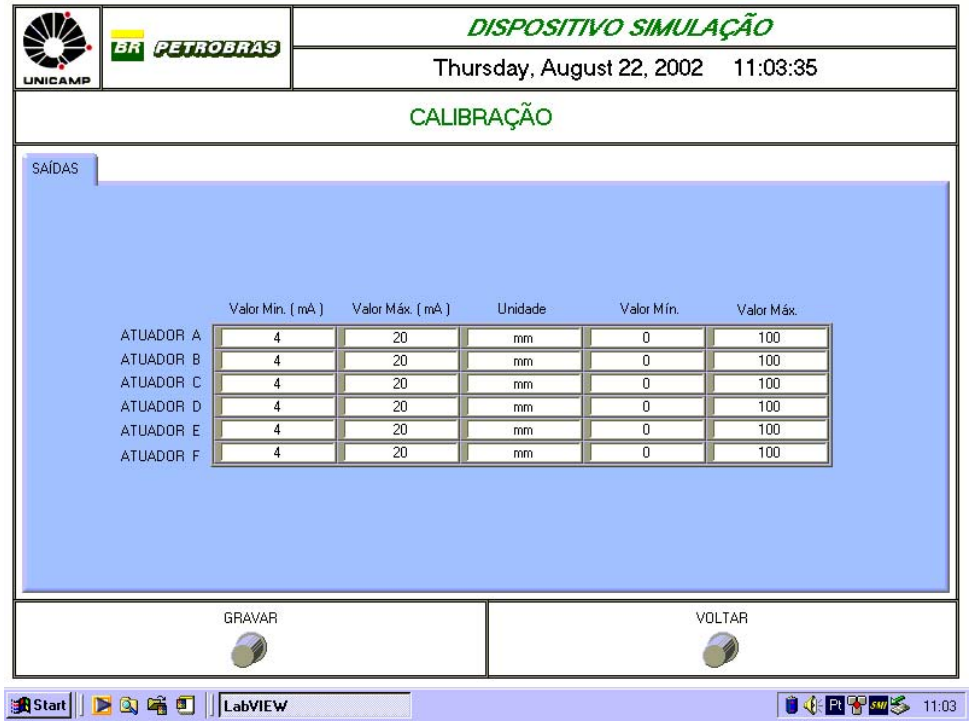


Figura B.32 - Tela de calibração dos cilindros de movimentação da Base

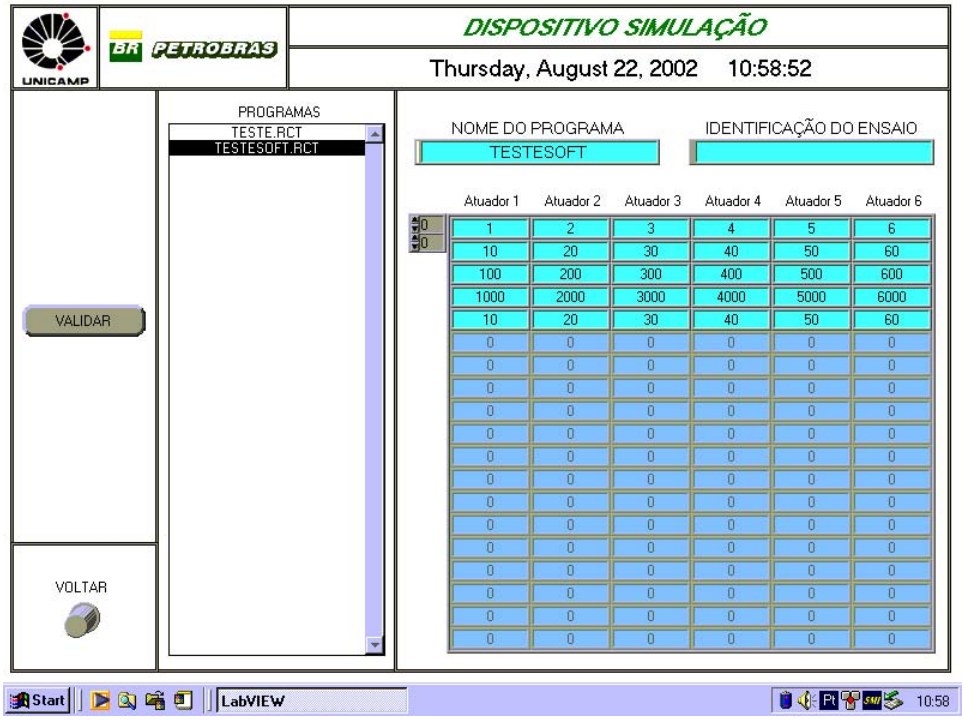


Figura B.33: Tela de carregamento de arquivos de dados

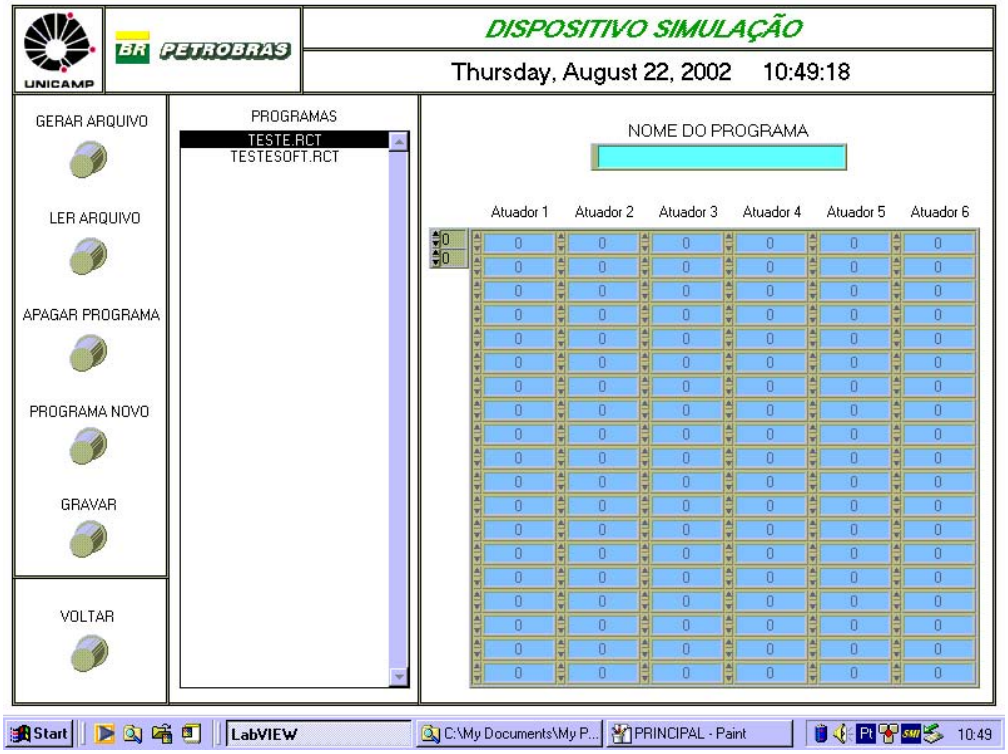


Figura B.34 - Tela de edição de pontos de movimentação

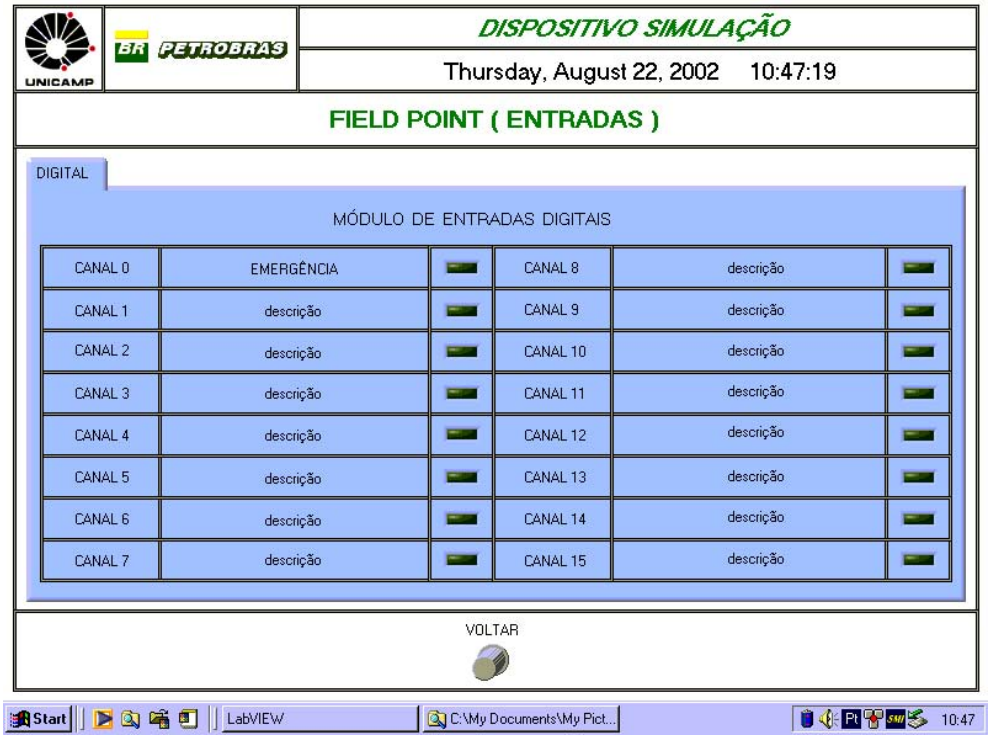


Figura B.35 - Tela de definição de pontos (Fieldpoint)

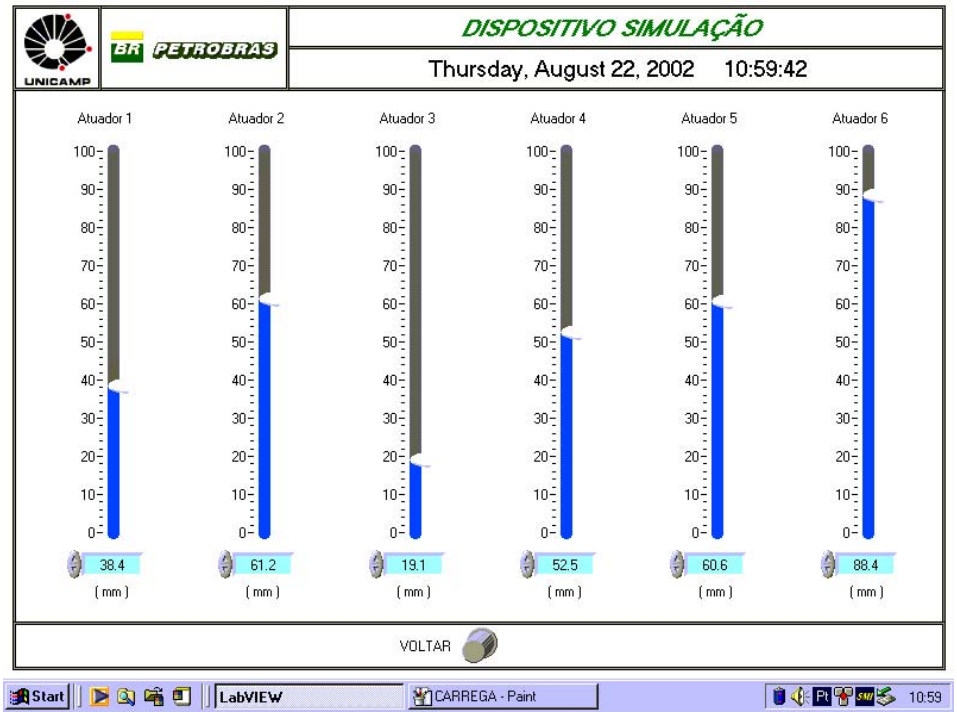


Figura B.36 - Tela de operação (modo manual)

Fotografias do Protótipo Implementado no LAR-UNICAMP

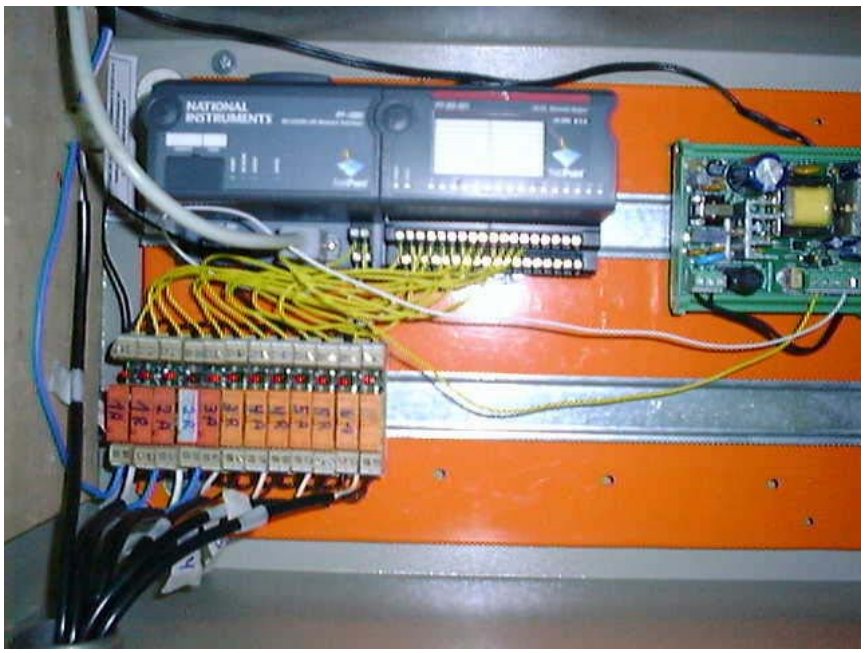


Figura B.37 - Controlador Field Point National Instruments

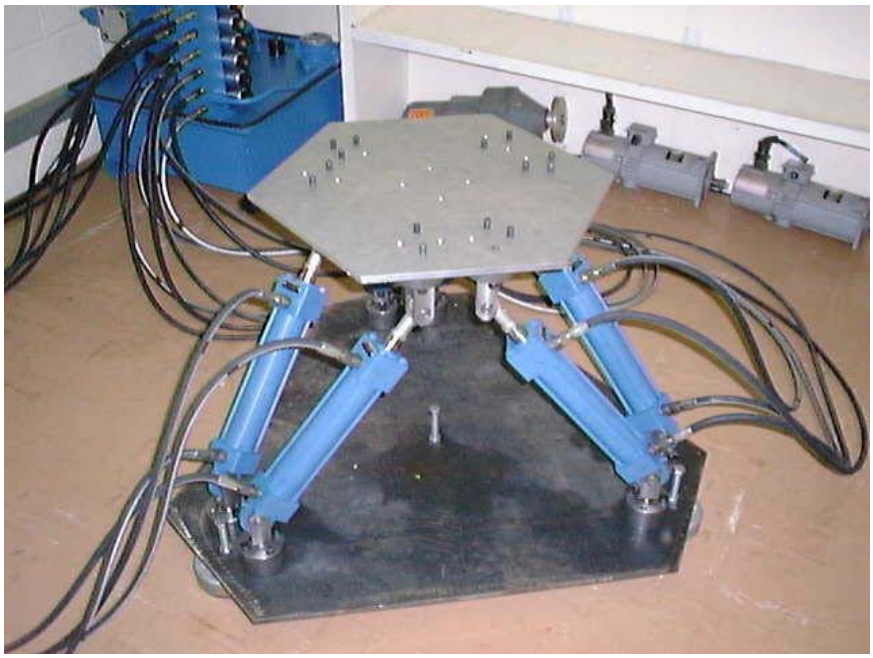


Figura B.38 - Dispositivo Mecânico.

ANEXO III

Programas Computacionais Implementados para Calibração do Zero Veículo

C.1 Descritivo dos programas computacionais implementados

Neste anexo é apresentado o conjunto de programas computacionais e bancos de dados desenvolvidos para esta aplicação. O software desenvolvido é composto basicamente de um programa gestor e quatro módulos aplicativos que poderão funcionar independentes.

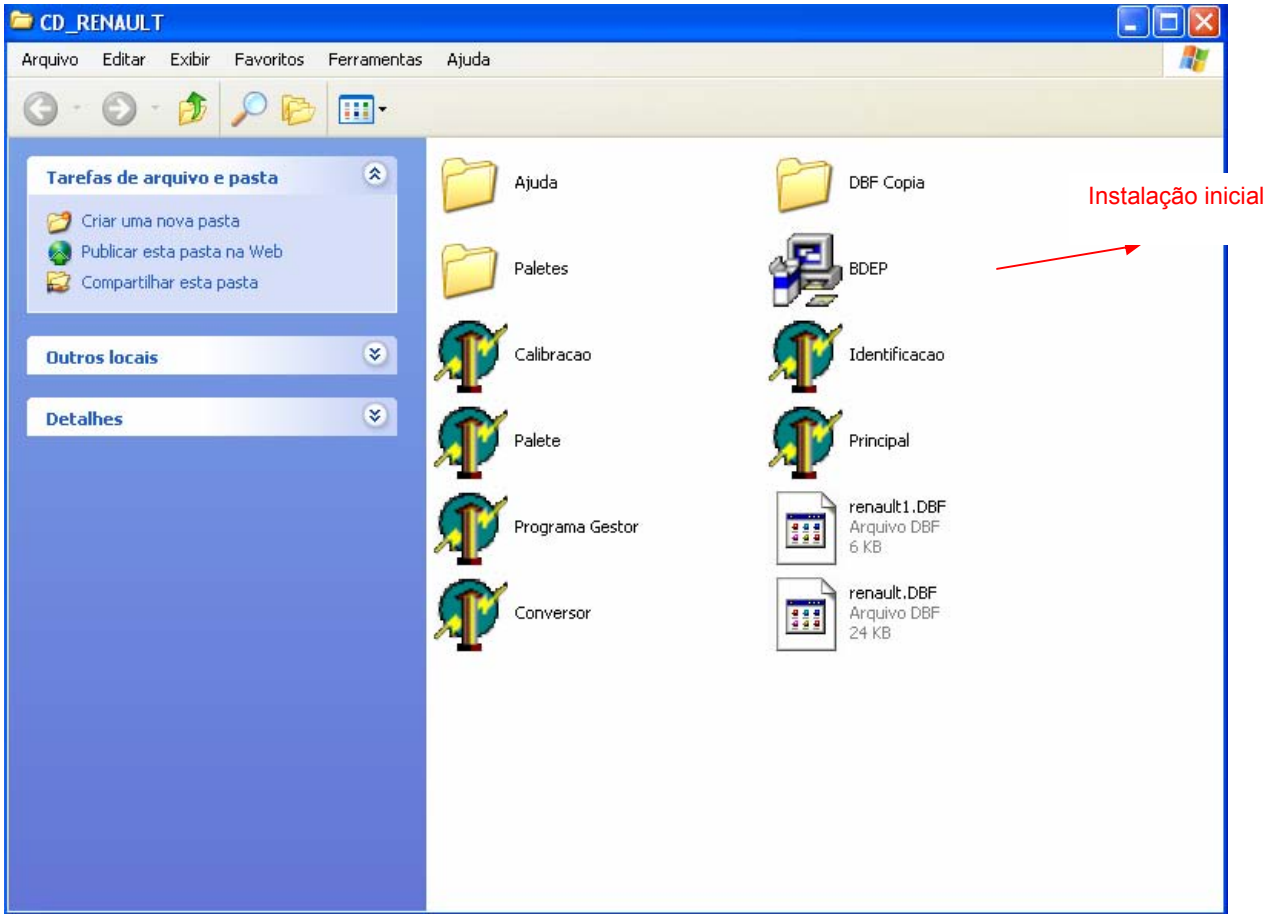


Figura C1: Conteúdo do Aplicativo desenvolvido.

C.2 **Descrição detalhada dos módulos implementados**

C.2.1 **Programa de conversão de ângulos Euler em RPY e vice-versa**

Neste módulo foi implementado um programa de conversão de ângulos do robô (Euler) para o software de programação off-line ROBCAD (RPY) e vice-versa. Para validação dos resultados foi realizado uma comparação entre os resultados obtidos com o software desenvolvido e o programa de conversão de ângulos EULER (robô) e RPY (software ROBCAD) a partir de arquivos de calibração fornecidos pelo fabricante (figura C2). Este programa também será utilizado no programa de calibração da ferramenta e do palete.

ROBCAD (RPY)			COMAU (ZYZ)			Arquivo
Rx	Ry	Rz	A	E	R	
137.199	4.235	7.600	-86.96	137.03	83.78	c1
158.193	15.386	40.629	-82.92	153.53	53.47	c1
-172,27	6.210	119.950	-111,48	170.10	-51.01	c1
-136,3	-10.602	165.049	-115,85	135.29	-105,16	c1
100.880	16.062	13.054	-79.99	100.45	73.66	e1
114.752	0.690	5.808	-84.51	114.75	89.24	e1
60.598	-24.120	24.388	-78.58	63.38	117.20	e1
92.241	30.541	19.939	-71.20	91.93	59.44	e1
-130,41	-70,863	-47,749	3.44	102.27	-165,2	d1
141.317	-68,018	64,278	23.47	106.99	165,84	d1
-120,84	-32,751	-80,289	-8,19	115.54	-126,84	d1
-102,35	-20,289	-85,425	0.23	101.58	-110,73	d1
-115,94	38.965	-147,55	-40,54	109,89	-48,03	g1
-125,99	29.807	-113,21	-3,36	120,66	-54,7	g1
-103,37	9.822	-109,09	-16,77	103,17	-79,91	g1
-135,68	3.182	-97,304	-4,05	135,59	-85,45	g1
-157,82	9.795	-156,86	-44,21	155.86	-65,42	h1
167,513	34.215	161.873	3.37	143.84	17.64	h1
-176,46	5.279	-117,58	28.54	173.65	-33,71	h1
-155,45	29.216	108.191	-114,91	142,55	-36,61	h4
-155,45	29.216	-113,28	23.61	142,55	-36,61	h4
-155,45	29.216	162.491	-60,61	142,55	-36,61	h4
-[illegible],258	37.724	172.712	-118,02	65.41	-47,71	h4

Figura C2: Arquivo de pontos de testes – ângulos de orientação obtidos através do software ROBCAD e do robô COMAU utilizado.

O software implementado, conforme mostra a figura C3, poderá trabalhar de duas maneiras: utilizando os ângulos de Euler ou quarternions de orientação para indicar a orientação da ferramenta de manipuladores com diferentes descrições dos ângulos de orientação.

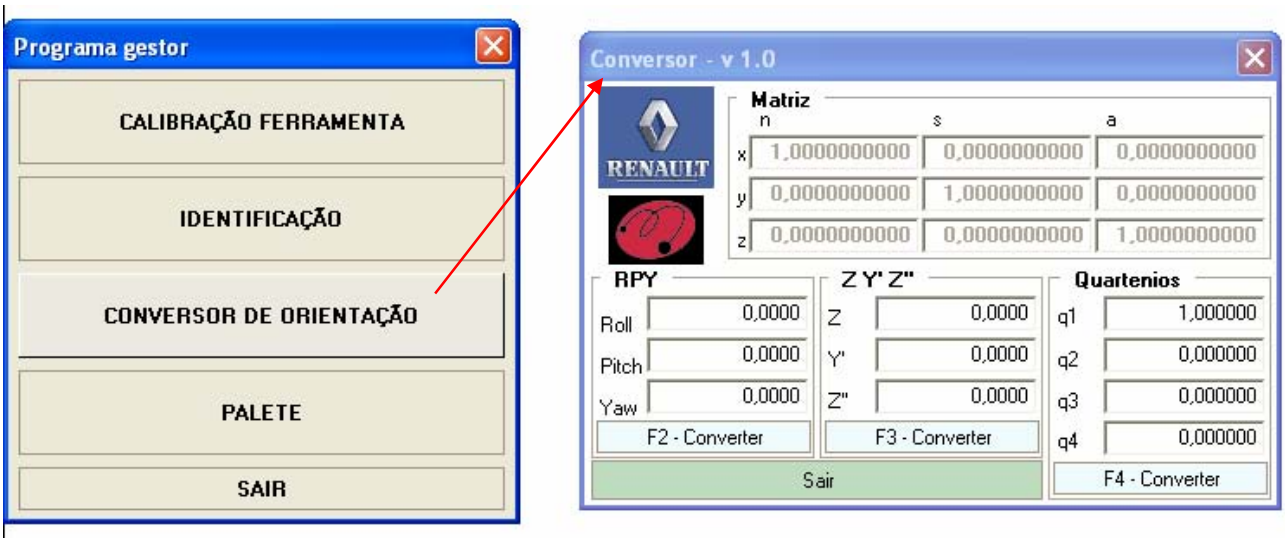


Figura C3: Tela do Programa de Conversão de Ângulos de Orientação.

A figura C4 apresenta um exemplo de utilização deste utilitário. A entrada dos valores angulares em graus poderá ser descrita através dos:

a) ângulos Roll, Pitch e Yaw fornecidos pelo software ROBCAD (RPY): neste caso entrar com os dados e teclar F2, e serão obtidos os valores angulares utilizados pelo robô COMAU (ângulos de Euler, Z,Y,Z);

b) ângulos de Euler (Z,Y,Z) fornecidos pelo robô COMAU: neste caso entrar com os dados e teclar F3, e serão obtidos os valores angulares utilizados pelo software ROBCAD (ângulos RPY);

c) representação através de quartenions de orientação: neste caso entrar com os dados e teclar F4, e serão obtidos os valores angulares utilizados pelo software ROBCAD (ângulos RPY) e ângulos de Euler (Z,Y,Z) fornecidos pelo robô COMAU.

Podemos também observar o aplicativo computacional implementado sempre fornecerá informações adicionais concernentes a matriz de orientação e representação desses ângulos através de quartenions de orientação.

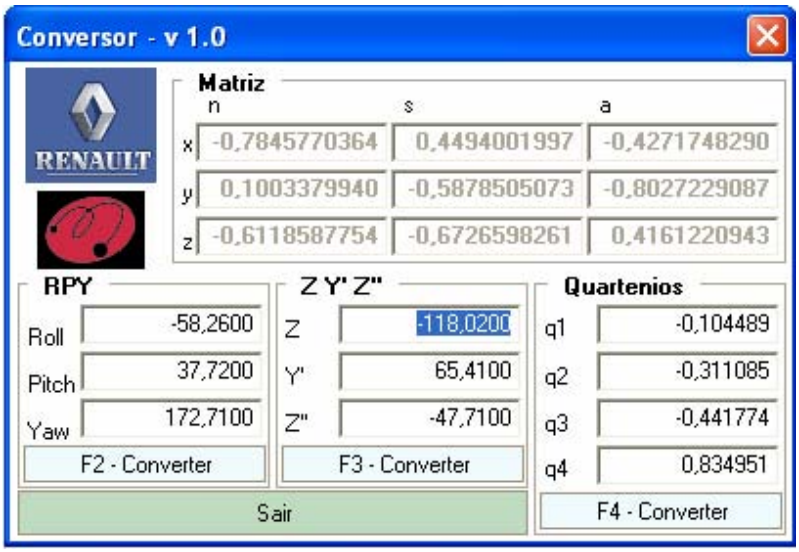


Figura C4: Exemplo de utilização.

C.2.2 **Editor de Paletes**

O Editor de Paletes é um programa para entrada de dados do palete, no formato de planilha Excel, que permite a edição, alteração e escolha de pontos de referencia do palete (figura C5), onde a escolha dos dados de entrada deverá ter o mesmo formato do programa ROBOCAD, servindo também de referência para implementação de trajetórias de calibração no software ROBCAD que deverão ser transferidos ao robô COMAU. Também esta prevista um programa de entrada de dados manual através de dispositivo externo de calibração (braço de calibração HOMMER (XYZ) disponível pelo construtor).

Para sistematizarmos este aplicativo, recomenda-se a implementação de diretório para armazenamento de informações de paletes existentes na linha de soldagem.

C.2.2.1 **Seqüência de instruções implementadas**

a) **Inserção de novo palete:** O palete deverá conter as seguintes informações: Nome de palete, Pin = número do pino e as coordenadas cartesianas (X, Y e Z) referentes a posição do pino no palete.

b) **Edição de pontos:** A entrada dos pinos de calibração deverá obedecer à seqüência apresentada na figura C6, ou seja (1,2,3, ..).

c) **Exclusão de um palete:** Um palete poderá ser excluído a partir da simples digitação do nome do mesmo, e seleção do comando Excluir

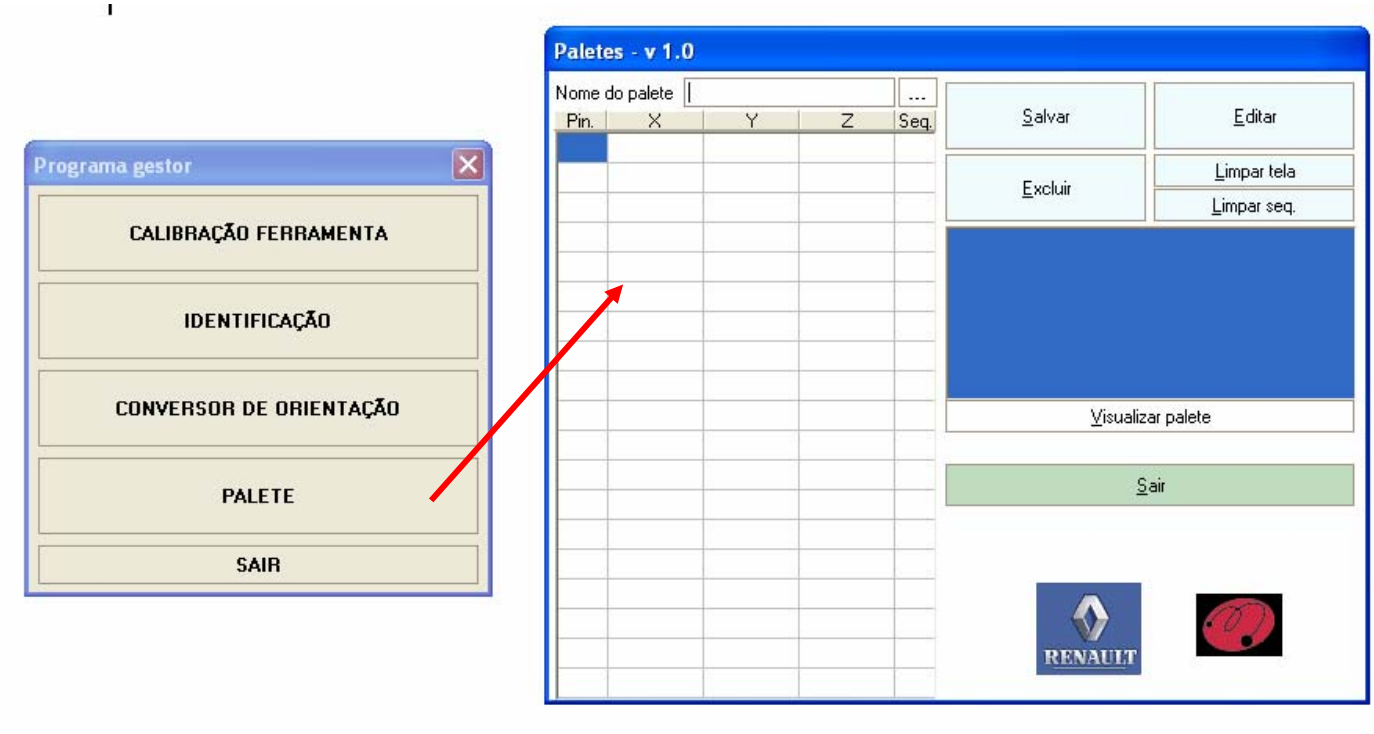


Figura C5: Tela do software de gerenciamento e edição de paletes.

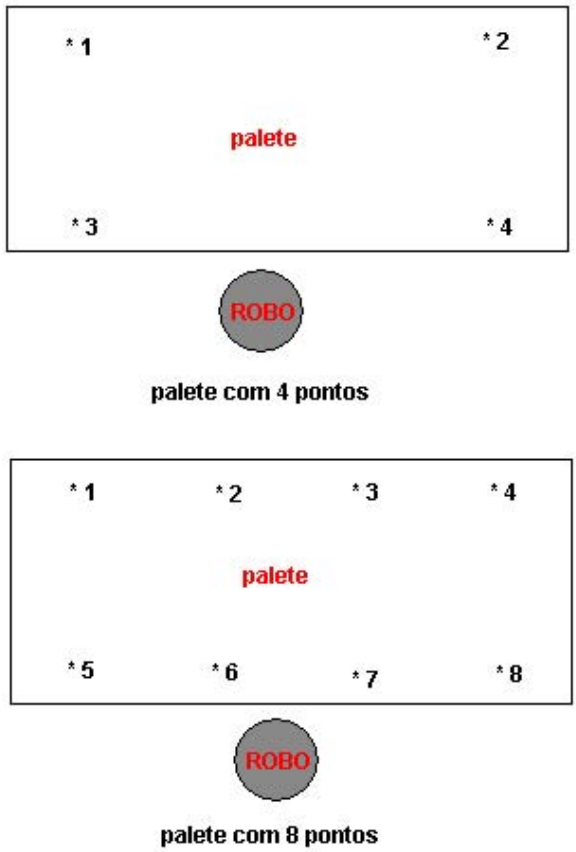


Figura C6: Seqüência de pontos do palete a serem inseridos no programa de edição.

A figura C7 apresenta um exemplo de edição de pontos no módulo Editor de Paletes, onde podemos observar:

a) janela com o nome do palete, que poderá ser editado ou recuperado sob a forma de arquivo de dados;

b) uma tabela contendo as seguintes informações nas colunas:

Pin: número do pino

X, Y,Z: Coordenadas cartesianas correspondentes ao palete;

Seq: seqüência de pinos de calibração a ser utilizada no programa de calibração;

c) Visualização do palete e pontos associados ao mesmo.

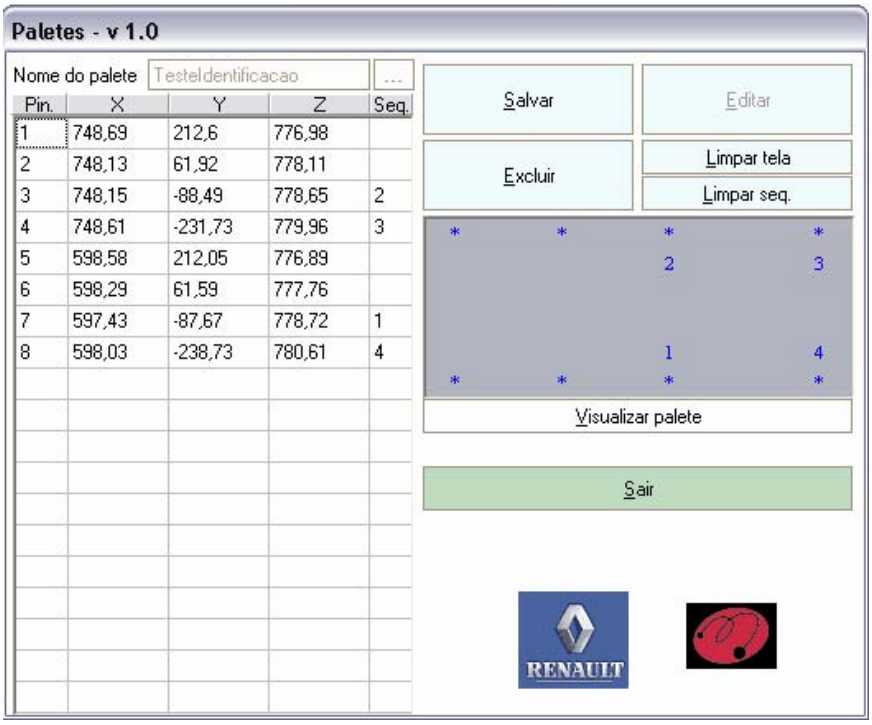


Figura C7: Seqüência de pontos do palete inseridos no programa de edição.

C.2.3 **Programa de calibração da ferramenta terminal do robô**

Este módulo consiste no cálculo das dimensões da ferramenta a partir de quatro informações de orientação num determinado ponto do espaço (a orientação é variável e a posição mantida sempre constante) Os resultados obtidos durante a calibração da ferramenta (dimensões da ferramenta na forma de X, Y e Z) deverão ser utilizados no programa de calibração do palete (entrada de dados) utilizando o toque manual de quatro pontos (procedimento clássico) ou o programa automático de calibração do sistema de ferramenta, utilizando sistema de visão e software desenvolvido em LABVIEW (figura C8).

Através da utilização desses procedimentos, podemos observar que:

a) No caso de utilizarmos o sistema tradicional de toque num dado ponto, o valor X,Y,Z da ferramenta deverá ser o valor estimado da mesma, dado pelas dimensões aproximadas da mesma;

b) No caso de utilizarmos o sistema de Visão desenvolvido, as dimensões da mesma deverão ser o valor aproximado da mesma até a esfera, após ser realizado o ajuste de foco (ver procedimento de calibração utilizando a câmera);

c) Para calibração da ferramenta, poderá ser utilizado diretamente o software de calibração existente no robô COMAU, utilizando os mesmos procedimentos apresentados.

Figura C8: Tela do software de modo de calibração da ferramenta.

A figura C9 sintetiza a descrição detalhada de utilização do software de calibração da ferramenta terminal do robô, onde podemos observar que os pontos relacionados na planilha de dados apresentada acima representam pontos em relação ao punho do robô (TCP – Tool Center Point).

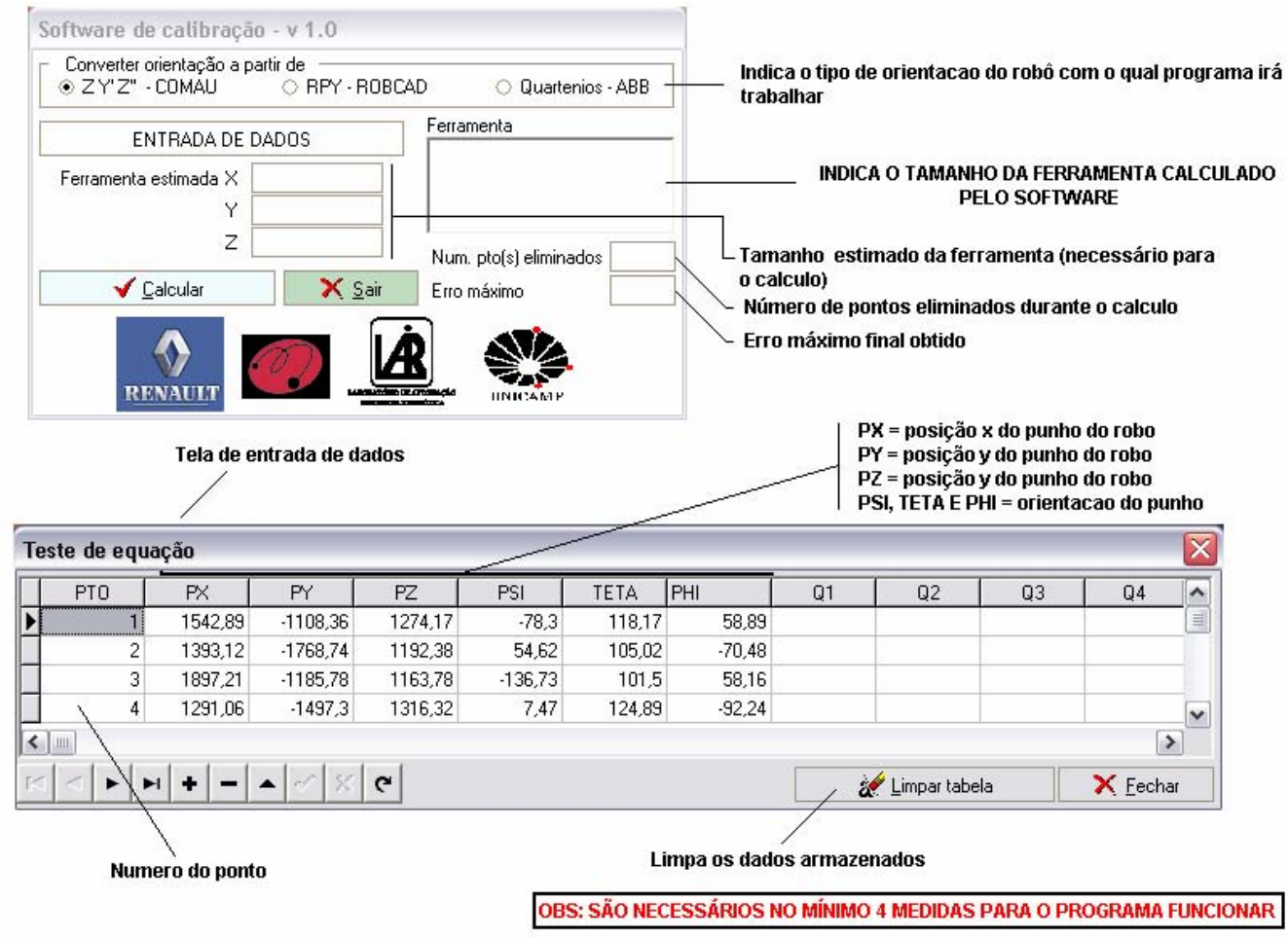


Figura C9: Descrição do software de modo de calibração da ferramenta.

C.2.4 **Exemplo completo de utilização do software de calibração da ferramenta**

Para calibração da ferramenta terminal do robô os seguintes passos deverão ser realizados:

Passo 1: Iniciar o programa principal (GESTOR) e clicar sobre o botão **CALIBRAÇÃO DE FERRAMENTA** (figura C8).

Passo 2: Após clicar sobre o botão será aberto uma janela correspondente ao programa de calibração de ferramentas.

Passo 3: Selecione o arquivo (com extensão .lsv) desejado de pontos do robô (COMAU) clicando sobre o botão.

Passo 4: Neste momento aparecera na tela de seleção de arquivos apresentada na figura C10, permitindo a seleção do arquivo em qualquer diretório desejado.

Passo 5: Selecione o arquivo TesteCalibracao.lsv (marque o arquivo e clique em OK ou clique duas vezes sobre o arquivo (figura C11).

Passo 6: Nos campos para ferramenta estimada coloque os valores correspondentes a dimensão da mesma em mm: X = -599, Y = 0 e Z = 343 (figura C12).

Passo 7: Clique no campo CALCULAR, e após o calculo serão apresentados os valores correspondentes ao tamanho da ferramenta calculado (em azul), número de pontos que não foram utilizados pelo software (desconsiderados, por serem considerados de baixa qualidade de medição) e erro máximo obtido.

Figura C10: Tela de seleção de arquivos do programa de calibração de ferramentas.

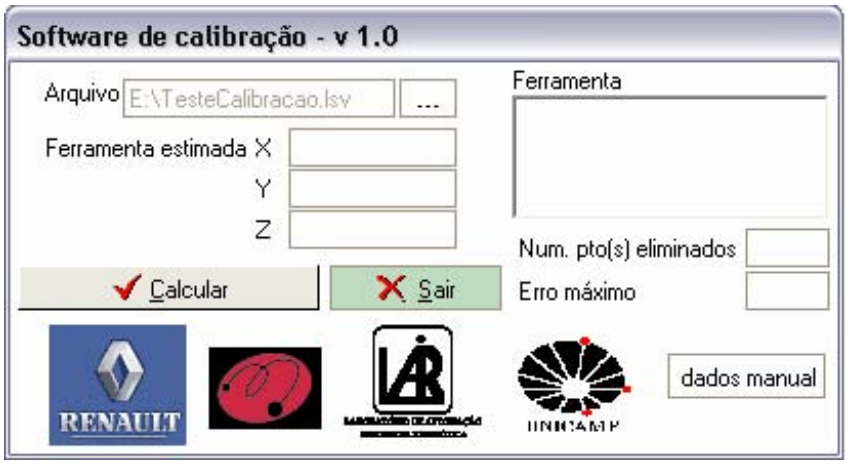


Figura C11: Tela principal do programa de calibração de ferramentas.

Figura C12: Entrada de dados no programa de calibração de ferramentas.

C.2.5 **Exemplo de entrada de dados na forma manual**

O botão DADOS MANUAL permite que sejam observados os dados lidos a partir dos arquivos existentes com a extensão *.lsv, como também a entrada de dados na forma manual (figura C13). Para utilizar este recurso os seguintes procedimentos deverão ser realizados:

a) No caso de uma tabela já existente, para limpar a mesma utilizar o botão **LIMPAR TABELA** e inserir os novos valores desejados.

b) Para navegar na grade, utilize a tecla TAB ou as setas do teclado.

c) Para criar uma nova linha utilize a seta para baixo,

d) Para gravar a ultima linha escrita crie uma linha em branco e clique em **FECHAR**.

```
--
-- File A:\C2\PROG_99.VAR   Thu Dec 16 09:56:08 2004
-- SW Version 5.21
--
--$$   ARM: 1 AXES: 6 AUX:0 MASK: 63

pnt0019p POS  Priv
 X:  -612.78 Y:-1874.57 Z: 598.31 A:  -55.70 E: 165.06 R:-158.22  CNFG: ''

pnt0020p POS  Priv
 X:-375.25 Y:-980.98 Z: 597.72 A:  78.72 E: 165.06 R:  -158.22  CNFG: 'T2:-1'

pnt0021p POS  Priv
 X:-555.81 Y:-1640.42 Z: 748.74 A:  -1.91 E: 134.52 R: -154.46  CNFG: ''

pnt0024p POS  Priv
 X: -832.37 Y:-1375.70 Z: 795.55 A:  10.82 E: 107.52 R:177.17   CNFG: ''''
```

Figura C13: Exemplo de arquivo de calibração de ferramentas do robô COMAU.

A figura C14 mostra a tela de entrada de dados, relativo ao arquivo apresentado anteriormente. A seqüência de pontos (PTO) tem de ser seqüencial como é mostrado nesta figura.

Figura C.14: Dados lidos a partir do arquivo de calibração da ferramenta *. lsv.

C.2.6 **Programa de Identificação do Zero Robô**

O módulo de identificação do Zero do Robô (figura C15) consiste no cálculo das coordenadas do novo referencial zero do robô a partir da leitura de pontos precisos de posicionamento da ferramenta em relação à base (a partir da execução de programa implementado em ROBCAD, a partir da aproximação da ferramenta sobre quatro pontos do palete, discriminados no programa de edição de paletes, onde a orientação permanece constante e a posição varia).

As dimensões da ferramenta, obtidas durante a calibração da ferramenta (dimensões da ferramenta na forma de X, Y e Z) deverão ser utilizados neste programa. Dois procedimentos práticos foram propostos:

a) Toque manual de pontos do palete, utilizando arquivo pré-definido de entrada de pontos;
b) Programa automático de calibração do sistema de ferramenta, utilizando sistema de visão e software desenvolvido em LABVIEW, para identificação de pontos do palete a partir de posicionamento do robô.

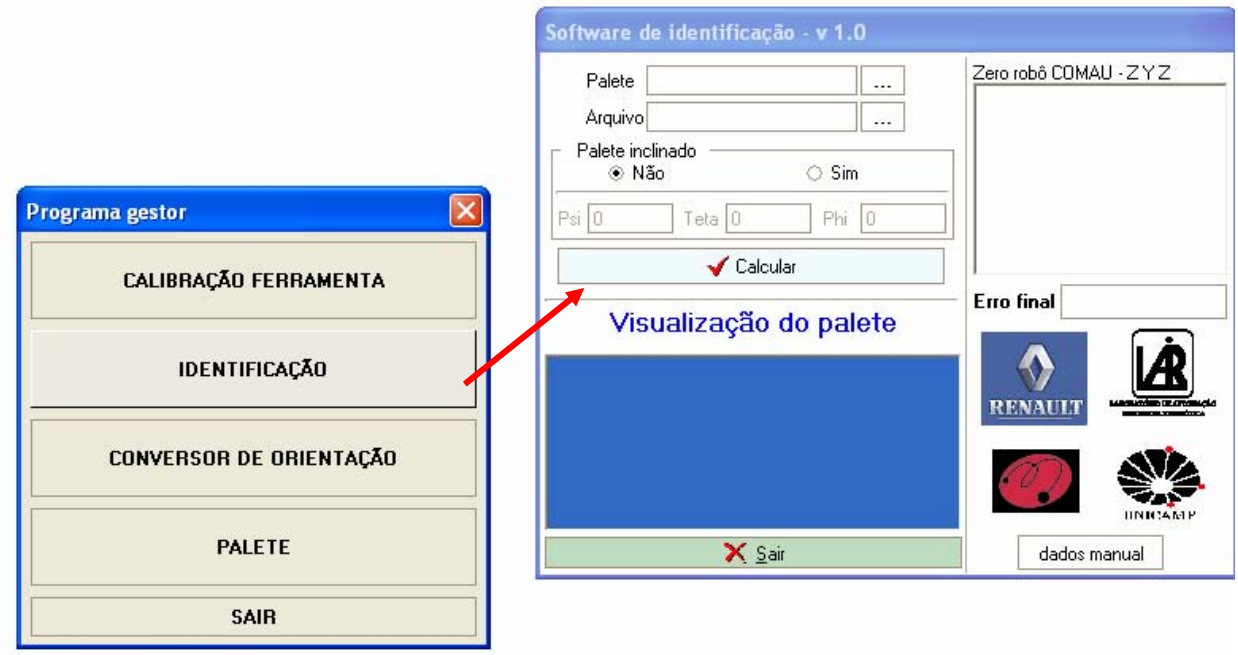


Figura C15: Programa de Identificação do Zero do Robô.

A seguir apresentaremos um exemplo detalhado de utilização do módulo de obtenção do zero do robô, apresentando todos os passos para implementação desse procedimento, a partir das informações de calibração fornecidas em arquivo com extensão *.lsv do robô COMAU, correspondente aos quatro pontos de precisão obtidos no palete através da aproximação da ferramenta do robô sob esses pontos.

C.2.7 **Descrição detalhada dos passos a serem realizados**

Passo 1: Iniciar o software principal (GESTOR) e clicar sobre o botão PALETE (figura C15).

Passo 2: Selecionar o palete no qual foi realizada a medida pelo robô. Como exemplo selecione o pallete **TesteIdentificacao.xls**, para isso) clicando sobre o botão. A figura C16 mostra o arquivo correspondente em formato Excel e a figura C17 apresentam o arquivo carregado.

	A	B	C	D	E	F
1	1	748,69	212,6	776,98		
2	2	748,13	61,92	778,11		
3	3	748,15	-88,49	778,65	2	
4	4	748,61	-231,73	779,96	3	
5	5	598,58	212,05	776,89		
6	6	598,29	61,59	777,76		
7	7	597,43	-87,67	778,72	1	
8	8	598,03	-238,73	780,61	4	
9						
10						
11						

Figura C16: Arquivo do Palete de calibração em formato Excel.

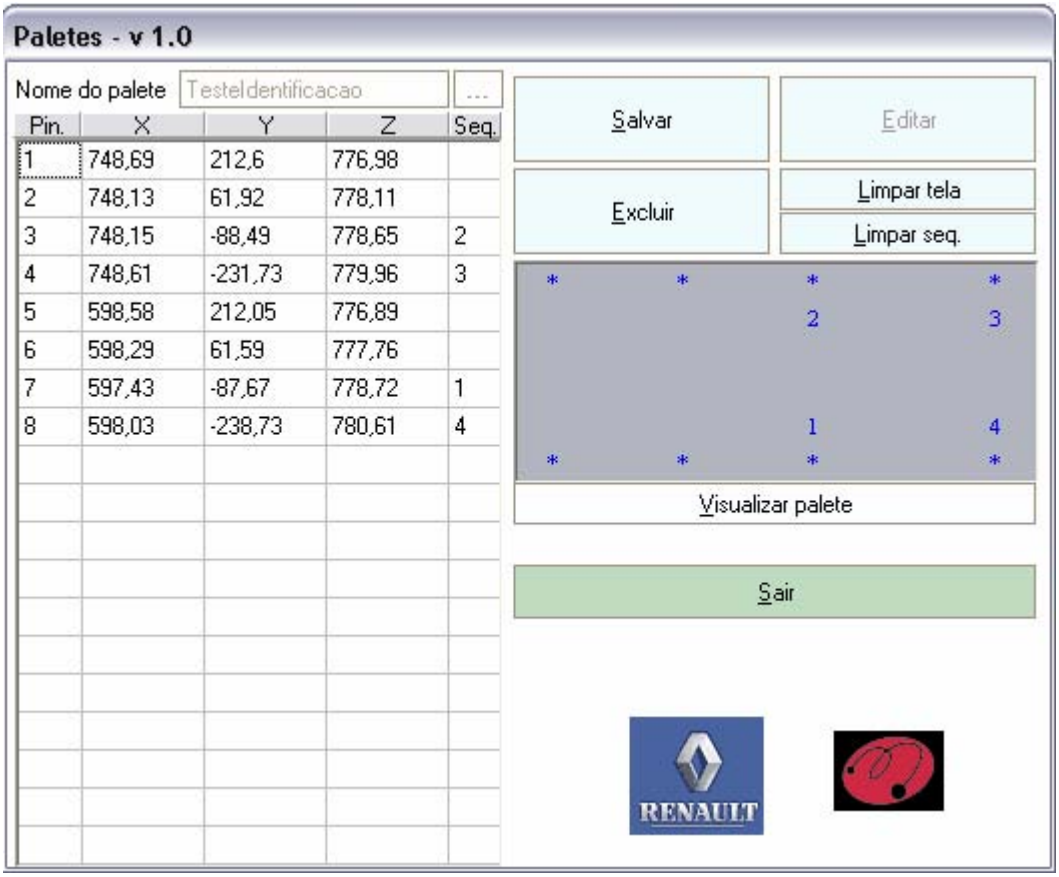


Figura C17: Arquivo de Dados do Palete.

Passo 3: Após a seleção do palete indicar seqüência (Seq.) na qual os pontos foram realizados. Neste exemplo, o ponto 7 do palete foi o primeiro tocado pelo robô, o

ponto 3 o segundo, o ponto 4 terceiro e o ponto 8 por último. Salve o palete (opção **Salvar**) e feche o editor de Paletes.

Passo 4: No software principal (GESTOR), clique sobre o botão IDENTIFICACAO. O software de identificação será aberto.

Passo 5: Selecionar o palete no qual foi realizada a medida pelo robô (por exemplo, **TesteIdentificacao.xls**) clicando sobre o botão**.** Após o arquivo correspondente ao palete ser aberto, a seqüência realizada pelo mesmo poderá ser visualizado na tela.

Passo 6: Selecionar o arquivo de pontos gerado pelo robô com extensão **.lsv** (por exemplo, **TesteIdentificacao.lsv**) clicando sobre o botão **...**.

Passo 7: Selecionar a opção **Não** para **Palete inclinado** (figura C19). No caso do palete estar inclinado em relação ao referencial zero do robô, o novo valor correspondente a orientação do palete em relação ao robô (assumido zero por *default*) deverá ser estimado e introduzido nos parâmetros Psi, Teta, Phi,.

Passo 8: Após selecionar o arquivo de pontos, clicar em **dados manual** e entrar com os valores da altura do pino (circulo em vermelho na figura abaixo). Para confirmar a gravação de uma entrada basta digitá-la na grid na célula desejada e clicar sobre outra célula qualquer da grid. Caso a altura do pino seja constante não é necessário entrar os valores da altura do pino (figura C20).

Passo 9: Clique em **CALCULAR**, após o calculo serão apresentados aos valores do novo zero do robô (em azul), e o erro final obtido (figura C21).

```
--
-- File A:\C2\PROG_99.VAR   Thu Dec 16 09:56:08 2004
-- SW Version 5.21
--
--$$   ARM: 1 AXES: 6 AUX:0 MASK: 63

pnt0019p POS  Priv
 X:  546.30 Y:-136.80 Z: 775.00 A:  179.79 E: -0.42 R:-134.43
 CNFG: ''

pnt0020p POS  Priv
 X:697.20 Y:-137.50 Z: 774.90 A:  179.79 E: -0.42 R:-134.43
 CNFG: 'T2:-1'

pnt0021p POS  Priv
 X:697.50 Y:-287.60 Z: 776.20 A:  179.79 E: -0.42 R:-134.43
 CNFG: ''

pnt0024p POS  Priv
 X: 546.90 Y:-287.80 Z: 776.80 A:  179.79 E: -0.42 R:-134.43
 CNFG: ''
```

Figura C18: Arquivo de pontos de calibração obtidos no robô.

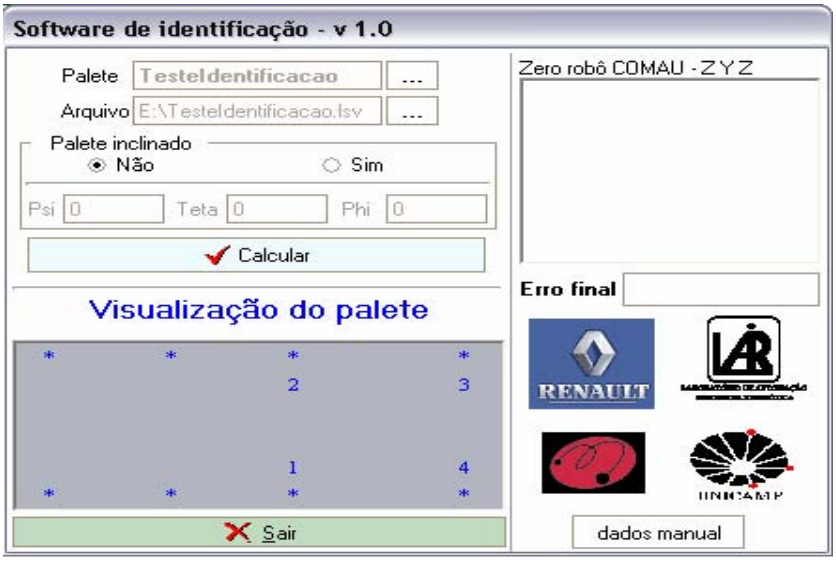


Figura C19: Tela de visualização do Palete.

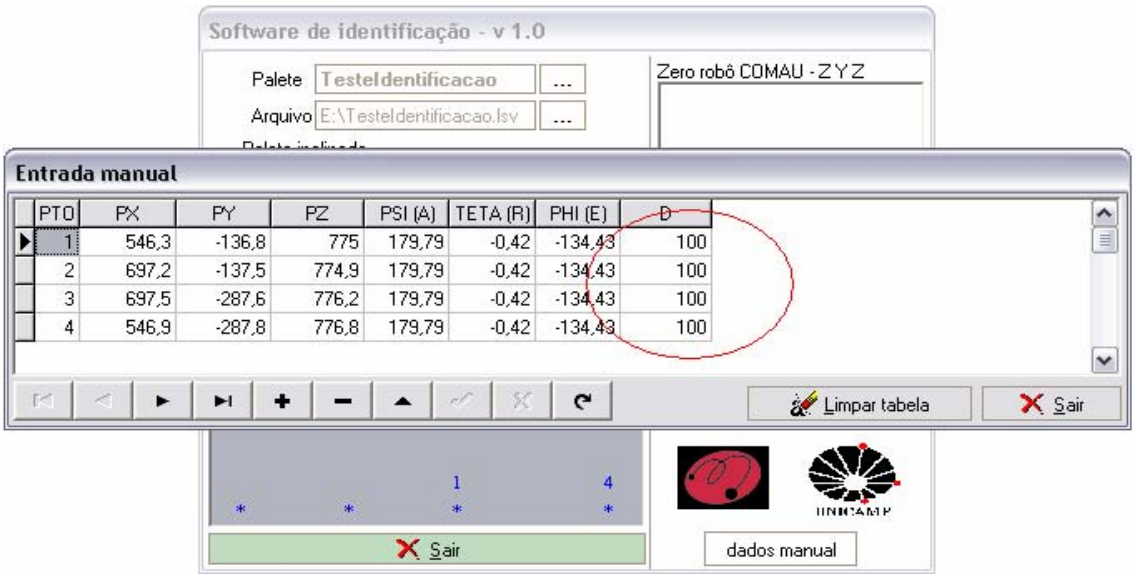


Figura C20: Tela de entrada de dados do arquivo *.lsv do robô COMAU.

Figura C21: Tela de resultados – obtenção do zero do robô.

ANEXO IV

Programas Computacionais Implementados para Integração de Robôs Cooperativos

D.1 **Descritivo dos programas computacionais implementados**

Neste anexo é apresentado inicialmente o descritivo funcional das Entradas e Saídas do Sistema de Supervisão e Controle implementado na aplicação de Integração de Robôs Cooperativos, apresentado detalhadamente no capitulo V desta dissertação. A seguir são apresentados o conjunto de programas computacionais implementados nos robôs ABB IRB 1400 e IRB 140 e CLP Koyo, com sistema de supervisão e controle implementado em software LabVIEW.

D.2 **Especificações dos blocos funcionais implementados na CLP Koyo e Supervisor implementado em LABVIEW**

Tela 1: Inicio de Programa - Modos de Operação (S_0)

		Entradas		Saídas	
Menu	**Lógica de Comando**	C_1	C_2	S_1	S_2
1	**Menu de Inicio de Programa**	**0**	**0**	**0**	**0**
2	**Modo Movimenta Mesa**	**0**	**1**	**1**	**0**
3	**Modo Movimenta Robô**	**1**	**0**	**0**	**1**
4	**Sair do Programa de Supervisão**	**x**	**x**	**0**	**0**

$\mathbf{C_0} = 1$ (sistema em funcionamento)
$\mathbf{S_0} = 1$

Tela 2: Modo de Operação: Movimentação da Mesa (S_1)

		Entradas			Saídas				
Menu	**Lógica de Comando**	C_3	C_4	C_5	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}
	Modo Movimenta Mesa	0	0	0	0	0	0	0	0
1	**Movimento de Translação da Mesa**	1	1	1	1	0	0	0	0
2	**Rot. R1 Mesa (Face 1 e 3)**	0	0	1	0	1	0	0	0
3	**Rot. R1 Mesa (Face 2 e 4)**	0	1	0	0	0	1	0	0
4	**Rot. R2 Mesa (inclin.) – Robô 1**	1	0	1	0	0	0	1	0
5	**Rot. R2 Mesa (inclin.) – Robô 2**	1	1	0	0	0	0	0	1
6	**Retornar ao Menu Inicial**	x	x	X	x	x	x	x	x

$\mathbf{C_{30}} = 1$: enviar valores das variáveis $\mathbf{C_3}$, $\mathbf{C_4}$, $\mathbf{C_5}$ para CLP
$\mathbf{S_1} = 1$
$\mathbf{C_1}$, $\mathbf{C_2} = 0$ – retorno ao menu inicial ($\mathbf{S_0}$)

Tela 3: Modo de Movimentação dos Robôs (S_2)

Menu	**Lógica de Comando**	C_3	C_4	S_3	S_4	S_5
	Modo Robôs	0	0	0	0	0
1	**Init_Rob**	0	1	1	0	0
2	**Task1 (Faces)**	1	0	0	1	0
3	**Task2 (Mesa Inclinada)**	1	1	0	0	1
4	**Retornar ao Menu Inicial**	X	x	x	x	X

$\mathbf{C_{30}} = 1$: enviar valores das variáveis $\mathbf{C_3}$, $\mathbf{C_4}$ para CLP
$\mathbf{S_2} = 1$
$\mathbf{C_1}$, $\mathbf{C_2} = 0$ – retorno ao menu inicial ($\mathbf{S_0}$)

Tela 4: Modo Init_Rob – Posição Inicial – Robôs (S_3)

		Entradas				Saídas		
Menu	**Lógica de Comando**	**C_5**	**C_5**	**C_{15}**	**C_{16}**	**S_{31}**	**S_{32}**	**S_{33}**
	Init Rob	**X**	**x**	**x**	**x**	**0**	**0**	**0**
1	**Pos Inicial – Robô (R1) - $Traj_{11}$**	**0**	**0**	**1**	**0**	**1**	**0**	**0**
2	**Pos Inicial – Robô (R2) – $Traj_{21}$**	**1**	**1**	**0**	**1**	**0**	**1**	**0**
3	**Pos Cooperativo Robôs (R1 e R2)**	**1**	**1**	**1**	**1**	**0**	**0**	**1**
4	**Retornar ao Menu Inicial**	**X**	**x**	**x**	**X**	**X**	**x**	**x**

$\mathbf{C_{40}} = 1$: enviar valores das variáveis $\mathbf{C_5}$, $\mathbf{C_6}$ para CLP
$\mathbf{S_3} = 1$
$\mathbf{C_3}$, $\mathbf{C_4} = 0$ – retorno ao menu anterior: Modo Movimentação Robôs ($\mathbf{S_2}$)

Tela 5: Modo Task 1 – Trajetórias – faces 1 e 3 (S_4)

		Entradas				Saídas		
Menu	**Lógica de Comando**	**C_5**	**C_6**	**C_{15}**	**C_{16}**	**S_{41}**	**S_{42}**	**S_{43}**
	Task 1 (Faces)	**x**	**x**	**x**	**x**	**0**	**0**	**0**
1	**Traj Robô – Face (R1) - $Traj_{12}$**	**0**	**1**	**1**	**0**	**1**	**0**	**0**
2	**Traj Robô – Face (R2) – $Traj_{22}$**	**1**	**0**	**0**	**1**	**0**	**1**	**0**
3	**Traj Cooperativa (R1 e R2)**	**1**	**1**	**1**	**1**	**0**	**0**	**1**
4	**Retornar ao Menu Inicial**	**x**	**x**	**x**	**x**	**x**	**x**	**x**

$\mathbf{C_{40}} = 1$: enviar valores das variáveis $\mathbf{C_5}$, $\mathbf{C_6}$ para CLP
$\mathbf{S_4} = 1$
$\mathbf{C_3}$, $\mathbf{C_4} = 0$ – retorno ao menu anterior: Modo Movimentação Robôs ($\mathbf{S_2}$)

Tela 6: Modo Task 2 - Trajetórias – face superior (S_5)

		Entradas				Saídas		
Menu	Lógica de Comando	C_5	C_6	C_{15}	C_{16}	S_{51}	S_{52}	S_{53}
	Task2 (Mesa Inclinada)	x	x	X	X	0	0	0
1	Rot. R2 Mesa (inclin.) – Robô 1	0	1	1	0	1	0	0
2	Rot. R2 Mesa (inclin.) – Robô 2	1	0	0	1	0	1	0
4	Retornar ao Menu Inicial	x	x	x	x	x	x	X

$\mathbf{C_{40}} = 1$: enviar valores das variáveis $\mathbf{C_5}$, $\mathbf{C_6}$ para CLP
$\mathbf{S_5} = 1$
$\mathbf{C_3}$, $\mathbf{C_4} = 0$ – retorno ao menu anterior: Modo Movimentação Robôs ($\mathbf{S_2}$)

Comunicação Supervisor-CLP (entradas e saídas)

Modo Movimentação da Mesa (S_1)

		Entradas				Saídas			Rot
Menu	Lógica de Comando	X_0	X_1	X_2	X_3	Y_0	Y_1	Y_2	S
	Modo Movimenta Mesa	1	0	0	0	1	0	0	
1	Movimento de Translação da Mesa	1	1	0	0	0	1	1	S_{11}
2	Rot. R1 Mesa (Face 1 e 3)	1	0	1	0	1	0	1	S_{12}
3	Rot. R1 Mesa (Face 2 e 4)	1	0	0	1	1	1	0	S_{13}
4	Rot. R2 Mesa (inclin.) – Robô 1	1	1	1	0	0	0	1	S_{14}
5	Rot. R2 Mesa (inclin.) – Robô 2	1	1	0	1	0	1	0	S_{15}

$\mathbf{C_0} = 1$ (sistema em funcionamento)

Modo Posição Inicial Robôs (S_3)

		Entradas				Saídas			Rot
Menu	Lógica de Comando	X_4	X_5	X_6	X_7	Y_3	Y_4	Y_5	S
1	$Traj_{11}$: Pos Inicial do Robô R1	1	1	1	0	1	1	1	S_{31}
2	$Traj_{21}$: Posição Inicial do Robô R2	0	0	1	0	0	1	1	S_{32}

$\mathbf{C_0} = 1$ (sistema em funcionamento)

Modo Task 1 – Trajetórias – faces 1 e 3 (S_4)

		Entradas				Saídas			Rot
Menu	Lógica de Comando	X_4	X_5	X_6	X_7	Y_3	Y_4	Y_5	S
1	$Traj_{12}$: Traj. Robô 1 – Face 1 (2)	1	1	0	1	1	1	0	S_{41}
2	$Traj_{22}$: Traj. Robô 2 – Face 3 (4)	0	0	0	1	0	1	0	S_{42}

$\mathbf{C_0} = 1$ (sistema em funcionamento)

Modo Task 2 - Trajetórias – face superior (S_5)

		Entradas				Saídas			Rot
Menu	Lógica de Comando	X_4	X_5	X_6	X_7	Y_3	Y_4	Y_5	S
1	$Traj_{13}$: Traj. Robô 1 – Mesa Inclin.	1	1	1	1	1	0	1	S_{51}
2	$Traj_{23}$: Traj. Robô 2 – Mesa Inclin.	0	0	1	1	0	0	1	S_{52}

$\mathbf{C_0} = 1$ (sistema em funcionamento)

D.3 **Especificações das variáveis de E/S implementadas na CLP e Robô ABB**

Tarefas (subprogramas):

Nº	Var_Aux	Descrição Funcional
1	S_0	Inicio do Programa
2	S_1	Modo Movimenta Mesa
3	S_{11}	Movimento de Translação da Mesa
4	S_{12}	Rotação R1 Mesa (Face 1 e 3)
5	S_{13}	Rotação R1 Mesa (Face 2 e 4)
6	S_{14}	Rotação R2 Mesa (inclinação) – Robô 1
7	S_{15}	Rotação R2 Mesa (inclinação) – Robô 2
8	S_2	Modo Movimenta Robô
9	S_3	Init_Rob
10	S_{31}	Posição Inicial – Robô (R1) - $Traj_{11}$
11	S_{32}	Posição Inicial – Robô (R2) - $Traj_{21}$
12	S_{33}	Posicionamento Cooperativo Robôs (R1 e R2)
13	S_4	Task1 (Faces)
14	S_{41}	Posição Inicial – Robô (R1) - $Traj_{11}$
15	S_{42}	Posição Inicial – Robô (R2) - $Traj_{21}$
16	S_{43}	Posicionamento Cooperativo Robôs (R1 e R2)
17	S_5	Task2 (Mesa Inclinada)
18	S_{51}	$Traj_{13}$: Trajetória Robô 1 – Mesa Inclinada
19	S_{52}	$Traj_{23}$: Trajetória Robô 2 – Mesa Inclinada

Lógica de comando do supervisor associados aos subprogramas da CLP

C_1 C_2	C_3 C_4	C_5 C_6		
S_0 (0) (inicio)				
	S_1 (1) (mesa)			
		S_{11} (7)		Movimento de Translação da Mesa
		S_{12} (4)		Rotação R1 Mesa (Face 1 e 3)
		S_{13} (2)		Rotação R1 Mesa (Face 2 e 4)
		S_{14} (5)		Rotação R2 Mesa (inclinação) – Robô 1
		S_{15} (3)		Rotação R2 Mesa (inclinação) – Robô 2
	S_2 (2) (Robô)			
		S_3 (2) (Init_Rob)		
			S_{31}(2)	Posição Inicial – Robô (R1) - $Traj_{11}$
			S_{32}(1)	Posição Inicial – Robô (R2) - $Traj_{21}$
			S_{33}(3)	Posicionamento Cooperativo Robôs (R1 e R2)
		S_4 (1) (Task1_Rob)		
			S_{41}(2)	$Traj_{12}$: Trajetória Robô 1 – Face 1 (2)
			S_{42}(1)	$Traj_{22}$: Trajetória Robô 2 – Face 3 (4)
			S_{43}(3)	Tarefa Cooperativa Robôs (R1 e R2)
		S_5 (3) (Task2_Rob)		
			S_{51}(2)	$Traj_{13}$: Trajetória Robô 1 – Mesa Inclinada
			S_{52}(1)	$Traj_{23}$: Trajetória Robô 2 – Mesa Inclinada

Variáveis Auxiliares:

Nº	Var_Aux	Descrição Funcional
1	C_0	Chave Liga-Desliga
2	C_1	Lógica de Comando Supervisor (Modo Mesa e Robô)
3	C_2	Lógica de Comando Supervisor (Modo Mesa e Robô)
4	C_3	Lógica de Comando
5	C_4	Lógica de Comando
6	C_5	Lógica de Comando
7	C_6	Lógica de Comando Supervisor
8	C_7	Variável Auxiliar

Estruturação Lógica (supervisor)

Inicio do Programa – Seleção Modo Movimenta Mesa ou Robô

	1	2		
C_1	C_2	Val	S	Lógica de Comando
0	0	0	S_0	Menu de Inicio de Programa
0	1	1	S_1	Modo Movimenta Mesa
1	0	2	S_2	Modo Movimenta Robô
1	1	3	---	NC

Modo Mesa: Seleção de possibilidades de movimentação da Mesa

		1	2	4			
C_1	C_2	C_3	C_4	C_5	Val	S	Lógica de Comando
0	1	0	0	0	0	S_1	MODO – Movimenta Mesa
0	1	1	1	1	7	S_{11}	Movimento de Translação da Mesa
0	1	0	0	1	4	S_{12}	Rotação R1 Mesa (Face 1 e 3)
0	1	0	1	0	2	S_{13}	Rotação R1 Mesa (Face 2 e 4)
0	1	1	0	1	5	S_{14}	Rotação R2 Mesa (inclinação) – Robô 1
0	1	1	1	0	3	S_{15}	Rotação R2 Mesa (inclinação) – Robô 2
0	0	X	X	X	X	S_0	Menu de Inicio de Programa

Modo Robô: Seleção de possibilidades de movimentação do Robô

		1	2	1	2			
C_1	C_2	C_3	C_4	C_5	C_6	Val	S	Lógica de Comando
0	0	X	X	X	X	X	S_0	Menu de Inicio de Programa
1	0	0	0	X	X	0	S_2	MODO – Movimenta Robô
1	0	0	1	X	X	2	S_3	Init_Rob
1	0	0	1	0	1	2	S_{31}	Posição Inicial – Robô (R1) - $Traj_{11}$
1	0	0	1	1	0	1	S_{32}	Posição Inicial – Robô (R2) - $Traj_{21}$
1	0	0	1	1	1	3	S_{33}	Posicionamento Cooperativo Robôs (R1 e R2)
1	0	1	0	X	X	1	S_4	Task1 (Faces)
1	0	1	0	0	1	2	S_{41}	Posição Inicial – Robô (R1) - $Traj_{11}$
1	0	1	0	1	0	1	S_{42}	Posição Inicial – Robô (R2) - $Traj_{21}$
1	0	1	0	1	1	3	S_{43}	Posicionamento Cooperativo Robôs (R1 e R2)
1	0	1	1	X	X	3	S_5	Task2 (Mesa Inclinada)
1	0	1	1	0	1	2	S_{51}	$Traj_{13}$: Trajetória Robô 1 – Mesa Inclinada
1	0	1	1	1	0	1	S_{52}	$Traj_{23}$: Trajetória Robô 2 – Mesa Inclinada

Entradas e Saídas da CLP (Koyo)

Entradas (sensores):

Nº	Output	Descrição Funcional
1	X_0	Lógica de Comando (mesa)
2	X_1	Lógica de Comando (mesa)
3	X_2	Lógica de Comando (mesa)
4	X_3	Lógica de Comando (mesa)
5	X_4	Lógica de Comando (Robô - Trajetórias)
6	X_5	Lógica de Comando (Robô - Trajetórias)
7	X_6	Lógica de Comando (Robô - Trajetórias)
8	X_7	Lógica de Comando (Robô - Trajetórias)

Lógica de Entradas da CLP (Koyo)

Sinais da mesa

1	2	4	8		
X_0	X_1	X_2	X_3	Val	Ordem
1	0	0	0	1	Modo Mesa
1	1	0	0	3	Posição Inicial da Mesa (translação)
1	0	1	0	5	Rotação 1 da Base da Mesa – Face 1-3
1	0	0	1	9	Rotação 1 da Base da Mesa – Face 2-4
1	1	1	0	7	Rotação 2 da Mesa (inclinação) – Robô 1
1	1	0	1	11	Rotação 2 da Mesa (inclinação) – Robô 2

Sinais dos robôs

16	32	64	128		
X_4	X_5	X_6	X_7	Val	Ordem
1	0	0	0	16	Modo Robô
1	1	1	0	112	$Traj_{11}$: Posição Inicial do Robô R1
1	1	0	1	196	$Traj_{12}$: Trajetória Robô 1 – Face 1 (2)
1	1	1	1	240	$Traj_{13}$: Trajetória Robô 1 – Mesa Inclinada
0	0	1	0	64	$Traj_{21}$: Posição Inicial do Robô R2
0	0	0	1	128	$Traj_{22}$: Trajetória Robô 2 – Face 3 (4)
0	0	1	1	192	$Traj_{23}$: Trajetória Robô 2 – Mesa Inclinada

Saídas (CLP Koyo):

Nº	Output	Descrição Funcional
1	Y_0	Lógica de Comando (mesa)
2	Y_1	Lógica de Comando (mesa)
3	Y_2	Lógica de Comando (mesa)
4	Y_3	Lógica de Comando (Robô - Trajetórias)
5	Y_4	Lógica de Comando (Robô - Trajetórias)
6	Y_5	Lógica de Comando (Robô - Trajetórias)

Lógica de Saídas da CLP (Koyo)

Sinais da mesa

1	2	4		
Y_0	Y_1	Y_2	Val	Ordem
0	0	0	0	Nada
1	0	0	1	Modo Mesa
0	1	1	6	Movimento de Translação da Mesa
1	0	1	5	Rotação 1 da Base da Mesa – Face 1-3
1	1	0	3	Rotação 1 da Base da Mesa – Face 2-4
0	0	1	4	Rotação 2 da Mesa (inclinação) – Robô 1
0	1	0	2	Rotação 2 da Mesa (inclinação) – Robô 2

Sinais dos robôs

8	16	32		
Y_3	Y_4	Y_5	Val	Ordem
1	0	0	8	Modo Robô
1	1	1	56	$Traj_{11}$: Posição Inicial do Robô R1
1	1	0	24	$Traj_{12}$: Trajetória Robô 1 – Face 1 (2)
1	0	1	40	$Traj_{13}$: Trajetória Robô 1 – Mesa Inclinada
0	1	1	48	$Traj_{21}$: Posição Inicial do Robô R2
0	1	0	16	$Traj_{22}$: Trajetória Robô 2 – Face 3 (4)
0	0	1	32	$Traj_{23}$: Trajetória Robô 2 – Mesa Inclinada

Lógica do Programa de Supervisor

Tela 1: Inicio - Modos de Operação

		Entradas			Saídas			SUPERV
Menu	Lógica de Comando	C_0	C_1	C_2	S_0	S_1	S_2	Variáveis
1	Menu de Inicio de Programa	1	0	0	1	0	0	----
2	Modo Movimenta Mesa	1	0	1	0	1	0	AUX0
3	Modo Movimenta Robô	1	1	0	0	0	1	AUX1
4	Sair do Programa de Supervisão	0	x	x	0	0	0	----

Tela 2: Modo de Operação Mesa

Menu	Lógica de Comando	C_0	C_2	C_3	C_4	C_5	S_1	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	Var.
	Modo Movimenta Mesa	1	1	0	0	0	1	0	0	0	0	0	AUX2
1	Movimento de Translação da Mesa	1	1	1	1	1	1	1	0	0	0	0	AUX3
2	Rot. R1 Mesa (Face 1 e 3)	1	1	0	0	1	1	0	1	0	0	0	AUX4
3	Rot. R1 Mesa (Face 2 e 4)	1	1	0	1	0	1	0	0	1	0	0	AUX5
4	Rot. R2 Mesa (inclin.) – Robô 1	1	1	1	0	1	1	0	0	0	1	0	AUX6
5	Rot. R2 Mesa (inclin.) – Robô 2	1	1	1	1	0	1	0	0	0	0	1	AUX7
6	Retornar ao Menu Inicial	0	1	x	x	X	1	x	x	x	x	x	-----

Comunicação Supervisor-CLP (entradas e saídas)

		Entradas				Saídas		
Menu	Lógica de Comando	X_0	X_1	X_2	X_3	Y_0	Y_1	Y_2
	Modo Movimenta Mesa	1	0	0	0	1	0	0
1	Movimento de Translação da Mesa	1	1	0	0	0	1	1
2	Rot. R1 Mesa (Face 1 e 3)	1	0	1	0	1	0	1
3	Rot. R1 Mesa (Face 2 e 4)	1	0	0	1	1	1	0
4	Rot. R2 Mesa (inclin.) – Robô 1	1	1	1	0	0	0	1
5	Rot. R2 Mesa (inclin.) – Robô 2	1	1	0	1	0	1	0

Tela 3: Modo de Operação Robôs

Menu	Lógica de Comando	C_0	C_1	C_3	C_4	S_2	S_3	S_4	S_5	VAR
	Modo Robôs	1	1	0	0	1	0	0	0	----
1	Init_Rob	1	1	0	1	0	1	0	0	AUX8
2	Task1 (Faces)	1	1	1	0	0	0	1	0	AUX9
3	Task2 (Mesa Inclinada)	1	1	1	1	0	0	0	1	AUX10
4	Retornar ao Menu Inicial	0	1	x	x	X	x	x	X	----

Tela 4: Modo Init_Rob

Menu	Lógica de Comando	C_0	C_3	C_4	C_5	C_6	S_3	S_{31}	S_{32}	S_{33}	S_{33}	AUX
	Init Rob	1	0	1	x	x	1	0	0	0	0	----
1	Pos Inicial – Robô (R1) - $Traj_{11}$	1	0	1	0	1	0	1	0	0	0	AUX11
2	Pos Inicial – Robô (R2) – $Traj_{21}$	1	0	1	1	0	0	0	1	0	0	AUX12
3	Pos Cooperativo Robôs (R1 e R2)	1	0	1	1	1	0	0	0	1	1	AUX13
4	Retornar ao Menu Inicial	0	0	1	x	x	x	x	x	x	x	----

Comunicação Supervisor-CLP (entradas e saídas)

		Entradas				Saídas		
Menu	Lógica de Comando	X_4	X_5	X_6	X_7	Y_3	Y_4	Y_5
	Modo Robôs	1	0	0	0	1	0	0
1	$Traj_{11}$: Pos Inicial do Robô R1	1	1	1	0	1	1	1
2	$Traj_{21}$: Posição Inicial do Robô R2	0	0	1	0	0	1	1

Tela 5: Modo Task 1

Menu	Lógica de Comando	C0	C3	C4	C5	C6	S4	S41	S42	S43	AUX
	Task 1 (Faces)	1	1	0	x	x	1	0	0	0	----
1	Pos Inicial – Robô (R1) - $Traj_{11}$	1	1	0	0	1	0	1	0	0	AUX14
2	Pos Inicial – Robô (R2) – $Traj_{21}$	1	1	0	1	0	0	0	1	0	AUX15
3	Pos Cooperativo Robôs (R1 e R2)	1	1	0	1	1	0	0	0	1	AUX16
4	Retornar ao Menu Inicial	0	1	0	x	x	x	x	x	x	----

Comunicação Supervisor-CLP (entradas e saídas)

Entradas	Saídas						
Menu	Lógica de Comando	X_4	X_5	X_6	X_7	Y_3	Y_4	Y_5
1	$Traj_{12}$: Traj. Robô 1 – Face 1 (2)	1	1	0	1	1	1	0
2	$Traj_{22}$: Traj. Robô 2 – Face 3 (4)	0	0	0	1	0	1	0

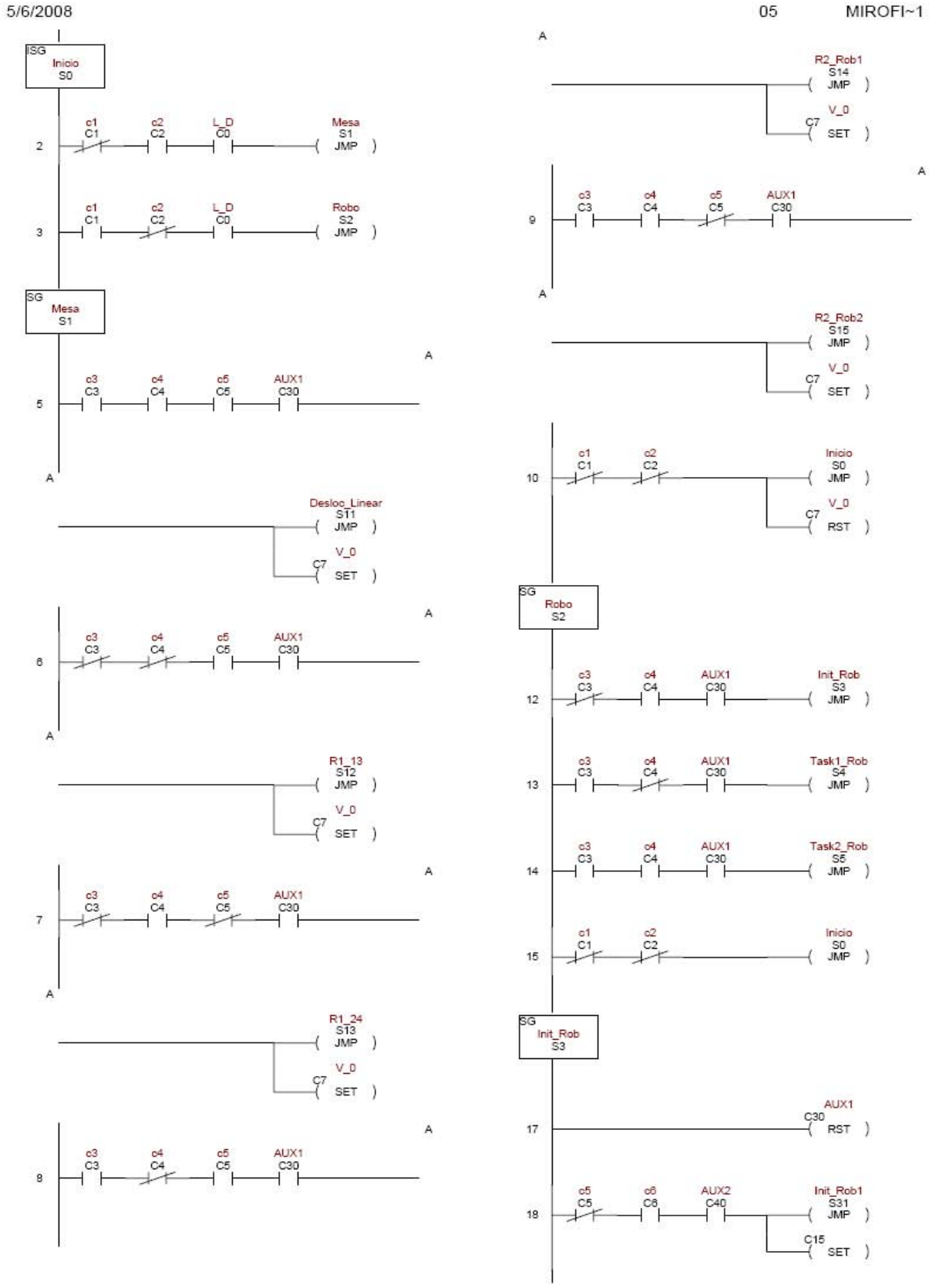
Tela 6: Modo Task 2

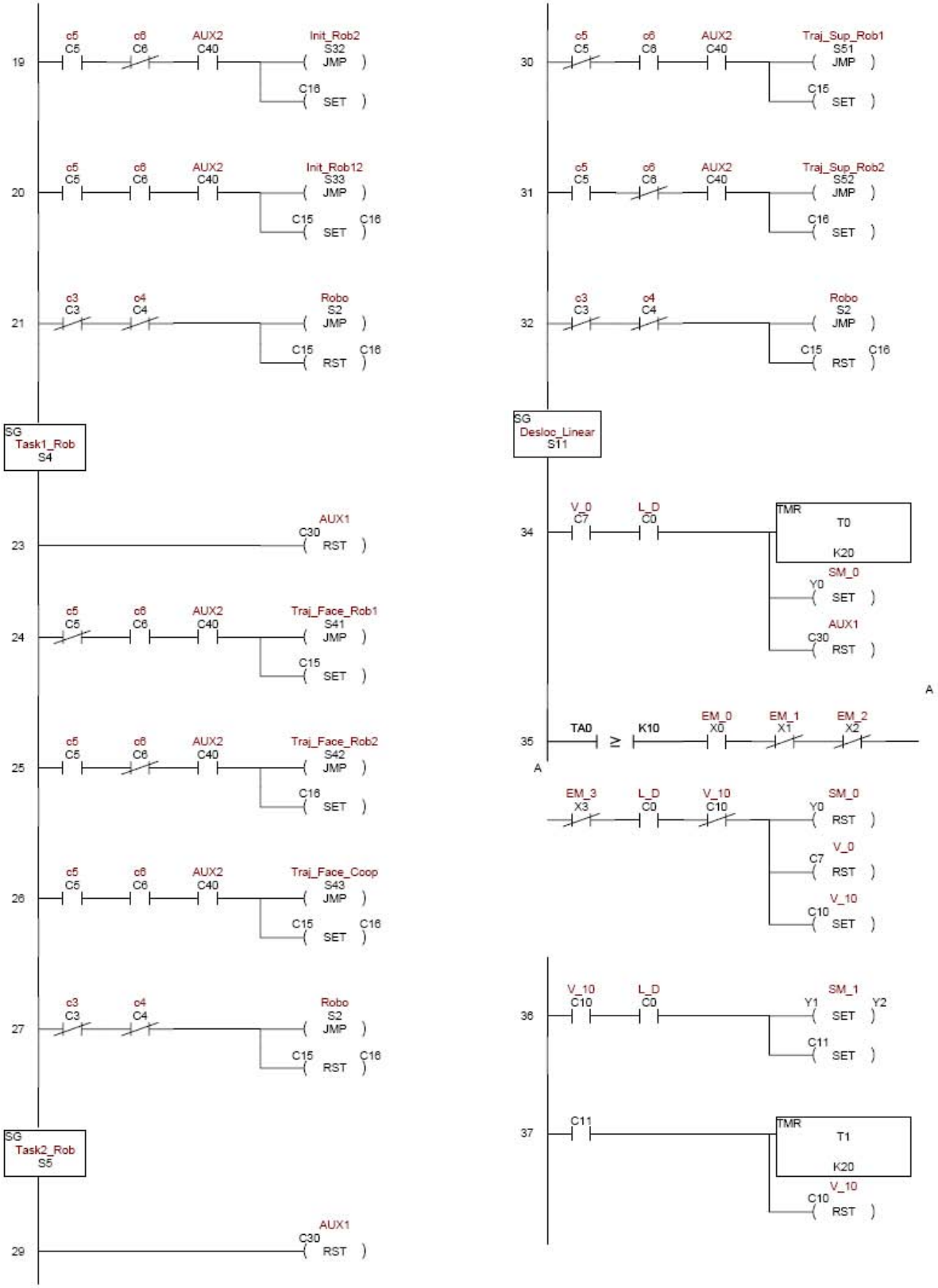
Menu	Lógica de Comando	C_0	C_3	C_4	C_5	C_6	S_5	S_{51}	S_{52}	S_{53}	AUX
	Task2 (Mesa Inclinada)	1	1	1	x	x	1	0	0	0	----
1	Rot. R2 Mesa (inclin.) – Robô 1	1	1	1	0	1	0	1	0	0	AUX17
2	Rot. R2 Mesa (inclin.) – Robô 2	1	1	1	1	0	0	0	1	0	AUX18
4	Retornar ao Menu Inicial	0	1	1	x	x	x	x	x	X	----

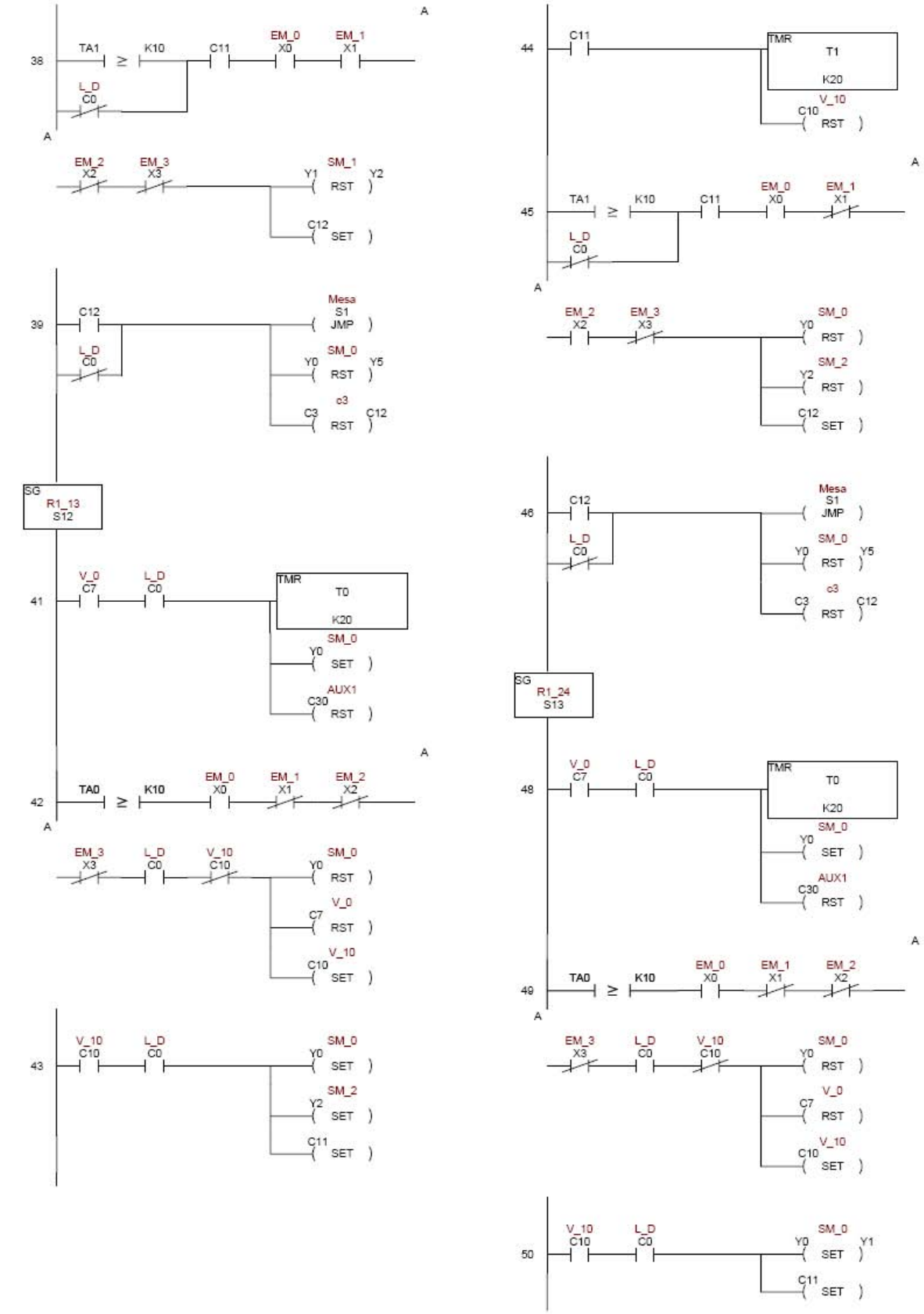
Comunicação Supervisor-CLP (entradas e saídas)

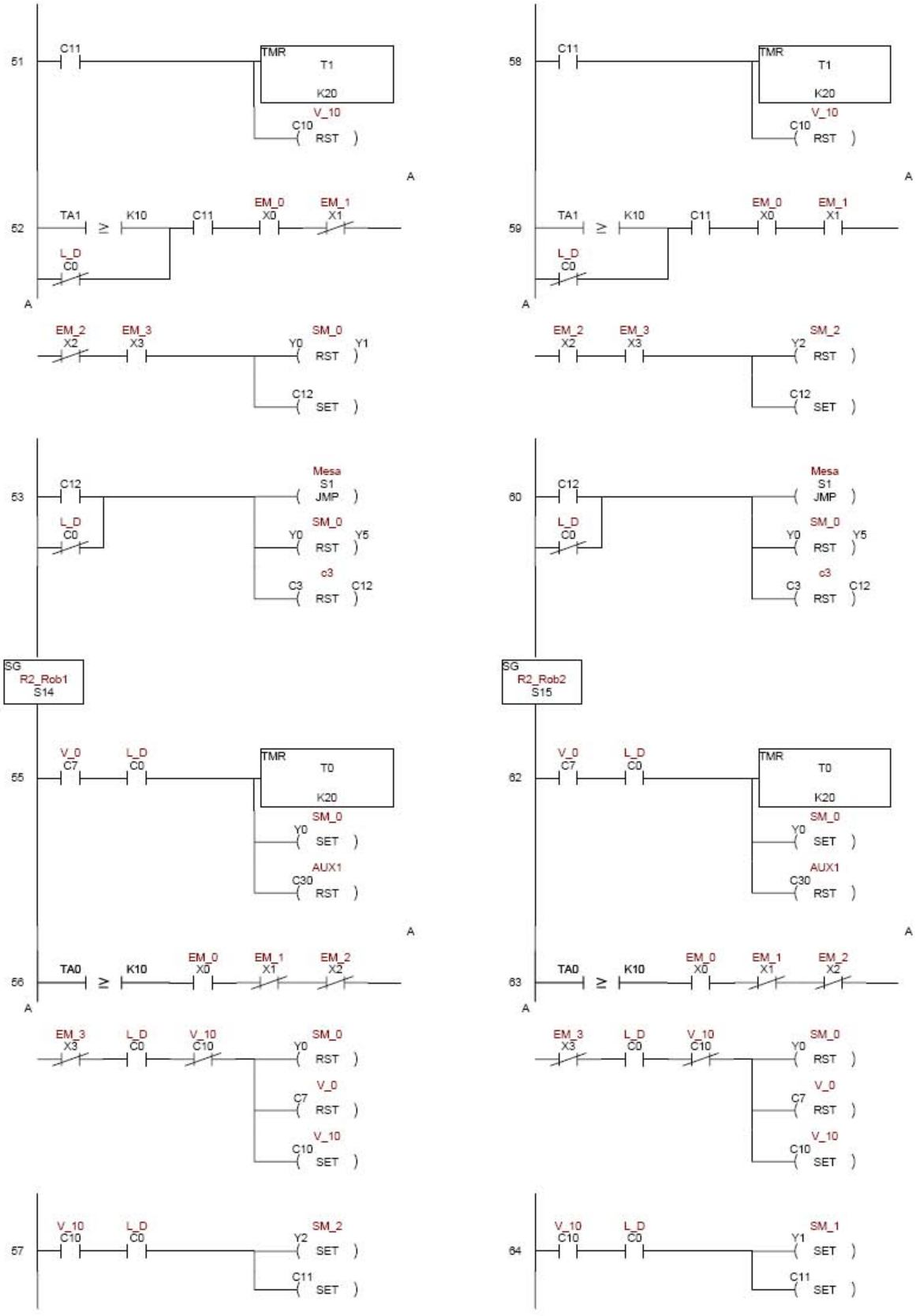
		Entradas				Saídas		
Menu	Lógica de Comando	X_4	X_5	X_6	X_7	Y_3	Y_4	Y_5
1	$Traj_{13}$: Traj. Robô 1 – Mesa Inclin.	1	1	1	1	1	0	1
2	$Traj_{23}$: Traj. Robô 2 – Mesa Inclin.	0	0	1	1	0	0	0

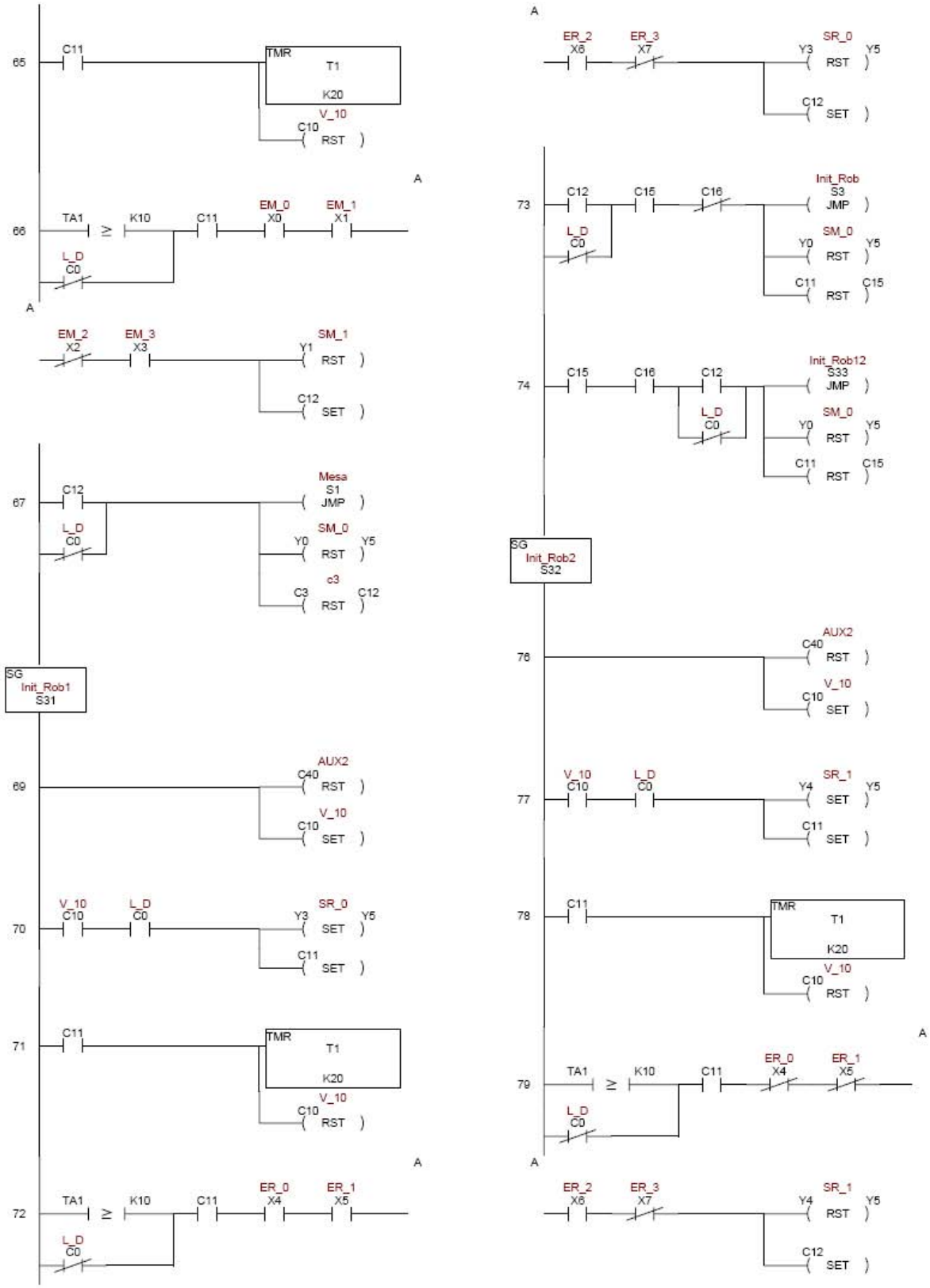
D.4 **Programa implementado na CLP Koyo**

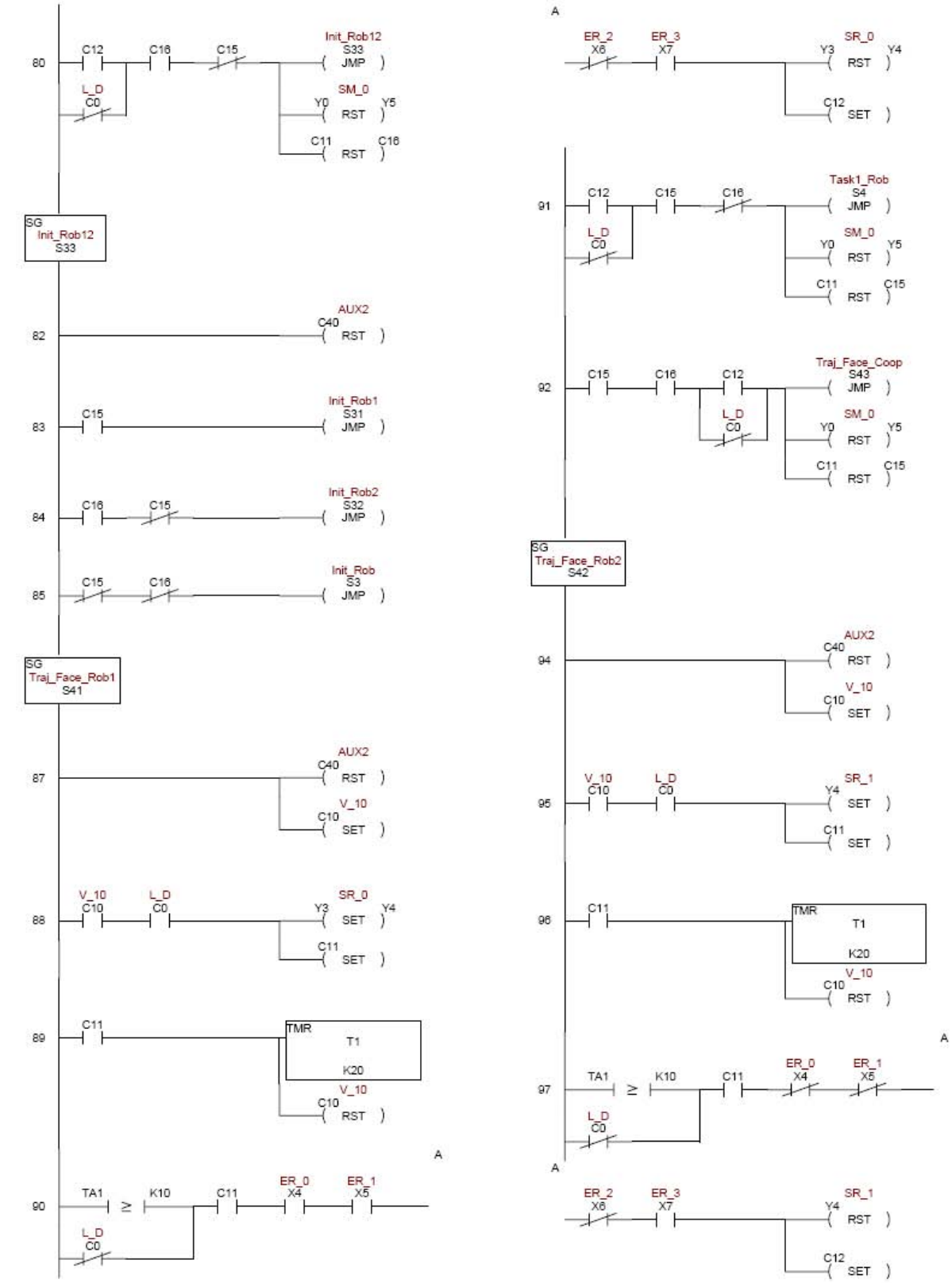


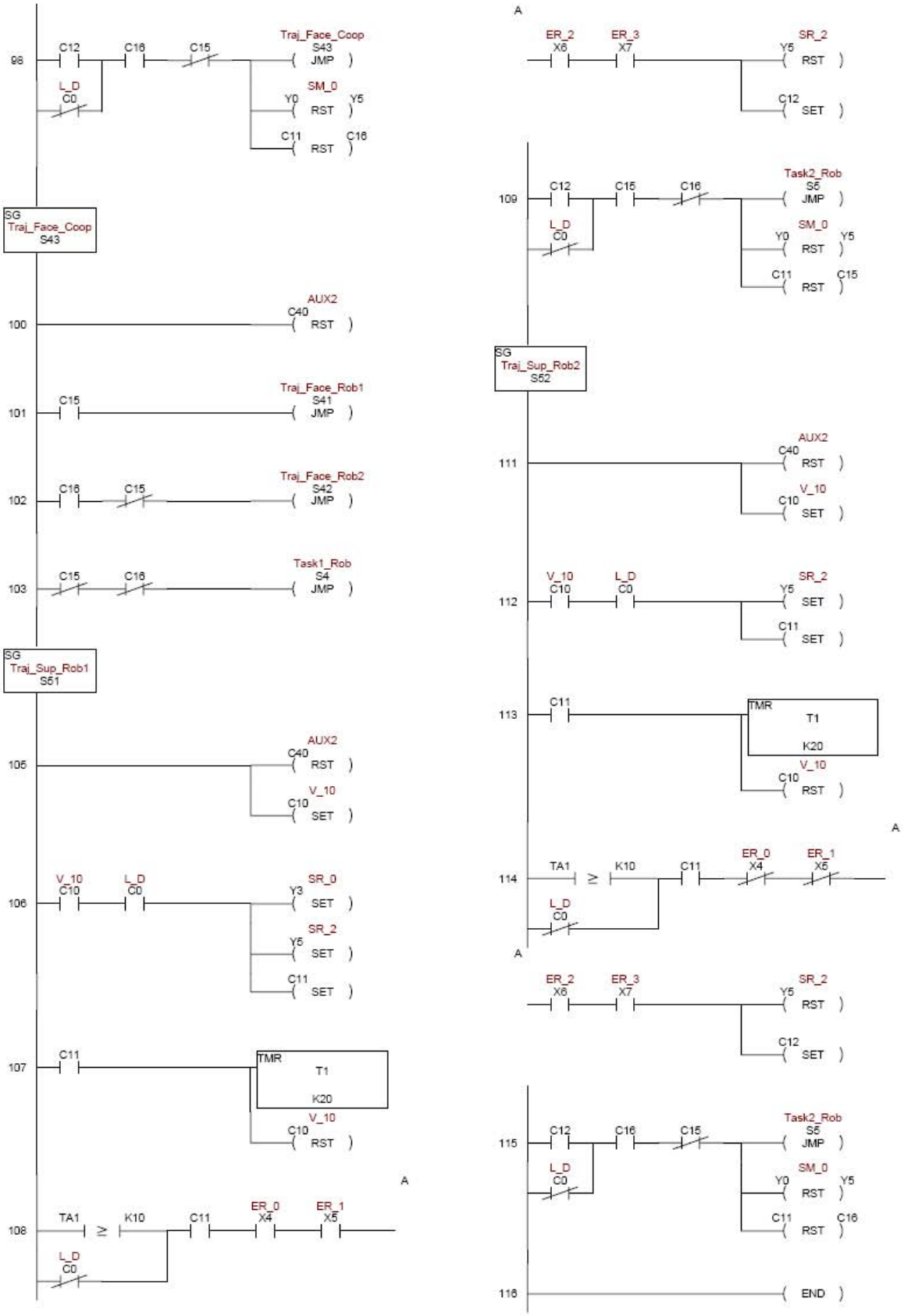

19 c5 C5 c6 C6 AUX2 C40 Init_Rob2 S32 JMP C16 SET
20 c5 C5 c6 C6 AUX2 C40 Init_Rob12 S33 JMP C15 SET C16
21 c3 C3 c4 C4 Robo S2 JMP C15 RST C16
SG Task1_Rob S4
23 AUX1 C30 RST
24 c5 C5 c6 C6 AUX2 C40 Traj_Face_Rob1 S41 JMP C15 SET
25 c5 C5 c6 C6 AUX2 C40 Traj_Face_Rob2 S42 JMP C16 SET
26 c5 C5 c6 C6 AUX2 C40 Traj_Face_Coop S43 JMP C15 SET C16
27 c3 C3 c4 C4 Robo S2 JMP C15 RST C16
SG Task2_Rob S5
29 AUX1 C30 RST
30 c5 C5 c6 C6 AUX2 C40 Traj_Sup_Rob1 S51 JMP C15 SET
31 c5 C5 c6 C6 AUX2 C40 Traj_Sup_Rob2 S52 JMP C16 SET
32 c3 C3 c4 C4 Robo S2 JMP C15 RST C16
SG Desloc_Linear ST1
34 V_0 C7 L_D C0 TMR T0 K20 SM_0 Y0 SET AUX1 C30 RST
A
35 TA0 ≥ K10 EM_0 X0 EM_1 X1 EM_2 X2
A
EM_3 X3 L_D C0 V_10 C10 SM_0 Y0 RST V_0 C7 RST V_10 C10 SET
36 V_10 C10 L_D C0 SM_1 Y1 Y2 SET C11 SET
37 C11 TMR T1 K20 V_10 C10 RST


38
TA1
≥
K10
C11
EM_0
X0
EM_1
X1
L_D
C0
A
EM_2
X2
EM_3
X3
SM_1
Y1
RST
Y2
C12
SET
39
C12
Mesa
S1
JMP
L_D
C0
SM_0
Y0
RST
Y5
c3
C3
RST
C12
SG
R1_13
S12
41
V_0
C7
L_D
C0
TMR
T0
K20
SM_0
Y0
SET
AUX1
C30
RST
42
TA0
≥
K10
EM_0
X0
EM_1
X1
EM_2
X2
EM_3
X3
L_D
C0
V_10
C10
SM_0
Y0
RST
V_0
C7
RST
V_10
C10
SET
43
V_10
C10
L_D
C0
SM_0
Y0
SET
SM_2
Y2
SET
C11
SET
44
C11
TMR
T1
K20
V_10
C10
RST
45
TA1
≥
K10
C11
EM_0
X0
EM_1
X1
L_D
C0
EM_2
X2
EM_3
X3
SM_0
Y0
RST
SM_2
Y2
RST
C12
SET
46
C12
Mesa
S1
JMP
L_D
C0
SM_0
Y0
RST
Y5
c3
C3
RST
C12
SG
R1_24
S13
48
V_0
C7
L_D
C0
TMR
T0
K20
SM_0
Y0
SET
AUX1
C30
RST
49
TA0
≥
K10
EM_0
X0
EM_1
X1
EM_2
X2
EM_3
X3
L_D
C0
V_10
C10
SM_0
Y0
RST
V_0
C7
RST
V_10
C10
SET
50
V_10
C10
L_D
C0
SM_0
Y0
SET
Y1
C11
SET


51
C11
TMR
T1
K20
V_10
C10
RST
A
52
TA1
≥
K10
C11
EM_0
X0
EM_1
X1
L_D
C0
A
EM_2
X2
EM_3
X3
SM_0
Y0
Y1
RST
C12
SET
53
C12
Mesa
S1
JMP
L_D
C0
SM_0
Y0
Y5
RST
c3
C3
C12
RST
SG
R2_Rob1
S14
55
V_0
C7
L_D
C0
TMR
T0
K20
SM_0
Y0
SET
AUX1
C30
RST
A
56
TA0
≥
K10
EM_0
X0
EM_1
X1
EM_2
X2
A
EM_3
X3
L_D
C0
V_10
C10
SM_0
Y0
RST
V_0
C7
RST
V_10
C10
SET
57
V_10
C10
L_D
C0
SM_2
Y2
SET
C11
SET
58
C11
TMR
T1
K20
V_10
C10
RST
A
59
TA1
≥
K10
C11
EM_0
X0
EM_1
X1
L_D
C0
A
EM_2
X2
EM_3
X3
SM_2
Y2
RST
C12
SET
60
C12
Mesa
S1
JMP
L_D
C0
SM_0
Y0
Y5
RST
c3
C3
C12
RST
SG
R2_Rob2
S15
62
V_0
C7
L_D
C0
TMR
T0
K20
SM_0
Y0
SET
AUX1
C30
RST
A
63
TA0
≥
K10
EM_0
X0
EM_1
X1
EM_2
X2
A
EM_3
X3
L_D
C0
V_10
C10
SM_0
Y0
RST
V_0
C7
RST
V_10
C10
SET
64
V_10
C10
L_D
C0
SM_1
Y1
SET
C11
SET


65
C11
TMR
T1
K20
V_10
C10
RST
A
66
TA1
≥
K10
C11
EM_0
X0
EM_1
X1
L_D
C0
A
EM_2
X2
EM_3
X3
SM_1
Y1
RST
C12
SET
67
C12
Mesa
S1
JMP
L_D
C0
SM_0
Y0
Y5
RST
c3
C3
C12
RST
SG
Init_Rob1
S31
69
AUX2
C40
RST
V_10
C10
SET
70
V_10
C10
L_D
C0
SR_0
Y3
Y5
SET
C11
SET
71
C11
TMR
T1
K20
V_10
C10
RST
A
72
TA1
≥
K10
C11
ER_0
X4
ER_1
X5
L_D
C0
A
ER_2
X6
ER_3
X7
SR_0
Y3
Y5
RST
C12
SET
73
C12
C15
C16
Init_Rob
S3
JMP
L_D
C0
SM_0
Y0
Y5
RST
C11
C15
RST
74
C15
C16
C12
Init_Rob12
S33
JMP
L_D
C0
SM_0
Y0
Y5
RST
C11
C15
RST
SG
Init_Rob2
S32
76
AUX2
C40
RST
V_10
C10
SET
77
V_10
C10
L_D
C0
SR_1
Y4
Y5
SET
C11
SET
78
C11
TMR
T1
K20
V_10
C10
RST
A
79
TA1
≥
K10
C11
ER_0
X4
ER_1
X5
L_D
C0
A
ER_2
X6
ER_3
X7
SR_1
Y4
Y5
RST
C12
SET


80
C12
C16
C15
Init_Rob12
S33
JMP
L_D
C0
SM_0
Y0
RST
Y5
C11
RST
C16
SG
Init_Rob12
S33
82
AUX2
C40
RST
83
C15
Init_Rob1
S31
JMP
84
C16
C15
Init_Rob2
S32
JMP
85
C15
C16
Init_Rob
S3
JMP
SG
Traj_Face_Rob1
S41
87
AUX2
C40
RST
V_10
C10
SET
88
V_10
C10
L_D
C0
SR_0
Y3
SET
Y4
C11
SET
89
C11
TMR
T1
K20
V_10
C10
RST
A
90
TA1
≥
K10
C11
ER_0
X4
ER_1
X5
L_D
C0
A
ER_2
X6
ER_3
X7
SR_0
Y3
RST
Y4
C12
SET
91
C12
C15
C16
Task1_Rob
S4
JMP
L_D
C0
SM_0
Y0
RST
Y5
C11
RST
C15
92
C15
C16
C12
Traj_Face_Coop
S43
JMP
L_D
C0
SM_0
Y0
RST
Y5
C11
RST
C15
SG
Traj_Face_Rob2
S42
94
AUX2
C40
RST
V_10
C10
SET
95
V_10
C10
L_D
C0
SR_1
Y4
SET
C11
SET
96
C11
TMR
T1
K20
V_10
C10
RST
A
97
TA1
≥
K10
C11
ER_0
X4
ER_1
X5
L_D
C0
A
ER_2
X6
ER_3
X7
SR_1
Y4
RST
C12
SET


98
C12
C16
C15
Traj_Face_Coop
S43
JMP
L_D
C0
SM_0
Y0
Y5
RST
C11
C16
SG
Traj_Face_Coop
S43
100
AUX2
C40
RST
101
C15
Traj_Face_Rob1
S41
JMP
102
C16
C15
Traj_Face_Rob2
S42
JMP
103
C15
C16
Task1_Rob
S4
JMP
SG
Traj_Sup_Rob1
S51
105
AUX2
C40
RST
V_10
C10
SET
106
V_10
C10
L_D
C0
SR_0
Y3
SET
SR_2
Y5
SET
C11
SET
107
C11
TMR
T1
K20
V_10
C10
RST
A
108
TA1
≥
K10
C11
ER_0
X4
ER_1
X5
L_D
C0
A
ER_2
X6
ER_3
X7
SR_2
Y5
RST
C12
SET
109
C12
C15
C16
Task2_Rob
S5
JMP
L_D
C0
SM_0
Y0
Y5
RST
C11
C15
SG
Traj_Sup_Rob2
S52
111
AUX2
C40
RST
V_10
C10
SET
112
V_10
C10
L_D
C0
SR_2
Y5
SET
C11
SET
113
C11
TMR
T1
K20
V_10
C10
RST
A
114
TA1
≥
K10
C11
ER_0
X4
ER_1
X5
L_D
C0
A
ER_2
X6
ER_3
X7
SR_2
Y5
RST
C12
SET
115
C12
C16
C15
Task2_Rob
S5
JMP
L_D
C0
SM_0
Y0
Y5
RST
C11
C16
116
END


117
(NOP)

D.5 **Programa implementado no robô ABB IRB 140**

```
%%%
VERSION:1
LANGUAGE:ENGLISH
%%%
MODULE SER140
CONST robtarget p1_1:=[[-76.68,98.87,105.59],[0.128567,-
0.089171,0.904031,-0.397802],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+
09,9E+09]];
PERS wobjdata sgua1:=[FALSE,TRUE,"",[[696.657,-93.3763,371.144],
[0.651924,0.651938,-0.273821,-0.27385]],[[1.06305,1.72323,-0.33319],
[0.999997,0.002284,2.4E-05,8E-06]]];
PERS wobjdata sgua2:=[FALSE,TRUE,"",[[680.569,84.2146,529.411],
[0.25455,0.252133,-0.657116,-0.663198]],[[0,0,0],[1,0,0,0]]];
PERS wobjdata sgua5:=[FALSE,TRUE,"",[[443.149,-92.6114,582.218],
[0.854368,0.149149,-0.345695,-0.358198]],[[0,0,0],[1,0,0,0]]];
CONST robtarget p5_1:=[[12.61,66.1,-98.09],[0.007483,-
0.387995,0.921632,-0.000787],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+
09,9E+09]];
CONST robtarget p5:=[[25.61,78.15,-219.28],[0.007467,-
0.387989,0.921634,-0.000799],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+
09,9E+09]];
CONST robtarget p2_1:=[[-16.63,26.44,2.2],[0.367924,0.82536,-
0.417823,-0.094],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p2:=[[-50.47,-60.32,-143.08],[0.367927,0.825353,-
0.417832,-0.094012],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p1:=[[21.91,98.41,5.04],[0.128568,-0.089182,0.904027,-
0.397808],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
PERS tooldata pincel:=[TRUE,[[7.86115,-30.1179,124.792],[1,0,0,0]],
[0.2,[85,0,65],[1,0,0,0],0.01,0.01,0.01]];
PERS tooldata Bic:=[TRUE,[[0.500839,-0.574904,226.276],[1,0,0,0]],
[0.25,[85,0,65],[1,0,0,0],0.01,0.01,0.01]];
!
_____________________________________________________________________
_____________________________________________________________________
PROC pOS_INIT_R1()
MoveAbsJ [[-120,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v300,z50,Bic\WObj:=sgua1;
ENDPROC
!
_____________________________________________________________________
_____________________________________________________________________
PROC R1FACE1()
!
! Circulo
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua1;
MoveL p1_1,v100,z50,Bic\WObj:=sgua1;
MoveL Offs(p1,0,0,20),v100,z0,Bic\WObj:=sgua1;
MoveL p1,v100,z0,Bic\WObj:=sgua1;
MoveC Offs(p1,75,75,0),Offs(p1,150,0,0),v100,z0,Bic\WObj:=sgua1;
MoveC Offs(p1,75,-75,0),Offs(p1,0,0,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,0,0,20),v100,z0,Bic\WObj:=sgua1;
!
! Triangulo
```

```
!
MoveJ Offs(p1,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,139.952,-37.5,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,113.971,7.5,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,100.981,30,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,49.0192,30,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,75,75,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua1;
!
! Retas Quebradas do Triangulo
!
MoveL Offs(p1,62.0096,52.5,20),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,87.9904,52.5,20),v100,z0,Bic\WObj:=sgua1;
!
MoveL Offs(p1,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,100.981,30,0),v100,z0,Bic\WObj:=sgua1;
!
MoveL Offs(p1,100.981,30,20),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,49.0192,30,20),v100,z0,Bic\WObj:=sgua1;
!
MoveL Offs(p1,49.0192,30,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua1;
!
MoveL Offs(p1,36.0289,7.5,20),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,113.971,7.5,20),v100,z0,Bic\WObj:=sgua1;
!
MoveL Offs(p1,113.971,7.5,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua1;
!
MoveL Offs(p1,126.989,-15.0478,20),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,23.0109,-15.0478,20),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua1;
MoveL Offs(p1,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua1;
!
MoveL Offs(p1,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua1;
MoveL p1_1,v100,z50,Bic\WObj:=sgua1;
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua1;
ENDPROC
!
```

```
PROC R1FACE2()
!
! Circulo
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua2;
! MoveAbsJ [[0,0,0,-20,15,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]]\NoEOffs,v200,z50,Bic\WObj:=sgua2;
MoveJ p2_1,v100,z50,Bic\WObj:=sgua2;
MoveL Offs(p2,0,0,20),v100,z0,Bic\WObj:=sgua2;
```

```
MoveL p2,v100,z0,Bic\WObj:=sgua2;
MoveC Offs(p2,75,75,0),Offs(p2,150,0,0),v100,z0,Bic\WObj:=sgua2;
MoveC Offs(p2,75,-75,0),Offs(p2,0,0,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,0,0,20),v100,z0,Bic\WObj:=sgua2;
!
! Triangulo
!
MoveJ Offs(p2,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,139.952,-37.5,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,113.971,7.5,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,100.981,30,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,49.0192,30,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,75,75,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua2;
!
! Retas Quebradas do Triangulo
!
MoveL Offs(p2,62.0096,52.5,20),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,87.9904,52.5,20),v100,z0,Bic\WObj:=sgua2;
!
MoveL Offs(p2,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,100.981,30,0),v100,z0,Bic\WObj:=sgua2;
!
MoveL Offs(p2,100.981,30,20),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,49.0192,30,20),v100,z0,Bic\WObj:=sgua2;
!
MoveL Offs(p2,49.0192,30,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua2;
!
MoveL Offs(p2,36.0289,7.5,20),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,113.971,7.5,20),v100,z0,Bic\WObj:=sgua2;
!
MoveL Offs(p2,113.971,7.5,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua2;
!
MoveL Offs(p2,126.989,-15.0478,20),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,23.0109,-15.0478,20),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua2;
MoveL Offs(p2,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua2;
!
MoveL Offs(p2,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua2;
MoveJ p2_1,v100,z50,Bic\WObj:=sgua2;
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua2;
!
ENDPROC
!
_____________________________________________________________________________
_____________________________
PROC R1_MES_INC()
!
```

```
! Circulo
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua5;
! MoveAbsJ [[0,0,0,0,45,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua5;
MoveJ p5_1,v400,z50,Bic\WObj:=sgua5;
MoveL Offs(p5,0,0,20),v100,z0,Bic\WObj:=sgua5;
MoveL p5,v100,z0,Bic\WObj:=sgua5;
MoveC Offs(p5,75,75,0),Offs(p5,150,0,0),v100,z0,Bic\WObj:=sgua5;
MoveC Offs(p5,75,-75,0),Offs(p5,0,0,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,0,0,20),v100,z0,Bic\WObj:=sgua5;
!
! Triangulo
!
MoveJ Offs(p5,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua5;
! MoveL Offs(p5,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,139.952,-37.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,113.971,7.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,100.981,30,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,49.0192,30,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,75,75,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua5;
!
! Retas Quebradas do Triangulo
!
MoveL Offs(p5,62.0096,52.5,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,87.9904,52.5,20),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,100.981,30,0),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,100.981,30,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,49.0192,30,20),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,49.0192,30,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,36.0289,7.5,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,113.971,7.5,20),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,113.971,7.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,126.989,-15.0478,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,23.0109,-15.0478,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua5;
MoveJ p5_1,v400,z50,Bic\WObj:=sgua5;
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
```

```
\NoEOffs,v200,z50,Bic\WObj:=sgua5;
!
!
ENDPROC
!
!
************************************************************************
************************
PROC main()
!
! INICIALIZAÇAO DO ROBO
!
! Posicao Inicial Robo R1
!
IF DI10_3=1 AND DI10_4=1 AND DI10_5=1 THEN
!
pOS_INIT_R1;
!
Set DO10_4;
Set DO10_5;
Set DO10_6;
WaitTime 2;
Reset DO10_4;
Reset DO10_5;
Reset DO10_6;
ENDIF
!
! Trajetoria Robo 1 - Face 1(2)
!
IF DI10_4=1 AND DI10_5=0 AND DI10_3=1 THEN
!
R1FACE1;
R1FACE2;
!
Set DO10_4;
Set DO10_5;
Set DO10_7;
WaitTime 2;
Reset DO10_4;
Reset DO10_5;
Reset DO10_7;
ENDIF
!
! Trajetoria Robo 1 - Mesa Inclinada
!
IF DI10_5=1 AND DI10_3=1 AND DI10_4=0 THEN
!
R1_MES_INC;
!
Set DO10_4;
Set DO10_5;
Set DO10_6;
Set DO10_7;
WaitTime 2;
Reset DO10_4;
Reset DO10_5;
Reset DO10_6;
```

```
Reset DO10_7;
ENDIF
ENDPROC
ENDMODULE
```

D.6 **Programa implementado no robô ABB IRB 1400**

```
%%%
VERSION:1
LANGUAGE:ENGLISH
%%%
MODULE SER1400
PERS wobjdata sgua3:=[FALSE,TRUE,"",[[1383.42,183.711,811.144],
[0.649735,0.652177,-0.28073,-0.271478]],[[1.06305,1.72323,-0.33319],
[0.999997,0.002284,2.4E-05,8E-06]]];
PERS wobjdata sgua4:=[FALSE,TRUE,"",[[1529.4,317.814,815.56],
[0.255224,0.262435,-0.667034,-0.648888]],[[0,0,0],[1,0,0,0]]];
PERS wobjdata sgua5:=[FALSE,TRUE,"",[[1275.91,183.04,846.869],
[0.845215,0.136804,-0.364511,-0.3661]],[[0,0,0],[1,0,0,0]]];
CONST robtarget p5_1:=[[-13.45,77.89,137.49],[0.013321,0.390585,-
0.92037,-0.013705],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p5:=[[19.1,98.98,20.1],[0.013349,0.390597,-0.920364,-
0.013697],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p4_1:=[[116.92,110.47,199.54],[0.306494,0.783191,-
0.494542,-0.219323],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p4:=[[60.72,105.56,21.71],[0.306492,0.783199,-
0.494532,-0.219322],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p3:=[[28.04,102.06,21.56],[0.156073,-
0.373344,0.879487,-0.250515],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+
09,9E+09]];
PERS tooldata pincel:=[TRUE,[[7.86115,-30.1179,124.792],[1,0,0,0]],
[0.2,[85,0,65],[1,0,0,0],0.01,0.01,0.01]];
PERS tooldata Bic:=[TRUE,[[0.500839,-0.574904,226.276],[1,0,0,0]],
[0.25,[85,0,65],[1,0,0,0],0.01,0.01,0.01]];
!
_____________________________________________________________________
_____________________________
PROC pOS_INIT_R2()
MoveAbsJ [[120,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v100,z50,Bic\WObj:=sgua3;
ENDPROC
!
_____________________________________________________________________
_____________________________
PROC R2FACE3()
!
! Circulo
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua3;
! MoveJ p3,v400,z50,Bic\WObj:=sgua3;
MoveL Offs(p3,0,0,20),v100,z0,Bic\WObj:=sgua3;
MoveL p3,v100,z0,Bic\WObj:=sgua3;
MoveC Offs(p3,75,75,0),Offs(p3,150,0,0),v100,z0,Bic\WObj:=sgua3;
MoveC Offs(p3,75,-75,0),Offs(p3,0,0,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,0,0,20),v100,z0,Bic\WObj:=sgua3;
!
! Triangulo
!
MoveJ Offs(p3,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua3;
```

```
! MoveL Offs(p3,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,139.952,-37.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,113.971,7.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,100.981,30,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,49.0192,30,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,75,75,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua3;
!
! Retas Quebradas do Triangulo
!
MoveL Offs(p3,62.0096,52.5,20),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,87.9904,52.5,20),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,100.981,30,0),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,100.981,30,20),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,49.0192,30,20),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,49.0192,30,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,36.0289,7.5,20),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,113.971,7.5,20),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,113.971,7.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,126.989,-15.0478,20),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,23.0109,-15.0478,20),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua3;
ENDPROC
!
_____________________________________________________________________________
_____________________________
PROC R2FACE4()
!
! Circulo
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua4;
MoveAbsJ [[0,0,0,-20,15,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua4;
MoveJ p4_1,v100,z50,Bic\WObj:=sgua4;
MoveL Offs(p4,0,0,20),v100,z0,Bic\WObj:=sgua4;
MoveL p4,v100,z0,Bic\WObj:=sgua4;
MoveC Offs(p4,75,75,0),Offs(p4,150,0,0),v100,z0,Bic\WObj:=sgua4;
MoveC Offs(p4,75,-75,0),Offs(p4,0,0,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,0,0,20),v100,z0,Bic\WObj:=sgua4;
!
```

```
! Triangulo
!
MoveJ Offs(p4,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua4;
! MoveL Offs(p4,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,139.952,-37.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,113.971,7.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,100.981,30,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,49.0192,30,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,75,75,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua4;
!
! Retas Quebradas do Triangulo
!
MoveL Offs(p4,62.0096,52.5,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,87.9904,52.5,20),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,100.981,30,0),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,100.981,30,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,49.0192,30,20),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,49.0192,30,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,36.0289,7.5,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,113.971,7.5,20),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,113.971,7.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,126.989,-15.0478,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,23.0109,-15.0478,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua4;
MoveJ p4_1,v100,z50,Bic\WObj:=sgua4;
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua4;
!
ENDPROC
!
_____________________________________________________________________________
_____________________________
PROC R2_MES_INC()
!
! Circulo
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua5;
MoveAbsJ [[0,0,0,0,45,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
```

```
\NoEOffs,v200,z50,Bic\WObj:=sgua5;
MoveJ p5_1,v400,z50,Bic\WObj:=sgua5;
MoveL Offs(p5,0,0,20),v100,z0,Bic\WObj:=sgua5;
MoveL p5,v100,z0,Bic\WObj:=sgua5;
MoveC Offs(p5,75,75,0),Offs(p5,150,0,0),v100,z0,Bic\WObj:=sgua5;
MoveC Offs(p5,75,-75,0),Offs(p5,0,0,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,0,0,20),v100,z0,Bic\WObj:=sgua5;
!
! Triangulo
!
MoveJ Offs(p5,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua5;
! MoveL Offs(p5,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,139.952,-37.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,113.971,7.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,100.981,30,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,49.0192,30,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,75,75,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua5;
!
! Retas Quebradas do Triangulo
!
MoveL Offs(p5,62.0096,52.5,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,87.9904,52.5,20),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,100.981,30,0),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,100.981,30,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,49.0192,30,20),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,49.0192,30,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,36.0289,7.5,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,113.971,7.5,20),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,113.971,7.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,126.989,-15.0478,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,23.0109,-15.0478,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua5;
MoveJ p5_1,v400,z50,Bic\WObj:=sgua5;
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua5;
!
!
ENDPROC
```

```
!
!
*************************************************************************
*********
!
PROC main()
!
! OBS.: WObj:=sgua3 e 4 são as duas faces do quadrado;
! WObj:=sgua5 é a face superior quando inclina o
quadrado
!
! INICIALIZAÇAO DO ROBO
!
!
! Posicao Inicial Robo R2
!
IF DI10_3=0 AND DI10_4=1 AND DI10_5=1 THEN
!
pOS_INIT_R2;
!
Set DO10_6;
!reg1 :=0;
WaitTime 2;
Reset DO10_6;
ENDIF
!
! Trajetoria Robo 2 - Face 3 e 4
!
IF DI10_4=1 AND DI10_5=0 AND DI10_3=0 THEN
!
R2FACE3;
R2FACE4;
!
Set DO10_7;
!reg1 :=0;
WaitTime 2;
Reset DO10_7;
ENDIF
!
! Trajetoria Robo 2 - Mesa Inclinada
!
IF DI10_5=1 AND DI10_3=0 AND DI10_4=0 THEN
!
R2_MES_INC;
!
Set DO10_6;
Set DO10_7;
! reg1 :=0;
WaitTime 2;
Reset DO10_6;
Reset DO10_7;
ENDIF
ENDPROC
ENDMODULE
```

ANEXO V

Programas Computacionais Implementados para Plataforma Robótica WEB

E.1 **Descritivo dos programas computacionais implementados**

Neste anexo é apresentado inicialmente o descritivo funcional das Entradas e Saídas do Sistema de Supervisão e Controle implementado na aplicação da Plataforma Robótica WEB dentro do projeto Kyatera apresentado no capitulo V desta dissertação. A seguir são apresentados o conjunto de programas computacionais implementados no robô IRB 1400 e CLP Koyo, com sistema de supervisão e controle implementado em software LabView.

E.2 **Especificações dos blocos funcionais implementados na CLP Koyo e LABVIEW**

Tela Proposta para o Supervisor

		Entradas				Saídas			
Botões	Lógica de Comando	C_0	C_1	C_2	C_3	S_0	S_1	S_2	S_3
1	Inicio	1	1	0	0	1	0	0	0
2	Reset	1	0	x	x	1	0	0	0
3	Inicialização	1	1	1	0	0	1	0	0
4	Operação Face 1	1	1	0	1	0	0	1	0
5	Operação Face 2	1	1	1	1	0	0	0	1
6	Sair do Programa de Supervisão	0	0	x	x	1	X	X	X

$C_0 = 1$ (sistema em funcionamento)
$S_0 = 1$

E.3 **Especificações das variáveis de E/S implementadas na CLP e Robô ABB**

Comunicação Supervisor-CLP (E/S)

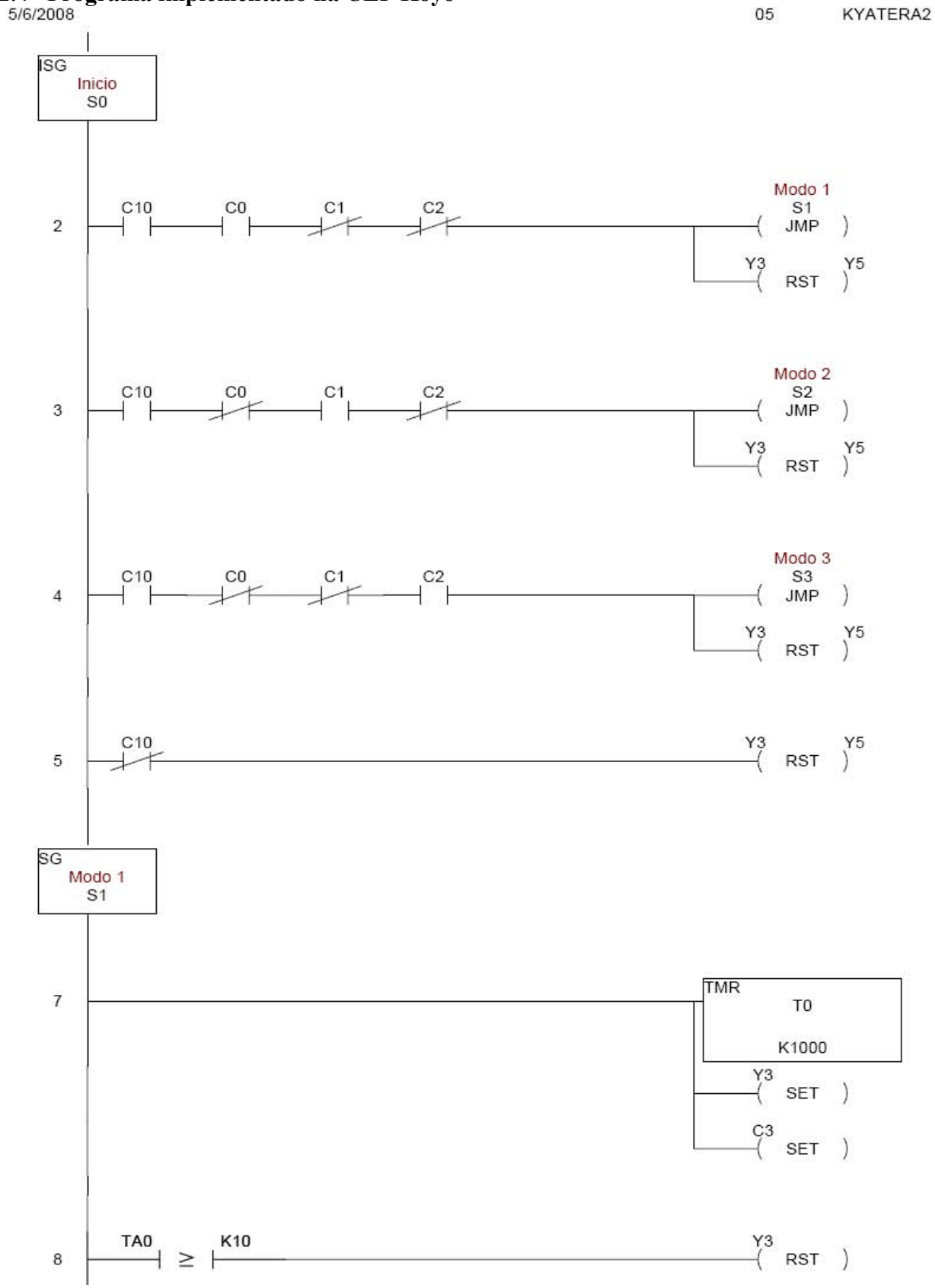
Menu	Lógica de Comando	X_3	Y_0	Y_1	Y_2	S
1	Inicio/Reset SW/ Sair	1	0	0	0	S_0
2	Reset externo	0	0	0	0	S_0
3	Inicialização do Robö	1	1	0	0	S_1
4	Operação na Face 1	1	0	1	0	S_2
5	Operação na Face 2	1	0	0	1	S_3

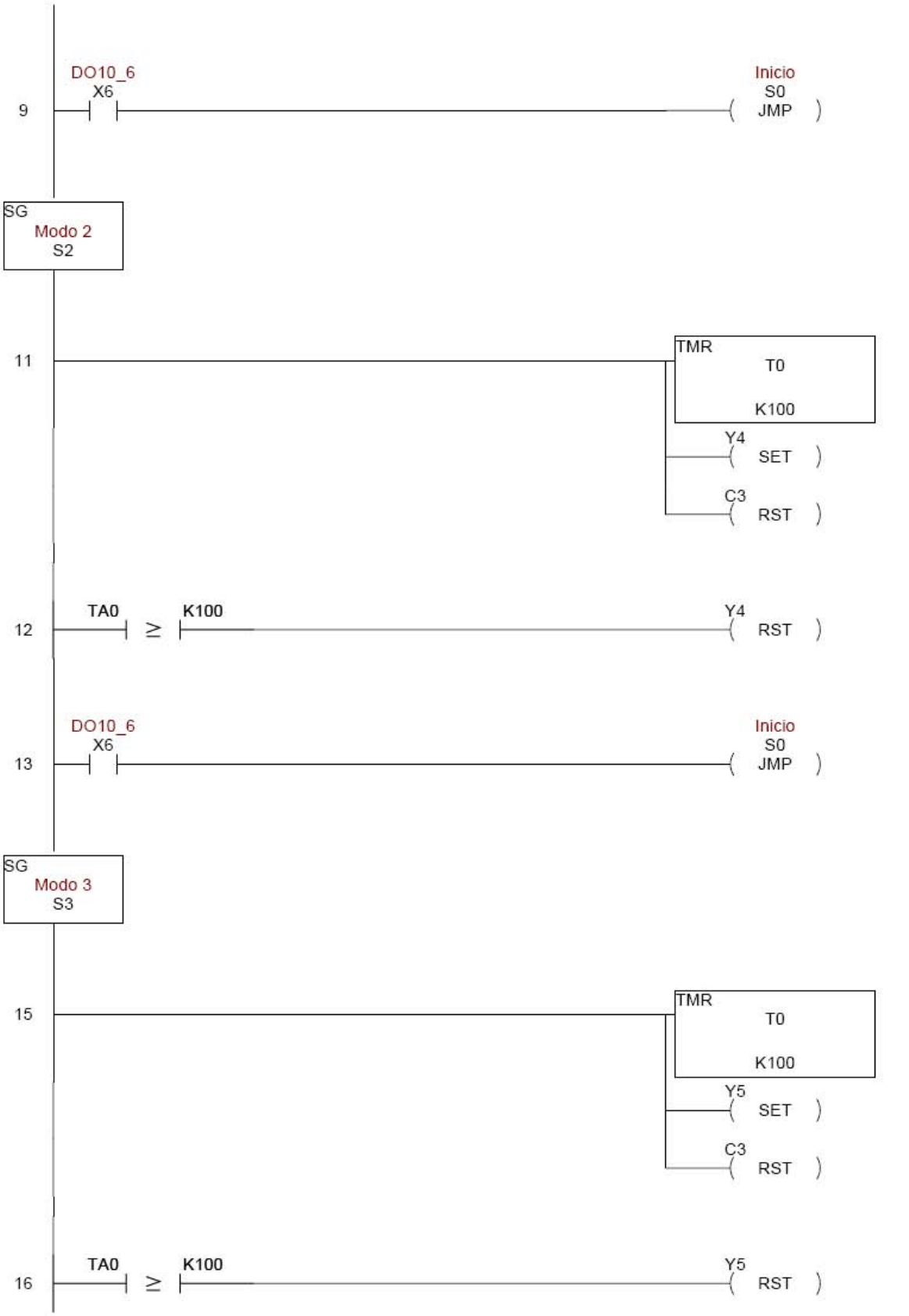
$C_0 = 1$ (sistema em funcionamento)
$X_0 = 1$ (botão de funcionamento)
$X_1 = 1$ (robô em movimento)
$X_1 = 0$ (final de movimento do robô)

Variáveis Robô (E/S)

CLP	Robö	Tipo
X_0	DO10_6	Saida
Y_0	DI10_3	Entrada
Y_1	DI10_4	Entrada
Y_2	DI10_5	Entrada

E.4 **Programa implementado na CLP Koyo**




DO10_6
X6
9
Inicio
S0
JMP
SG
Modo 2
S2
11
TMR
T0
K100
Y4
SET
C3
RST
TA0
≥
K100
12
Y4
RST
DO10_6
X6
13
Inicio
S0
JMP
SG
Modo 3
S3
15
TMR
T0
K100
Y5
SET
C3
RST
TA0
≥
K100
16
Y5
RST


DO10_6
X6
Inicio
S0
17
JMP
18
END
19
NOP

E.5 **Programa implementado no robô ABB IRB 1400**

```
%%%
VERSION:1
LANGUAGE:ENGLISH
%%%
MODULE KyaTera
PERS wobjdata sgua3:=[FALSE,TRUE,"",[[1383.42,183.711,811.144],
[0.649735,0.652177,-0.28073,-0.271478]],[[1.06305,1.72323,-0.33319],
[0.999997,0.002284,2.4E-05,8E-06]]];
PERS wobjdata sgua4:=[FALSE,TRUE,"",[[1529.4,317.814,815.56],
[0.255224,0.262435,-0.667034,-0.648888]],[[0,0,0],[1,0,0,0]]];
PERS wobjdata sgua5:=[FALSE,TRUE,"",[[1275.91,183.04,846.869],
[0.845215,0.136804,-0.364511,-0.3661]],[[0,0,0],[1,0,0,0]]];
CONST robtarget p5_1:=[[-13.45,77.89,137.49],[0.013321,0.390585,-
0.92037,-0.013705],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p5:=[[19.1,98.98,20.1],[0.013349,0.390597,-0.920364,-
0.013697],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p4_1:=[[116.92,110.47,199.54],[0.306494,0.783191,-
0.494542,-0.219323],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p4:=[[60.72,105.56,21.71],[0.306492,0.783199,-
0.494532,-0.219322],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p3:=[[28.04,102.06,21.56],[0.156073,-
0.373344,0.879487,-0.250515],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+
09,9E+09]];
PERS tooldata pincel:=[TRUE,[[7.86115,-30.1179,124.792],[1,0,0,0]],
[0.2,[85,0,65],[1,0,0,0],0.01,0.01,0.01]];
PERS tooldata Bic:=[TRUE,[[0.500839,-0.574904,226.276],[1,0,0,0]],
[0.25,[85,0,65],[1,0,0,0],0.01,0.01,0.01]];
!
_____________________________________________________________________
_____________________________
PROC pOS_INIT_R2()
MoveAbsJ [[120,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v100,z50,Bic\WObj:=sgua3;
ENDPROC
!
_____________________________________________________________________
_____________________________
PROC R2FACE3()
!
! Circulo
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua3;
! MoveJ p3,v400,z50,Bic\WObj:=sgua3;
MoveL Offs(p3,0,0,20),v100,z0,Bic\WObj:=sgua3;
MoveL p3,v100,z0,Bic\WObj:=sgua3;
MoveC Offs(p3,75,75,0),Offs(p3,150,0,0),v100,z0,Bic\WObj:=sgua3;
MoveC Offs(p3,75,-75,0),Offs(p3,0,0,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,0,0,20),v100,z0,Bic\WObj:=sgua3;
!
! Triangulo
!
MoveJ Offs(p3,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua3;
! MoveL Offs(p3,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua3;
```

```
MoveL Offs(p3,139.952,-37.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,113.971,7.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,100.981,30,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,49.0192,30,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,75,75,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua3;
!
! Retas Quebradas do Triangulo
!
MoveL Offs(p3,62.0096,52.5,20),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,87.9904,52.5,20),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,100.981,30,0),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,100.981,30,20),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,49.0192,30,20),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,49.0192,30,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,36.0289,7.5,20),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,113.971,7.5,20),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,113.971,7.5,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,126.989,-15.0478,20),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,23.0109,-15.0478,20),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua3;
MoveL Offs(p3,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua3;
!
MoveL Offs(p3,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua3;
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua3;
ENDPROC
!
_____________________________________________________________________________
_____________________________
PROC R2FACE4()
!
! Circulo
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua4;
MoveAbsJ [[0,0,0,-20,15,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua4;
MoveJ p4_1,v100,z50,Bic\WObj:=sgua4;
MoveL Offs(p4,0,0,20),v100,z0,Bic\WObj:=sgua4;
MoveL p4,v100,z0,Bic\WObj:=sgua4;
MoveC Offs(p4,75,75,0),Offs(p4,150,0,0),v100,z0,Bic\WObj:=sgua4;
MoveC Offs(p4,75,-75,0),Offs(p4,0,0,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,0,0,20),v100,z0,Bic\WObj:=sgua4;
```

```
!
! Triangulo
!
MoveJ Offs(p4,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua4;
! MoveL Offs(p4,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,139.952,-37.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,113.971,7.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,100.981,30,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,49.0192,30,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,75,75,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua4;
!
! Retas Quebradas do Triangulo
!
MoveL Offs(p4,62.0096,52.5,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,87.9904,52.5,20),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,100.981,30,0),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,100.981,30,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,49.0192,30,20),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,49.0192,30,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,36.0289,7.5,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,113.971,7.5,20),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,113.971,7.5,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,126.989,-15.0478,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,23.0109,-15.0478,20),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua4;
MoveL Offs(p4,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua4;
!
MoveL Offs(p4,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua4;
MoveJ p4_1,v100,z50,Bic\WObj:=sgua4;
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua4;
!
ENDPROC
!
_____________________________________________________________________________
_____________________________
PROC R2_MES_INC()
!
! Circulo
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua5;
```

```
MoveAbsJ [[0,0,0,0,45,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua5;
MoveJ p5_1,v400,z50,Bic\WObj:=sgua5;
MoveL Offs(p5,0,0,20),v100,z0,Bic\WObj:=sgua5;
MoveL p5,v100,z0,Bic\WObj:=sgua5;
MoveC Offs(p5,75,75,0),Offs(p5,150,0,0),v100,z0,Bic\WObj:=sgua5;
MoveC Offs(p5,75,-75,0),Offs(p5,0,0,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,0,0,20),v100,z0,Bic\WObj:=sgua5;
!
! Triangulo
!
MoveJ Offs(p5,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua5;
! MoveL Offs(p5,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,139.952,-37.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,113.971,7.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,100.981,30,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,49.0192,30,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,75,75,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,62.0096,52.5,0),v100,z0,Bic\WObj:=sgua5;
!
! Retas Quebradas do Triangulo
!
MoveL Offs(p5,62.0096,52.5,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,87.9904,52.5,20),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,87.9904,52.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,100.981,30,0),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,100.981,30,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,49.0192,30,20),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,49.0192,30,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,36.0289,7.5,0),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,36.0289,7.5,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,113.971,7.5,20),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,113.971,7.5,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,126.989,-15.0478,0),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,126.989,-15.0478,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,23.0109,-15.0478,20),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,23.0109,-15.0478,0),v100,z0,Bic\WObj:=sgua5;
MoveL Offs(p5,10.0481,-37.5,0),v100,z0,Bic\WObj:=sgua5;
!
MoveL Offs(p5,10.0481,-37.5,20),v100,z0,Bic\WObj:=sgua5;
MoveJ p5_1,v400,z50,Bic\WObj:=sgua5;
MoveAbsJ [[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
\NoEOffs,v200,z50,Bic\WObj:=sgua5;
!
!
```

```
ENDPROC
!
!
*************************************************************************
*********
!
PROC main()
!
! OBS.: WObj:=sgua3 e 4 são as duas faces do quadrado;
! WObj:=sgua5 é a face superior quando inclina o
quadrado
!
!
! INICIALIZAÇAO DO ROBO
!
!
! Posicao Inicial Robo R2
!
IF DI10_3=1 AND DI10_4=0 AND DI10_5=0 THEN
!
pOS_INIT_R2;
!
Set DO10_6;
!reg1 :=0;
WaitTime 5;
Reset DO10_6;
ENDIF
!
! Trajetoria Robo 2 - Face 3
!
IF DI10_3=0 AND DI10_4=1 AND DI10_5=0 THEN
!
R2FACE3;
!
Set DO10_6;
!reg1 :=0;
WaitTime 5;
Reset DO10_6;
ENDIF
!
! Trajetoria Robo 2 - Face 4
!
IF DI10_3=0 AND DI10_4=0 AND DI10_5=1 THEN
!
R2FACE4;
!
Set DO10_6;
! reg1 :=0;
WaitTime 5;
Reset DO10_6;
ENDIF
ENDPROC
ENDMODULE
```