Ricardo Carvalho Quesada

Projeto e Concepção de Células Robotizadas para Aplicações em Automação

49/2014

Campinas
2014.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

Ricardo Carvalho Quesada

Projeto e Concepção de Células Robotizadas para Aplicações em Automação

Dissertação de Mestrado apresentada à Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Mecânica, na Área de Mecânica dos Sólidos e Projeto Mecânico.

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO RICARDO CARVALHO QUESADA E ORIENTADA PELO PROF. DR. JOÃO MAURÍCIO ROSÁRIO.

.....

ASSINATURA DO ORIENTADOR

PROF. DR. JOAO MAURICIO ROSÁRIO
FEM / UNICAMP
Matrícula 062456

Campinas
2014.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Elizangela Aparecida dos Santos Souza - CRB 8/8098

Q37p

Quesada, Ricardo Carvalho, 1989-
Projeto e concepção de células robotizadas para aplicações em automação / Ricardo Carvalho Quesada. – Campinas, SP : [s.n.], 2014.

Orientador: João Maurício Rosário.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Automação industrial. 2. Integração. 3. Robótica. 4. Célula robotizada. 5. Integração funcional. I. Rosário, João Maurício,1959-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Project and desing of robotic cells fot applications in automation
Palavras-chave em inglês:
Industrial automation
Integration
Robotics
Robotic cell
Functional integration
Área de concentração: Mecânica dos Sólidos e Projeto Mecânico
Titulação: Mestre em Engenharia Mecânica
Banca examinadora:
João Maurício Rosário [Orientador]
Ely Carneiro de Paiva
Leonimer Flávio de Melo
Data de defesa: 19-03-2014
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 ACADEMICO

Projeto e Concepção de Células Robotizadas para Aplicações em Automação

Autor: Ricardo Carvalho Quesada

Orientador: Prof. Dr. João Maurício Rosário

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:

Prof. Dr. João Maurício Rosário , Presidente
DPM/FEM/UNICAMP

Prof. Dr. Ely Carneiro de Paiva
DPM/FEM/UNICAMP

Prof. Dr. Leonimer Flávio de Melo
UEL/Londrina

Campinas, 19 de Março de 2014.

Dedicatória

Dedico esse trabalho aos meus pais, irmãos, a minha família e amigos.

Agradecimentos

Este trabalho não seria possível sem a ajuda e motivação de várias pessoas as quais presto aqui minha homenagem:

Aos meus pais, pois sem eles eu não estaria aqui e esse trabalho nunca seria realizado.

Ao meu orientador por ter me dado esta oportunidade de crescimento e pelo auxilio e orientação.

Aos meus irmãos, família e amigos que me apoiaram e me ajudaram nos momentos difíceis e de depressão e estresse.

Aos meus amigos do LAIR, em especial ao Almiro, Rayanne e Renato que foram os que mantive maior contato e intimidade.

"O único lugar aonde o sucesso vem antes do trabalho é no dicionário."

Albert Einstein

Resumo

QUESADA, Ricardo Carvalho, Projeto e Concepção de Células Robotizadas para Aplicações em Automação, Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2013. Dissertação (Mestrado)

A necessidade atual de procedimentos automatizados em ambientes industriais exige o desenvolvimento e utilização de métodos, ferramentas e dispositivos com o objetivo de prototipagem rápida para a concepção, especificação e validação desses ambientes industriais, justificando ainda que a subutilização de dispositivos, que nesta área pode apresentar um custo elevado e se for demorado, à obsolescência dos equipamentos utilizados, fazendo com que a empresa não tenha gastos desnecessários e possa otimizar seu funcionamento.

Este trabalho tem como objetivo, desenvolver metodologias, a partir da utilização de ferramentas disponíveis no mercado para a concepção de células automatizadas em ambientes industriais com integração de dispositivos robóticos. Para o desenvolvimento e validação dos conceitos e ferramentas apresentados nesse trabalho será realizado um estudo de caso, com a modelagem de dispositivos industriais utilizando o software de programação off-line de robôs RobotStudio™, o formalismo GRAFCET como ferramenta de modelagem e integração de Sistemas Automatizados, e o ambiente Labview™ para supervisão e controle, de modo a permitir a completa automação desse estudo de caso, e ainda a possibilidade de estabelecer que o sistema seja controlado remotamente através de comunicação via WEB.

Palavras chave: Automação Industrial, Integração, Robótica, Célula Robotizada, Programação Off-Line, Sistema Supervisório.

Abstract

QUESADA, Ricardo Carvalho, Project and Design of Robotic Cells to Applications in Automation, Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2013. Dissertation (Master Degree)

The current need for automated procedures in industrial environments requires the development and use of methods, tools and devices for the purpose of rapid prototyping for the design, specification and validation of these industrial environments, yet justifying the underuse of devices that this area may have a high cost and if delayed, the obsolescence of the equipment used, making the company has not unnecessary expenses and can optimize its operation.

This paper aims to develop methodologies, from the use of commercially available tools for designing cells in industrial environments with automated integration of robotic devices. For the development and validation of the concepts and tools presented in this work will be a case study with industrial devices using modeling software off-line programming of robots RobotStudio™, GRAFCET formalism as a modeling tool and integration of Automated Systems, and environment LabVIEW™ supervision and control, to allow complete automation of this case study, and the possibility of establishing the system to be controlled remotely through WEB communication.

Keywords: Industrial Automation, Integration, Robotic Cell, Off-Line Programming, Robotics

Lista de Figuras

Figura 1.1 – Metodologia Proposta para Implementação 1
Figura 1.2 - Célula Robotizada 2
Figura 1.3 - Projeto e Concepção de Células Automatizadas 4
Figura 2.1 - Interface do Workspace Robot Simulator 8
Figura 2.2 - Interface Easy-Rob 8
Figura 2.3 - Interface de software RoboGuide 9
Figura 2.4 - Interface RobCAD 9
Figura 2.5 - Visual de FactoryLink 11
Figura 2.6 – Exemplo de painel em RSview 11
Figura 2.7 - Exemplo de software Elipse 12
Figura 2.8 - Exemplo do Software InTouch para supervisão e controle 12
Figura 2.9 - Interface do software Intellution Fix 13
Figura 2.10 - Exemplo de Wizcon 13
Figura 2.11 - Visual de Operate it 14
Figura 2.12 - Exemplo de supervisor em Labview 14
Figura 2.13 - Tecnologias integradas (CASTILLO 2010) 16
Figura 2.14 - Rede de Petri em sua forma mais básica 17
Figura 2.15 - Exemplo de rede marcada 18
Figura 2.16 - Furadeira usada como exemplo de G7 19
Figura 2.17 - Exemplo de G7 20
Figura 2.18 - Conceito de Automação Industrial Colaborativa 21
Figura 2.19 - Exemplo de Ethernet Industrial 22
Figura 2.20 – Exemplo de sistemas cooperativos via WEB 23
Figura 3.1 - Procedimento para Implementação 26
Figura 3.2 - Demonstrativo do Ambiente RobotStudio 29
Figura 3.3 - Exemplo de Programação em LabView 30
Figura 3.4 - Exemplo de Supervisório em LabView 32
Figura 4.1 - Célula dada pela metodologia 37

Figura 4.2 - Célula em Proposta 38
Figura 4.3 - Robô ABB IRB-140 40
Figura 4.4 - IRB-140 em ambiente virtual 40
Figura 4.5 - Procedimento de calibração de ferramenta 41
Figura 4.6 - Coordenadas para calibração da ferramenta terminal 42
Figura 4.7 - Sistema para calibração 45
Figura 4.8 - Método de calibração zero do robô 46
Figura 4.9 - Peça de calibração 50
Figura 4.10 – Plataforma Robótica de Posicionamento 53
Figura 4.11 – Plataforma de Posicionamento em representação virtual 54
Figura 4.12 - Esquema do modelo matemática da Plataforma 54
Figura 4.13 - Robô em Conjunto com a Plataforma 57
Figura 4.14 - Path para criação de programa 57
Figura 4.15 - Robô e PRP Realizando Tarefa em Conjunto 58
Figura 4.16 - Programa criado a partir de path 59
Figura 4.17 - Equipamentos de aquisição e envio de dados para controle da célula 60
Figura 5.1 - Célula robótica em estudo 63
Figura 5.2 – G7 Funcional do Robô 65
Figura 5.3 – G7 Funcional da MP 66
Figura 5.4 - Integração dos Diagramas 67
Figura 5.6 - Elipses de teste 69
Figura 5.7 - Circulo de Trajetória 71

Lista de Nomenclaturas

CAD – Computer Aided Design

CAE – Computer Aided Engineering

CAM – Computer Aided Manufacturing

CLP – Controladores Lógicos Programáveis

CRF – Célula Robotizada Flexível

E/S – Entradas e saídas

FPGA – Field Programmable Gate Array

G7 ou GRAFCET – Grafo de Comando Etapa e Transição

I/O – protocolo de comunicação entrada / saída (INPUT/OUTPUT)

IHM – Interface Homem Máquina

LAIR – Laboratório de Automação Integrada e Robótica

PAC – Programmable Automation Controller

PCI – Peripheral Component Interconnect

PRP – Plataforma Robótica de Posicionamento

RS – RobotStudio

SCADA – Supervisory Control and Data Acquisition

TCP – Tool Center Point

TCP/IP – Protocolo de comunicação

Sumário

Resumo xiii

Abstract xv

Lista de Figuras xvii

Lista de Nomenclaturas xix

1. Introdução 1

1.1. Justificativa 3

1.2. Objetivo 3

1.3. Objetivos Específicos 4

1.4. Estrutura do Trabalho 5

2. Revisão Bibliográfica 7

2.1. Ferramentas de programação off-line 7

2.2. Ferramentas para sistemas de supervisão e controle 10

2.3. Integração de Sistemas 15

2.4. Modelagem de Sistemas 16

2.4.1. Redes de Petri 17

2.4.2. Modelagem de sistemas a partir de GRAFCET 18

2.5. Sistemas Colaborativos 20

2.6. Sistemas cooperativos via WEB 22

2.7. Considerações Finais 24

3. Modelagem de Células Robotizadas 26

3.1. Prototipagem rápida de Células Flexíveis 27

3.2. Programação Off-Line RobotStudio^{TM} 28

3.3. Integração a partir do aplicativo LABVIEW 29

3.3.1. Software de Supervisão e Controle Baseado no LABVIEW 30
3.3.2. Hardware dedicado a instrumentação virtual 32
a) PXI e CompactPCI 32
b) Compact FieldPoint 33
c) Compact Vision System 33
d) CompactRIO 34
3.4. Considerações Finais 35
4. Estudo de Casos 36
4.1. Apresentação da célula robótica flexível 36
4.2. Descrição dos Elementos de Integração 39
4.2.1. Robô ABB IRB-140 39
4.2.1.1. Calibração da Ferramenta Terminal 41
4.2.1.2. Identificação do ponto zero do Robô 45
4.2.2. Plataforma Robótica de Posicionamento 53
4.2.2.1. Modelagem Cinemática Direta 54
4.3. Programação off-line 56
4.4. Supervisão e Controle a partir do aplicativo Labview™ 59
4.5. Sistema de Visão 60
4.6. Considerações Finais 61
5. Validação experimental 62
5.1. Implementação da Célula Integrada 62
5.2. Funcionalidade da Célula Robotizada 63
5.2.1. Descritivo Robô 64
5.2.2. Descritivo da Plataforma de Posicionamento 66
5.2.3. Integração com GRAFCET 67

5.3. Programação e simulação em RobotStudio™ 68
5.4. Considerações Finais 72
6. Conclusões e Perspectivas Futuras 73
7. Bibliografia 75
Anexo 78

CAPÍTULO 1

1. Introdução

No LAIR foram desenvolvidos vários trabalhos para a elaboração e criação de células robóticas, desde seu projeto até sua concepção, tais projetos demostram métodos para a facilitação da implementação as CRF, então neste trabalho demonstrarei uma metodologia proposta a partir de todos os trabalhos anteriores de meus colegas, e introduzirei um conceito de programação OFF-LINE, pelo qual se dará uma maior facilidade e agilidade na troca de dispositivos robóticos ou processos de manufatura.

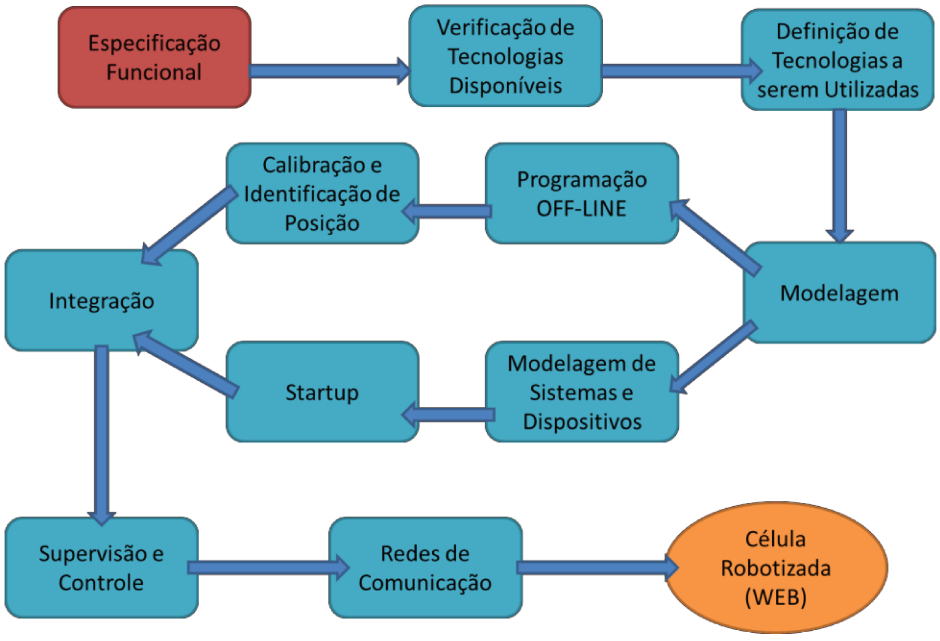


Figura 1.1 – Metodologia Proposta para Implementação

A figura 1.1 demonstra a metodologia proposta para o desenvolvimento e implementação de uma célula automatizada.

Os robôs industriais são altamente versáteis podendo trabalhar em conjunto com outros dispositivos, criando assim uma célula robótica. Nem toda célula robótica ou automatizada é necessariamente uma célula flexível.

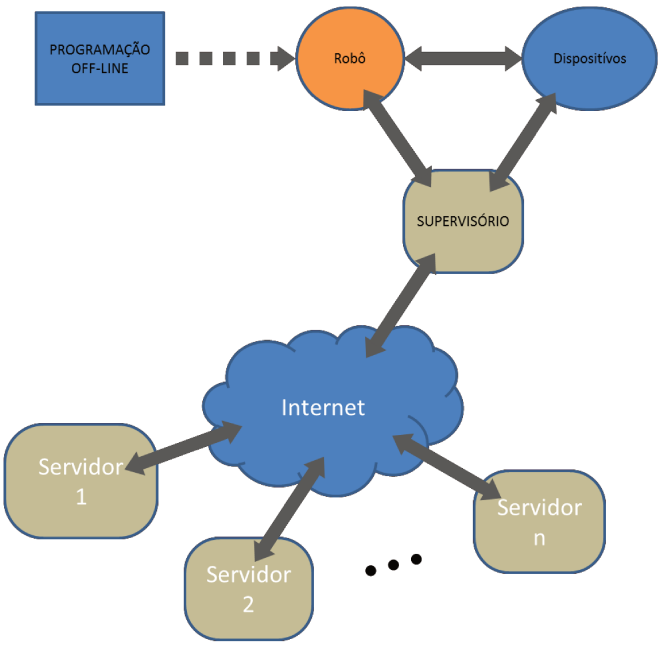


Figura 1.2 - Célula Robotizada

A figura 1.2 mostra uma Célula Robotizada Flexível (CRF) com programação Off-Line, que será estudada neste trabalho.

Uma célula automatizada flexível é aquela onde não se restringe a um único tipo de produto, mas sim, trabalha vários produtos distintos. Também deve ser de fácil modificação caso necessária, como troca de equipamentos e dispositivos.

Em um nível mais abrangente a acelerada evolução da tecnologia, trás consigo, uma necessidade de mudanças cada vez mais intensas e rápidas, porém nem sempre uma mudança drástica é viável, principalmente se tratando de equipamentos de elevado custo, como é o caso das células automatizadas, o alto custo dos dispositivos empregados nestas, impede com que sejam completamente substituídos, assim é mais viável remanejar e otimizar os já existentes, e caso realmente necessário, pode-se trocar apenas um elemento da célula e não a célula como um todo.

Levando em consideração estes aspectos, podemos introduzir métodos para tornar uma célula robótica mais flexível, e assim otimizar seu funcionamento explorando ao máximo os recursos dos dispositivos envolvidos, melhorando, portanto os processos de fabricação aumentando os lucros das empresas e diminuindo gastos desnecessários.

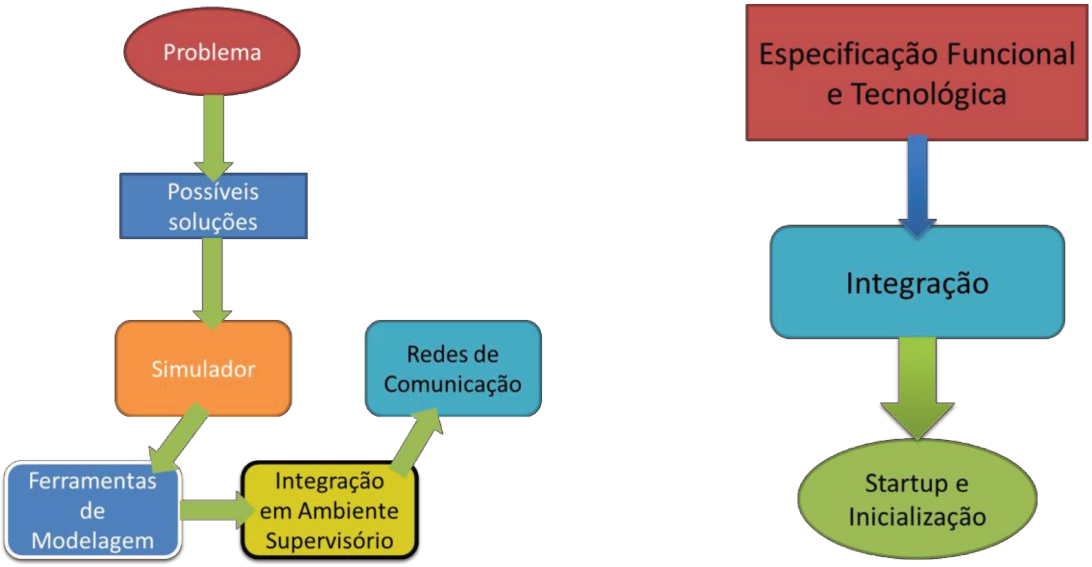
1.1. Justificativa

Dada a acelerada evolução na área tecnológica, os dispositivos e elementos voltados à automação e robótica estão cada vez mais eficientes e eficazes, tornando-se necessária à modificação de equipamentos desatualizados ou sua realocação no chão de fábrica, contudo, pelo alto custo que é gerado em uma parada de processo fabril, deve-se ter um método pratico e eficaz para que tanto o equipamento novo quando a mudança de processo no antigo seja realizada rápida e precisamente.

Com base na afirmação anterior, se faz importante a proposta de uma metodologia que engloba todos os estudos já realizados anteriormente com estudos novos demonstrando uma maior agilidade nesta área.

1.2. Objetivo

O objetivo deste trabalho é demonstrar métodos para integração de dispositivos com foco em células industriais automatizadas. Visto que as empresas modernas estão necessitando, cada vez mais, de se automatizar para suprir as necessidades do mercado competitivo.



a) Desenvolvimento Esquemático **b) Implementação e Validação**

Figura 1.3 - Projeto e Concepção de Células Automatizadas

Organizar e demonstrar uma metodologia unificando os trabalhos anteriores desenvolvidos no Laboratório de Automação Integrada e Robótica (LAIR), estabelecendo um método para dissimular a complexidade de projetos e especificação de ambientes automatizados utilizando robôs, usando apenas ferramentas disponíveis no meio industrial.

1.3.Objetivos Específicos

- Demonstrar métodos de integração de dispositivos.
- Simular uma célula automatizada em ambiente computacional.
- Apresentar formas de comunicação entre dispositivos.
- Demonstrar métodos que facilitam a troca de processos.
- Simular uma célula automatizada, fisicamente em ambiente laboratorial.
- Concepção através da prototipagem rápida
- Atualização rápida de ambientes e dispositivos de acionamento e controle *(retrofitting)*
- Sistemas colaborativos de automação

1.4.Estrutura do Trabalho

Este trabalho está disposto da seguinte maneira:

- Capítulo 1 – Introdução; este capítulo dá uma introdução ao trabalho e expõe a importância do mesmo, assim como demonstra os objetivos desta dissertação.

- Capítulo 2 – Revisão da Literatura; onde estão expostos os conceitos estudados e as ferramentas pertinentes a este trabalho. Neste capítulo se expõe vários conceitos e ferramentas existentes no mercado atual para a solução de problemas das indústrias automatizadas. Bem como softwares de simulação, controle e supervisão, métodos de modelagem e utilização da WEB para controle a distancia.

- Capítulo 3 – Modelagem de Células Automatizadas; aqui são expostas as ferramentas de modelagem utilizadas para a solução do problema proposto para o trabalho. É neste capítulo que se explica o motivo da escolha das ferramentas usadas, suas funcionalidades e seu funcionamento.

- Capítulo 4 – Estudo de caso; aqui é proposto um estudo de uma célula robótica específica, em ambiente virtual para simulações e testes, para que posteriormente seja implementada uma célula real.

  Neste capitulo ainda, são demonstradas as utilizações dos métodos deste trabalho e as possíveis soluções para os problemas apresentados.

- Capítulo 5 – Resultados e Discussões; neste capítulo serão expostos os resultados obtidos durante o estudo de caso e sua aplicação na célula real. Estes serão apresentados de forma simples e de fácil entendimento já que a proposta do trabalho é justamente facilitar e simplificar a realização de tarefas.

- Capítulo 6 – Conclusão: aqui são apresentadas as principais conclusões obtidas no desenvolvimento deste trabalho, analisando os principais resultados obtidos através de implementação experimental de célula robotizada flexível.

CAPÍTULO 2

2. Revisão Bibliográfica

Neste capítulo será feita uma breve apresentação das principais ferramentas utilizadas para a implementação de células robóticas flexíveis, usando apenas recursos disponíveis no mercado atual, e tornando a modernização mais viável e palpável.

Para efeitos exemplificativos, serão também apresentados neste capítulo, exemplos genéricos das ferramentas utilizadas no desenvolvimento deste trabalho.

2.1. Ferramentas de programação off-line

Atualmente existem diversos aplicativos de modelagem de células robóticas e programação OFF-LINE de robôs industriais, cada fabricante desenvolve sua própria linguagem de programação, porém existem plataformas gerais que não pertencem a nenhum destes fabricantes, e que pode ser usado para a simulação do ambiente industrial com a mesma precisão que um software específico.

Estes programas são softwares de CAD/CAM de alta complexidade, pois englobam todas as funcionalidades de um dispositivo real, bem como sua modelagem cinemática e dinâmica. Deste modo se consegue determinar um programa completo que visa realizar simulações e testes virtuais o mais próximo possível da realidade, também se torna possível à operação remota dos dispositivos.

Alguns dos principais softwares utilizados com este objetivo são o RobotStudio que será utilizado neste trabalho e, portanto será o mais detalhado, o Workspace, Easy-Rob, entre outros.

- Workspace – é um software abrangente, não pertence a nenhuma fabricante de robôs, portanto não apresenta os modelos cinemáticos do fabricante, o tornando

mais complexo para a programação off-line o que o faz ser mais utilizado apenas para a simulação virtual do ambiente da célula robótica.

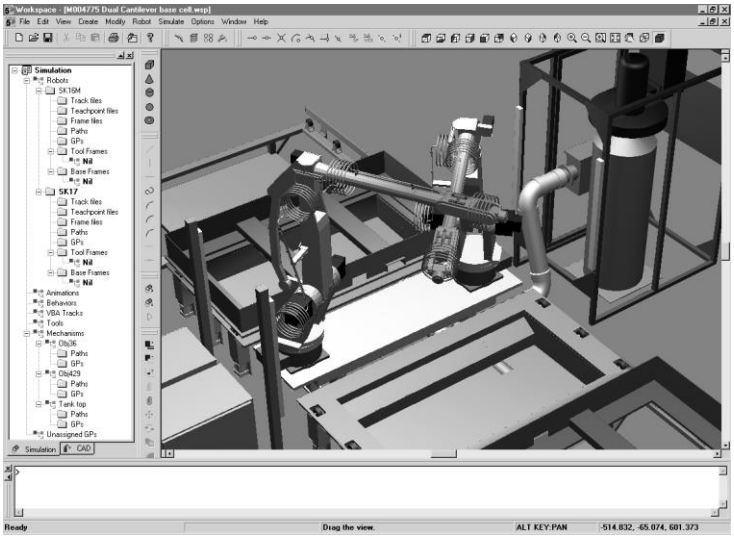


Figura 2.1 - Interface do Workspace Robot Simulator

- Easy-Rob – programa com uma interface simplificada ao usuário que visa a facilidade de utilização e simulação dos robôs industriais.

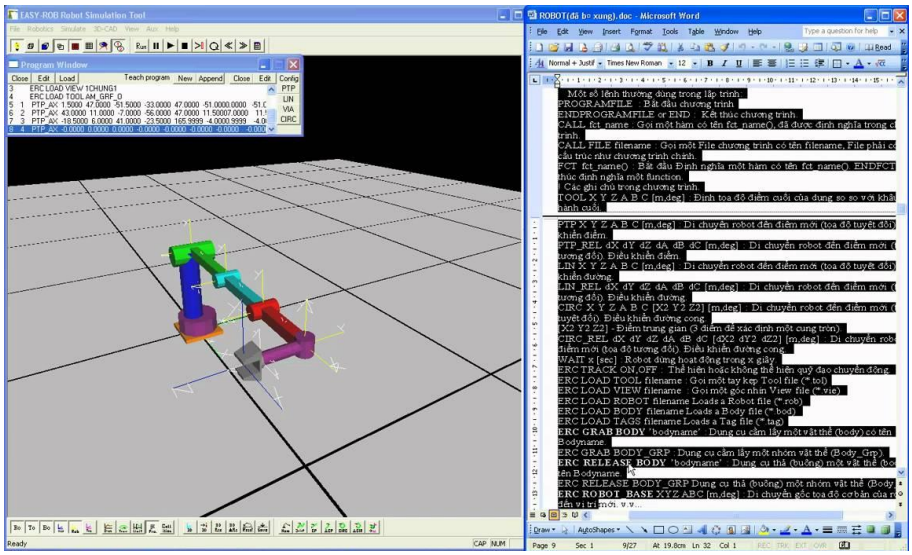


Figura 2.2 - Interface Easy-Rob

- A KUKA propõe uma ampla gama de softwares para a programação de cada aplicação possível dos robôs que fabrica.
- RoboGuide – é um software específico, fornecido pela FANUC, apresenta módulos de programação para as possíveis aplicações dos dispositivos fabricados ela mesma.

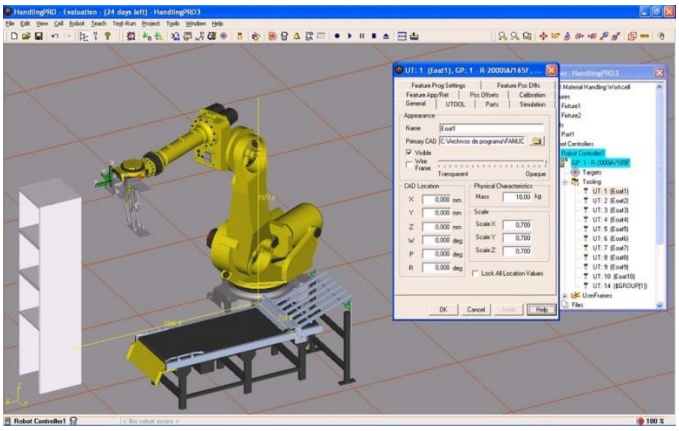


Figura 2.3 - Interface de software RoboGuide

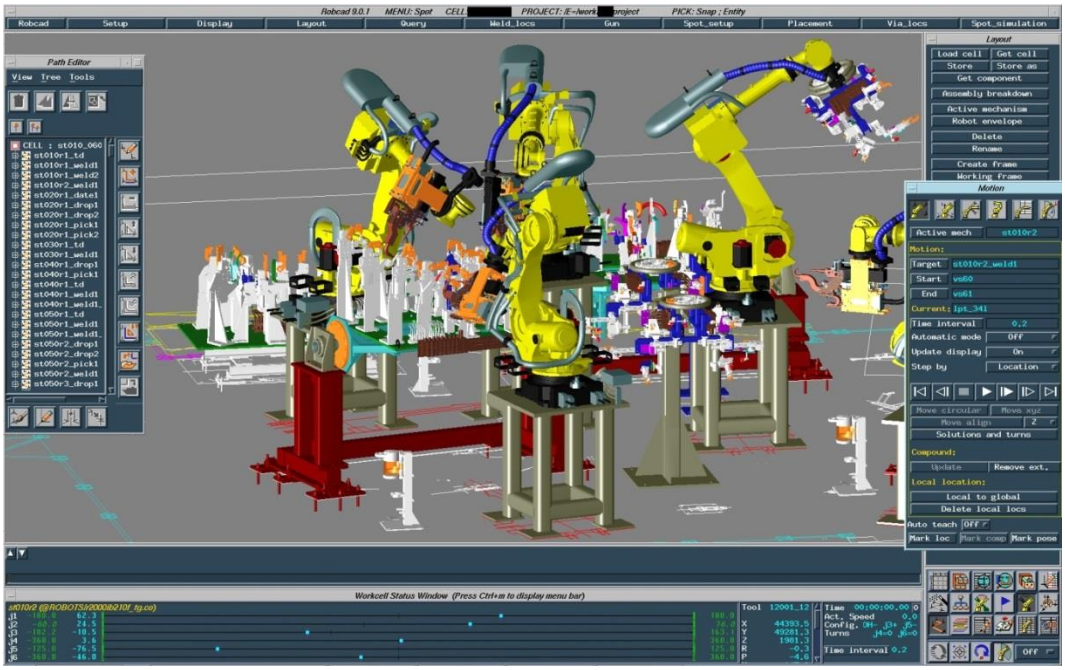


Figura 2.4 - Interface RobCAD

- RobCAD (figura 2.4) - é uma plataforma de simulação e programação de robôs industriais desenvolvida pela Telematix, é de uso geral, ou seja, não é específico de nenhum fabricante de robôs e pode ser usada para criação de novos dispositivos. Como principal desvantagem, além da modelagem matemática que é necessária para o desenvolvimento destes dispositivos virtuais, também apresenta um custo demasiado elevado o que inviabiliza sua utilização.

2.2. Ferramentas para sistemas de supervisão e controle

A interface homem máquina ou IHM se dá através de sistemas que traduzem a linguagem das máquinas para uma linguagem mais acessível para o homem, tais implementações são realizadas pelos sistemas SCADA (Supervisory Control and Data Acquisition), estes softwares são amplamente utilizados para facilitar a visualização e controle de toda uma indústria, tornando esta controlável por apenas um computador ou um único operador que monitora tudo o seu funcionamento. Também são usados para supervisionar uma célula ou até mesmo uma única máquina.

Com o avanço tecnológico nas áreas de telecomunicação, áudio visual, redes de comunicação, os sistemas supervisórios ganharam uma nova face, podendo passar de botões e luzes, para telas de computadores e agora com as telas sensíveis ao toque e as redes sem fio, todas as funcionalidades e toda a supervisão de um chão de fábrica pode ser simplificado a uma tela de um "*Tablet*". Sendo assim, faz-se cada vez mais necessária a evolução dos softwares de controle e supervisão.

No mercado atual existem várias ferramentas industriais, desenvolvidas para controle e supervisão de sistemas.

- Gênesis -

- FactoryLink

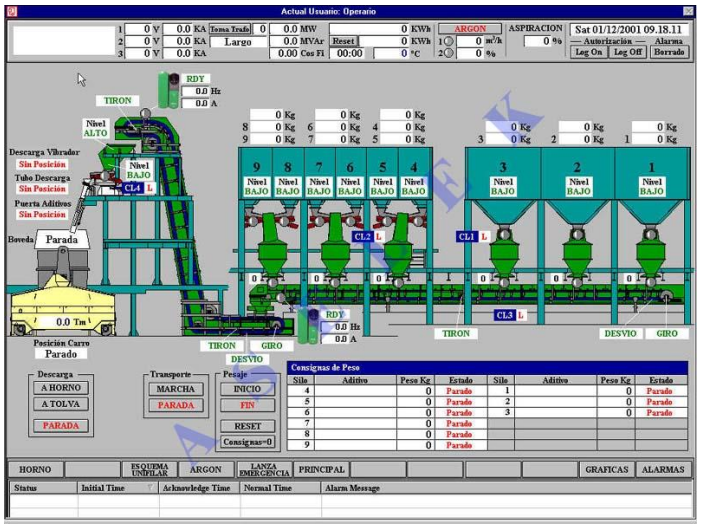


Figura 2.5 - Visual de FactoryLink
(http://www.plm.automation.siemens.com/pt_br/products/tecnomatix/production_management/factorylink/)

- Unisoft
- Citect
- RSView (Rockwell – Allen Bradley)

Figura 2.6 – Exemplo de painel em RSview
(http://www.rockwellautomation.com/rockwellsoftware/performance/view32/overview.page)

- Elipse (nacional)

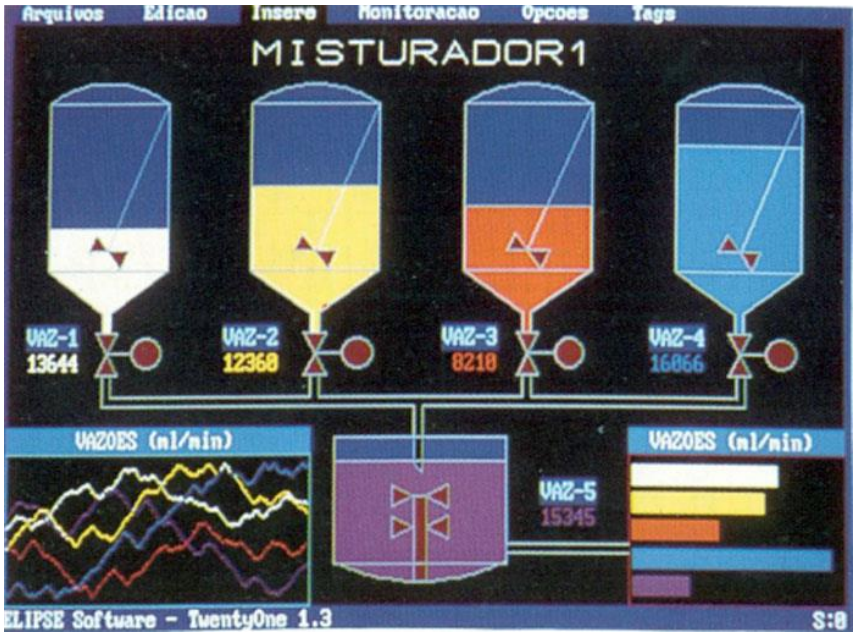


Figura 2.7 - Exemplo de software Elipse (http://www.elipse.com.br/port/scada.aspx)

- Intouch (Wonderware)

Figura 2.8 - Exemplo do Software InTouch para supervisão e controle (http://software.invensys.com/products/wonderware/hmi-and-supervisory-control/)

- Fix (Intellution) –

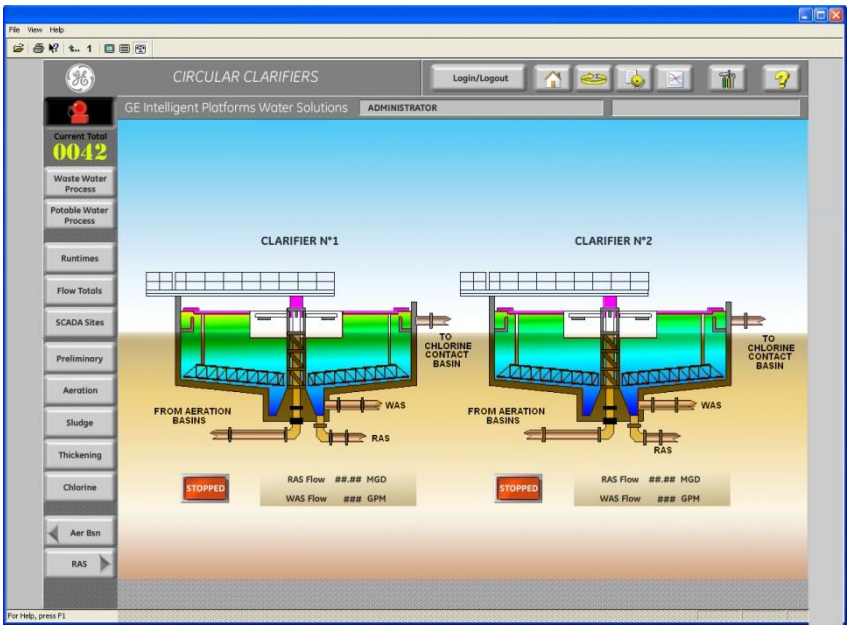


Figura 2.9 - Interface do software Intellution Fix (http://www.ge-ip.com/products/proficy-hmi-scada-ifix/p3311)

- Wizcon –

Figura 2.10 - Exemplo de Wizcon (http://www.getcontrolmaestro.com/scada-solutions.html)

- Operate it (ABB) –

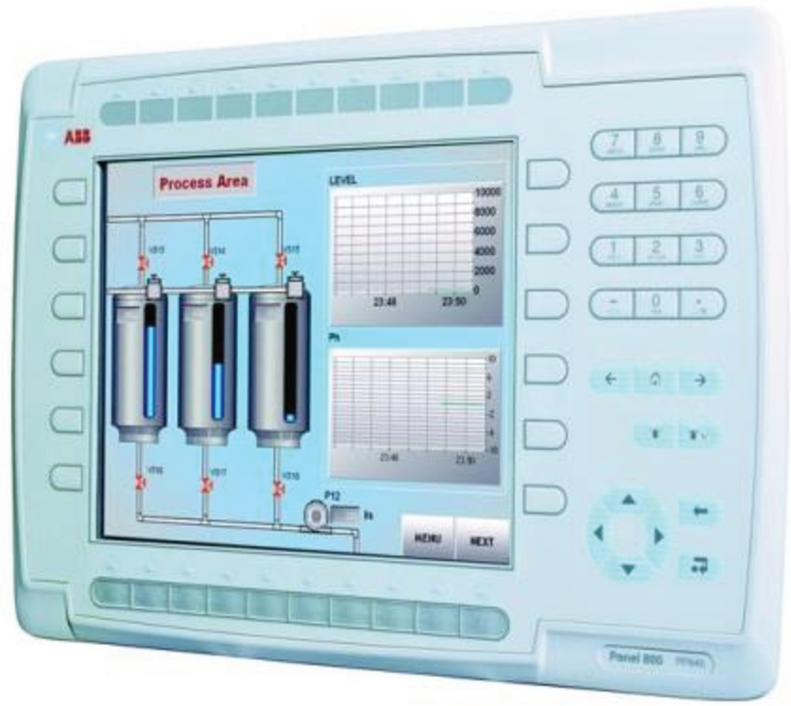


Figura 2.11 - Visual de Operate it (http://www05.abb.com/global/scot/scot296.nsf/veritydisplay/f56c7d6f1c7994a3c1256a730072aa17/$file/3bus094162r0001_-_en_operate_it_overview_brochure.pdf)

- LabView™

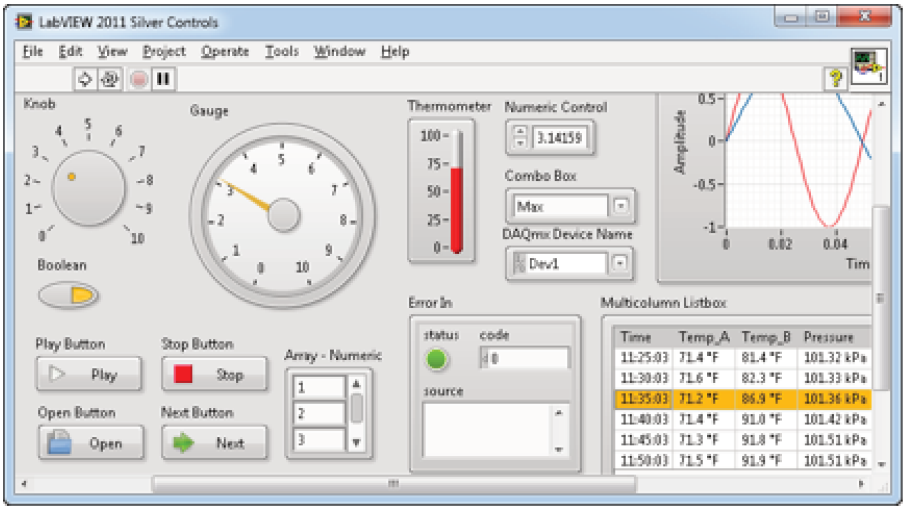


Figura 2.12 - Exemplo de supervisor em Labview™ (http://www.ni.com/labview/pt/)

2.3. Integração de Sistemas

A seleção das tecnologias que vão ser integradas na plataforma colaborativa leva em consideração que estas estão presentes de maneira integrada nos SAPs atualmente implementados nas empresas. Aliás, estas tecnologias oferecem a possibilidade de propor e realizar inúmeras práticas experimentais, e na atualidade são utilizadas de forma separada na maioria de cursos de Automação e Robótica demonstrando sua eficácia como complemento para a assimilação de conceitos isolados, mas denotam uma baixa eficiência como apoio na aquisição das competências necessárias para lidar com SAPs modernos. Consequentemente a sua utilização de maneira integrada possibilitará a criação de um ambiente interdisciplinar que estimulará a participação ativa dos estudantes nos processos de formação e pesquisa ao interior das IES. As principais tecnologias selecionadas para serem integradas neste trabalho são:

- Sensores e Atuadores Industriais;
- Manipuladores Robóticos;
- Processamento de Imagens;
- Supervisão, Comando e Comunicação;
- Eletrônica Reprogramável Estruturada.

Pela figura 2.13, podemos perceber que as tecnologias de sensores e atuadores industriais junto com os manipuladores robóticos (à direita e esquerda no diagrama e representadas na cor verde) correspondem à Parte Operativa (PO) de um SAP enquanto as tecnologias de supervisão, comando e comunicação junto com o processamento de imagens (na parte superior e inferior no diagrama, e representadas na cor azul) estão associadas à Parte de Comando (PC). A parte de comando da plataforma cujo objetivo é integrar e comunicar entre si às outras tecnologias é implementada através de eletrônica reprogramável estruturada. (CASTILLO et all. 2010)

Figura 2.13 - Tecnologias integradas (CASTILLO 2010)

2.4.Modelagem de Sistemas

Quando se fala em modelagem de sistemas, a organização é prioritária para que se tenha um entendimento pleno do problema e uma visualização completa da solução, ou possível solução para tal problema. Portanto foram criados os diagramas de blocos que são utilizados para tal compreendimento.

Com a complexidade e grande abrangência dos diagramas de blocos se fez necessária à criação de outros diagramas mais específicos para determinadas funções, assim pode-se simplificar os métodos de modelagem. Para uma lógica mais algébrica da modelagem de sistemas temos as redes de Petri e para um diagrama mais funcional se criou o GRAFCET (G7).

Pode-se definir a Modelagem de um sistema como uma representação de um objeto, sistema ou ideia 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 proposito de entender o mesmo, ou seja, um modelo é uma descrição do sistema real.

Desta forma um modelo passa a ser uma réplica ou uma abstração com a caracteristica essencial de um sistema ou processo. Atraves destes problemas que desobedecem a soluções diretas por causa do tamanho, complexidade ou estrutura, são frequentemente avaliados atraves de modelos de simulação, ou seja, o modelo passa a ser uma represnteação simplificada de uma parte da realidade de sistemas, podendo ser estes sistemas de diferentes tipos. (Aihara, 2005).

2.4.1. Redes de Petri

É uma técnica de modelagem com forte base matemática, isto permite a modelagem de sistemas paralelos, assíncronos, concorrentes e não determinísticos. Sua representação gráfica apresenta, em sua forma mais básica, dois componentes: a transição representada por uma barra e o lugar representado por um circulo.

- Transição: variável ativa denota uma ação realizada pelo sistema.
- Lugar: variável passiva demonstra o estado em que o sistema se encontra.

As redes de Petri podem ser enfocadas através de três fundamentações diferentes. A primeira utiliza a teoria de bag como suporte. A segunda usa os conceitos da álgebra matricial. A última se fundamenta na estrutura definida por relações.

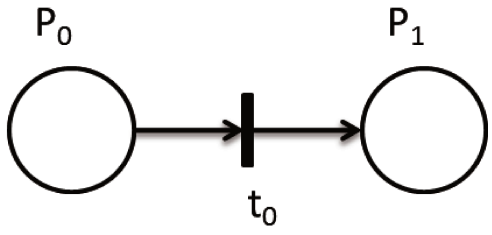


Figura 2.14 - Rede de Petri em sua forma mais básica

Geralmente se utiliza as redes de Petri marcadas, mostrada na Figura 2.15, as marcas (tokens) são informações atribuídas aos lugares para representar a situação (estado) da rede em um determinado momento. Define-se uma rede de Petri marcada pela dupla RM = (R, M_0), onde R e a estrutura da rede e M_0 a marcação inicial. Assim, para simular o comportamento dinâmico dos sistemas a marcação da rede é modificada a cada ação realizada (transição disparada).

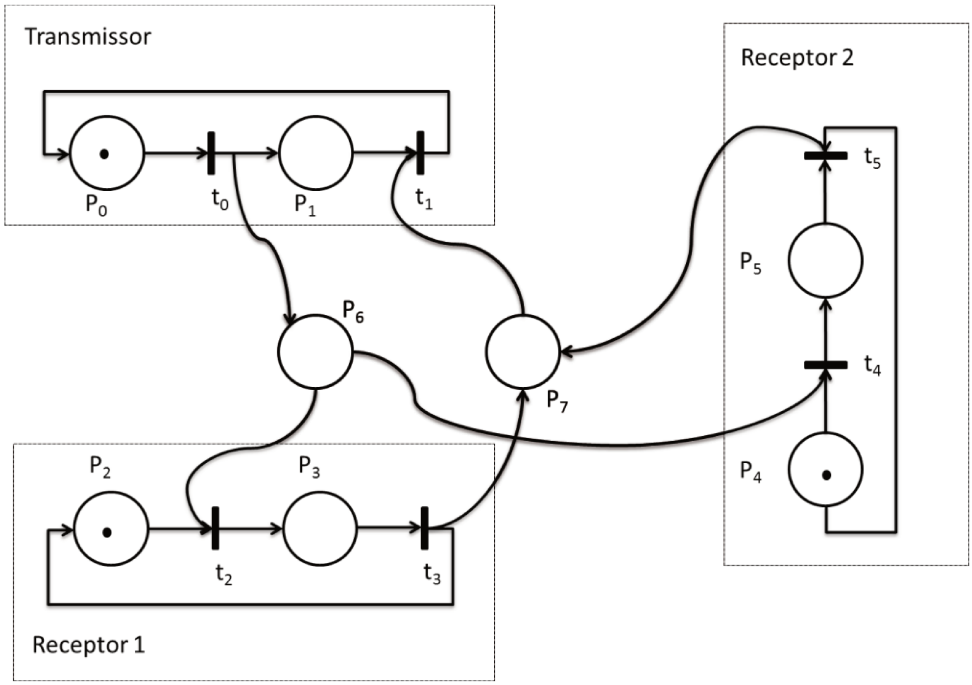


Figura 2.15 - Exemplo de rede marcada

Em anexo pode-se encontrar um descritivo mais detalhado de como se criar uma rede de Petri.

2.4.2. Modelagem de sistemas a partir de GRAFCET

Quando se fala em modelagem de sistemas, a organização é prioritária para que se tenha um entendimento pleno do problema e uma visualização completa da solução, ou possível solução para tal problema. Assim, os diagramas de blocos foram criados, mas são muito abrangentes, e em alguns casos sua visualização é demasiada complexa, assim sendo, se deu a

necessidade de se melhorar ou recriar o diagrama com uma lógica mais exata e simples. Então, a partir de uma iniciativa de pesquisadores franceses e gerentes industriais, obteve-se um sistema mais fácil e completo, visando a funcionalidade mais voltada a manufatura, este sistema foi denominado de GRAFCET, que chamaremos de G7.

Para demontrar, um exemplo de G7 seria de um furadeira que realizará uma operação de furação de uma peça com controle de velocidade de aproximação, avanço e retorno. Uma furadeira automatizada é programada para descer em alta velocidade da posição alta (h) até achar o sensor b1 , descendo em velocidade baixa até (b2), retornando após até a posição (h) em alta velocidade .

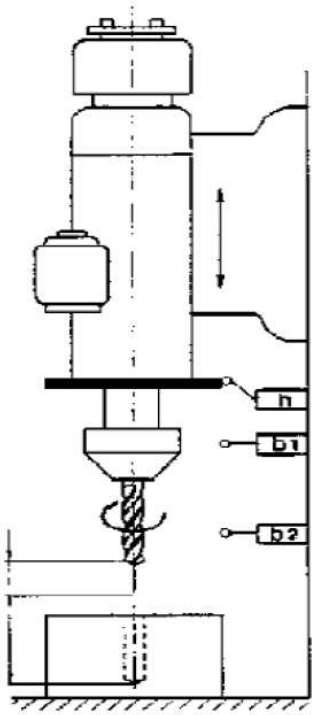


Figura 2.16 - Furadeira usada como exemplo de G7

O G7 é um conjunto de etapas e transições ligadas entre si através de ligações orientadas, quem sintetizam um sistema complexo de ações e tarefas que um dispositivo ou conjunto de dispositivos terá que realizar.

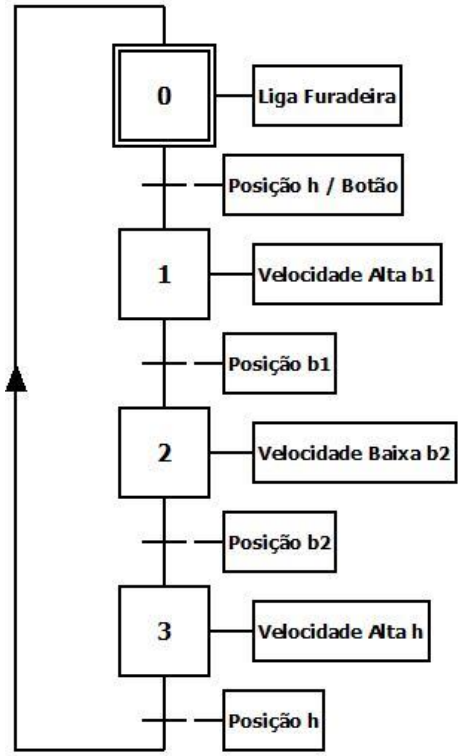


Figura 2.17 - Exemplo de G7

Em anexo a este trabalho se encontra um resumo do funcionamento do G7.

2.5.Sistemas Colaborativos

Devido à mudança de uma perspectiva econômica local para uma economia global, desde a última década do século XX os ambientes de manufatura também têm evoluído nas suas dimensões econômica, técnica e organizacional; fazendo com que as tendências na Automação Industrial tenham que se adaptar a uma necessidade de produção em pequenos e medianos lotes, com uma incrementada diferenciação de famílias de partes e/ou produtos, sendo estes quase customizados às necessidades de cada cliente e acrescentando também valor ao componente intangível dos produtos (software, serviços de ajuda incluídos, suporte on-line, etc.). Portanto

novos paradigmas como a Gestão da Manufatura Colaborativa (CMM) e consequentemente os Sistemas Automatizados de Produção Colaborativos (CSAP) surgem como uma evolução e uma resposta às novas necessidades não satisfeitas completamente pelo conceito original CIM; o qual proporcionava aos SAP um grau muito pequeno de flexibilidade e integração de hardware e software, baseado em uma arquitetura de controle fortemente hierárquica1 e centralizada e em uma estrutura de planejamento sequencial, que não permitia a estes sistemas se adaptar rapidamente às mudanças do ambiente.

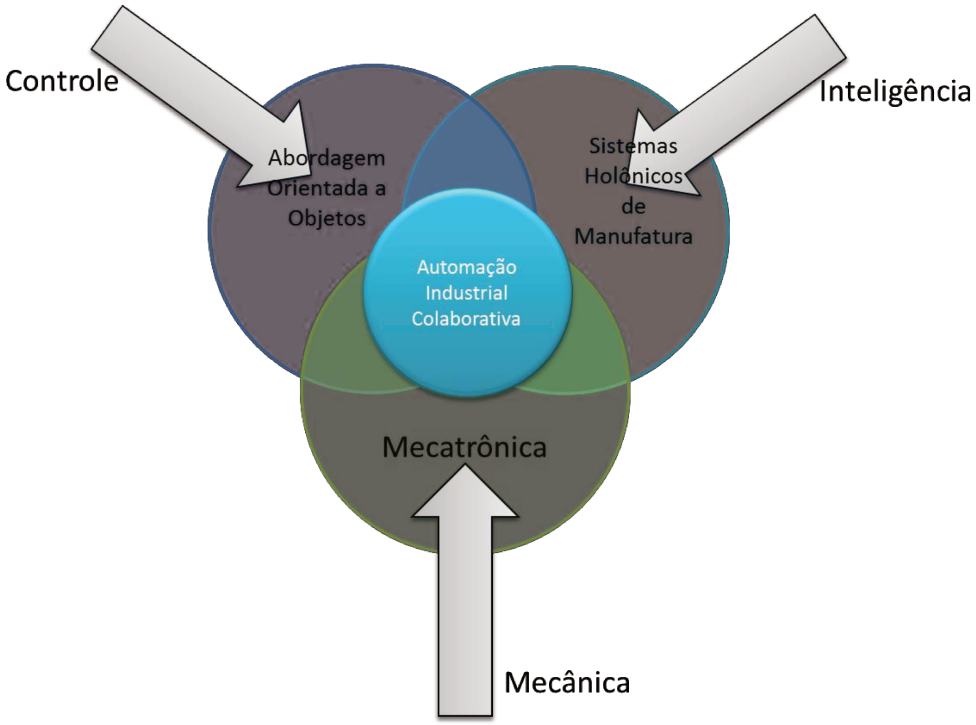


Figura 2.18 - Conceito de Automação Industrial Colaborativa

Os CSAP além de melhorar as características anteriores (flexibilidade e possibilidade de integração) acrescentam agilidade, modularidade, tolerância a falhas, possibilidade de reutilização, interação entre componentes e gestão da rentabilidade aos Sistemas de Produção.

Desta forma os CSAP podem atingir os objetivos globais e locais de manufatura, baseados em uma estrutura já não hierárquica, mas heterárquica2. Esta nova abordagem está baseada no

desenvolvimento e integração de tecnologias emergentes como: controle orientado a objetos (descentralizado), Sistemas Inteligentes de Manufatura (IMS) e Mecatrônica, Figura 2.18. As características de controle e arquitetura modular, distribuída e descentralizada destes CSAP exigem também a implementação de capacidades de supervisão e controle locais e remotas. (COLOMBO et al., 2004).

2.6.Sistemas cooperativos via WEB

Segundo Langrafe, Ferraz e Lopes (2011). À medida que os fabricantes procuram melhorar processos, aumentar a produtividade e integrar as redes de produção e negócios, muitos estão se voltando para a tecnologia Ethernet em sua planta. Esta migração está rapidamente ganhando impulso. Uma vez considerada uma solução que se limitou a ambientes de rede corporativa, a tecnologia Ethernet tem se mostrado uma alternativa robusta que pode satisfazer as necessidades específicas do ambiente de produção.

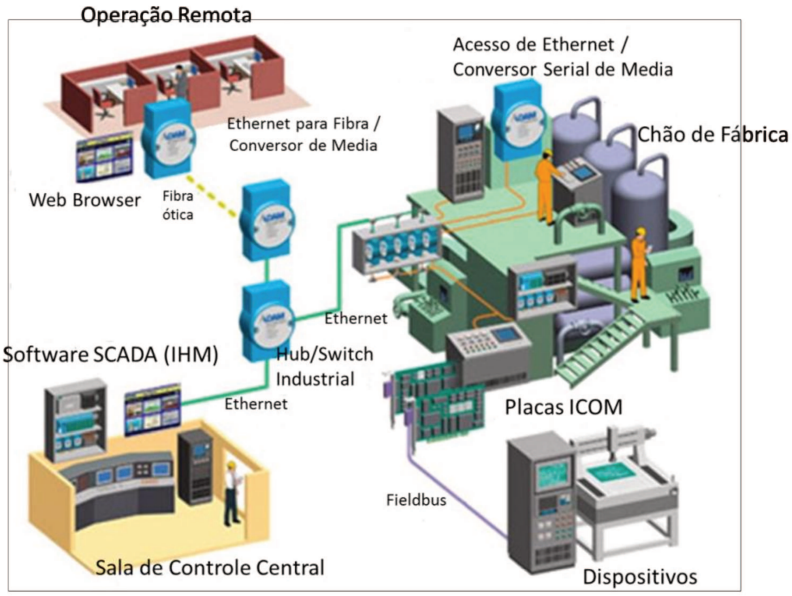


Figura 2.19 - Exemplo de Ethernet Industrial

A migração de redes industriais tradicionais para a ethernet abre novos caminhos para uma integração muito maior entre as diversas partes da indústria e ainda possibilita a inserção desta a WEB, ou seja, eleva o controle não só ao ambiente interno da indústria, mas também ao exterior, permitindo o controle a longas distancias como assistências externas como mostra o exemplo da figura 2.19, não sendo mais necessária a presença do técnico no local, ou do operador ou ainda permitindo um nível ainda maior de troca de dados com filiais da empresa em continentes distantes.

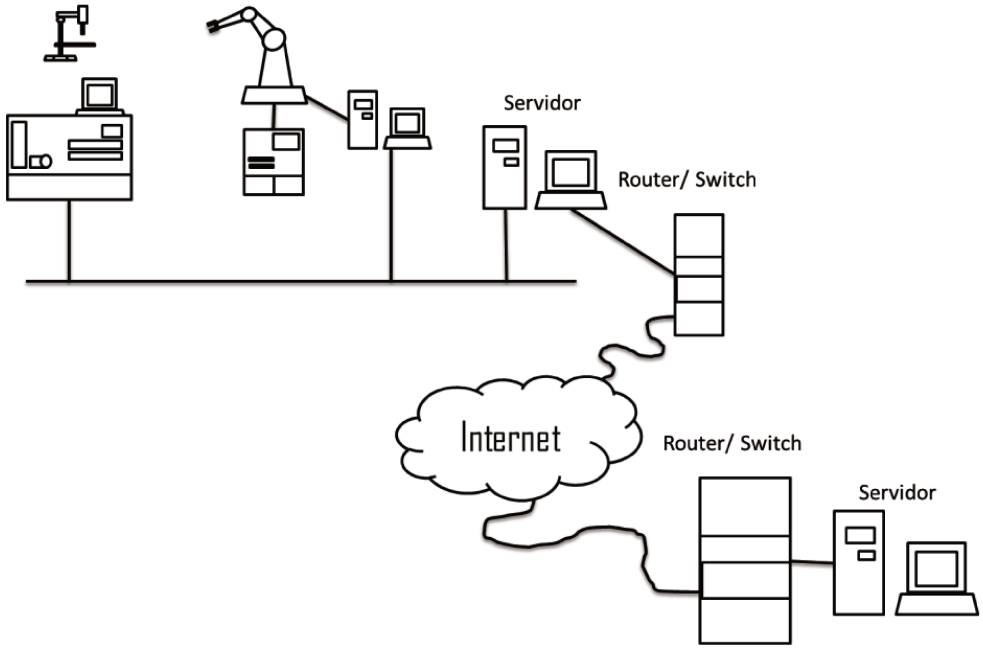


Figura 2.20 – Exemplo de sistemas cooperativos via WEB

Hoje em dia as TIC baseadas na internet possibilitam a interação com manipuladores robóticos em ambientes de aprendizagem a distância por meio da Tele operação e a tele presença, surgindo desta forma a Tele robótica.

A tele operação é um modo de controle de um sistema físico onde cada um dos comandos dados pelo operador é transmitido para ser executado pelo tele operador (dispositivo remoto). (CASTILLO, 2010)

- Existem dois tipos de tele operação direta ou continua e tele operação indireta ou tele programação, dependendo se os comandos são transmitidos ao escravo de forma ON-LINE ou OFF-LINE respectivamente.
- A tele presença é possível quando é existe uma realimentação ON-LINE de Vídeo e de Dados desde o dispositivo Remoto até o operador (Supervisão).

O projeto e desenvolvimento de laboratórios remotos e virtuais é possível devido à disponibilidade de tecnologias como a tele presença e a tele operação as quais permitem operar e controlar dispositivos através da internet enviando dados, áudio e vídeo não só em um sentido, mas tendo realimentação (SAIGYN, 2004). A administração destes laboratórios usualmente requer o uso de servidores para gerenciar tanto a utilização do sistema pelos usuários quanto os equipamentos integrados nesse sistema (DORMIDO, 2004 apud ALIANE, 2007).

2.7. Considerações Finais

Este capítulo apresentou os conceitos principais para a concepção de células robotizadas flexíveis, apresentando as principais ferramentas disponíveis no mercado. Demonstrando a extensa gama destas ferramentas para o projeto e concepção de CRF, mostrando os principais componentes para tal trabalho, pode-se perceber que a limitação para a elaboração de células é apenas uma questão financeira e de conhecimento.

Dentre as vantagens de se sistematizar a elaboração destas células, pode-se destacar a facilidade que isto gera, principalmente por haver tantos aplicativos, a escolha destes pode se tornar uma tarefa trabalhosa que é evitada com esta esquematização das tarefas que devem ser seguidas.

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 as ferramentas de modelagem, programação e integração que serão utilizados neste trabalho para exemplificar a elaboração de células robotizadas, bem como será feita uma descrição destes elementos.

CAPITULO 3

3. Modelagem de Células Robotizadas

O procedimento é obtido através de etapas associadas a concepção que esta sendo mostrada na figura 3.1. E que serão descritos a seguir:

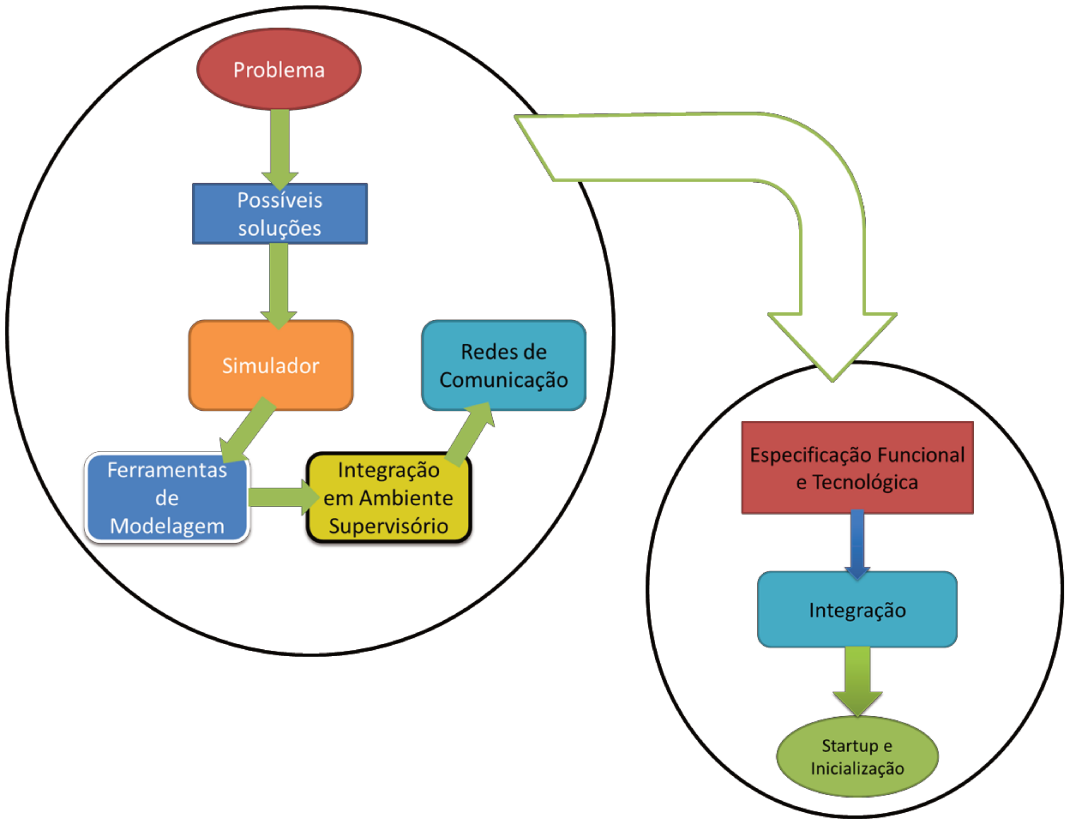


Figura 3.1 - Procedimento para Implementação

Este procedimento é de extrema importância para a concepção da célula robotizada flexível, pois graças a ele se dá a possibilidade de redução de tempo de implementação e diminuição dos custos com o projeto.

3.1. Prototipagem rápida de Células Flexíveis

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. (Oliveira, 2008).

Basicamente damos ênfase em dois tipos de prototipagem rápida:

- Prototipagem rápida em células flexíveis;
- Prototipagem rápida em instrumentação virtual.

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).

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.

3.2. Programação Off-Line RobotStudio™

Uma das principais vantagens da programação off-line é a não necessidade de ter disponíveis os equipamentos para desenvolvimento do programa, necessitando numa fase inicial, apenas do desenho da planta e especificações funcionais e tecnológicas dos elementos da célula e posteriormente da geração de movimentos do manipulador e posteriormente com a lógica de programação e cálculos.

Isto possibilitará num planejamento de tarefas, ambientes e ferramentas, não necessitando a parada dos equipamentos durante a implantação de novos programas em células existentes, e no caso de implementação de novas células. Toda a programação e estudo de viabilidade poderão ser realizados sem investimento nestes equipamentos a partir da alteração de documentação inicialmente estabelecida, considerando inclusive tempos de execução, permitindo assim a orientação de clientes para a tomada de decisões em cima de uma solução robotizada.

O processo de programação off-line permite inclusive uma abordagem mais apurada na estruturação dos programas, sendo extremamente necessária uma participação efetiva do programador na estruturação da lógica a ser definida. (Freitas et. all, 2004)

Para este trabalho optou-se por utilizar o software RobotStudio (RS) versão 3.1 da ABB para realizar as programações off-line, esta escolha foi feita com base em que é o software que se encontra disponível para utilização no LAIR e que acompanha o robô existente no laboratório, ABB IRB140 que será o mesmo usado na célula. Este software, por ser uma versão antiga apresenta algumas limitações, uma delas é não poder se fazer o controle da Plataforma por ele, assim deve-se para este trabalho utilizar outro software de controle para a PRP, que será explicado mais a frente.

O RS é de originalidade da ABB, uma fabricante suíça de robôs industriais e dispositivos, através dele é possível realizar simulações em ambiente virtual, também é possível fazer a programação do robô e até mesmo controlá-lo remotamente visualizando o que está sendo feito pela representação virtual deste.

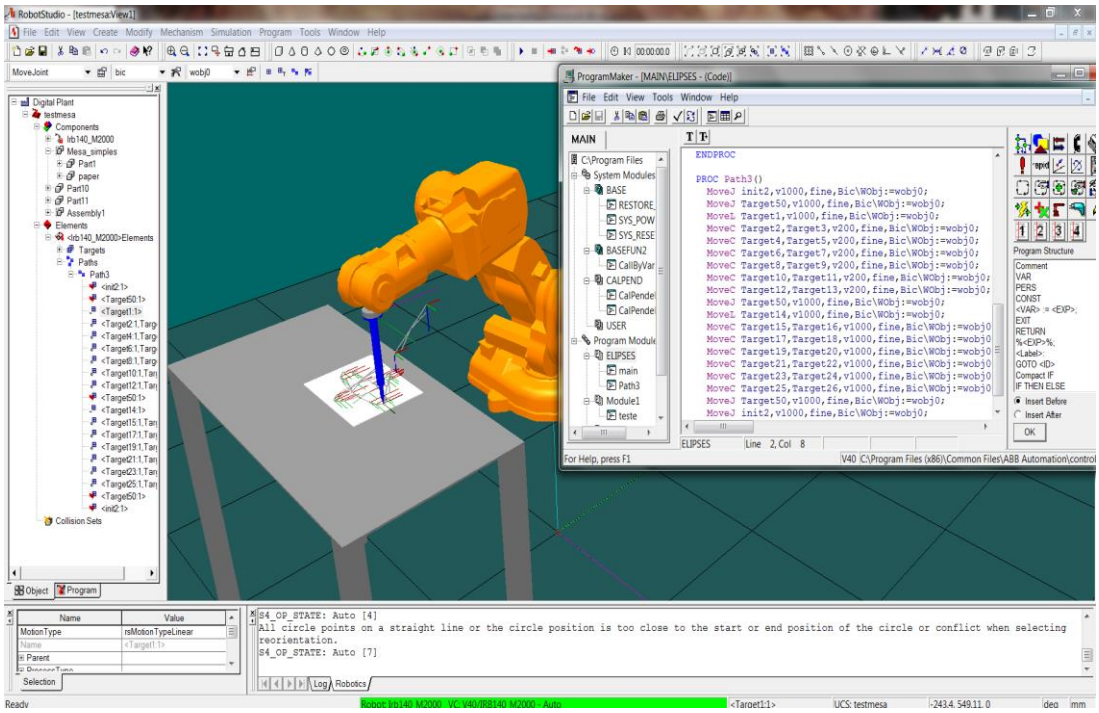


Figura 3.2 - Demonstrativo do Ambiente RobotStudio

Como pode-se perceber pela figura 3.2, a representação virtual é bem próximo ao sistema real, e com isso o programa é visivelmente confiável para realizar o que propõe. Esta representação normalmente é associada a telas gráficas de monitoramento e supervisão e hardware de instrumentação associado.

3.3. Integração a partir do aplicativo LABVIEW

O ambiente Labview mostra-se uma ferramenta importante para esse desenvolvimento. A seguir apresentaremos o conceito de integração de sistemas automatizados a partir do Labview com ênfase no desenvolvimento de telas de supervisão, controle e instrumentação virtual.

3.3.1. Software de Supervisão e Controle Baseado no LABVIEW

Por ser um software que utiliza de ícones gráficos intuitivos e ligações que se assemelha a um fluxograma, é usado por milhões de engenheiros e cientistas para desenvolvimento de medidas, simulações, testes, controle e supervisão de sistemas. Ele oferece integração com milhares de dispositivos e fornece centenas de projetos em suas bibliotecas para analise avançada de visualização de instrumentação virtual.

É um programa de alta flexibilidade que através da abstração da linguagem de baixo nível e integrando as ferramentas necessárias, permite ao usuário ir desde o projeto até os testes de sistemas simples e complexos refinando seus processos para se atingir o máximo de desempenho.

A programação é feita a partir de diagramas de blocos, facilitando a visualização e o entendimento do problema.

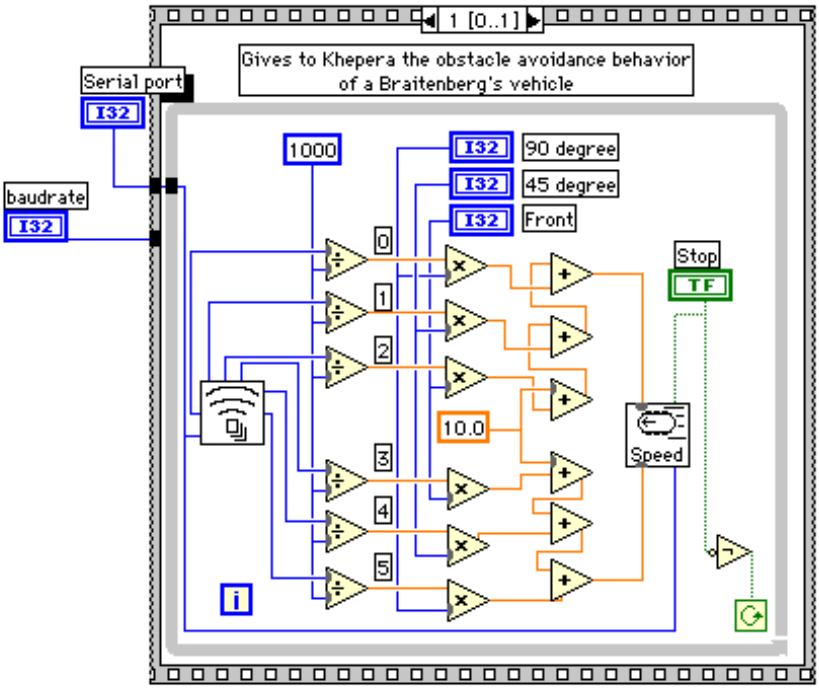


Figura 3.3 - Exemplo de Programação em LabView

O Labview além de fazer simulações, também pode ser utilizado em sistemas reais ou ainda integrando com tais sistemas para supervisioná-los, através das ferramentas que dispõe para aquisição e envio de dados.

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, podem-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.

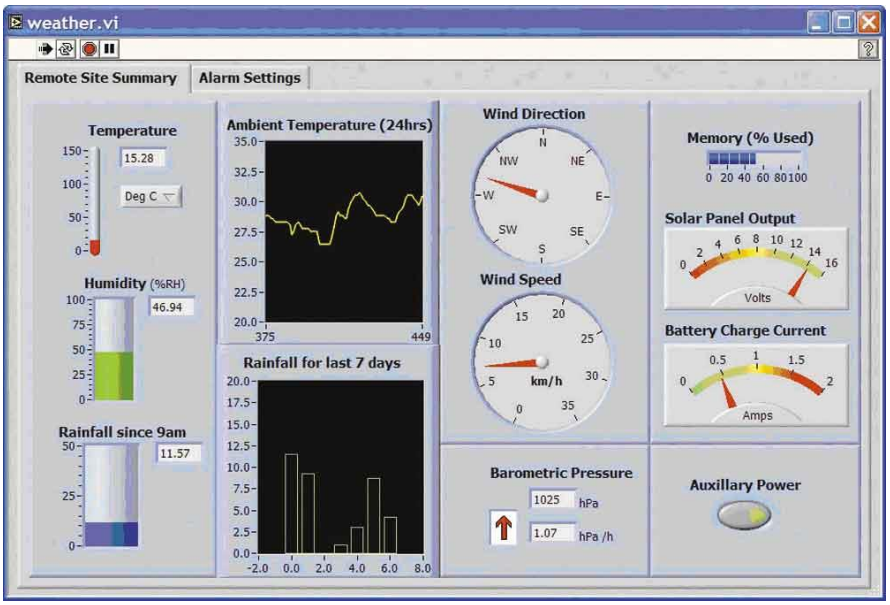


Figura 3.4 - Exemplo de Supervisório em LabView

3.3.2. Hardware dedicado a instrumentação virtual

A National Instruments oferece cinco plataformas PAC baseadas em LabVIEW: PXI, CompactPCI, CompactFieldPoint, CompactVisionSistem, CompactRIO, que detalharemos a seguir:

a) 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.

b) Compact FieldPoint

O Compact FieldPoint consiste de módulos de E/S analógicas e digitais que são hotswappable 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.

c) 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.

d) 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.

3.4. Considerações Finais

Aqui foram expostas as ferramentas utilizadas neste trabalho para a concepção de uma célula robótica, fazendo-se uma descrição destas e uma breve explicação do porque de sua escolha. Não havendo correlação nenhuma com a metodologia, sendo que esta pode ser aplicada a quaisquer aplicativos e dispositivos que sejam escolhidos pelas partes envolvidas no projete de uma célula robótica.

CAPÍTULO 4

4. Estudo de Casos

Primeiramente foi definido um problema, a partir deste problema pode-se fazer um estudo teórico e referencial bibliográfico sobre as técnicas e tecnologias disponíveis para a solução de tal problema, sabendo-se das ferramentas existentes no mercado atual, cria-se uma metodologia, pode-se dizer que foi sistematizado um meio, ou caminho, pelo qual pode-se seguir para se chegar à concepção de uma célula robótica flexível (CRF).

A partir das pesquisas e definições dos capítulos anteriores se podem chegar a elaboração da CRF. E para fundamentar a metodologia proposta no trabalho, neste capítulo apresentarei um estudo de caso, ou seja, uma proposta de uma célula flexível, criada a partir de aplicativos que foram descritos no capítulo 3.

4.1. Apresentação da célula robótica flexível

No ambiente em que foi realizado este trabalho, dispunha-se de poucos recursos para a criação de uma célula robótica, bem como de pouco espaço físico, levando em consideração estes fatos, se propôs uma célula em que todos os problemas focem solucionados da melhor forma e o mais rápido possível. Para tanto se propõe uma tarefa para a célula que será virtualmente simulada para posteriormente ser implementada em uma CRF real, é sempre válido ressaltar que a célula poderia ser outra e com outras tarefas, já que o objetivo é validar a metodologia proposta, esta célula é apenas um exemplo para tal validação.

A célula robótica flexível em estudo teve seus elementos escolhidos com base na disponibilidade, já que este também é um ponto chave neste trabalho. Todos se encontram no

Laboratório de Automação Integrada e Robótica (LAIR) da Faculdade de Engenharia Mecânica que fica situada na Universidade Estadual de Campinas (Unicamp), onde será implementada a célula.

Logo após a escolha dos aplicativos e tecnologias, faz-se a preparação para programação e integração dos dispositivos, só então se faz a programação off-line do robô.

É de extrema importância ressaltar que a metodologia proposta é geral podendo-se emprega-la para qualquer dispositivo e em quantidades indeterminadas. Sendo que a concepção de tal tipo de projeto é limitado apenas pelo orçamento e pela escolha dos softwares e conhecimento empregado.

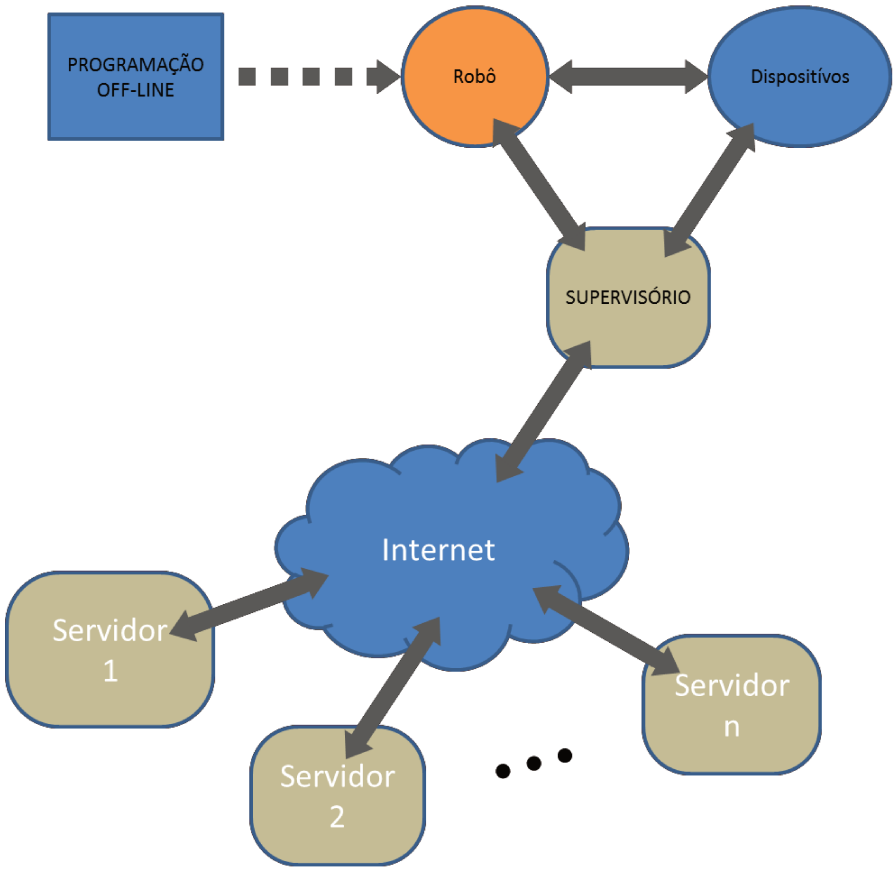


Figura 4.1 - Célula dada pela metodologia

A célula que será estudada é formada por:

- Um robô industrial ABB IRB140.
- Uma Plataforma Robótica de Posicionamento (PRP) que foi desenvolvida no próprio LAIR.
- Computador dedicado para programação e supervisão.
- Softwares para programação off-line.
- Software para controle e supervisão da célula.
- Placas de aquisição de dados.
- CLP's que controlaram e integrarão os componentes.
- Servidores externos caso se faça necessária operações à distância.
- Sistema de visão para tele operação.

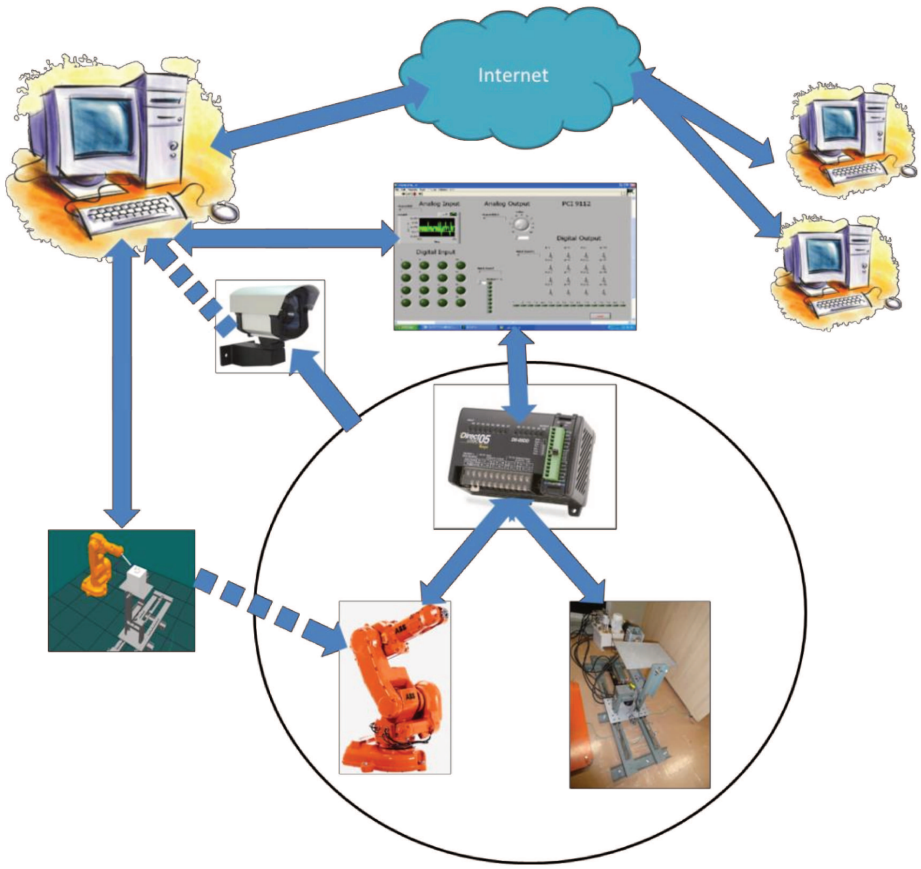


Figura 4.2 - Célula em Proposta

Com a célula determinada proponho um estudo de caso para a simulação virtual dos dispositivos.

Neste estudo, definiu-se uma tarefa para a célula e a partir desta, pode-se realizar todas as elaborações dos programas que seriam submetidos à simulação. A programação off-line do robô é uma etapa muito importante para a concepção desta metodologia, pois é a partir desta que puderam ser realizadas todas as simulações, que evitaram complicações futuras para na célula real, já que a simulação é capaz de prever possíveis colisões com o ambiente.

O objetivo da célula é trabalhar uma peça qualquer utilizando uma ponteira reta, que simula uma ponteira de solda, ao robô que faz com que ele faça uma trajetória sobre a peça a ser confeccionada, já a função da Plataforma de Posicionamento é rotacionar a peça de maneira que esta fique em posição para o robô já que ele não tem um grande alcance o que dificulta o trabalho. Esta tarefa foi escolhida por ser uma das mais usadas para o modelo de robô empregado, que é a solda em espaço reduzido.

4.2.Descrição dos Elementos de Integração

4.2.1. Robô ABB IRB-140

Este dispositivo robótico foi admitido para a célula estudada pela disponibilidade e compatibilidade com os recursos empregados. Este é um robô de pequeno porte com movimentos curtos, porém bem estruturado e de grande flexibilidade, pois pode ser utilizado para aplicação em várias posições por ser inteiramente blindado, não havendo vazamento de fluidos, mesmo suspenso.

O robô IRB-140 atende todas as necessidades da célula proposta, por ser de um tamanho reduzido, permitindo que realize tarefas em ambientes com pouco espaço de trabalho, seu alcance é pequeno, porém será compensado pela PRP. Sendo um elemento chave na célula robotizada,

realizará a maior parte das tarefas da célula e para tanto apresenta uma boa confiabilidade e precisão.

Figura 4.3 - Robô ABB IRB-140

Além de estar presente no LAIR, também apresenta compatibilidade total com o software que será usado, pois sua representação virtual e controle estão presentes por serem ambos desenvolvidos pela mesma empresa (ABB).

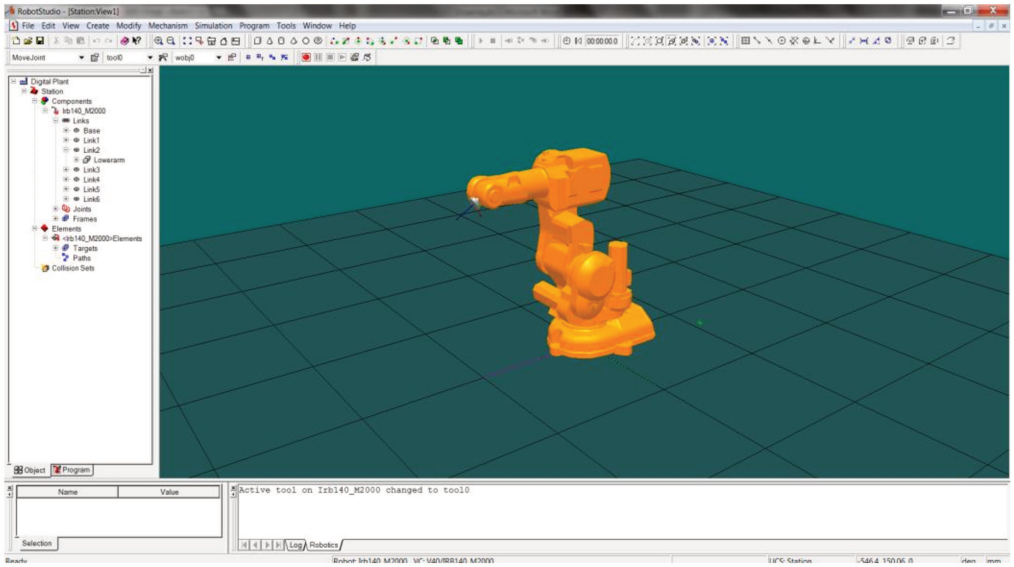


Figura 4.4 - IRB-140 em ambiente virtual

4.2.1.1. Calibração da Ferramenta Terminal

Existe a necessidade de realizar alguns procedimentos ao iniciar o trabalho com um robô industrial, dentre eles devemos considerar, para um perfeito funcionamento do sistema e melhor performance do robô, dois procedimentos básicos:

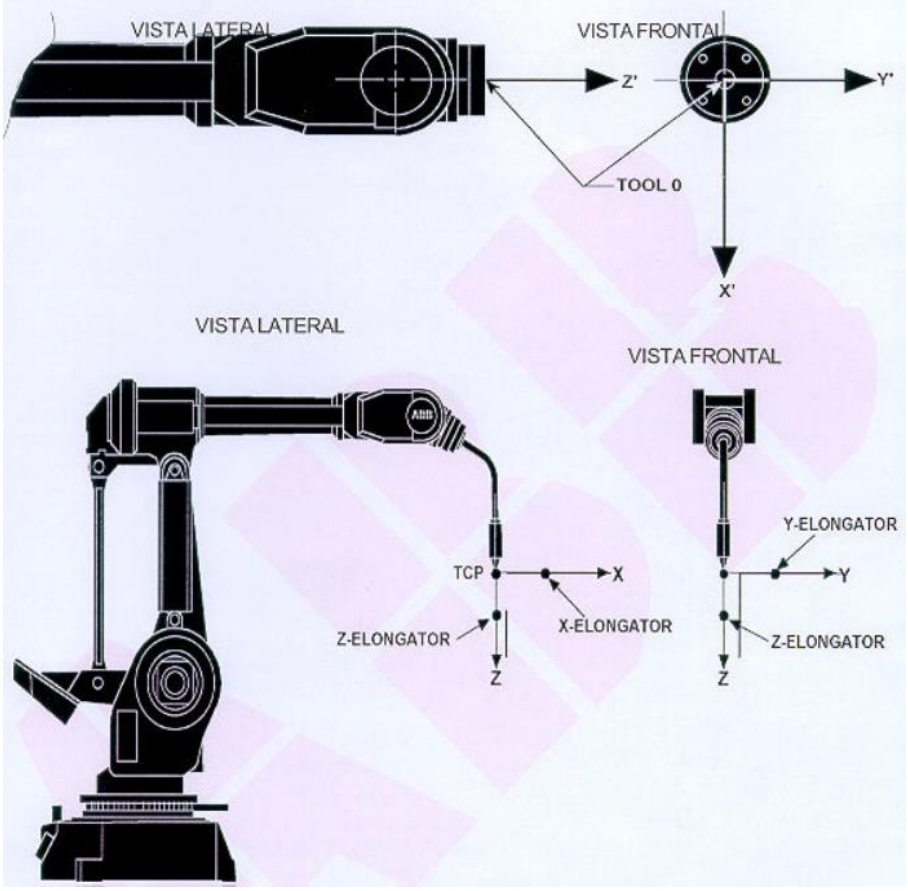


Figura 4.5 - Procedimento de calibração de ferramenta

- Sincronização das juntas: os contadores das informações provenientes dos encoders (ou resolvers no caso de interrupção de energia ou desconexão do robô) deverão ser zerados, com o robô posicionado nas marcas de calibração;

- Calibração de sua ferramenta terminal TCP: nesta etapa deve inicialmente informar ao sistema de controle do robô (painel de comando), as características principais da ferramenta (massa, dimensão, etc.), para que o sistema de controle possa realizar todo o procedimento de calibração da ferramenta terminal do robô (figura 4.5).

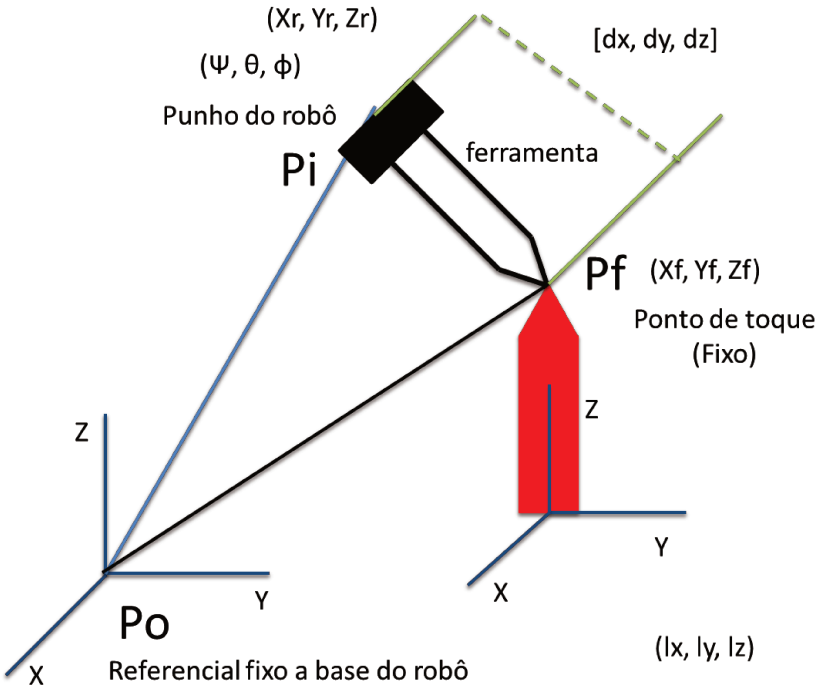


Figura 4.6 - Coordenadas para calibração da ferramenta terminal

Para obter a calibração da ferramenta, deve-se encontrar o ponto onde a ferramenta terminal se encontra no espaço, para isso faz se os cálculos que serão demonstrados aqui, onde as coordenadas estão mostradas na figura 4.6, podemos tirar da figura as seguintes considerações:

P(i) [X, Y, Z]: ponto P(i), dimensão 3 x 1

$$[\Psi(i), \theta(i), \Phi(i)] = \begin{bmatrix} Nx & Sx & Ax \\ Ny & Sy & Ay \\ Nz & Sz & Az \end{bmatrix}$$: matriz de orientação Euler (3 x 3)

$$Nx = C\Phi * C\theta; \qquad Ny = C\theta * S\Phi; \qquad Nz = -S\theta$$

$$Sx = C\Phi * S\theta * S\Psi - C\Psi * S\Phi; \qquad Sy = S\Phi * S\theta * S\Psi + C\Phi * C\Psi \qquad Sz = C\theta * S\Psi$$

$$Ax = C\Phi * S\theta * C\Psi + S\Phi * S\Psi; \qquad Ay = S\Phi * S\theta * C\Psi - C\Phi * S\Psi \qquad Az = C\theta * C\Psi$$

$$C\Psi = \cos(\Psi) \qquad C\theta = \cos(\theta) \qquad C\Phi = \cos(\Phi)$$

$$S\Psi = \sin(\Psi) \qquad S\theta = \sin(\theta) \qquad S\Phi = \sin(\Phi)$$

Com as informações dadas temos os parâmetros de medida, onde os pontos são:

$P(0): [X(0), Y(0), Z(0)], [\Psi(0), \theta(0), \Phi(0)]$: ponto 0, referencial fixo a base do robô ou, zero do robô, bem conhecido (0, 0, 0).

$P(i): [X(i), Y(i), Z(i)], [\Psi(i), \theta(i), \Phi(i)]$: ponto i, ponto das medidas efetuadas em relação ao ponto fixo de calibração, lido no punho do robô e é perfeitamente conhecido a partir do robô.

$Pt: [Xt, Yt, Zt], [\Psi t, \theta t, \Phi t]$: ponto de contato da ferramenta terminal (tool), ele é fixo, mas este ponto deverá ser estimado e atualizado em cada iteração.

[dx, dy, dz]: dimensões da ferramenta são atualizadas a cada iteração.

[lx, ly, lz]: distância do ponto de contato de ferramenta (Pt) até o referencial zero do robô P(0) e também é atualizado a cada iteração.

Com os parametros encontrados e determinados usamos as seguintes equações:

$PoPi = PoPt + PtPi$ (equação vetorial)

$$[P(i) - P(0)]_{3x1} = [Pt - P(0)]_{3x1} + [P(i) - Pt]_{3x1} \qquad (1)$$

$$[P(i) - Pt]_{3x1} = [\Psi(i) - \Psi t, \theta(i) - \theta t, \Phi(i) - \Phi t]_{3x3} * [dx, dy, dz]_{3x1} \qquad (2)$$

$$[Pt - P(0)]_{3x1} = [\Psi t, \theta t, \Phi t]_{3x3} * [lx, ly, lz]_{3x1} \qquad (3)$$

Pode-se perceber que se somarmos as equações (2) + (3) = (1), assim:

$$[\mathrm{P(i)} - \mathrm{P(0)}]_{3x1} = [\Psi(\mathrm{i}) - \Psi\mathrm{t}, \theta(\mathrm{i}) - \theta\mathrm{t}, \Phi(\mathrm{i}) - \Phi\mathrm{t}]_{3x3} * [\mathrm{dx}, \mathrm{dy}, \mathrm{dz}]_{3x1} + [\Psi\mathrm{t}, \theta\mathrm{t}, \Phi\mathrm{t}]_{3x3} * [\mathrm{lx}, \mathrm{ly}, \mathrm{lz}]_{3x3}$$

Onde:

$\mathrm{P(i)} = [\mathrm{X(i)}, \mathrm{Y(i)}, \mathrm{Z(i)}]$: vetor de dimensão (3x1) obtido a partir do robô seu valor é conhecido.

$\mathrm{P(0)} = [\mathrm{X(0)}, \mathrm{Y(0)}, \mathrm{Z(0)}]$: vetor de dimensão (3x1) obtido a partir do robô zero do robô seu valor é conhecido: $\mathrm{P(0)} = [0, 0, 0]$

$[\Psi(\mathrm{i}+1) - \Psi(1), \theta(\mathrm{i}+1) - \theta(1), \Phi(\mathrm{i}+1) - \Phi(1)]$: matriz de orientação (3x3) conhecida e resultante entre o ponto 1 do palete e o ponto de medida i+1 e seu valor deverá ser sempre atualizado.

$[\Psi(0), \theta(0), \Phi(0)]$: matriz de orientação (3x3) zero do robô é uma matriz identidade.

$[\Psi(\mathrm{i}), \theta(\mathrm{i}), \Phi(\mathrm{i})]$: matriz de orientação (3x3) relativa ao ponto i de toque, obtida sempre a partir do robô.

$[\Psi\mathrm{t}, \theta\mathrm{t}, \Phi\mathrm{t}]$: matriz de orientação (3x3) do ponto Pt em relação ao zero P(0), seu valor deverá sempre ser atualizado.

Assim apresenta-se apenas o básico dos cálculos utilizados, veja em anexo para ter o calculo completo de calibração de ferramenta terminal.

4.2.1.2. Identificação do ponto zero do Robô

A integração de robôs no ambiente de trabalho necessitam do posicionamento real do mesmo em relação às tarefas a serem realizadas através da programação e elaboração de trajetórias pré-definidas em robôs, independente da operação.

Este processo pode ser otimizado através da utilização de softwares industriais para programação Off-Line, que permitem simular com relativa precisão todo o trabalho a ser realizado por um robô, desde a sua instalação, otimização de tarefas e carregamento do programa no robô.

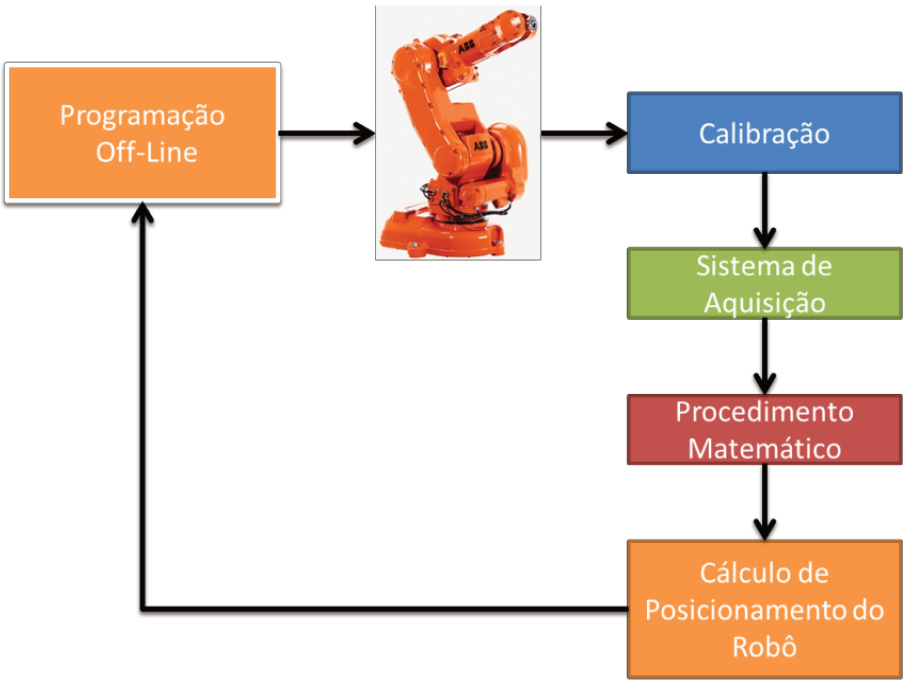


Figura 4.7 - Sistema para calibração

A qualidade da metodologia utilizada para programação é função do posicionamento real do robô em relação ao referencial zero. A execução de tarefas pré-programadas deverá considerar a identificação exata dos sistemas de referências externos ao robô, através de procedimentos de calibração automáticos, que a partir de uma série de medidas simples predefinidas será possível à

determinação do sistema real de posicionamento do robô em relação ao referencial zero e em relação ao sistema externo de medição. A figura 4.7 mostra as diferentes etapas descritas anteriormente que serão estudadas e avaliadas para realizar o objetivo preconizado neste projeto de pesquisa. (Freitas et. all, 2004)

Sabendo-se disto podemos realizar alguns cálculos para determinar o posicionamento do robô, este cálculo mostra ao robô onde ele irá trabalhar.

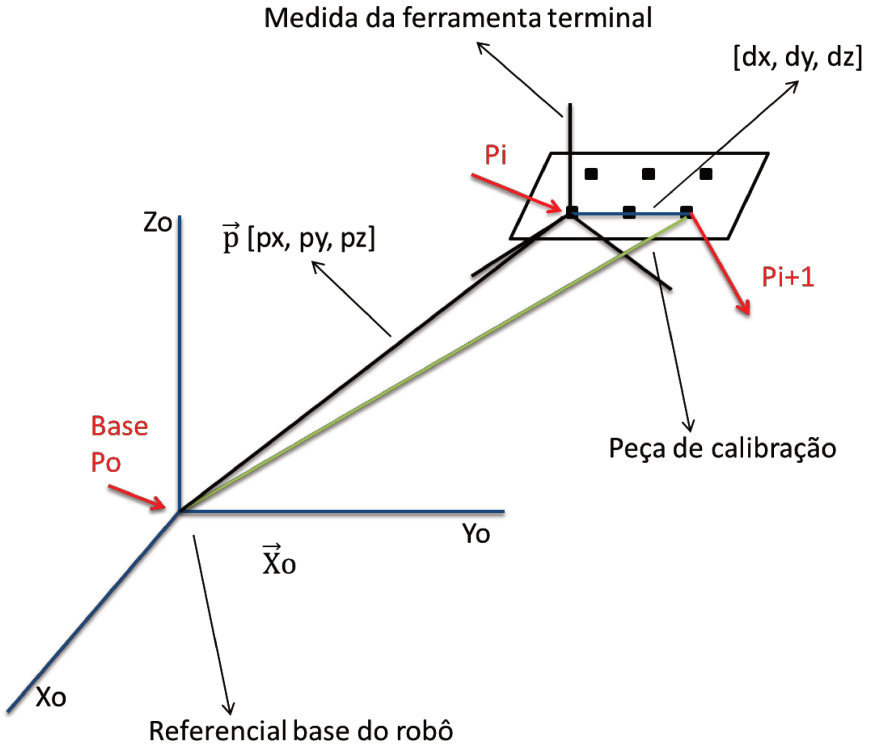


Figura 4.8 - Método de calibração zero do robô

A partir da figura 4.8, podemos obter as seguintes informações:

$P(i)[X, Y, Z]$: Ponto P(i), dimensão 3 x 1

$[\Psi(i), \theta(i), \Psi(i)] = \begin{bmatrix} Nx & Sx & Ax \\ Ny & Sy & Ay \\ Nz & Sz & Az \end{bmatrix}$: matriz de orientação Euler (3 x 3)

$Nx = C\Phi * C\theta;$ $\quad Ny = S\Phi * C\theta;$ $\quad Nz = -S\theta$

$Sx = C\Phi * S\theta * S\Psi - C\Psi * S\Phi;$ $Sy = S\Phi * S\theta * S\Psi + C\Phi * C\Psi;$ $Sy = C\theta * S\Psi$

$Ax = C\Phi * S\theta * C\Psi + S\Phi * S\Psi;$ $Ay = S\Phi * S\theta * C\Psi - C\Phi * S\Psi;$ $Az = C\theta * C\Psi$

$CPsi = \cos(\Psi)$ $C\theta = \cos(\theta)$ $CPhi = \cos(\Phi)$

$SPsi = \sin(\Psi)$ $S\theta = \sin(\theta)$ $SPhi = \sin(\Phi)$

$$RPY\ [\Phi(z), \theta(y), \Psi(x)] = Rot\ [z, \Phi] * Rot\ [y, \theta] * Rot\ [x, \Psi]$$

$$= \begin{bmatrix} C\Phi & -S\Phi & 0 \\ S\Phi & C\Phi & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\Psi & -S\Psi \\ 0 & S\Psi & C\Psi \end{bmatrix}$$

$$= \begin{bmatrix} Nx & Sx & Ax \\ Ny & Sy & Ay \\ Nz & Sz & Az \end{bmatrix}$$

Utilizando a matriz de orientação Euler (3 x 3) que representa o mesmo sistema do robô:

$Nx = C\Phi * C\theta;$ $Ny = S\Phi * C\theta;$ $Nz = -S\theta$

$Sx = C\Phi * S\theta * S\Psi - C\Psi * S\Phi;$ $Sy = S\Phi * S\theta * S\Psi + C\Phi * C\Psi;$ $Sz = C\theta * S\Psi$

$Ax = C\Phi * S\theta * C\Psi + S\Phi * S\Psi;$ $Ay = S\Phi * S\theta * C\Psi - C\Phi * S\Psi;$ $Az = C\theta * C\Psi$

$C\Psi = \cos(\Psi)$ $C\theta = \cos(\theta)$ $C\Phi = \cos(\Phi)$

$S\Psi = \sin(\Psi)$ $S\theta = \sin(\theta)$ $S\Phi = \sin(\Phi)$

Com os parâmetros podemos fazer o cálculo do Jacobiano para pequenas rotações:

δ_x : Rotação infinitesimal em torno do eixo x

δ_y : rotação infinitesimal em torno do eixo y

δ_z : rotação infinitesimal em torno do eixo z

Aproximação: $\sin(\theta) = \theta,$ $\cos(\theta) = 1$

$$Rot\,[x,\delta_x] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -\delta_x \\ 0 & \delta_x & 1 \end{bmatrix}$$

$$Rot\,[y,\delta_y] = \begin{bmatrix} 1 & 0 & \delta_y \\ 0 & 1 & 0 \\ -\delta_y & 0 & 1 \end{bmatrix}$$

$$Rot\,[z,\delta_z] = \begin{bmatrix} 1 & -\delta_z & 0 \\ \delta_z & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Podendo encontrar a matriz de transformação diferencial que denota as rotações em (Z, Y, X)

$$RPY\,[\delta_z,\delta_y,\delta_x] = Rot\,[z,\delta_z] * Rot\,[y,\delta_y] * Rot\,[x,\delta_x]$$

$$= \begin{bmatrix} 1 & -\delta_z + \delta_y\delta_x & \delta_z\delta_x + \delta_y \\ \delta_z & 1 + \delta_z\delta_y\delta_x & -\delta_z + \delta_y\delta_x \\ -\delta_y & \delta_x & 1 \end{bmatrix} = \begin{bmatrix} 1 & -\delta_z & \delta_y \\ \delta_z & 1 & -\delta_z \\ -\delta_y & \delta_x & 1 \end{bmatrix}$$

Obs: simplificação de termos de 2ª e 3ª ordens

$Nx = 1$; $Ny = \delta_z$; $Nz = -\delta_y$

$Sx = -\delta_z$; $Sy = 1$; $Sz = \delta_x$

$Ax = \delta_y$; $Ay = -\delta_z$; $Az = 1$;

Teorema 1: Uma rotação diferencial δ_θ em torno de um vetor K [K_x, K_y, K_z] é equivalente a três rotações sucessivas [δ_z, δ_y, δ_x] em torno dos eixos x, y, z.

$$K_x,\delta_\theta = \delta_x$$

$$K_y,\delta_\theta = \delta_y$$

$$K_z,\delta_\theta = \delta_z$$

Teorema 2: Rotações diferenciais são independentes da ordem de rotação (quando desprezamos os termos de segunda e terceira ordem).

$$Rot\left[y,\delta_y\right] * Rot\left[x,\delta_x\right] = Rot\left[x,\delta_x\right] * Rot\left[y,\delta_y\right]$$

Tendo os parâmetros de medida:

a) Pontos de referência:

$P(0)$: $[X(0), Y(0), Z(0)], [\Psi(0), \theta(0), \Phi(0)]$: ponto 0, referencial fixo a base do robô.

$P(1)$: $[X(1), Y(1), Z(1)], [\Psi(1), \theta(1), \Phi(1)]$: ponto 1, primeiro ponto de medida que é conhecido a partir do robô.

$P(i+1)$: $[X(i+1), Y(i+1), Z(i+1)], [\Psi(i+1), \theta(i+1), \Phi(i+1)]$: Ponto i+1, ponto qualquer de medida que também é conhecido através do robô.

b) Parâmetros desconhecidos e valores iniciais atribuídos:

$P(0)$: $[X(0), Y(0), Z(0)] = [0\ 0\ 0] \quad [\Psi(0), \theta(0), \Phi(0)] = [0\ 0\ 0]$

$[px, py, pz]$: distância do ponto P(0) (zero do robô) até o primeiro ponto da peça (atualizado a cada iteração).

c) Parâmetros bem conhecidos:

$P(1)$: $[X(1), Y(1), Z(1)], [\Psi(1), \theta(1), \Phi(1)]$: obtido através do robô e deverá ser atualizado a cada iteração pelo valor real do robô considerando o novo zero do robô obtido anteriormente.

$P(i+1) = [X(i+1), Y(i+1), Z(i+1)], [\Psi(i+1), \theta(i+1), \Phi(i+1)]$: obtido através do robô e também deverá ser atualizado a cada iteração pelo valor real do robô considerando o novo zero do robô obtido anteriormente

$[dx, dy, dz]$: vetor de posição ao longo da peça, ponto em relação a P(1) que é conhecido através do desenho da peça de calibração utilizada.

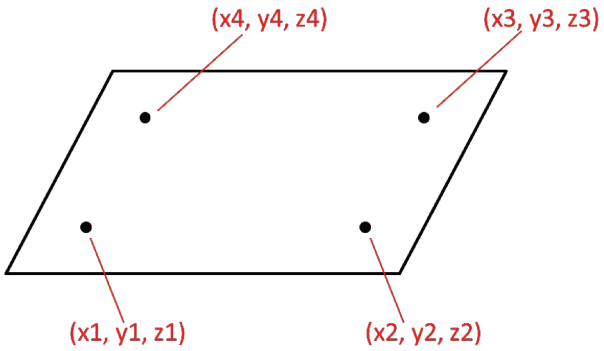


Figura 4.9 - Peça de calibração

Equações utilizadas:

$$[P(1) - P(0)] = [\Psi(0), \theta(0), \Phi(0)] * [\Psi(1), \theta(1), \Phi(1)] * [px, py, pz] \quad (1)$$

$$[P(i+1) - P(1)] = [\Psi(i+1) - \Psi(i), \theta(i+1) - \theta(i), \Phi(i+1) - \Phi(i)] * [dx, dy, dz] \quad (2)$$

$$[P(i+1) - P(0)] = [P(i+1) - P(1)] + [P(1) - P(0)] \quad (3)$$

Então, se somarmos as equações (1) + (2) teremos a equação (3) que pode ser escrita da seguinte forma:

$$[P(i+1) - P(0)] = [\Psi(i+1) - \Psi(1), \theta(i+1) - \theta(1), \Phi(i+1) - \Phi(1)] * [dx, dy, dz] +$$

$$+ [\Psi(0), \theta(0), \Phi(0)] * [\Psi(1), \theta(1), \Phi(1)] * [px, py, pz]$$

Onde:

$P(i+1) = X(i+1), Y(i+1), Z(i+1)$: vetor de dimensão (3 x 1) obtido a partir do robô e seu valor deverá ser sempre atualizado.

$P(0) = X(0), Y(0), Z(0)$: vetor de dimensão (3 x 1) obtido a partir do robô, seu valor deverá ser sempre atualizado.

$[dx, dy, dz]$: vetor de dimensão (3 x 1) conhecido a partir do peça de calibração

$[px, py, pz]$: vetor de dimensão (3 x 1) relativo a distância do ponto 1 da peça ao zero do robô, este valor estimado e atualizado a cada medida.

$[\Psi(i+1) - \Psi(1), \theta(i+1) - \theta(1), \Phi(i+1) - \Phi(1)]$: matriz de orientação (3 x 3) conhecida e resultante entre o ponto 1 da peça de calibração e o ponto de medida i+1, seu valor deverá ser sempre atualizado.

$[\Psi(0), \theta(0), \Phi(0)]$: matriz de orientação (3 x 3) estimada a partir do zero do robô, valor estimado e atualizado a cada medida.

$[\Psi(1), \theta(1), \Phi(1)]$: matriz de orientação (3 x 3) relativa ao ponto 1 do pacote conhecida a partir do robô deve-se sempre atualizar seu valor.

As equações de medição genéricas, são dadas por:

$$X(i+1) = X(0) + N_1x * dx + S_1x * dy + A_1x * dz + N_2x * px + S_2x * py + A_2x * pz$$

$$Y(i+1) = Y(0) + N_1y * dx + S_1y * dy + A_1y * dz + N_2y * px + S_2y * py + A_2y * pz$$

$$Z(i+1) = Z(0) + N_1z * dx + S_1z * dy + A_1z * dz + N_2z * px + S_2z * py + A_2z * pz$$

Para obter os valores de X(0), Y(0) e Z(0) que são estimados a cada iteração. E para a obtenção de N_1, S_1, A_1 e dx, dy, dz que são constantes, mas devendo ser sempre atualizados pelas informações do robô e da peça:

$N_1x = \cos[\Phi(i+1) - \Phi(1)] * \cos[\theta(i+1) - \theta(1)];$

$N_1y = \cos[\theta(i+1) - \theta(1)] * \sin[\Phi(i+1) - \Phi(1)];$

$N_1z = -\sin[\theta(i+1) - \theta(1)];$

$$S_1x = \cos[\Phi(i+1) - \Phi(1)] * \sin[\theta(i+1) - \theta(1)] * \sin[\Psi(i+1) - \Psi(1)] + \\ - \cos[\Psi(i+1) - \Psi(1)] * \sin[\Phi(i+1) - \Phi(1)];$$

$$S_1x = \sin[\Phi(i+1) - \Phi(1)] * \sin[\theta(i+1) - \theta(1)] * \sin[\Psi(i+1) - \Psi(1)] + \\ + \cos[\Phi(i+1) - \Phi(1)] * \cos[\Psi(i+1) - \Psi(1)];$$

$$S_1z = \cos[\theta(i+1) - \theta(1)] * \sin[\Psi(i+1) - \Psi(1)];$$

$$A_1x = \cos[\Phi(i+1) - \Phi(1)] * \sin[\theta(i+1) - \theta(1)] * \cos[\Psi(i+1) - \Psi(1)] + \\ + \sin[\Phi(i+1) - \Phi(1)] * \sin[\Psi(i+1) + \Psi(1)];$$

$$A_1y = \sin[\Phi(i+1) - \Phi(1)] * \sin[\theta(i+1) - \theta(1)] * \cos[\Psi(i+1) - \Psi(1)] + \\ + \sin[\Phi(i+1) + \Phi(1)] * \sin[\Psi(i+1) - \Psi(1)];$$

$$A_1z = \cos[\theta(i+1) - \theta(1)] * \cos[\Psi(i+1) - \Psi(1)];$$

[dx dy dz]: obtidos diretamente a partir das informações da peça de calibração.

Obs.: os valores Φ(i+1), Φ(1), θ(i+1), θ(1), Ψ(i+1), Ψ(1) são todos valores obtidos diretamente através do robô. Observar que através dessa diferença, obtemos precisamente o posicionamento da peça de calibração em relação ao robô e possíveis diferenças na orientação da ferramenta no momento do toque na peça.

Para não deixar excessivamente extenso, foram apresentados apenas parte dos cálculos de posicionamento, assim sendo, o restante dos cálculos encontram-se em anexo neste trabalho.

4.2.2. Plataforma Robótica de Posicionamento

A Plataforma é formada por três partes que apresentam movimento:

- Sua base com uma trajetória linear horizontal sobre um trilho, o movimento é dado por um cilindro hidráulico e medido por um potenciômetro.
- A segunda parte gira em torno de um eixo perpendicular ao solo, seus movimentos são medidos por um encoder.
- Na ultima parte da PRP os movimentos são giratórios, porém paralelos ao solo, sua medição também se dá através de encoder, esta também é a base de apoio onde será fixada a peça a ser trabalhada pelo robô.

Figura 4.10 – Plataforma Robótica de Posicionamento

O acionamento da PRP é puramente hidráulico e deve ser feito a partir de uma placa específica que controlará as bobinas que liberam ou impedem o fluido de transcorrer para os motores fazendo a movimentação da mesma.

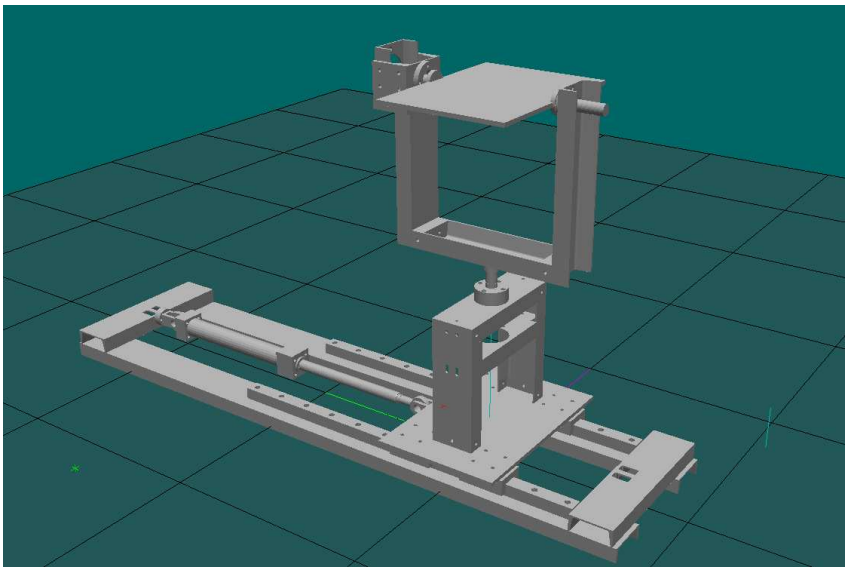


Figura 4.11 – Plataforma de Posicionamento em representação virtual

4.2.2.1. Modelagem Cinemática Direta

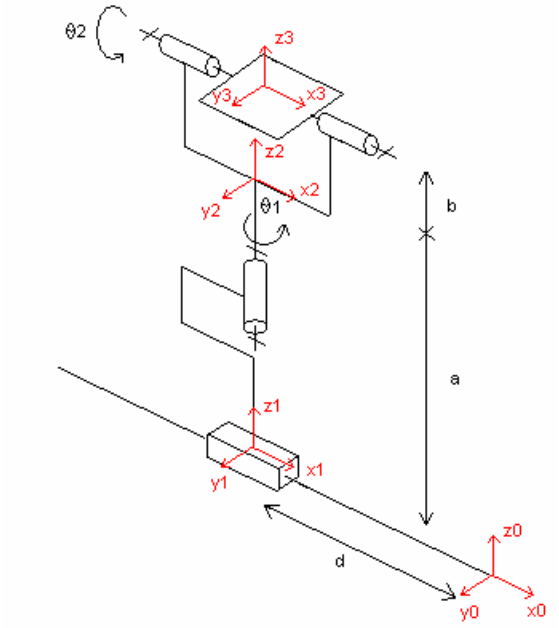


Figura 4.12 - Esquema do modelo matemática da Plataforma

Para determinar o modelo geométrico, torna-se necessário à implementação do mesmo através da convenção de Denavit-Hartenberg (“D-H”), a partir de coeficientes contidos em tabela correspondente a posição relativa dos diferentes graus de liberdade do dispositivo (figura 4.12).

Tabela 1 - Parametros para modelagem matemática (PPR)

J	α_j	d_j	θ_j	r_j
1	0	D	0	0
2	0	0	θ_1	A
3	θ_2	0	0	B

A partir desses parâmetros, é obtida a matriz de passagem apresentada a seguir:

$$\begin{bmatrix} \cos\theta_j & -\sin\theta_j & 0 & d_j \\ \cos\alpha_j \sin\theta_j & \cos\alpha_j \cos\theta_j & -\sin\alpha_j & -r_j \sin\alpha_j \\ \sin\alpha_j \sin\theta_j & \sin\alpha_j \cos\theta_j & \cos\alpha_j & r_j \cos\alpha_j \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

obtendo as matrizes de passagem descritas a seguir:

$${}^0T_1 = \begin{bmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^1T_2 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^2T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_2 & -\sin\theta_2 & 0 \\ 0 & \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$
{}^{0}T_{3}\begin{bmatrix}
\cos\theta_1 & -\sin\theta_1\cos\theta_2 & \sin\theta_1\sin\theta_2 & d \\
\sin\theta_1 & \cos\theta_1\cos\theta_2 & -\cos\theta_1\sin\theta_2 & 0 \\
0 & \sin\theta_2 & \cos\theta_2 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

A partir da matriz de orientação 0T3 associada às coordenadas articulares, pode-se obter a posição e orientação final a partir da tabela .

$$
{}^{0}T_{3} = \begin{bmatrix}
s & n & a & x \\
s & n & a & y \\
s & n & a & z \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

onde s, n e a correspondem a três vetores para a orientação de x, y e z referente as coordenadas de referência da origem.

Estes cálculos são importantes para a modelagem da Plataforma, assim podendo se esquematizar e fazer todos os cálculos de posicionamento, do robô e da PRP.

4.3.Programação off-line

Com esta plataforma de programação é possível recriar virtualmente a célula automatizada, podendo assim programar o robô e fazer simulações do ambiente de trabalho, para que durante a implementação sejam minimizados os imprevistos, visto que é uma representação fiel do espaço da célula.

A PRP foi recriada em software CAD e importado para o RS, e com as medidas de posicionamento real, pode-se recriar uma célula robótica virtualmente. Assim a programação se torna mais próxima da realidade e ainda mais precisa.

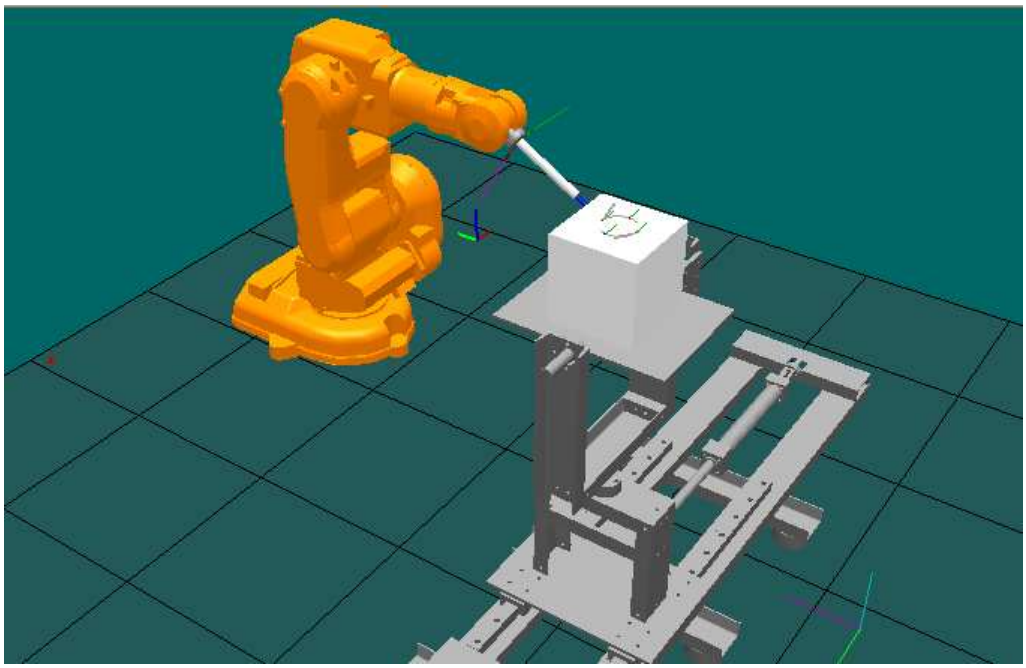


Figura 4.13 - Robô em Conjunto com a Plataforma

Uma vez importada a Plataforma, cria-se um objeto para trabalho que neste caso foi um cubo de dimensões de 200x200 mm, conforme pode-se visualizar através da figura 4.13.

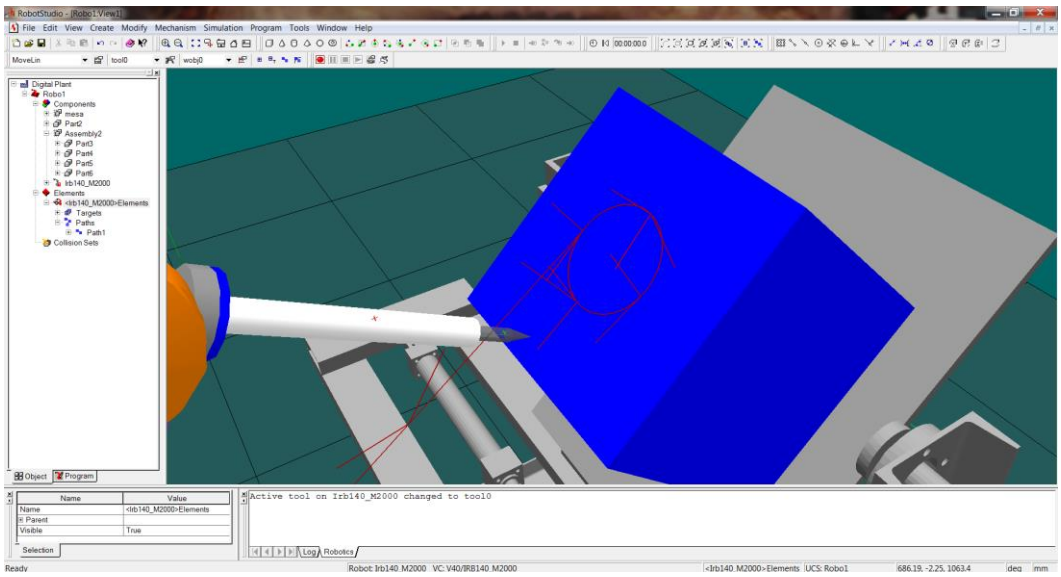


Figura 4.14 - Path para criação de programa

A programação off-line é feita apenas para o robô, este e a PRP são controlados separadamente por controladores distintos, porém se comunicam para realizar as tarefas em conjunto. Pelo fato da versão do software empregado ser uma versão antiga, esta não permite que se façam as movimentações da Plataforma robótica automaticamente pelo programa, portanto teve-se que utilizar das coordenadas manualmente para se fazer os giros e ângulos, conforme figura 4.15.

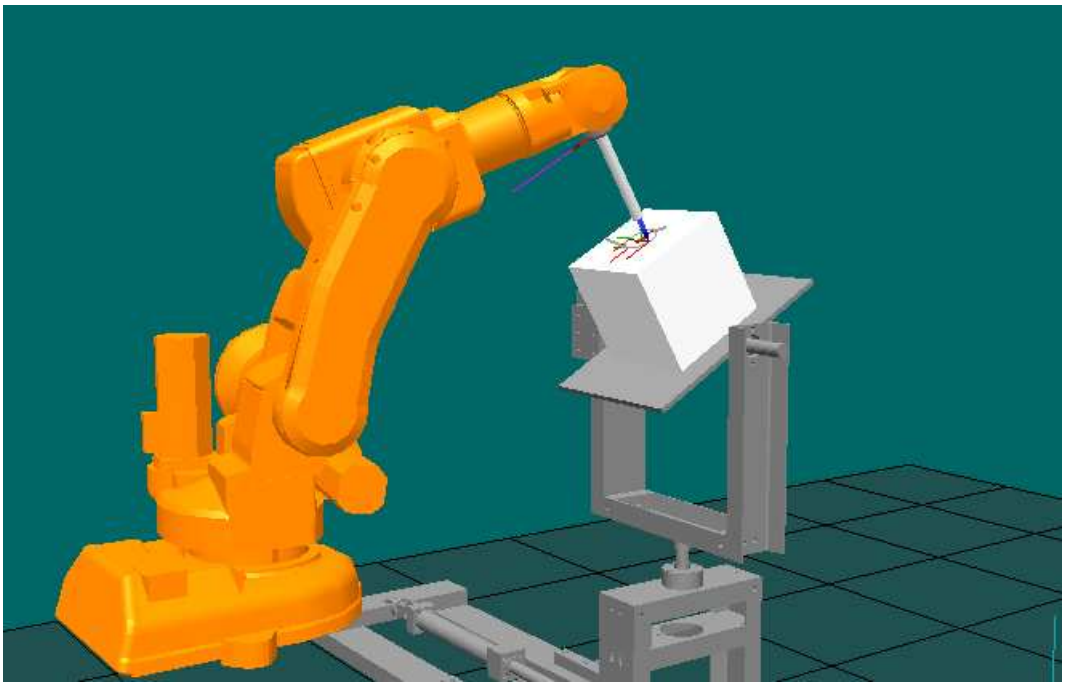


Figura 4.15 - Robô e PRP Realizando Tarefa em Conjunto

A movimentação da Plataforma é muito importante, pois sem ela, como pode-se perceber pela extensão do braço robótico que sem a angulação correta da PRP não seria possível a realização do trabalho, sendo que o robô não teria o alcance necessário para tal. Assim sendo demonstra-se a importância da integração entre os dispositivos empregados na célula.

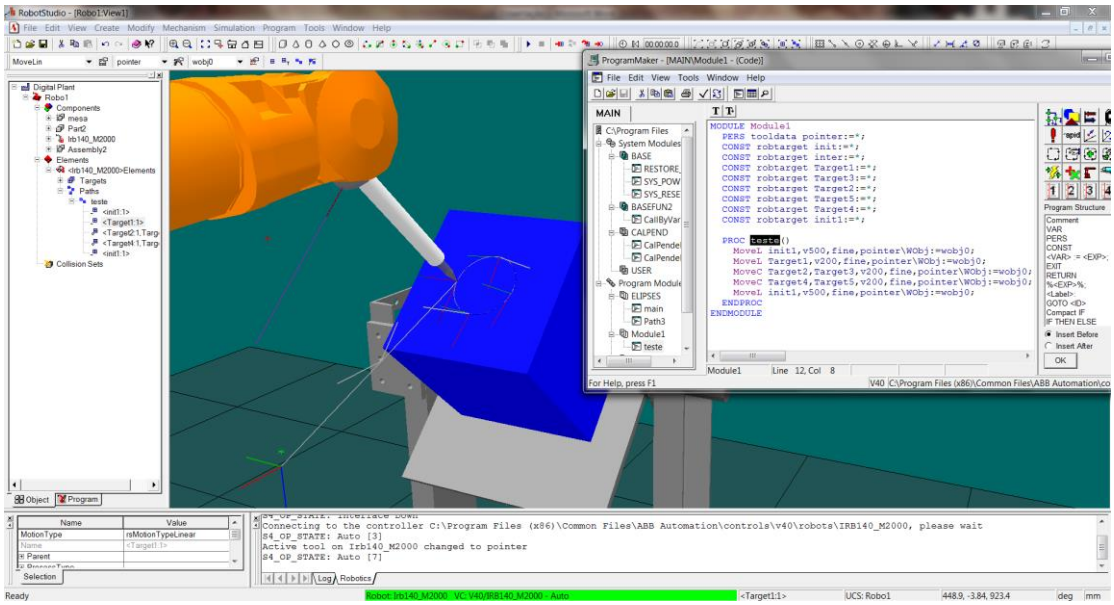


Figura 4.16 - Programa criado a partir de path

A Plataforma deve ser controlada a partir de outro software e feita sua integração com a célula através de um programa de controle que foi determinado que seria o LabView, este mesmo faz a supervisão da célula e integra os dispositivos.

4.4.Supervisão e Controle a partir do aplicativo Labview™

O LabView foi escolhido por motivos de afinidade com o laboratório, já que este vinha sendo empregado no LAIR já a algum tempo.

A plataforma robótica de posicionamento, como foi desenvolvida no LAIR, deve ser controlada a partir do software LabView, este software é responsável pelo controle da PRP e comunicação com o robô, assim pode-se fazer o sincronismo dos dois dispositivos da célula. Também é ele que faz a supervisão da célula e envia e recebem dados, podendo fazer o controle da célula via WEB, ou seja, controlar a célula de outro espaço físico.

Figura 4.17 - Equipamentos de aquisição e envio de dados para controle da célula

Numa visão mais interna da célula, o LabView recebe os sinais da Plataforma e do robô a través de sensores para poder coordenar os movimentos, com os sinais calcula-se os próximos movimentos necessários para o funcionamento da célula, e envia os dados para o robô e para a PRP executar as tarefas, para a tela para ser supervisionado e via WEB caso esteja sendo feito a supervisão a distância.

4.5.Sistema de Visão

O sistema de visão é formado por câmeras que terão a função de enviar imagens para o supervisório para então envia-las via WEB, ou seja, é este sistema que levará ao conhecimento externo, o que está sendo feito na célula em tempo real.

As câmeras também podem ser utilizadas juntamente com um software de imagem, para reconhecer os objetos que serão trabalhados e assim fazer uma seleção do programa a ser executado, automaticamente, assim retirando o erro de escolha manual. Porém este meio já é mais complexo e exigiria um tempo muito maior que o cronograma de realização do trabalho.

4.6. Considerações Finais

Neste capítulo é demonstrada a metodologia, e como aplica-la, através de uma simulação de célula robótica, descrevendo como foi usada para a concepção desta, e quais os dispositivos e porque foram escolhidos, assim exemplificando a metodologia.

Aqui também são mostrados os principais cálculos para a elaboração das calibrações e posicionamento dos dispositivos para que possam ser integrados e funcionem cooperativamente, assim podendo perceber a complexidade do projeto, demonstrando a necessidade e importância de uma formação específica para este tipo de concepção na área de automação e robótica.

CAPÍTULO 5

5. Validação experimental

5.1. Implementação da Célula Integrada

Para validar a metodologia proposta neste trabalho, foi realizada a implementação em ambiente físico, em uma célula robótica voltada a ensino e pesquisa no laboratório LAIR, da Faculdade de Engenharia Mecânica da UNICAMP.

Os elementos que constituem a célula robótica em estudo são:

- Robô ABB IRB-140: Sua função é trabalhar o objeto, sendo controlado pelo controlador que lhe é empregado pela própria fabricante, o qual recebe informações para seu funcionamento em conjunto com a célula.
- Controlador ABB: este tem a tarefa de controlar o robô, intercambiar informações entre o robô e o supervisor que dirá se pode continuar ou deve esperar mais instruções.
- Plataforma Robótica de Posicionamento: sua função é posicionar a peça para facilitar o trabalho do robô que muitas vezes não tem alcance suficiente para tal.
- PC para utilização dos softwares: este realiza a supervisão, abriga os softwares utilizados na célula e faz backup das informações e programas.
- Software RobotStudio: Software responsável pela programação do robô e caso necessário viabiliza seu controle a distância.
- CLP's: são responsáveis pelo controle e integração dos dispositivos, envia e recebe dados para o software de controle e supervisão.
- Software LabView: Responsável pela supervisão, controle, cálculos e intercâmbio de informações das partes da célula.
- Painel de aquisição de dados (I/O): este recebe os sinais dos sensores convertendo-os para uma linguagem entendível, armazena e envia dados para o funcionamento da célula.

- Câmera de visão: esta é usada para obter imagens e envia-las para supervisão via WEB, em tempo real, também grava imagens da célula para eventuais consultas.
- Objeto de trabalho: é a peça a ser trabalhada e posicionada.

Figura 5.1 - Célula robótica em estudo

5.2.Funcionalidade da Célula Robotizada

Para fins acadêmicos a célula real utilizada foi recriada virtualmente para o estudo de caso no capítulo 4, esta célula foi utilizada e estudada, por já existir e não se fazer necessária a elaboração e estruturação de uma nova célula, já que a proposta deste trabalho é justamente utilizar de dispositivos disponíveis.

A partir das simulações feitas no software RS, cria-se programas para o robô que podem ser passados para ele e executados na célula real, programas estes que se encontram em anexo, com isso foram feitos testes primordiais para que se atingisse um grau de certeza ao utilizar o programa complexo para que não causasse nenhum dano nos dispositivos empregados.

Uma vez testado os programas foram inseridos no robô e executados. O funcionamento da célula robótica se torna mais fácil e preciso quando programada em método off-line já que os programas são criados a partir de formas geométricas, o que facilita a criação de formas complexas como trajetos curvos e longos, bem como evitar possíveis acidentes entre as partes envolvidas.

Para o funcionamento desta célula, depois de feito a pesquisa apresentada nos capítulos anteriores, foi determinado uma série de ações que a célula iria empregar. Para isso se cria a logica de sequenciamento que é apresentado aqui.

5.2.1. Descritivo Robô

Primeiramente deve-se ressaltar que o controlador do robô que se comunica com todas as partes, pois é ele que dirá se deve ou não prosseguir com as tarefas.

O Robô permanece em sua posição inicial (*HOME*) até que algum sinal indique que deve iniciar o processo, a primeira etapa é a de escolha ou a determinação de com qual objeto ele deve interagir. Cada peça exige um método de interação, que será predeterminado pela programação. Logo em seguida o robô trabalhara a peça e a assim que terminar sua tarefa enviará um sinal para que a Plataforma faça sua parte, o robô realizara a sequencia a seguir:

- Espera rotina (sensor)
- Ir para posição próxima à peça
- Determina como trabalhar a peça
- Avança rapidamente até próximo ao objeto

- Avança lentamente até encostar a ferramenta
- Realiza a tarefa
- Afasta lentamente a ferramenta
- Afasta rapidamente voltando a posição inicial

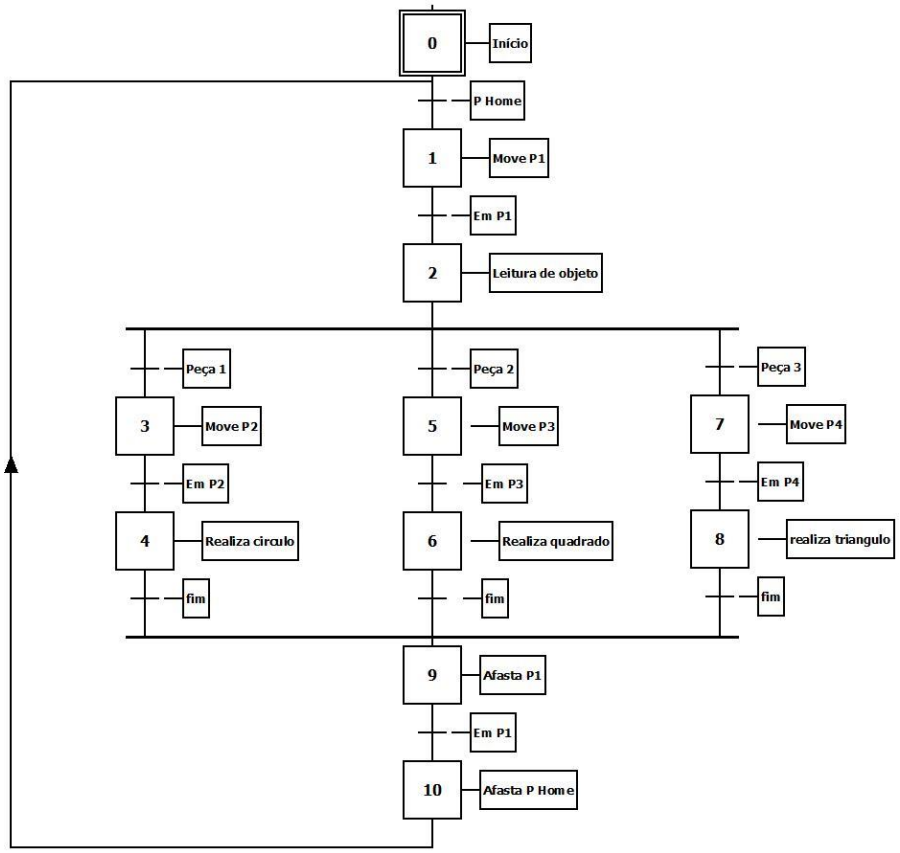


Figura 5.2 – G7 Funcional do Robô

Como se pode perceber existe a necessidade de várias etapas para se realizar uma única tarefa. Para se criar o primeiro ambiente favorável a estas tarefas, se faz o diagrama G7 do robô, seguindo as descrições.

5.2.2. Descritivo da Plataforma de Posicionamento

A tarefa proposta para a plataforma robótica é mais simples, porém não menos importante. Esta ficaria a cabo de realizar a retirada da peça do local do robô para outro local que seria o local de trabalho, sendo assim a Plataforma apresenta dois pontos, o local de carregamento e o local de trabalho, ambos são alcançados através do acionamento do cilindro hidráulico da base da PRP.

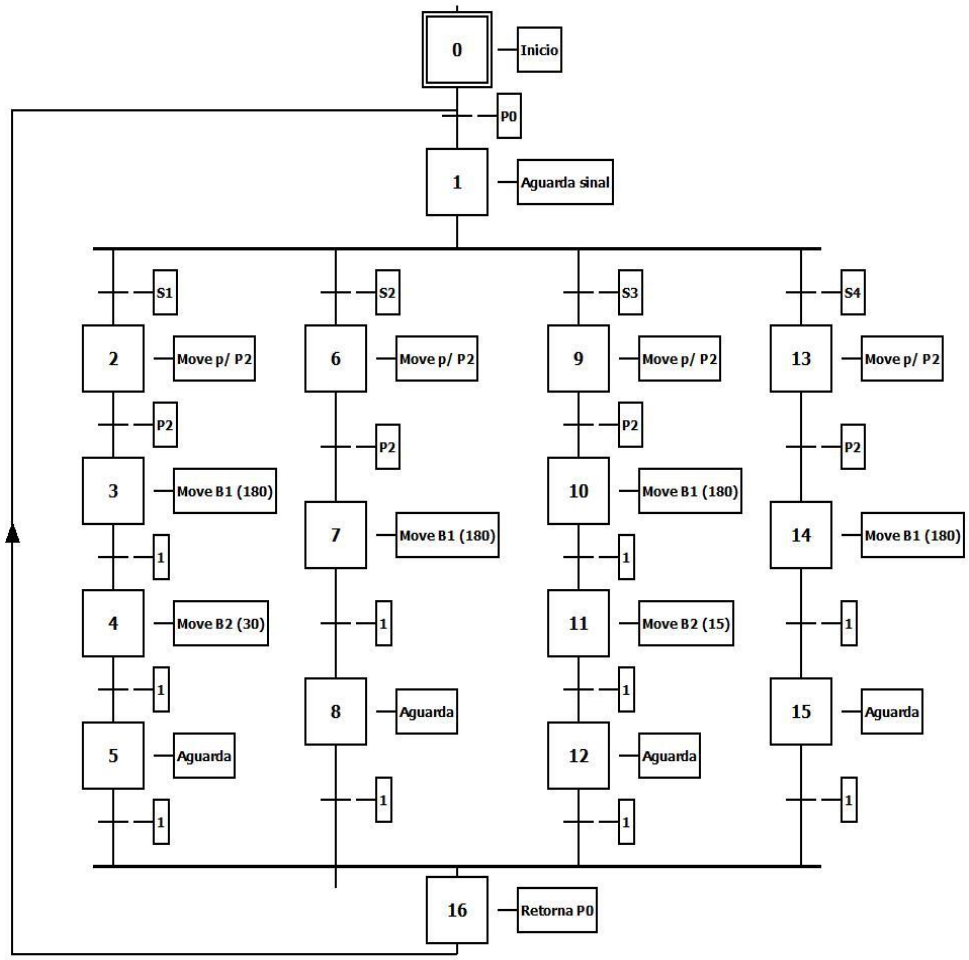


Figura 5.3 – G7 Funcional da MP

Para realizar a tarefa de posicionamento do objeto para trabalho a Plataforma Robótica fará as tarefas de acordo com as descrições abaixo:

- Aguarda um sinal de que a peça esta em posição
- Recua cilindro hidráulico
- Gira a 180º a base total da PRP
- Gira 30º a base de peças
- Aguarda sinal de termino de trabalho
- Ao termino do trabalho a PRP retorna a posição inicial

5.2.3. Integração com GRAFCET

Para dar o devido andamento ao trabalho deve-se fazer a integração dos dispositivos, e para isso se utiliza de ferramentas gráficas para facilitar a visualização e o entendimento da problemática.

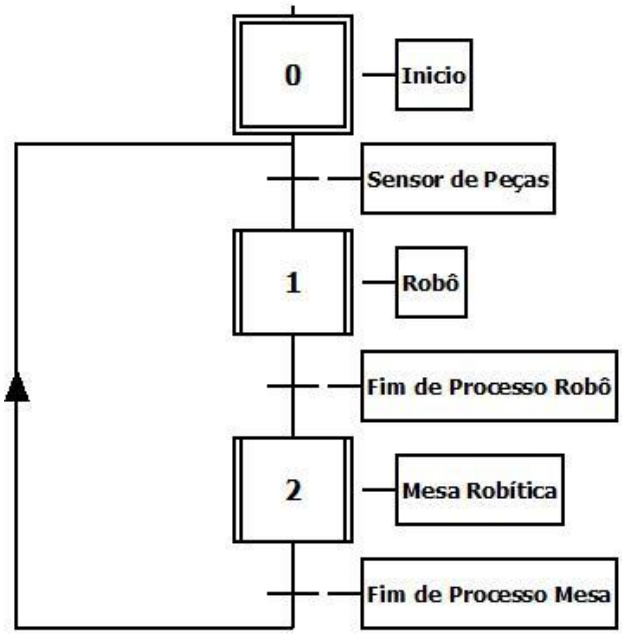


Figura 5.4 - Integração dos Diagramas

Usando-se das descrições dos dispositivos demonstrados logo acima, podemos criar diagramas para cada dispositivo separadamente e uni-los em um único diagrama fazendo assim a integração. Para que estes funcionem em conjunto.

Utiliza-se os blocos de chamada de programa para facilitar a visualização do diagrama, pois se utilizasse todo o diagrama para integrar, este se tornaria relativamente extenso e de difícil entendimento.

5.3. Programação e simulação em RobotStudio™

Com este trabalho foi percebido que os trabalhos anteriores realizados no LAIR, são satisfatórios no âmbito de suas propostas para a realização de uma célula robótica, porém em nenhum visava a programação off-line como meio de agilizar a integração dos robôs industriais neste processo, embora alguns mostravam benefícios deste método de programação.

Dito isto, neste trabalho acarretou numa maior ênfase neste tipo de programação, assim desenvolvendo simulações para o robô industrial, levando a resultados que serão apresentados aqui.

Para o intuito de teste para a funcionalidade do programa e aprendizado da ferramenta empregada, foi feito vários objetos de trabalho para estudo em ambiente virtual e como mostra a figura 5.6 relativa as elipses que foram testadas no robô.

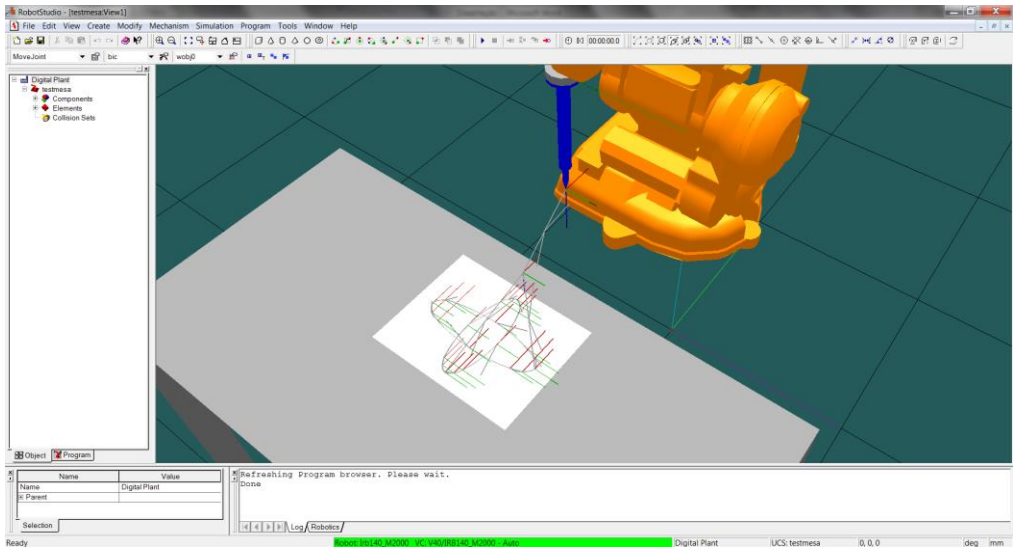


Figura 5.5 - Elipses de teste

```
MODULE ELIPSES
  PERS tooldata Bic:=[TRUE,[[0,0,220],[1,0,0,0]],[0.3,[0,0,100],[1,0,0,0],0,0,0]];
  CONST robtarget init2:=[[0,-
450,427],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget Target50:=[[5,-
570,342],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget Target7:=[[-100,-
675,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget Target6:=[[-88.0621,-
659.407,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target13:=[[110,-
675,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget Target14:=[[4.99957,-
760,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget Target16:=[[23.7045,-
741.456,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target15:=[[16.5956,-
752.991,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target18:=[[23.705,-
608.544,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target17:=[[35.3585,-
675,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget Target20:=[[5.00031,-
590,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget Target19:=[[16.5962,-
597.009,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target22:=[[-13.7045,-
608.544,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
```

```
  CONST robtarget Target21:=[[-6.59564,-
597.009,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target24:=[[-13.705,-
741.456,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target23:=[[-25.3585,-
675,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget Target26:=[[4.99957,-
760,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget Target25:=[[-6.59619,-
752.991,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target11:=[[79.246,-
696.213,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target2:=[[98.0624,-
659.408,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target5:=[[-69.2462,-
653.786,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target4:=[[5.00012,-
644.784,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target3:=[[79.2463,-
653.787,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target8:=[[-88.0623,-
690.592,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target9:=[[-69.2464,-
696.213,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target10:=[[4.99988,-
705.216,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];
  CONST robtarget Target1:=[[110,-
675,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget Target12:=[[98.0623,-
690.593,292],[0,0.707107,0.707107,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];

  PROC main()
    Path1;
  ENDPROC

  PROC Path3()
    MoveJ init2,v1000,fine,Bic\WObj:=wobj0;
    MoveJ Target50,v1000,fine,Bic\WObj:=wobj0;
    MoveL Target1,v1000,fine,Bic\WObj:=wobj0;
    MoveC Target2,Target3,v200,fine,Bic\WObj:=wobj0;
    MoveC Target4,Target5,v200,fine,Bic\WObj:=wobj0;
    MoveC Target6,Target7,v200,fine,Bic\WObj:=wobj0;
    MoveC Target8,Target9,v200,fine,Bic\WObj:=wobj0;
    MoveC Target10,Target11,v200,fine,Bic\WObj:=wobj0;
    MoveC Target12,Target13,v200,fine,Bic\WObj:=wobj0;
    MoveJ Target50,v1000,fine,Bic\WObj:=wobj0;
    MoveL Target14,v1000,fine,Bic\WObj:=wobj0;
```

```
    MoveC Target15,Target16,v1000,fine,Bic\WObj:=wobj0;
    MoveC Target17,Target18,v1000,fine,Bic\WObj:=wobj0;
    MoveC Target19,Target20,v1000,fine,Bic\WObj:=wobj0;
    MoveC Target21,Target22,v1000,fine,Bic\WObj:=wobj0;
    MoveC Target23,Target24,v1000,fine,Bic\WObj:=wobj0;
    MoveC Target25,Target26,v1000,fine,Bic\WObj:=wobj0;
    MoveJ Target50,v1000,fine,Bic\WObj:=wobj0;
    MoveJ init2,v1000,fine,Bic\WObj:=wobj0;
  ENDPROC
ENDMODULE
```

Já na figura 5.7 o robô já realiza trabalhos sobre a plataforma que foi devidamente posicionada e neste programa foi feito a funcionalidade na Plataforma da célula robótica estudada, para isso foi criado um circulo no objeto a ser trabalhado onde o robô deveria segui-lo.

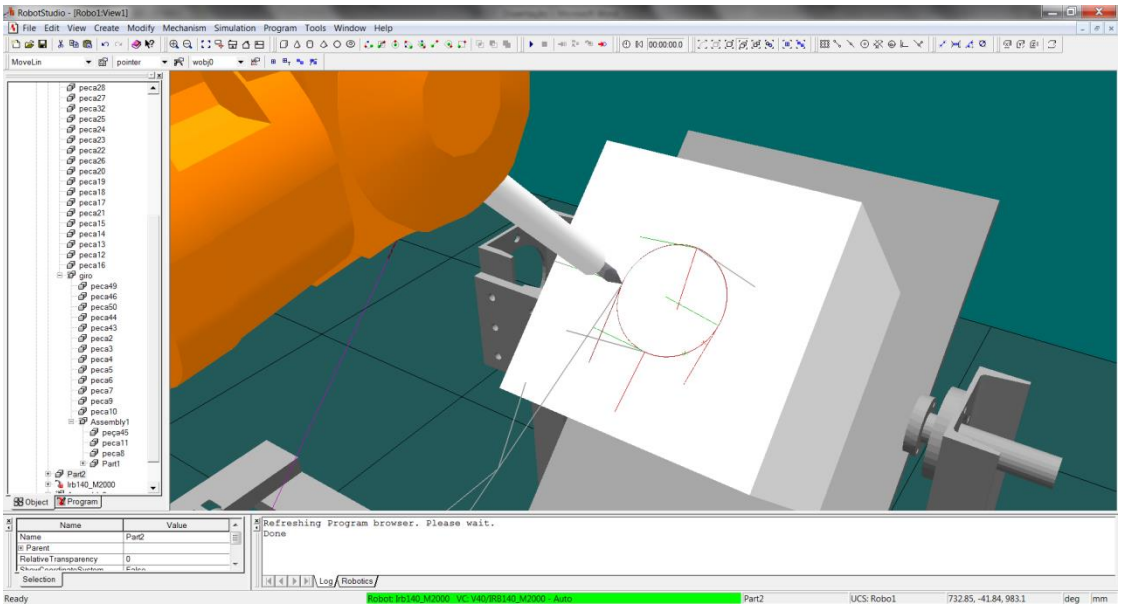


Figura 5.6 - Circulo de Trajetória

```
MODULE Circulo
  PERS tooldata
pointer:=[TRUE,[[0,0,220],[1,0,0,0]],[0.3,[0,0,100],[1,0,0,0],0,0,0]];
  CONST robtarget
init:=[[450,0,426.99],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget
inter:=[[550,0,580],[0.25882,0,0.965925,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+0
9,9E+09]];
  CONST robtarget
init1:=[[450,0,427],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
  PROC teste()
    MoveL init1,v500,fine,pointer\WObj:=wobj0;
    MoveL Target1,v200,fine,pointer\WObj:=wobj0;
    MoveC Target2,Target3,v200,fine,pointer\WObj:=wobj0;
    MoveC Target4,Target5,v200,fine,pointer\WObj:=wobj0;
    MoveL init1,v500,fine,pointer\WObj:=wobj0;
  ENDPROC
ENDMODULE
```

Como se pode perceber, o programa realizado pelo software RobotStudio é bem enxuto, de simples visualização e modificação, observa-se na primeira parte que o software gera constantes internas onde são armazenadas as posições reais dos pontos de trabalho e a calibração da ferramenta de trabalho que é denominada de *tooldata*.

5.4. Considerações Finais

Este capítulo demonstra a integração a partir do G7 e os resultados da programação off-line além de mostrar a célula robótica real que se encontra disponível no laboratório (LAIR) onde foi realizada toda a pesquisa e elaboração da metodologia. Sendo uma célula formada por dispositivos que estavam disponíveis para pesquisa, pode-se atingir um dos objetivos deste trabalho que era justamente, utilização e otimização de equipamentos disponíveis.

CAPÍTULO 6

6. Conclusões e Perspectivas Futuras

Esta dissertação teve como objetivo, demonstrar métodos de implementação de células robotizadas flexíveis, utilizando de dispositivos e ferramentas existentes no mercado atual.

No decorrer do trabalho pode-se perceber a importância da modelagem virtual e simulação dos dispositivos da célula empregada, como este trabalho utiliza uma célula já existente e real onde foi colocado em funcionamento o robô industrial, pode-se concluir que a programação off-line de robôs traz uma grande facilidade e agilidade no processo de implantação destes dispositivos em meios industriais.

Com a utilização dos outros trabalhos anteriores realizados no laboratório, pode-se concluir que a utilização desta metodologia é válida pela facilidade que traria à implementação. Nas indústrias, onde o tempo é de grande importância, não se pode demorar em realizar tais tipos de procedimentos, assim, como o método apresentado neste trabalho visa a modelagem virtual anterior a concepção real, não se gasta tempo de produção fabril para a implementação.

Este método foi pensado em um nível geral podendo ser utilizado para células independendo de sua complexidade e tamanho.

Pelas visões aqui adquiridas pode-se chegar a conclusão que apesar de aparentemente simples, a quantidade de conhecimento empregado na elaboração de uma célula e operação da mesma, é relativamente grande. Sendo que se faz necessária a formação de um profissional qualificado em um curso específico para tal. Assim como para a operação desta célula também se deve a um profissional qualificado para isto. Devendo-se dar enfoque a um curso de robótica ou operações em automação que visasse formar tais profissionais.

Finalmente, este trabalho permitiu o aprofundamento de conhecimentos no domínio do projeto e concepção de sistemas automatizados de produção, bem como o desenvolvimento de trabalhos futuros direcionados a:

a) Elaboração e implementação de cursos de graduação e formação tecnológica que visa a formação de profissionais qualificados para a projeto e implementação de células robotizadas;
b) Diminuir os altos custos de desenvolvimento de projeto de células robóticas flexíveis.
c) Diminuição do tempo de startup
d) Redução de custos de componentes de implementação e tempo de desenvolvimento de software;
e) Implementação de sistemas de supervisórios e redes de comunicacão em um nível mais amigável, abrangendo tanto a programação quanto métodos de integração e manutenção de dispositivos, para que possa fazer com que a célula se mantenha em funcionamento pleno todo o tempo de trabalho.

7. Bibliografia

AIHARA, Cintia Kimie, Projeto e Implementação de Plataforma Didática para Ensino e Pesquisa de Sistemas Integrados Automatizados, Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2000. 104 p. Dissertação (Mestrado)

BOTAZZI, Vitor Santos, Framework para Programação Offline de Robôs, Escola de Engenharia, Universidade do Minho, Braga, 2006. 77p. Dissertação (Mestrado)

CASSEMIRO, Edna Rodrigues, Proposta de um ambiente virtual de simulação para projeto de controlador preditivo robusto para juntas robóticas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2006. 155p. Dissertação (Mestrado)

CASTILLO ESTEPA, Ricardo Andrés, Proposta de arquitetura de supervisão e controle para uma plataforma automatizada (WebLab) orientada à formação e pesquisa em automação e robótica, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2010. 180p. Dissertação (Mestrado)

COSSO, Sérgio Guanaes, Implementação de Ferramentas de Automação Direcionadas a Aplicação de Telerobótica – Implementação de um Sistema de Supervisão e Controle num Sistema Teleoperado, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2002. 134 p. Dissertação (Mestrado)

EasyRob: Informações sobre software. Disponível em: http://www.easy-rob.com/en/easy-rob/. Acesso no segundo período de 2013.

FANUC: Roboguide simulation software. Disponível em: http://www.fanucrobotics.com/Products/vision-software/ROBOGUIDE-simulation-software.aspx. Acesso no segundo período de 2013.

FRANCÊS, Carlos Renato Lisboa, Introdução as Redes de Petri, Laboratório de Comunicação Aplicada – LACA, Universidade Federal do Pará, Agosto de 2003.

FREITAS, Julio Cesar de Almeida, Concepção, Projeto e implementação de Células Automatizadas Utilizando Conceitos de Programação Off-Line de Robôs, Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2004. 151 p. Dissertação (Mestrado)

Industrial Comunication. Disponível em: http://www.advantech.com.br/eAutomation/Industrial_Communication/. Acesso no primeiro período de 2013.

KUKA ROBOTICS: Application software. Disponível em: http://www.kuka-robotics.com/br/products/software/application_software/. Acesso no segundo período de 2013.

LabView Student: O que é LabView. Disponível em: http://www.ieee.org/membership_services/membership/students/NI_offer.html. Acesso no segundo período de 2013.

LANGRAFE, Eduardo; FERRAZ, Cristiano H.; LOPES, Eduardo V., Ethernet como Solução Confiável para o Segmento de Utilities, Seminário Nacional de Produção e Transmissão de Energia Elétrica, Florianópolis, 2011. v-1. 8 p. Artigo

National Instrumensts: Informações sobre LabView. Disponível em: http://www.ni.com/labview/pt/. Acesso no segundo período de 2013.

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

ROJAS, Jaime Humberto Carvajal, Metodologia de Modelagem, Simulação e Programação Off-Line de Robôs e Mecanismos Mecatrônicos Integrados e Direcionados à Células Flexíveis de Manufatura, Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2004. 168 p. Tese (Doutorado)

SIMSOL: robotic simulation ROBCAD. Disponível em: http://www.simsol.co.uk/robotic_simulation_robcad.php. Acesso no segundo período de 2013.

URIBE QUEVEDO, Alvaro Joffre, Manipulação de objetos em ambiente virtual utilizando uma garra antropomórfica acoplada a um robô industrial, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2008. 115p. Dissertação (Mestrado)

ANEXO

1. Funcionamento do G7

1.1.Etapa

As etapas são subdivididas em etapas iniciais, etapas intermediárias e etapas globais.

As etapas iniciais como o próprio nome já denota, são etapas que iniciam o diagrama, são representadas através de um quadrado ou retângulo duplo.

As etapas regulares são aquelas que compõem o diagrama, são elas que formam o corpo logico e determinam as ações que serão realizadas.

Já as etapas de chamada são menos usadas, mas não menos importantes, são utilizadas para facilitar a visualização de diagramas mais complexos ou muito extensos, estas etapas são formadas por sub diagramas.

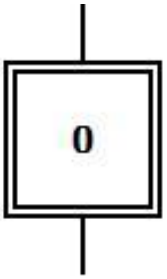


Figura 1.1 - Etapa Inicial

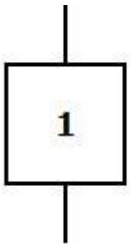


Figura 1.2 - Etapa Regular

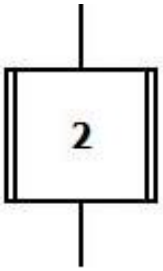


Figura 1.3 - Etapas de Chamada

Deve-se lembrar que uma etapa nunca pode vir seguida de outra, bem como uma transição também nunca deve vir seguida de outra.

1.2. Transição

A transição é a possibilidade de evolução de uma etapa para outra, é representada graficamente por uma linha cortada por um traço. Esta função lógica simboliza um ou mais elementos do dispositivo, como, sensores, botões, leitores, ou qualquer elemento que emita um sinal de confirmação para o sistema.

Figura 1.4 - Representação Gráfica de Transição

1.3. Ligações orientadas

As etapas e transições são ligadas através de ligações orientadas, estas geralmente tem o sentido de cima para baixo e quando forem invertidas devem ser acompanhadas de setas que indicarão a sua orientação.

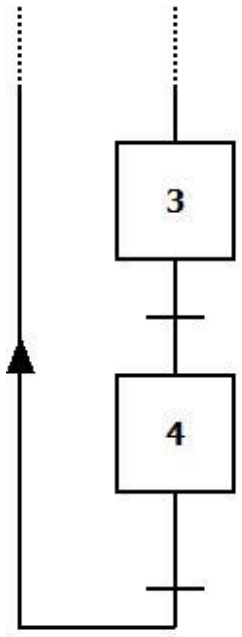


Figura 1.5 - Exemplo de Ligações Orientadas

Cruzamentos entre ligações devem ser evitados, o mais comum é, quando necessário, interromper o diagrama e continua-lo em outra página, ou ainda utilizar uma etapa de chamada que contem outra parte do diagrama em um bloco separado.

Os diagramas podem ser sequenciais, que são aqueles que apresentam ligações simples, são comumente utilizados em lógicas básicas, onde a sequencia de ações deve ser seguida a risca sem intervenções externas ou dispensam escolhas feitas automática ou manualmente.

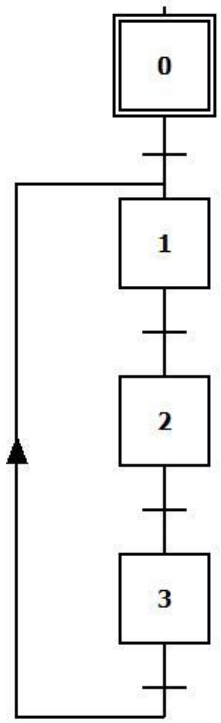


Figura 1.6 - Exemplo de GRAFCET Sequencial

Também podem ser ramificados, onde as ligações são dispostas estrategicamente para se evitar o cruzamento de linhas, para que não se complique a visualização.

Um diagrama ramificado pode ser divergente em E ou divergente em OU.

Divergente em E ou sequencia simultânea, são iniciadas paralelamente, ou seja, com uma única transição todas as ações são realizadas simultaneamente.

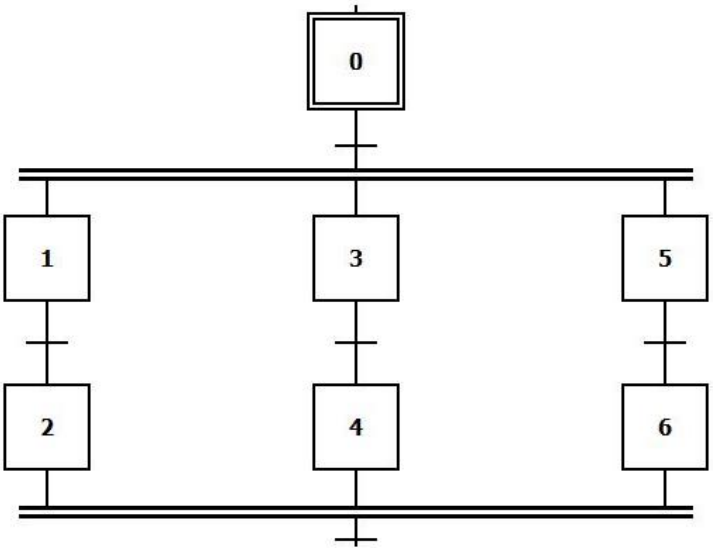


Figura 1.7 - Exemplo de Divergente em E

Divergente em OU, ou sequencia seletiva, são diagramas onde as ações são iniciadas separadamente a partir de uma escolha feita pelo sistema ou manualmente, são lógicas com uma complexidade mais elevada, requerendo mais tempo para seu desenvolvimento. Em um diagrama deste modo, o sistema só poderá avançar para próxima sequencia quando a primeira for concluída e assim sucessivamente.

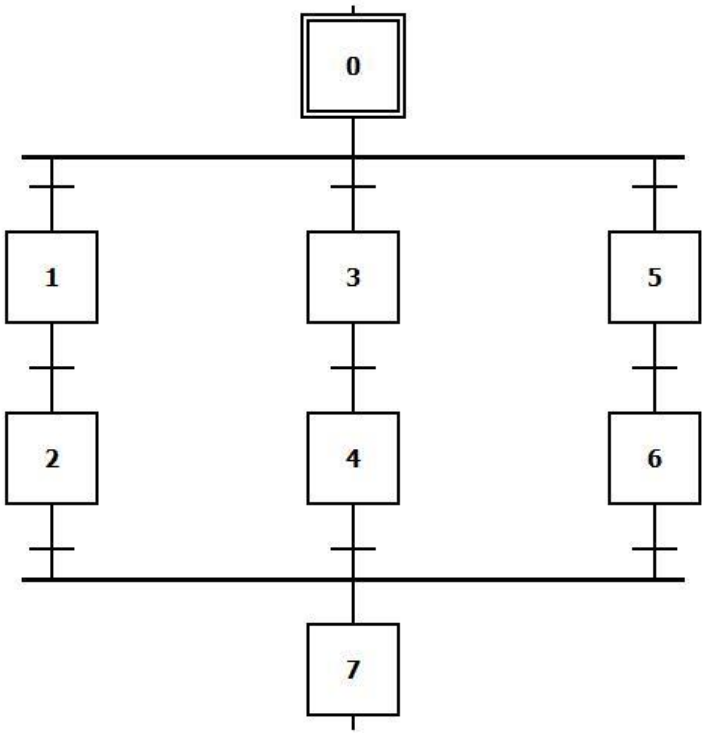


Figura 1.8 - Exemplo Divergente em OU

2. Redes de Petri (Introdução a redes de petri - Lisboa Francês, C R)

Rede de Petri é uma técnica de modelagem que permite a representação de sistemas, utilizando como alicerce uma forte base matemática [MAC96]. Essa técnica possui a particularidade de permitir modelar sistemas paralelos, concorrentes, assíncronos e não-determiníticos [VAL80].

A representação gráfica de uma rede de Petri básica é formada por dois componentes: um ativo chamado de transição (barra) e outro passivo denominado lugar (círculo). Os lugares equivalem às variáveis de estado e as transições correspondem às ações realizadas pelo sistema [MAC96]. Esses dois componentes são ligados entre si através de arcos dirigidos. Os arcos podem ser únicos ou múltiplos. A figura 2.1 mostra os elementos básicos de um grafo associado às redes de Petri.

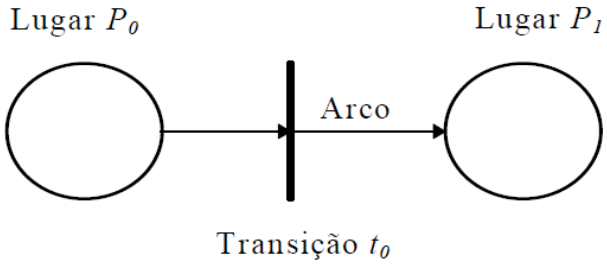


Figura 2.1 - Grafo e seus Elementos Básicos

2.1.Definições

As redes de Petri podem ser enfocadas através de três fundamentações diferentes. A primeira utiliza a teoria bag como suporte. A segunda usa os conceitos da álgebra matricial. A última se fundamenta na estrutura definida por relações. A seguir são apresentadas as definições formais de cada uma dessas fundamentações.

2.1.1. Fundamentações para Redes de Petri

- **Definição 1:** uma rede de Petri R é uma quíntupla R = (P, T, I, O, K), onde P = {p1, p2,...,pn} é um conjunto finito não-vazio de lugares, T = {t1, t2,..., tm} é um conjunto finito não-vazio de transições. I : T → P é um conjunto de bags 1 que representa o mapeamento de transições para lugares de entrada. O : T → P é um conjunto de bags que representa o mapeamento de transições para lugares de saída. K : P → N é o conjunto da capacidades associadas a cada lugar, podendo assumir um valor infinito [PET81].

Para exemplificar a definição 1, supõe-se que se deseje representar um ano letivo de uma Universidade. O ano letivo começa com o primeiro período (semestre) letivo, seguido das

primeiras férias (de julho), logo após, tem-se o segundo período letivo, e finalmente as férias de final de ano. Assim, o ano letivo poderia ser representado conforme a figura 2.2.

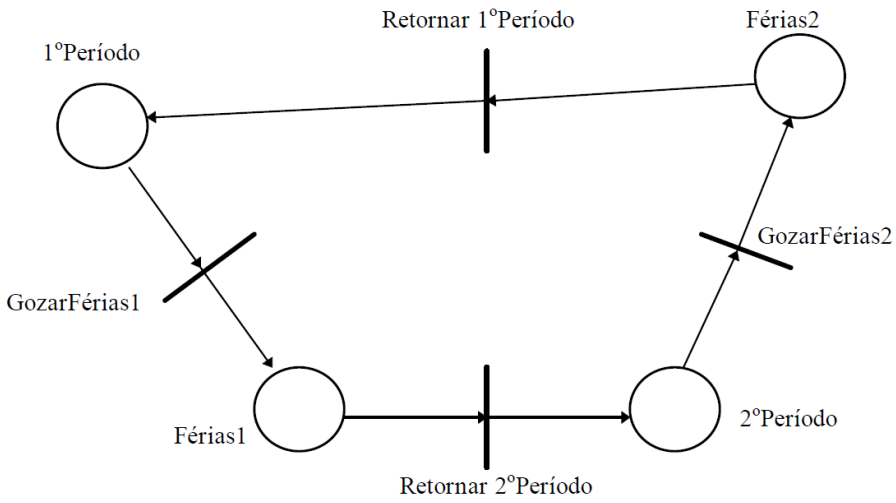


Figura 2.2 - Ano letivo representado graficamente em rede de Petri

A figura 2.2 pode ser descrita da seguinte forma, utilizando-se a definição 1:

RAno_Letivo = (P, T, I, O, K), onde o conjunto de lugares P é:

P = {1oPeríodo, Férias1, 2oPeríodo, Férias2};

o conjunto de transições T é:

T = {GozarFérias1, Retornar2oPeríodo, GozarFérias2, Retornar1oPeríodo};

o conjunto de bags de entrada I é:

I = { I (GozarFérias1) = [1oPeríodo], I (Retornar2oPeríodo) = [Férias1], I (GozarFérias2) = [2oPeríodo], I (Retornar1oPeríodo) = [Férias2] };

o conjunto de bags de saída O é:

O = {O (GozarFérias1) = [Férias1], O (Retornar2oPeríodo) = [2oPeríodo], O (GozarFérias2) = [Férias2], O (Retornar1oPeríodo) = [1oPeríodo] };

e o conjunto de capacidades dos lugares é:

K = { K1oPeríodo = 1, K Férias1 = 1, K2oPeríodo = 1, K Férias2 = 1}.

- Definição 2: a estrutura de uma rede de Petri, segundo o ponto de vista matricial, é uma quíntupla R = (P, T, I, O, K), onde P é um conjunto finito de lugares. T é um conjunto finito de transições, I : P x T → N é a matriz de pré-condições. O : P x T → N é a matriz de pós-condições. K é o vetor das capacidades associados aos lugares (K : P → N) [PET81].

Tomando-se como base novamente a figura 2.2, tem-se:

Os conjuntos de lugares e transições são idênticos àqueles vistos para a definição1. A matriz I (pré-condições) é:

I =

GozarFérias1	Retornar2°Período	GozarFérias2	Retornar1°Período	
1	0	0	0	1°Período
0	1	0	0	Férias1
0	0	1	0	2°Período
0	0	0	1	Férias2

A matriz O (pós-condições) é:

O =

GozarFérias1	Retornar2°Período	GozarFérias2	Retornar1°Período	
0	0	0	1	1°Período
1	0	0	0	Férias1
0	1	0	0	2°Período
0	0	1	0	Férias2

É importante ressaltar que as matrizes I e O representam as pré e pós-condições, respectivamente, de todas as transições da rede.

- Definição 3: a estrutura de redes de Petri, usando-se relações, é formada por uma quíntupla R = (P, T, A, V, K), onde P é o conjunto de lugares, T o de transições, A o conjunto dos arcos e V corresponde ao conjunto de valorações desses arcos. Os elementos de A são arcos

que conectam transições a lugares ou lugares a transições ($A \subseteq (P \times T) \cup (T \times P)$). Assim, os elementos de A podem ser agrupados em dois subconjuntos - o conjunto das entradas às transições e o de saída às transições, I = {(pi, tj)} e O = {(tj, pi)}, respectivamente [MUR89].

Tomando-se ainda como referência a figura 2.2, tem-se que os conjuntos de lugares (P), de transições (T) e de capacidades (K) permanecem inalterados. Entretanto, na notação que utiliza relações, há o surgimento de dois novos conjuntos: o conjunto de arcos (A) e o conjunto de valores para esses arcos (V).

o conjunto de arcos A é:

A = {(1oPeríodo, GozarFérias1), (GozarFérias1, Férias1), (Férias1, Retornar2oPeríodo), (Retornar2oPeríodo, 2oPeríodo), (2oPeríodo, GozarFérias2), (GozarFérias2, Férias2), (Férias2, Retornar1oPeríodo), (Retornar1oPeríodo, 1oPeríodo)}

o conjunto de valores dos arcos V é:

V = {1, 1, 1, 1, 1, 1, 1, 1}

2.1.2. Redes de Petri Marcadas

Marcas (tokens) são informações atribuídas aos lugares, para representar a situação (estado) da rede em um determinado momento. Define-se uma rede de Petri marcada pela dupla RM = (R, Mo), onde R é a estrutura da rede e Mo a marcação inicial [MAC96].

Assim, para simular o comportamento dinâmico dos sistemas, a marcação da rede de Petri é modificada a cada ação realizada (transição disparada). A figura 2.2 [MAC96] ilustra uma rede marcada.

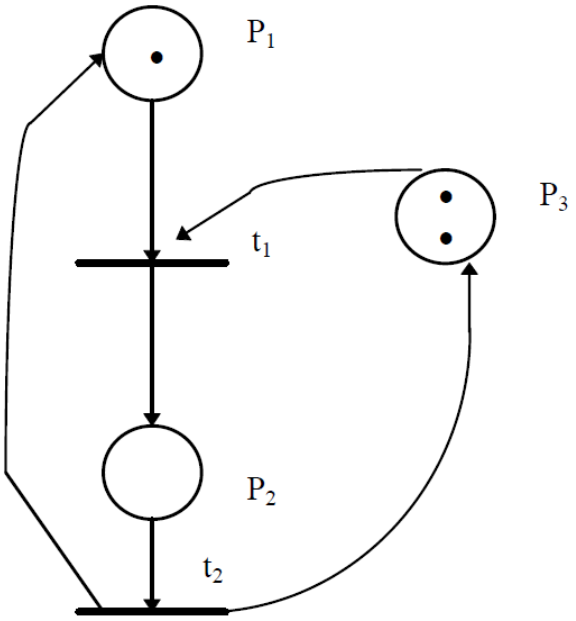


Figura 2.3 - Rede Marcada

2.1.3. Notações Particulares

Em alguns casos, deseja-se representar a diferença entre transições, visando melhorar a clareza do modelo. Além disso, em muitas situações, pretende-se representar a execução de uma condição externa ao sistema modelado. Para representar rótulos de transições, utiliza-se um alfabeto qualquer associado à rede (por exemplo, o alfabeto a,b,c,....,z). Para representar as condições externas, usa-se o mesmo esquema utilizado para rotular transições, entretanto, os símbolos vêm entre parênteses (conforme ilustra a figura 2.4).

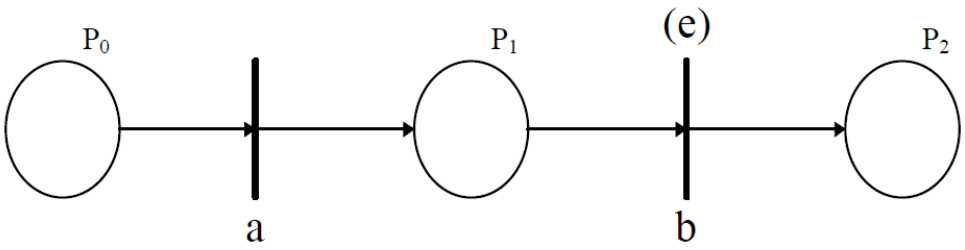


Figura 2.4 - Rótilos e condições externas as transições

2.2. Classes das Redes de Petri

Podem-se agrupar as redes de Petri em duas grandes classes: as Ordinárias e Não-Ordinárias (de Alto nível) [MAC96]. As redes ordinárias se caracterizam pelo tipo de suas marcas, ou seja, suas marcas são do tipo inteiro e não negativo, enquanto que as de alto nível possuem marcas de tipos particulares. As redes ordinárias se subdividem em:

- Rede Binária: é a rede mais elementar dentre todas. Essa rede só permite no máximo um token em cada lugar, e todos os arcos possuem valor unitário.

- Rede Place-Transition: é o tipo de rede que permite o acúmulo de marcas no mesmo lugar, assim como valores não unitários para os arcos.

As redes de alto nível são caracterizadas pelos tipos de suas marcas, que não são mais elementos do tipo inteiro positivo. Esse tipo de rede permite a individualização de uma marca (pertencente a um grupo) em um mesmo lugar. Essa individualização pode ser realizada através de vários artifícios, como por exemplo, cor da marca ou objetos representando os tokens. Redes não-ordinárias não aumentam o poder de representação de um modelo. Entretanto, elas permitem uma maior clareza e um maior (ou menor) nível de abstração ao modelo.

2.2.1. Redes Elementares

Nesta seção, são apresentadas algumas redes que, a partir delas, derivam muitas outras redes mais complexas. São discutidas as redes representativas de seqüenciamento, distribuição, junção, escolha não-determinística e atribuição.

- Seqüenciamento: é a rede que representa a execução de uma ação, desde que uma determinada condição seja satisfeita. Após a execução dessa ação, pode-se ter outra ação, desde que satisfeita outra determinada condição (figura 2.5).

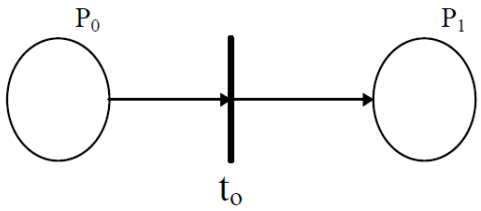


Figura 2.5 - Sequenciamento

- Distribuição: é a rede elementar utilizada na criação de processos paralelos a partir de um processo pai. Os processos filhos são criados através da distribuição dos tokens encontrados no processo (lugar) pai. A distribuição é mostrada na figura 2.5.

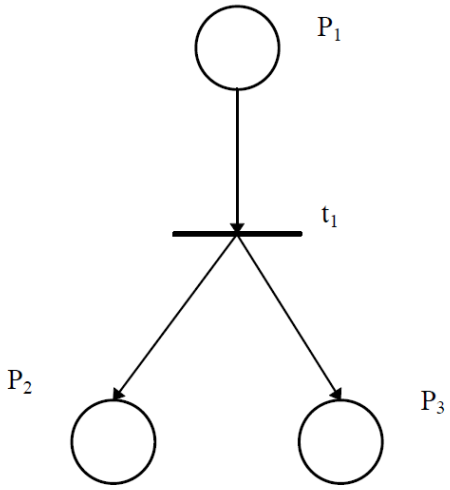


Figura 2.6 – Distribuição

- Junção: é a rede que modela a sincronização entre atividades concorrentes. No exemplo da figura 2.6, a transição t1 só dispara quando existirem fichas tanto em P1, quanto em P2, estabelecendo, assim, o sincronismo.

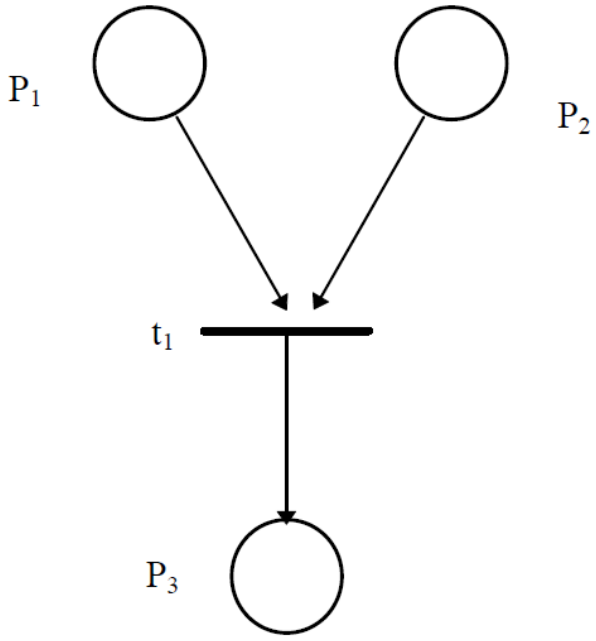


Figura 2.7 – Junção

- Escolha Não-Determinística: é uma rede que ao se disparar uma transição, inabilita-se a outra. Entretanto, não existe possibilidade de escolha (conforme figura 2.7). O fator não-determinístico dessa rede gera uma situação chamada de conflito [MAC96]. O conflito pode ser classificado como estrutural ou efetivo. Ambos os conflitos estão associados ao fato de duas transições possuírem o mesmo lugar como entrada. Porém, se a rede não possuir tokens, o conflito é dito estrutural. Contudo, se há uma única marca no lugar comum às transições, diz-se que o conflito é efetivo. A figura 2.8 [MAC96] ilustra os dois tipos de conflito.

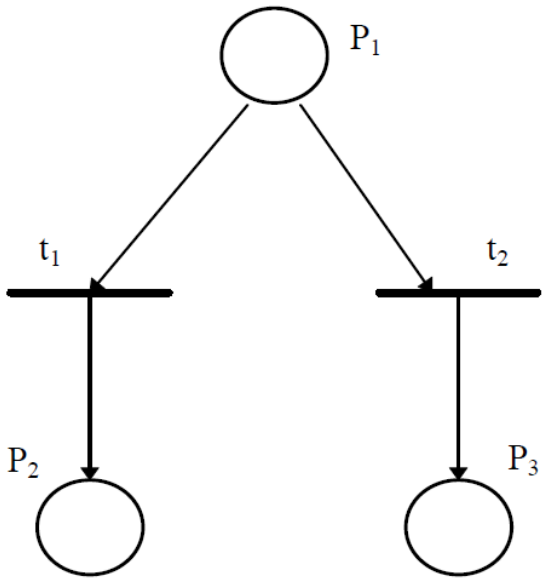


Figura 2.8 - Escolha Não-Determinística

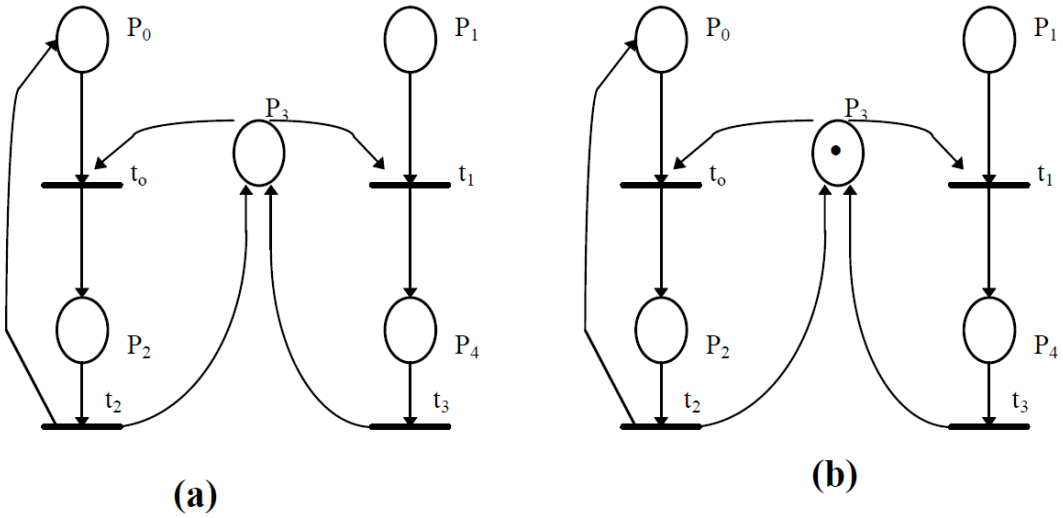


Figura 2.9 - (a) Conflito Estrutural e (b) Conflito Efetivo

A escolha determinística da transição a ser disparada não é um recurso abordado nas redes elementares. Porém, essa deficiência das redes de Petri originais são resolvidas em algumas extensões propostas, abordadas em seções posteriores.

2.3. Redes de Petri em Protocolos de Comunicação

Uma das áreas mais interessantes para aplicação de modelagem (especialmente redes de Petri) é a área da representação de protocolos de comunicação. Em protocolos, geralmente, as transições são bem nítidas (por exemplo, a transmissão ou recepção de uma mensagem). Muitas situações que são mutuamente exclusivas se fazem presentes (como a escolha entre um receptor dentre vários). A figura 2.1 [MAC96] mostra o comportamento de um protocolo bastante simples. Apesar de simples, o modelo apresenta situações interessantes para discussão.

O funcionamento do protocolo da figura 2.9 se fundamenta basicamente na decisão de qual receptor (1 ou 2) deve aceitar a mensagem. Nesse exemplo específico, a escolha é não-determinística, isto é, não se pode decidir qual o arco que o token deve seguir. Além disso, a

escolha é mutuamente exclusiva, ou seja, apenas uma das transições será habilitada (t2 ou t4). A escolha citada se processa quando o token, que vem de t0, chega ao lugar P6.

Nesse ponto, o token ou parte para o Receptor1 (habilitando a transição t2), ou para o Receptor2 (habilitando a transição t4). A partir daí, após a escolha do receptor, o token segue para um buffer (P3 no Receptor 1, ou P4 no Receptor 2), para ser reconhecido pelo disparo da transição t3 (Receptor 1) ou t5 (Receptor 2).

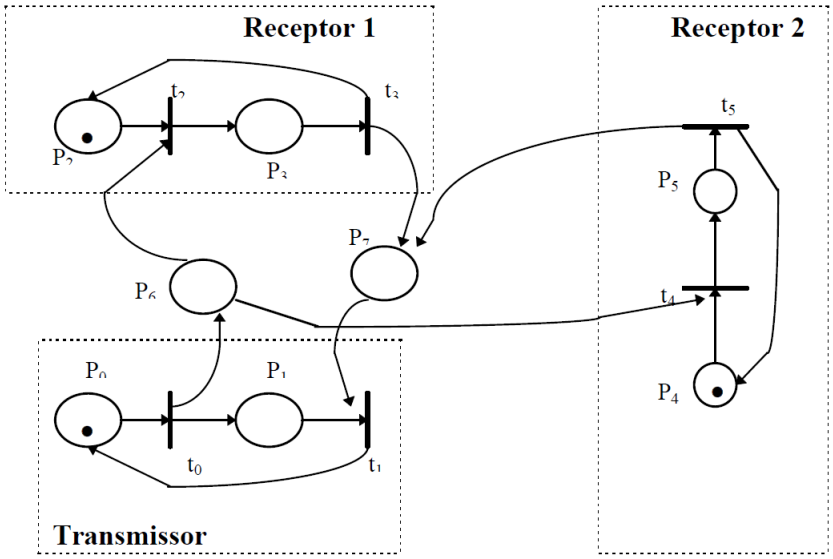


Figura 2.10 - Protocolo de comunicação

O modelo é interessante para representar um protocolo que trabalhe de maneira aleatória. Entretanto, nem sempre é interessante se valer dessa situação (a eventualidade).

Para os casos em que se deseja estipular o destino que o token deve seguir, ou seja, trabalhar de forma determinística, foram propostas algumas extensões às redes de Petri originais. Essas extensões são discutidas na próxima seção.

2.4. Extensões às Redes de Petri

Nesta seção são vistas algumas extensões propostas com a finalidade de aumentar a aplicação das redes de Petri. São discutidas as redes de Petri coloridas, hierárquica e temporizadas determinísticas.

2.4.1. Redes de Petri Coloridas

As redes de Petri coloridas têm por objetivo reduzir o tamanho do modelo, permitindo que os tokens sejam individualizados, através de cores atribuídas a eles; assim, diferentes processos ou recursos podem ser representados em uma mesma rede. As cores não significam apenas cores ou padrões. Elas podem representar tipos de dados complexos, usando a nomenclatura de colorida apenas para referenciar a possibilidade de distinção entre os tokens [JEN90]. A figura 2.10 apresenta uma rede colorida, possuindo a representação original, onde são realmente utilizadas cores para os tokens. Nessa figura, os arcos são rotulados com cores (a, b, c).

No exemplo da figura 2.10, utiliza-se o modo mais elementar de redes coloridas, no qual se associa ao arco uma determinada cor, assim, o token se destinará ao arco cuja cor for idêntica a da marca. Observando-se essa figura, pode-se perceber que os tokens de P0 não habilitarão a transição t0, pois o arco que liga P0 a t0 só aceita cores do tipo “a”, e o lugar P0 só possui marcas do tipo “d”. Em contrapartida, P1 possui marcas do tipo “a”, podendo habilitar a transição t1.

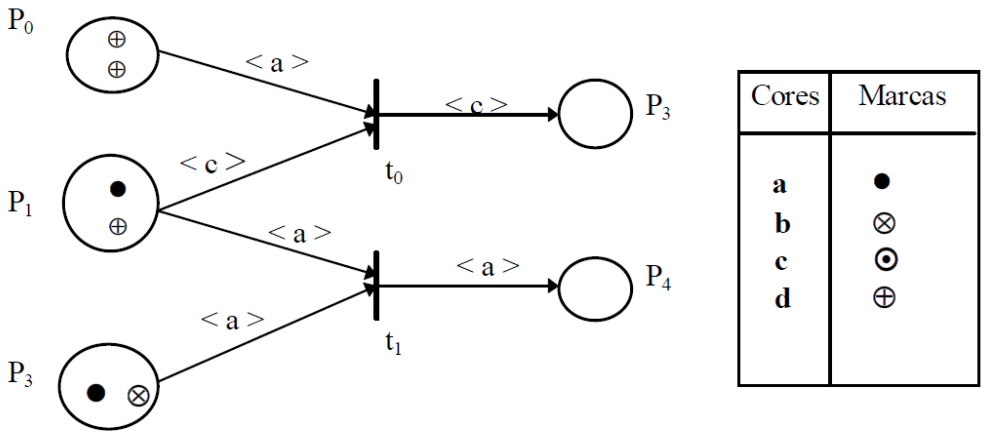


Figura 2.11 - Rede de Petri Colorida

Ainda que um tanto quanto rudimentar, a rede colorida original provê mecanismos que possibilitam efetuar uma escolha determinística. Esse poder de escolha já significa um grande avanço em direção a uma representação mais clara de um modelo, porém modificações (acréscimos) posteriores vieram dar maior adequação às redes coloridas, com relação à representação das escolhas não-determinísticas. A seguir, processa-se uma discussão a respeito de algumas melhorias adicionadas às redes coloridas, tornando-as mais poderosas. Para facilitar o uso da nomenclatura, faz-se referência às redes coloridas com as melhorias adicionais, chamando-as somente por redes de Petri coloridas.

As redes de Petri coloridas são compostas por três diferentes partes [MAC96]:

- estrutura,

- declarações,

- inscrições.

Estrutura é um grafo dirigido com dois tipos de vértices (lugares e transições). Os lugares são representados graficamente por círculos (ou por elipses) e as transições por retângulos. Essa representação herda a propriedade das redes coloridas originais de poder armazenar em cada lugar marcas de tipos diferentes, além de poder representar valores associados a tipos de dados mais complexos. Declarações compreendem a especificação dos conjuntos de cores e declarações

de variáveis. As inscrições variam de acordo com o componente da rede. Os lugares possuem três tipos de inscrições: nomes, conjunto de cores e expressão de inicialização (marcação inicial). As transições têm dois tipos de inscrições: nomes e expressões guarda, e os arcos apenas um tipo de inscrição dado pela expressão.

Como formas para distinguir as inscrições, nomes são escritos com letras normais, cores em itálico, expressões de inicialização sublinhadas e as expressões guarda são colocadas entre colchetes.

Nomes, quando associados a lugares, não têm significado formal, apenas facilitam a identificação. As expressões guarda associadas às transições são expressões booleanas que devem ser atendidas para que seja possível o disparo das transições [MAC96].

Para ilustrar a aplicação de redes de Petri coloridas, será mostrada uma situação clássica de geração de impasse: o jantar dos filósofos [TAN95]. Essa situação consiste de três filósofos que podem estar em três estados diferentes: comendo, pensando ou com fome. Os filósofos estão à volta de uma mesa, sendo que cada um deles tem à sua frente um garfo e um prato de comida. São, no entanto, necessários dois garfos para que um filósofo possa comer, ou seja, um filósofo precisa do seu garfo e do de seu vizinho. O impasse ocorrerá quando todos os filósofos pegarem o garfo da direita e aguardarem a liberação do garfo da esquerda. A figura 2.9 [MAC96] apresenta o jantar dos filósofos modelado em rede de Petri colorida.

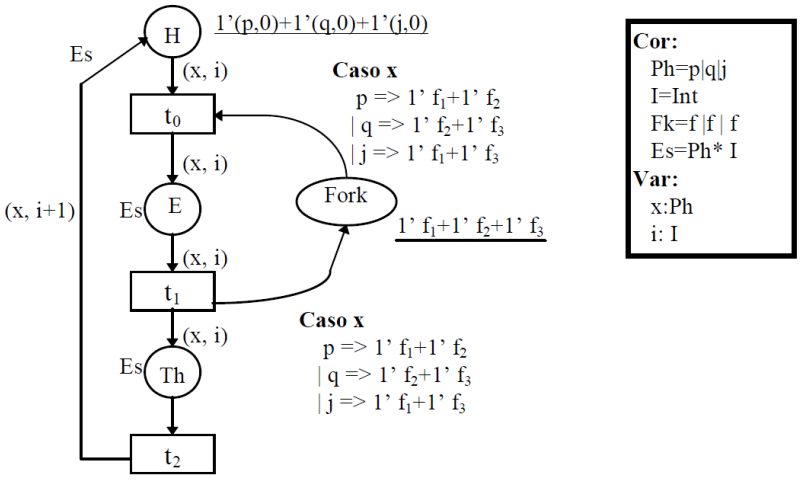


Figura 2.12 - Jantar dos Filósofos em redes coloridas

Na figura 2.9, os lugares representam os estados possíveis para os filósofos (H para “com fome”, E para “comendo” e Th para “pensando”) e os recursos do sistema (no caso, os garfos representados por Fork). A variável x indica o filósofo que irá passar para o estado “comendo” (E) e a variável i indica o número de iterações que já ocorreram. Na primeira iteração, o lugar H possui três marcas (marcação inicial) que estão sublinhadas.

Dependendo da atribuição dada à variável x, uma das três expressões é avaliada no arco que liga o lugar Fork à transição t0. Assim, se, por exemplo, x = p, então a expressão 1’ f1 + 1’ f2 é avaliada e apenas a marcação 1’(p, 0) do lugar H irá à transição t0, significando que apenas o filósofo p irá passar para o estado “comendo”. Desta forma, para cada atribuição de x (p, q ou j) haverá uma situação diferente, ou seja, uma marca diferente no lugar E (estado “comendo”). A modelagem colorida, além de evitar o impasse (o que também poderia ser obtido através de uma rede não-determinística), ainda possibilita uma representação mais clara e concisa ao modelo.

Apenas para efeito de comparação, apresenta-se também a figura 2.9, ilustrando o mesmo problema da figura 2.9, o jantar dos filosofos. Só que na 2.10 [MAC96], a representação utilizada segue a metodologia das redes de Petri ordinárias. A comparação evidencia o poder de concisão e clareza que as redes coloridas proporcionam. Como é usada apenas para efeito ilustrativo, a rede ordinária não será explicada. Essa comparação é plenamente discutida em [MAC96].

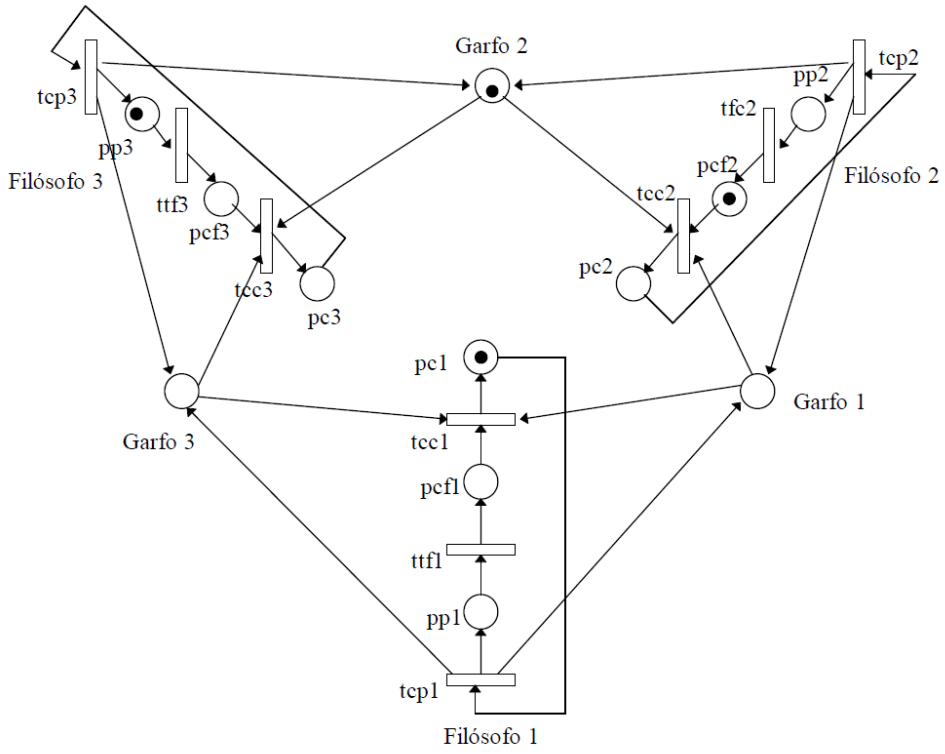


Figura 2.13 - Jantar dos Filósofos em rede Ordinária

2.4.2. Redes Hierárquicas

Um dos problemas apresentados nas redes de Petri originais é o fato que, à medida que o tamanho do sistema cresce, vai se tornando cada vez mais difícil manter a clareza do modelo. Esse fato se deve, em grande parte, à falta de hierarquização dos modelos originais. Para amenizar essa limitação, foram criados mecanismos que possibilitam o agrupamento ou o refinamento de partes do modelo. Um dos problemas dessa abordagem é manter a consistência com os elementos vizinhos àqueles que sofrem um agrupamento. Na abordagem hierárquica, mostrada nesta seção, lugares e transições podem ser apresentados sob uma ótica de mais alto nível.

Na representação hierárquica, dois componentes são fundamentais para viabilizar uma representação em mais alto nível: a superpágina e a subpágina [DIT95]. A primeira representa um

agrupamento de componentes (transições, lugares e arcos), visando gerar um modelo mais compacto e inteligível, como se fosse uma "caixa preta". Já as subpáginas são o detalhamento de uma superpágina, de forma a esclarecer alguns detalhes omitidos na representação em alto nível. A figura 2.11 apresenta uma representação em alto nível para o exemplo do protocolo de comunicação visto na figura 2.7. O exemplo mostra a representação do transmissor e dos receptores 1 e 2 em forma de superpáginas.

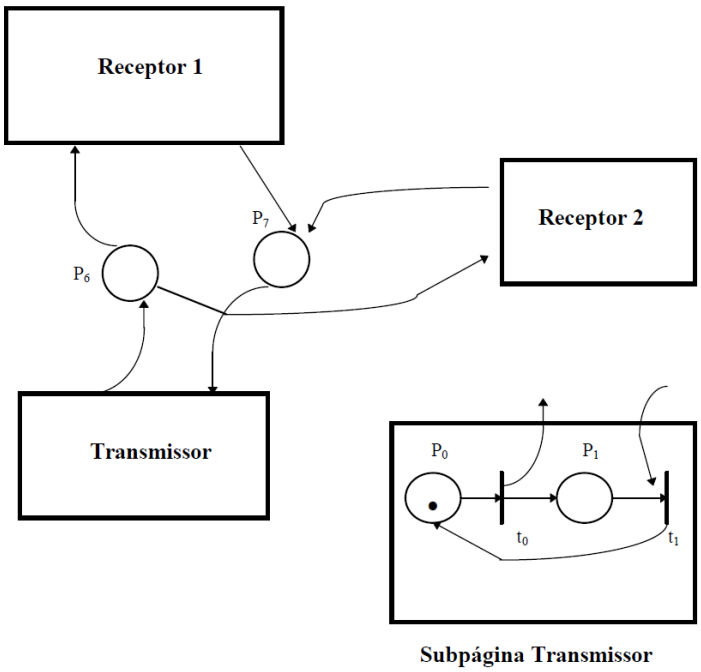


Figura 2.14 - Hierarquia utilizando Superpáginas

2.4.3. Rede de Petri Temporizada Determinística

Para suprir a necessidade de escolha determinística (assim como nas redes coloridas), foram desenvolvidas as redes de Petri temporizadas. Essas redes possibilitam a representação do comportamento dinâmico de sistemas que possuam atividades concorrentes, assíncronas e não-determinísticas, através da adição do conceito de tempo no modelo [COO83]. O tempo também pode ser usado de maneira probabilística, ou seja, o disparo de transições está associado a

distribuições de probabilidade. Essas redes são denominadas redes de Petri estocásticas, pois seus comportamentos podem ser descritos por processos estocásticos.

A associação do tempo a componentes da rede pode se realizar de várias maneiras.

As principais são [MAC96]:

- O tempo associado aos lugares. Assim, os tokens (após o disparo de uma transição) só estarão disponíveis para disparar uma nova transição após um determinado tempo que está associado ao lugar.

- O tempo associado às marcas. Nesse caso o tempo indica quando a marca estará disponível para disparar uma transição.

- O tempo associado às transições. O objetivo desta seção é enfocar as redes de Petri temporizadas determinísticas com tempos associados às transições. Por uma questão prática, quando se fizer referência à rede de Petri temporizada, estará subentendido que está-se referindo às redes de Petri temporizadas determinísticas com tempos associados às transições.

Um exemplo de rede de Petri temporizada determinística é apresentado na figura 2.12.

Nela, as transições t1 e t2 possuem tempos associados diferentes, o que significa que uma transição será disparada antes da outra. Supondo-se que d1 < d2, então o token chegará primeiro ao lugar P1. Assim, de maneira determinística, pode-se estabelecer a ordem em que os eventos devem ocorrer.

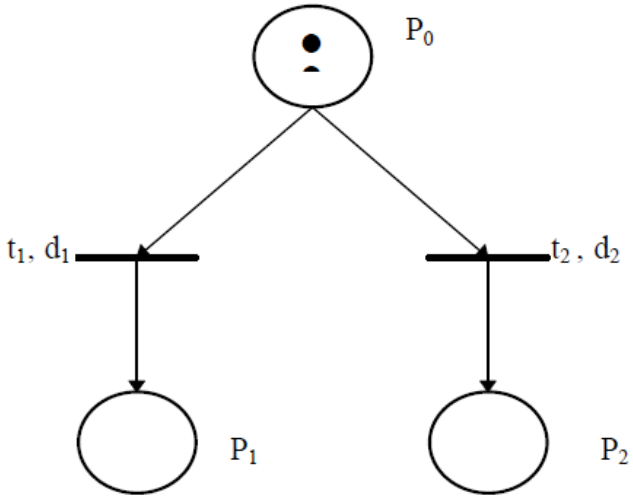


Figura 2.15 - Rede de Petri temporizada determinística

2.5. Análise das Redes de Petri

Há três grupos de métodos para se analisar as redes de Petri: análise baseada na árvore de cobertura, métodos baseados na equação fundamental das redes de Petri e as técnicas de redução [MUR89]. Esta seção enfoca a análise baseada na equação fundamental das redes de Petri (EFRP).

2.5.1. Análise da EFRP

A equação fundamental, ou equação de estados, possibilita a análise da acessibilidade das marcações, bem como o número de vezes que cada transição deve ser disparada para que se obtenha a referida marcação [MUR77].

A Equação Fundamental das Redes de Petri é:

$$M' (p) = M0 (p) + C. s, \forall p \in P$$

Onde s é o vetor característico cujos componentes si são naturais e representam o número de vezes que cada transição ti foi disparada para obter-se a marcação M'(p) a partir de M0(p), e C é a matriz de incidência, dada pela diferença entre as matrizes de pós e précondições (C = O - I). Assim, pode-se determinar se uma marcação Mk é acessível a partir de M0.

Observando-se a figura 2.16 [MAC96], pode-se analisar a acessibilidade da marcação M' = (0,0,0,1,1,0,1) a partir da marcação inicial apresentada.

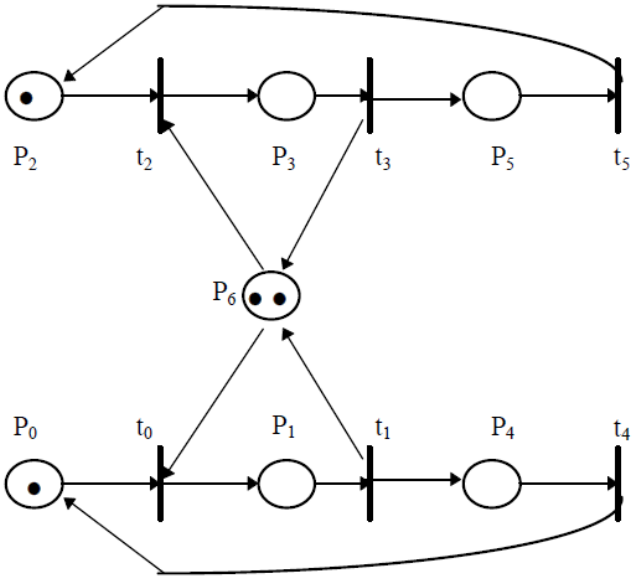


Figura 2.16 - Acessibilidade de uma marcação

A partir da EFRP, tem-se o seguinte sistema matricial:

$$\begin{vmatrix} 0\\0\\0\\1\\1\\0\\1 \end{vmatrix} - \begin{vmatrix} 1\\0\\1\\0\\0\\0\\2 \end{vmatrix} = \begin{vmatrix} -1 & 0 & 0 & 0 & 1 & 0\\ 1 & -1 & 0 & 0 & 0 & 0\\ 0 & 0 & -1 & 0 & 0 & 1\\ 0 & 0 & 1 & -1 & 0 & 0\\ 0 & 1 & 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1 & 0 & -1\\ -1 & 1 & -1 & 1 & 0 & 0 \end{vmatrix} \text{ X } \begin{vmatrix} s_0\\s_1\\s_2\\s_3\\s_4\\s_5 \end{vmatrix}$$

A partir da solução do sistema matricial, podem-se determinar as relações entre os números de disparos de das transições. Por exemplo, tem-se que s0 = s4 + 1, isto é, o número de disparos de t0 é uma unidade maior que o número de disparos da transição t4, e s0 = s1, indicando que o número de disparos de t0 deve ser igual ao número de disparos de t1. Assim, a partir dos elementos de s, podem-se obter as seguintes relações:

$$s0 = s4 + 1,\ s1 = s0\ ,\ s2 = s5 + 1,\ s3 = s2 - 1,\ s4 = s1 - 1,\ s5 = s3$$

Se considerar-se s0 = 1 e s3 = 0, tem-se como solução do problema s = (1,1,1,0,0,0).

Significando que com os disparos das transições t0, t1 e t2, obtém-se a marcação M', a partir de M0, isto é, M' é alcançável a partir de M0. Vale observar que a marcação M' pode ser alcançada em outras relações de disparo de transições.

2.6. Redes de Petri Estocásticas

As redes de Petri, ao contrário das redes de filas, não foram desenvolvidas originalmente para prover avaliação de desempenho, apesar de toda a sua potencialidade para representar sistemas complexos, os quais naturalmente requerem cuidados a esse respeito. Esse panorama se modificou em 1982, quando M. K. Molloy (Molloy, 1982) apresentou as redes de Petri estocásticas (Stochastic Petri Nets - SPN) como uma técnica capaz de, além de especificar sistemas, também apresentar uma análise probabilística dos mesmos.

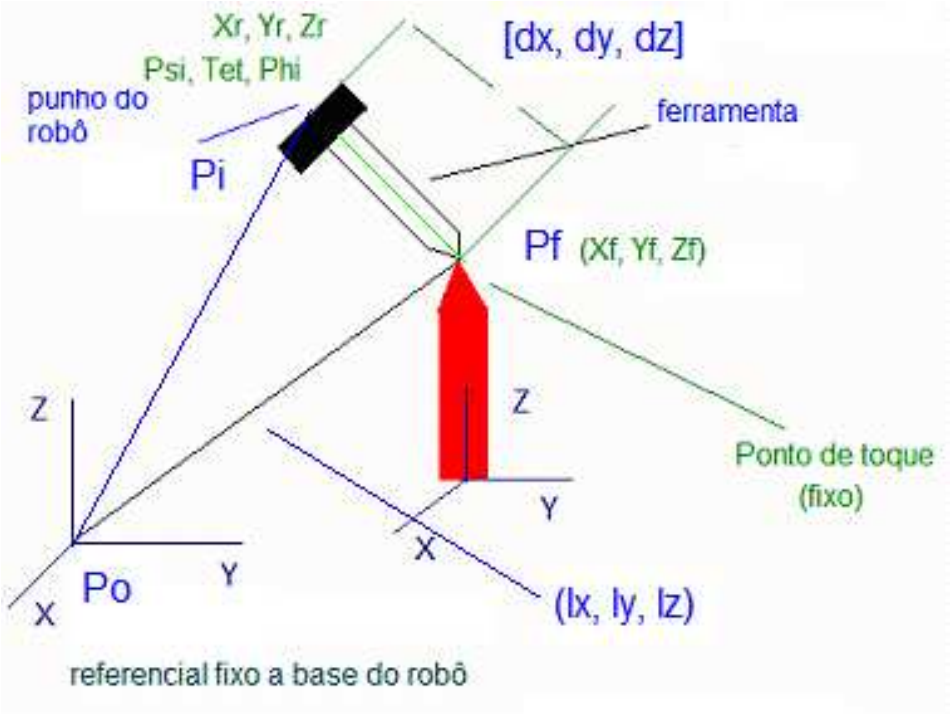
Molloy definiu que todas as transições em uma SPN eram temporizadas (timed), e que possuíam um retardo exponencialmente distribuído. Através dessa implicação, as SPN seriam isomórficas às cadeias de Markov, e assim poderiam prover medidas de desempenho.

Posteriormente, G. Chiola (Chiola et. al. 1993) apresentou uma melhoria às SPN, denominada Redes de Petri Estocásticas Generalizadas (Generalized Stochastic Petri Nets - GSPN), cuja diferença fundamental está em admitir que as transições também podem ser não-estocásticas, isto é, uma transição também pode ser imediata, como nas RP convencionais. Chiola

definiu que as transições imediatas deveriam ter retardo de disparo igual a zero, e que somente as transições estocásticas tinham retardos associados diferentes de zero.

3. Modelo de calibração da ferramenta

3.1. Referencia de Medida



Observação:

$P(i)$ $[X, Y, Z]$: ponto P(i), dimensão 3 x 1

$[\Psi(i), \theta(i), \Phi(i)] = \begin{bmatrix} Nx & Sx & Ax \\ Ny & Sy & Ay \\ Nz & Sz & Az \end{bmatrix}$: matriz de orientação Euler (3 x 3) (mesmo sistema do robô)

$Nx = C\Phi * C\theta$; $Ny = C\theta * S\Phi$; $Nz = -S\theta$

$Sx = C\Phi * S\theta * S\Psi - C\Psi * S\Phi$; $Sy = S\Phi * S\theta * S\Psi + C\Phi * C\Psi$ $Sz = C\theta * S\Psi$

$Ax = C\Phi * S\theta * C\Psi + S\Phi * S\Psi$; $Ay = S\Phi * S\theta * C\Psi - C\Phi * S\Psi$ $Az = C\theta * C\Psi$

$C\Psi = \cos(\Psi)$ $C\theta = \cos(\theta)$ $C\Phi = \cos(\Phi)$

$S\Psi = \sin(\Psi)$ $S\theta = \sin(\theta)$ $S\Phi = \sin(\Phi)$

3.2. Parametros de Medida

Pontos:

$P(0): [X(0), Y(0), Z(0)], [\Psi(0), \theta(0), \Phi(0)]$: ponto 0, referencial fixo a base do robô (zero do robô, bem conhecido (0, 0, 0))

$P(i): [X(i), Y(i), Z(i)], [\Psi(i), \theta(i), \Phi(i)]$: ponto i, ponto das medidas efetuadas em relação ao ponto fixo de calibração, lido no punho do robô (perfeitamente conhecido a partir do robô)

$Pt: [Xt, Yt, Zt], [\Psi t, \theta t, \Phi t]$: ponto de contato do tool (ele é fixo, mas este ponto deverá ser estimado e atualizado em cada iteração)

Parâmetros desconhecidos e valores iniciais atribuídos:

$Pt: [Xt, Yt, Zt], [\Psi t, \theta t, \Phi t]$: ponto de contato do tool (ponto a ser estimado e atualizado a cada iteração)

[dx, dy, dz]: dimensões da ferramenta (atualizado a cada iteração).

[lx, ly, lz]: distância do ponto de contato de ferramenta (Pt) até o referencial zero do robô P(0) (atualizado a cada iteração).

Parâmetros bem conhecidos:

$P(0): [X(0), Y(0), Z(0)], [\Psi(0), \theta(0), \Phi(0)]$: ponto zero do robô

$P(i): [X(i), Y(i), Z(i)], [\Psi(i), \theta(i), \Phi(i)]$: TCP da ferramenta obtido através do robô

3.3.Equações utilizadas

$PoPi = PoPt + PtPi$ (equação vetorial)

$$[P(i) - P(0)]_{3x1} = [Pt - P(0)]_{3x1} + [P(i) - Pt]_{3x1} \quad (1)$$

$$[P(i) - Pt]_{3x1} = [\Psi(i) - \Psi t, \theta(i) - \theta t, \Phi(i) - \Phi t]_{3x3} * [dx, dy, dz]_{3x1} \quad (2)$$

$$[Pt - P(0)]_{3x1} = [\Psi t, \theta t, \Phi t]_{3x3} * [lx, ly, lz]_{3x1} \quad (3)$$

(2)+(3) = (1):

$$[P(i) - P(0)]_{3x1} = [\Psi(i) - \Psi t, \theta(i) - \theta t, \Phi(i) - \Phi t]_{3x3} * [dx, dy, dz]_{3x1} + [\Psi t, \theta t, \Phi t]_{3x3} * [lx, ly, lz]_{3x3}$$

Onde:

$P(i) = [X(i), Y(i), Z(i)]$: vetor de dimensão (3x1) obtido a partir do robô (seu valor sempre é bem conhecido)

$P(0) = [X(0), Y(0), Z(0)]$: vetor de dimensão (3x1) obtido a partir do robô (zero do robô) (seu valor sempre é bem conhecido): $P(0) = [0, 0, 0]$

$[dx, dy, dz]$: vetor tamanho da ferramenta de dimensão (3x1) (valor atualizado a cada iteração)

$[lx, ly, lz]$: vetor de dimensão (3x1) relativo a distância do ponto de calibração da ferramenta (Pt) em relação ao zero do robô P(0) (valor estimado e atualizado a cada medida)

$[\Psi(i+1) - \Psi(1), \theta(i+1) - \theta(1), \Phi(i+1) - \Phi(1)]$: matriz de orientação (3x3) conhecida e resultante entre o ponto 1 da peça e o ponto de medida i+1 (seu valor deverá ser sempre atualizado)

$[\Psi(0), \theta(0), \Phi(0)]$: matriz de orientação (3x3) (zero do robô) (matriz identidade)

$[\Psi(i), \theta(i), \Phi(i)]$: matriz de orientação (3x) relativa ao ponto i de toque (obtida sempre a partir do robô)

$[\Psi t, \theta t, \Phi t]$: matriz de orientação (3x3) do ponto Pt em relação ao zero P(0) (seu valor deverá sempre ser atualizado)

Observação: valores assinalados em verde são bem conhecidos:

3.4. Equações de Medição (genéricas)

$$X(i) = N_1x * dx + S_1x * dy + A_1x * dz + N_2x * lx + S_2x * ly + A_2x * lz$$

$$X(i) = N_1y * dx + S_1y * dy + A_1y * dz + N_2y * lx + S_2y * ly + A_2y * lz$$

$$X(i) = N_1z * dx + S_1z * dy + A_1z * dz + N_2z * lx + S_2z * ly + A_2z * lz$$

Obs.: $P(0) = [0\ 0\ 0]$

3.4.1. Obtenção de N1, S1, A1

$N_1x = \cos[\Phi(i) - \Phi t] * \cos[\theta(i) - \theta t];$

$N_1y = \cos[\theta(i) - \theta t] * \sin[\Phi(i) - \Phi t];$

$N_1z = \sin[\theta(i) - \theta t];$

$S_1x = \cos[\Phi(i) - \Phi t] * \sin[\theta(i) - \theta t] * \sin[\Psi(i) - \Psi t] - \cos[\Psi(i) - \Psi t] * \sin[\Phi(i) - \Phi t];$

$S_1y = \sin[\Phi(i) - \Phi t] * \sin[\theta(i) - \theta t] * \sin[\Psi(i) - \Psi t] + \cos[\Phi(i) - \Phi t] * \cos[\Psi(i) - \Psi t];$

$S_1z = \cos[\theta(i) - \theta t] * \sin[\Psi(i) - \Psi t];$

$A_1x = \cos[\Phi(i) - \Phi t] * \sin[\theta(i) - \theta t] * \cos[\Psi(i) - \Psi t] + \sin[\Phi(i) - \Phi t] * \sin[\Psi(i) - \Psi t];$

$A_1y = \sin[\Phi(i) - \Phi t] * \sin[\theta(i) - \theta t] * \cos[\Psi(i) - \Psi t] - \cos[\Phi(i) - \Phi t] * \sin[\Psi(i) - \Psi t]$

$A_1z = \cos[\theta(i) - \theta t] * \cos[\Psi(i) - \Psi t];$

3.4.2. Obtenção de N2, S2, A2 (valores a serem atualizados em cada iteração)

$N_2x = \cos[\Phi t] * \cos[\theta t];$

$N_2y = \cos[\theta t] * \sin[\Phi t];$

$N_2z = \sin[\theta t];$

$S_2x = \cos[\Phi t] * \sin[\theta t] * \sin[\Psi t] - \cos[\Psi t] * \sin[\Phi t];$

$S_2y = \sin[\Phi t] * \sin[\theta t] * \sin[\Psi t] + \cos[\Phi t] * \cos[\Psi t];$

$S_2z = \cos[\theta t] * \sin[\Psi t];$

$A_2x = \cos[\Phi t] * \sin[\theta t] * \cos[\Psi t] + \sin[\Phi t] * \sin[\Psi t];$

$A_2y = \sin[\Phi t] * \sin[\theta t] * \cos[\Psi t] - \cos[\Phi t] * \sin[\Psi t];$

$A_2z = \cos[\theta t] * \cos[\Psi t];$

3.5. Cálculo do Jacobiano (J)

3.5.1. Equações

$$F_1 = X(i) = N_1x * dx + S_1x * dy + A_1x * dz + N_2x * lx + S_2x * ly + A_2x * lz$$

$$F_2 = Y(i) = N_1y * dx + S_1y * dy + A_1y * dz + N_2y * lx + S_2y * ly + A_2y * lz$$

$$F_3 = Z(i) = N_1z * dx + S_1z * dy + A_1z * dz + N_2z * lx + S_2z * ly + A_2z * lz$$

3.5.2. Obtenção do Jacobiano

O procedimento utilizado é análogo ao calculo do zero robô pela peça de calibração e da mesma forma, não dependerá dos valores iniciais e no processo de estimação, além das dimensões da ferramenta, serão estimados a distancia do ponto Pt até Po (vetor lx, ly, lz) e a orientação do de Pt em relação a base do robô (Ψt, Φt, θt).

Para o cálculo da matriz JACOBIANA, podemos observar que resolvendo o sistema (DIRETO).

$$\left|\frac{\delta F}{\delta t}\right|_{3x1} = J_{3x9} * \left|\frac{\delta \vec{W}}{\delta t}\right|_{9x1}$$

Onde:

$$\left|\frac{\delta F}{\delta t}\right|_{3x1} = \left|\frac{\delta F_1}{dt} \quad \frac{\delta F_2}{dt} \quad \frac{\delta F_3}{dt}\right| = |\delta X \quad \delta Y \quad \delta Z|_{3x1}$$

$$\left|\frac{\delta \vec{W}}{\delta t}\right|_{9x1} = \left|\frac{\delta dx}{\delta t} \quad \frac{\delta dy}{\delta t} \quad \frac{\delta dz}{\delta t} \quad \frac{\delta \Psi_T}{\delta t} \quad \frac{\delta \theta_T}{\delta t} \quad \frac{\delta \Phi_T}{\delta t} \quad \frac{\delta lx}{\delta t} \quad \frac{\delta ly}{\delta t} \quad \frac{\delta lz}{\delta t}\right|$$

J_{3x9} matiz Jacobiana

$$|J|_{3x9} = \begin{vmatrix} J_{11} & J_{12} & J_{13} & J_{14} & J_{15} & J_{16} & J_{17} & J_{18} & J_{19} \\ J_{21} & J_{22} & J_{23} & J_{24} & J_{25} & J_{26} & J_{27} & J_{28} & J_{29} \\ J_{31} & J_{32} & J_{33} & J_{34} & J_{35} & J_{36} & J_{37} & J_{38} & J_{39} \end{vmatrix}$$

3.5.3. Calculo dos elementos do Jacobiano

$J_{11} = \frac{dF_1}{d(dx)} = N_1 x$ $\quad$ $J_{12} = \frac{dF_1}{d(dy)} = S_1 x$ $\quad$ $J_{13} = \frac{dF_1}{d(dz)} = A_1 x$

$J_{17} = \frac{dF_1}{d(lx)} = N_2 x$ $\quad$ $J_{18} = \frac{dF_1}{d(ly)} = S_2 x$ $\quad$ $J_{19} = \frac{dF_1}{d(lz)} = A_2 x$

$J_{21} = \frac{dF_2}{d(dx)} = N_1 y$ $\quad$ $J_{22} = \frac{dF_2}{d(dy)} = S_1 y$ $\quad$ $J_{23} = \frac{dF_2}{d(dz)} = A_1 y$

$J_{27} = \frac{dF_2}{d(lx)} = N_2 y$ $\quad$ $J_{28} = \frac{dF_2}{d(ly)} = S_2 y$ $\quad$ $J_{29} = \frac{dF_2}{d(lz)} = A_2 y$

$J_{31} = \frac{dF_3}{d(dx)} = N_1 z$ $\quad$ $J_{32} = \frac{dF_3}{d(dy)} = S_1 z$ $\quad$ $J_{33} = \frac{dF_3}{d(dz)} = A_1 z$

$$J_{37} = \frac{dF_3}{d(lx)} = N_2 z \qquad J_{38} = \frac{dF_3}{d(ly)} = S_2 z \qquad J_{39} = \frac{dF_3}{d(lz)} = A_2 z$$

$$J_{14} = \frac{dF_1}{d(\Psi t)} = \left|\frac{dN_1 x}{d(\Psi t)}\right| * dx + \left|\frac{dS_1 x}{d(\Psi t)}\right| * dy + \left|\frac{dA_1 x}{d(\Psi t)}\right| * dz + \left|\frac{dN_2 x}{d(\Psi t)}\right| * lx + \left|\frac{dS_2 x}{d(\Psi t)}\right| * ly + \left|\frac{dA_2 x}{d(\Psi t)}\right| * lz$$

$$J_{15} = \frac{dF_1}{d(\theta t)} = \left|\frac{dN_1 x}{d(\theta t)}\right| * dx + \left|\frac{dS_1 x}{d(\theta t)}\right| * dy + \left|\frac{dA_1 x}{d(\theta t)}\right| * dz + \left|\frac{dN_2 x}{d(\theta t)}\right| * lx + \left|\frac{dS_2 x}{d(\theta t)}\right| * ly + \left|\frac{dA_2 x}{d(\theta t)}\right| * lz$$

$$J_{16} = \frac{dF_1}{d(\Phi t)} = \left|\frac{dN_1 x}{d(\Phi t)}\right| * dx + \left|\frac{dS_1 x}{d(\Phi t)}\right| * dy + \left|\frac{dA_1 x}{d(\Phi t)}\right| * dz + \left|\frac{dN_2 x}{d(\Phi t)}\right| * lx + \left|\frac{dS_2 x}{d(\Phi t)}\right| * ly + \left|\frac{dA_2 x}{d(\Phi t)}\right| * lz$$

$$J_{24} = \frac{dF_2}{d(\Psi t)} = \left|\frac{dN_1 y}{d(\Psi t)}\right| * dx + \left|\frac{dS_1 y}{d(\Psi t)}\right| * dy + \left|\frac{dA_1 y}{d(\Psi t)}\right| * dz + \left|\frac{dN_2 y}{d(\Psi t)}\right| * lx + \left|\frac{dS_2 y}{d(\Psi t)}\right| * ly + \left|\frac{dA_2 y}{d(\Psi t)}\right| * lz$$

$$J_{25} = \frac{dF_2}{d(\theta t)} = \left|\frac{dN_1 y}{d(\theta t)}\right| * dx + \left|\frac{dS_1 y}{d(\theta t)}\right| * dy + \left|\frac{dA_1 y}{d(\theta t)}\right| * dz + \left|\frac{dN_2 y}{d(\theta t)}\right| * lx + \left|\frac{dS_2 y}{d(\theta t)}\right| * ly + \left|\frac{dA_2 y}{d(\theta t)}\right| * lz$$

$$J_{26} = \frac{dF_2}{d(\Phi t)} = \left|\frac{dN_1 y}{d(\Phi t)}\right| * dx + \left|\frac{dS_1 y}{d(\Phi t)}\right| * dy + \left|\frac{dA_1 y}{d(\Phi t)}\right| * dz + \left|\frac{dN_2 y}{d(\Phi t)}\right| * lx + \left|\frac{dS_2 y}{d(\Phi t)}\right| * ly + \left|\frac{dA_2 y}{d(\Phi t)}\right| * lz$$

$$J_{34} = \frac{dF_3}{d(\Psi t)} = \left|\frac{dN_1 z}{d(\Psi t)}\right| * dx + \left|\frac{dS_1 z}{d(\Psi t)}\right| * dy + \left|\frac{dA_1 z}{d(\Psi t)}\right| * dz + \left|\frac{dN_2 z}{d(\Psi t)}\right| * lx + \left|\frac{dS_2 z}{d(\Psi t)}\right| * ly + \left|\frac{dA_2 z}{d(\Psi t)}\right| * lz$$

$$J_{35} = \frac{dF_3}{d(\theta t)} = \left|\frac{dN_1 z}{d(\theta t)}\right| * dx + \left|\frac{dS_1 z}{d(\theta t)}\right| * dy + \left|\frac{dA_1 z}{d(\theta t)}\right| * dz + \left|\frac{dN_2 z}{d(\theta t)}\right| * lx + \left|\frac{dS_2 z}{d(\theta t)}\right| * ly + \left|\frac{dA_2 z}{d(\theta t)}\right| * lz$$

$$J_{36} = \frac{dF_3}{d(\Phi t)} = \left|\frac{dN_1 z}{d(\Phi t)}\right| * dx + \left|\frac{dS_1 z}{d(\Phi t)}\right| * dy + \left|\frac{dA_1 z}{d(\Phi t)}\right| * dz + \left|\frac{dN_2 z}{d(\Phi t)}\right| * lx + \left|\frac{dS_2 z}{d(\Phi t)}\right| * ly + \left|\frac{dA_2 z}{d(\Phi t)}\right| * lz$$

Obs.:

$\frac{d[\sin[\Psi(i)-\Psi t]]}{d(\Psi t)} = -\cos[\Psi(i) - \Psi t$ $\frac{d[\cos[\Psi(i)-\Psi t]]}{d(\Psi t)} = \sin[\Psi(i) - \Psi t$

Derivadas parciais (N_1)

$$\left|\frac{dN_1 x}{d(\Psi t)}\right| = 0$$

$$\left|\frac{dS_1 x}{d(\Psi t)}\right| = -\cos[\Phi(i) - \Phi t] * \sin[\theta(i) - \theta t] * \cos[\Psi(i) - \Psi t] - \sin[\Phi(i) - \Phi t] * \sin[\Psi(i) - \Psi t]$$

$$\left|\frac{dA_1 x}{d(\Psi t)}\right| = \cos[\Phi(i) - \Phi t] * \sin[\theta(i) - \theta t] * \sin[\Psi(i) - \Psi t] - \sin[\Phi(i) - \Phi t] * \cos[\Psi(i) - \Psi t]$$

$$\left|\frac{dN_1 y}{d(\Psi t)}\right| = 0$$

$$\left|\frac{dS_1 y}{d(\Psi t)}\right| = -\sin[\Phi(i) - \Phi t] * \sin[\theta(i) - \theta t] * \cos[\Psi(i) - \Psi t] + \cos[\Phi(i) - \Phi t] * \sin[\Psi(i) - \Psi t]$$

$$\left|\frac{dA_1 y}{d(\Psi t)}\right| = \sin[\Phi(i) - \Phi t] * \sin[\theta(i) - \theta t] * \sin[\Psi(i) - \Psi t] + \cos[\Phi(i) - \Phi t] * \cos[\Psi(i) - \Psi t]$$

$$\left|\frac{dN_1 z}{d(\Psi t)}\right| = 0$$

$$\left|\frac{dS_1 z}{d(\Psi t)}\right| = -\cos[\theta(i) - \theta t] * \cos[\Psi(i) - \Psi t]$$

$$\left|\frac{dA_1 z}{d(\Psi t)}\right| = \cos[\theta(i) - \theta t] * \sin[\Psi(i) - \Psi t]$$

$$\left|\frac{dN_1 x}{d(\theta t)}\right| = \cos[\Phi(i) - \Phi t] * \sin[\theta(i) - \theta t]$$

$$\left|\frac{dS_1 x}{d(\theta t)}\right| = -\cos[\Phi(i) - \Phi t] * \sin[\Psi(i) - \Psi t] * \cos[\theta(i) - \theta t]$$

$$\left|\frac{dA_1 x}{d(\theta t)}\right| = -\cos[\Phi(i) - \Phi t] * \cos[\Psi(i) - \Psi t] * \cos[\theta(i) - \theta t]$$

$$\left|\frac{dN_1 y}{d(\theta t)}\right| = \sin[\Phi(i) - \Phi t] * \sin[\theta(i) - \theta t]$$

$$\left|\frac{dS_1 y}{d(\theta t)}\right| = -\sin[\Phi(i) - \Phi t] * \sin[\Psi(i) - \Psi t] * \cos[\theta(i) - \theta t]$$

$$\left|\frac{dA_1 y}{d(\theta t)}\right| = -\sin[\Phi(i) - \Phi t] * \cos[\Psi(i) - \Psi t] * \cos[\theta(i) - \theta t]$$

$$\left|\frac{dN_1 z}{d(\theta t)}\right| = \cos[\theta(i) - \theta t]$$

$$\left|\frac{dS_1 z}{d(\theta t)}\right| = \sin[\Psi(i) - \Psi t] * \sin[\theta(i) - \theta t]$$

$$\left|\frac{dA_1 z}{d(\theta t)}\right| = \cos[\Psi(i) - \Psi t] * \sin[\theta(i) - \theta t]$$

$$\left|\frac{dN_1 x}{d(\Phi t)}\right| = \cos[\theta(i) - \theta t] * \sin[\Phi(i) - \Phi t]$$

$$\left|\frac{dS_1 x}{d(\Phi t)}\right| = \sin[\theta(i) - \theta t] * \sin[\Psi(i) - \Psi t] * \sin[\Phi(i) - \Phi t] + \cos[\Psi(i) - \Psi t] * \cos[\Phi(i) - \Phi t]$$

$$\left|\frac{dA_1x}{d(\Phi t)}\right| = \sin[\theta(i) - \theta t] * \cos[\Psi(i) - \Psi t] * \sin[\Phi(i) - \Phi t] - \sin[\Psi(i) - \Psi t] * \sin[\Phi(i) - \Phi t]$$

$$\left|\frac{dN_1y}{d(\Phi t)}\right| = -\cos[\theta(i) - \theta t] * \cos[\Phi(i) - \Phi t]$$

$$\left|\frac{dS_1y}{d(\Phi t)}\right| = -\sin[\Psi(i) - \Psi t] * \sin[\theta(i) - \theta t] * \cos[\Phi(i) - \Phi t] + \cos[\Psi(i) - \Psi t] * \sin[\Phi(i) - \Phi t]$$

$$\left|\frac{dA_1y}{d(\Phi t)}\right| = -\cos[\Psi(i) - \Psi t] * \sin[\theta(i) - \theta t] * \cos[\Phi(i) - \Phi t] - \sin[\Psi(i) - \Psi t] * \sin[\Phi(i) - \Phi t]$$

$$\left|\frac{dN_1z}{d(\Phi t)}\right| = 0 \qquad \left|\frac{dS_1z}{d(\Phi t)}\right| = 0 \qquad \left|\frac{dA_1z}{d(\Phi t)}\right| = 0$$

Derivadas parciais (N_2)

$$\left|\frac{dN_2x}{d(\Psi t)}\right| = 0$$

$$\left|\frac{dS_2x}{d(\Psi t)}\right| = \cos[\Phi t] * \sin[\theta t] * \cos[\Psi t] + \sin[\Phi t] * \sin[\Psi t]$$

$$\left|\frac{dA_2x}{d(\Psi t)}\right| = -\cos[\Phi t] * \sin[\theta t] * \sin[\Psi t] + \sin[\Phi t] * \cos[\Psi t]$$

$$\left|\frac{dN_2y}{d(\Psi t)}\right| = 0$$

$$\left|\frac{dS_2y}{d(\Psi t)}\right| = \sin[\Phi t] * \sin[\theta t] * \cos[\Psi t] - \cos[\Phi t] * \sin[\Psi t]$$

$$\left|\frac{dA_2y}{d(\Psi t)}\right| = -\sin[\Phi t] * \sin[\theta t] * \sin[\Psi t] - \cos[\Phi t] * \cos[\Psi t]$$

$$\left|\frac{dN_2z}{d(\Psi t)}\right| = 0$$

$$\left|\frac{dS_2z}{d(\Psi t)}\right| = \cos[\theta t] * \cos[\Psi t]$$

$$\left|\frac{dA_2z}{d(\Psi t)}\right| = -\cos[\theta t] * \sin[\Psi t]$$

$$\left|\frac{dN_2x}{d(\theta t)}\right| = -\cos[\Phi t] * \sin[\theta t]$$

$$\left|\frac{dS_2x}{d(\theta t)}\right| = \cos[\Phi t] * \sin[\Psi t] * \cos[\theta t]$$

$$\left|\frac{dA_2x}{d(\theta t)}\right| = \cos[\Phi t] * \cos[\Psi t] * \cos[\theta t]$$

$$\left|\frac{dN_2y}{d(\theta t)}\right| = -\sin[\Phi t] * \sin[\theta t]$$

$$\left|\frac{dS_2y}{d(\theta t)}\right| = \sin[\Phi t] * \sin[\Psi t] * \cos[\theta t]$$

$$\left|\frac{dA_2y}{d(\theta t)}\right| = \sin[\Phi t] * \cos[\Psi t] * \cos[\theta t]$$

$$\left|\frac{dN_2z}{d(\theta t)}\right| = -\cos[\theta t]$$

$$\left|\frac{dS_2z}{d(\theta t)}\right| = -\sin[\Psi t] * \sin[\theta t]$$

$$\left|\frac{dA_2z}{d(\theta t)}\right| = -\cos[\Psi t] * \sin[\theta t]$$

$$\left|\frac{dN_2x}{d(\Phi t)}\right| = -\cos[\theta t] * \sin[\Phi t]$$

$$\left|\frac{dS_2x}{d(\Phi t)}\right| = -\sin[\theta t] * \sin[\Psi t] * \sin[\Phi t] - \cos[\Psi t] * \cos[\Phi t]$$

$$\left|\frac{dA_2x}{d(\Phi t)}\right| = -\sin[\theta t] * \cos[\Psi t] * \sin[\Phi t] + \sin[\Psi t] * \sin[\Phi t]$$

$$\left|\frac{dN_2y}{d(\Phi t)}\right| = \cos[\theta t] * \cos[\Phi t]$$

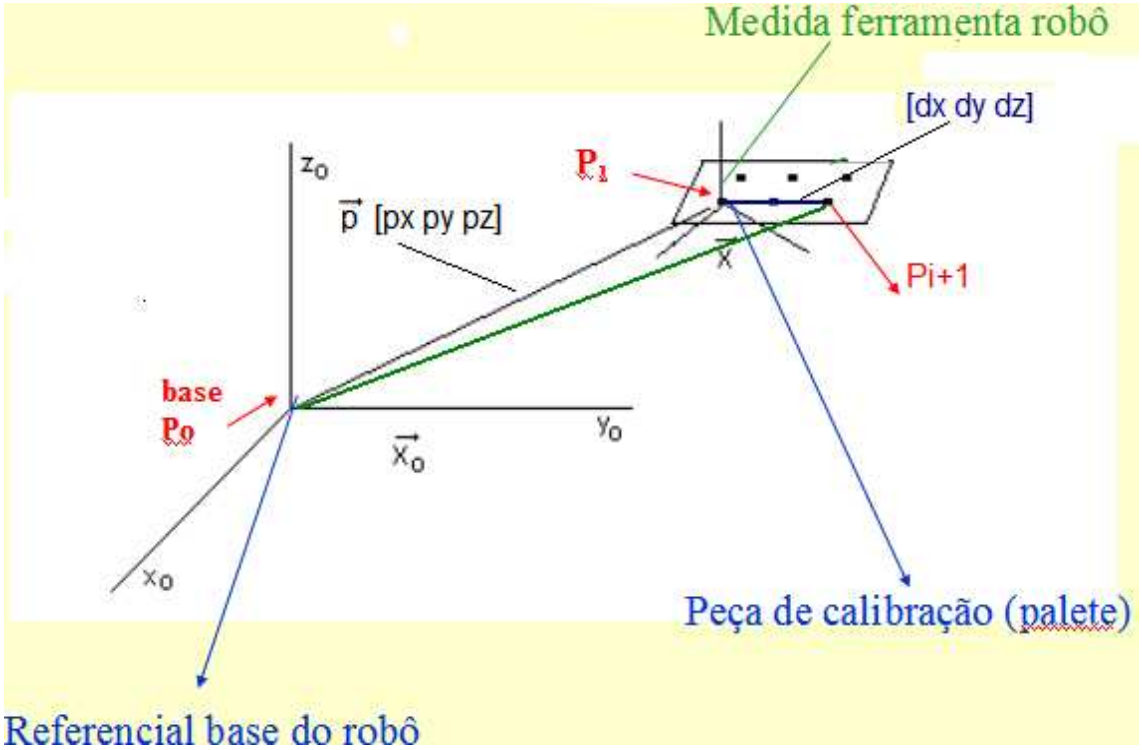
$$\left|\frac{dS_2y}{d(\Phi t)}\right| = \sin[\Psi t] * \sin[\theta t] * \cos[\Phi t] - \cos[\Psi t] * \sin[\Phi t]$$

$$\left|\frac{dA_2y}{d(\Phi t)}\right| = \cos[\Psi t] * \sin[\theta t] * \cos[\Phi t] + \sin[\Psi t] * \sin[\Phi t]$$

$\left|\frac{dN_2z}{d(\Phi t)}\right| = 0$ $\left|\frac{dS_2z}{d(\Phi t)}\right| = 0$ $\left|\frac{dA_2z}{d(\Phi t)}\right| = 0$

4. Método de calibração do ponto zero do Robô

4.1. Referenciais de Medida



Observação:

$P(i)[X,Y,Z]$: Ponto P(i), dimensão 3 x 1

$[\Psi(i),\theta(i),\Psi(i)] = \begin{bmatrix} Nx & Sx & Ax \\ Ny & Sy & Ay \\ Nz & Sz & Az \end{bmatrix}$: matriz de orientação Euler (3 x 3)

$Nx = C\Phi * C\theta;$ $\quad Ny = S\Phi * C\theta;$ $\quad Nz = -S\theta$

$Sx = C\Phi * S\theta * S\Psi - C\Psi * S\Phi;$ $\quad Sy = S\Phi * S\theta * S\Psi + C\Phi * C\Psi;$ $\quad Sy = C\theta * S\Psi$

$Ax = C\Phi * S\theta * C\Psi + S\Phi * S\Psi;$ $\quad Ay = S\Phi * S\theta * C\Psi - C\Phi * S\Psi;$ $\quad Az = C\theta * C\Psi$

$CPsi = \cos(\Psi)$ $\quad C\theta = \cos(\theta)$ $\quad CPhi = \cos(\Phi)$

$SPsi = \sin(\Psi)$ $\quad S\theta = \sin(\theta)$ $\quad SPhi = \sin(\Phi)$

4.2.Fundamentos Básicos

RPY – Orientação

$$RPY\ [\Phi(z), \theta(y), \Psi(x)] = Rot\ [z, \Phi] * Rot\ [y, \theta] * Rot\ [x, \Psi]$$

$$= \begin{bmatrix} C\Phi & -S\Phi & 0 \\ S\Phi & C\Phi & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\Psi & -S\Psi \\ 0 & S\Psi & C\Psi \end{bmatrix}$$

$= \begin{bmatrix} Nx & Sx & Ax \\ Ny & Sy & Ay \\ Nz & Sz & Az \end{bmatrix}$: matriz de orientação Euler (3 x 3) **(mesmo sistema do robô)**

$Nx = C\Phi * C\theta;$ $Ny = S\Phi * C\theta;$ $Nz = -S\theta$

$Sx = C\Phi * S\theta * S\Psi - C\Psi * S\Phi;$ $Sy = S\Phi * S\theta * S\Psi + C\Phi * C\Psi;$ $Sz = C\theta * S\Psi$

$Ax = C\Phi * S\theta * C\Psi + S\Phi * S\Psi;$ $Ay = S\Phi * S\theta * C\Psi - C\Phi * S\Psi;$ $Az = C\theta * C\Psi$

$C\Psi = \cos(\Psi)$ $C\theta = \cos(\theta)$ $C\Phi = \cos(\Phi)$

$S\Psi = \sin(\Psi)$ $S\theta = \sin(\theta)$ $S\Phi = \sin(\Phi)$

Cálculo do Jacobiano para pequenas rotações

δ_x : Rotação infinitesimal em torno do eixo x

δ_y : rotação infinitesimal em torno do eixo y

δ_z : rotação infinitesimal em torno do eixo z

Aproximação: $\sin(\theta) = \theta,$ $\cos(\theta) = 1$

$$Rot\ [x, \delta_x] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -\delta_x \\ 0 & \delta_x & 1 \end{bmatrix}$$

$$Rot\ [y, \delta_y] = \begin{bmatrix} 1 & 0 & \delta_y \\ 0 & 1 & 0 \\ -\delta_y & 0 & 1 \end{bmatrix}$$

$$Rot\,[z,\delta_z] = \begin{bmatrix} 1 & -\delta_z & 0 \\ \delta_z & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matriz de Transformação Diferencial (rotações em Z, Y, X)

$$RPY\,[\delta_z,\delta_y,\delta_x] = Rot\,[z,\delta_z] * Rot\,[y,\delta_y] * Rot\,[x,\delta_x]$$

$$= \begin{bmatrix} 1 & -\delta_z + \delta_y\delta_x & \delta_z\delta_x + \delta_y \\ \delta_z & 1 + \delta_z\delta_y\delta_x & -\delta_z + \delta_y\delta_x \\ -\delta_y & \delta_x & 1 \end{bmatrix} = \begin{bmatrix} 1 & -\delta_z & \delta_y \\ \delta_z & 1 & -\delta_z \\ -\delta_y & \delta_x & 1 \end{bmatrix}$$

Obs: simplificação de termos de 2ª e 3ª ordens

$Nx = 1$; $Ny = \delta_z$; $Nz = -\delta_y$

$Sx = -\delta_z$; $Sy = 1$; $Sz = \delta_x$

$Ax = \delta_y$; $Ay = -\delta_z$; $Az = 1$;

Teorema 1: Uma rotação diferencial δ_θ em torno de um vetor K [K_x, K_y, K_z] é equivalente a três rotações sucessivas [δ_z, δ_y, δ_x] em torno dos eixos x, y, z.

$$K_x,\delta_\theta = \delta_x$$

$$K_y,\delta_\theta = \delta_y$$

$$K_z,\delta_\theta = \delta_z$$

Teorema 2: Rotações diferenciais são independentes da ordem de rotação (quando desprezamos os termos de segunda e terceira ordem).

$$Rot\,[y,\delta_y] * Rot\,[x,\delta_x] = Rot\,[x,\delta_x] * Rot\,[y,\delta_y]$$

4.3.Parâmetros de Medida

d) Pontos:

$P(0)$: $[X(0), Y(0), Z(0)], [\Psi(0), \theta(0), \Phi(0)]$: ponto 0, referencial fixo a base do robô (a ser identificado)

$P(1)$: $[X(1), Y(1), Z(1)], [\Psi(1), \theta(1), \Phi(1)]$: ponto 1, primeiro ponto de medida (conhecido a partir do robô)

$P(i+1)$: $[X(i+1), Y(i+1), Z(i+1)], [\Psi(i+1), \theta(i+1), \Phi(i+1)]$: Ponto i+1, ponto qualquer de medida (conhecido a partir do robô)

e) Parâmetros desconhecidos e valores iniciais atribuídos:

$P(0)$: $[X(0), Y(0), Z(0)] = [0\ 0\ 0]$ $[\Psi(0), \theta(0), \Phi(0)] = [0\ 0\ 0]$

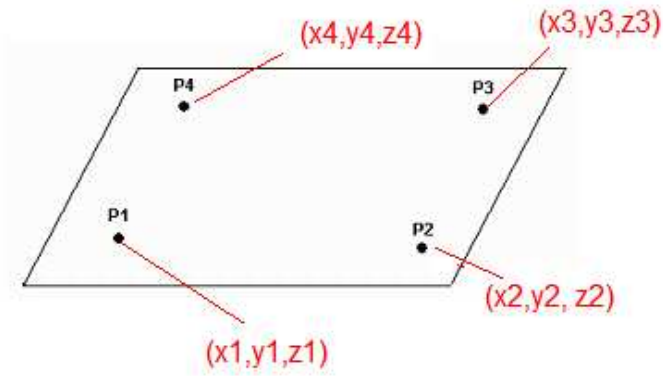
$[px, py, pz]$: distância do ponto P(0) (zero do robô) até o primeiro ponto do palete (atualizado a cada iteração).

f) Parâmetros bem conhecidos:

$P(1)$: $[X(1), Y(1), Z(1)], [\Psi(1), \theta(1), \Phi(1)]$: obtido através do robô (deverá ser atualizado a cada iteração pelo valor real do robô considerando o novo zero do robô obtido anteriormente)

$P(i+1) = [X(i+1), Y(i+1), Z(i+1)], [\Psi(i+1), \theta(i+1), \Phi(i+1)]$: obtido através do robô (deverá ser atualizado a cada iteração pelo valor real do robô considerando o novo zero do robô obtido anteriormente)

$[dx, dy, dz]$: vetos posição ao longo do palete (ponto em relação a P(1)) (conhecido através do desenho do palete de calibração utilizado)



4.4. Equações utilizadas

$$[P(1) - P(0)] = [\Psi(0), \theta(0), \Phi(0)] * [\Psi(1), \theta(1), \Phi(1)] * [px, py, pz] \quad (1)$$

$$[P(i+1) - P(1)] = [\Psi(i+1) - \Psi(i), \theta(i+1) - \theta(i), \Phi(i+1) - \Phi(i)] * [dx, dy, dz] \quad (2)$$

$$[P(i+1) - P(0)] = [P(i+1) - P(1)] + [P(1) - P(0)] \quad (3)$$

(1) + (2) = (3):

$$[P(i+1) - P(0)] = [\Psi(i+1) - \Psi(1), \theta(i+1) - \theta(1), \Phi(i+1) - \Phi(1)] * [dx, dy, dz] +$$

$$+ [\Psi(0), \theta(0), \Phi(0)] * [\Psi(1), \theta(1), \Phi(1)] * [px, py, pz]$$

Onde:

$P(i+1) = X(i+1), Y(i+1), Z(i+1)$: vetor de dimensão (3 x 1) obtido a partir do robô. (seu valor deverá ser sempre atualizado)

$P(0) = X(0), Y(0), Z(0)$: vetor de dimensão (3 x 1) obtido a partir do robô. (seu valor deverá ser sempre atualizado)

$[dx, dy, dz]$: vetor de dimensão (3 x 1) conhecido a partir do palete de calibração

$[px, py, pz]$: vetor de dimensão (3 x 1) relativo a distância do ponto 1 do palete ao zero do robô (valor estimado e atualizado a cada medida)

$[\Psi(i+1) - \Psi(1), \theta(i+1) - \theta(1), \Phi(i+1) - \Phi(1)]$: matriz de orientação (3 x 3) conhecida e resultante entre o ponto 1 do palete e o ponto de medida i+1 (seu valor deverá ser sempre atualizado)

$[\Psi(0), \theta(0), \Phi(0)]$: matriz de orientação (3 x 3) estimada (zero do robô) (valor estimado e atualizado a cada medida)

$[\Psi(1), \theta(1), \Phi(1)]$: matriz de orientação (3 x 3) relativa ao ponto 1 do pacote conhecida a partir do robô (seu valor deverá ser sempre atualizado)

Observação: valores assinalados em verde são vem conhecidos.

4.4.1. Equações de medição (genéricas)

$$X(i+1) = X(0) + N_1x * dx + S_1x * dy + A_1x * dz + N_2x * px + S_2x * py + A_2x * pz$$

$$Y(i+1) = Y(0) + N_1y * dx + S_1y * dy + A_1y * dz + N_2y * px + S_2y * py + A_2y * pz$$

$$Z(i+1) = Z(0) + N_1z * dx + S_1z * dy + A_1z * dz + N_2z * px + S_2z * py + A_2z * pz$$

Obtenção de X(0), Y(0) e Z(0) (valores estimados a cada iteração)

4.4.2. Obtenção de N1, S1, A1 e dx, dy, dz (valores constantes, mas devendo ser sempre atualizados pelas informações do robô e palete)

$$N_1x = \cos[\Phi(i+1) - \Phi(1)] * \cos[\theta(i+1) - \theta(1)];$$

$$N_1y = \cos[\theta(i+1) - \theta(1)] * \sin[\Phi(i+1) - \Phi(1)];$$

$$N_1z = -\sin[\theta(i+1) - \theta(1)];$$

$$S_1x = \cos[\Phi(i+1) - \Phi(1)] * \sin[\theta(i+1) - \theta(1)] * \sin[\Psi(i+1) - \Psi(1)] + \\ - \cos[\Psi(i+1) - \Psi(1)] * \sin[\Phi(i+1) - \Phi(1)];$$

$$S_1x = \sin[\Phi(i+1) - \Phi(1)] * \sin[\theta(i+1) - \theta(1)] * \sin[\Psi(i+1) - \Psi(1)] + \\ + \cos[\Phi(i+1) - \Phi(1)] * \cos[\Psi(i+1) - \Psi(1)];$$

$$S_1z = \cos[\theta(i+1) - \theta(1)] * \sin[\Psi(i+1) - \Psi(1)];$$

$$A_1x = \cos[\Phi(i+1) - \Phi(1)] * \sin[\theta(i+1) - \theta(1)] * \cos[\Psi(i+1) - \Psi(1)] + \\ + \sin[\Phi(i+1) - \Phi(1)] * \sin[\Psi(i+1) + \Psi(1)];$$

$$A_1y = \sin[\Phi(i+1) - \Phi(1)] * \sin[\theta(i+1) - \theta(1)] * \cos[\Psi(i+1) - \Psi(1)] + \\ + \sin[\Phi(i+1) + \Phi(1)] * \sin[\Psi(i+1) - \Psi(1)];$$

$$A_1z = \cos[\theta(i+1) - \theta(1)] * \cos[\Psi(i+1) - \Psi(1)];$$

[dx dy dz]: obtidos diretamente a partir das informações do palete

Obs.: os valores Φ(i+1), Φ(1), θ(i+1), θ(1), Ψ(i+1), Ψ(1) são todos valores obtidos diretamente através do robô. Observar que através dessa diferença, obtemos precisamente o posicionamento do palete em relação ao robô e possíveis diferenças na orientação da ferramenta no momento do toque no palete.

4.4.3. Obtenção de N2, S2, A2 e px, py, pz (parte dessas variáveis deverão ser estimadas e atualizadas a cada iteração)

$$[N_2, S_2, A_2] = \begin{bmatrix} N_0x & S_0x & A_0x \\ N_0y & S_0y & A_0y \\ N_0z & S_0z & A_0z \end{bmatrix} * \begin{bmatrix} N_1x & S_1x & A_1x \\ N_1y & S_1y & A_1y \\ N_1z & S_1z & A_1z \end{bmatrix}$$

$N_2x = N_0x * N_1x + S_0x * N_1y + A_0x * N_1z;$

$N_2y = N_0y * N_1x + S_0y * N_1y + A_0y * N_1z;$

$N_2z = N_0z * N_1x + S_0z * N_1y + A_0z * N_1z;$

$S_2x = N_0x * S_1x + S_0x * S_1y + A_0x * S_1z;$

$S_2y = N_0y * S_1x + S_0y * S_1y + A_0y * S_1z;$

$S_2z = N_0z * S_1x + S_0z * S_1y + A_0z * S_1z;$

$A_2x = N_0x * A1x + S_0x * A1y + A_0x * A_1z;$

$A_2y = N_0y * A1x + S_0y * A1y + A_0y * A_1z;$

$A_2z = N_0z * A1x + S_0z * A1y + A_0z * A_1z;$

Onde:

$N_0x = \cos[Phi(0)] * \cos[\theta(0)];$

$N_0y = \cos[\theta(0)] * \sin[Phi(0)];$

$N_0z = -\sin[\theta(0)];$

$$S_0x = \cos[\Phi(0)] * \sin[\theta(0)] * \sin[\Psi(0)] - \cos[\Psi(0)] * \sin[\Phi(0)];$$

$$S_0y = \sin[\Phi(0)] * \sin[\theta(0)] * \sin[\Psi(0)] + \cos[\Phi(0)] * \cos[\Psi(0)];$$

$$S_0z = \cos[\theta(0)] * \sin[\Psi(0)];$$

$$A_0x = \cos[\Phi(0)] * \sin[\theta(0)] * \cos[\Psi(0)] + \sin[\Phi(0)] * \sin[\Psi(0)];$$

$$A_0y = \sin[\Phi(0)] * \sin[\theta(0)] * \cos[\Psi(0)] - \cos[\Phi(0)] * \sin[\Psi(0)];$$

$$A_0z = \cos[\theta(0)] * \cos[\Psi(0)];$$

$$N_1x = \cos[\Phi(1)] * \cos[\theta(1)];$$

$$N_1y = \cos[\theta(1)] * \sin[\Phi(1)];$$

$$N_1z = -\sin[\theta(1)];$$

$$S_1x = \cos[\Phi(1)] * \sin[\theta(1)] * \sin[\Psi(1)] - \cos[\Psi(1)] * \sin[\Phi(1)];$$

$$S_1y = \sin[\Phi(1)] * \sin[\theta(1)] * \sin[\Psi(1)] + \cos[\Phi(1)] * \cos[\Psi(1)];$$

$$S_1z = \cos[\theta(1)] * \sin[\Psi(1)];$$

$$A_1x = \cos[\Phi(1)] * \sin[\theta(1)] * \cos[\Psi(1)] + \sin[\Phi(1)] * \sin[\Psi(1)];$$

$$A_1y = \sin[\Phi(1)] * \sin[\theta(1)] * \cos[\Psi(1)] - \cos[\Phi(1)] * \sin[\Psi(1)];$$

$$A_1z = \cos[\theta(1)] * \cos[\Psi(1)];$$

[px py pz]: (valores estimados a cada iteração)

Obs.: os valores Φ(0), θ(0), Ψ(0) são estimados a cada iteração, entretanto os valores Φ(1), θ(1), Ψ(1), são bem conhecidos (obtidos através do robô), entretanto, deverão ser sempre corrigidos a cada iteração (basta subtrair pelo novo zero do robô). Os valores px, py, pz distancia do referencial da base do robô até o ponto P0 deverá ser estimado a cada iteração.

4.5. Calculo do Jacobiano (J)

4.5.1. Equações

$$F_1 = [X(i+1) - (N_1x * dx + S_1x * dy + A_1x * dz)] =$$
$$= X(0) + (N_0x * N_1x + S_0x * N_1y + A_0x * N_1z) * px +$$
$$+ (N_0x * S_1x + S_0x * S_1y + A_0x * S_1z) * py +$$
$$+ (N_0x * A_1x + S_0x * A_1y + A_0x * A_1z) * pz$$

$$F_2 = [Y(i+1) - (N_1y * dx + S_1y * dy + A_1y * dz)] =$$
$$= Y(0) + (N_0y * N_1x + S_0y * N_1y + A_0y * N_1z) * px +$$
$$+ (N_0y * S_1x + S_0y * S_1y + A_0y * S_1z) * py +$$
$$+ (N_0y * A_1x + S_0y * A_1y + A_0y * A_1z) * pz$$

$$F_3 = [Z(i+1) - (N_1y * dx + S_1z * dy + A_1z * dz)] =$$
$$= Z(0) + (N_0z * N_1x + S_0z * N_1y + A_0z * N_1z) * px +$$
$$+ (N_0z * S_1x + S_0z * S_1y + A_0z * S_1z) * py +$$
$$+ (N_0z * A_1x + S_0z * A_1y + A_0z * A_1z) * pz$$

Onde:

$N_0x = \cos[\Phi(0)] * \cos[\theta(0)];$

$N_0y = \cos[\theta(0)] * \sin[\Phi(0)];$

$N_0z = -\sin[\theta(0)];$

$S_0x = \cos[\Phi(0)] * \sin[\theta(0)] * \sin[\Psi(0)] - \cos[\Psi(0)] * \sin[\Phi(0)];$

$$S_0y = \sin[\Phi(0)] * \sin[\theta(0)] * \sin[\Psi(0)] + \cos[\Phi(0)] * \cos[\Psi(0)];$$

$$S_0z = \cos[\theta(0)] * \sin[\Psi(0)];$$

$$A_0x = \cos[\Phi(0)] * \sin[\theta(0)] * \cos[\Psi(0)] + \sin[\Phi(0)] * \sin[\Psi(0)];$$

$$A_0y = \sin[\Phi(0)] * \sin[\theta(0)] * \cos[\Psi(0)] - \cos[\Phi(0)] * \sin[\Psi(0)];$$

$$A_0z = \cos[\theta(0)] * \cos[\Psi(0)];$$

$$N_1x = \cos[\Phi(1)] * \cos[\theta(1)];$$

$$N_1y = \cos[\theta(1)] * \sin[\Phi(1)];$$

$$N_1z = -\sin[\theta(1)];$$

$$S_1x = \cos[\Phi(1)] * \sin[\theta(1)] * \sin[\Psi(1)] - \cos[\Psi(1)] * \sin[\Phi(1)];$$

$$S_1y = \sin[\Phi(1)] * \sin[\theta(1)] * \sin[\Psi(1)] + \cos[\Phi(1)] * \cos[\Psi(1)];$$

$$S_1z = \cos[\theta(1)] * \sin[\Psi(1)];$$

$$A_1x = \cos[\Phi(1)] * \sin[\theta(1)] * \cos[\Psi(1)] + \sin[\Phi(1)] * \sin[\Psi(1)];$$

$$A_1y = \sin[\Phi(1)] * \sin[\theta(1)] * \cos[\Psi(1)] - \cos[\Phi(1)] * \sin[\Psi(1)];$$

$$A_1z = \cos[\theta(1)] * \cos[\Psi(1)];$$

4.5.2. Obtenção do Jacobiano

Para o calculo da matriz JACOBIANA, podemos observar que resolvendo o sistema (DIRETO)

$$|F|_{3x1} = J_{3x9} * \left|\vec{W}\right|_{9x1}$$

Onde: $|F|_{3x1} = [F_1 \quad F_2 \quad F_3]$

$\left|\vec{W}\right|_{9x1} = [x_0 \quad y_0 \quad z_0 \quad \Psi_0 \quad \theta_0 \quad \Phi_0 \quad p_x \quad p_y \quad p_z]$

J_{3x9} Matriz Jacobiana

Sendo perfeitamente conhecidos os valores para de_x, de_y, de_z e pontos de toque do robô (Pi, P(i+1).

4.5.3. Matriz Jacobiana (derivadas parciais)

$$\frac{dF_1}{dX(0)} = 1$$

$$\frac{dF_1}{dY(0)} = 0$$

$$\frac{dF_1}{dZ(0)} = 0$$

$$\frac{dF_1}{d\Psi(0)} = \left[\left(\frac{dN_0x}{d\Psi(0)}\right) * N_1x + \left(\frac{dS_0x}{d\Psi(0)}\right) * N_1y + \left(\frac{dA_0x}{d\Psi(0)}\right) * N_1z\right] * px + \left[\left(\frac{dN_0x}{d\Psi(0)}\right) * S_1x + \left(\frac{dS_0x}{d\Psi(0)}\right) * S_1y + \left(\frac{dA_0x}{d\Psi(0)}\right) * S_1z\right] * py + \left[\left(\frac{dN_0x}{d\Psi(0)}\right) * A_1x + \left(\frac{dS_0x}{d\Psi(0)}\right) * A_1y + \left(\frac{dA_0x}{d\Psi(0)}\right) * A_1z\right] * pz$$

$$\frac{dF_1}{d\theta(0)} = \left[\left(\frac{dN_0x}{d\theta(0)}\right) * N_1x + \left(\frac{dS_0x}{d\theta(0)}\right) * N_1y + \left(\frac{dA_0x}{d\theta(0)}\right) * N_1z\right] * px$$
$$+ \left[\left(\frac{dN_0x}{d\theta(0)}\right) * S_1x + \left(\frac{dS_0x}{d\theta(0)}\right) * S_1y + \left(\frac{dA_0x}{d\theta(0)}\right) * S_1z\right] * py$$
$$+ \left[\left(\frac{dN_0x}{d\theta(0)}\right) * A_1x + \left(\frac{dS_0x}{d\theta(0)}\right) * A_1y + \left(\frac{dA_0x}{d\theta(0)}\right) * A_1z\right] * pz$$

$$\frac{dF_1}{d\Phi(0)} = \left[\left(\frac{dN_0x}{d\Phi(0)}\right) * N_1x + \left(\frac{dS_0x}{d\Phi(0)}\right) * N_1y + \left(\frac{dA_0x}{d\Phi(0)}\right) * N_1z\right] * px$$
$$+ \left[\left(\frac{dN_0x}{d\Phi(0)}\right) * S_1x + \left(\frac{dS_0x}{d\Phi(0)}\right) * S_1y + \left(\frac{dA_0x}{d\Phi(0)}\right) * S_1z\right] * py$$
$$+ \left[\left(\frac{dN_0x}{d\Phi(0)}\right) * A_1x + \left(\frac{dS_0x}{d\Phi(0)}\right) * A_1y + \left(\frac{dA_0x}{d\Phi(0)}\right) * A_1z\right] * pz$$

$$\frac{dF_1}{dPx(0)} = N_0x * N_1x + S_0x * N_1y + A_0x * N_1z$$

$$\frac{dF_1}{dPy(0)} = N_0x * S_1x + S_0x * S_1y + A_0x * S_1z$$

$$\frac{dF_1}{dPz(0)} = N_0x * A_1x + S_0x * A_1y + A_0x * A_1z$$

$$\frac{dF_2}{dX(0)} = 0$$

$$\frac{dF_2}{dY(0)} = 1$$

$$\frac{dF_2}{dZ(0)} = 0$$

$$\frac{dF_2}{d\Psi(0)} = \left[\left(\frac{dN_0y}{d\Psi(0)}\right) * N_1x + \left(\frac{dS_0y}{d\Psi(0)}\right) * N_1y + \left(\frac{dA_0y}{d\Psi(0)}\right) * N_1z\right] * px$$
$$+ \left[\left(\frac{dN_0y}{d\Psi(0)}\right) * S_1x + \left(\frac{dS_0y}{d\Psi(0)}\right) * S_1y + \left(\frac{dA_0y}{d\Psi(0)}\right) * S_1z\right] * py$$
$$+ \left[\left(\frac{dN_0y}{d\Psi(0)}\right) * A_1x + \left(\frac{dS_0y}{d\Psi(0)}\right) * A_1y + \left(\frac{dA_0y}{d\Psi(0)}\right) * A_1z\right] * pz$$

$$\frac{dF_2}{d\theta(0)} = \left[\left(\frac{dN_0y}{d\theta(0)}\right) * N_1x + \left(\frac{dS_0y}{d\theta(0)}\right) * N_1y + \left(\frac{dA_0y}{d\theta(0)}\right) * N_1z\right] * px$$
$$+ \left[\left(\frac{dN_0y}{d\theta(0)}\right) * S_1x + \left(\frac{dS_0y}{d\theta(0)}\right) * S_1y + \left(\frac{dA_0y}{d\theta(0)}\right) * S_1z\right] * py$$
$$+ \left[\left(\frac{dN_0y}{d\theta(0)}\right) * A_1x + \left(\frac{dS_0y}{d\theta(0)}\right) * A_1y + \left(\frac{dA_0y}{d\theta(0)}\right) * A_1z\right] * pz$$

$$\frac{dF_2}{d\Phi(0)} = \left[\left(\frac{dN_0y}{d\Phi(0)}\right) * N_1x + \left(\frac{dS_0y}{d\Phi(0)}\right) * N_1y + \left(\frac{dA_0y}{d\Phi(0)}\right) * N_1z\right] * px$$
$$+ \left[\left(\frac{dN_0y}{d\Phi(0)}\right) * S_1x + \left(\frac{dS_0y}{d\Phi(0)}\right) * S_1y + \left(\frac{dA_0y}{d\Phi(0)}\right) * S_1z\right] * py$$
$$+ \left[\left(\frac{dN_0y}{d\Phi(0)}\right) * A_1x + \left(\frac{dS_0y}{d\Phi(0)}\right) * A_1y + \left(\frac{dA_0y}{d\Phi(0)}\right) * A_1z\right] * pz$$

$$\frac{dF_2}{dPx(0)} = N_0y * N_1x + S_0y * N_1y + A_0y * N_1z$$

$$\frac{dF_2}{dPy(0)} = N_0y * S_1x + S_0y * S_1y + A_0y * S_1z$$

$$\frac{dF_2}{dPz(0)} = N_0y * A_1x + S_0y * A_1y + A_0y * A_1z$$

$$\frac{dF_3}{dX(0)} = 0$$

$$\frac{dF_3}{dY(0)} = 0$$

$$\frac{dF_3}{dZ(0)} = 1$$

$$\frac{dF_3}{d\Psi(0)} = \left[\left(\frac{dN_0z}{d\Psi(0)}\right) * N_1x + \left(\frac{dS_0z}{d\Psi(0)}\right) * N_1y + \left(\frac{dA_0z}{d\Psi(0)}\right) * N_1z\right] * px$$
$$+ \left[\left(\frac{dN_0z}{d\Psi(0)}\right) * S_1x + \left(\frac{dS_0z}{d\Psi(0)}\right) * S_1y + \left(\frac{dA_0z}{d\Psi(0)}\right) * S_1z\right] * py$$
$$+ \left[\left(\frac{dN_0z}{d\Psi(0)}\right) * A_1x + \left(\frac{dS_0z}{d\Psi(0)}\right) * A_1y + \left(\frac{dA_0z}{d\Psi(0)}\right) * A_1z\right] * pz$$

$$\frac{dF_3}{d\theta(0)} = \left[\left(\frac{dN_0z}{d\theta(0)}\right) * N_1x + \left(\frac{dS_0z}{d\theta(0)}\right) * N_1y + \left(\frac{dA_0z}{d\theta(0)}\right) * N_1z\right] * px$$
$$+ \left[\left(\frac{dN_0z}{d\theta(0)}\right) * S_1x + \left(\frac{dS_0z}{d\theta(0)}\right) * S_1y + \left(\frac{dA_0z}{d\theta(0)}\right) * S_1z\right] * py$$
$$+ \left[\left(\frac{dN_0z}{d\theta(0)}\right) * A_1x + \left(\frac{dS_0z}{d\theta(0)}\right) * A_1y + \left(\frac{dA_0z}{d\theta(0)}\right) * A_1z\right] * pz$$

$$\frac{dF_3}{d\Phi(0)} = \left[\left(\frac{dN_0z}{d\Phi(0)}\right) * N_1x + \left(\frac{dS_0z}{d\Phi(0)}\right) * N_1y + \left(\frac{dA_0z}{d\Phi(0)}\right) * N_1z\right] * px$$
$$+ \left[\left(\frac{dN_0z}{d\Phi(0)}\right) * S_1x + \left(\frac{dS_0z}{d\Phi(0)}\right) * S_1y + \left(\frac{dA_0z}{d\Phi(0)}\right) * S_1z\right] * py$$
$$+ \left[\left(\frac{dN_0z}{d\Phi(0)}\right) * A_1x + \left(\frac{dS_0z}{d\Phi(0)}\right) * A_1y + \left(\frac{dA_0z}{d\Phi(0)}\right) * A_1z\right] * pz$$

$$\frac{dF_3}{dPx(0)} = N_0z * N_1x + S_0z * N_1y + A_0z * N_1z$$

$$\frac{dF_3}{dPy(0)} = N_0z * S_1x + S_0z * S_1y + A_0z * S_1z$$

$$\frac{dF_3}{dPz(0)} = N_0z * A_1x + S_0z * A_1y + A_0z * A_1z$$

Calculo das Parciais (N0, S0, A0)

$$\frac{dN_0x}{d\Psi(0)} = 0$$

$$\frac{dN_0x}{d\theta(0)} = -\cos[\Phi(0)] * \sin[\theta(0)]$$

$$\frac{dN_0x}{d\Phi(0)} = -\sin[\Phi(0)] * \cos[\theta(0)]$$

$$\frac{dN_0y}{d\Psi(0)} = 0$$

$$\frac{dN_0y}{d\theta(0)} = -\sin[\theta(0)] * \sin[\Phi(0)]$$

$$\frac{dN_0y}{d\Phi(0)} = \cos[\theta(0)]\cos[\Phi(0)]$$

$$\frac{dN_0z}{d\Psi(0)} = 0$$

$$\frac{dN_0z}{d\theta(0)} = \cos[\theta(0)]$$

$$\frac{dN_0z}{d\Phi(0)} = 0$$

$$\frac{dS_0x}{d\Psi(0)} = \cos[\Phi(0)] * \sin[\theta(0)] * \cos[\Psi(0)] + \sin[\Psi(0) * \sin[\Phi(0)]$$

$$\frac{dS_0x}{d\theta(0)} = \cos[\Phi(0)] * \cos[\theta(0)] * \sin[\Psi(0)]$$

$$\frac{dS_0x}{d\Phi(0)} = -\sin[\Phi(0)] * \sin[\theta(0)] * \sin[\Psi(0)] - \sin[\Psi(0)] * \cos[\Phi(0)]$$

$$\frac{dS_0y}{d\Psi(0)} = \sin[\Phi(0)] * \sin[\theta(0)] * \cos[\Psi(0)] - \cos[\Phi(0)] * \sin[\Psi(0)]$$

$$\frac{dS_0y}{d\theta(0)} = \sin[\Phi(0)] * \cos[\theta(0)] * \sin[\Psi(0)]$$

$$\frac{dS_0y}{d\Phi(0)} = \cos[\Phi(0)] * \sin[\theta(0)] * \sin[\Psi(0)] - \sin[\Phi(0)] * \cos[\Psi(0)]$$

$$\frac{dS_0z}{d\Psi(0)} = \cos[\theta(0)] * \cos[\Psi(0)]$$

$$\frac{dS_0z}{d\theta(0)} = -\sin[\theta(0)] * \sin[\Psi(0)]$$

$$\frac{dS_0z}{d\Phi(0)} = 0$$

$$\frac{dA_0x}{d\Psi(0)} = -\cos[\Phi(0)] * \sin[\theta(0)] * \sin[\Psi(0)] + \sin[\Phi(0)] * \cos[\Psi(0)]$$

$$\frac{dA_0x}{d\theta(0)} = \cos[\Phi(0)] * \cos[\theta(0)] * \cos[\Psi(0)]$$

$$\frac{dA_0x}{d\Phi(0)} = -\cos[\Phi(0)] * \sin[\theta(0)] * \sin[\Psi(0)] + \cos[\Psi(0)] * \sin[\Phi(0)]$$

$$\frac{dA_0y}{d\Psi(0)} = -\sin[\Phi(0)] * \sin[\theta(0)] * \sin[\Psi(0)] - \cos[\Psi(0)] * \sin[\Phi(0)]$$

$$\frac{dA_0y}{d\theta(0)} = \cos[\Phi(0)] * \sin[\theta(0)] * \cos[\Psi(0)] + \sin[\Phi(0)] * \sin[\Psi(0)]$$

$$\frac{dA_0z}{d\Psi(0)} = -\cos[\theta(0)] * \sin[\Psi(0)]$$

$$\frac{dA_0z}{d\theta(0)} = -\sin[\theta(0)] * \cos[\Psi(0)]$$

$$\frac{dA_0 z}{d\Phi(0)} = 0$$

4.5.4. Obtenção do Inverso da Matriz Jacobiana

O INVERNO da matriz Jacobiana permitira a obtenção dos parâmetros desconhecidos x_0, y_0, z_0, Ψ_0, θ_0, Φ_0, p_x, p_y e p_z), isto é:

$$\left|\vec{W}\right|_{9x1} = J^{+}_{3x9} * |F|_{3x1}$$

Onde

J^+ é a matriz pseudo-inversa.

e F terá que ser substituído por:

$$F_1 = [X(i+1) - (N_1 x * dx + S_1 x * dy + A_1 x * dz)]$$

$$F_2 = [Y(i+1) - (N_1 y * dx + S_1 y * dy + A_1 y * dz)]$$

$$F_3 = [Z(i+1) - (N_1 z * dx + S_1 z * dy + A_1 z * dz)]$$

Valores perfeitamente conhecidos

Observação (Claudio): Se você utilizar esses procedimentos notará que não é mais necessário o conhecimento aproximado de como esta o referencial do robô em relação ao palete de calibração.

No equacionamento do palete existem dois valores para N_1x, N_1y,...:

Um é dado por:

$$N_1 x = \cos[\Phi(1)] * \cos[\theta(1)]$$

$$N_1 y = \cos[\theta(1)] * \sin[\Phi(1)]$$

$$N_1 z = -\sin[\theta(1)]$$

$$S_1 x = \cos[\Phi(1)] * \sin[\theta(1)] * \sin[\Psi(1)] - cos[\Psi(1)] * sin[\Phi(1)]$$

$$S_1 y = \sin[\Phi(1)] * \sin[\theta(1)] * \sin[\Psi(1)] + \cos[\Phi(1)] * \cos[\Psi(1)]$$

$$S_1 z = \cos[\theta(1)] * \sin[\Psi(1)]$$

$$A_1 x = \cos[\Phi(1)] * \sin[\theta(1)] * \cos[\Psi(1)] + \sin[\Phi(1)] * \sin[\Psi(1)]$$

$$A_1 y = \sin[\Phi(1)] * \sin[\theta(1)] * \cos[\Psi(1)] - \cos[\Phi(1)] * \sin[\Psi(1)]$$

$$A_1 z = \cos[\theta(1)] * \cos[\Psi(1)]$$

E o outro por:

$$N_1 x = \cos[\Phi - \Phi(1)] * \cos[\theta - \theta(1)]$$

$$N_1 y = \cos[\theta - \theta(1)] * \sin[\Phi - \Phi(1)]$$

$$N_1 z = \sin[\theta - \theta(1)]$$

$$S_1 x = \cos[\Phi - \Phi(1)] * \sin[\theta - \theta(1)] * \sin[\Psi - \Psi(1)] - cos[\Psi - \Psi(1)] * sin[\Phi - \Phi(1)]$$

$$S_1 y = \sin[\Phi - \Phi(1)] * \sin[\theta - \theta(1)] * \sin[\Psi - \Psi(1)] + \cos[\Phi - \Phi(1)] * \cos[\Psi - \Psi(1)]$$

$$S_1 z = \cos[\theta - \theta(1)] * \sin[\Psi - \Psi(1)]$$

$$A_1 x = \cos[\Phi - \Phi(1)] * \sin[\theta - \theta(1)] * \cos[\Psi - \Psi(1)] + \sin[\Phi - \Phi(1)] * \sin[\Psi - \Psi(1)]$$

$$A_1 y = \sin[\Phi - \Phi(1)] * \sin[\theta - \theta(1)] * \cos[\Psi - \Psi(1)] - \cos[\Phi - \Phi(1)] * \sin[\Psi - \Psi(1)]$$

$$A_1 z = \cos[\theta - \theta(1)] * \cos[\Psi - \Psi(1)]$$